TÉLÉ-UNIVERSITÉ DU QUÉBEC À MONTRÉAL

OntoCASE:
MÉTHODOLOGIE ET ASSISTANT LOGICIEL POUR UNE INGÉNIERIE ONTOLOGIQUE FONDÉE SUR LA TRANSFORMATION D'UN MODÈLE SEMI-FORMEL

THÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN INFORMATIQUE COGNITIVE

PAR
MICHEL HÉON

MARS 2010

REMERCIEMENTS

Je désire offrir toute ma grattitude aux membres de ma famille. L'amour de ma vie, Élisabeth Camus, sans qui cette thèse n'existerait pas. Toujours présente, supportante, écoutante et encourageante, elle a su, dès les premiers moments, me soutenir, m'accompagner dans ce long parcours. Cette thèse, c'est aussi un peu la sienne. Dans ce cercle, je joins aussi mes deux fils, Frédérick et David, qui, par leur présence, ont donné un sens à cette démarche. Finalement, je désire offrir des remerciements pleins de tendresse à mon père Gustave et à ma mère Rita.

J'ai aussi beaucoup de reconnaissance pour mes deux directeurs de thèse, Gilbert Paquette et Josianne Basque. Merci pour leur patience, leur rigueur et leur soutien constant tout au long de mon cheminement.

Merci aussi à mes ami(e)s : Catherine Escojido ma relectrice de dernière instance, Pierre-François Hébert, mon copain d'écoute et de présence, et tous les autres… Thierry et Stéphanie, Kathleen Blair et Louis Dubeau, Gilles et Pascale, Jean-Claude, Pascale et Alain, Mathieu et Tanya, Albert et Nancy, Yves, Émilie, Bruno et Brigitte, Philippe et Catherine, Chantale et André, Carole Galland, Stéphanie Larrue, Annie et Henry, Sylvie et Gary, Soizig et Nicolas, Christa et Julien, Michel et Marilianne, André et Marie Camus et bien sûr, tous les autres, qui de près ou de loin ont fait partie de mon parcours.

Une pensée pour Alphonse Grypinich et Léo Jolie.

Mes derniers remerciements vont aux organismes subventionnaires suivants, qui ont appuyé financièrement cette recherche:

- Emploi-Québec en collaboration avec le centre local de développement (CLD) d'Argenteuil. Tout particulièrement, M. Stéphane Braney;

- Le fonds à l'accessibilité et à la réussite des études (Bourse FARE) de l'UQAM;
- La chaire de recherche du centre de recherche LICEF de la TELUQ;
- Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) qui finance le programme de recherche sur l'ingénierie ontologique et le Web Sémantique (PRIOWS);
- Le fonds québécois de recherche sur la nature et les technologies (FQRNT).

AVANT-PROPOS

L'idée de cette thèse a émergé en 2003. À ce moment, j'étais architecte et concepteur informatique pour Hydro-Québec. Dès cette époque, plusieurs signes de la problématique de la perte d'expertise en entreprise commençaient à se manifester, en raison notamment du départ à la retraite d'une couche importante d'employés de la génération des *Baby Boomers*. Il m'apparaissait évident que l'informatique avait un rôle à jouer dans la résolution de cette problématique et que le domaine des systèmes experts était une voie à envisager. Par la nature de mon métier, j'étais aussi fasciné par la représentation graphique de la réalité. L'idée de représenter des connaissances de manière schématique était quelque chose qui me semblait porteur. Ayant été mis au courant d'un projet de transfert d'expertise alors mené chez Hydro-Québec (Basque, Paquette, Pudelko et Léonard, 2004), j'entrepris de me documenter sur le projet et c'est ainsi que je découvris le langage de modélisation par objets typés (MOT). J'ai été charmé par sa simplicité d'utilisation et par la puissance de son expressivité. Parallèlement, les ontologies, l'*Ontology Web Language* (OWL) et ses applications, faisaient leur apparition dans l'univers du Web. Le double volet représentationnel et opérationnel d'une ontologie m'apparaissait comme une qualité importante à exploiter dans l'éventualité de la conception d'un système expert.

L'idée à l'origine du projet de thèse était de développer un système expert pour la conception de systèmes experts. Il s'agissait de construire une application informatique dont la simplicité d'utilisation permettrait à une personne experte d'un domaine de connaissances et possédant un minimum de connaissances en programmation informatique de concevoir un système expert. C'est dans ce contexte que je fis la connaissance du concepteur du langage MOT, Gilbert Paquette, ainsi que de Josianne Basque, tout deux chercheurs au Centre de recherche LICEF de la Télé-université, qui acceptèrent de me diriger pour la réalisation de cette thèse. Progressivement, au fil des discussions, notamment par la précision des besoins en conception d'ontologies par des non-initiés à la programmation informatique, le

projet s'est précisé pour se fixer sur la conception d'une méthodologie et d'un assistant logiciel pour la transformation de modèles semi-formels en ontologies.

La figure ci-dessous présente le nuage des mots-clés de la thèse, qui comme on peut le constater, met en évidence le domaine de la connaissance avec une multitude de thèmes s'y rapportant. Ce nuage de mots-clés offre un survol des éléments de contenu abordés dans la thèse.

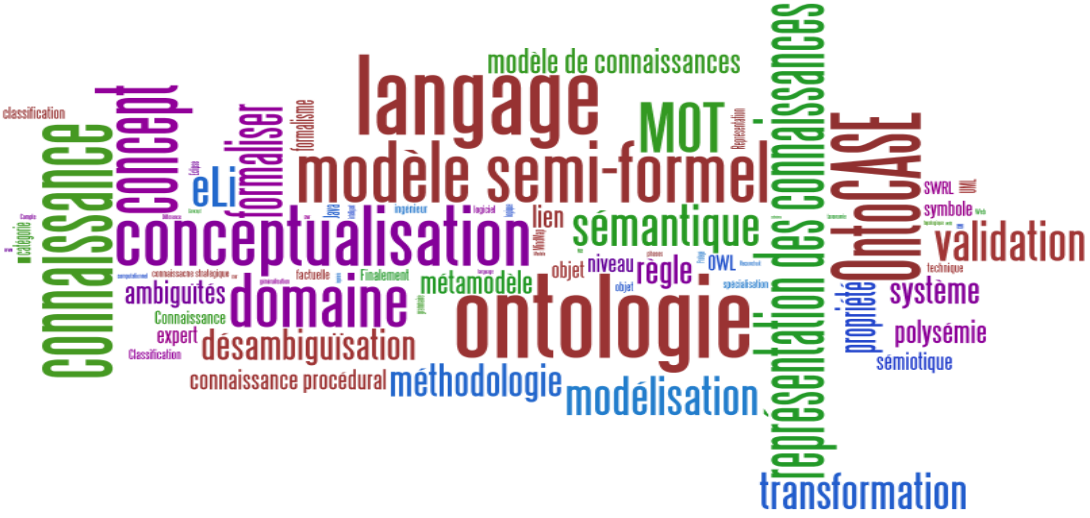


Figure 1 : Nuage de mots-clés de la thèse.

Pour faciliter la lecture, nous avons segmenté la thèse en deux volumes. Le premier comporte le développement de la thèse et le deuxième, ses appendices. À de multiples endroits dans la thèse, nous avons appuyé nos explications par des schémas en langage de modélisation par objets typés (MOT). Le lecteur, qui n'est pas familier avec ce langage, peut consulter l'appendice A pour en apprendre les rudiments d'utilisation.

Une version d'OntoCASE 1.0 sera éventuellement disponible sur le site www.cotechnoe.com. Les diagrammes MOT de la thèse ont été édités avec l'éditeur de modèles MOT eLi, qui est une composante d'OntoCASE. La préparation de la thèse aura permis une mise à l'essai intensive d'eLi en vue d'en faire un produit stable et robuste.

TABLE DES MATIÈRES

REMERCIEMENTS II

AVANT-PROPOS IV

TABLE DES MATIÈRES VI

LISTE DES FIGURES XVI

LISTE DES TABLEAUX XXI

RÉSUMÉ XXV

VOLUME I

INTRODUCTION 1

CHAPITRE 1 RECENSION DES ÉCRITS 8

1.1 Concept de représentation 8

1.1.1 Théorie des idées 9

1.1.2 Représentation dans un système d'information 10

1.1.3 Sémiotique 11

1.1.4 Abstraction et couche d'abstraction 13

1.1.5 Généralisation et abstraction 15

1.1.6 Modèle sémiotique de la synonymie et de la polysémie 16

1.1.7 Donnée, information et connaissance 18

1.1.8 Représentation des connaissances 19

1.1.9 Représentation des connaissances en sciences cognitives 20

1.2 Modélisation des connaissances 22

1.2.1 Langage 22
1.2.2 Modèle 23
1.2.3 Classification des langages de représentation des connaissances selon leur degré de formalisme 24
1.2.4 Ontologies dans la perspective des sciences cognitives 25
1.3 Classification des ontologies 29
1.3.1 Classification des ontologies selon la précision sémantique 30
1.3.2 Expressivité d'un système de représentation de degré formel 31
1.3.3 Classification selon le degré de généralité du domaine représenté 32
1.3.4 Ontologie et métadonnée 33
1.3.5 Ontologie cadre 34
1.4 Langages semi-formels de représentation de la connaissance 38
1.4.1 *MindMap* 39
1.4.2 Carte conceptuelle *(Concept map)* 40
1.4.3 *Thinking Maps* 41
1.4.4 Modèle orienté objet 41
1.4.5 Modélisation par objets typés 42
1.5 Langages formels de représentation de la connaissance 44
1.5.1 Taxonomie et le thesaurus 44
1.5.2 Langage de descriptions pour la représentation de connaissances 45
1.5.3 Graphe conceptuel 47
1.5.4 Langage ontologique 48
1.5.5 Langage à base de règles 51
1.5.6 Business Process Modeling Notation 54
1.6 Transformation de modèles par la technique de métamodélisation 55
1.6.1 Abstraction et couches d'abstraction 55
1.6.2 Espace de modélisation 57
1.6.3 Processus de transformation de modèles 58
1.6.4 Principe de transformation parallèle 60

1.6.5 Principe de la transformation orthogonale 61
1.6.6 Transformation d'un modèle semi-formel en ontologie 63
1.7 Ingénierie des connaissances 66
1.7.1 Structure de conception d'une méthodologie 66
1.7.2 Élicitation des connaissances 68
1.7.3 Méthodologies d'ingénierie ontologique 71
1.8 Environnements informatiques d'assistance à l'usager 76
1.9 En résumé 78

CHAPITRE 2 MÉTHODOLOGIE D'ONTOCASE 79
2.1 Hypothèse 79
2.2 Objectifs 80
2.3 Produits de la recherche 81
2.3.1 Volets d'OntoCASE 81
2.3.2 Ontologie de transformation d'OntoCASE 84
2.3.3 Portrait synthèse de la méthodologie d'OntoCASE 85
2.4 Ontologie du langage semi-formel 86
2.5 Ontologie de référence 86
2.5.1 Structure interne de l'ontologie de référence 87
2.5.2 Entités de l'ontologie de référence 88
2.5.3 Relations du modèle de référence 91
2.6 Ontologie cadre en représentation des connaissances d'OntoCASE 93
2.6.1 Ressources de l'ontologie cadre d'OntoCASE 95
2.6.2 Propriétés de l'ontologie cadre d'OntoCASE 96
2.6.3 L'ontologie cadre d'OntoCASE en OWL-N3 97
2.7 Méthodologie de conception d'une ontologie formelle à partir d'un modèle semi-formel 97
2.7.1 Méthode de conception d'un modèle semi-formel 99
2.7.2 Méthode de formalisation en ontologie du domaine 101

2.7.3 Méthode de validation de l'ontologie de domaine 109
2.8 En résumé 116

CHAPITRE 3 DÉMARCHE DE CONSTRUCTION D'ONTOCASE 118
3.1 Phase 1: Mise en place des composants représentationnels, procéduraux et informatique de la méthodologie. 122
3.1.1 Élaborer le processus de formalisation de l'ontologie 122
3.1.2 Choix du langage cible : l'*Ontology Web Language* 123
3.1.3 Choix du langage source: le langage de modélisation par objets typés MOT 124
3.1.4 Déterminer les catégories de l'ontologie de référence 130
3.1.5 Déterminer la structure de l'ontologie cadre 134
3.1.6 Représenter de manière formelle le vocabulaire de MOT 135
3.1.7 Choix de l'environnement informatique 146
3.1.8 Choix des outils informatiques associés à l'édition des ontologies 153
3.2 Phase 2: Agrégation des composants ontologiques, procéduraux et informatiques d'OntoCASE 155
3.2.1 Développement du module d'édition 158
3.2.2 Développement du module d'importation 160
3.2.3 Développement du module de traitement 162
3.2.4 Développement du module de désambiguïsation 163
3.2.5 Développement du module de conversion 168
3.2.6 Développement du module de validation 174
3.2.7 Développement des interfaces utilisateurs 177
3.3 Phase 3: Consolidation 180
3.3.1 Cas de test 180
3.3.2 Concevoir le banc d'essais 181
3.3.3 Concevoir et exécuter les cas de test 183

CHAPITRE 4 VALIDATION D'ONTOCASE 184
4.1 Modèle d'utilisabilité 185
4.2 Généralité des types de connaissances à formaliser et mesure de l'efficacité d'OntoCASE 187
4.2.1 Scénario de tests unitaires 187
4.2.2 Scénario de tests d'intégration 189
4.2.3 Scénario de tests systèmes 190
4.2.4 Scénario de tests du mécanisme de détection d'incohérence 192
4.3 Ergonomie d'OntoCASE 192
4.3.1 Mécanismes visant à assurer l'efficience d'OntoCASE et la satisfaction de l'usager 193
4.3.2 Mécanisme de rétroaction d'OntoCASE 193
4.3.3 Mécanisme de pilotage de l'ingénieur 194
4.3.4 Évaluation expérimentale de l'ergonomie d'OntoCASE 195
4.3.5 Déroulement du test d'utilisabilité 197
4.3.6 Modèle semi-formel d'expérimentation 198
4.3.7 Texte à modéliser en MOT et à formaliser 199
4.3.8 Bilan et commentaires 200
4.4 Généricité de la méthode de formalisation des systèmes semi-formels 202
4.4.1 Métamodèle de *MindMap* 202
4.4.2 Méthode d'intégration du langage *MindMap* à OntoCASE 205
4.4.3 Définition du langage semi-formel et production du scénario de test 206
4.4.4 Concevoir l'ontologie du langage semi-formel 207
4.4.5 Conception du module d'import/export 208
4.4.6 Adapter les ontologies 209
4.4.7 Exécuter le scénario de test 211
4.5 Conclusion 213

CHAPITRE 5 CONCLUSION ET DISCUSSION 215

5.1 Réalisations 215
5.2 Contributions 219
5.2.1 Apports en représentation des connaissances 219
5.2.2 Apports en ingénierie ontologique et aux applications du Web sémantique 221
5.2.3 Apports en gestion des connaissances 222
5.3 Limites de la recherche 223
5.3.1 Limites en représentation des connaissances 223
5.3.2 Limites informatiques du prototype 224
5.3.3 Limites du volet méthodologique d'OntoCASE 225
5.4 Perspectives 225
5.4.1 Perspectives de développement et d'utilisation 225
5.4.2 Perspectives en recherche sur la modélisation 226
5.4.3 Perspectives en représentation des connaissances 226
5.4.4 Perspectives en gestion des connaissances 227
5.4.5 Perspective de recherche pour les Environnements Informatiques pour l'Apprentissage Humain (EIAH) 228

VOLUME II

APPENDICE A GUIDE DU LANGAGE DE MODÉLISATION PAR OBJETS TYPÉS MOT 230
A.1 Structure du langage MOT 230
A.2 Alphabet du langage MOT 231
A.3 Types des connaissances en MOT 232
A.3.1 Alphabet de MOT associé aux types de connaissances 232
A.3.2 Sémantique de MOT associée aux types de connaissances 233
A.3.3 Stéréotype 234
A.4 Type de relations dans MOT 234
A.4.1 Alphabet des relations 234

A.4.2 Sémantique des relations 235
A.5 Sémantique des éléments grammaticaux du langage MOT 238
A.5.1 Composition 238
A.5.2 Spécialisation 239
A.5.3 Régulation 240
A.5.4 Instanciation 241
A.5.5 Intrant et le produit 242
A.5.6 Précédence 242
A.5.7 Lien d'application 243
A.5.8 Propriété et l'attribut 243
A.5.9 Règle 244
A.6 Résumé 245

APPENDICE B CATALOGUE DE LA SÉMANTIQUE FORMELLE DE MOT .. 246
B.1 Utilisation du catalogue 246
B.2 Entités atomiques d'OntoCASE 247
B.3 Spécialisation et l'instance 251
B.3.1 Spécialisation entre deux concepts, deux procédures et deux principes ... 251
B.3.2 Instanciation entre une abstraction et son observable 253
B.4 Intrant et le produit 255
B.4.1 Intrant-produit entre connaissances d'action et d'objet de niveau abstrait 255
B.4.2 Intrant-produit entre des connaissances d'action et d'objet observables .. 257
B.5 Régulation 258
B.5.1 Régulation entre connaissances abstraites 258
B.5.2 Interprétation de la régulation entre un énoncé et des observables d'objets, de procédures et de principes. 260
B.6 Propriété 262
B.6.1 Définition d'un domaine et d'une image à une propriété entre des objets 262

B.6.2 Propriétés unaires....264
B.6.3 Exemple de représentation en langage MOT....264
B.7 Composition entre connaissances....265
B.7.1 Holonyme entre deux abstractions....265
B.7.2 Composition entre deux connaissances observables....267
B.7.3 Composition entre des connaissances de niveau d'abstraction différent..269
B.8 Attribut de connaissances déclaratives....271
B.9 Lien de précédence....274
B.9.1 Relation de précédence entre connaissances d'actions....274
B.9.2 Précédence entre connaissances d'actions et stratégique....275
B.10 Règle....278
B.10.1 Règle à partir de connaissances abstraites....278
B.10.2 Règle à partir de connaissances factuelles....281
B.11 Connaissance qui englobe des connaissances....284

APPENDICE C CATALOGUE DES COHÉRENCES DE L'ONTOLOGIE CADRE D'ONTOCASE....287
C.1 Axiomes d'identification des erreurs de cohérences....287
C.2 Règles de génération des erreurs....293

APPENDICE D SCÉNARIOS DE CAS DE TESTS....295
D.1 Scénario de tests unitaires....295
D.2 Scénario de tests fonctionnels....299
D.3 Scénario de tests systémiques....303
D.3.1 Scénarios de tests systémiques issus de l'expérimentation....304
D.3.2 Scénarios de tests systémiques issus d'ouvrage en modélisation....306
D.4 Scénarios de tests d'incohérences....313

APPENDICE E ÉLÉMENTS POUR GUIDER LA MODÉLISATION SEMI-FORMELLE À L'AIDE DE CONCEPTS ONTOLOGIQUES....317

E.1 Métamodèle MOT d'OWL 317
E.1.1 Ressource 318
E.1.2 Concept de classe 319
E.1.3 Propriété 320
E.1.4 Relations 321
E.2 La notation N3 322
E.3 Heuristiques de modélisation 323
E.3.1 Polysémie 323
E.3.2 Modéliser selon le bon niveau d'abstraction 324
E.3.3 Représentation d'un attribut 328
E.3.4 Désambiguïser la synonymie d'agrégation 329
E.3.5 Résumé 331

APPENDICE F CODE SOURCE 332
F.1 Base de règles de la désambiguïsation typologique 332
F.2 Base de règles de la désambiguïsation topologique 343
F.3 Base de règles de transformation: état initial 346
F.4 Base de règles de transformation: état création 346
F.5 Base de règles de transformation: état classification 351
F.6 Base de règles de transformation: état final 359
F.7 Base de règles de transformation: état validation 359
F.8 Ontologie du langage semi-formel MOT 359
F.9 Ontologie du traitement des ambiguïtés 362
F.10 Ontologie de référence 364
F.11 Ontologie cadre en représentation des connaissances d'OntoCASE 367
F.12 Ontologie de transformation 369
F.13 Bibliothèque de codes *Java* nécessaire à l'implantation du modèle de conception *commande* 383
F.13.1 Structure de la *built-in command* swrlbi.owl 383

F.13.2 Implantation de SWRLBuiltInLibraryImpl.java 384
F.13.3 Interface Command.java 385
F.13.4 Interface Receiver.java 385
F.13.5 Interface Invoker.java 385
F.13.6 Implantation InvokerImpl.java 385
F.14 Quelques implantations de *Receiver* 385
F.14.1 Implantation de CreerUneOntologieCmdReceiver.java 385
F.14.2 Implantation de OWLCreerUneClasseCmdReceiver.java 386
F.14.3 Implantation de OWLAjoutDuneProprieteEntreResourceCmdReceiver.java 386

APPENDICE G ÉVALUATION EXPÉRIMENTALE 388
G.1 Protocole d'expérimentation 389
G.2 Certificat d'éthique 392
G.3 Réponse commentée des participants 393
G.3.1 Réponse commentée du participant 1 (d'après son verbatim) 393
G.3.2 Réponse commentée du participant 2 (d'après son verbatim) 396
G.3.3 Réponse commentée du participant 3 (d'après le verbatim) 397
G.3.4 Réponse commentée du participant 4 (d'après le verbatim) 399

BIBLIOGRAPHIE 404

LISTE DES FIGURES

Figure 1.1: L'objet et son abstraction: l'idée 9
Figure 1.2: Le modèle de la représentation d'un domaine dans un système d'information. (Tirée et adaptée d'Olivé, 2007 p. 46) 10
Figure 1.3: Tableau *La trahison des images*, Magritte 1929. (Tirée de Paquet, 2006, p. 9) 11
Figure 1.4: Le triangle sémiotique. 12
Figure 1.5: Exemple d'un triangle sémiotique associé à la conceptualisation du chien Bahia. 12
Figure 1.6: Les deux mystères. (Magritte 1966) 14
Figure 1.7: Le concept d'abstraction représenté à partir du triangle sémiotique. 15
Figure 1.8 : Le modèle sémiotique représenté en MOT 16
Figure 1.9 : Exemple de synonymie entre *4* et *quatre.* 17
Figure 1.10 : Exemple de polysémie du mot *feu*. 17
Figure 1.11: Donnée, information et connaissances dans le modèle sémiotique. 18
Figure 1.12: Le métaconcept de représentation des connaissances. 19
Figure 1.13: Modèle cognitif de la représentation des connaissances. (Tirée de Helbig, 2006, p. 8) 20
Figure 1.14: Langage de modélisation. 23
Figure 1.15: Positionnement constructiviste de la définition d'ontologie. 26
Figure 1.16: Définition d'une ontologie en informatique. 29
Figure 1.17: Échelle de classification d'ontologies en fonction du spectre sémantique de McGuiness. (Tirée de Breitman *et al.*, 2007, p. 26) 31
Figure 1.18: Taxonomie des classes de haut niveau dans SUMO-OWL. 35
Figure 1.19: Taxonomie de patrons de conception ontologique du projet NeOn. (Tirée de ODP.org, 2008) 37
Figure 1.20: Exemple de schéma *MindMap*. (Tirée de Okada *et al,* 2008, p. xi) 39
Figure 1.21: Exemple de modèle représenté dans le formalisme *Concept map.* (Tirée d'Okada *et al,* 2008, p.xi) 40
Figure 1.22: Synthèse du langage des *Thinking Maps.* (Tirée de Hyerle 2008, p. 50) 41

Figure 1.23: Diagramme de classes UML pour représenter un arbre de décision. (Tirée de Rhem 2006, p. 181) 42

Figure 1.24: Métamodèle du langage MOT présenté dans le langage MOT. (Tirée de Paquette 2002, p. 73) 43

Figure 1.25: Illustration d'un graphe conceptuel tel que proposé par Sowa. (Tirée de Chein et Mugnier, 2008, p. 26.) 47

Figure 1.26: Évolution des langages ontologiques. (Tirée de Calero *et al*., 2006, p. 37) 49

Figure 1.27: Métamodèle UML du *Semantic Web Rule Language.* (Tirée de Brockmans *et al.*, 2006,p. 9). 53

Figure 1.28: Éléments d'un modèle BPMN. (Tirée de Weske, 2007, p. 209)..... 54

Figure 1.29: Exemple de réseau de Pétri pour représenter un *workflow.* (Tirée de Weske, 2007, p. 271)..... 55

Figure 1.30: Couches d'abstraction de la métamodélisation. 56

Figure 1.31: Espaces de modélisation EBNF, RDFS et MOF adaptée de Gašević *et al.* (2006e p. 132). 57

Figure 1.32: Processus de transformation de modèles dans l'OMG-MDA. 59

Figure 1.33: Processus de transformation parallèle. 60

Figure 1.34: Processus de transformation orthogonale. 61

Figure 1.35: Modèle du processus de transformation orthogonale. 62

Figure 1.36 : La modélisation d'ontologie dans le contexte de l'ACM et du Web sémantique. (Tirée de Gašević *et al.*, 2006c, p. 175) 64

Figure 1.37: Modèle MOT de la définition d'une méthodologie. (Tirée et adaptée en MOT de Gómez-Pérez *et al.*, 2003, p. 109) 68

Figure 1.38: Le cycle des connaissances dans MASK. (Tirée d'Ermine et Matta, 2003, p. 8) 69

Figure 1.39: Processus de développement d'une ontologie. (Tirée de Gómez-Pérez *et al.*2003, p. 110)..... 71

Figure 1.40: Processus de développement d'une ontologie *On-To-Knowledge.* (Tirée de Staab *et al.* 2001) 72

Figure 1.41: Cycle de vie du processus de développement d'ontologie de METHONTOLOGY. (Tirée de Gómez-Pérez *et al.*, 2003, p.127)..... 73

Figure 2.1: Positionnement d'OntoCASE en fonction des degrés de formalisation et classification d'exemples de formalisme. 80

Figure 2.2: Volets méthodologique, computationnel et représentationnel d'OntoCASE. 83
Figure 2.3: Architecture de l'ontologie de transformation. 84
Figure 2.4: Principe de transformation guidée par l'ontologie de référence. 87
Figure 2.5: Ontologie cadre de la biologie. 94
Figure 2.6: Exemple de classification avec L'ontologie cadre d'OntoCASE. 95
Figure 2.7: Modèle de la méthodologie de conception d'une ontologie à partir d'un modèle semi-formel. 98
Figure 2.8: Méthode de conception d'un modèle semi-formel. 99
Figure 2.9: Méthode de formalisation d'un modèle semi-formel en ontologie du domaine. 101
Figure 2.10: Méthode de validation de l'ontologie du domaine. 110
Figure 2.11: Processus de validation syntaxique. 111
Figure 2.12: Processus de validation sémantique. 112
Figure 3.1: Objets et activités de la démarche de construction d'OntoCASE. 121
Figure 3.2: Représentation taxonomique des classes et propriétés de l'ontologie cadre. 135
Figure 3.3: Modèle procédural EMF de production de code source *Java* à partir d'une conceptualisation. 148
Figure 3.4: Modèle *ecore* de la structure de donnée d'eLi pour la modélisation en langage MOT. 150
Figure 3.5: Processus de construction d'un éditeur graphique GMF. (Tirée du GMF Tutorial 2009d) 152
Figure 3.6: Les composants de l'application OntoCASE. 156
Figure 3.7: Relations entre les modules de l'application et les méthodes d'OntoCASE. 157
Figure 3.8: Interface utilisateur de l'application OntoCASE. 158
Figure 3.9: Processus de conception du programme *Eli2OWL.java.* 161
Figure 3.10: Modèle du processus de conception du module de désambiguïsation. 163
Figure 3.11: Développer le module de conversion de l'ontologie du modèle semi-formel désambiguïsé en ontologie du domaine. 168
Figure 3.12: Relations et utilisations d'un invocateur, d'une commande et d'un receveur. 169

Figure 3.13: Sous-processus et ontologies impliqués dans la création de l'ontologie du domaine. 172
Figure 3.14: Modèle procédural du développement du module de validation. 175
Figure 3.15: Interfaces de communication avec l'utilisateur. 177
Figure 3.16: Le tableau de bord à la formalisation. 179
Figure 3.17: Le tableau de bord à la validation. 180
Figure 3.18: Exécution d'un cas de test pour la mesure d'efficacité d'OntoCASE. . 182
Figure 4.1: Modèle semi-formel d'expérimentation: l'*Objet céleste.* 199
Figure 4.2: Texte présenté lors de la phase 3 de l'expérimentation. 200
Figure 4.3: Métamodèle *ecore* du langage *MindMap. (Tirée de Reitsma et al., 2008).* 204
Figure 4.4: Méthode d'intégration d'un nouveau langage source à OntoCASE. 205
Figure 4.5: Modèles *MindMap* sur le thème de l'automobile et le thème de la rédaction d'une thèse. 207
Figure 4.6: Structure de l'ontologie de transformation après l'intégration des éléments structurels nécessaires à la transformation d'un modèle *MindMap.* 210
Figure 4.7: Restauration dans le formalisme de MOT du thème de l'automobile (a) et de la rédaction d'une thèse (b). 213
Figure A.1 : Structure générale d'un langage. 230
Figure A.2 : Structure de l'alphabet de MOT. 231
Figure A.3: Représentation des connaissances en langage MOT. 233
Figure A.4: Représentation de la procédure P dont le stéréotype une méthode. 234
Figure A.5: Hiérarchie des relations typées utilisées en langage MOT et leur représentation dans l'ontologie du langage semi-formel. 235
Figure B.1:La représentation en UML des différentes valeurs possibles des entités d'OntoCASE. 248
Figure B.2 : Exemple de définition des entités atomiques de MOT et leur stéréotype de désambiguïsation possible. 248
Figure E.1: Hiérarchie des concepts de haut niveau d'OWL selon ODM. 318
Figure E.2: Hiérarchie des classes d'OWL. 319
Figure E.3: Hiérarchie des propriétés d'OWL. 320
Figure E.4: Relations unissant les divers concepts d'OWL. 321

Figure E.5: Représentation en OWL-N3 de l'énoncé: *la pomme, qui est un Fruit, est de couleur rouge.* 322

Figure G.1 : Structure de chaque réponse commentée..... 393

LISTE DES TABLEAUX

Tableau 1.1 Expressivité de divers systèmes de représentation formels 31
Tableau 1.2 Éléments du Dublin Core (tirée de Breitman *et al.*, 2007, p. 178)..... 34
Tableau 1.3 L'ontologie de KR (Tirée de Sowa, 2003) 36
Tableau 1.4 Exemple d'une base de connaissances en DL (inspiré de Baader *et al.*, 2004) 46
Tableau 1.5 Constructeurs de la DL et leur correspondance langagière (tirée de Gómez-Pérez *et al.*, 2003, p. 17)..... 46
Tableau 1.6 Sommaire des caractéristiques des langages ontologiques (adapté de Gómez-Pérez *et al.*, 2003, p. 286)..... 50
Tableau 1.7 Sommaire comparatif du processus de développement d'ontologies (extrait de Gómez-Pérez *et al.*, 2003, p.151) 75
Tableau 2.1 Méthodes, techniques et outils de la méthodologie OntoCASE 85
Tableau 2.2 Structure de classes de l'ontologie du langage semi-formel 86
Tableau 2.3 Catégorie des entités du modèle de référence 89
Tableau 2.4 Catégorie des entités et leur couleur correspondante 89
Tableau 2.5 Exemple de représentation d'une règle em MOT 91
Tableau 2.6 Catégorie des relations du modèle de référence..... 92
Tableau 2.7 Ressources de l'ontologie cadre d'OntoCASE..... 96
Tableau 2.8 Propriétés de l'ontologie cadre d'OntoCASE..... 97
Tableau 2.9 Études de cas représentés en langage MOT et en langage OWL-N3... 105
Tableau 2.10 Modèle semi-formel formalisé en OWL-N3 108
Tableau 2.11 L'Objet céleste: le modèle semi-formel et son interprétation possible en langage naturel 113
Tableau 2.12 Bilan du rapport de validation syntaxique du modèle L*'Objet Céleste* 114
Tableau 2.13 Rapport de validation sémantique généré par OntoCASE..... 115
Tableau 3.1 Représenter une agrégation en MOT 125
Tableau 3.2 Représenter l'utilisation d'un instrument en MOT 126
Tableau 3.3 Représenter selon le niveau d'abstraction en MOT..... 127
Tableau 3.4 Interprétation formelle de la règle brûler des déchets..... 128

Tableau 3.5 Représenter un attribut en MOT 129
Tableau 3.6 Polysémie de MOT 130
Tableau 3.7 Vocabulaire du langage MOT 131
Tableau 3.8 Vocabulaire du diagramme UML de classes (tiré de Rhem, 2006) 132
Tableau 3.9 Vocabulaire du diagramme UML de cas d'utilisation (tiré de Rhem, 2006) 132
Tableau 3.10 Vocabulaire du diagramme UML d'état (tiré de Rhem, 2006) 132
Tableau 3.11 Vocabulaire du diagramme BPMN (tiré de OMG BPDM, 2007 ; Weske, 2007) 132
Tableau 3.12 Classification des entités des divers langages pour chacune des catégories d'entités de l'ontologie de référence 133
Tableau 3.13 Classification des relations de divers langages pour chacune des catégories de langage de l'ontologie de référence 134
Tableau 3.14 Correspondance formelle de représentations MOT des connaissances déclaratives 137
Tableau 3.15 Correspondance formelle de représentations MOT des connaissances procédurales 139
Tableau 3.16 Correspondance formelle de représentations MOT des connaissances stratégiques 140
Tableau 3.17 Correspondance formelle de représentations MOT des connaissances mixtes 141
Tableau 3.18 Correspondance formelle de représentations MOT des connaissances mixtes associées par un lien de régulation 142
Tableau 3.19 Correspondance formelle de représentations MOT des connaissances mixtes pour la formation d'une règle produisant une conclusion 144
Tableau 3.20 Correspondance formelle de représentations MOT des connaissances mixtes pour la formation d'une règle résultant de l'exécution d'une opération 145
Tableau 3.21 Représentation MOT, XMI et OWL-N3 de *Bahia est un Berger Allemand qui est une sorte de Chien.* 159
Tableau 3.22 Ontologie cadre (OWL) du vocabulaire de MOT [a) et b)], et exemple d'importation en OWL-N3 d'un modèle *MOT-XMI* [c)] 162
Tableau 3.23 Structure de l'ontologie de traitement des ambiguïtés 165

Tableau 3.24 Règle de désambiguïsation topologique dans le cas d'un principe lié par lien R d'un concept (domaine) à un autre (codomaine) 166
Tableau 3.25 Exemple d'implantation d'une commande Java appelée lors du déclenchement d'une règle SWRL 171
Tableau 3.26 Axiomes et règles impliqués dans la conversion d'une spécialisation entre deux concepts 173
Tableau 3.27 Exemple de rapport complet de validation syntaxique 176
Tableau 4.1 Structure de présentation des divers cas de test unitaires..... 187
Tableau 4.2 Quelques cas de test unitaire 188
Tableau 4.3 Quelques cas de tests d'intégration..... 189
Tableau 4.4 Quelques cas de tests systèmes 191
Tableau 4.5 Icônes de rétroaction des ambiguïtés et des erreurs d'incohérence 194
Tableau 4.6 Ontologie du langage *MindMap* dans le formalisme OWL-N3 207
Tableau 4.7 Représentation *XMI* (a) et *OWL* (b) du modèle *MindMap* sur le thème de l'automobile 209
Tableau 4.8 Règles de conversion pour un hyperonyme 211
Tableau 4.9 Rapport de validation sémantique 212
Tableau A.1 Type de connaissances dans le langage MOT et leur symbole associé. 234
Tableau A.2 Sémantique des relations typées dans MOT (adapté de Paquette 2002b, 2010) 236
Tableau A.3 Grammaire des relations MOT (adapté de Paquette 2002b) 237
Tableau A.4 Exemple de représentation d'une composition entre connaissances ... 238
Tableau A.5 Exemple de représentation de la spécialisation de connaissances 239
Tableau A.6 Exemple de représentation d'une régulation entre des connaissances. 240
Tableau A.7 Exemple de représentation de l'instanciation entre une connaissance abstraite et un connaissance factuelle 241
Tableau A.8 Exemple représentation d'un intrant et d'un produit entre une connaissance procédurale et une connaissance conceptuelle..... 242
Tableau A.9 Exemple de représentation de la préséance..... 242
Tableau A.10 Exemple de représentation d'un lien d'application 243
Tableau A.11 Exemple de représentation d'un attribut et d'une propriété 244
Tableau A.12 Exemple de représentation d'une règle..... 245

Tableau B.1 Structure de chaque élément du catalogue 246
Tableau B.2: Tableau des règles concernant chaque relation légale en MOT combiné au numéro de page de l'interprétation..... 247
Tableau C.1 Axiomes d'identification des incohérences 288
Tableau C.2 Règles de génération des erreurs 294
Tableau D.1 nom du test, modèle d'origine et bilan du scénario de test unitaire..... 296
Tableau D.2 nom du test, modèle d'origine et bilan du scénario de test fonctionnel..... 300
Tableau D.3 Scénario de test systémique: *Le système solaire* 304
Tableau D.4 Scénario de test systémique: *La gestion des déchets* 305
Tableau D.5 Scénarios de test d'incohérences 314
Tableau E.1 Polysémie de MOT 324
Tableau E.2 Représenter selon le niveau d'abstraction en MOT 326
Tableau E.3 Interprétation formelle de la règle d'incinération des déchets 327
Tableau E.4 Représenter un attribut en MOT 328
Tableau E.5 Représenter une agrégation en MOT 329
Tableau E.6 Interprétation formelle des modèles semi-formels du tableau e.5 330
Tableau E.7 Test de complétude pour les modèles du tableau e.5..... 331

RÉSUMÉ

Concevoir une ontologie formelle demande une expertise certaine, qui est la plupart du temps, peu accessible à des experts de contenu. En revanche, de plus en plus d'experts utilisent la modélisation semi-formelle pour représenter leur expertise, car ce type de langage est notamment reconnu pour sa simplicité d'utilisation et sa capacité à représenter des connaissances de types déclaratifs, procédurales, stratégiques et factuels. La modélisation semi-formelle, qui peut constituer une première démarche dans la mise en place d'une mémoire d'entreprise, n'élimine en rien la nécessité de représenter formellement la connaissance, obligeant ainsi à mettre en oeuvre une étape de formalisation du modèle semi-formel. Nous avons conçu une méthodologie de transformation d'un modèle semi-formel en ontologie et développé un assistant logiciel qui semi-automatise, ou automatise les processus de la méthodologie de transformation. La démarche de conception de la méthodologie et de son assistant informatique se divise en trois phases: 1) la phase de *Mise en place des composants architecturaux, procéduraux et informatiques de la méthodologie* est la phase initiale de la démarche*;* 2) la phase d'*Agrégation des composants ontologiques, procéduraux et informatiques* est la phase de développement et d'harmonisation des modules de l'assistant aux processus de la méthodologie; 3) la phase de *confirmation* est l'étape de tests et de raffinement de la fonctionnalité, de l'assistant informatique et de la méthodologie. Quatre champs disciplinaires sont concernés par cette thèse: en gestion des connaissances, notre approche offre une méthode de formalisation de la connaissance fondée sur une représentation semi-formelle de la connaissance; en ingénierie ontologique, nos travaux offrent un cadre architectural et procédural qui formalise et instrumente le processus de construction d'une ontologie à partir d'une représentation semi-formelle des connaissances; en représentation des connaissances, notre thèse approfondit l'étude d'une catégorisation formelle de la représentation des connaissances qu'elles soient déclaratives, procédurales, stratégiques ou factuelles; et finalement, d'un point de vue informatique, cette recherche présente une architecture

et des outils informatiques qui formalisent et rendent exécutable le processus de transformation.

Mots-clés: Ingénierie ontologique, système expert à la formalisation, représentation de connaissances, architecture conduite par les modèles, modélisation semi-formelle, modélisation d'ontologie, transformation d'un modèle semi-formel en ontologie.

INTRODUCTION

Dans le contexte actuel du vieillissement de la population, les organisations sont de plus en plus confrontées à une problématique de rétention de leur culture d'entreprise et des savoirs spécifiques à leur domaine d'intervention. Que ce soit sur le plan des processus organisationnels ou des connaissances liées au domaine d'activité, les départs massifs à la retraite des employés risquent d'handicaper fortement les organisations. On n'a qu'à penser à la problématique actuelle du système de santé dont l'une des composantes identifiées et reconnues a été la mise à la retraite massive des infirmières et des médecins dans les années 1990. Même quand le départ des employés n'est pas forcé, comme ce fut le cas pour les infirmières, le problème de perte d'expertise risque fort de mettre en péril, dans les années à venir, l'efficacité des organisations selon le *comité de travail sur l'intégration des jeunes à la fonction publique québécoise* (2001). La mise en place de stratégies de « gestion de connaissances » (*Knowledge Management*) s'impose donc rapidement puisque la qualité des services offerts par les organisations en dépend.

De surcroît, le contexte économique des années 1980-1990 n'a pas favorisé l'embauche des jeunes de cette époque. Pour les organisations, cette situation restreint l'accès à une relève pouvant assurer la pérennité d'un savoir de terrain durement acquis. Elle force la mise en place de politiques d'embauche massive de jeunes. Par exemple, le *Comité de travail sur l'intégration des jeunes à la fonction publique québécoise* (2001) évalue le pourcentage des moins de 35 ans à 6,9% en 2001, à 13,4% en 2006 et à 20,3% en 2011. Cet état de fait impose aux organisations la mise en place de politiques et stratégies visant à assurer la formation efficiente de la relève.

Selon la proposition d'Apostolou, Mentzas, Young et Abecker (2000), la gestion des connaissances en entreprise peut se réaliser selon deux approches, soit une approche centrée **processus** ou une approche centrée **produit**. L'approche centrée processus valorise les échanges directs entre les personnes expertes et les personnes apprenantes

pour l'échange de connaissances. Le processus de communication sociale est au centre de celle-ci. Quant à l'approche centrée produit, qui nous intéresse plus particulièrement dans cette thèse, elle valorise la production de documents et l'entreposage des connaissances organisationnelles dans des systèmes informatiques. Elle vise à constituer la « mémoire » de l'organisation. Le besoin de rendre les connaissances le plus accessibles possible à l'ensemble des employés impose l'utilisation d'un système de codification de l'information pour maximiser cette accessibilité. La codification doit se faire au niveau sémantique, ce que l'ingénierie ontologique, appuyée par la technologie du Web sémantique, permet de faire.

Cette thèse se situe à l'intersection de plusieurs domaines: la gestion des connaissances, l'élicitation des connaissances, la représentation des connaissances, l'ingénierie ontologique et l'assistance automatisée à la conception d'artéfacts logiciels. Ces domaines doivent être considérés dans l'utilisation du Web pour traiter la sémantique des connaissances, au-delà de la syntaxe des termes qui figurent dans les pages ou les documents diffusés sur le Web.

L'avènement du Web sémantique procure aux gestionnaires de la connaissance un langage standardisé qui permet de représenter les connaissances sous la forme d'un modèle que l'on nomme *ontologie*, ainsi qu'un ensemble d'outils qui permet de manipuler les connaissances représentées dans l'ontologie. Une ontologie utilise un langage formel, c'est-à-dire utilisable par un logiciel, qui se présente sous la forme d'un fichier définissant les concepts, les propriétés, les axiomes et les faits concernant un domaine.

Concevoir une ontologie est une activité complexe, qui nécessite notamment la participation de ceux qui détiennent l'expertise dans l'organisation. On pourrait croire, en pensant économiser temps et énergie, qu'il est préférable d'éliciter les connaissances de ces experts directement dans une représentation de type ontologique. Nous pensons au contraire que le processus de conception d'une

ontologie gagne à être décomposé en au moins deux étapes bien distinctes: une étape d'élicitation des connaissances dans un formalisme semi-formel relativement évolué qui optimise l'expressivité tout en structurant la représentation explicitée à un premier niveau de formalisme, puis une étape de formalisation plus poussée des connaissances, où le modèle semi-formel est transformé dans un formalisme ontologique. Bien qu'un modèle semi-formel contienne toujours des éléments d'ambigüité, sa souplesse d'expression, surtout lorsqu'il fait appel à un format graphique, permet d'accéder plus facilement à l'identification des connaissances tacites des experts. Dans un tel cadre, la spontanéité n'est pas bloquée par une charge cognitive trop lourde associée à une formalisation poussée de la pensée (Basque *et al.*, 2008b). Nous pensons que l'usage d'un système de représentation plus convivial que les systèmes de représentation ontologique, tels ceux utilisés dans les éditeurs d'ontologies Protégé[1] ou TopBraid[2], pourrait permettre: (1) d'élargir le bassin des personnes aptes à contribuer activement aux premières étapes de la représentation de leurs connaissances, et ce, sans l'aide d'un ingénieur des connaissances; (2) d'économiser du temps lors de l'élicitation des connaissances des experts, les libérant ainsi plus rapidement pour la réalisation de leur travail habituel.

La stratégie de co-modélisation des connaissances, qui amène de petits groupes d'employés à représenter leurs connaissances à l'aide d'un langage graphique semi-formel et qui a été expérimentée par Basque *et al*. (2008b), s'accorde bien à l'aspect consensuel de la définition d'ontologie fournie par Gruber (1993a) et ajustée par Borst (1997) : « *An ontology is a formal, explicit specification of a shared conceptualization.*» Cette activité nous semble ainsi pouvoir être mise à profit aux premières étapes de la conception d'une ontologie. Ainsi, nous pensons que le processus de construction d'une ontologie doit être décomposé en trois phases bien distinctes: une phase *d'élicitation* de la connaissance dans un formalisme de degré

[1] Protégé Home Site, *Welcome to protégé*: http://protege.stanford.edu/

[2] TopQuadrant, *TopBraid Composer (TM)*: http://www.topquadrant.com/products/TB_Composer.html

semi-formel relativement évolué, une phase de *formalisation* des connaissances où le modèle semi-formel est transformé dans un formalisme ontologique, puis une phase de *validation* qui assure la cohérence syntaxique et sémantique (par rapport au domaine de connaissances ciblé) de l'ontologie produite. C'est l'ensemble de ce processus qu'OntoCASE, une méthodologie instrumentée d'un assistant logiciel qui est présentée dans cette thèse, vise à soutenir.

Les avancées des dernières années en représentation des connaissances ainsi qu'en intelligence artificielle, notamment celles qui sont associées au Web sémantique, ont permis d'accroître la disponibilité et l'accessibilité des outils informatiques intelligents et puissants. Nous pensons que le développement d'une méthodologie comme celle développée ici doit inclure l'intégration de tels outils pour faciliter l'utilisation des méthodes et des techniques qui la composent, la rendant ainsi plus efficiente.

La modélisation est une activité importante dans plusieurs méthodologies de développement de logiciels. Qu'il s'agisse de la représentation de cas d'utilisations, de fonctionnalités de logiciels, de la structure de déploiement des modules de logiciels, de structures de classes pour la génération automatique de codes sources, le modèle issu de l'activité de modélisation est un artéfact au cœur du processus de génie logiciel. Or, ce modèle est de type représentationnel, c'est-à-dire que la principale fonction du modèle est de représenter « quelque chose » alors que l'opérationnalité (qui est la propriété d'exécuter des opérations ou des commandes) est concrétisée par la génération du code source à partir du modèle. Une caractéristique importante de cette façon de faire est que le modèle et le code source qui lui correspond forment deux artéfacts distincts. Cette distinction occasionne parfois des problèmes de synchronicité entre le modèle et le code source puisque les deux peuvent évoluer de façon indépendante.

En informatique, on peut considérer l'ontologie comme une sorte de conceptualisation d'un domaine qui inclut une fonctionnalité à la fois représentationnelle et opérationnelle, c'est-à-dire que l'ontologie sert à représenter « quelque chose » et permet l'exécution automatique de commandes informatiques. Ainsi, l'expression des ces deux fonctionnalités soutenue par un seul artéfact, comme le propose OntoCASE, offre plusieurs avantages. D'une part, elle permet une synchronisation en temps réel de l'une et l'autre des fonctionnalités selon que l'on modifie l'ontologie pour des raisons de représentationnalité ou pour des raisons d'opérationnalité et, d'autre part, l'ontologie, de par sa nature représentationnelle, constitue un document de référence compréhensible.

OntoCASE apporte des solutions concrètes et fonctionnelles aux problématiques organisationnelles exposées plus haut et qui vont être développées dans cette thèse. Ce travail a déjà été présenté dans des congrès scientifiques (Héon, Basque *et al.*, 2010 ; Héon, Paquette *et al.*, 2008, 2009a, 2009b) et dans un chapitre de livre dédié à la modélisation des connaissances à paraître en 2010 (Héon et Paquette).

Cette thèse est divisée en cinq chapitres et en sept appendices. Le premier chapitre, *La recension des écrits*, délimite le domaine de connaissances utilisé pour la thèse. Nous aborderons les concepts liés à la représentation et à la modélisation des connaissances. Le type de modèle de connaissances particulier que constituent les ontologies ainsi que leurs classifications seront aussi abordés et un aperçu sera fourni de quelques formalismes semi-formels et formels utilisés pour représenter des connaissances. Nous aborderons également le concept de métamodélisation en identifiant les processus qui permettent de réaliser la transformation de modèles. C'est par une implantation de cette technique que nous réaliserons la transformation d'un modèle semi-formel en ontologie. Une méthodologie d'ingénierie des connaissances sera ensuite présentée de manière générale. Nous y aborderons la structure d'une méthodologie, l'élicitation de connaissances, la conception de systèmes experts et l'ingénierie ontologique. Finalement, nous exposerons à la section huit quelques

notions concernant le volet assistance automatique de cette thèse, un aspect important qui sera implanté dans la méthodologie et les outils OntoCASE.

Le chapitre 2, *Méthodologie d'OntoCASE*, décrit la méthodologie et l'assistant informatique d'OntoCASE développés dans cette thèse. Les première et deuxième sections de ce chapitre posent l'hypothèse de recherche associée au cadre conceptuel d'OntoCASE ainsi que les objectifs de la recherche. La section 3 présente la description des volets d'OntoCASE, de la structure de l'ontologie de transformation et du portrait synthèse de la méthodologie à développer. Les sections 4 à 6 exposent successivement la structure de l'ontologie cadre, de l'ontologie de référence ainsi que la méthodologie de conception d'une ontologie à partir d'une conceptualisation semi-formelle.

Le chapitre 3, *La démarche de construction d'OntoCASE*, présente la démarche de recherche qui a permis de construire le volet procédural, représentationnel et computationnel d'OntoCASE. Le processus de construction d'OntoCASE a été divisé en trois phases, soit la phase 1 *Mise en place des composants architecturaux, procéduraux et informatiques,* la phase 2 *Agrégation des composants ontologiques, procéduraux et informatiques* et la phase 3 *Consolidation* dont l'objectif est de valider la méthodologie développée.

Le chapitre 4, *Validation d'OntoCASE*, présente les résultats de la validation méthodologique et computationnelle d'OntoCASE. Les résultats obtenus sont évalués selon les critères d'utilisabilité que sont l'efficacité, l'efficience et la satisfaction. La démonstration de la généricité de la méthodologie à transformer des modèles de connaissances dans divers formalismes est réalisée par l'implantation complète d'un langage semi-formel autre que celui utilisé à l'étape initiale de construction d'OntoCASE.

Le chapitre 5, *Conclusion et discussion,* présente le sommaire des réalisations, des contributions, des limites et des perspectives de la recherche selon les axes

informatiques, de la gestion des connaissances, de la représentation des connaissances et de l'ingénierie ontologique

En annexe à cette thèse l'appendice A présentent un *guide du langage de modélisation par objet typé* (MOT) qui sera utile au lecteur non familier avec le langage MOT. L'appendice B (*catalogue de la sémantique formelle de MOT*) présente la sémantique formelle d'exemples de modèles MOT qui couvrent l'utilisation de la totalité des règles de la grammaire et du vocabulaire du langage MOT. L'appendice C (*catalogue des cohérences de l'ontologie cadre d'OntoCASE*) présente les axiomes et les règles qui permettent de valider la cohérence de l'ontologie du domaine produit par la méthodologie. L'appendice D (S*cénarios de cas de tests*) présente les divers scénarios de tests qui permettent de valider la méthodologie par la réalisation des tests unitaires, des tests fonctionnels, des tests de systèmes et des tests d'incohérences. L'appendice E (*Éléments pour guider la modélisation semi-formelle à l'aide de concepts ontologiques*) est une contribution de la thèse qui sert d'outil méthodologique pour la validation sémantique d'une ontologie produite à partir d'une modélisation semi-formelle. L'appendic F (*Code source*) présente le code *Java* et *OWL* des principaux programmes et ontologies utilisés par OntoCASE. L'appendice G (*Évaluation expérimentale*) présente le protocole d'expérimentation et le certificat d'éthique utilisé pour réaliser l'expérimentation d'OntoCASE, suivis du résumé des commentaires de chacun des participants.

CHAPITRE 1

RECENSION DES ÉCRITS

Ce chapitre se divise en huit sections. La première section présente les fondements théoriques du concept de représentation autant du point de vue des sciences de l'information que de celui de la sémiotique et des sciences cognitives. Cette section introduit aussi le concept de couche d'abstraction. La deuxième section aborde le concept de modélisation des connaissances, dont celle des ontologies. La troisième section présente diverses catégories de classification des ontologies, qu'il s'agisse de la classification selon le spectre sémantique, selon le degré d'expressivité ou selon le degré de généralité du domaine représenté. La section présente aussi quelques exemples d'ontologies de différents degrés de généralité. Les sections 4 et 5 présentent respectivement les caractéristiques et des exemples de modèles de connaissances représentés dans des formalismes semi-formel et formel. La section 6 présente le concept de métamodélisation en tant que technique pour la transformation de modèles. La section 7 aborde les principaux sujets entourant l'ingénierie des connaissances, tels que la structure d'une méthodologie, l'élicitation des connaissances, la conception d'un système expert et l'ingénierie ontologique. Finalement, la section 8 développe les principaux concepts qui sont liés aux fonctions d'assistance à l'usager dans les environnements informatisés.

1.1 Concept de représentation

Le concept de représentation est au cœur de cette thèse. Il existe plusieurs façons d'interpréter ce concept selon le contexte dans lequel il est employé. Par le biais de la philosophie, de la psychologie et de la sémiotique, nous examinons la définition de ce

concept dans le domaine de la représentation des connaissances ainsi que dans le domaine de la représentation dans les systèmes d'informations. Nous ferons aussi appel à la sémiotique pour définir les concepts d'« abstraction », de « couche d'abstraction », de « polysémie » et de « synonymie ». Ces notions sont utiles pour orienter le design du processus de formalisation ainsi que du processus de désambiguïsation nécessaire à la transformation d'un modèle semi-formel en ontologie[3].

1.1.1 Théorie des idées

On retrouve chez Platon (428-348 av. J.C.) les notions de *monde des apparences* et de *monde réel* (voir la figure 1.1) par l'utilisation de l'abstraction entre un *objet* et son *idée*. Pour Platon, l'objet fait partie du monde des apparences, des choses périssables et éphémères. C'est l'idée qui définit la véritable nature des choses, qui, elle, fait partie du monde réel. De manière contemporaine, nous pouvons remplacer la notion d'idée par celle de *concept* (Jarrosson, 1992). La théorie des idées de Platon souligne la polarisation entre entités *objectives* et entités *subjectives*.

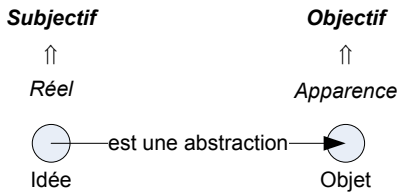


Figure 1.1: L'objet et son abstraction: l'idée

Le pôle de la réalité *subjective* fait référence à des éléments de l'esprit humain alors que les pôles de la réalité *objective* englobent les éléments observables de l'Univers, résidant hors de l'esprit humain, qui existent en dehors de la présence ou de la non-présence de l'humain (Dietz, 2006 p. 36).

---

[3] Tout au long de cette thèse, nous considérons entre autres que l'ontologie est un modèle formel. Une définition plus explicite d'ontologie sera donnée un peu plus loin dans ce chapitre.

1.1.2 Représentation dans un système d'information

Schématisée à la figure 1.2, le *modèle de la représentation* dans un système d'information a pour objectif d'établir une concordance représentationnelle entre les éléments de la réalité et leur représentation dans le système d'information. En plus de diviser la réalité entre *objectivité* et *subjectivité*, la représentation est aussi divisée entre *domaine* (représentant l'Univers réel[4]) et *système d'information* (représentant l'Univers virtuel). Dans cette représentation, la relation qui unit le *concept* et l'*objet* est la relation d'*instanciation.* L'instanciation est une relation d'abstraction non transitive qui associe un ou plusieurs objets concrets à une abstraction (le *concept)*. Le *type* et le *symbole* constituent les signes (cf. section suivante) qui représentent respectivement le concept et l'objet dans le système d'information. Le modèle de la représentation de la figure 1.2 ne représente pas la sémantique qui unit les éléments du domaine aux éléments du système d'information. Le modèle de la représentation qui intègre la sémantique relève de la sémiotique.

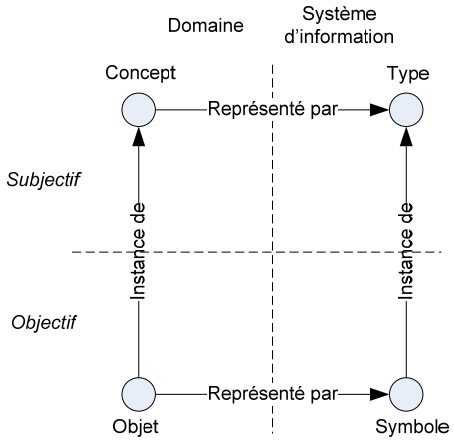


Figure 1.2: Le modèle de la représentation d'un domaine dans un système d'information. (Tirée et adaptée d'Olivé, 2007 p. 46)

[4] Dans ce contexte, tant les objets que leurs abstractions (les concepts) sont considérés comme faisant partie de l'Univers réel.

1.1.3 Sémiotique

"Ceci n'est pas une pipe" est la célèbre phrase incluse par Magritte (1898-1967) dans son tableau de 1929, intitulé *La trahison des images* et présenté à la figure 1.3. En première approche, cette affirmation semble paradoxale mais elle permet de souligner la différence existant entre un *objet*, son *représenté* et ce qu'il *signifie*. Normalement, la signification d'une image est « située » entre l'image et l'objet qu'elle représente. En incluant cette phrase dans le tableau, Magritte nous invite à déplacer la signification entre le spectateur et la toile, laissant sous-entendre que ce qui est observé par le spectateur n'est pas une pipe mais plutôt *l'image* d'une pipe.



Figure 1.3: Tableau *La trahison des images*, Magritte 1929. (Tirée de Paquet, 2006, p. 9)

En philosophie, Charles Sanders Peirce (1839-1914) est considéré comme le père de la sémiotique[5] moderne. Les notions de base associées à l'activité de modélisation de la connaissance tirent leurs origines de la sémiotique, dont le triangle sémiotique en est la représentation (voir la figure 1.4). L'*objet* est un observable, une chose proprement dite, par exemple: une personne, un animal ou une maison. L'objet possède des caractéristiques propres et il peut devoir son existence à l'agrégation de plusieurs autres objets. Par exemple, pour exister, une automobile se compose d'un

[5] Selon le Dictionnaire Antidote RX, la sémiotique est la "*science des modes de production, de fonctionnement et de réception des différents systèmes de signes de communication entre individus ou collectivités*".

moteur, de quatre roues, d'une carrosserie, etc. Les objets dont nous venons de discuter sont des objets dits *concrets* ou *matériels*. Or, l'existence de certains objets est intrinsèquement liée à la nature de l'existence humaine, que l'on pense au monde des émotions et des sentiments. Nous parlons alors d'objets *abstraits* ou *immatériels.* Des notions comme la vitesse, la liberté ou encore la grandeur sont aussi considérées comme des objets immatériels, qui peuvent parfois être même imaginaires tels qu'une licorne (Helbig, 2006). Selon le contexte d'utilisation, des mots comme *chose, référent, "thing"* peuvent remplacer le mot *objet.*

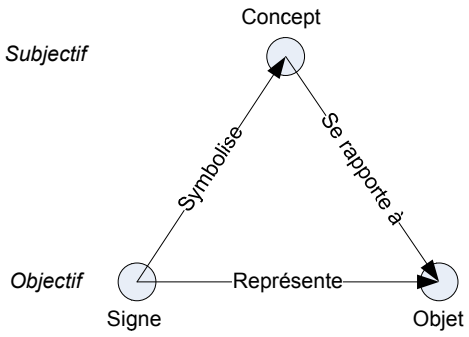


Figure 1.4: Le triangle sémiotique.

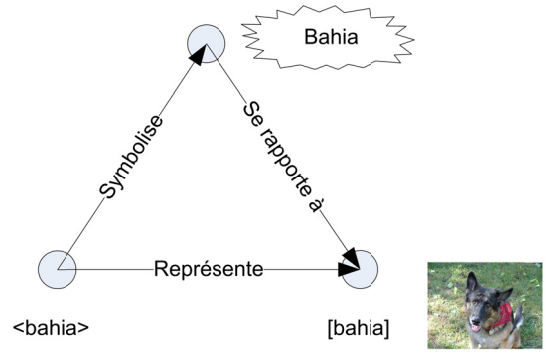


Figure 1.5: Exemple d'un triangle sémiotique associé à la conceptualisation du chien Bahia.

Le *signe* est aussi une chose de la réalité objective qui a pour fonction de *représenter* un objet. Par exemple, le nom <bahia>[6] est un signe qui représente spécifiquement notre chien familial [bahia][7]-[8], ou encore le signe <CXQ-511> sur une plaque d'immatriculation représente une automobile spécifique. Dans les systèmes d'information, la donnée (*data*) est un signe qui sert à représenter une observation de

[6] Le terme entre "< >" représente le signe associé à un objet concret.

[7] Le terme entre "[ ]" représente l'objet concret.

[8] Puisque nous ne pouvons pas inclure directement l'objet dans le texte, nous avons placé l'image de l'objet pour symboliser l'objet.

la réalité. Les termes tels que *symbole, mot, representamen* ou *signifiant* sont aussi utilisés pour désigner le *signe*.

Le *concept*, quant à lui, fait partie de la réalité subjective : il est une abstraction qui *se rapporte à* l'objet de la réalité objective et qui est *symbolisé* par le signe. Le concept est une abstraction issue du travail de classification de l'esprit humain. À l'instar des objets, il existe des concepts *abstraits* et des concepts *concrets*. La représentation mentale de « Bahia »[9] est un concept de type concret, car il renvoie à un objet de la réalité objective. En revanche, les concepts tels que « tout », « au moins un » ou encore « appartient à », sont des concepts de type abstrait. Des termes tels que *signifié, idée, interprétant, sens, image mentale* peuvent être substitués au terme *concept.*

La relation de *représentation* entre un signe et un objet existe grâce à la dynamique qui existe entre le signe qui *symbolise* un concept et le concept qui *se rapporte* à l'objet. En effet, la relation de représentation entre un signe et son objet n'existe que par le sens que l'esprit lui accorde. Sans l'esprit, il n'existe aucun lien entre un objet et son signe. C'est l'esprit qui associe une *sémantique* (un sens) à la représentation entre un signe et son objet.

1.1.4 Abstraction et couche d'abstraction

C'est en 1966 que René Magritte peint la toile *Les deux mystères* (voir la figure 1.6). On y voit en premier plan le tableau d'une pipe sur lequel est écrit « C*eci n'est pas une pipe »*. À l'arrière plan, est représentée une pipe peinte sur un mur. Dans cette toile, le tableau du premier plan définit l'objet en arrière plan en indiquant que l'objet peint n'est pas l'objet représenté. En réalité, l'objet peint est l'image d'une pipe. La signification associée à chaque image se superpose ainsi pour former des couches d'abstraction.

---

[9] Le nom entre « .. » représente le sens associé à l'objet concret



Figure 1.6: Les deux mystères. (Magritte 1966)

La superposition de triangles sémiotiques est un procédé qui a été exploité par Sowa (2000a) pour élaborer son concept de représentation des connaissances. Nous utilisons le même procédé pour élaborer le concept de *couche d'abstraction* (voir la figure 1.7). De manière pragmatique, nous considérons l'abstraction comme étant la relation inverse de « se rapporte à » que l'on pourrait traduire par « est-conceptualisé-par ». Elle est d'abord une relation non transitive entre les objets de l'Univers objectif et les concepts de l'Univers subjectif relevant de la représentation mentale des objets. En ce sens, nous dirons qu'un *concept* est une abstraction d'un *objet,* c'est-à-dire qu'il est produit par le processus de la pensée humaine.

Une caractéristique importante de la pensée humaine est la propriété qu'elle possède a classer les choses, par exemple en fonction des usages qu'elle désire en faire. Pour elle, le concept émergeant d'un premier processus de classification peut très bien être considéré en tant qu'objet dans un processus subséquent, ce qui est parfois nommé l'autoréférencement (Jarrosson, 1992 ; Pitrat, 1990, 1993). Ce nouvel objet pourra alors être désigné par un nouveau signe qui, lui-même, servira à représenter un nouveau concept créant des couches d'abstraction. C'est ainsi qu'en traitant « Bahia » en tant qu'objet, il est permis d'abstraire le concept « Chien » qui, à son tour, servira à abstraire le concept « Classe ». On pourra aussi dire que « Chien » est le métaconcept

de « Bahia » et que « Classe » est le métaconcept de « Chien » ainsi que de « Bahia ». Dans cette structure, le concept « Bahia » est désigné par <Classe:Chien:Bahia>

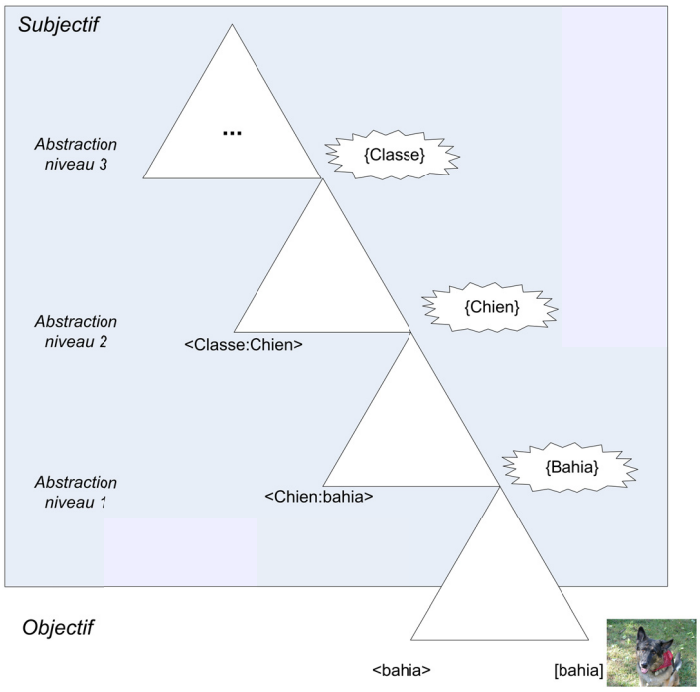


Figure 1.7: Le concept d'abstraction représenté à partir du triangle sémiotique.

1.1.5 Généralisation et abstraction

Soit A qui est subsumé par B et B qui est subsumé par C, alors il est permis de conclure que A est subsumé par C. C'est la propriété de *transitivité* que possède la subsomption qui nous permet de déduire cette conclusion. En langage naturel, la subsomption s'interprète par le terme « sorte-de » qui est un processus de généralisation entre concepts qui appartiennent au même niveau d'abstraction. Ainsi, l'énoncé de début s'interprète de la façon suivante: A est une sorte de B et B est une sorte de C, alors il est permis de conclure que A est une sorte C. En remplaçant A par « Chien », B par « Mammifère » et C par « Animal », alors l'énoncé s'écrit: soit chien est une sorte de mammifère et mammifère est une sorte d'animal, alors chien est une sorte d'animal. Ce qui est un énoncé vrai. Par contre, de l'énoncé suivant : « chien est

une sorte de mammifère et mammifère est une sorte de classe biologique », il n'est pas permis de conclure que chien est une sorte de classe biologique. Ici nous touchons à une distinction importante entre généralisation et abstraction qui, dans ce dernier cas, est utilisée pour changer de couche d'abstraction. Dans ce contexte, cette proposition fait référence à la définition de termes et non à une classification de ceux-ci. Lorsque la proposition de généralisation est utilisée pour définir un terme, la relation n'est pas *transitive* et on emploie plutôt le terme « est un ». Il faudrait donc dire « chien est une sorte de mammifère et mammifère est une classe biologique ».

1.1.6 Modèle sémiotique de la synonymie et de la polysémie

La polysémie et la synonymie sont des notions qui se trouvent au cœur de l'activité de désambiguïsation d'OntoCASE. À titre d'exemple, la figure 1.8 présente le modèle de la sémiotique (voir la figure 1.4) schématisé dans le langage graphique de *Modélisation par Objets Typés* (MOT) (Paquette, 2002b), dont la description est disponible à l'appendice A de cette thèse.

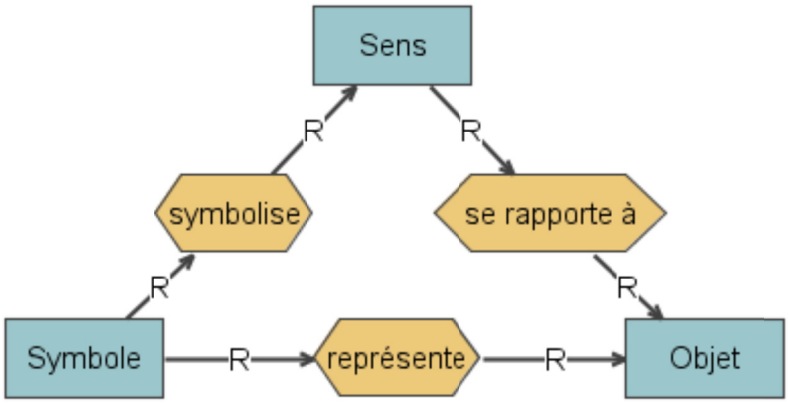


Figure 1.8 : Le modèle sémiotique représenté en MOT.

L'*objet* (la chose objectivement réelle) est *représenté* par un symbole. Le symbole *symbolise* le *sens* (la sémantique) qui se *rapporte* à l'*objet*.

Dans la perspective sémiotique, nous définissons la *synonymie* par une sémantique (le sens) qui est symbolisée par au moins deux symboles qui représentent *le même objet*. Dans l'exemple de la figure 1.9, le sens *un ensemble composé de trois X plus un X (=4)*[10] est symbolisé par les synonymes *quatre* et *4*.

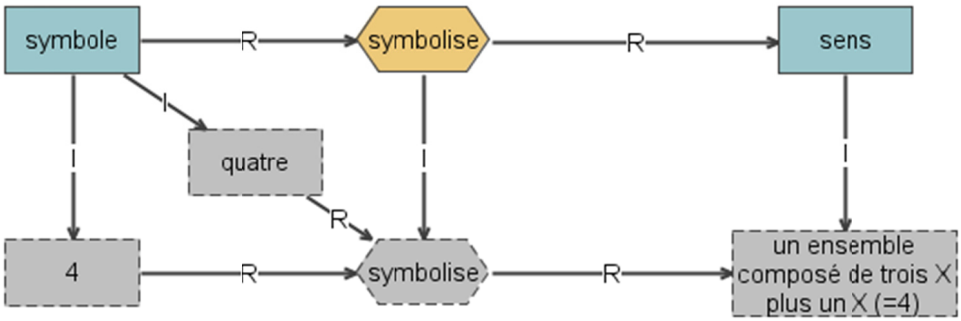


Figure 1.9 : Exemple de synonymie entre *4* et *quatre.*

Toujours dans le contexte du modèle sémiotique, la polysémie est un *symbole* qui *symbolise* deux *sens* ou un *symbole* qui *représente* deux *objets différents*. Par exemple, la figure 1.10 schématise la polysémie associée au mot *feu* qui symbolise d'une part, *une source de chaleur*, et d'autre part, un *décès depuis peu*[11].

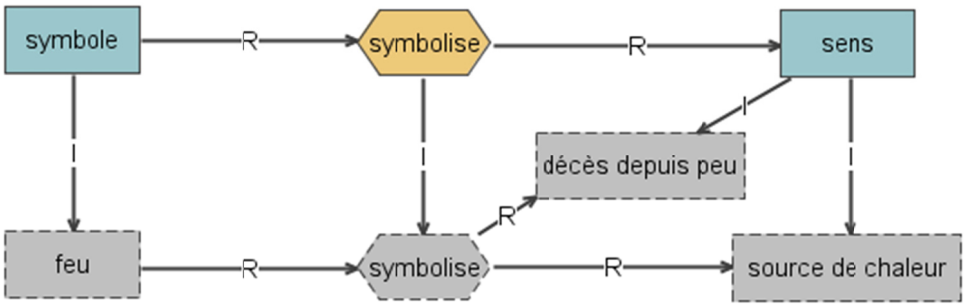


Figure 1.10 : Exemple de polysémie du mot *feu*.

[10] Définition extraite du dictionnaire Antidote HD.

[11] *Ibid*.

1.1.7 Donnée, information et connaissance

Nous pouvons associer les termes classiquement distingués en gestion des connaissances de « donnée », « information » et « connaissance » aux trois pôles du triangle sémiotique. En gestion des connaissances, Prax (2003) définit la connaissance ainsi:

> *[…] Pour qu'une information devienne connaissance, il faut que le sujet puisse construire une représentation qui fasse sens. Pour cela, l'information reçue subit une série d'interprétations (filtre retraitements), liées aux croyances générales (paradigmes), au milieu socioprofessionnel, au point de vue, à l'intention, au projet de l'individu porteur. Contrairement à l'information, la connaissance n'est pas seulement mémoire. Item figé en stock, mais toujours activable selon une finalité, une intention, un projet. Il y a dans la connaissance une notion de process, construction d'une représentation finalisante d'une situation en vue d'une « bonne fin » […]*[12]

Une analyse sémiotique de la connaissance (voir la figure 1.11) place à la base du triangle sémiotique la relation de représentation entre *information* et *donnée.*

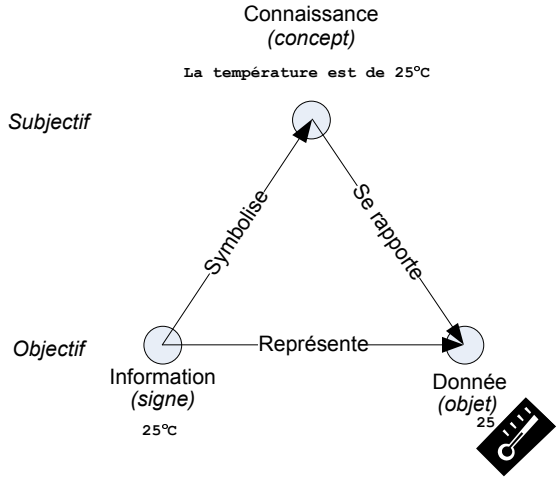


Figure 1.11: Donnée, information et connaissances dans le modèle sémiotique.

La donnée est le résultat d'un processus de perception de la réalité. Dans l'exemple de la figure 1.11, l'objet [25] qui est une donnée devient une information lorsqu'elle est

---

[12] Prax, *Le manuel du knowledge management* p.63.

contextualisée en degrés Celsius <25 °C>. L'information symbolise le concept « la température est de 25 °C » ce qui peut, par exemple, amener un sujet à inférer qu'« il faut assez beau pour faire un pique-nique ». Nous considérons la connaissance comme étant la signification que l'on donne à une information. De manière sémiotique, on pourrait dire que la connaissance est le signifié (ou la sémantique) d'une information.

1.1.8 Représentation des connaissances

Du point de vue sémiotique (voir la figure 1.12), la représentation des connaissances est un métaconcept (un concept décrivant un concept) qui se rapporte à une connaissance (Sowa, 2000a). Dans cette superposition, la connaissance devient l'objet se rapportant au concept de représentation des connaissances. Ainsi, la connaissance est représentée par un symbole qui en devient le signe. C'est donc dire que le concept de représentation des connaissances est un processus qui traite la connaissance comme un objet représenté par un symbole.

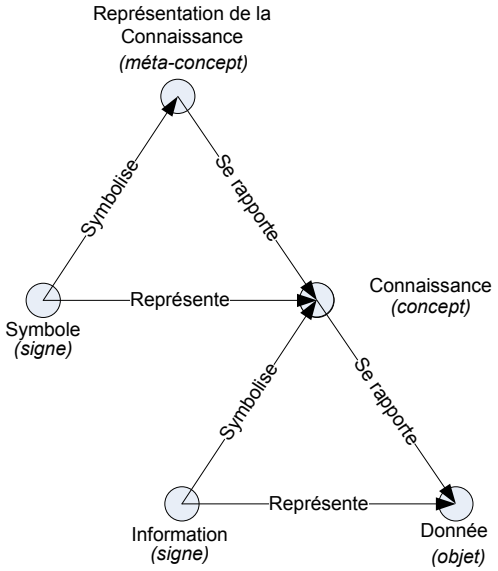


Figure 1.12: Le métaconcept de représentation des connaissances.

1.1.9 Représentation des connaissances en sciences cognitives

Dans le processus de communication, Heldig (2006) positionne la représentation des connaissances entre le modèle mental et le langage humain (voir la figure 1.13). L'hypothèse du modèle mental (Johnson-Laird, 2004) postule qu'à l'intérieur de chaque esprit, il existe une représentation mentale de la réalité qui est issue de la perception que nous avons de notre environnement. Cette représentation forme un modèle mental qui permet à chaque personne de réaliser des déductions nécessaires à toute communication. La représentation des connaissances peut être définie comme un processus d'extériorisation de ce modèle mental sous la forme d'un modèle externe, mais qui ne lui est pas nécessairement fidèle. L'interprétation plus ou moins formelle du modèle par des personnes ou des systèmes computationnels permet la communication entre les agents cognitifs.

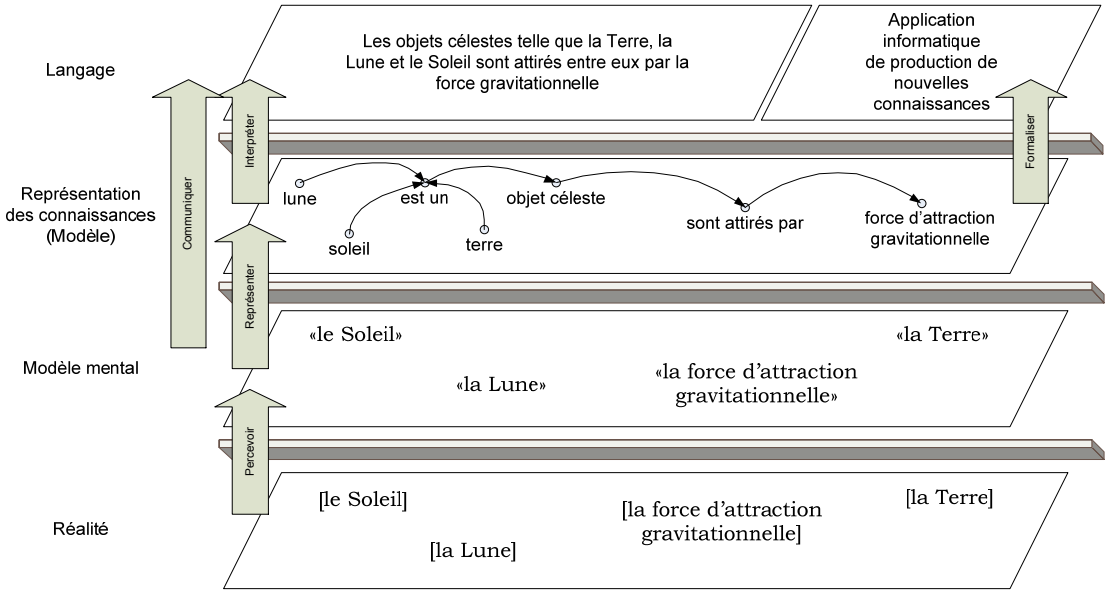


Figure 1.13: Modèle cognitif de la représentation des connaissances. (Tirée de Helbig, 2006, p. 8)

La figure 1.13 représente les divers niveaux d'abstraction du modèle cognitif de la représentation des connaissances. Le premier niveau, celui de la *réalité*, contient un

certain nombre d'objets qui sont perçus par les sens ([la lune], [le soleil], etc.). Les objets de la réalité perçus sont représentés sous une forme quelconque dans l'esprit pour former un modèle mental. Dans le schéma de l'exemple, les modèles mentaux sont représentés par une encapsulation dans des guillemets français (« ») pour en faciliter la lecture. Il existe plusieurs hypothèses sur la nature des représentations dans le modèle mental, dont les deux principales sont l'hypothèse connexionniste et l'hypothèse symbolique. L'hypothèse connexionniste, que nous avons étudiée dans des travaux antérieurs à cette thèse (Héon, 1998 ; Héon *et al.*, 1998), postule que la cognition est le résultat de l'interaction entre des entités élémentaires d'un système. Le modèle couramment utilisé est celui du neurone (l'entité élémentaire) relié via la synapse à un autre neurone (l'interaction) constituant ainsi un réseau de neurones (le système). La force de l'interaction synaptique entre les neurones est pondérée par le poids synaptique et la réponse du neurone aux stimulations intrants est modulée par la fonction de réponse du neurone, qui dans le modèle classique est une fonction sigmoïde. La modulation des poids synaptiques et la modulation des paramètres de la fonction de réponse de chacun des neurones, suite à une stimulation, permettent de reproduire certains aspects de la cognition. L'approche symbolique postule que la cognition est le produit de la manipulation de symboles. Le langage naturel, la lecture de la musique, l'utilisation des mathématiques sont autant de systèmes de codage qui mettent en opération la cognition. Nous allons d'ailleurs traiter largement de ce sujet dans cette thèse.

Le modèle mental peut être extériorisé sous la forme d'un modèle externe plus ou moins formel de ses connaissances. Une des limites des langages de représentation de connaissances est liée au type de connaissances qu'ils sont en mesure de représenter. Par exemple, l'ontologie est spécialement dévolue à la représentation de connaissances *déclaratives* de type *Concept*, alors que d'autres langages formels, tels que le *Business Process Modeling Notation (BPMN[13])*, sont spécialisés dans la

[13] OMG BPMN, *Object Management Group/Business Process Management Initiative*: http://www.bpmn.org/

représentation de connaissances *procédurales*. Notons finalement que la représentation de connaissances *stratégiques* (Paquette, 2002b ; Paris *et al.*, 1983) associée à l'utilisation de conditions ou de contraintes[14] est aussi couramment utilisée par exemple dans les arbres de déduction ou de décision. Un avantage notoire du langage MOT est qu'il peut représenter dans un même modèle, voire dans un même schéma, l'un et l'autre des types de connaissances énumérées ci-haut. L'un des défis de cette thèse réside dans la représentation de ces divers types de connaissances par une ontologie qui, par définition, représente des connaissances déclaratives.

1.2 Modélisation des connaissances

De façon générale, la modélisation est l'activité de construction d'un modèle externe de connaissances (Alhir, 2002). Pour réaliser cette activité, nous devons clarifier les concepts de langage et de modèle. Nous les abordons dans cette section, de même que ceux d'ontologie, de métamodélisation, d'abstraction et d'espace de modélisation. Les principes de transformation de modèles sont également discutés. La discussion concernant ces concepts permet de définir les notions théoriques nécessaires à la transformation d'un modèle semi-formel en ontologie.

1.2.1 Langage

Le langage est un moyen d'expression utilisé pour exprimer et échanger de l'information ou des connaissances. Pour Alhir (2002), la structure d'un langage se divise en trois parties: la syntaxe, la grammaire et la sémantique (voir la figure 1.14). La syntaxe définit le vocabulaire utilisé par le langage alors que la sémantique définit le sens à donner aux différents mots composant le vocabulaire. La grammaire, quant à elle, définit les règles d'utilisation du vocabulaire.

---

[14] OMG OCL, *Object Constraint Language Specification, version 2.0*: http://www.omg.org/technology/documents/formal/ocl.htm

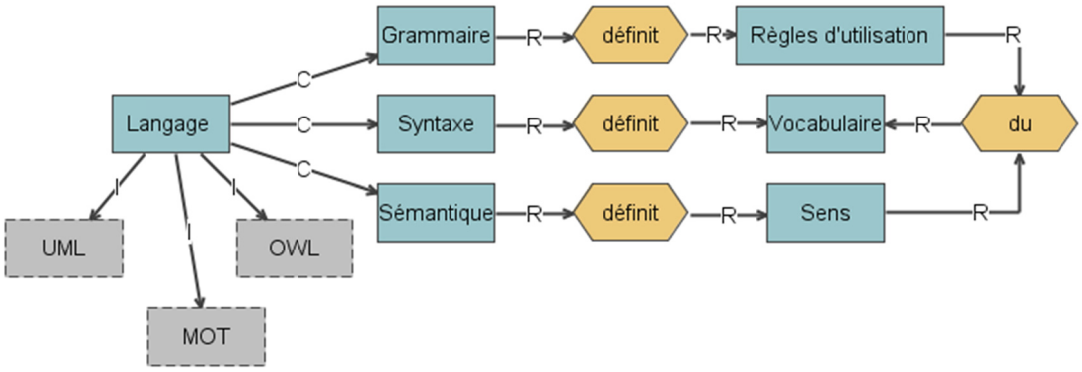


Figure 1.14: Langage de modélisation.

Les langages *Unified Modeling Language* (UML) et *Modélisation par Objets Typés* (MOT) sont des exemples de langages de modélisation permettant la conception de modèles graphiques. L'*Ontology Web Language* (OWL) est aussi un langage qui permet la conception d'un modèle de type ontologique.

1.2.2 Modèle

Dans nos activités courantes, il est plutôt rare que nous traitions ou analysions un seul objet. Normalement, nous sommes confrontés à l'analyse d'un ensemble d'objets mis en relation les uns avec les autres. Les caractéristiques d'un objet donné peuvent évoluer dans le temps ou encore changer d'état en présence d'un autre objet. La dynamique et les relations qui agissent sur les objets forment un tout global : ils forment un *système,* ce que Gruber (1993b) appelle l'« Univers du discours ». Le modèle est une abstraction qui permet de *représenter* un *système*. Deux concepts fondamentaux composent le modèle, soit *l'entité* et la *relation* (Chen, 2002). L'entité renvoie à l'objet (concret ou abstrait) par exemple une automobile ou la vitesse alors que la relation (ex : appartient à, est une sorte de) renvoie aux objets abstraits qui permettent de mettre en relation des objets concrets ou d'autres objets abstraits entre eux. Pour qu'un modèle soit interprétable, les signes et les règles d'utilisation de ces signes – qui forment un *langage* – permettent de décrire le « rapport » entre un

concept et son objet. Le modèle doit être vérifiable et véridique pour le système que le modèle tente de représenter (Tarski, 1944). Le modèle est donc une abstraction décrivant un système selon un certain point de vue exprimé avec un certain langage.

1.2.3 Classification des langages de représentation des connaissances selon leur degré de formalisme

Uschold et Gruninger (1996) classifient les langages de représentation des connaissances selon quatre degrés de formalisme:

- Le langage naturel, dont l'expressivité est très élevée, constitue ce que ces auteurs appellent le degré *hautement informel*. Ce type de représentation des connaissances présente un haut degré d'ambiguïté, que le cerveau humain arrive à décoder en partie, mais que l'ordinateur arrive mal à traiter, comme en témoignent les difficultés qui subsistent dans les efforts d'interprétation automatisée du langage naturel.
- Le degré *semi-informel* s'exprime au moyen d'une forme restreinte et structurée du *langage naturel*, telle que celle que l'on retrouve dans la description d'algorithmes en langage stéréotypé appelé pseudocode. Ce formalisme réduit la spontanéité et le degré d'expressivité de la représentation, mais, en revanche, il en rehausse la clarté en réduisant certaines ambiguïtés bien que de manière encore intraitable par un système informatique.
- Le degré *semi-formel* fait référence au formalisme du *langage artificiel* employé pour le traitement partiellement automatique des connaissances. Le langage de modélisation MOT (Paquette, 2002b ; Paquette *et al.*, 2007) de même que le langage graphique UML (OMG UML, 1997-2006) largement utilisé en informatique sont deux exemples de langages de représentation semi-formels. Les représentations de ce type peuvent encore contenir des ambigüités. En revanche, une certaine relâche des contraintes langagières,

comparativement aux langages formels, favorise et stimule l'expressivité des connaissances et la communication entre humains.

- Finalement, le degré *rigoureusement formel* que l'on retrouve dans les langages comme OWL (Martin, 2002 ; McGuinness et Harmelen, 2004) ou encore les graphes conceptuels de Sowa (2000a) offre une représentation des connaissances dont l'expressivité est certes réduite, par rapport au langage naturel, mais qui en élimine les ambiguïtés ce qui permet un traitement informatique des connaissances représentées.

1.2.4 Ontologies dans la perspective des sciences cognitives

Chez les Grecs anciens, le terme **ontologie** signifie : « *étude de l'être en tant qu'être, indépendamment de ses déterminations particulières* »[15] laissant sous-entendre un discours indépendant d'un point de vue quelconque. Il importe donc de faire une distinction entre l' « observateur » des choses et les choses « observées ». En philosophie, il existe trois hypothèses sur l'appréhension de la connaissance : l'objectivisme, le subjectivisme et le constructivisme (Dietz, 2006). L*'objectivisme* défend l'hypothèse selon laquelle le monde existe par lui-même, de façon indépendante de l'observateur. L'observateur croit en la vérité d'une réalité objective. Le *subjectivisme* défend l'hypothèse que la réalité n'existe pas en dehors du sujet qui l'observe et qu'à la limite, chaque observateur a sa propre image de la réalité. Quant au point de vue *constructiviste,* il admet l'hypothèse qu'il n'y a pas d'objectivité absolue de la réalité, que la réalité existe aussi par l'interprétation de celle-ci. En contrepartie, il admet aussi que les éléments interprétés font partie d'une réalité que l'on pourrait qualifier de semi-objective. Pour Dietz, l'ontologie s'interprète selon un point de vue constructiviste, qui est exprimé schématiquement à la figure 1.15. L'ontologie est une composante particulière de la réalité qui sert de base de communication au discours sur une partie de la réalité (objectivisme). En même

[15] Dictionnaire Antidote RH

temps, l'ontologie est le résultat d'une construction issue de l'interprétation d'un agent intelligent (subjectivisme). Les termes *sérialisable et interopérable* sont empruntés au domaine de l'informatique. L'attribut sérialisable indique que l'encodage fait en sorte que l'ontologie peut être utilisée pour l'entreposage en mémoire ou le transport sur un réseau. L'interopérabilité est l'attribut qui indique que l'ontologie reste véridique et vérifiable indépendamment de l'agent cognitif qui l'interprète, qu'il soit humain ou machine.

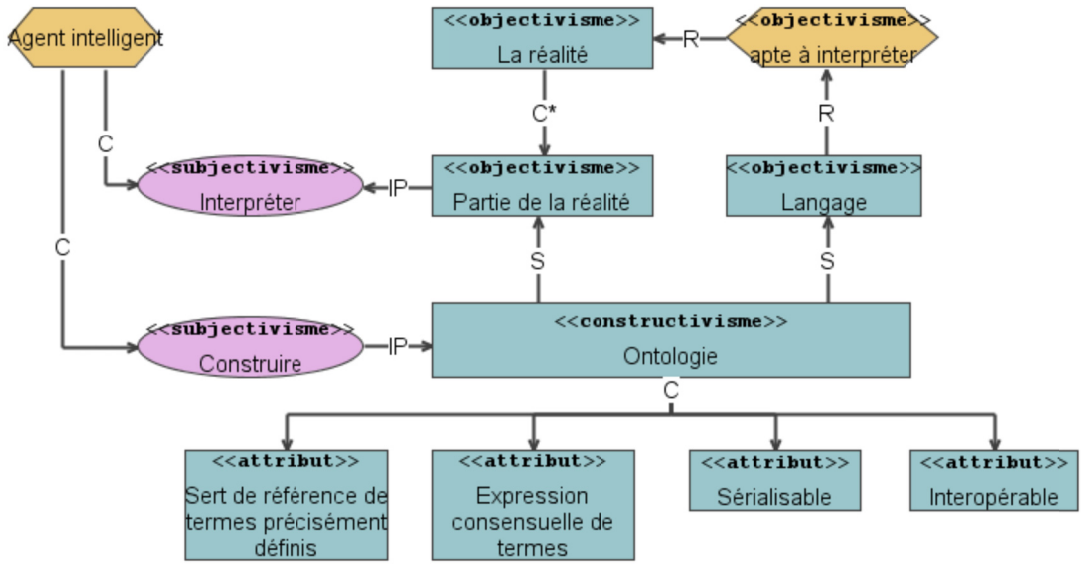


Figure 1.15: Positionnement constructiviste de la définition d'ontologie.

1.2.4.1 Ontologies en tant que systèmes de représentation des connaissances de degré formel

Tel que déjà mentionné, une définition du terme « ontologie » généralement admise est celle proposée par Gruber (1993a) et adaptée pour son volet consensuel par Borst (1997) : « *an ontology is a formal, explicit specification of a shared conceptualization* ».

Du point de vue de l'ingénierie ontologique, cette définition est interprétée par Calero, Ruiz et Piattini (2006) de la façon suivante :

> La « conceptualisation » fait référence au modèle abstrait d'un phénomène dont les concepts pertinents ont été identifiés :
>
> - le terme « explicite » fait référence au fait que les types de concepts utilisés et les contraintes de leur utilisation sont explicitement définis;
> - le terme « formelle » renvoie au fait que l'ontologie devrait être lisible ou interprétable par la machine;
> - le terme « partagée » associé à « conceptualisation » porte l'idée que l'ontologie exprime une connaissance consensuelle, c'est-à-dire acceptée par un groupe. [16]

Gómez-Pérez *et al.* (1996) fournissent, pour leur part, une interprétation de la définition de l'ontologie du point de vue de sa forme. Selon eux,

> une ontologie définit le vocabulaire d'un domaine d'application ainsi que les règles de combinaison des termes et des relations permettant de définir les extensions à ce vocabulaire [17]:

Pour l'essentiel, les ontologies, en tant que systèmes de représentation, peuvent être considérées au même titre que des systèmes de modélisation conceptuelle, tels que le modèle *entité-relation* de Chen (1976) ou le modèle *UML* de Rumbaugh, Jacobson et Booch (1999). Cependant, l'ontologie se distingue de ces systèmes de représentation par les points suivants (Oberle, 2006):

- l'idée primaire des ontologies est d'établir un consensus autour d'un signifiant à attribuer à un vocabulaire afin de faciliter l'échange d'informations entre applications ;
- les ontologies sont formalisées au moyen d'un système de représentation logique dont la sémantique est spécifiée de façon non ambigüe ;

---

[16] *«A 'conceptualization' refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used, and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable. 'Shared' reflects the notion that ontology captures consensual knowledge that is, it is not private to some individual, but accepted by a group »*

[17] *«Defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary»*

- l'ontologie est exprimée au moyen d'un langage de représentation qui, à l'état d'exécution, sert de représentation à des applications de recherche et de raisonnement.

1.2.4.2 Ontologies dans la perspective informatique

De façon contemporaine, le concept d'ontologie est réapparu vers les années quatre-vingt par l'introduction des théories de la logique dans la construction de systèmes en IA. La figure 1.16 présente sous forme schématique une synthèse des principales définitions d'une ontologie du point de vue informatique (Gómez-Pérez et al., 2003; T. Gruber, 1993; Guarino, 1995; Lacy, 2005; Studer et al., 1998).

En ingénierie informatique, une ontologie est un artéfact (parfois un fichier *XML*) dont la structure permet de représenter des concepts, des relations, des axiomes et des faits. Le *concept* est la représentation d'une entité abstraite de la réalité à décrire. Les vocables de *terme* ou *classe* sont aussi parfois employés en tant que synonymes de *concept*. À un niveau d'abstraction factuelle, on retrouve l'*instance* ou l'*individu* dont le représenté est en lien avec un objet concret de la réalité observée. L'action de créer un fait à partir d'un concept se nomme : *instancier*. Finalement, l'*axiome,* qui définit le concept, est une assertion (c'est-à-dire un énoncé qui est supposé vrai dans le domaine représenté) à propos des instances en relation avec les concepts de l'ontologie.

De manière conceptuelle, une ontologie est la description d'une réalité et la formalisation d'un modèle conceptuel, qui, selon Gruber, correspond à une *conceptualisation*. Le processus d'*interprétation* est assumé par un agent intelligent formel. L'agent a pour fonction de simuler la réalité par un mécanisme formel de déduction dont l'ontologie est le sujet. L'ontologie est donc le résultat d'une double activité de modélisation et de formalisation. La *modélisation* est un processus de représentation de la réalité dont la résultante est un modèle de connaissances. La *formalisation* est un processus « de mise en forme » de ce modèle conceptuel afin que

l'artéfact résultant (l'ontologie) soit interprétable, de façon cohérente et non ambigüe, par un agent informatique (formel). La logique de descriptions (Baader, Horrocks et Sattler, 2004; Haarslev, 2005; Lutz, 2007), dont la logique des prédicats du premier ordre est le surensemble, est un système de représentation couramment employé pour formaliser les ontologies.

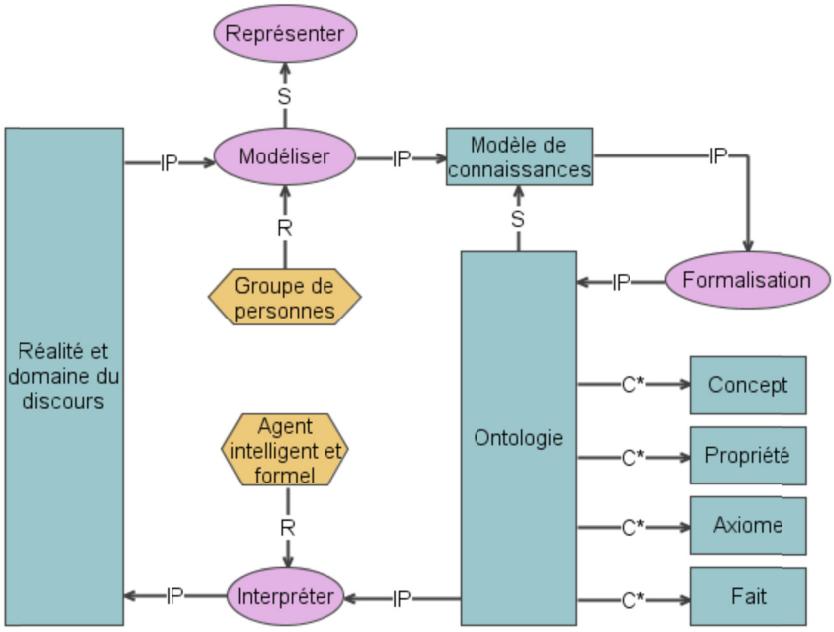


Figure 1.16: Définition d'une ontologie en informatique.

1.3 Classification des ontologies

Nous avons vu à la section 1.2.4.1 que, d'un certain point de vue, l'ontologie peut être considérée comme un système de représentation de connaissances, voire un langage. En tant que sujet d'étude, l'ontologie est un objet qu'il est possible de classifier selon certains attributs, qu'il s'agisse de son degré de précision sémantique, de son échelle d'expressivité, ou encore du degré de généralité du domaine représenté. L'interopérabilité des ontologies ─ c'est-à-dire la propriété qu'elles possèdent d'exister indépendamment de la nature ou du type de systèmes informatiques qui les manipulent ─ est l'une des propriétés que la catégorisation tente de mettre en valeur. Le développement d'une ontologie représentant un ensemble de métadonnées ou

d'une ontologie cadre (que nous définissons plus loin) sont des exemples d'implantation d'ontologies qui valorisent des aspects particuliers de l'interopérabilité. Dans les pages qui suivent, nous traitons de ces différents critères de classification des ontologies. Ces critères ont permis de classifier les diverses ontologies que nous avons développées pour OntoCASE.

1.3.1 Classification des ontologies selon la précision sémantique

Une classification des ontologies selon leur précision sémantique a été proposée par McGuiness (2003), qui les a placées sur un « spectre sémantique ». Selon cet auteur et contrairement à ce qui a déjà été énoncé, la classification selon la précision sémantique implique qu'une ontologie n'est pas nécessairement de degré formel. Présenté à la figure 1.17, le spectre sémantique est composé de neuf degrés de précision sémantique, allant du simple catalogue de termes à l'ontologie sophistiquée. Les quatre premiers niveaux définissent une classification de termes selon une sémantique exprimée en langage naturel. Le catalogue et le glossaire constituent des regroupements de termes avec leur définition. Le thésaurus ajoute le concept d'association permettant le référencement entre les termes. La catégorie "informal is-a" fait référence à une classification informelle de termes, exprimée en langage naturel, alors que la catégorie "formal is-a" fait plutôt référence à une classification d'ordre taxonomique. Le *frame* est un modèle proposé par Minsky (1985) qui fait référence aux concepts de *classes,* possédant des *propriétés*, des *attributs* et des « cases » (*slots*). Finalement, le niveau ultime de classification est l'ontologie formelle apte à représenter des restrictions, tel que le propose la logique du premier ordre.

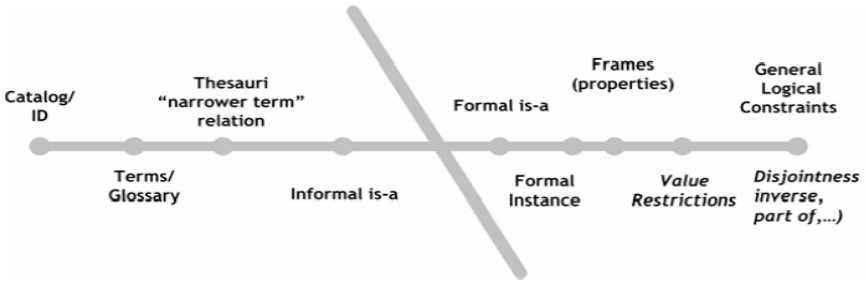


Figure 1.17: Échelle de classification d'ontologies en fonction du spectre sémantique de McGuiness. (Tirée de Breitman *et al.*, 2007, p. 26)

1.3.2 Expressivité d'un système de représentation de degré formel

Le tableau 1.1, inspiré de Hepp et *al.* (2008), présente l'échelle d'expressivité de divers systèmes de représentation formels.

Tableau 1.1
Expressivité de divers systèmes de représentation formels

Échelle d'expressivité	***Système de représentation***
Forte	Logique d'ordre supérieur
	Logique du premier ordre
	Logique descriptive
	Taxonomie
	Entité/Relation
Faible	Vocabulaire

L'expressivité associée à un système de représentation de degré formel est une mesure de sa capacité à exprimer les parties de la réalité à représenter. Une expressivité forte permet la mise en place de mécanismes de raisonnement sophistiqués de l'interprétation. En contrepartie, elle nécessite un plus grand effort de conception et d'interprétation et elle augmente le coût computationnel du raisonnement (Hepp *et al.*, 2008 ; Oberle, 2006).

1.3.3 Classification selon le degré de généralité du domaine représenté

Guarino (1998) a été le premier à proposer une classification des ontologies selon la généralité du domaine qu'elles représentent:

- l'ontologie de haut niveau (*upper level ontology*) décrit les concepts de haut niveau d'un domaine. Par exemple, des sujets tels que le temps ou l'espace peuvent être décrits dans ce type d'ontologies;
- l'ontologie de domaine (*domain ontology)* sert à décrire le vocabulaire d'un domaine spécifique. Les concepts qui la composent sont des spécialisations de haut niveau;
- l'ontologie de tâche (*task ontology)* décrit le vocabulaire nécessaire à la réalisation d'une tâche associée aux concepts définis dans l'ontologie de haut niveau;
- l'ontologie d'application *(application ontology)* décrit le vocabulaire d'une application spécifique.

Gómez-Pérez *et al.* (2003) proposent une échelle de généralité légèrement différente pour la classification des ontologies qui est plus élaborée, précise et sans nul doute plus complexe à utiliser que celle présentée par Guarino:

- l'ontologie de représentation des connaissances (*Knowledge Representation Ontology*) décrit le vocabulaire associé aux éléments de modélisation d'un système de représentation de la connaissance;
- l'ontologie d'usage générique (*Generic and Common Use Ontology*) décrit les connaissances de sens commun utilisées dans un domaine particulier de connaissances;
- l'ontologie de haut niveau (*Upper Ontology*) décrit les concepts généraux tels que ceux de temps et d'espace;

- d'autres ontologies possèdent la spécificité portée par le sens de leur nom telles que les suivantes : *Domain Ontology, Task Ontology, "Domain-task Ontology*, *Method Ontology* et *Application Ontology"*.

1.3.4 Ontologie et métadonnée

Une métadonnée est une donnée concernant une autre donnée. Par exemple, les informations au sujet d'un document, telles que le nom de l'auteur, la date de parution, le nom de la maison d'édition, etc., sont toutes des métadonnées. Dans le contexte du Web sémantique, Berners-Lee (1997) définit ainsi la métadonnée : *"a machine understandable information about Web resources or other things"*. Les ontologies de type métadonnées visent à définir le vocabulaire d'un domaine, mais pas nécessairement la sémantique d'un domaine. Elles servent donc de base de référencement à des systèmes de localisation et de recherche de ressources ou d'informations. La norme Dublin Core (DC[18]) et son implantation en langage OWL (DC-OWL[19]) est un exemple d'ontologie de type métadonnée. Structuré selon les éléments présentés au tableau 1.2, le DC-OWL permet le référencement de documents et son traitement par des applications compatibles avec OWL.

D'autres ontologies de type métadonnées ont été développées. Par exemple, le *Platform for Internet Content Selection* (PICS[20]), mis au point en 1995 par le consortium international W3C, est une ontologie dédiée au contrôle de l'accès au contenu de pages Web. Quant au *vCard file format,* proposé par le consortium Versit (Apple, IBM, AT&T et Siemens), il est dédié aux applications de type *e-business* (Breitman *et al.*, 2007).

---

[18] *Dublin Core Metadata Initiative (DCMI)*: http://www.dublincore.org/

[19] *Dubling Core - OWL*: http://protege.stanford.edu/plugins/owl/dc/protege-dc.owl

[20] W3C PICS, *Platform for Internet Content Selection (PICS)*: http://www.w3.org/PICS/

Tableau 1.2
Éléments du Dublin Core (tirée de Breitman *et al.*, 2007, p. 178)

Element	Definition
Title	object name
Creator	person/people responsible for the intellectual property of the object
Subject	main topic
Description	description of the object contents
Publisher	agent or agency responsible for making the object available
Contributor	person/people that made significant contributions to the object
Date	publication date
Type	object type, e.g., fiction, reference, novel, poem
Format	physical manifestation of the object, e.g., executable file, PDF
Identifier	an unambiguous reference to the resource within a given context
Relation	reference to a related resource
Source	a Reference to a resource from which the present resource is derived
Language	language of the intellectual content
Coverage	spatial location and temporal duration of the object
Legal rights	legal rights information about the object in question

Dans OntoCASE, nous utilisons une ontologie de type métadonnée pour définir le vocabulaire du langage semi-formel qui sera traité.

1.3.5 Ontologie cadre

Nous utilisons le terme d'ontologie cadre dans le même sens que celui que lui attribuent Guarino (1998) et Gómez-Pérez *et al.* (2003), c'est-à-dire une ontologie qui sert à décrire des concepts généraux. De par leur nature générique, certains projets ont pour objectif de concevoir des ontologies cadres disponibles et partageables par l'ensemble d'une communauté. Dans OntoCASE, nous avons construit une ontologie de ce type pour décrire les concepts généraux associés à la représentation des connaissances. Cette ontologie est au cœur du processus de formalisation.

Développée par le *IEEE Standard Upper Ontology* (SUO) *Working Group,* l'ontologie *Suggested Upper Merged Ontology* (SUMO[21]), originalement conçue dans la notation KIF *(Knowledge Interchange Format*), est aussi disponible en OWL (voir la figure 1.18). SUMO contient les concepts généraux concernant les sujets suivants (Niles et Pease, 2001):

- la structure liée au concept, telle que l'instance et la sous-classe;
- les types généraux comme l'*objet* et le *processus;*
- les abstractions incluant la théorie des ensembles, les attributs et les relations;
- les unités de mesure;
- les concepts liés au temps comme la durée;
- les relations de méronymie et d'holonymie
- les relations sémiotiques de base

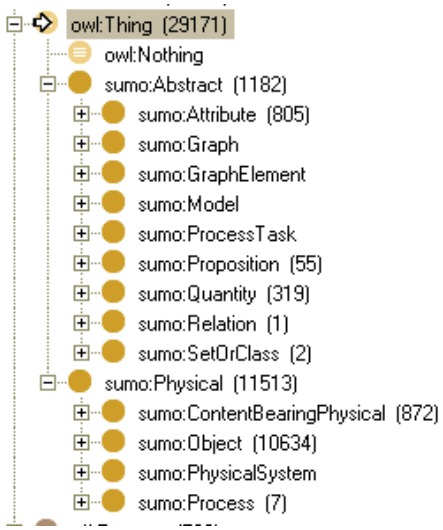


Figure 1.18: Taxonomie des classes de haut niveau dans SUMO-OWL.

L'ontologie de la représentation des connaissances (*KR Ontology)* de Sowa (2000b) se base sur la sémiotique de Charles Sanders Peirce et sur la catégorisation de l'existence de Alfred North Whitehead (Breitman *et al.*, 2007). Trois grands axes

[21] IEEE Standard Upper Ontology Working Group, *Suggested Upper Merged Ontology (SUMO)*: http://www.ontologyportal.org/

structurent l'ontologie KR (voir le tableau 1.3), soit (1) les entités de catégorie *Physiques* (qui existent dans l'espace et le temps) et de catégorie *Abstraites* (qui existent mais qui ne sont pas dans l'espace et ne sont pas dans le temps.)*;* (2) les choses qui sont des *Continuants* (qui sont indépendantes du temps) ou des *Occurants* (qui évoluent, changent dans le temps); (3) les choses qui peuvent être *Indépendantes* (qui n'ont pas à être mises en relation avec d'autres choses pour exister)*, Relatives* (qui existent par rapport à quelque chose d'autre) ou *Médiationnelles* (qui existent pour mettre en relation deux choses).

Tableau 1.3
L'ontologie de KR (Tirée de Sowa, 2003)

	Physical		**Abstract**	
	Continuant	**Occurrent**	**Continuant**	**Occurrent**
Independent	Object	Process	Schema	Script
Relative	Juncture	Participation	Description	History
Mediating	Structure	Situation	Reason	Purpose

NeOn[22], le projet de la communauté européenne de développement d'une méthodologie et d'outils propres au Web sémantique, possède sa propre ontologie cadre organisée sous la forme de patrons de conception ontologique (*Ontology Design Pattern-ODP)*. Proposé par Gamma *et al.* (1999), le concept de patron de conception désigne une abstraction qui établit une structure afin d'encadrer les bonnes pratiques associées aux activités d'un domaine d'application. Les ODP sont des patrons de conception associés à la modélisation d'ontologies. La taxonomie de haut niveau des divers types d'ODP est présentée à la figure 1.19. L'ODP offre un catalogue de petites ontologies qui sont modulaires, évolutives et regroupées en domaines d'application.

---

[22] Motta, *NeOn Project*: http://www.neon-project.org/web-content/

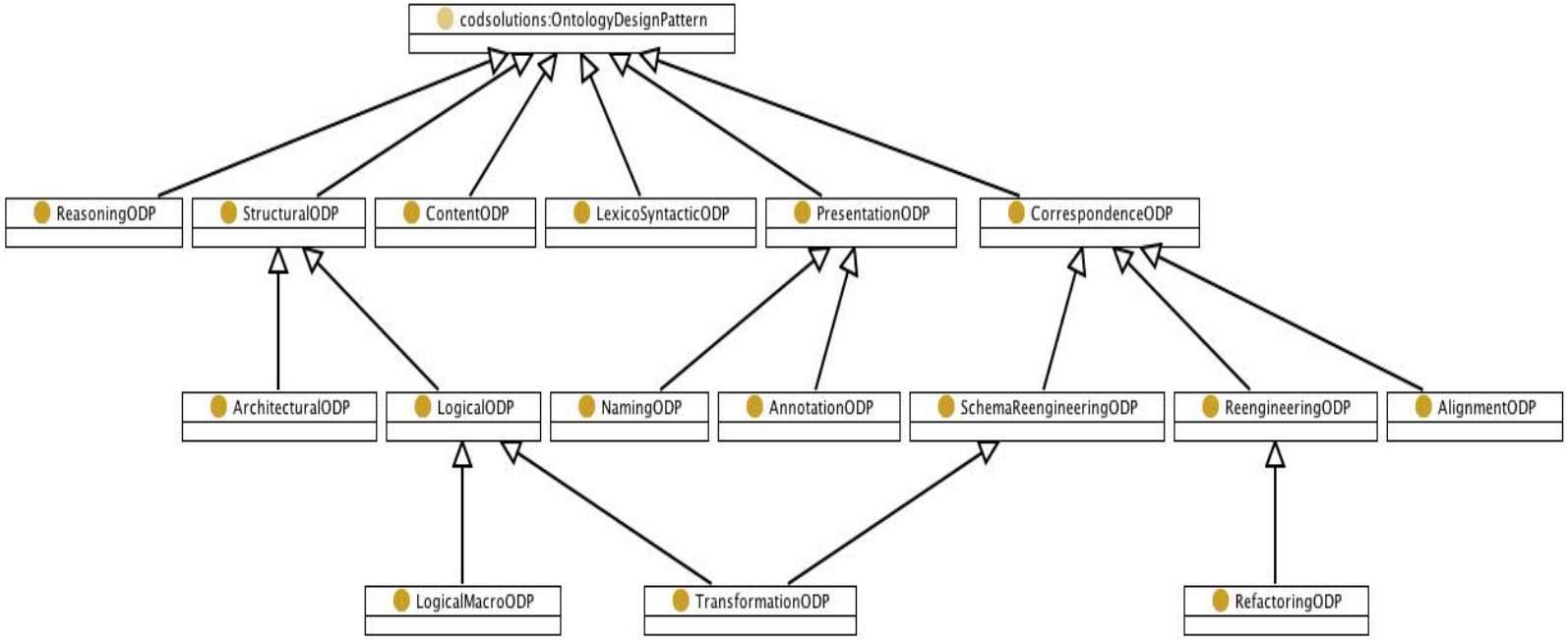


Figure 1.19: Taxonomie de patrons de conception ontologique du projet NeOn. (Tirée de ODP.org, 2008)

Le projet CYC a pour objectif de créer une ontologie cadre qui contient la définition de concepts couvrant une multitude de domaines d'application. Matuszek, Cabral, Witbrock et Deoliveira (2006) indiquent qu'il a été évalué que pour ce projet, près de 900 jours-personnes ont été employés pour construire CYC pendant plus de vingt ans et que près de 2,2 millions d'assertions, de faits et de règles ont été nécessaires pour décrire les 250 000 termes de cette ontologie. Le projet OpenCYC[23] est une version de type *OpenSource* de CYC. Une version dans le formalisme OWL est aussi disponible dans le projet OpenCYC. Sous *Things*, on retrouve les deux branches *Intangible Things* (qui sont les choses qui ne sont pas physiques) et *Individuals* (qui concerne la collecte de toutes les choses qui ne sont pas des ensembles ou collections). Les choses peuvent être concrètes ou abstraites et inclure, entre autres, des objets physiques, des événements, des nombres, des relations et des groupes (Cycorp Inc., 2002).

1.4 Langages semi-formels de représentation de la connaissance

Un langage de degré semi-formel (Uschold et Gruninger, 1996) est un formalisme d'expression de la connaissance dont la souplesse grammaticale et sémantique rend conviviale son utilisation. Ce type de formalisme, qui fait généralement appel au format graphique, est parfois employé pendant l'étape de conception d'un système (Rumbaugh *et al.*, 1999) ou encore pour favoriser le transfert d'expertise dans les organisations (Basque *et al.*, 2008b ; Basque et Pudelko, 2010b) ou encore l'apprentissage dans des situations éducatives (Basque et Pudelko, 2010c ; Kinshuk *et al.*, 2008 ; Paquette, 2002a), ainsi que pour susciter les échanges pendant une séance de remue-méninges, ou plus simplement pour représenter graphiquement un énoncé. Point de départ pour l'utilisation d'OntoCASE, nous présentons dans les pages qui suivent quelques langages semi-formels qui pourraient éventuellement tous être intégrés à OntoCASE.

---

[23] Cycorp Inc., *Formalized Common Knowledge*: http://www.opencyc.org/

1.4.1 *MindMap*

C'est dans les années 1970 que Tony Buzan développa une technique de représentation de connaissances appelée *Mind Mapping*, qui fait appel à un langage graphique combinant du texte, des images et des lignes pour représenter une idée ou un thème (voir l'exemple de la figure 1.20). Les éléments graphiques d'une *MindMap* ne possèdent aucune sémantique particulière sinon qu'une branche signifie une ramification du concept se trouvant à la racine de la branche. C'est pourquoi ce type de représentation est dit semi-formel. Au centre du graphique, est placé le concept à développer. Chacune des branches concerne un thème à développer autour du concept central. Chacun des thèmes peut aussi se développer en sous-thèmes. Le logiciel *MindMap*[24], développé pour la conception de ce type de représentation, permet d'associer une image ou un document textuel à chaque thème et sous-thème. Le langage du *Mind Mapping* est un formalisme qui possède très peu de contraintes et d'exigences, laissant ainsi libre cours à la liberté d'expression et à la créativité. En contrepartie, le manque de sémantique formelle associé à ce type de représentation rend pratiquement impossible un traitement automatisé par un système informatique.

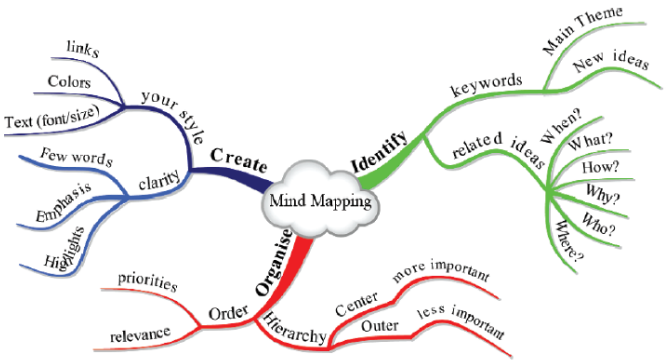


Figure 1.20: Exemple de schéma *MindMap*. (Tirée de Okada *et al,* 2008, p. xi)

[24] ThinkBuzan.com, *iMindMap software*: http://www.thinkbuzan.com/uk

1.4.2 Carte conceptuelle *(Concept map)*

S'apparentant au graphe d'entités-relations de Chen (1976), la technique de la *carte conceptuelle* (*concept mapping*) développée par Joseph Novak en 1972 (Novak et Cañas, 2006) permet de représenter une hiérarchisation de concepts selon une lecture du haut vers le bas (voir l'exemple de la figure 1.21). Les concepts sont reliés les uns aux autres par des relations portant des étiquettes textuelles qui définissent les propriétés unissant les concepts. Ici encore, la sémantique du concept et de la relation n'est pas formellement définie. Un logiciel intitulé *CMapTool*[25s] a été développé sur la base de la technique de Novak à l'*Institute for Human and Machine Cognition* (IHMC).

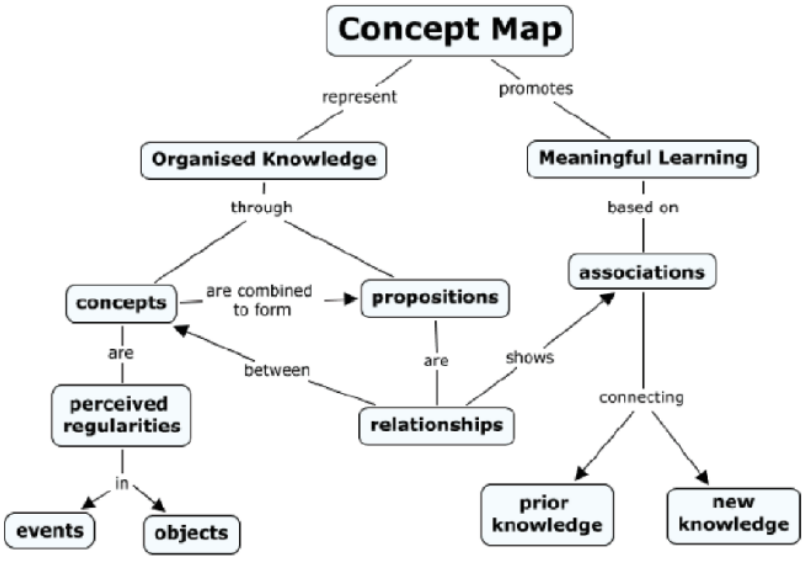


Figure 1.21: Exemple de modèle représenté dans le formalisme *Concept map.* (Tirée d'Okada *et al,* 2008, p.xi)

[25] CmapTools, *IHMC CmapTools knowledge modeling kit*: http://cmap.ihmc.us/conceptmap.html

1.4.3 *Thinking Maps*

Les *Thinking Maps* (Hyerle, 2008) font référence à différentes représentations graphiques associées chacune à une habileté cognitive différente telle que la définition dans un contexte, la discrimination d'attributs, la comparaison, la composition, la classification ou encore le raisonnement de cause à effet. La figure 1.22 présente une synthèse des représentations utilisées dans les *Thinking Maps*. On constate que pour une habileté cognitive particulière, il existe un formalisme correspondant. Plus complexe dans son utilisation que les deux formalismes précédents, le langage des *Thinking Maps* est un langage qui stimule le raisonnement formel selon Hyerle.

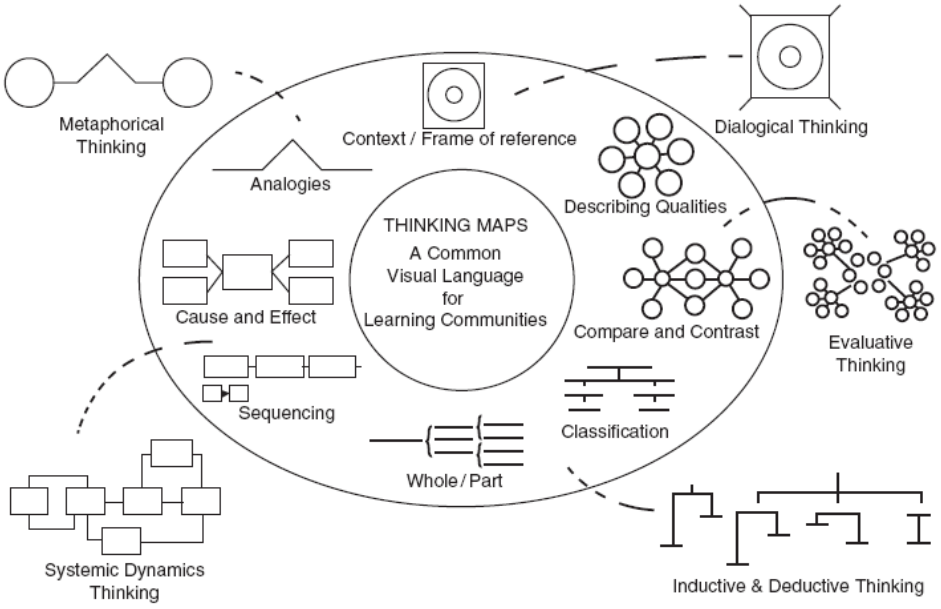


Figure 1.22: Synthèse du langage des *Thinking Maps*. (Tirée de Hyerle 2008, p. 50)

1.4.4 Modèle orienté objet

À la frontière entre le degré semi-formel et formel, le modèle orienté objet encapsule dans un objet (la *classe*) les abstractions *concept*, *attribut* et *méthode*. La classe est ainsi un objet qui possède des attributs dont la dynamique est encapsulée dans la

méthode. Certaines relations telles que la généralisation et la composition sont prédéfinies dans la sémantique du langage. Les objets peuvent aussi être reliés entre eux par des associations. Le *Unified Modeling Language* (UML) est le langage de modélisation orienté objet le plus répandu et normalisé par l'*Object Management Group (OMG UML, 1997-2006)*. Bien que spécialement conçu pour la modélisation en développement de logiciels, l'UML peut aussi être utilisé pour la représentation de connaissances (Berardi *et al.*, 2005 ; Martin, 2002 ; Rhem, 2006). L'exemple de la figure 1.23 présente le schéma d'un arbre de décision représenté par un diagramme de classe UML.

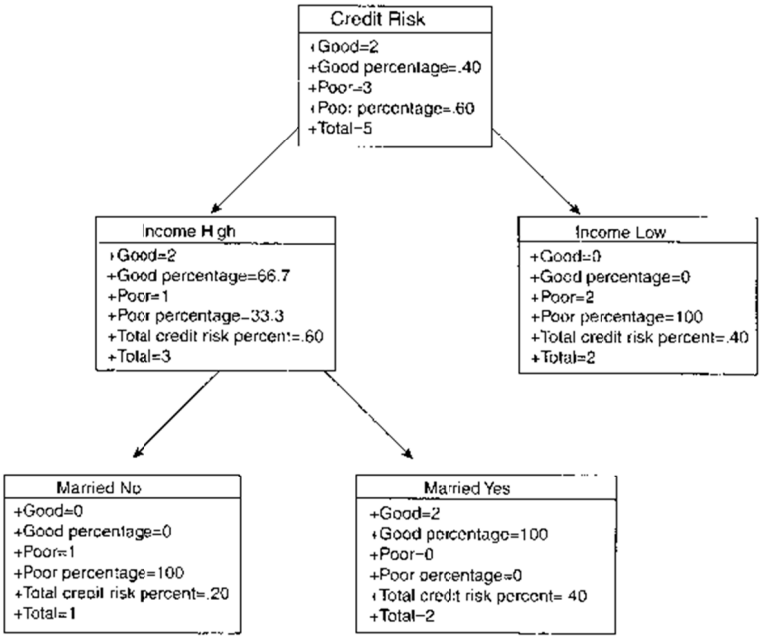


Figure 1.23: Diagramme de classes UML pour représenter un arbre de décision. (Tirée de Rhem 2006, p. 181)

1.4.5 Modélisation par objets typés

Le langage de modélisation par objets typés (MOT) développé par Paquette (2002a, 2010) est aussi un langage de représentation de connaissances de degré semi-formel. Spécialement conçu pour l'ingénierie pédagogique, ce langage a aussi été utilisé dans

des projets visant le transfert d'expertise dans les organisations (Basque *et al.*, 2008b ; Basque et Pudelko, 2010b) et l'apprentissage dans des situations de formation (Basque et Pudelko, 2010a). Ce langage permet la représentation de connaissances déclaratives, procédurales, stratégiques et factuelles ainsi que la mise en relation de ces connaissances par des relations typées (spécialisation, composition, instanciation, intrant/produit, précédence, régulation et application). Des détails sur ce langage, qui est le langage de représentation utilisé dans cette thèse, sont fournis à l'appendice A. L'utilisation d'objets typés ainsi que la propriété de représenter des connaissances selon un double niveau d'abstraction (factuel et abstrait) font de ce langage un système de représentation apte à être traité par un système de raisonnement automatique dans la mesure où la polysémie associée à certaines primitives est levée. À titre d'exemple de modèle, la figure 1.24 présente le modèle descriptif (le métamodèle) du langage MOT en langage MOT.

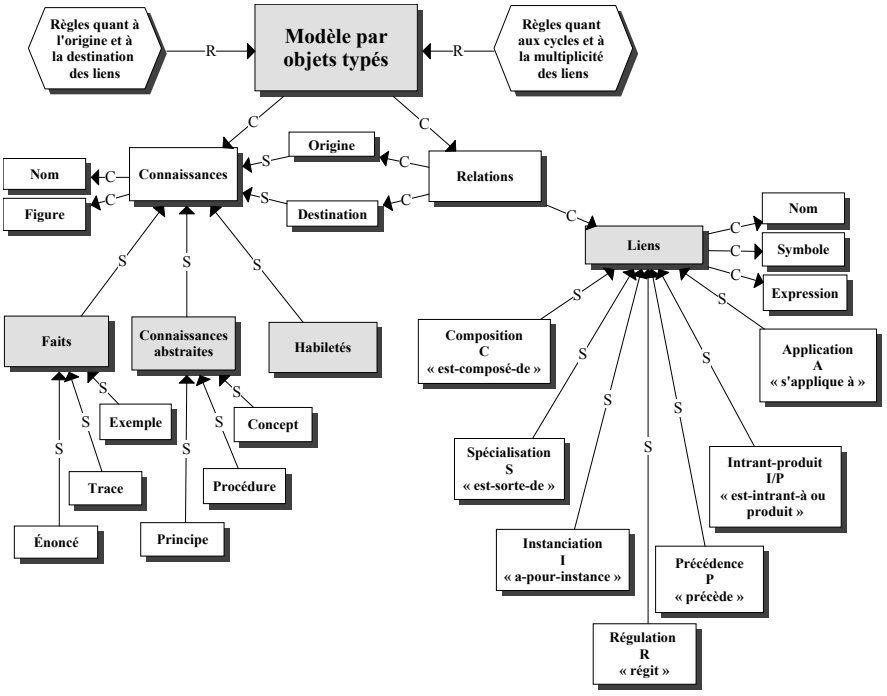


Figure 1.24: Métamodèle du langage MOT présenté dans le langage MOT. (Tirée de Paquette 2002, p. 73)

1.5 Langages formels de représentation de la connaissance

Un langage de degré formel (Uschold et Gruninger, 1996) est un formalisme dont la sémantique sans ambiguïté autorise un traitement automatisé du modèle exprimé dans ce langage. Pour Chein et Mugnier (2008), un système de représentation formel doit :

- comporter une sémantique formelle;
- posséder un fondement logique;
- permettre de représenter la connaissance de manière structurée;
- posséder de bonnes capacités computationnelles et
- permettre à un utilisateur de comprendre et avoir le contrôle de chacune des étapes de construction d'une base de connaissances.

Dans les pages qui suivent, nous présentons quelques langages formels envisagés pour servir de langages cibles à OntoCASE.

1.5.1 Taxonomie et le thesaurus

Dans le contexte de la théorie de l'information, Daconta *et al* (2003, p.146) définissent une taxonomie en une classification hiérarchique de termes selon le principe parent-enfant par l'utilisation de la relation de généralisation (*is-a, type-of*)[26]. Par exemple, si l'on considère un fait **a** comme étant de type A et que le type A est un sous-type de B, alors il est permis de conclure que **a** est aussi un fait de type B.

Le consortium ANSI/NISO (2005) définit un thésaurus comme suit:

> *"A controlled vocabulary in a known order and structured so that equivalence, homographic, hierarchical, and associative relationship among terms are displayed clearly and identified by standardized relationship indicator that are:(a) to facilitate retrieval of document and (b) to facilitate consistency in the*

[26] *the classification of information entities in the form of a hierarchy, according to the presumed relationship of a real-world that they represent*

> *indexing of written or otherwise recorded documents and other items, mainly for post coordinate information storage and retrieval systems."*

Ainsi, en plus de posséder une structure de type taxonomique, le thésaurus permet de relier les termes qui le composent par des relations de synonymie (équivalence) et des relations associatives.

La taxonomie et le thésaurus sont des types d'ontologies. Ce sont des ontologies restreintes toutefois, utilisant les propriétés de subsomption (pour la taxonomie et le thésaurus), d'équivalence et d'association (pour le thésaurus), mais non les propriétés d'axiomatisation, de quantification et de cardinalité propres aux ontologies.

1.5.2 Langage de descriptions pour la représentation de connaissances

La logique des descriptions (DL) est un système de représentation des connaissances qui est un sous-ensemble de la logique du premier ordre. Représenter les connaissances d'un domaine en DL implique un classement de celles-ci selon un plan *terminologique* et un plan *assertionnel* (Gašević *et al.*, 2006d, p.25). Dans une base de connaissances, la terminologie est définie dans la *TBox*. La *ABox*, elle, permet la définition des assertions selon le vocabulaire défini dans la TBox. Le vocabulaire se compose de *concepts* et de *rôles.* Les concepts servent à définir un ensemble d'*individus*, alors que les *rôles,* eux, servent à définir les relations binaires entre les individus. Dans la *TBox*, les concepts et les rôles sont représentés de façon *atomique* (le nom des concepts et des rôles) ou de façon *complexe* (termes de concepts et de rôles). Le tableau 1.4 présente l'exemple d'une base de connaissances exprimée en DL.

Tableau 1.4
Exemple d'une base de connaissances en DL (inspiré de Baader *et al.*, 2004)

Représentation en DL	Signification
La terminologie du domaine (TBox)	
Femme ≡ Personne ⊓ Féminin Homme ≡ Personne ⊓ ¬Femme Mère ≡ Femme ⊓ ∃aUnEnfant.Personne	Une Femme est une Personne de sexe féminin Un Homme est une personne qui n'est pas une Femme Une Mère est une Femme pour laquelle il existe une personne qui est son enfant.
Les assertions (ABox)	
Femme(MARIE) Mère(JULIE) aUnEnfant(JULIE, MARIE)	Marie est une Femme Julie est une Mère Julie a un enfant du nom de Marie

Point optimum entre expressivité et décidabilité, la DL est utilisée en tant que langage abstrait pour la représentation ontologique d'un domaine. Le tableau 1.5 présente les constructeurs en association avec les attributs[27] langagiers de la DL.

Tableau 1.5
Constructeurs de la DL et leur correspondance langagière (tirée de Gómez-Pérez *et al.*, 2003, p. 17)

Construct	Syntax	Language
Concept	A	FL_0
Role name	R	FL_0
Intersection	$C \cap D$	FL_0
Value restriction	$\forall R.C$	FL_0
Limited existential quantification	$\exists R$	FL_0
Top or Universal	$\imath N$	FL_0
Bottom	$\perp$	FL_0
Atomic negation	$\neg A$	FL^-
Negation	$\neg C$	AL
Union	$C \cup D$	U
Existential restriction	$\exists R.C$	E
Number restrictions	$(\geq n R)$ $(\leq n R)$	N
Nominals	$\{a_1 \ldots a_n\}$	O
Role hierarchy	$R \subseteq S$	H
Inverse role	R^-	I
Qualified number restriction	$(\geq n R.C)$ $(\leq n R.C)$	Q

S

[27] Par exemple, à l'OWL-DL est associé le DL-SHOIN Horrocks, *OWL: A Description Logic Based Ontology Language*

1.5.3 Graphe conceptuel

Inspiré des travaux de Pierce, c'est en 1976 que Sowa met au point une représentation graphique de la logique du premier ordre : le *Conceptual Graph (CG)* (Sowa, 1976). À la base du vocabulaire graphique du CG, on retrouve le rectangle pour représenter un *concept*, l'ellipse pour représenter la *propriété* qui unit les concepts et le *vecteur* pour représenter le flux de lecture du graphe. La représentation de l'*instance* d'un concept est assurée par l'introduction du délimiteur deux-points ':' dans la construction de l'étiquette d'un concept. À l'exemple de la figure 1.25, la représentation `Child:Paul` indique que *Paul est un enfant (Child)*. L'ellipse `possess` indique que *Paul possède (possess) une voiture (Car)*. Les chiffres posées sur les vecteurs sert à guider l'ordre de lecture des éléments du graphe. Ainsi, autour de la propriété `play3`, nous pouvons lire que « il y a *Paul est une personne qui joue (play) avec sa(possess) petite (attr Size :Small) voiture.*

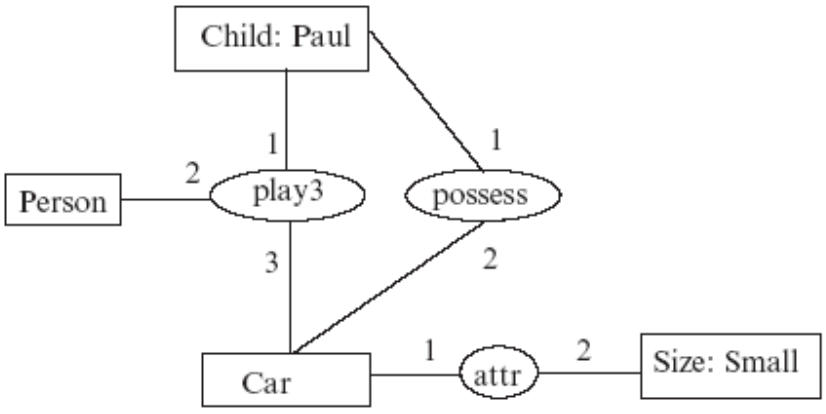


Figure 1.25: Illustration d'un graphe conceptuel tel que proposé par Sowa. (Tirée de Chein et Mugnier, 2008, p. 26.)

1.5.4 Langage ontologique

Le concept d'ontologie en est un complexe que nous avons abordé aux sections 1.2 et 1.3. Dans la présente section, nous nous intéressons à des aspects tels que l'historique de l'évolution des langages ontologiques et leurs particularités.

La figure 1.26 indique que c'est au début des années 1980 qu'est apparue la première implantation d'un langage ontologique. Depuis lors, une variété d'implantations rend accessible une multitude de langages comportant chacun leurs particularités propres.

Adapté de Gómez-Pérez *et al.* (2003, p. 286), le tableau 1.6 présente un bilan comparatif, datant de janvier 2003, des principales particularités de plusieurs langages ontologiques. Dans les cellules, un '+' indique que le langage possède la une particularité alors qu'un '-' indique que la particularité n'appartient pas au langage, et finalement, un 'W' indique que la particularité peut être réalisé de façon détournée (« *workaround* »). On constate que les langages *OCML*, *Ontolingua*, et *LOOM* semblent particulièrement intéressants, car ils possèdent de nombreuses particularités. Certaines d'entre elles permettent la représentation de connaissances procédurales et la représentation de règles. En ce sens, l'OWL est plus limité, d'autant plus qu'aucun mécanisme de détournement (« *workaround* ») ne semble exister pour la représentation de connaissances procédurales. Cependant, si on le compare au RDF(S), qui est une alternative à OWL pour le Web sémantique, OWL offre une meilleure expressivité, notamment pour l'expression des restrictions dans la taxonomie de concepts. Combiné à SWRL, OWL permet la représentation de connaissances à base de règles.

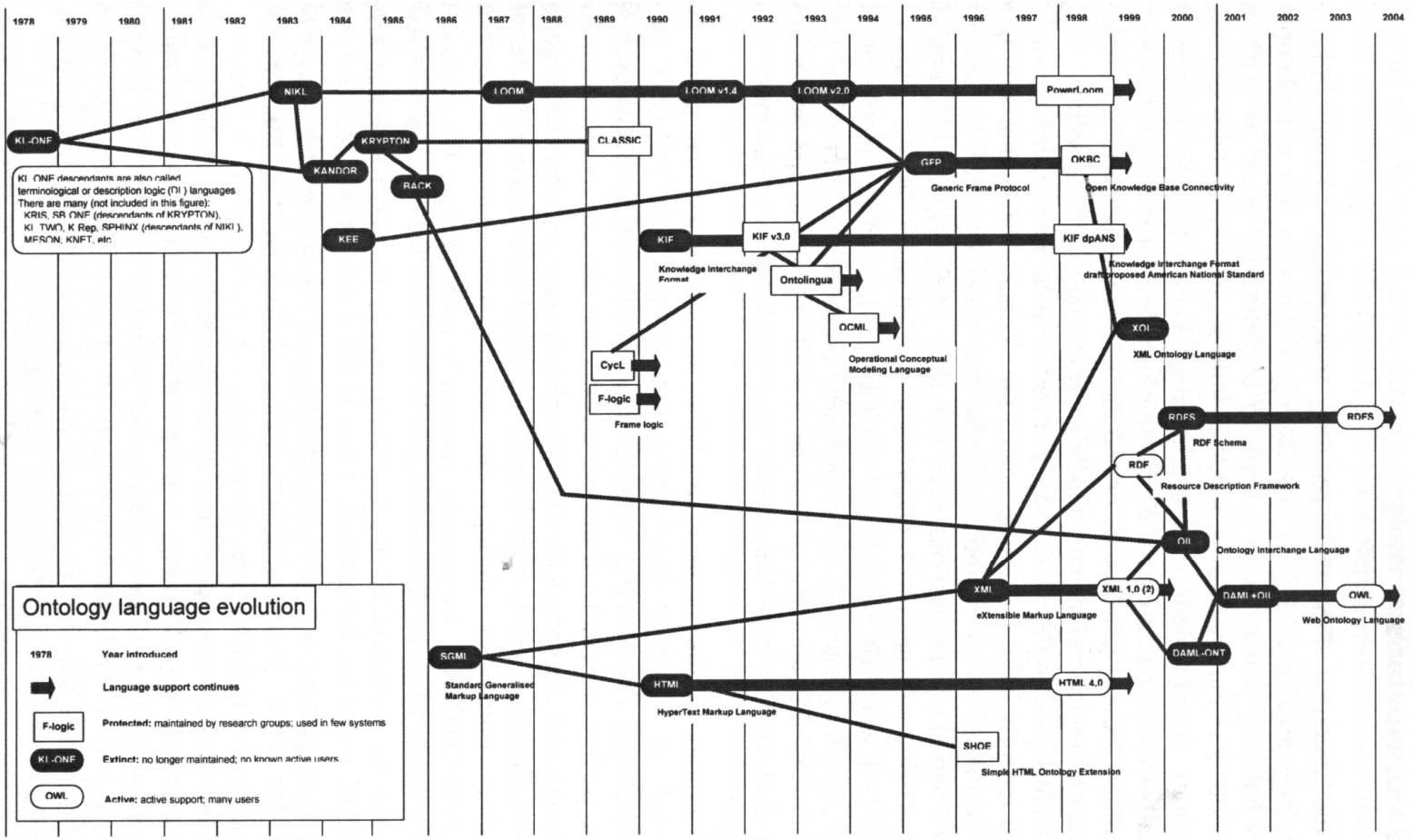


Figure 1.26: Évolution des langages ontologiques. (Tirée de Calero *et al*., 2006, p. 37)

Tableau 1.6
Sommaire des caractéristiques des langages ontologiques (adapté de Gómez-Pérez *et al.*, 2003, p. 286)

	Ontolingua	LOOM	OKBC	OCML	FLogic	SHOE	XOL	RDF(S)	OIL	DAML+OIL	OWL
CONCEPTS											
Attributes											
Instance attributes	+	+	+	+	+	+	+	+	+	+	+
Class attributes	+	-	+	+	+	-	+	W	-	W	W
Facets											
Type constraints	+	+	+	+	+	+	+	+	+	+	+
Cardinality constraints	+	+	+	+	W	-	+	-	+	+	+
Procedural knowledge	-	+	-	+	-	-	-	-	-	-	-
CONCEPT TAXONOMIES											
Subclass-Of	+	+	+	+	+	+	+	+	+	+	+
Disjoint-Decomposition	+	+	-	+	W	-	-	-	+	+	+
Exhaustive-Decomposition	+	-	-	W	W	-	-	-	W	W	W
Partition	+	+	-	+	W	-	-	-	+	+	W
RELATIONS											
Binary relations	+	+	+	+	+	+	+	+	+	+	+
n-ary relations	+	+	W	+	W	+	W	W	W	W	W
Relation hierarchies	+	+	-	+	W	-	-	+	+	+	+
Integrity constraints	+	+	-	+	+	-	-	-	-	-	-
FUNCTIONS											
Binary functions	+	+	W	+	+	-	W	-	+	+	+
n-ary functions	+	+	-	+	+	-	-	-	-	-	-
OTHER COMPONENTS											
Formal Axioms	+	+	-	+	+	-	-	-	-	-	-
Instances	+	+	+	+	+	+	+	+	+	+	+
Procedures	+	+	-	+	-	-	-	-	-	-	-
Rules	-	+	-	+	-	+	-	-	-	-	swrl

1.5.5 Langage à base de règles

Les langages de représentation de connaissances, tels que OWL, sont conçus pour décrire un domaine de connaissances par la construction de classes pour représenter les concepts du domaine, d'individus pour en représenter les faits et de propriétés pour représenter les relations entre les faits du domaine. En contrepartie, les langages à base de règles sont conçus pour générer de nouvelles assertions à partir de faits préalablement établies (Breitman *et al.*, 2007). Les langages à base de règles, tels que le PROLOG, sont développés selon les principes de la programmation logique (Lloyd, 1987).

Dans la représentation de connaissances, une sous-classe de la DL, la *Description Logical Programming* (DLP) (Grosof *et al.*, 2003) permet l'addition de *règles* en complément à la TBox et à la ABox. L'expression d'une règle dans sa forme logique s'écrit comme suit: $C \Rightarrow D$ (« si un individu est une instance de C, alors l'individu appartient aussi à D où C est un *antécédent* et *D* est le *conséquent*») ou encore dans sa forme informatique elle s'écrit comme suit : « IF … THEN… » .

Au moment d'écrire ces lignes, le *Semantic Web Rule Language* (SWRL) en est à l'étape de proposition au W3C. Ce langage à base de règles est un sous-ensemble de RuleML[28]. Spécialement conçu pour s'intégrer à une base de connaissances en OWL, le langage SWRL propose un formalisme à base d'implications entre un groupe d'antécédents et une conclusion, qui sont en fait des agrégations d'atomes (voir le métamodèle de la figure 1.27 tel que proposé par l'*Ontology Definition Model* (ODM[29])). Un atome se compose d'une liste de termes et/ou d'une composition de prédicats. En voici quelques exemples : *sameAs(x,y)*, où x et y sont des variables ou

---

[28] Boley, *The Rule Markup Initiative*: http://ruleml.org/

[29] OMG ODM, *Ontology Definition Metamodel: OMG Adopted Specification*: http://www.omg.org/spec/ODM/1.0/Beta2/PDF/

des individus OWL; *différentFrom(x,y)*, où x et y sont des variables ou des individus OWL et *builtIn(r, x, ...)* où r est un prédicat et x ainsi que chaque élément de la suite sont des variables ou des données OWL. Le détail de la discussion concernant la prédicat de type *builtIn* est présenté à la section 3.1.8.2 p. 154 et à la section 3.2.5, p 168.

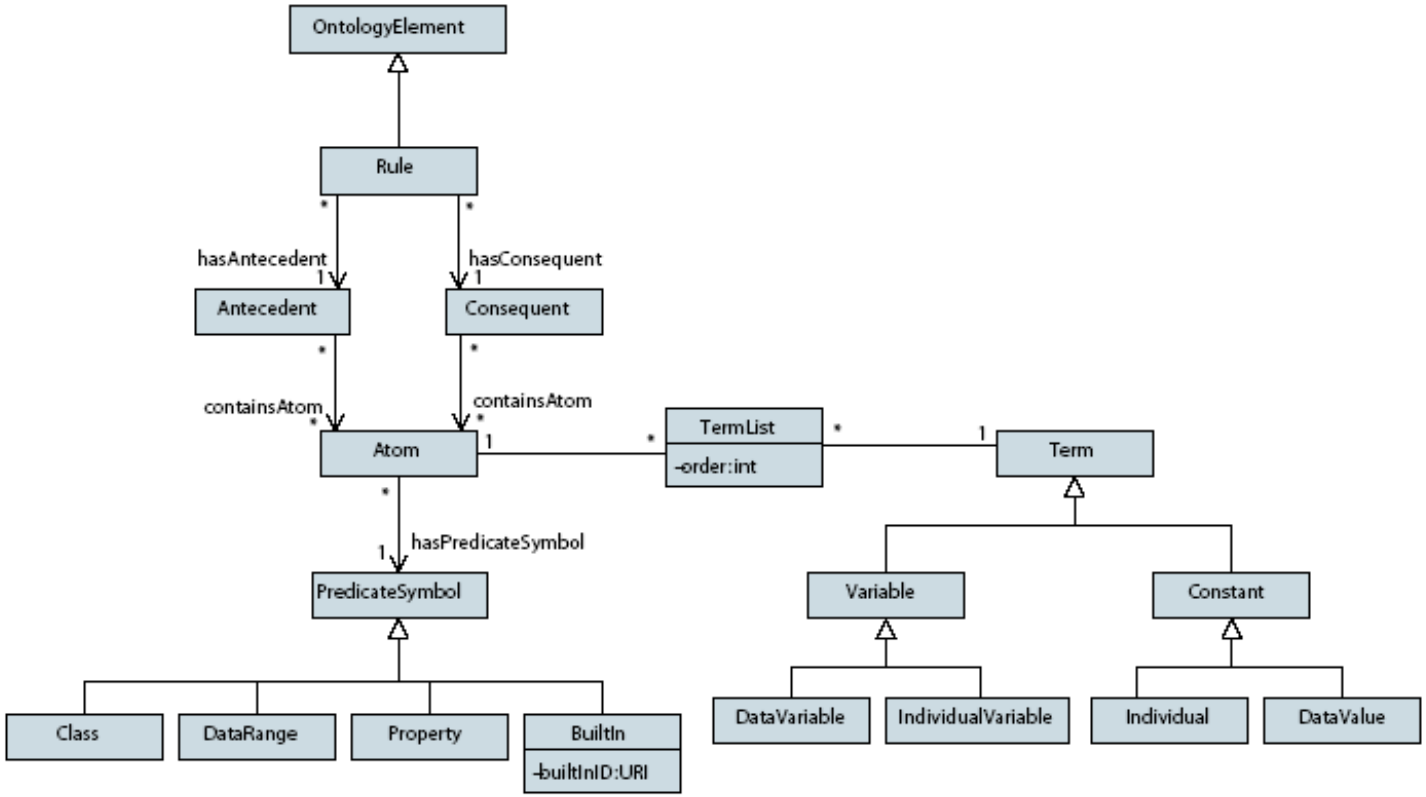


Figure 1.27: Métamodèle UML du *Semantic Web Rule Language.* (Tirée de Brockmans *et al.*, 2006,p. 9).

1.5.6 Business Process Modeling Notation

Le *Business Process Modeling Notation* (BPMN[30]) n'est pas un langage ontologique, mais un langage de modélisation de degré formel orienté vers la représentation de connaissances *procédurales*. Chaque élément de la notation (voir la figure 1.28) possède une sémantique non ambigüe qui permet un traitement par un système automatique.

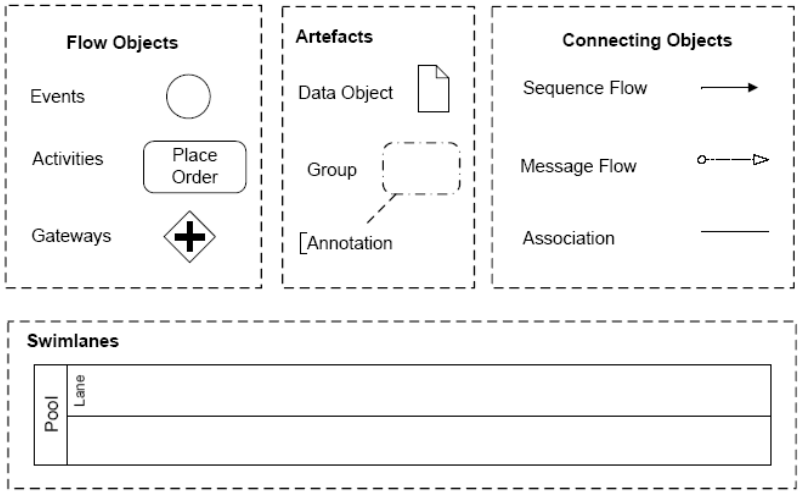


Figure 1.28: Éléments d'un modèle BPMN. (Tirée de Weske, 2007, p. 209)

La computationnalité de ce langage réside en une transformation du modèle sous la forme d'un réseau de Petri, où chacun des nœuds du réseau représente soit une transition (le carré), soit une opération de contrôle du flux (le cercle), et où le vecteur représente le flux de l'information. Dans l'exemple de la figure 1.29, une solution possible de l'exécution de la transition *t2* résulte de l'exécution de la transition *t1* et *t4*.

[30] OMG BPMN, *Object Management Group/Business Process Management Initiative*: http://www.bpmn.org/

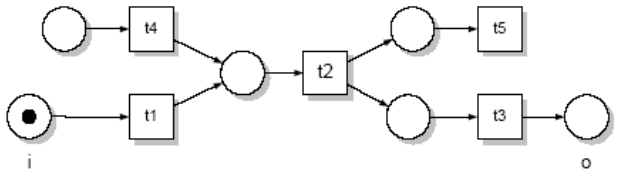


Figure 1.29: Exemple de réseau de Pétri pour représenter un *workflow.* (Tirée de Weske, 2007, p. 271)

1.6 Transformation de modèles par la technique de métamodélisation

La relation "méta" est utilisée en représentation pour relier deux entités d'un même sujet, dont les niveaux d'abstraction diffèrent. Par exemple, une métaconnaissance est une connaissance à propos d'une connaissance (Pitrat, 1990). De même, un métamodèle est un modèle à propos d'un modèle. La relation 'à-propos-de' est une relation de définition entre les objets reliés. Le métaobjet *définit* l'objet. Ainsi, l'objet est une *instance* du métaobjet. En ce sens, un métamodèle est la définition des symboles, de la syntaxe et de la sémantique des modèles qui servent à représenter un domaine d'application (Kadima, 2005).

Dans les pages qui suivent, nous revenons sur les concepts d'abstraction et de couches d'abstraction et présentons quelques applications liées à ces concepts. Dans le développement d'OntoCASE, les concepts abstraction et couches d'abstraction sont utilisés pour la définition du cadre théorique et pour l'implantation informatique du processus de transformation de modèle.

1.6.1 Abstraction et couches d'abstraction

La métamodélisation est utilisée en développement logiciel (Gašević *et al.*, 2006b ; Kadima, 2005 ; OMG MDA, 2007 ; Steinberg *et al.*, 2008) et en développement d'ontologies (Đurić, 2004 ; Đurić *et al.*, 2005b ; Đurić *et al.*, 2005a ; Gašević *et al.*, 2006d, 2006a ; Gerbé et Mineau, 2008 ; IBM, 2007 ; OMG ODM, 2007 ; Yang, 2006). Cette technique permet la représentation d'un domaine sous la forme de

modèles de données du domaine, de *modèles du domaine*, de *langage de modélisation* ainsi que de *métalangages* (voir la figure 1.30). Les données du domaine de la couche M0 sont des représentations factuelles d'un domaine du discours, dont les modèles du domaine de la couche M1 constituent l'abstraction. Le modèle de la couche M2, aussi nommé métamodèle, définit le langage de modélisation utilisé pour modéliser le domaine. Le modèle de la couche M3, nommé métamétamodèle, définit le langage de la couche M2. Le modèle de la couche M3 est réflexif, c'est-à-dire qu'il a la propriété de se définir lui-même.

Pour un métamétamodèle donné, l'architecture de métamodélisation assure une interopérabilité entre les langages de modélisation (modèles de la couche M2). Cette interopérabilité soutient la traduction d'un modèle de domaine représenté selon un langage source vers un modèle de domaine représenté selon un langage cible. La transformation de modèles de domaine entre langages est l'activité principale que soutient cette architecture.

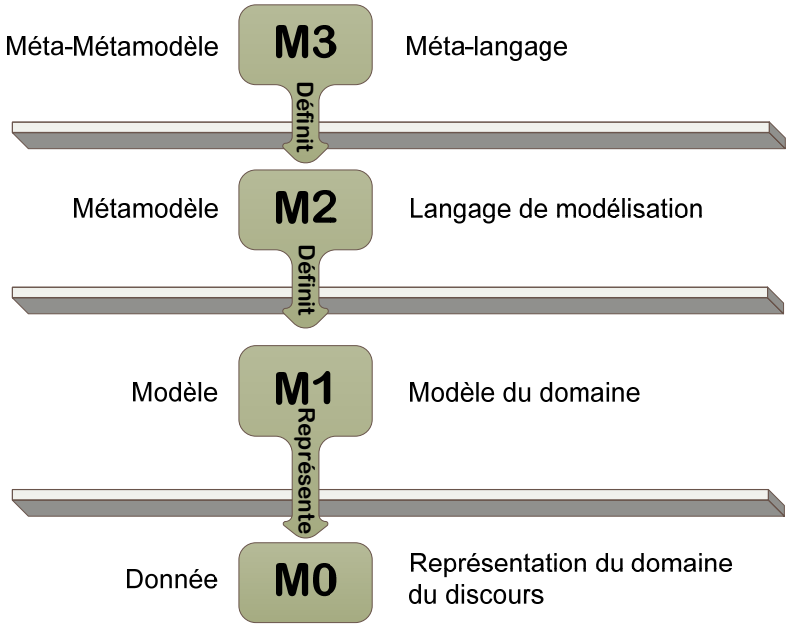


Figure 1.30: Couches d'abstraction de la métamodélisation.

1.6.2 Espace de modélisation

L'espace de modélisation (EM) (Gašević *et al.*, 2006e) est une sttructure de modélisation réalisée à partir d'un métalangage particulier. La figure 1.31 présente un exemple de différents espaces de modélisation.

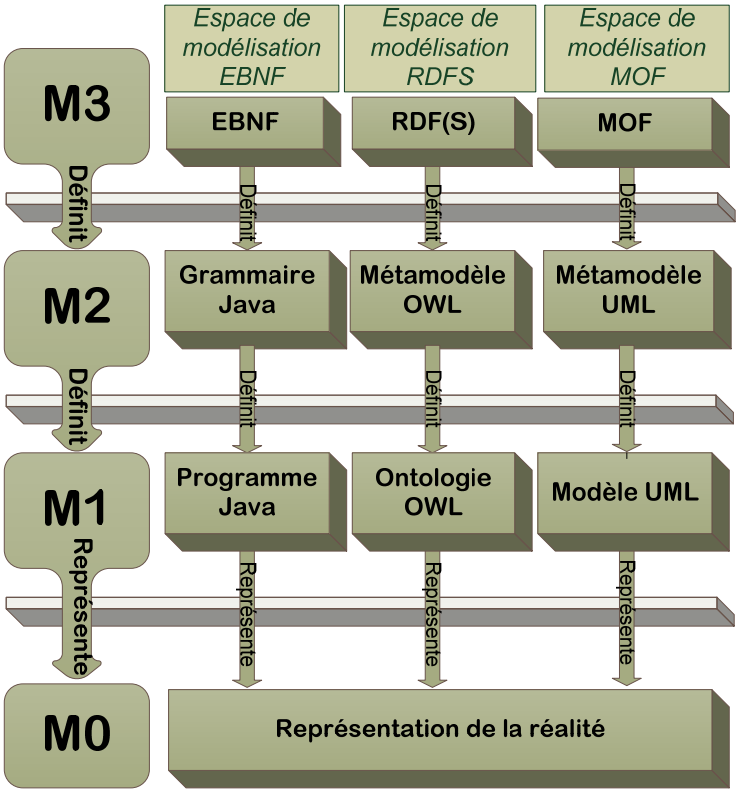


Figure 1.31: Espaces de modélisation EBNF, RDFS et MOF adaptée de Gašević *et al.* (2006e p. 132).

Le premier EM présenté est l'*Extended Backus-Naur Form* (EBNF[31]), modèle qui définit la grammaire du langage *Java.* Celui-ci sert de modèle aux programmes *Java* qui, eux-mêmes, sont des modèles pour décrire une représentation de la réalité. Un autre EM, le RDFS[32], est le métalangage de *l'Ontology Web Language* (OWL[33]) qui

[31] ISO-14977, *The standard metalanguage Extended BNF*: http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf

[32] Beckett et McBride, *RDF/XML Syntax Specification (Revised)*: http://www.w3.org/TR/rdf-syntax-grammar/

permet de construire des ontologies, autres modèles représentant la réalité. À la figure 1.31, le MOF *Meta Object Facility* (MOF[34]) sert d'EM pour le métamodèle décrivant l'*Unified Modeling Language* (UML[35]) qui sert à la construction des modèles UML. Pour réaliser la transformation entre modèles, il importe que chaque métamodèle (ou langage de modélisation) soit dans le même espace de modélisation. Si ce n'est pas le cas, une importation dans l'EM, supportant la transformation, doit être réalisée.

1.6.3 Processus de transformation de modèles

La transformation d'un modèle est l'une des applications importantes de l'*Architecture Conduite par les Modèles* (ACM[36]) de l'OMG-MDA (2007). Elle vise la production automatique d'un modèle cible à partir d'un modèle source par l'exécution d'un ensemble de règles de transformation qui régissent le déroulement du processus de transformation. Ce *processus de transformation* (schématisé à la figure 1.32) produit un modèle cible à partir des informations contenues dans le modèle source. Les règles d'utilisation des informations du modèle source sont dictées par le métamodèle (couche M2) du modèle source. Le métamodèle source est aussi un intrant au processus de transformation. Les règles d'interprétation des informations du modèle cible sont contenues dans le métamodèle cible. Le métamodèle cible est aussi un intrant du processus de transformation. Remarquons que, même si les règles de représentation changent d'un modèle à l'autre, les deux modèles servent à représenter le même objet de la réalité.

---

[33] W3C OWL, *Ontology Web Language*: http://www.w3.org/2004/OWL/

[34] OMG MOF, *OMG's MetaObject Facility*: http://www.omg.org/mof

[35] OMG UML, *UML® Resource Page*: http://www.uml.org/

[36] Traduction de *Model Driven Architecture (MDA);* OMG MDA, *OMG Model Driven Architecture*: http://www.omg.org/mda/

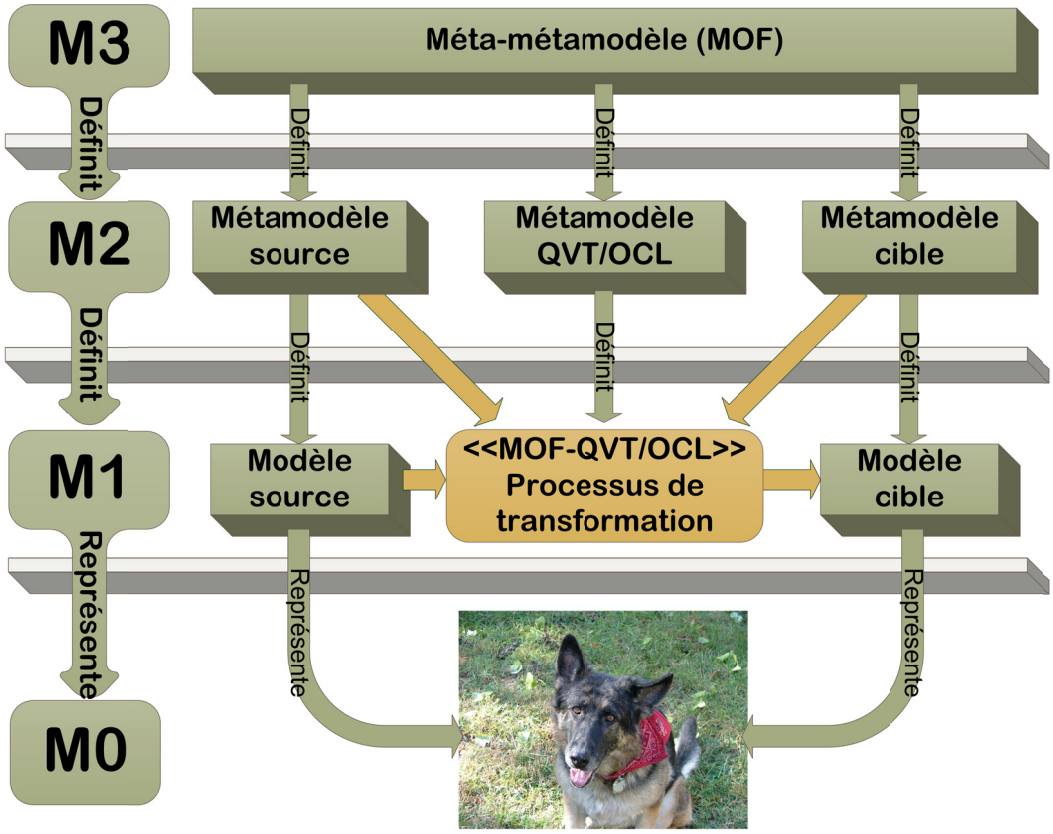


Figure 1.32: Processus de transformation de modèles dans l'OMG-MDA.

Du point de vue technologique, les outils comme le *MOF-Query/View/Transformation* (MOF-QVT[37]) pour la *recherche* dans les modèles, la production de *vues* dans les métamodèles et la *transformation* de modèles ainsi que *l'Object Constraint Language* (OCL[38]) qui permet l'expression formelle de contraintes sur des éléments de modèle UML, sont utilisés pour conduire des processus de transformation entre des modèles, pourvu que ces processus soient dans l'espace de modélisation MOF.

---

[37]OMG MOF-QVT, *Documents associated with Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0* http://www.omg.org/spec/QVT/1.0/

[38] OMG OCL, *Object Constraint Language Specification, version 2.0*: http://www.omg.org/technology/documents/formal/ocl.htm

1.6.4 Principe de transformation parallèle

Le principe de transformation parallèle a comme pré-condition que le modèle source et le modèle cible possèdent une structure de métamodèle similaire permettant la conception de règles de transformation de type "un à un". Par exemple (voir la figure 1.33), une relation R et une entité E du modèle source sont directement transposées en relation R' et en entité E' dans le modèle cible. Paquette et Rogozan (2006) utilisent cette approche pour la conception du langage MOT-OWL, mais ils se restreignent à une sous-classe de modèles MOT proche du langage des ontologies OWL-DL.

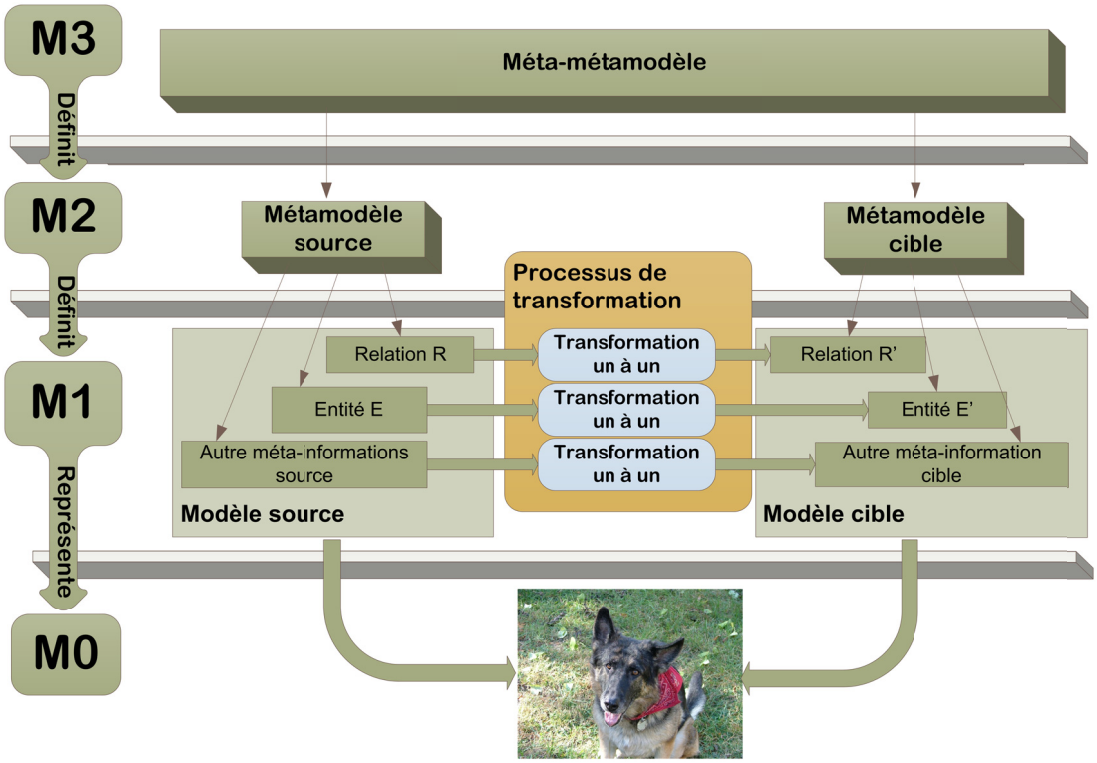


Figure 1.33: Processus de transformation parallèle.

Une limite importante de la transformation parallèle est que le processus de transformation est réalisé sans tenir compte de la sémantique des langages en cause dans la transformation. En comparaison avec la traduction en linguistique, la

transformation parallèle est un processus qui produit une traduction en mode mot-à-mot sans considérer le sens des mots qui sont traduits, limitant ainsi les capacités de désambiguïsation automatique

1.6.5 Principe de la transformation orthogonale

Le but spécifique de la transformation orthogonale est de prendre en compte la grammaire du langage source afin de lever les ambiguïtés sémantiques contenues dans le modèle source. En plus du modèle source à transformer, le processus de transformation orthogonale a, en entrée, une représentation du métamodèle source ainsi qu'un modèle de transformation qui comporte un ensemble de règles permettant au processus de transformation de fonctionner adéquatement (voir la figure 1.34). Notons que le modèle source est exprimé dans la même syntaxe que le métamodèle cible.

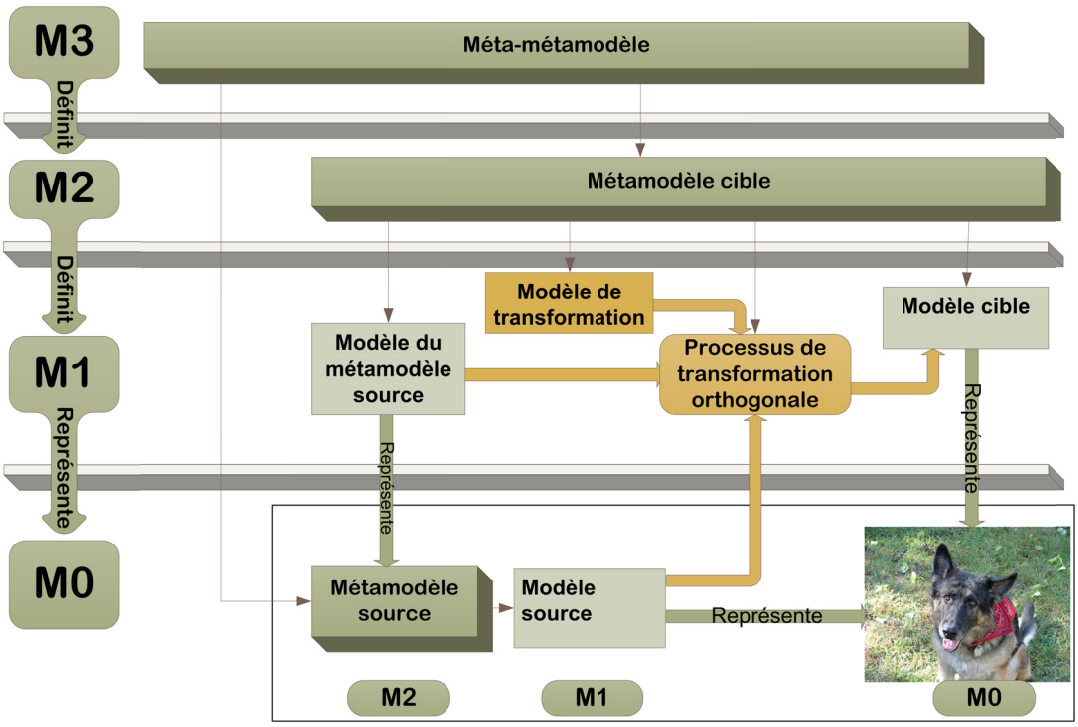


Figure 1.34: Processus de transformation orthogonale.

Schématisé à la figure 1.35, le processus de transformation orthogonale est un processus contrôlé par le *modèle de transformation.* Ce modèle, composé d'un *modèle de désambiguïsation* et d'un *modèle de référence*, contient un ensemble de règles qui contrôlent les sous-processus de *désambiguïsation* et de *conversion*. Fort d'une représentation de la grammaire du modèle source (*le modèle du métamodèle source*), le sous-processus de désambiguïsation interprète les éléments contenus dans le modèle source et les classent en fonction des concepts contenus dans le modèle de référence. Nous verrons un peu plus loin (voir la section 3.1.3.2, p.129) les détails concernant le modèle de référence. Nous dirons alors que les éléments du modèle source sont désambiguïsés.

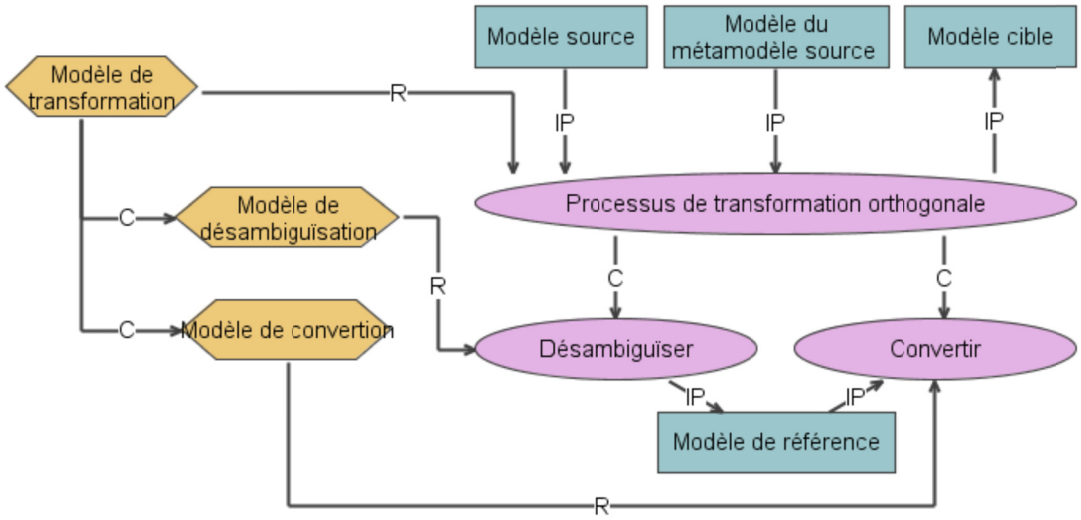


Figure 1.35: Modèle du processus de transformation orthogonale.

Après l'exécution de la désambiguïsation, le processus de transformation orthogonale exécute l'étape de conversion des éléments désambiguïsés contenus dans le modèle de référence afin de produire le modèle cible.

1.6.6 Transformation d'un modèle semi-formel en ontologie

Compte tenu de l'importance de la formalisation des connaissances dans le processus de construction d'une ontologie, quelques auteurs proposent des solutions pour formaliser un modèle semi-formel en ontologie.

1.6.6.1 Transformation d'un modèle de type *Concept Map*

La carte conceptuelle (voir la section 1.4.2) est un système de représentation composé de nœuds pour représenter des entités et de liens pour représenter des relations entre les entités. Pour la conception d'ontologies OWL à partir de ce type de représentation semi-formelle, Eskridge, Hayes et Hoffman (2006) ont conçu l'application *CmapTools Ontology Editor* (COE). Castro *et al.* (2006) utilisent la carte conceptuelle pour l'élicitation semi-formelle de l'expertise avant de la formaliser en ontologie OWL au moyen de COE. Quant à Park et Hunting (2003), plutôt orientés vers le « *TopicMaps* », ils proposent un ensemble de méthodes, techniques et technologies pour intégrer le référencement sémantique d'objets du Web par l'intermédiaire d'un modèle de type *ConceptMap* de degré semi-formel.

1.6.6.2 Architecture conduite par les modèles et le développement d'ontologies

L'*Ontology Definition Metamodel* (ODM)[39] est une application de la méthode d'architecture conduite par les modèles qui est dédiée à la production d'ontologies OWL à partir de modèles UML. La figure 1.36 présente le contexte de modélisation d'une ontologie dans la perspective de l'architecture conduite par les modèles selon la spécification de l'OMG[40]. Dans l'espace de modélisation du MOF, le métamodèle

---

[39] OMG ODM, *Ontology Definition Metamodel: OMG Adopted Specification*: http://www.omg.org/spec/ODM/1.0/Beta2/PDF/

[40] OMG MDA, *OMG Model Driven Architecture*: http://www.omg.org/mda/

ontologique (ODM[41]) définit la structure interne du document OWL dans le formalisme MOF. Un convertisseur XML/XSLT assure la traduction de l'ontologie de l'espace de modélisation MOF à l'espace de modélisation RDFS (Gašević *et al.*, 2006c, p. 218). À l'origine du processus de construction de l'ontologie, un modèle UML est produit pour représenter le domaine du discours à modéliser. Les éléments du modèle sont transformés selon l'une des méthodes de transformation décrites à la section 1.6.6.

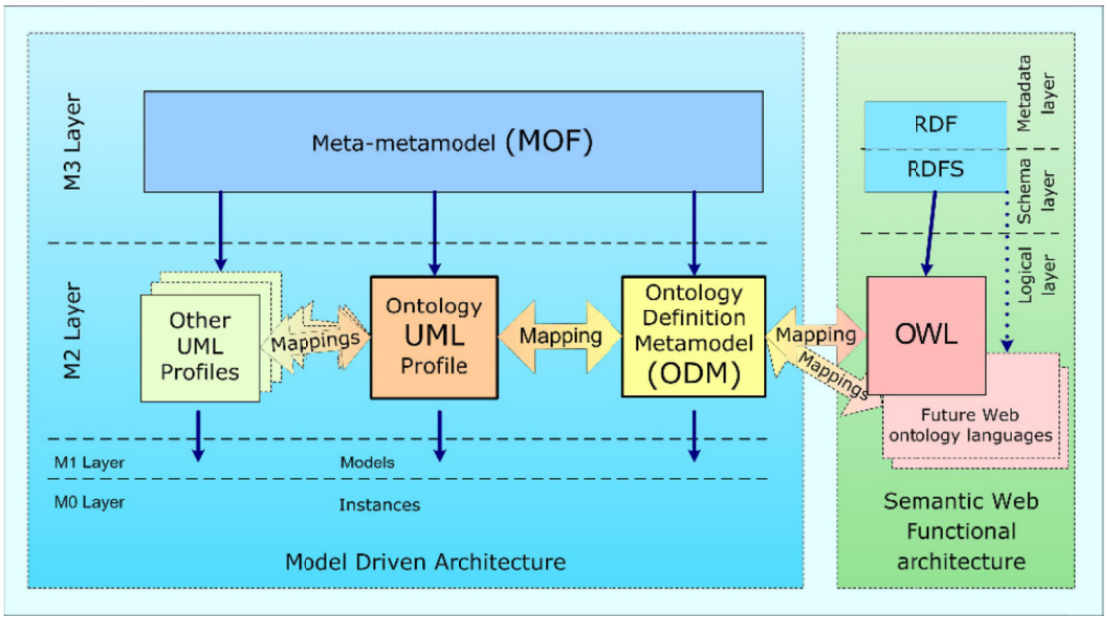


Figure 1.36 : La modélisation d'ontologie dans le contexte de l'ACM et du Web sémantique. (Tirée de Gašević *et al.*, 2006c, p. 175)

Le projet de modélisation (MDT)[42] de la communauté des développeurs d'*Eclipse* consacre ses activités à la promotion des technologies basées sur l'architecture conduite par les modèles au sein de la Communauté Eclipse en fournissant un ensemble unifié de métamodèles, d'outils et de normes de mise en œuvre. L'*EMF*

[41] Le métamodèle de l'ODM est présenté à l'appendice E.

[42] Eclipse foundation, *Eclipse Modeling Project*: http://www.eclipse.org/modeling/

Ontology Definition Metamodel (EODM[43]), qui était jusqu'en 2008[44] un sous-projet du projet de modélisation d'Eclipse, offre une représentation de l'ODM dans l'espace de modélisation MOF-EMF[45]. Ma *et al.* (2004) ont développé l'*IBM Integrated Ontology Development Toolkit* qui est une application *Eclipse* basée sur l'EODM qui permet la construction d'une ontologie OWL à partir d'un modèle UML. Notre évaluation personnelle de cette application nous a permis de constater qu'il s'agit d'une approche prometteuse de formalisation de modèles en ontologies.

1.6.6.3 Transformation de modèles UML et de graphes entités-relations

Le *Unified Modeling Language* (UML[46]) et les graphes entités-relations sont des systèmes de représentation couramment utilisés par les informaticiens. Brockmans *et al.* (2006) proposent d'utiliser l'UML pour représenter de façon « méta » les composantes ontologiques et les composantes à base de règles du vocabulaire d'un domaine. Berardi *et al.* (2005) utilisent les modèles UML, notamment les propriétés associées aux diagrammes de classes, pour réaliser des raisonnements par une traduction des modèles en logique descriptive. Quant à eux, An, Borgida et Mylopoulos (2005) proposent une méthode pour formaliser les graphes entités-relations associés à des tables de bases de données en ontologies OWL pour un accès aux tables à partir de systèmes à base de règles. Finalement, Faucher *et al.* (2008) proposent une méthode de conception d'ontologies à partir d'un diagramme de classes UML qui est annoté. Basé sur l'ODM, chaque composant du modèle est annoté en utilisant la propriété de stéréotype intrinsèque à UML. Par exemple, une classe est stéréotypée par `rdfs:Class`, la généralisation par `rdfs:subCLassOf`,

---

[43] Yang, *EMF Ontology Definition Metamodel*: http://www.eclipse.org/modeling/mdt/eodm/docs/articles/EODM_Documentation/

[44] Hussey, *MDT EODM Termination Review*: http://www.eclipse.org/project-slides/EODM%20Termination%20Review.pdf

[45] *Eclipse Graphical Modeling Framework (GMF)*: http://www.eclipse.org/modeling/gmf

[46] OMG UML, *UML® Resource Page*: http://www.uml.org/

l'association par `rdf:Property` et ainsi de suite. La transformation est assumée par le MOF QVT.

Fait étonnant, nous n'avons trouvé aucune recherche qui tente de formaliser des diagrammes d'états, de séquences ou de cas d'utilisation, ce qui le cas échéant permettrait de représenter des connaissances procédurales et stratégiques dans une ontologie.

1.7 Ingénierie des connaissances

L'ingénierie des connaissances est la discipline des sciences appliquées qui vise le développement des techniques, des méthodes et des outils nécessaires à l'élicitation, la pérennisation et la diffusion de connaissances. Plusieurs éléments de l'ingénierie des connaissances ont été utilisés pour, d'une part, structurer la démarche de conception d'OntoCASE et, d'autre part, produire un modèle pour son usage. Nous abordons dans cette section la notion de conception d'une méthodologie, puis nous faisons un tour d'horizon des concepts liés à l'élicitation des connaissances et finalement nous abordons le sujet de l'ingénierie ontologique afin d'identifier les éléments méthodologiques qui serviront à l'élaboration du volet méthodologique d'OntoCASE.

1.7.1 Structure de conception d'une méthodologie

Dans le dictionnaire Antidote, le terme *méthodologie* est défini de la manière suivante : *"l'étude des méthodes scientifiques ou techniques"* et de manière plus spécifique: *"l'ensemble des méthodes d'un domaine donné"*. Une méthodologie est donc, en quelque sorte, souvent décrite en tant que métaméthode, c'est-à-dire une méthode qui sert à définir des méthodes ou encore en tant que mégaméthode qui se compose de plusieurs méthodes. La définition standardisée par le IEEE (1990, 1996) du terme *méthodologie* va dans le même sens : « ensemble de méthodes encadré par des techniques créant une théorie générale systémique concernant la façon dont une

certaine catégorie de travail devrait être effectuée [47]». Quant au terme *méthode*, il fait « référence à un processus ou une procédure ordonné utilisé en ingénierie de produits ou de services [48]», alors qu'une *technique* est définie « en tant que procédure technique et de gestion utilisée pour atteindre un objectif donné [49]». C'est cette dernière définition qui a inspiré Gómez-Pérez *et al.* (2003) pour la conception de la méthodologie METHONTOLOGY.

Toujours selon le IEEE, un *processus* est une fonction devant être exécutée dans le cycle de vie d'un logiciel et un processus est composé d'« activités et de fonctions devant être exécutées dans le cycle de vie d'un logiciel [50]» . Gómez-Pérez *et al* (2003), citant le IEEE (1996), définissent une *activité* en tant que « tâche constitutive d'un processus [51]» et une *tâche* en tant que « directive de travail bien définie pour un ou plusieurs membres d'un projet [52]». Ces auteurs ajoutent que des tâches connexes sont généralement regroupées pour former des activités. La figure 1.37 schématise en langage MOT le concept de méthodologie tel que décrit par Gómez-Pérez *et al.* (2003).

Dans notre projet, nous avons utilisé cette structure d'une méthodologie pour, d'une part, structurer la démarche de recherche de la thèse, et d'autre part, structurer les composants de l'objet de la thèse, soit la conception d'une *méthodologie de formalisation*.

---

[47] *[...] a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought intensive work ought be performed[...]*

[48] *[...] orderly process or procedure used in the engineering of a product or performing a service[...]*

[49] *[...] a technical and managerial procedure used to achieve a given objective[...]*

[50] *[...] function that must be performed in the software life cycle. A process is composed of activities*

[51] *[...] a constituent task of a process*

[52] *[...] a task a well defined work assignment for one or more project members. Related task are usually grouped to form activities »* (

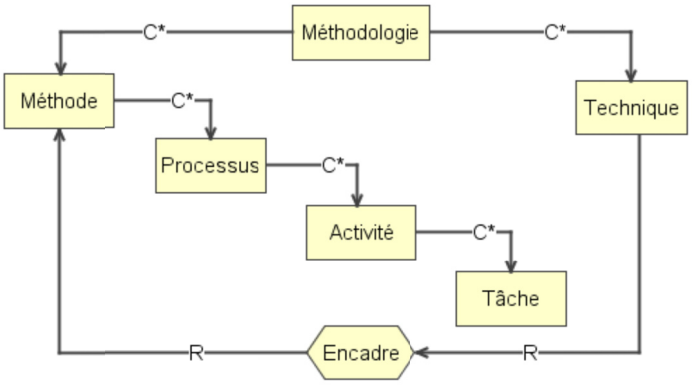


Figure 1.37: Modèle MOT de la définition d'une méthodologie. (Tirée et adaptée en MOT de Gómez-Pérez *et al.*, 2003, p. 109)

1.7.2 Élicitation des connaissances

Le processus d'*élicitation* des connaissances permet de définir le *vocabulaire* du domaine d'application, les *types* de connaissances manipulées ainsi que les *relations* qui existent entre les types de connaissances (Gerbé *et al.*, 2003). Plusieurs techniques permettent l'élicitation de ces connaissances par un ingénieur de connaissances et un ou plusieurs experts du domaine. Nous en avons déjà présenté quelques-unes aux sections 1.4 p.38 et 1.5 p.44 qui, comme on l'a vu pour certaines, font appel à la modélisation sous une forme schématique.

Ultimement, et spécialement dans la perspective de construire une ontologie, l'élicitation a pour objectif de produire les artéfacts transitoires suivants (Gómez-Pérez *et al.*, 1996):

- un dictionnaire de données (*Data Dictionary*) rassemblant l'identification des concepts, instances, attributs et leurs valeurs correspondantes;
- un arbre de classification des concepts (*Concept Classification Tree*), pour classifier les concepts;
- une table des matières (*Table of Constants)* pour la description des constantes;

- une table de faits (*Table of Instances)* et une table d'attributs (*Tables of Class attributes*) pour décrire les attributs associés aux classes et à leurs faits;
- une table des formules (*Table of Formulas*) décrivant les formules utilisées pour inférer de nouvelles valeurs numériques;
- une taxonomie des attributs (*Attribute Classification Tree*) détaillant les attributs impliqués dans des inférences spécifiques;
- une table de faits du domaine (*Table of Instances*) décrivant des instances du domaine d'application.

La structure de la représentation des connaissances est souvent influencée par la manière dont elle est élicitée. Par exemple, la méthode MASK (Ermine et Matta, 2003) est une méthode visant la capitalisation et le partage de connaissances dans les organisations, dont le cycle « vertueux » est schématisé à la figure 1.38. Le « livre de connaissances » est l'artéfact concret issu du processus d'explicitation des connaissances par les experts, qui permet le partage de la connaissance au sein de l'organisation.

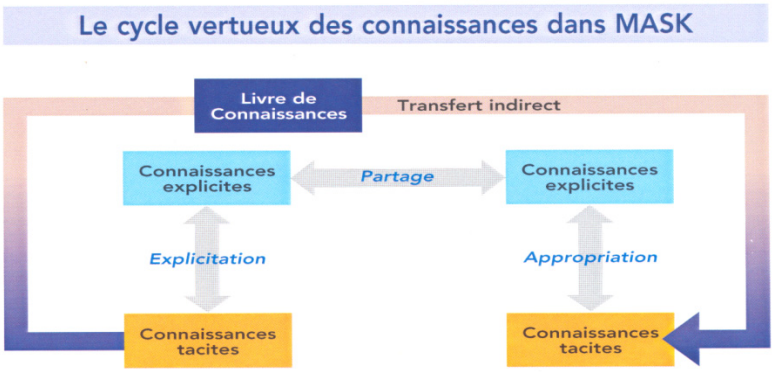


Figure 1.38: Le cycle des connaissances dans MASK. (Tirée d'Ermine et Matta, 2003, p. 8)

Le livre de connaissances est un agglomérat de plusieurs types de modèles, soit le modèle de patrimoine des connaissances, le modèle des phénomènes, le modèle d'activités, le modèle de concepts, le modèle de tâches, le modèle historique et le modèle de lignées. Chacun de ces types de modèles possède une structure graphique

exprimée dans un langage semi-formel, qui permet d'exploiter de manière partiellement automatique la connaissance qu'elle contient.

La modélisation semi-formelle telle qu'utilisée par Basque et *al.* (2004 ; 2010c, 2010b) est aussi une méthode d'élicitation de la connaissance. La méthode consiste à faire produire des modèles MOT par des petits groupes de personnes composées d'experts et de novices de domaine ciblé. Malgré le fait qu'un modèle semi-formel peut comporter des éléments d'ambigüité, la souplesse et le caractère moins artificiel d'un langage semi-formel permettent d'accéder plus facilement à l'expression de connaissances tacites, car la spontanéité n'est pas bloquée par la charge cognitive associée à une formalisation plus poussée (Basque *et al.*, 2008b). L'usage d'un système de représentation plus convivial permet aussi d'élargir le bassin des personnes aptes à représenter leurs connaissances en même temps qu'il évite la démobilisation des experts de leur tâche principale, laquelle coûte cher à une organisation. L'élicitation dans un formalisme de degré semi-formel pourrait ainsi représenter une économie de temps et surtout un gain de qualité dans la représentation des connaissances. De plus, la convivialité du langage semi-formel fait en sorte que des modèles de degré semi-formel peuvent être conçus par les experts du domaine sans l'assistance d'un ingénieur de la connaissance, lequel peut ensuite les formaliser avec la participation minimale des experts les ayant conçus ou encore d'autres experts. Les opérations d'élicitation peuvent ainsi être intégrées de manière plus souple dans les activités des experts du domaine.

Ainsi, tel que mentionné précédemment, nous pensons qu'une phase *d'élicitation* de la connaissance dans un formalisme de degré semi-formel relativement évolué devrait précéder celle de la *formalisation* des connaissances, où le modèle semi-formel est transformé dans un formalisme ontologique.

1.7.3 Méthodologies d'ingénierie ontologique

L'ingénierie ontologique est la discipline des sciences appliquées responsable du développement de méthodes et des outils nécessaires à la conception, la réalisation, la mise en œuvre et la maintenance de systèmes informatiques à base d'ontologies (Gómez-Pérez *et al.*, 2003). Tel qu'illustré à la figure 1.39, le processus de développement d'ontologies englobe des activités de gestion, de développement ainsi que de support.

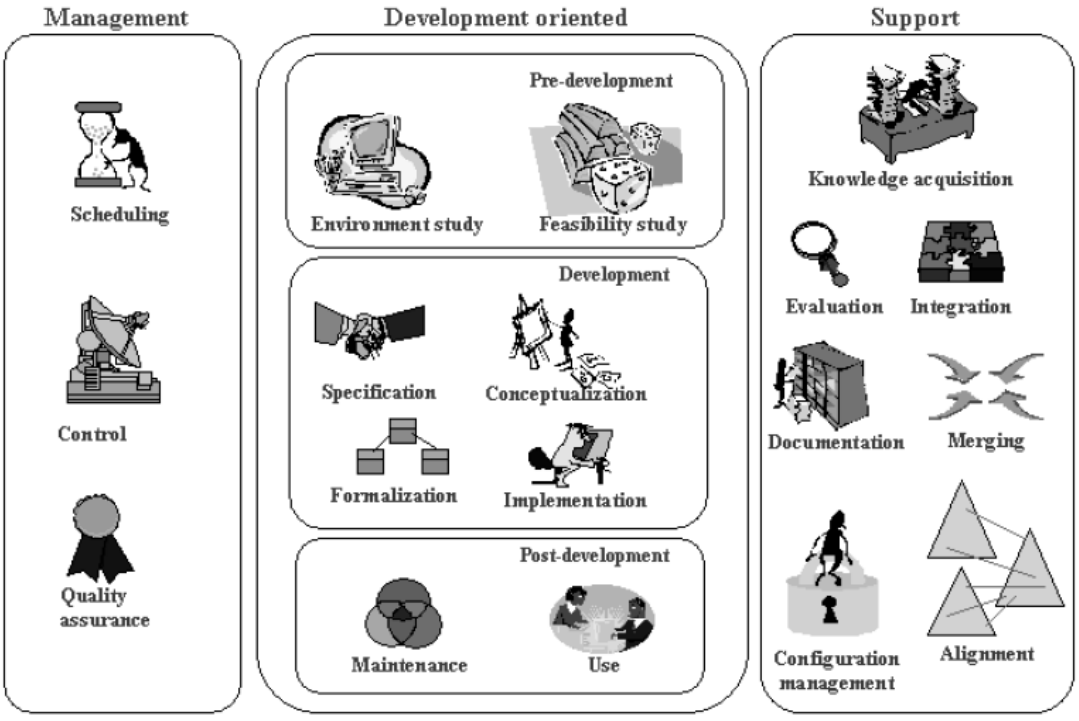


Figure 1.39: Processus de développement d'une ontologie. (Tirée de Gómez-Pérez *et al.*2003, p. 110).

Pour Kendal et Creen (2007) et Uschold et Gruninger (1996), le développement d'une ontologie est un processus linéaire divisé en six stades, soit les suivants: (1) l'étude de faisabilité, (2) l'acquisition de connaissances, (3) la conception, (4) l'implantation, (5) la validation et (6) l'élaboration de la documentation. Inspirés par l'approche itérative couramment utilisée en développement de logiciels, Staab,

Studer, Schnurr et Sure (2001) présentent un processus de développement d'ontologies qui inclut des rétroactions entre les phases de raffinement, d'évaluation et de maintenance (voir la figure 1.40).

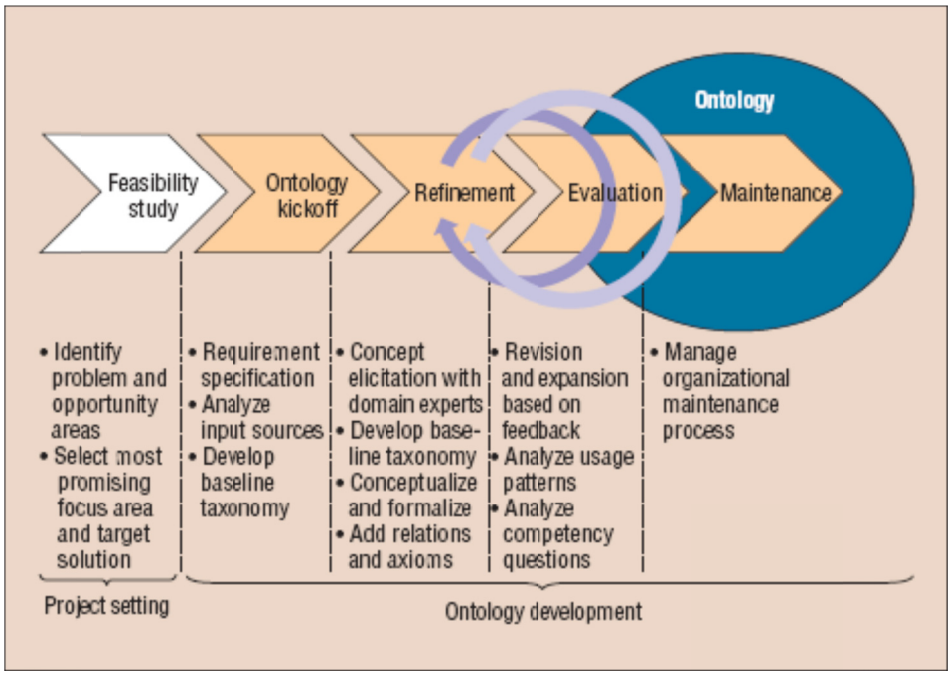


Figure 1.40: Processus de développement d'une ontologie *On-To-Knowledge.* (Tirée de Staab *et al.* 2001)

Quant à la méthodologie METHONTOLOGY, développée par Gómez-Pérez *et al.*, (2003), elle est basée sur un cycle de vie de développement d'ontologies qui est régi par des activités de gestion (voir la figure 1.41). Spécialement adaptée au développement d'ontologies de grande envergure, cette méthodologie intègre le concept de construction d'une ontologie par évolution de prototypes, dont les itérations sont régulées par une planification et pilotées par des activités de contrôle de qualité.

Le projet KACTUS (1996) utilise, pour sa part, une approche de développement d'ontologies axée sur la réutilisation des connaissances. À chacun des cycles de développement de l'application, l'ontologie servant de base de connaissances est raffinée. L'ontologie, produit de l'itération courante, est construite à partir

d'ontologies construites lors des itérations précédentes et par l'intégration d'ontologies déjà existantes.

Enfin, Grüninger et Fox (1995) proposent une méthodologie inspirée du développement de systèmes à base de connaissances utilisant la logique du premier ordre. La conception démarre par l'élaboration d'un scénario interrogatif entre un expert et le futur système. Les concepts, les propriétés et les axiomes de l'ontologie sont construits à partir des connaissances extraites pendant le déroulement du scénario.

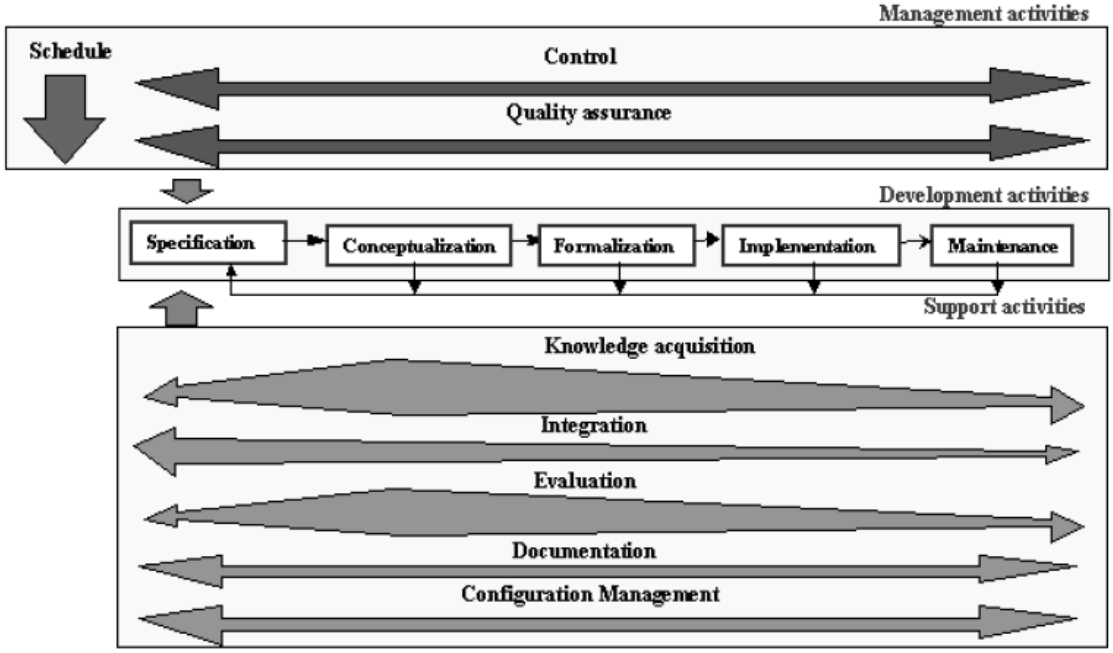


Figure 1.41: Cycle de vie du processus de développement d'ontologie de METHONTOLOGY. (Tirée de Gómez-Pérez *et al.*, 2003, p.127)

Le tableau 1.7 présente un sommaire comparatif des principales méthodologies que nous venons de présenter. Le choix de l'une ou l'autre de ces méthodologies doit être guidé par un questionnement concernant la disponibilité des experts, l'ampleur de l'ontologie à développer, les processus déjà existants de développement de logiciels déjà existants dans l'organisation ou encore les énergies à déployer pour la gestion du

risque lié à tout processus de développement. La valeur ‘NP’ signifie qu’en vertu de la documentation disponible, cette activité n’est pas considérée.

Tableau 1.7
Sommaire comparatif du processus de développement d'ontologies (extrait de Gómez-Pérez *et al.*, 2003, p.151)

Feature			Cyc	Uschold & King	Grüninger & Fox	KACTUS	METHONTOLOGY	SENSUS	On-To-Knowledge
Ontology management activities	Scheduling		NP	NP	NP	NP	*Proposed*	NP	*Described*[12]
	Control		NP	NP	NP	NP	*Proposed*	NP	*Described*
	Quality assurance		NP	NP	NP	NP	*Proposed*	NP	*Described*
Ontology development-oriented activities	Pre-development processes	Environment study	NP	NP	NP	NP	NP	NP	*Proposed*
		Feasibility study	NP	NP	NP	NP	NP	NP	*Described*
	Development processes	Specification	NP	*Proposed*	*Described in detail*	*Proposed*	*Described in detail*	*Proposed*	*Described in detail*
		Conceptualization	NP	NP	*Described in detail*	*Proposed*	*Described in detail*	NP	*Proposed*
		Formalization	NP	NP	*Described in detail*	*Described*	*Described*	NP	*Described*
		Implementation	*Proposed*	*Proposed*	*Described*	*Proposed*	*Described in detail*	*Described*	*Described*
	Post-development processes	Maintenance	NP	NP	NP	NP	*Proposed*	NP	*Proposed*
		Use	NP	NP	NP	NP	NP	NP	*Proposed*
Ontology support activities	Knowledge acquisition		*Proposed*	*Proposed*	*Proposed*	NP	*Described in detail*	NP	*Described*
	Evaluation		NP	Proposed	*Described in detail*	NP	*Described in detail*	NP	*Proposed*
	Integration		*Proposed*[13]	*Proposed*	*Proposed*	*Proposed*	*Proposed*	NP	*Proposed*
	Configuration management		NP	NP	NP	NP	*Described*	NP	*Proposed*
	Documentation		*Proposed*	*Proposed*	*Proposed*	NP	*Described in detail*	NP	*Described*
	Merging and Alignment		NP	NP	NP	NP	NP	NP	NP

Il existe d'autres méthodologies non répertoriées par Gómez-Pérez *et al.* (2003). Par exemple, utilisant les langages OWL et F-Logic (Meštrović et Čubrilo, 2009), le projet NeOn (2006) préconise l'utilisation d'une bibliothèque de patrons de conception d'ontologies (Gangemi *et al.*, 2008). Orienté sur la réutilisation, le principe sous-jacent de la méthodologie NeOn est inspiré de la conception de logiciels par les patrons de conception de Gamma *et al.* (1999). Le patron de conception d'ontologies *"Ontology Design Pattern* (OP)*"* est une solution de modélisation permettant de résoudre un problème de conception récurrent.

1.8 Environnements informatiques d'assistance à l'usager

Dans cette section, nous présentons les niveaux de caractérisation de la fonction d'assistance d'un environnement informatisé.

Plusieurs formes d'assistance peuvent être fournies à l'usager dans un environnement informatique. Paquette *et al.*(1994), citant Wenger (1987) et Jonassen (1991), divise en quatre niveaux le degré d'aide qu'un système intelligent peut apporter à un utilisateur. C'est ce qui va nous servir de guide pour classifier le niveau d'assistance que l'application OntoCASE veut offrir.

Au premier niveau, on retrouve la *rétroaction simple.* Il s'agit d'un mécanisme d'aide qui concentre son action dans la production de messages fondées sur les informations fournies au système par les actions de l'usager. Celui-ci applique une action et le système produit une réponse.

Le deuxième niveau, l'*interventionnisme moyen,* offre à l'utilisateur une aide contextualisée et à la demande. L'aide fournie est toujours en lien avec le contexte d'usage de la fonctionnalité sollicitée. Le contexte d'aide peut être interactif et cette interaction avec le système est une séquence d'événements qui alterne entre *actions d'aide* fournies par le système et *actions posées* par l'utilisateur.

Le troisième niveau, l'*interventionnisme élevé,* est un niveau d'assistance plus intelligent que les précédents puisqu'en plus d'offrir une aide contextualisée, le système peut appliquer des déductions à partir d'un modèle de l'usager dans le but d'offrir des conseils sur l'utilisation du système.

Le quatrième niveau, le *tutorat,* est le plus intelligent des systèmes d'aide. À ce niveau, le système est surtout dédié à l'apprentissage et l'utilisateur est principalement un apprenant. C'est le système qui a l'initiative d'exercer un guidage de l'apprenant. Les interventions du système sont impératives, les erreurs sont soulignées et l'apprenant doit apporter les corrections nécessaires.

À titre d'exemple de système d'assistance de niveau *moyen* et que nous préconisons pour OntoCASE, citons l'*Atelier de Génie Logiciel* (AGL), aussi appelé *Computer Aided Software Engineering* (CASE), qui est un outil informatique qui assiste l'ingénieur logiciel dans le développement de programmes informatiques en tenant compte des paradigmes d'une méthodologie de développement de logiciels (les phases génériques de planification, d'analyse, de conception, d'essais et de maintenance). En plus d'assister l'ingénieur dans le développement de l'application informatique proprement dite, l'AGL intègre des outils de gestion de projets, d'assurance qualité et de documentation (Fuggetta, 1993). Les applications telles qu'*Eclipse*[53] et *NetBeans*[54] sont considérées comme des outils AGL. À ce sujet, notons que plusieurs applications concernant le Web sémantique sont développées

[53] The Eclipse Foundation, *The Eclipse home page*: http://www.eclipse.org/

[54] *NetBeans home page*: http://netbeans.org/

avec la plateforme *Eclipse*, notamment *Protégé 4.0*[55]*, TopBrain Composer*[56]*, NeOn Toolkit*[57] et *Eclipse Ontology Definition Metamodel (EODM) Workbench*[58].

1.9 En résumé

Ce chapitre nous a permis d'aborder les concepts liés à la sémiotique, le concept de niveau d'abstraction ainsi que les concepts de synonymie et de polysémie qui ont permis de définir un vocabulaire pour l'élaboration du cadre théorique d'OntoCASE. De plus, cette recension des écrits a permis de cadrer le concept actuel de modélisation des connaissances, notamment par la classification des langages de représentation des connaissances et la classification des ontologies. Nous avons également présenté un inventaire de quelques langages de représentation des connaissances de degré semi-formel et formel qu'il nous apparaissait important d'étudier en vue de produire une méthodologie générique. Nous avons également exploré des techniques normalisées et dédiées à la transformation de modèles, que nous comptons adapter pour le processus de transformation de modèles semi-formels en ontologies à implanter dans OntoCASE. En ingénierie des connaissances, nous avons identifié une structure métaméthodologique pour concevoir une méthodologie, éliciter des connaissances et répertorier quelques méthodes d'ingénierie ontologique existantes. Finalement, nous avons traité des environnements informatiques pour l'assistance selon une classification des fonctions d'assistance. Dans cette thèse, nous visons pour OntoCASE un niveau moyen d'assistance moyen à l'usager tel que défini par Paquette *et al.* (1994).

---

[55] Protégé Home Site, *Welcome to protégé*: http://protege.stanford.edu/

[56] TopQuadrant, *TopBraid Composer (TM)*: http://www.topquadrant.com/products/TB_Composer.html

[57] NeOn project, *Neon toolkit*: http://neon-toolkit.org/wiki/Main_Page

[58] IBM, *IBM Integrated Ontology Development Toolkit: An ontology toolkit for storage, manipulation, query, and inference of ontologies and corresponding instances.*: http://www.alphaworks.ibm.com/tech/semanticstk

CHAPITRE 2

MÉTHODOLOGIE D'ONTOCASE

Ce chapitre décrit la méthodologie d'OntoCASE. La première section est consacrée à la formulation de l'hypothèse de la recherche, alors que la deuxième section nous permet d'en spécifier les objectifs. La troisième section présente les principales contributions visées de la recherche. Les différents volets d'OntoCASE (*méthodologique, informatique, représentationnel*), les fonctions et les structures de l'ontologie de transformation ainsi que la structure de la méthodologie d'OntoCASE sont autant de thèmes qui seront abordés dans cette section. La section quatre a pour propos l'*ontologie de référence*. Sa structure interne, les entités et relations qui la composent sont autant de sujets qui seront abordés. La section cinq, *l'ontologie-cadre,* présente la définition et l'utilité de cette ontologie et elle en décrit les ressources et propriétés. La section six, dont le sujet est la méthodologie OntoCASE conçue dans le cadre de cette thèse, présente les composants de la méthodologie soit: la méthode de conception d'un modèle semi-formel, la méthode de formalisation en ontologie et la méthode de validation syntaxique et sémantique.

2.1 Hypothèse

L'hypothèse de cette recherche est la suivante : *à partir d'un modèle de connaissances exprimé dans un langage semi-formel qui représente des connaissances de différents types, il est possible de procéder de manière automatique ou semi-automatique, à sa formalisation en ontologie syntaxiquement et sémantiquement valide.*

Cette méthodologie de transformation assistée, appelée *OntoCASE* (voir la forme ovoïde à la figure 2.1), dont l'intrant est un modèle de degré semi-formel et l'extrant un modèle de degré formel, regroupe un ensemble de méthodes et de techniques qui sont instrumentées par un assistant informatique intelligent. La figure 2.1 présente aussi des exemples de systèmes de représentation classifiés selon le quantum des degrés de formalisation. Ceux utilisés dans la présente recherche apparaissent en vert, à savoir : le modèle de connaissances MOT (en tant que prototype de systèmes de représentation de degré semi-formel) et l'ontologie OWL (en tant que prototype de systèmes de représentation de degré formel).

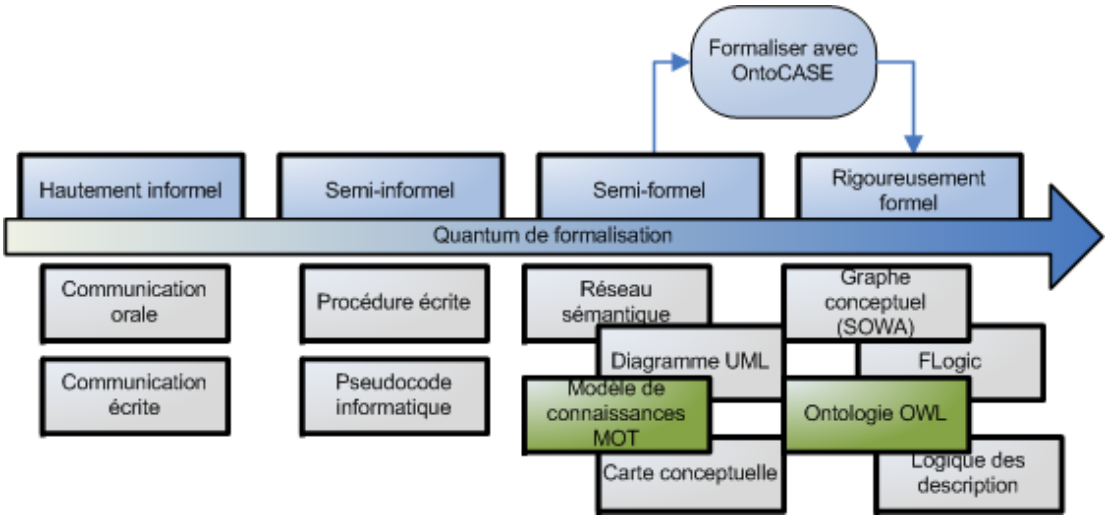


Figure 2.1: Positionnement d'OntoCASE en fonction des degrés de formalisation et classification d'exemples de formalisme.

2.2 Objectifs

De l'hypothèse, découlent ces trois objectifs:

- concevoir une *méthodologie* d'ingénierie ontologique intégrant une méthode de modélisation semi-formelle, une méthode de formalisation en ontologie et une méthode de validation syntaxique et sémantique de l'ontologie produite;

- développer un *outil* informatique *intelligent* et intégré, qui instrumente de manière efficace, efficiente et satisfaisante les différentes méthodes et la méthodologie;

- créer une *ontologie* offrant une représentation *méta du domaine de la représentation des connaissances* qui opérationnalise les assistants informatiques de la méthodologie et qui sert de représentation pour la méthodologie.

2.3 Produits de la recherche

Dans le chapitre précédent, nous avons présenté plusieurs concepts théoriques à la base d'OntoCASE. Cette section nous permettra de donner forme à ces concepts, d'abord sur un plan descriptif par la présentation de l'architecture de haut niveau, puis sur le plan procédural, par la présentation des éléments méthodologiques d'OntoCASE.

2.3.1 Volets d'OntoCASE

OntoCASE, acronyme composé des mots "Ontologie" et « CASE » (*Computer Aided Software Engineering),* se compose d'un volet méthodologique, d'un volet informatique, aussi appelé volet computationnel, ainsi que d'un volet représentationnel (voir la figure 2.2). Le volet méthodologique décrit l'ensemble des éléments procéduraux, des acteurs et des ressources contribuant à la formalisation d'un modèle semi-formel en ontologie. Le volet computationnel décrit, quant à lui, les composants informatiques, ainsi que leurs utilisations, afin d'assister les acteurs dans l'application de la méthode. À chacune des méthodes est associé un module d'assistance informatique qui peut être utilisé par les acteurs humains de la méthode. Finalement, le volet représentationnel comporte un ensemble de guides qui permettent d'orienter le déroulement de chacune des méthodes ainsi qu'une ontologie de transformation qui régit l'opérationnalité des composants du volet computationnel.

Trois composants de haut niveau forment le volet computationnel d'OntoCASE. L'éditeur de modèle semi-formel (appelé eLi) est un éditeur de diagrammes offrant les fonctionnalités nécessaires à la construction d'un modèle semi-formel, inspiré de l'éditeur MOT (Paquette, 2002b, 2010) et de son *langage MOT* (voir l'appendice A). Le système expert à la formalisation basé sur un *catalogue de la sémantique formelle de MOT* (voir l'appendice B) offre les ressources permettant d'assister les acteurs de la méthodologie dans le processus de désambiguïsation et de transformation d'un modèle semi-formel en ontologie. Finalement, l'assistant à la validation, qui implante le *catalogue des cohérences de l'ontologie cadre* et *les bonnes pratiques de modélisation semi-formelle en vue d'en faire une Ontologie* (voir l'appendice C et E), est une application informatique qui offre des fonctionnalités permettant à l'ingénieur et à l'expert de valider syntaxiquement et sémantiquement l'ontologie produite au cours de l'étape de formalisation.

La méthode *intégrer un nouveau formalisme semi-formel à la méthodologie* n'est pas à proprement parler une méthode qui participe à la conception d'une ontologie du domaine; elle a plutôt un rôle métaméthodologique qui assure l'intégration de nouveaux formalismes à la méthodologie par le développement des entités d'OntoCASE (les méthodes, les modules informatiques et les ontologies) permettant de les traiter.

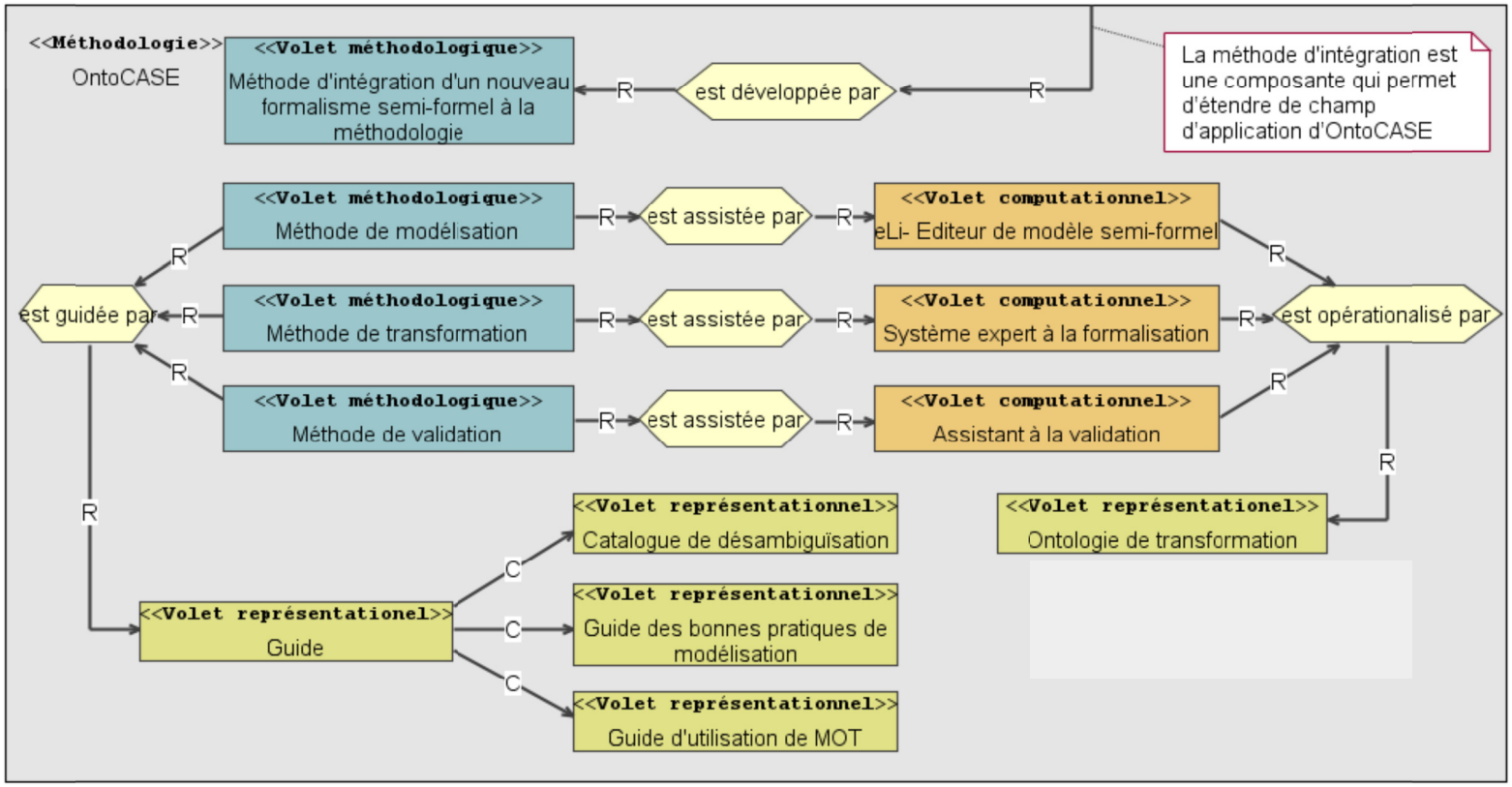


Figure 2.2: Volets méthodologique, computationnel et représentationnel d'OntoCASE.

2.3.2 Ontologie de transformation d'OntoCASE

Au cœur d'OntoCASE, l'ontologie de transformation possède une double fonction. Dans le volet méthodologique, cette ontologie a un rôle représentationnel qui est décrit aux sections 2.4 et 2.6. Dans le volet computationnel, *l'ontologie de transformation* (voir la figure 2.3), qui en est une de haut niveau[59], a un rôle opératoire puisqu'elle constitue la base de connaissances de l'assistant intelligent à la formalisation.

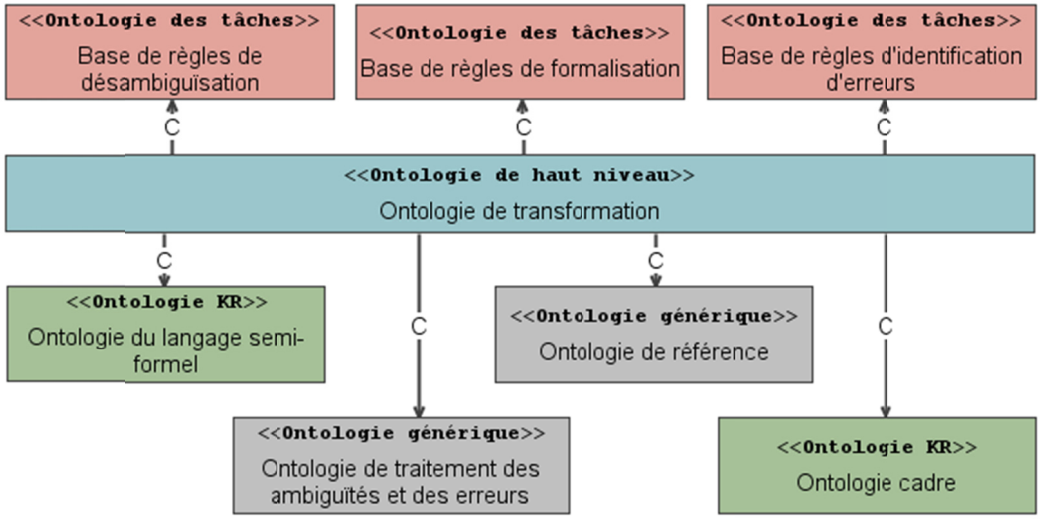


Figure 2.3: Architecture de l'ontologie de transformation.

Quatre composants forment l'ontologie de transformation: (1) l'*ontologie du langage semi-formel* représente les éléments du vocabulaire et de la grammaire du langage semi-formel utilisé pour construire les modèles sources; (2) l'*ontologie de traitement des ambiguïtés et des erreurs* contient des règles servant à la désambiguïsation et à l'identification des erreurs de catégorisation; (3) l'*ontologie de référence* représente les diverses catégories de connaissances et opérationnalise le processus de désambiguïsation et de transformation; et finalement (4) l'*ontologie cadre* encadre les

[59] Les catégories mise en stéréotype à chacune des ontologies sont décrites en détail à la section 1.3.3, p. 32

éléments du modèle cible dans une structure qui supporte la représentation de connaissances déclaratives, procédurales et stratégiques qu'elles soient concrètes ou abstraites.

2.3.3 Portrait synthèse de la méthodologie d'OntoCASE

Le tableau 2.1 présente un portrait synthèse de la méthodologie en mettant en relation les méthodes, techniques et outils qui la composent. La référence aux sections concernées est indiquée entre parenthèse.

Tableau 2.1
Méthodes, techniques et outils de la méthodologie OntoCASE

OntoCASE		
Méthodes	**Techniques**	**Outils**
Concevoir un modèle semi-formel (2.7.1)	Modélisation (2.7.1)	- **eLi**/Mot-plus/GMF-MindMap (3.2.1) - Guide des bonnes pratiques à la modélisation - Guide du langage MOT - Ontologie de MOT (2.4, 3.2.2, E.1)
	Modélisation assistée (2.7.1)	- **eLi** - Guide des bonnes pratiques à la modélisation - Guide du langage MOT
	Co-modélisation (2.7.1)	- **eLi**/Mot-plus/GMF-MindMap - Guide des bonnes pratiques à la modélisation - Guide du langage MOT
Formaliser en ontologie (2.7.2)	MDA (Model Driven Architecture) (1.6)	- **Assistant à la formalisation** - Catalogue à la désambiguïsation - Ontologie de référence (3.1.4)
	ODM (Ontology Data Model) (1.6.6.2, E.1)	
Valider (2.7.3)	Rétroactions inférentielles (3.2.7)	- **Assistant à la validation** - Ontologie cadre (3.1.5)
Intégrer un nouveau formalisme semi-formel à la méthodologie (4.4)	Méthodologie Objets	- **Java**
	MDA/MDD-MOF	- **Eclipse PDE/EMF/GMF (3.1.7**)
	Ingénierie ontologique: - Onto Dev Metamodel (ODM) - Ontology Dev - Rules Dev (3.1.8.2)	- **TopBraid (3.1.7)** - **Protégé (3.1.7**) o ***Java code generation*** o **SWRLTab**

2.4 Ontologie du langage semi-formel

L'ontologie du langage semi-formel (voir le tableau 2.2) contient la structure du langage semi-formel et permet de classifier chacun des éléments du modèle. Pour le langage MOT, les éléments du modèle sont répartis entre *connaissance* et *relation* avant d'être sous-classifiés dans leur classe respective (par exemple, en concept). Chaque langage semi-formel intégré à la méthodologie comporte sa propre ontologie du langage semi-formel qui le caractérise.

Tableau 2.2
Structure de classes de l'ontologie du langage semi-formel

MOT_Connaissance MOT_ConnaissanceAbstraite MOT_Concept MOT_Principe MOT_Procedure MOT_Fait MOT_Decision MOT_Enonce MOT_Exemple MOT_Trace MOT_NonType	MOT_Relation MOT_LienA MOT_LienC MOT_LienCm MOT_LienI MOT_LienIP MOT_LienNT MOT_LienP MOT_LienR MOT_LienS

2.5 Ontologie de référence

Représentée à la figure 2.4, l'ontologie de référence, de type générique, est au cœur du processus de transformation de la représentation semi-formelle à la représentation formelle. Pour le système de transformation, l'ontologie de référence est un modèle de travail; elle assure un entreposage temporaire des informations avant qu'elles ne soient utilisées par le sous-processus de conversion en langage formel. L'étape de désambiguïsation, spécifique à chaque langage de modélisation semi-formel, catégorise chacun des éléments du modèle source dans l'ontologie de référence. Chacun des formalismes possède son propre vocabulaire. Par exemple, en UML, un holonyme est représenté par une relation d'agrégation, alors qu'en langage MOT, il est représenté par un lien de composition. La désambiguïsation respective à chacun de

ces formalismes classera ces relations dans la classe « Holonyme » de l'ontologie de référence.

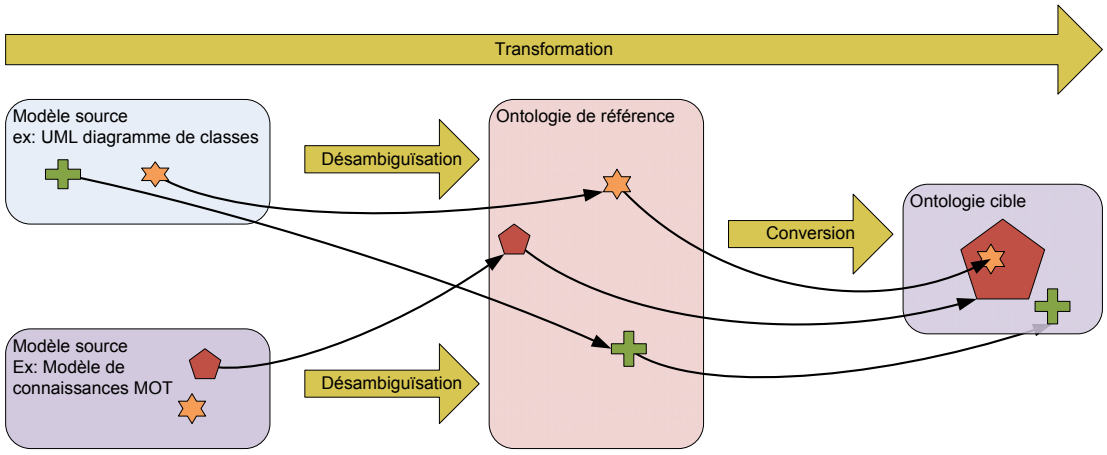


Figure 2.4: Principe de transformation guidée par l'ontologie de référence.

Après la désambiguïsation, est exécutée la conversion. Cette étape sert à créer une nouvelle ontologie dont la sémantique respecte celle du modèle semi-formel source. Deux mécanismes d'inférence sont utilisés par le processus de conversion. À partir des éléments du modèle source qui constituent la base de faits de l'ontologie de référence, le premier mécanisme d'inférence classifie les éléments du modèle. À partir de la classification, une inférence à base de règles est déclenchée créant ainsi, pour chacun des faits de l'ontologie de référence, une entité ontologique correspondante dans l'ontologie cible.

2.5.1 Structure interne de l'ontologie de référence

À sa base, la structure interne de *l'ontologie de référence* (voir le tableau 2.3 et le tableau 2.6) se divise en *entités* et *relations*. Ce modèle de structure proposé par Chen (1976, 2002) est utilisé par de nombreux systèmes de représentation. C'est le cas

notamment des représentations de type *TopicMap*[60], du *Concept Mapping*[61] ou encore de l'UML[62].

2.5.2 Entités de l'ontologie de référence

Nous proposons six types d'entités de base pour l'ontologie de référence, qui se distinguent selon le niveau de réalité (*subjectif* ou *objectif*) et le type de connaissances (*déclaratif, procédural ou stratégique*) qu'elles représentent (voir le tableau 2.3 et le tableau 2.4): le concept, la manifestation, l'action, l'acte, le principe et le facteur d'influence. La *manifestation* et le *concept* sont des entités *déclaratives* qui représentent respectivement les objets concrets et abstraits d'un domaine et font ainsi référence aux deux niveaux de réalité du domaine de connaissances (objectif et subjectif). Par exemple, « la 'chaise' de Paul » (une manifestation) ou « une 'chaise' (un concept) est composée de quatre pattes ». L'*action* et l'*acte* sont des entités qui représentent des connaissances *procédurales* de la réalité subjective et objective. À titre d'exemples de ces deux types de connaissances, citons « exécuter la 'procédure d'évacuation' » (action) et « Pierre 'utilise l'extincteur' pour maîtriser l'incendie » (acte). Finalement, le *principe* et le *facteur d'influence* sont des entités utilisées pour représenter des connaissances stratégiques abstraites et concrètes. Par exemple, « le rôle d'un 'politicien', c'est de gouverner » constitue un principe alors que « 'si l'eau est à 100 °C alors, elle bout' » constitue un facteur d'influence.

Chacune des entités est divisée en catégories selon le schème de couleur du tableau 2.4. Les dénominations de primaire (vert), secondaire (beige) et tertiaire (orange) ont été choisies de manière arbitraire. Elles permentent cependant d'établir une catégorisation hiérarchique des niveaux des sous-entités. L'entité *concept* est

---

[60] SC34/WG3 *Home page of the Topic Maps ISO standards*: http://www.isotopicmaps.org/

[61] IHMC, *IHMC CmapTools: Concept Mapping Home Site*: http://cmap.ihmc.us/

[62] OMG UML, *UML® Resource Page*: http://www.uml.org/

divisée en deux sous-entités: la *classe* représente des connaissances déclaratives abstraites quelconques, alors que le *schéma* est utilisé pour représenter des connaissances schématisées telles que les entiers (*int*), les chaînes de caractères (*string*) ou les nombres réels (*float*). L'entité *manifestation* est aussi divisée en deux sous-entités. D'une part, l'*objet* représente l'instance d'une classe, c'est-à-dire la représentation de la manifestation d'une connaissance quelconque, alors que la catégorie *attribut* est utilisée pour emmagasiner une valeur associée à un attribut. En revanche, la classe et l'objet font partie de la même catégorisation *primaire*. De même, le schéma et l'attribut font partie de la catégorie *secondaire*.

Tableau 2.3
Catégorie des entités du modèle de référence

<table>
<tr><th colspan="3" rowspan="2">ENTITÉ</th><th colspan="6">Type de connaissance</th></tr>
<tr><th colspan="3">Déclaratif
(quoi)</th><th colspan="2">Procédural
(comment)</th><th colspan="2">Stratégique
(pourquoi, qui, quand)</th></tr>
<tr><td rowspan="9">Niveau de réalité</td><td rowspan="5">Subjectif
(Abstrait)</td><td rowspan="5">Concept</td><td colspan="2">Classe</td><td rowspan="5">Action</td><td>Opération</td><td rowspan="5">Principe</td><td>Agent</td></tr>
<tr><td rowspan="4">Schéma</td><td>String</td><td rowspan="4">Procédure</td><td>Condition</td></tr>
<tr><td>Int</td><td rowspan="4">Règle</td><td>Nom</td></tr>
<tr><td rowspan="2">Float</td><td>Antécédent</td></tr>
<tr><td>Non décomposée</td></tr>
<tr><td>Propriété</td><td>Conclusion</td></tr>
<tr><td rowspan="4">Objectif
(Concret)</td><td rowspan="4">Manifestation</td><td colspan="2">Objet</td><td rowspan="4">Acte</td><td>Opération</td><td rowspan="4">Facteur d'influence</td><td>Agent</td></tr>
<tr><td colspan="2" rowspan="3">Attribut</td><td rowspan="3">Procédure</td><td>Condition</td></tr>
<tr><td rowspan="3">Règle</td><td>Nom</td></tr>
<tr><td>Antécédent</td></tr>
<tr><td>Non décomposée</td><td>Conséquent</td></tr>
<tr><td colspan="7">Assertion</td></tr>
</table>

Tableau 2.4
Catégorie des entités et leur couleur correspondante

Catégorie	Correspondance des couleurs
Primaire	
Secondaire	
Règle nom	
Règle antécédent	
Non décomposée	
Règle conséquent	
Tertiaire	

D'un point de vue *procédural*, chacune des entités *action* et *acte* est aussi respectivement divisée en deux sous-entités, soit la procédure et l'opération. La *procédure* est la connaissance procédurale qui fait référence à l'idée de mouvement. Elle explique le *comment* de la réalisation d'une chose. L'opération est aussi associée à l'idée de mouvement; cependant elle est spécialisée dans l'expression d'un *comment* qui se réalise à la suite de la détermination d'une condition. Par exemple, l'opération s'utilise dans une expression de la forme: « *si 'une condition' alors 'une opération'* ».

L'entité *principe* et son homologue concret, le *facteur d'influence*, sont divisés chacun en sept sous-entités similaires. La première d'entre elle sert à représenter des connaissances associées à des *agents*, des normes ou des contraintes. La deuxième sous-entité assure la représentation de *conditions* telles que celles utilisées dans les ordinogrammes. Les sous-entités *nom, antécédent, non-décomposée*[63] et *conséquent,* sont des sous-entités associées à la représentation d'une règle. Ces sous-entités sont utilisées pour la représentation d'énoncés de la forme :

« *La règle de 'Nom' = Si 'antécédent' alors 'conséquent'* »

ou encore:

« *'non-décomposée' = l'énoncé complet d'une règle* ».

Le tableau 2.5 présente un exemple de la représentation d'une règle dans le langage MOT. En **a** est représenté l'exemple d'une règle fragmentée en *nom de la règle, antécédent* et *conséquent.* En **b** est représenté l'exemple d'une règle dont la conclusion aboutie par l'exécution d'une opération, et en **c,** il est représenté l'exemple d'une règle *non-décomposée.*

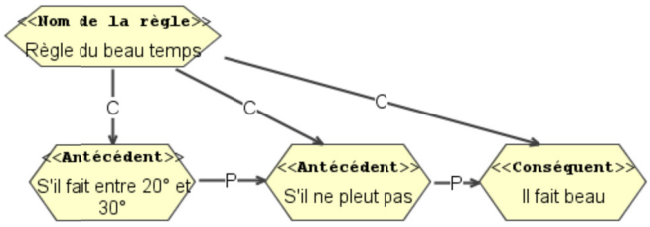
Finalement, quant aux sous-entités *propriété* et *assertion*, elles permettent de représenter le prédicat dans les énoncés de type « sujet/prédicat/objet ».

---

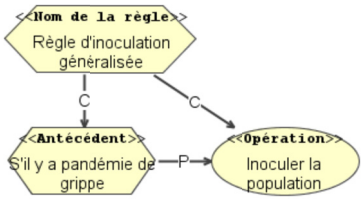
[63] Parfois, à certains endroits de la thèse, on utilise le terme « règle complète ».

Tableau 2.5
Exemple de représentation d'une règle en MOT

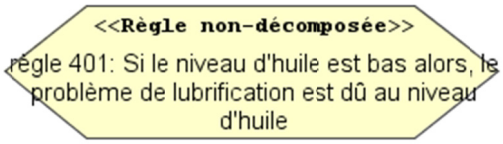
a) « La règle du beau temps »: « S'il fait entre 20° et 30° » et « s'il ne pleut pas » alors « il fait beau »



b) « La règle d'inoculation généralisée »: « S'il y a pandémie de grippe » alors « inoculer la population »



c)



2.5.3 Relations du modèle de référence

Le tableau 2.6 présente les catégories associées aux relations. La relation de type *flux* s'utilise pour représenter les intrants ou les extrants (qui sont des connaissances déclaratives) d'une connaissance procédurale. La catégorie *précédence* est utilisée pour représenter une séquence entre deux connaissances procédurales.

La relation d'*holonymie* représente l'idée d'éléments faisant partie d'un ensemble : il s'agit d'une relation transitive (Lacy, 2005 p. 39). La catégorie *englobe* est la moins contraignante des relations d'holonymie. Elle est utilisée pour représenter des connaissances qui font partie d'un sous-modèle de connaissances. Il n'y a donc aucune restriction sur les types de connaissances qui sont reliées par une relation 'englobe'.

Par exemple, une connaissance procédurale peut englober une ou plusieurs connaissances déclaratives, procédurales ou stratégiques. Par contre, les relations d'holonymie de catégories *agrégation* et *composition* ont une restriction sur le type des connaissances qu'elles relient puisqu'elles doivent relier des connaissances de même type. L'*agrégation* n'a aucun présumé concernant l'existence des objets qui sont unis. Dans l'énoncé « *l'automobile se compose de quatre roues »,* l'existence des roues n'est pas liée à l'existence de l'automobile. En revanche, dans l'énoncé « *un corps humain se compose de 2 bras »*, l'existence des bras est conditionnée par l'existence du corps humain. Nous parlerons alors d'une relation d'holonymie de type *composition.*

Tableau 2.6
Catégorie des relations du modèle de référence

RELATION	**Catégorie**
Flux	Intrant
	Produit
	Précédence
Holonymie	Agrégation
	Composition
	Englobe
Hyponymie	Généralisation
	Spécialisation
Méta	Définition
	Instanciation
Régulation	

La relation d'*hyponymie* est une relation de classification des entités qu'elle relie. L'hyponymie est transitive et elle est utilisée pour relier des connaissances de même type. Dans notre catégorisation, hyponymie et spécialisation sont synonymes alors que nous devrions utiliser le terme *hyperonyme* pour désigner la relation de *généralisation.* Ainsi, généralisation et spécialisation sont tous les deux des antonymes (de sens contraire). Nous avons tout de même pris la liberté de classer les catégories généralisation et spécialisation sous la relation hyponymie en prenant bien

note qu'un traitement particulier devra être réalisé afin de tenir compte de leurs sens contraire, tout comme c'est le cas pour intrant et produit dans les relations de flux.

Enfin, la relation *Méta,* qui est non transitive, est utilisée pour représenter le lien qui existe entre la représentation de deux entités de même type (*déclaratif, procédual ou stratégique*), mais de niveaux d'abstraction différents. Ainsi, l'*instanciation* représente la relation entre une entité abstraite et une entité concrète. Quant à la relation *définit,* surtout utilisée en métamodélisation, elle représente la relation entre deux entités abstraites qui sont de niveaux d'abstraction différents. Finalement, la relation de *régulation*, qui est une relation non-transitive, représente l'influence que porte une connaissance stratégique sur une autre connaissance.

2.6 Ontologie cadre en représentation des connaissances d'OntoCASE

L'ontologie cadre est une ontologie dont la définition des concepts concerne plusieurs disciplines. Les termes *upper ontology, generic ontology, top-level ontology* sont des synonymes anglais du vocable « ontologie cadre » (Oberle, 2006 p. 46). De plus, selon la catégorisation de Gómez-Pérez *et al.* (2003), l'ontologie cadre en est une de type "*Knowledge Representation"* puisqu'elle permet de décrire un domaine selon une typologie faisant référence à un système de représentation de connaissances. Cette ontologie permet de définir les éléments de l'ontologie du domaine afin d'en élargir l'expressivité. Outre l'expression de connaissances déclaratives, propre aux ontologies, l'ontologie cadre permet l'expression de connaissances procédurales et stratégiques dans l'ontologie du domaine.

L'exemple de la figure 2.5 illustre les niveaux des modèles de l'ontologie cadre dans le domaine de la biologie. La métamodélisation divise le domaine en plusieurs couches d'abstraction. Tel que déjà mentionné, les couches M0 et M1 représentent respectivement le domaine du discours et la sémantique du domaine du discours. Dans cet exemple, il est possible de conclure que <bahia> est un 'animal' puisque la transitivité de la relation de généralisation permet de relier 'Berger Allemand' à

'Chien', puis à 'Canin' … jusqu'à 'Animal'. Les éléments des couches M2 et M3 permettent la définition des éléments de la couche M1. Ainsi, le métamodèle définit 'Berger Allemand" comme étant une 'Race', puis 'Chien' en tant que 'Espèce', etc. La relation non transitive qui relie les éléments de couches d'abstraction différentes est appelée *instance ontologique*. En effet, cette relation est non transitive puisque, bien que <bahia> soit un 'Berger Allemand', on ne peut pas conclure que <bahia> soit une 'Race'. Par contre, on peut affirmer que: 'bahia' est un chien de race 'Berger Allemand'.

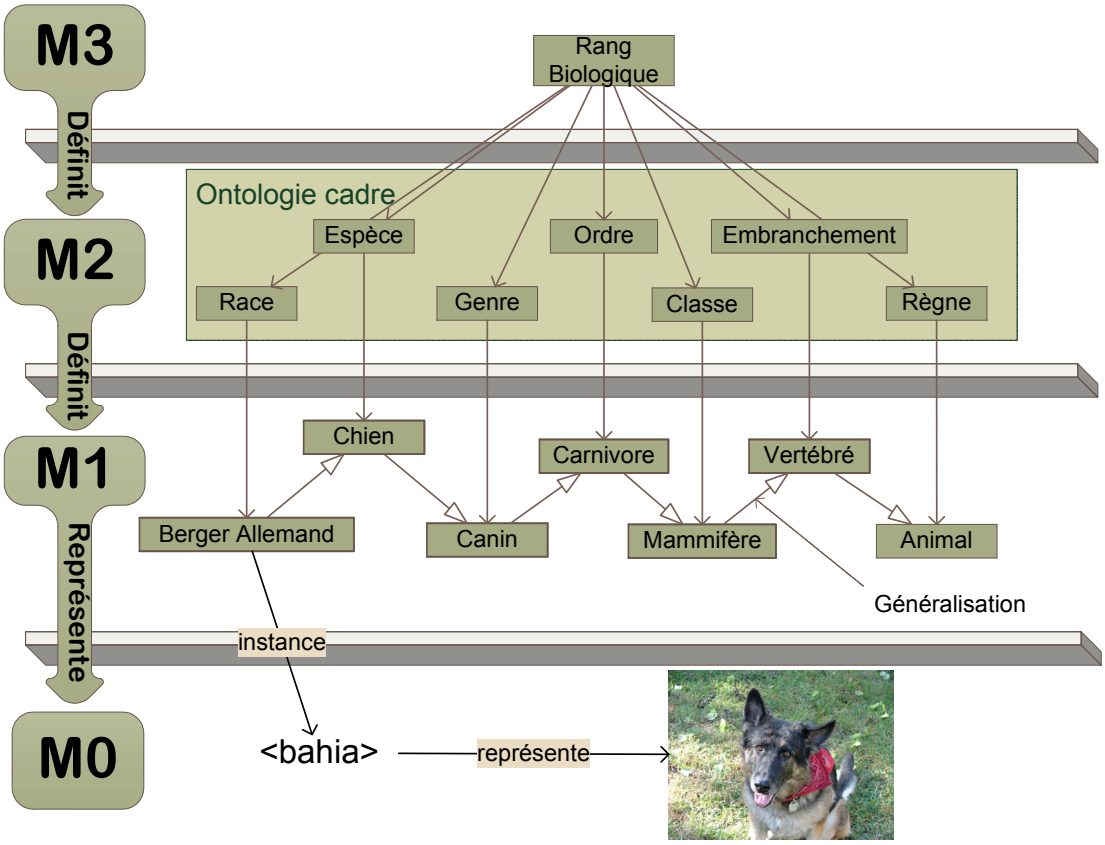


Figure 2.5: Ontologie cadre de la biologie.

Dans cette recherche, l'ontologie cadre d'OntoCASE encadre le modèle cible afin de compenser les éléments de syntaxe et de grammaire qui sont manquants dans le langage de modélisation cible. À titre d'exemple, supposons un modèle source conçu en langage MOT et sa conversion (le modèle cible) en langage OWL : le vocabulaire

de MOT permet l'expression de relations de composition ou encore l'expression de connaissances procédurales qui ne font pas partie d'OWL. Dans ce cas, le rôle de l'ontologie cadre est de contenir une définition des éléments d'agrégation et de connaissances procédurales et de préserver les propriétés sémantiques de ces éléments par une représentation adéquate de celles-ci dans le langage de modélisation cible.

La figure 2.6 présente un exemple de l'utilisation partielle de l'ontologie cadre d'OntoCASE pour le modèle *un berger allemand se compose de pattes.* L'ontologie cadre d'OntoCASE, au niveau M2, encapsule les éléments du modèle du niveau M1.

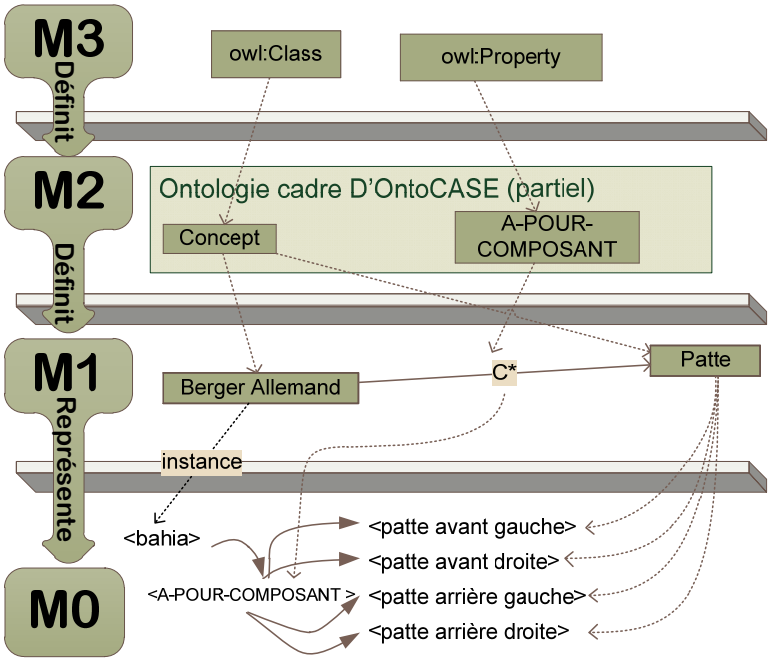


Figure 2.6: Exemple de classification avec L'ontologie cadre d'OntoCASE.

2.6.1 Ressources de l'ontologie cadre d'OntoCASE

Au tableau 2.7 sont présentées les ressources de l'ontologie cadre. On désigne par ressources, les classes d'une ontologie exprimées dans un formalisme spécifique au Web, par exemple l'Ontology Web Language (OWL). On y retrouve les types de

connaissances présentés dans l'ontologie de référence. En revanche, les niveaux de réalités n'y figurent pas puisque cette particularité de représentation fait partie de l'expressivité du formalisme ontologique. Les ressources de l'ontologie cadre sont en fait des méta-classes qui définissent les classes associées au domaine de connaissances à représenter.

Tableau 2.7
Ressources de l'ontologie cadre d'OntoCASE

Types de connaissances	**Catégorie**	
Déclarative	Concept	
	Schéma	Float
		Int
		String
Procédurale	Procédure	
	Opération	
Stratégique	Agent_Contrainte_Norme	
	Condition	
	Entite_Regle	Antécédent
		Non décomposée
		Conclusion
		Nom

2.6.2 Propriétés de l'ontologie cadre d'OntoCASE

De la même manière que pour les ressources, les propriétés de l'ontologie cadre sont en fait des méta-propriétés qui permettent de catégoriser les propriétés associées à un domaine de connaissances. Présentées au tableau 2.8, les propriétés de l'ontologie cadre couvrent une partie importante des types de relations pouvant relier deux connaissances entre elles. Les propriétés restantes étant fournies à même le langage OWL.

Tableau 2.8
Propriétés de l'ontologie cadre d'OntoCASE

Nom de la propriété		Nom de la propriété inverse	
A-POUR-ATTRIBUT			
A-POUR-COMPOSANT		EST-LA-PARTIE-DE	
PERMET-DE	ALORS	A-POUR-DEPENDANCE	
	A-POUR-CONCLUSION		EST-LA-CONCLUSION-DE
	EST-ANTECEDENT-DE		A-POUR-ANTECEDENT
	EST-PRECEDENT-DE		EST-LE-SUIVANT-DE
	PUIS-EVALUER		EVALUE-A-PARTIR-DE
	PUIS-EXECUTER		A-POUR-PREALABLE
ENGLOBE		EST-ENGLOBE-PAR	
INTRANT-PRODUIT	A-POUR-INTRANT	EST-L_INTRANT-DE	
	A-POUR-PRODUIT	EST-LE-PRODUIT-DE	
REGIT		OBEIT-A	
PROPRIETE			

La propriété « PROPRIETE » est une super propriété qui permet d'encapsuler les propriétés du domaine. INTRANT-PRODUIT, PERMET-DE et A-POUR-DEPENDANCE sont aussi des super-propriétés.

2.6.3 L'ontologie cadre d'OntoCASE en OWL-N3

L'appendice F.11 à la p.367 presente l'ontologie cadre en représentation des connaissances d'OntoCASE dans le notation N-3.

2.7 Méthodologie de conception d'une ontologie formelle à partir d'un modèle semi-formel

Schématisé à la figure 2.7, la méthodologie de conception d'une ontologie formelle à partir d'un modèle semi-formel se compose de trois méthodes, soit les suivantes: *concevoir un modèle semi-formel*, *formaliser en ontologie du domaine* et *valider l'ontologie du domaine.* Quatre artéfacts sont produits par la méthodologie, soit le *modèle semi-formel du domaine, l'ontologie du domaine,* le *rapport de validation syntaxique* et le *rapport de validation sémantique.* La méthodologie est itérative,

c'est-à-dire que le résultat de l'itération précédente sert d'intrant à l'itération en cours de réalisation. C'est ainsi que les rapports de validation réalisés à l'itération précédente servent d'intrants à la méthode de conception du modèle semi-formel de l'itération en cours. Le détail de chacune de ces méthodes sera développé dans les sections subséquentes.

Les acteurs qui agissent sur la méthodologie sont au nombre de cinq. Le premier de ces acteurs est l'*expert de contenu*. Son rôle principal est d'extérioriser l'expertise qu'il détient par la production d'un modèle semi-formel du domaine valide et cohérent face à son domaine d'expertise. Il est donc directement impliqué dans la réalisation des méthodes de conception du modèle semi-formel du domaine et dans la méthode de validation de l'ontologie du domaine.

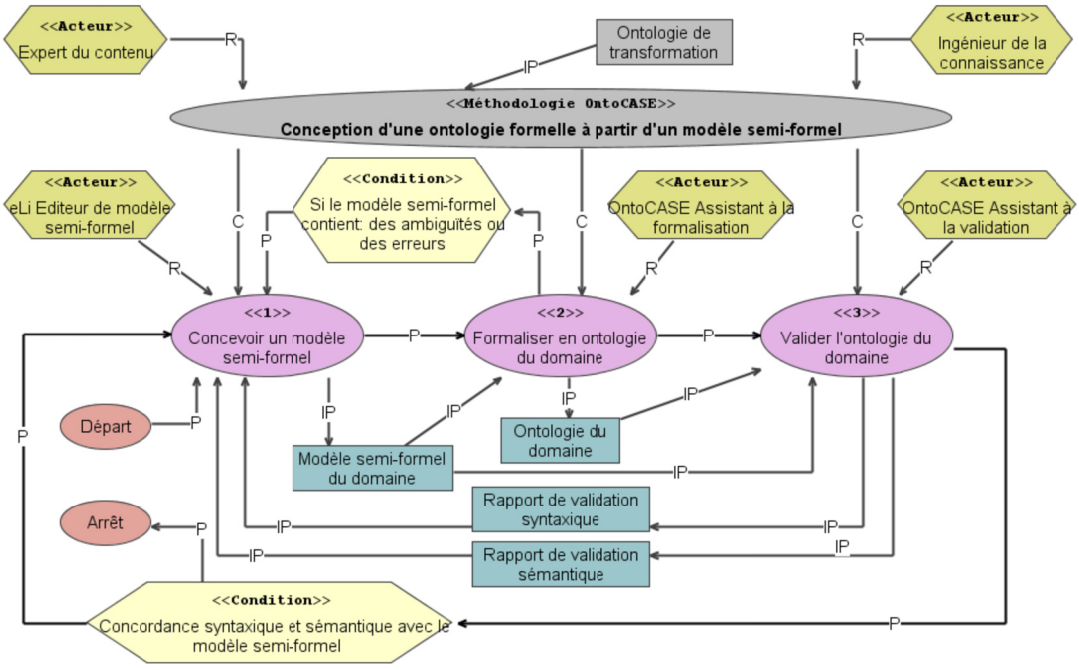


Figure 2.7: Modèle de la méthodologie de conception d'une ontologie à partir d'un modèle semi-formel.

Le deuxième acteur de la méthodologie est l'*ingénieur de la connaissance*. En raison de son expertise en conception d'ontologies, cet acteur a un rôle de support à l'expert

de contenu pour la formalisation de son expertise. L'ingénieur guide l'expert et parfois le novice pendant la conception du modèle semi-formel; il réalise l'exécution de la méthode de formalisation, puis il travaille en collaboration avec l'expert pour la validation de l'ontologie.

Les troisième, quatrième et cinquième acteurs sont les outils qui forment le volet computationnel d'OntoCASE, soit l'éditeur de modèle MOT eLi, le système expert à la formalisation et l'assistant à la validation qui sont respectivement décrits à la section 2.3.1, p. 81.

2.7.1 Méthode de conception d'un modèle semi-formel

Schématisée à la figure 2.8, la méthode *concevoir un modèle semi-formel* présente deux spécialisations : *modéliser* et *comodéliser*.

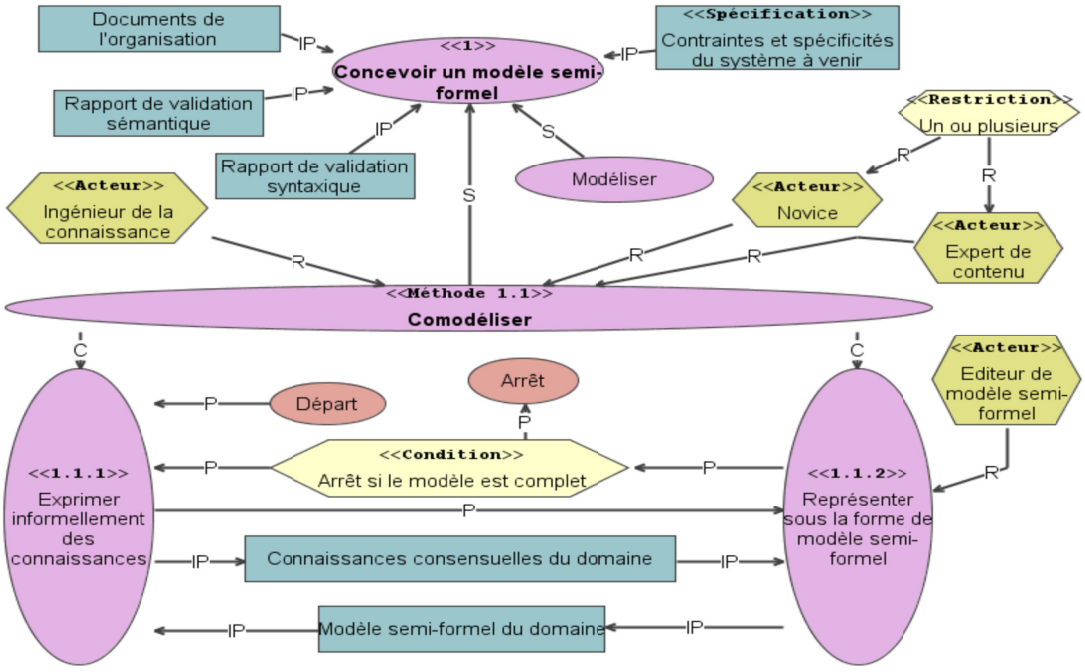


Figure 2.8: Méthode de conception d'un modèle semi-formel.

La modélisation, en vue de concevoir un système à base de connaissances, ou d'une ontologie, passe par une étape d'identification, d'explicitation, de structuration et de représentation de la connaissance avant qu'elle ne soit utilisée pour la conception du système à base de connaissances (Hart, 1986 ; Paquette et Roy, 1990). À sa plus simple expression, la modélisation est assumée par un ingénieur de la connaissance, aussi appelé cogniticien, et un expert de domaine. Une méthode particulièrement prometteuse appelée *comodélisation* (Basque *et al.*, 2004 ; Basque *et al.*, 2008b) est proposée dans le cadre de la méthodologie d'OntoCASE. Cette méthode implique la participation d'un ou plusieurs experts et éventuellement de personnes plus novices (Basque et Pudelko, 2010b) dans le domaine de connaissances ciblé, assistées de l'acteur ingénieur de la connaissance. Nous nous intéressons plus particulièrement à cette méthode puisque, de notre point de vue, elle nous semble le plus correspondre au caractère consensuel de toute ontologie selon Gruber (1993a) et Borst (1997).

La comodélisation vise à stimuler l'externalisation des connaissances par la mise en présence de plusieurs intervenants dans l'activité de modélisation. L'ingénieur est l'acteur responsable de l'animation de l'activité de comodélisation et s'occupe de transposer les connaissances exprimées sous la forme d'une représentation graphique structurée selon un certain langage. L'acteur novice (qui est un connaissant du domaine sans en être un expert par manque d'expérience sur le terrain) et l'acteur expert sont des rôles qui peuvent être tenus chacun par une ou plusieurs personnes. Le rôle de novice peut aussi être tenu par l'ingénieur de la connaissance. Les échanges entre les acteurs et les accords qu'ils établissent entre eux permettent la modélisation de connaissances consensuelles. Les connaissances consensuelles ainsi exprimées sont représentées dans un modèle semi-formel, qui, par la suite, sert de nouvelle base de discussion pour la production de nouvelles connaissances consensuelles. Le cycle d'expression et de modélisation se poursuit jusqu'à ce que les acteurs humains décident par consensus de la complétude du modèle. Plusieurs artéfacts servent d'intrants à la conception du modèle semi-formel, qu'il s'agisse des rapports de validation sémantique et syntaxique obtenus à l'étape de validation de l'itération

précédente (que nous discuterons à la section 2.7.3, p.109), des spécifications de normes et contraintes du système à venir ou de tout autre document de l'organisation. Des détails sur la mise en œuvre de la méthode de co-modélisation à des fins d'élicitation des connaissances sont fournis dans Basque et Pudelko (2008, 2010b) et dans Basque *et al*. (2008a)

2.7.2 Méthode de formalisation en ontologie du domaine

La méthode *formaliser en ontologie du domaine* (schématisée à la figure 2.9 et présentée sous la forme d'un exemple à la section 2.7.2.4 p. 104) est contrôlée par l'ingénieur de la connaissance. Trois processus composent la méthode, soit (1) *importer dans l'espace de modélisation ontologique,* (2) *désambiguïser* et (3) *convertir en ontologie du domaine.*

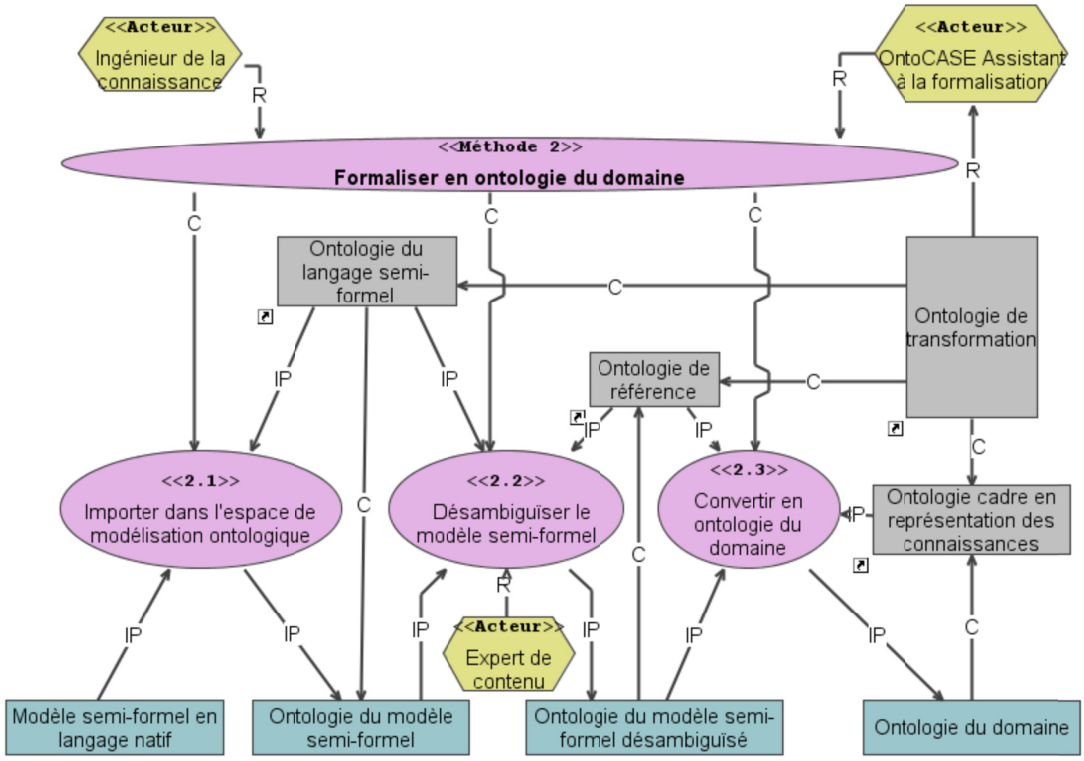


Figure 2.9: Méthode de formalisation d'un modèle semi-formel en ontologie du domaine.

2.7.2.1 Importer dans l'espace de modélisation ontologique

Le processus d'importation « 2.1 » sert à la traduction d'un modèle semi-formel employant le formalisme de l'éditeur de modèles semi-formel (par exemple le *XMI*) vers un modèle employant le formalisme ontologique utilisé par le modèle cible (par exemple le *OWL*). Il s'agit d'une représentation en OWL du modèle source. Quant au *lienC* mis entre les deux modèles, le premier modèle OWL représente l'ontologie du langage MOT et le second, les individus correspondant au modèle source. Un exemple détaillé d'importation est présenté au tableau 2.9 **a** et **b**.

2.7.2.2 Désambiguïser le modèle semi-formel

Le processus de désambiguïsation « 2.2 », qui dans certains cas est assisté par l'expert de contenu, consiste à supprimer les ambigüités contenues dans *l'ontologie du modèle semi-formel* afin de produire *l'ontologie du modèle semi-formel désambiguïsé.* Pour ce faire, le processus de désambiguïsation utilise la sémantique du langage source (l'ontologie du langage semi-formel) pour identifier le type de désambiguïsation appropriée du modèle semi-formel. L'ontologie du langage semi-formel en intrant de ce processus détermine la sémantique des éléments de l'ontologie du modèle semi-formel à désambiguïser. Trois catégories de désambiguïsation sont réalisées : *typologique, topologique* et *sémantique.*

La *désambiguïsation typologique* est la plus automatique des désambiguïsations. Elle associe le type d'un composant du modèle semi-formel à un élément ontologique. Par exemple (voir le cas 1 du tableau 2.9 et du tableau 2.10), le composant *lienS* (*sorte-de*) s'interprète comme une relation d'hyponymie (de type généralisation) et le composant *lienI* (*instance*) s'interprète comme une relation d'instanciation.

La *désambiguïsation topologique* est plus complexe et peut, dans quelques cas, n'être que semi-automatisée. Cette désambiguïsation s'établit en caractérisant les composants du modèle par l'identification d'un patron de disposition. À l'exemple du cas 2 du tableau 2.9 et du tableau 2.10, les concepts C4 et C5 sont interprétés comme

des classes et le principe P1 comme une propriété (binaire) mettant en relation ces classes, grâce à l'identification du patron de disposition suivant: un *principe* est uni par un *lienR* à un *concept* en intrant et est uni par un *lienR* à un *concept* extrant. Également, les connaissances stratégiques P1 et A1 qui sont représentées par des *Principes* dans le modèle semi-formel sont désambiguïsées en *Propriétés* pour P1 et en *connaissances stratégiques* pour A1.

Du point de vue de l'ingénieur de la connaissance, l'étape de *désambigüisation selon la sémantique du domaine* est délicate, car elle nécessite une compréhension du domaine qui est modélisé, ce qui implique la collaboration de l'expert de contenu afin de répondre aux questions de l'ingénieur. L'exemple d'interprétation du *lien de composition* dans le langage MOT est une bonne illustration de ce type d'ambiguïté (voir le cas 3 du du tableau 2.9 et du tableau 2.10). Le lien de composition s'interprète de deux façons. La première interprétation possible est la composition entre concepts (exemple: C7 *a pour partie* C8). La deuxième interprétation possible est l'utilisation du lien de composition pour assigner des attributs à un concept (exemple : C7 *a pour attribut* C9), c'est-à-dire que la relation unit une classe à un objet *DataType.* Seule une connaissance adéquate du domaine de connaissances permet de lever l'ambiguïté liée à cette double interprétation du *LienC.*

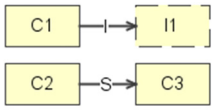
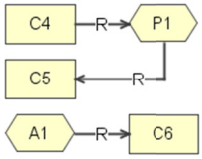
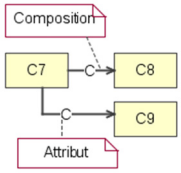
2.7.2.3 Convertir en ontologie du domaine

Finalement, à partir de l'ontologie du modèle semi-formel désambiguïsé, le processus de conversion en ontologie du domaine « 2.3 » produit une nouvelle ontologie en interprétant chacun des éléments de l'ontologie désambiguïsée en élément correspondant dans l'ontologie du domaine. Par exemple, un concept dans le modèle semi-formel sera transformé en classe dans l'ontologie du domaine. Dans sa forme finale, l'ontologie du domaine importe l'ontologie cadre assurant ainsi une caractérisation des éléments de l'ontologie du domaine (voir le tableau 2.9 et le tableau 2.10).

2.7.2.4 Exemple de formalisation d'un modèle semi-formel

Le tableau 2.9 présente un cas de figure de modélisation semi-formelle à formaliser. Les résultats de la réalisation des étapes de modélisation (voir le tableau 2.9 **a** et **b**); d'importation (voir le tableau 2.9 **c**), de désambiguïsation (voir le tableau 2.9 d) et de conversion (tableau 2.10) y sont respectivement représentés en langage MOT et OWL dans la Notation 3 (N3) (Berners-Lee, 1998) (voir l'appendice E pour un aperçu de la Notation 3).

Tableau 2.9
Études de cas représentés en langage MOT et en langage OWL-N3.

Cas 1) Désambiguïsation typologique	Cas 2) Désambiguïsation topologique	Cas 3) Désambiguïsation sémantique
a) Diagramme semi-formelle du domaine		
Relations d'instanciation et de subsomption	Un principe en tant que propriété ou agent	Relations de composition ou d'attribution
		
b) Représentation semi-formelle du domaine en notation XMI		

Cas 1)

```
<?xml version="1.0" encoding="UTF-8"?>
<mot:ModeleMot                    xmi:version="2.0"                    xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mot="www.cotechnoe.com/mot">
  <connaissances xsi:type="mot:Concept" nom="C1" id="C1_21">
    <relations     xsi:type="mot:LienI"     source="//@connaissances.0"     target="//@connaissances.1"
ontoRefStereotype="NON_DETERMINE"/>
  </connaissances>
  <connaissances xsi:type="mot:Exemple" nom="I1" id="I1_22" ontoRefStereotype="NON_DETERMINE"/>
  <connaissances xsi:type="mot:Concept" nom="C2" id="C2_23">
    <relations     xsi:type="mot:LienS"     source="//@connaissances.2"     target="//@connaissances.3"
ontoRefStereotype="NON_DETERMINE"/>
  </connaissances>
  <connaissances xsi:type="mot:Concept" nom="C3" id="C3_24"/>
</mot:ModeleMot>
```

Cas 2)

```
<?xml version="1.0" encoding="UTF-8"?>
<mot:ModeleMot                    xmi:version="2.0"                    xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mot="www.cotechnoe.com/mot">
  <connaissances xsi:type="mot:Concept" nom="C4" id="C4_25">
    <relations     xsi:type="mot:LienR"     source="//@connaissances.0"     target="//@connaissances.1"
ontoRefStereotype="NON_DETERMINE"/>
  </connaissances>
  <connaissances xsi:type="mot:Principe" nom="P1" id="P1_26">
    <relations     xsi:type="mot:LienR"     source="//@connaissances.1"     target="//@connaissances.2"
ontoRefStereotype="NON_DETERMINE"/>
  </connaissances>
  <connaissances xsi:type="mot:Concept" nom="C5" id="C5_27"/>
  <connaissances xsi:type="mot:Concept" nom="C6" id="C6_50"/>
  <connaissances xsi:type="mot:Principe" nom="A1" id="A1_51">
    <relations xsi:type="mot:LienR" source="//@connaissances.4" target="//@connaissances.3"/>
  </connaissances>
</mot:ModeleMot>
```

Cas 3)

```
<?xml version="1.0" encoding="UTF-8"?>
<mot:ModeleMot                    xmi:version="2.0"                    xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mot="www.cotechnoe.com/mot">
  <connaissances xsi:type="mot:Concept" messageErreur="" nom="C7" id="C7_0">
    <relations         xsi:type="mot:LienC"         messageErreur=""         source="//@connaissances.0"
target="//@connaissances.1"/>
    <relations         xsi:type="mot:LienC"         messageErreur=""         source="//@connaissances.0"
target="//@connaissances.2" ontoRefStereotype="Attribut"/>
  </connaissances>
  <connaissances xsi:type="mot:Concept" messageErreur="" nom="C8" id="C8_1"/>
  <connaissances        xsi:type="mot:Concept"        messageErreur=""        nom="C9"        id="C9_2"
ontoRefStereotype="Entite_Schema_String"/>
</mot:ModeleMot>
```

/…

…/

c) Représentation après importation dans l'espace de modélisation ontologique en notation -3

Cas 1)

```
:C1_0 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette        "C1"^^xsd:string ;...
:C2_2 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette        "C2"^^xsd:string ;...
:C3_3 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette        "C3"^^xsd:string ;...
:I1_1 a metaMot:MOT_Exemple ;
      metaMot:MOT_etiquette        "I1"^^xsd:string ;...
:LienI_C1_0_I1_1 a metaMot:MOT_LienI ;
      metaMot:MOT_connDestination :I1_1 ;
      metaMot:MOT_connSource      :C1_0 ;
      metaMot:MOT_nomLien         "LienI_C1_0_I1_1"^^xsd:string ;...
:LienS_C2_2_C3_3 a metaMot:MOT_LienS ;
      metaMot:MOT_connDestination :C3_3 ;
      metaMot:MOT_connSource      :C2_2 ;
      metaMot:MOT_nomLien         "LienS_C2_2_C3_3"^^xsd:string ;...
```

Cas 2)

```
:A1_4 a metaMot:MOT_Principe ;
      metaMot:MOT_etiquette        "A1"^^xsd:string ; ...
:C4_0 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette        "C4"^^xsd:string ; ...
:C5_2 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette        "C5"^^xsd:string ; ...
:C6_3 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette         "C6"^^xsd:string ; ...
:LienR_A1_4_C6_3 a metaMot:MOT_LienR ;
      metaMot:MOT_lrConnDest       :C6_3 ;
      metaMot:MOT_lrConnSrc        :A1_4 ; ...
:LienR_C4_0_P1_1 a metaMot:MOT_LienR ;
      metaMot:MOT_lrConnDest       :P1_1 ;
      metaMot:MOT_lrConnSrc        :C4_0 ; ...
:LienR_P1_1_C5_2 a metaMot:MOT_LienR ;
      metaMot:MOT_lrConnDest       :C5_2 ;
      metaMot:MOT_lrConnSrc        :P1_1 ; ...
:P1_1 a metaMot:MOT_Principe ;
      metaMot:MOT_etiquette        "P1"^^xsd:string ; ...
```

Cas 3)

```
:C7_0 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette            "C7"^^xsd:string ;...
:C8_1 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette            "C8"^^xsd:string ; ...
:C9_2 a metaMot:MOT_Concept ;
      metaMot:MOT_etiquette           "C9"^^xsd:string ; ...
:LienC_C7_0_C8_1 a metaMot:MOT_LienC ;
      metaMot:MOT_connDestination            :C8_1 ;
      metaMot:MOT_connSource             :C7_0 ; ...
:LienC_C7_0_C9_2 a metaMot:MOT_LienC ;
      metaMot:MOT_connDestination           :C9_2 ;
      metaMot:MOT_connSource            :C7_0 ; ...
```

/…

…/

d) Représentation après la désambiguïsation en notation-3	
Cas 1)	:C1_0 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C1"^^xsd:string ; ... **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; ... :C2_2 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C2"^^xsd:string ; ... **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; ... :C3_3 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C3"^^xsd:string ; ... ot:OT_EntiteEstDeType oAmbig:Concept_Classe ; ... :I1_1 a metaMot:MOT_Exemple ; metaMot:MOT_etiquette "I1"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Observable_Objet** ; ... :LienI_C1_0_I1_1 a metaMot:MOT_LienI ; metaMot:MOT_connDestination :I1_1 ; metaMot:MOT_connSource :C1_0 ; **ot:OT_EntiteEstDeType oAmbig:Relation_Instanciation** ; ... :LienS_C2_2_C3_3 a metaMot:MOT_LienS ; metaMot:MOT_connDestination :C3_3 ; metaMot:MOT_connSource :C2_2 ; **ot:OT_EntiteEstDeType oAmbig:Relation_Specialisation** ; ...
Cas 2)	:A1_4 a metaMot:MOT_Principe ; metaMot:MOT_etiquette "A1"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Agent_Contrainte_Norme** ; :C4_0 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C4"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; :C5_2 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C5"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; :C6_3 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C6"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; :LienR_A1_4_C6_3 a metaMot:MOT_LienR ; metaMot:MOT_lrConnDest :C6_3 ; metaMot:MOT_lrConnSrc :A1_4 ; **ot:OT_EntiteEstDeType oAmbig:Relation_Regulation** ; :LienR_C4_0_P1_1 a metaMot:MOT_LienR ; metaMot:MOT_lrConnDest :P1_1 ; metaMot:MOT_lrConnSrc :C4_0 ; **ot:OT_EntiteEstDeType oAmbig:Relation_Regulation** ; :LienR_P1_1_C5_2 a metaMot:MOT_LienR ; metaMot:MOT_lrConnDest :C5_2 ; metaMot:MOT_lrConnSrc :P1_1 ; metaMot:MOT_nomLien "LienR_P1_1_C5_2"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Relation_Regulation** ; :P1_1 a metaMot:MOT_Principe ; metaMot:MOT_etiquette "P1"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Propriete** ; ot:OT_nomEntiteDestination "C5_2"^^xsd:string ; ot:OT_nomEntiteSource "C4_0"^^xsd:string ;
Cas 3)	:C7_0 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C7"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; ... :C8_1 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C8"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Concept_Classe** ; ... :C9_2 a metaMot:MOT_Concept ; metaMot:MOT_etiquette "C9"^^xsd:string ; **ot:OT_EntiteEstDeType oAmbig:Entite_Schema_String** ; ... :LienC_C7_0_C8_1 a metaMot:MOT_LienC ; metaMot:MOT_connDestination :C8_1 ; metaMot:MOT_connSource :C7_0 ; **ot:OT_EntiteEstDeType oAmbig:Holonyme** ; ... :LienC_C7_0_C9_2 a metaMot:MOT_LienC ; metaMot:MOT_connDestination :C9_2 ; metaMot:MOT_connSource :C7_0 ; **ot:OT_EntiteEstDeType oAmbig:Attribut** ; ...

Chacun des cas est associé à un type de désambiguïsation particulier. Ainsi, le premier cas, qui présente un modèle composé de liens de subsomption et d'instanciation, fait appel à une désambigüisation de type *typologique* qui classe: les *Concepts* C1, C2, C3 en tant qu'`owl:Class`, avec C2 qui subsume C3; et l'*Exemple* I1 en tant qu'individu OWL appartenant à C1. Le deuxième cas, qui fait référence à une

désambiguïsation *topologique,* permet de désambiguïser les *Principes* P1 et A1 respectivement en `owl:ObjectProperty` et `owl:Class` de catégorie *Agent_Contrainte_Norme.* Finalement, le troisième cas nécessite une compréhension du domaine afin de désambiguïser de façon *sémantique* l'interprétation du *LienDeComposition*, qui, entre C7 et C8, se formalise par une `owl:ObjectProperty` de catégorie *A-POUR-COMPOSANT* dont le domaine est l'`owl:Class` C7 et l'`owl:Class` C8; et entre C7 et C9 qui se formalise par une `owl:DatatypeProperty` de catégorie *A-POUR-ATTRIBUT* dont le domaine est l'`owl:class` C7 et l'image une `xsd:string`.

Tableau 2.10
Modèle semi-formel formalisé en OWL-N3

Cas 1)	:C1_0 a owl:Class ; rdfs:label "C1"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :C2_2 a owl:Class ; rdfs:label "C2"^^xsd:string ; rdfs:subClassOf :C3_3 . :C3_3 a owl:Class ; rdfs:label "C3"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :I1_1 a :C1_0 ; rdfs:label "I1"^^xsd:string .
Cas 2)	:A1_4 a owl:Class ; rdfs:label "A1"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme . :A1_4_regit_C6_3 a owl:ObjectProperty ; rdfs:domain :A1_4 ; rdfs:label ""^^xsd:string ; rdfs:range :C6_3 ; rdfs:subPropertyOf metaDom:REGIT . :C4_0 a owl:Class ; rdfs:label "C4"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :C4_0_P1_1_C5_2 a owl:ObjectProperty ; rdfs:domain :C4_0 ; rdfs:label "P1"^^xsd:string ; rdfs:range :C5_2 ; rdfs:subPropertyOf :P1_1 . :C5_2 a owl:Class ; rdfs:label "C5"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :C6_3 a owl:Class ; rdfs:label "C6"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :P1_1 a owl:ObjectProperty ; rdfs:label "P1"^^xsd:string ; rdfs:subPropertyOf metaDom:MD_PROPRIETE .
Cas 3)	:C7_0 a owl:Class ; rdfs:label "C7"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :C7_0_aPourAttribut_C9_2 a ObjectProperty ; rdfs:domain :C7_0 ; rdfs:label "C9"^^xsd:string ; rdfs:range :C9_2 ; rdfs:subPropertyOf metaDom:A-POUR-ATTRIBUT . :C7_0_aPourComposant_C8_1 a owl:ObjectProperty ; rdfs:domain :C7_0 ; rdfs:label ""^^xsd:string ; rdfs:range :C8_1 ; rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT . :C8_1 a owl:Class ; rdfs:label "C8"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . :C9_2 a owl:Class ; rdfs:label "C9"^^xsd:string ; rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_String .

2.7.3 Méthode de validation de l'ontologie de domaine

En ingénierie ontologique, la validation est une étape essentielle à toute production d'ontologies. Gómez-Pérez (2004) présente les critères selon lesquels l'ontologie devrait être validée. La *consistance* concerne l'absence de contradictions entre les éléments ontologiques. La *complétude* assure que tous les éléments ontologiques sont soit explicitement déclarés, soit inférables. La *concision* est un principe qui stipule que seuls les éléments à être définis doivent être définis. L'*expansibilité* est la capacité d'ajouter de nouvelles connaissances sans modifier les anciennes. Finalement, la *sensibilité* est la capacité de l'ontologie à réagir à des modifications. En général, peu importe le domaine modélisé, il nous apparaît que les critères de validation s'appliquent aux aspects syntaxique et sémantique des connaissances qui sont représentées. Dans un modèle, qu'il soit semi-formel ou formel, on utilise un vocabulaire, une grammaire et une sémantique pour représenter les connaissances associées à un domaine. Le vocabulaire détermine les symboles qui sont utilisés pour représenter. La grammaire stipule les règles d'utilisation des symboles. La sémantique en définit le sens. Pendant la formalisation d'un modèle semi-formel, certaines opérations peuvent modifier des éléments de vocabulaire (le critère syntaxique) ou altérer le sens de la représentation (le critère sémantique). Il importe donc d'implanter un mécanisme de validation qui assure qu'aucune altération de la syntaxe et de la sémantique du modèle ne soit induite par le processus de transformation.

La méthode de validation, schématisée à la figure 2.10, régie par l'ingénieur, l'expert et l'assistant informatique intelligent à la validation, comporte un processus de *validation syntaxique « 3.1 »* et un processus de *validation sémantique « 3.2 »*. Pour son accomplissement, la méthode nécessite en intrant le modèle semi-formel de domaine et son correspondant ontologique, l'ontologie du langage de modélisation semi-formel. Après son exécution, la méthode produit un rapport de validation syntaxique ainsi qu'un rapport de validation sémantique.

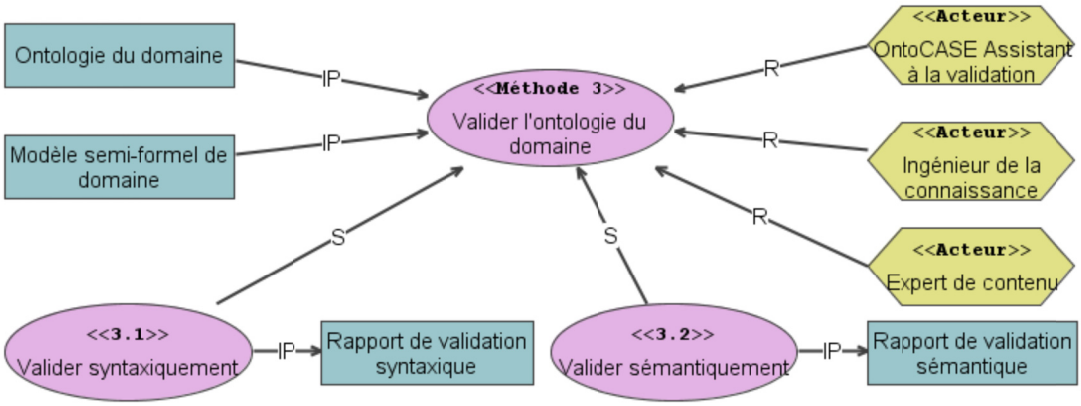


Figure 2.10: Méthode de validation de l'ontologie du domaine.

2.7.3.1 Processus de validation syntaxique

Le but de la validation syntaxique est de fournir au cogniticien un mécanisme de détection des erreurs informatiques qui peuvent survenir dans la production de l'ontologie cible. Deux types d'erreurs peuvent provoquer une anomalie syntaxique. L'erreur de regénération est causée par une erreur de programmation du module de regénération et doit être corrigée par le développeur. L'erreur de formalisation est causée par une erreur de programmation dans un des modules de l'assistant à la formalisation. Là encore, l'erreur doit être rapportée au développeur.

Comme son nom l'indique, la validation syntaxique concentre les efforts sur l'inspection des composants de vocabulaire et de grammaire du modèle d'origine. L'idée soutenant ce processus est que tous les éléments et relations du modèle d'origine doivent être représentés dans l'ontologie du domaine. De cette idée, nous supposons qu'un modèle reconstruit à partir de l'ontologie du domaine devrait être identique au modèle d'origine en termes d'entités et de relations. Schématisé à la figure 2.11, le sous-processus, *générer un modèle semi-formel,* produit un modèle semi-formel reconstruit à partir de l'ontologie du domaine. En comparant le modèle semi-formel d'origine avec le modèle semi-formel reconstruit, le sous-processus

comparer les modèles produit un rapport de validation syntaxique qui exprime les éventuelles anomalies syntaxiques entre les modèles.

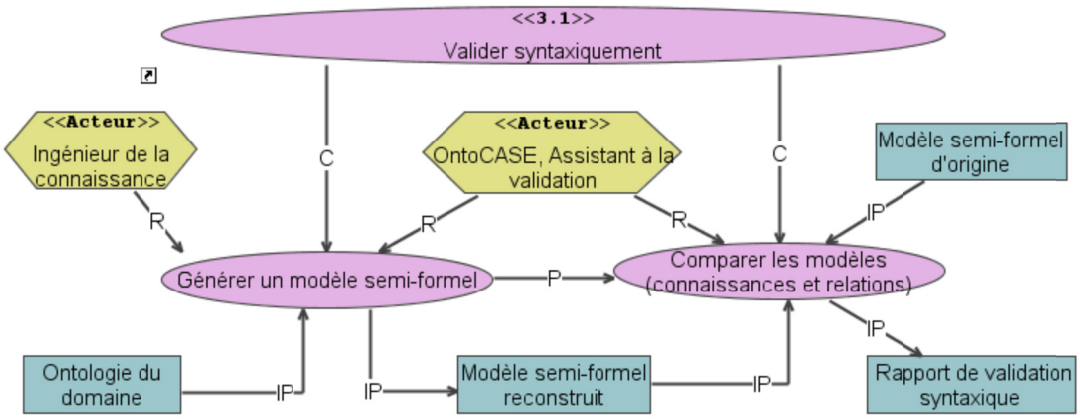


Figure 2.11: Processus de validation syntaxique.

2.7.3.2 Processus de validation sémantique

La validation sémantique est un processus qui s'intéresse à la signification du modèle. Le modèle représente-t-il bien la réalité qu'il désigne? L'*interprétation du modèle semi-formel,* régie par l'expert, permet de produire une interprétation humaine du modèle semi-formel. Quant au processus d'*interprétation automatique de l'ontologie,* il produit, à partir de l'ontologie du domaine, une interprétation automatique. Pour ce faire, le sous-processus utilise un moteur d'inférence pour générer de nouvelles conclusions qui serviront d'intrants au sous-processus de *comparaison des interprétations.* La comparaison des interprétations, qui est sous la responsabilité du l'expert de contenu, sert à la production du rapport d'interprétation. C'est sur la base de ce rapport que la décision de poursuivre ou non la démarche par une nouvelle itération de formalisation sera prise, soit pour modifier le rapport semi-formel ou soit encore pour améliorer son correspondant sous forme d'ontologie.

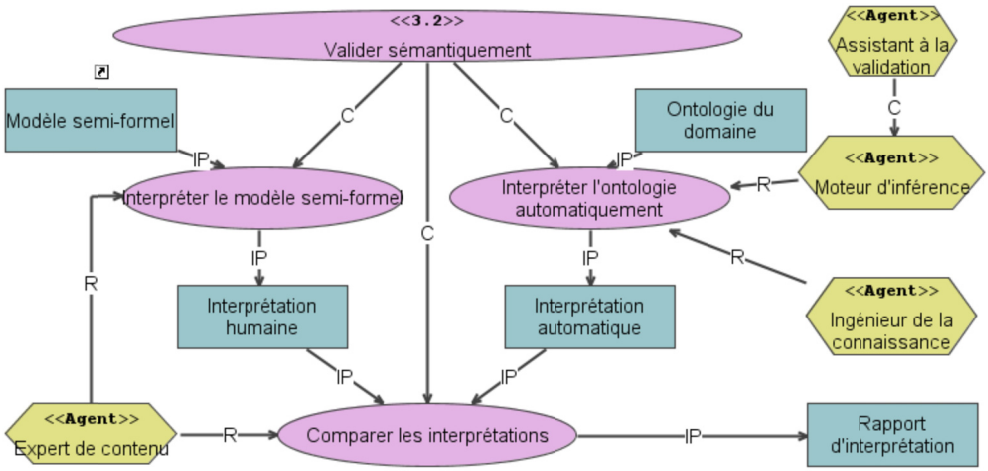


Figure 2.12: Processus de validation sémantique.

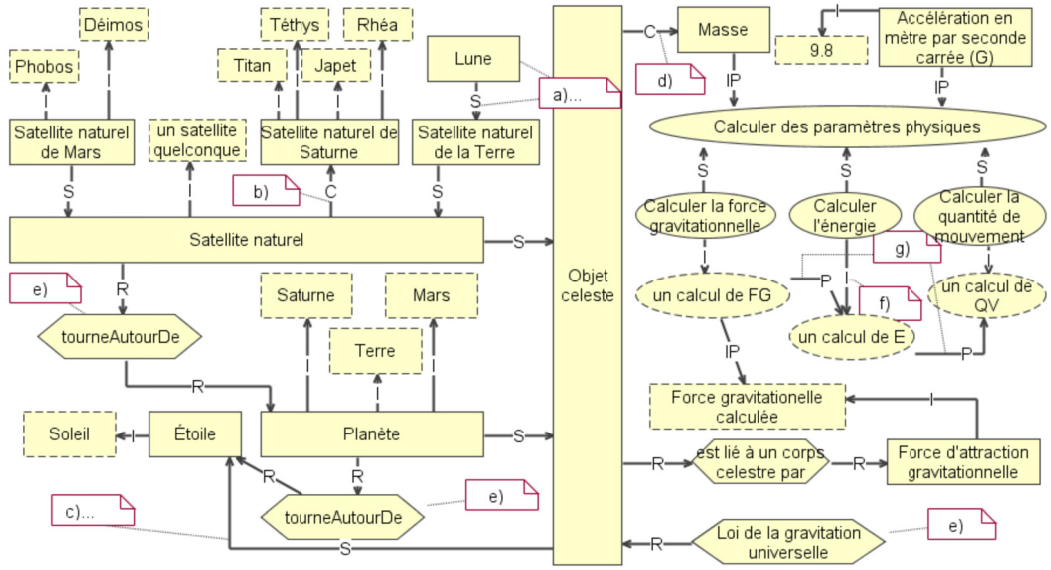
2.7.3.3 Exemple de validation syntaxique et sémantique

Examinons le modèle semi-formel suivant et son interprétation possible en langage naturel dont le sujet est l'Objet céleste (voir le tableau 2.11). La signification des étiquettes (de **a** à **g**) est détaillée à la section 4.3.6 p.198.

La validation syntaxique a pour objectif de s'assurer que chaque élément du modèle semi-formel est représenté dans l'ontologie du domaine. Pour mesurer cette concordance, un modèle semi-formel est généré à partir de l'ontologie du domaine. Le décompte des éléments de chacun des modèles et la comparaison des résultats permettent de vérifier la représentation de chaque élément du modèle semi-formel dans l'ontologie du domaine. Ainsi, pour le modèle présenté au tableau 2.11, le système identifie que le modèle se compose de 11 concepts, 4 procédures, 4 principes, 13 exemples, 3 traces, 3 *lienIP*, 7 *lienR*, 5 *lienC*, 9 *lienS*, 16 *lienI*, 2 *lienP* et 0 *lienA*, 0 *lienCm*, et 0 énoncé (voir le rapport de validation syntaxique au tableau 2.12).

Tableau 2.11
L'Objet céleste: le modèle semi-formel et son interprétation possible en langage naturel

a) Le modèle semi-formel



b) Interprétation possible en langage naturel

L'objet céleste est régi par le principe de la loi de la gravitation universelle. Il a pour attribut une masse et est lié à d'autres corps célestes par la force d'attraction gravitationnelle (FG). Cette force est calculée à partir de la masse et de l'accélération gravitationnelle (G) qui est de 9.8 m/s^2. D'autres paramètres physiques peuvent être calculés tels que l'énergie (E) et la quantité de mouvement (QV). Un calcul de la FG suivi d'un calcul de E et d'un calcul de QV sont des exemples spécifiques de calcul de paramètres physiques. Les étoiles, les planètes et les satellites naturels sont des corps célestes. Les planètes tournent autour des étoiles et les satellites naturels tournent autour des planètes. Saturne, Mars et la Terre sont des planètes, Déimos, Phobos sont des satellites naturels de Mars; Titan, Téthys, Japet et Rhéa sont des satellites naturels de Saturne. Finalement, la Lune est un satellite naturel de la Terre. Il existe aussi, quelque part, un satellite quelconque.

L'idée maîtresse de la validation sémantique est de comparer les conclusions inférées humainement à partir du modèle semi-formel avec les conclusions inférées automatiquement à partir de l'ontologie du domaine (voir le rapport de validation sémantique au tableau 2.13). La première phase de la validation est l'analyse de l'ontologie. Dans la taxonomie de classes de cette ontologie, le concept de Objet-céleste est subsumé par celui d'Étoile, ce qui voudrait dire que la Terre, qui est une sorte de Planète, serait considérée comme une Étoile. Cette erreur de classification provient de l'inversion de la direction du *LienS* (étiquette **c** de la figure 1). Le moteur

d'inférence appliqué à l'ontologie fera apparaître automatiquement cette incohérence qui pourra être détectée par l'utilisateur. Ainsi, l'utilisateur pourra corriger le modèle semi-formel initial.

Dans le cas de figure représenté par l'étiquette **b**, on pourrait interpréter que Phobos, en tant que satellite naturel, se compose d'un satellite naturel de Saturne tel que Titan. Or, une inférence automatique nous indiquera que Titan EST-LA-PARTIE-DE d'un satellite naturel alors qu'il est plus exact de dire que Titan et Phobos sont tous les deux des satellites naturels en remplaçant le *lienC* par un *lienS*.

Tableau 2.12
Bilan du rapport de validation syntaxique du modèle L*'Objet Céleste*

```
---------------------- BILAN --------------------------------
-----------------------------------------------------------
---------Totaux des divers composants des modèles ----------
Compte des Concepts  : Original = 11, Reconstruit = 11, Différence = 0
Compte des Procedures: Original = 4, Reconstruit = 4, Différence = 0
Compte des Principes : Original = 4, Reconstruit = 4, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 3, Reconstruit = 3, Différence = 0
Compte des Exemples  : Original = 13, Reconstruit = 13, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 16, Reconstruit = 16, Différence = 0
Compte des LienIP    : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienP     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienR     : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienS     : Original = 9, Reconstruit = 9, Différence = 0
```

La formalisation du cas de figure présenté à l'étiquette a, indique que la classe Lune est une sorte de satellite naturel de la Terre. Mécaniquement, nous pourrions instancier des individus de type Lune, ce qui, en termes représentationnels, est faux puisque la lune est un objet et non une classe d'objets.

L'étiquette **d** présente le cas où un *lienC* est formalisé en tant qu'attribut. Ce lien sera désambigüisé grâce à la distinction entre les métaliens A-POUR-COMPOSANT et A-POUR-ATTRIBUT.

L'étiquette **e** présente deux cas d'utilisation du Principe. Le premier des cas, Loi de la gravitation universelle, est désambiguïsé sous la forme owl:class de catégorie Agent_Contrainte_Norme. Le deuxième cas, tourneAutourDe, est formalisé en owl:ObjectProperty dont le domaine et l'image correspondent à la classe source et cible du Principe.

L'étiquette **f** indique que les entités « un calcul de FG », « un calcul de E » et « un calcul de QV », sont des traces des connaissances procédurales « calculer la force gravitationnelle », « calculer l'énergie » et « calculer la quantité de mouvement ». L'étiquette g présente un cas de figure fort intéressant puisqu'il concerne le traitement ontologique de connaissances procédurales. Après l'inférence, grâce aux méta-propriétés définies dans l'ontologie de référence, on aura que, le calcul de la quantité de mouvement A-POUR-DÉPENDANCE l'exécution des étapes « calculer la force gravitationnelle » et « calculer l'énergie ».

Tableau 2.13
Rapport de validation sémantique généré par OntoCASE

```
----------Début du processus de validation sémantique-------------
Accélération en mètre par seconde carrée (G) est un intrant de Calculer des paramètres physiques
Accélération en mètre par seconde carrée (G) est une sorte de (metaDom:MD_Declarative_Concept)
Calculer des paramètres physiques est une sorte de (metaDom:MD_Procedurale_Procedure)
Calculer l'énergie est une sorte de (metaDom:MD_Procedurale_Procedure)
Calculer l'énergie est une sorte de Calculer des paramètres physiques
Calculer la force gravitationnelle est une sorte de (metaDom:MD_Procedurale_Procedure)
Calculer la force gravitationnelle est une sorte de Calculer des paramètres physiques
Calculer la quantité de mouvement est une sorte de (metaDom:MD_Procedurale_Procedure)
Calculer la quantité de mouvement est une sorte de Calculer des paramètres physiques
Force d'attraction gravitationnelle est une sorte de (metaDom:MD_Declarative_Concept)
Loi de la gravitation universelle est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Loi de la gravitation universelle régit Objet celeste
Lune est une sorte de Satellite naturel de la Terre
Masse est un intrant de Calculer des paramètres physiques
Masse est une sorte de (metaDom:MD_Declarative_Concept)
Objet celeste a pour attribut Masse
Objet celeste est lié à un corps célestre par Force d'attraction gravitationnelle
Objet celeste est une sorte de Étoile
Planète est une sorte de Objet céleste
Planète tourneAutourDe Étoile
Satellite naturel de Mars est une sorte de Satellite naturel
Satellite naturel de Saturne est une sorte de (metaDom:MD_Declarative_Concept)
Satellite naturel de la Terre est une sorte de Satellite naturel
Satellite naturel est une sorte de Objet celeste
Satellite naturel se compose de Satellite naturel de Saturne
Satellite naturel tourneAutourDe Planète
[9.8] est de catégorie (metaDom:MD_Declarative_Concept)
[9.8] est un ( :Accélération en mètre par seconde carrée (G) )
[Déimos] est de catégorie (metaDom:MD_Declarative_Concept)
[Déimos] est un ( :Satellite naturel de Mars )
[Force gravitationelle calculée] est de catégorie (metaDom:MD_Declarative_Concept)
[Force gravitationelle calculée] est un ( :Force d'attraction gravitationnelle )
[Japet] est de catégorie (metaDom:MD_Declarative_Concept)
[Japet] est un ( :Satellite naturel de Saturne )
[Mars] est de catégorie (metaDom:MD_Declarative_Concept)
[Mars] est un ( :Planète )
[Phobos] est de catégorie (metaDom:MD_Declarative_Concept)
[Phobos] est un ( :Satellite naturel de Mars )
[Rhéa] est de catégorie (metaDom:MD_Declarative_Concept)
[Rhéa] est un ( :Satellite naturel de Saturne )
```

a) Devrait être un fait

b) Le principe est une classe …

e) … et une propriété

c) Le *lienC* signifiant 'attribut'

d) Devrait être lié par un *lienS*



2.8 En résumé

Dans ce chapitre, nous avons présenté la méthodologie OntoCASE de conception d'une ontologie à partir d'un modèle semi-formel d'un domaine de connaissances. Les volets méthodologique, représentationnel et computationnel de cette méthodologie sont harmonisés afin d'offrir une méthodologie complète et supportée par un assistant à la formalisation. Cet assistant appuie l'ingénieur dans le déroulement de la méthode de formalisation et le supporte dans la validation syntaxique et sémantique de l'ontologie du domaine. L'ontologie de transformation qui fait partie du volet

représentationnel et computationnel de la méthodologie se compose d'une ontologie du langage semi-formel, d'une ontologie de référence et d'une ontologie cadre. Un ensemble de documents qui servent de guides aux différents acteurs complète le volet représentationnel de la méthodologie. La méthodologie intègre les principes de consensualité et de formalité propres aux ontologies par l'utilisation de la méthode de comodélisation pour la production de modèles semi-formels. L'éditeur de modèle semi-formel eLi permet la conception d'un modèle semi-formel dans un langage interopérable et utilisable par OntoCASE. Finalement, la méthodologie permet la formalisation de connaissances déclaratives, procédurales, stratégiques et factuelles. Le chapitre suivant présente la démarche de recherche qui a permis la construction d'OntoCASE.

CHAPITRE 3

DÉMARCHE DE CONSTRUCTION D'ONTOCASE

Le présent chapitre décrit la démarche adoptée pour développer OntoCASE. Nous avons vu dans le chapitre précédent qu'OntoCASE comporte un volet méthodologique, un volet représentationnel et un volet informatique. De plus, les composants procéduraux et architecturaux d'OntoCASE y ont été identifiés. La liste qui suit spécifie les activités et les produits de la démarche de construction d'OntoCASE:

- Déterminer :
    - le contenu de l'ontologie de référence;
    - le contenu de l'ontologie du langage semi-formel;
    - le contenu de l'ontologie cadre;
    - les règles de désambiguïsation;
    - les règles de formalisation.
- Développer l'architecture et l'implantation des modules :
    - d'édition de modèles semi-formels;
    - d'importation;
    - de désambiguïsation;
    - de formalisation;
    - de validation syntaxique;
    - de validation sémantique.
- Concevoir :
    - le manuel des principes de modélisation;
    - le catalogue de désambiguïsation.

OntoCASE s’est avéré un produit complexe à développer pour plusieurs raisons. D'abord, son développement devait être orienté dans la perspective de permettre de confirmer ou non l'hypothèse de cette thèse. De plus, sa double vocation méthodologique et informatique imposait un développement intégré et conjoint de ces deux composants. Spécifiquement associé au volet informatique, le développement de l'assistant exigait un processus de conception qui implique à la fois un volet déclaratif (par le développement des ontologies nécessaires à la transformation) et un volet procédural (par le développement du code nécessaire à l'exécution des éléments méthodologiques). Les facettes déclarative et procédurale de l'assistant devaient être développées en parfaite synergie.

Les étapes de la démarche de recherche ont été segmentées en trois phases. La première phase, la *mise en place*, a pour objectif de fixer les composants informatiques en lien avec les composants méthodologiques présentés au chapitre précédent. La deuxième phase, l'*agrégation*, permet d'attacher les différents composants entre eux et de valider les choix architecturaux et théoriques. La troisième phase, la *consolidation*, permet de compléter la structure interne des différentes ontologies agrégées par l'ontologie de transformation. Cette phase assure aussi la mise en place des outils informatiques nécessaires à la validation d’OntoCASE par l'implantation de mécanismes d'exécution de scénarios de tests.

Compte tenu des particularités de l'objet de la démarche, nous avons préconisé l'utilisation d'une approche évolutive et itérative pour la réalisation de la démarche de recherche. La première étape de la réalisation de cette approche est la généralisation d'activités pouvant s'appliquer à de multiples phases de la démarche. La figure 3.1 schématise l'approche itérative de la démarche. En son centre, représenté dans la forme rectangulaire, on retrouve l'objet de la démarche, soit OntoCASE avec ses composants procéduraux et déclaratifs. Autour d'OntoCASE, figurent les activités qui, au besoin, sont invoquées pendant les phases de réalisation de la démarche et qui

permettent la construction de l'objet de la démarche. Ces activités sont associées à l'une ou l'autre des trois phases de la recherche que nous allons maintenant décrire.

Trois phases ont ponctué l'évolution de la démarche de construction d'OntoCASE:

- La phase 1, *Mise en place des composants architecturaux, procéduraux et informatiques de la méthodologie* a permis d'esquisser la phase initiale de la démarche. Pendant cette phase, nous esquissons le volet procédural et architectural de la méthodologie. Concurremment au développement de ces deux volets, nous avons identifié les technologies et outils informatiques apte à supporter le développement de l'assistant informatique. À la fin de cette phase, toutes les incertitudes concernant les aspects technologiques étaient levées et une première esquisse des éléments architecturaux et procéduraux était réalisée.
- La phase 2, *Agrégation des composants ontologiques, procéduraux et informatiques d'OntoCASE,* a constitué la phase de développement des modules de l'assistant, de la construction des ontologies liées à la transformation et de la rédaction du catalogue des ambiguïtés ainsi que de l'harmonisation de ces constituants avec les aspects procéduraux de la méthodologie. Cette phase a constitué le cœur de la recherche. C'est pendant son déroulement que la validité des aspects théoriques a été confirmée.
- La phase 3, *Confirmation,* a constitué l'étape de la démarche au cours de laquelle la fonctionnalité d'OntoCASE fut testée et raffinée. Plusieurs modèles semi-formels, comportant plusieurs cas de modélisation associés à des sémantiques variées, furent alors soumis à OntoCASE. C'est aussi pendant cette phase qu'ont été rédigés le document décrivant les *éléments pour guider la modélisation semi-formelle à l'aide de concepts ontologiques* (appendice E).

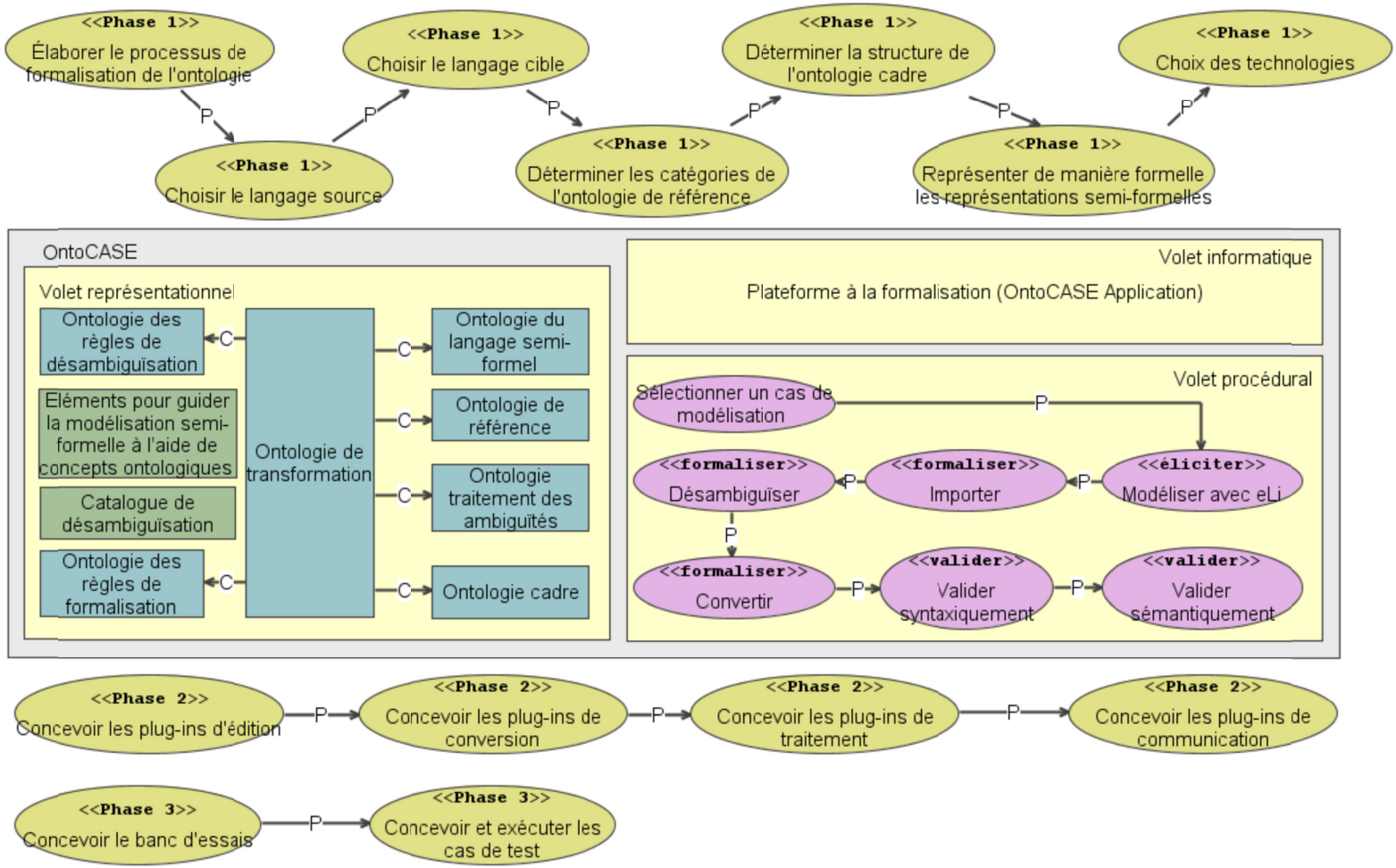


Figure 3.1: Objets et activités de la démarche de construction d'OntoCASE.

3.1 Phase 1: Mise en place des composants représentationnels, procéduraux et informatique de la méthodologie.

Cette première phase a pour objectif de mettre en place les différentes composantes de la méthodologie tel qu'indiqué en haut de la figure 3.1. L'élaboration du processus de formalisation a pour but de déterminer la structure procédurale utilisée dans la méthodologie pour formaliser une ontologie à partir d'une conceptualisation semi-formelle. Le choix du langage source et du langage cible détermine le cadre langagier utilisé par la méthodologie. La détermination du contenu de l'ontologie de référence, de l'ontologie cadre et de l'ontologie du langage source (par exemple MOT) fixe le volet représentationnel de la méthodologie. Finalement, le choix de l'environnement informatique de développement et des outils d'ingénierie aptes à manipuler les ontologies utilisées permettent de fixer le volet computationnel de la méthodologie.

3.1.1 Élaborer le processus de formalisation de l'ontologie

Tel qu'indiqué à la section 1.7.3 p.71 pour Kendal et Creen (2007) ainsi que pour Uschold et Gruninger (1996), le développement d'une ontologie se divise en six étapes : l'étude de faisabilité, l'acquisition de connaissances, la conception, l'implémentation, la validation et la maintenance. Chez Gómez-Pérez *et al* (2003), la méthodologie de développement d'ontologies intègre plusieurs composants de la gestion de projets. Pour Davies *et al* (2003), le développement d'ontologies est un processus instrumentalisé par l'utilisation de composants informatiques spécifiques. La méthode d'ingénierie ontologique présentée à la figure 1.40 p.72, proposée par Staab *et al.* (2001), présente un processus inspiré des méthodes itératives de développement de logiciels. Dans tous les cas, les étapes importantes du processus de conception d'une ontologie sont l'élicitation et la formalisation. En référence à ce processus de développement d'ontologie, OntoCASE concentre ses éléments méthodologiques sur les étapes de démarrage (*Ontology kickoff*), de raffinement et d'évaluation. Pour OntoCASE, nous parlerons plutôt d'étapes centrées sur les

méthodes de *conception d'un modèle semi-formel,* de méthode de *formalisation* et de méthode de *validation* (voir la section 2.7 p.97). La méthode de conception d'un modèle semi-formel implique un processus de modélisation d'un modèle semi-formel instrumenté par l'éditeur de modèle semi-formel eLi. La méthode de formalisation, qui est au cœur de cette recherche, se divise en trois processus: *l'importation*, la *désambiguïsation* et la *transformation*. L'assistant à la formalisation est l'application informatique intelligente qui supporte la réalisation de cette étape. Finalement, la méthode de validation permet d'évaluer la facette syntaxique et sémantique de la formalisation. Là aussi, un assistant informatique supporte cette méthode.

3.1.2 Choix du langage cible : l'*Ontology Web Language*

Pour la réalisation de la démarche, nous avons choisi d'utiliser la saveur DL de l'*Ontology Web Language*[64] *(OWL)* normalisé par le *World Wide Web Consortium* (W3C), en tant que langage cible à la méthodologie de formalisation. Le choix de OWL-DL *versus* OWL-Lite ou OWL-Full se justifie, d'une part, par l'expressivité OWL-DL qui est supérieure à celle OWL-Lite et, d'autre part, par la décidabilité qu'assure OWL-DL et qui n'est pas nécessairement assurée par OWL-Full. Outre OWL plusieurs autres formalismes ontologiques sont aussi disponibles (Ontolingua[65] , LOOM[66], FLogic[67] OKBC[68], OCML[69], etc). Cependant, notre choix s'est arrêté sur OWL pour des raisons de standardisation et de grande disponibilité des outils

---

[64] McGuinness et Harmelen, *OWL Web Ontology Language Overview* http://www.w3.org/TR/owl-features/; Patel-Schneider, Hayes et Horrocks, *OWL Web Ontology Language Semantics and Abstract Syntax* http://www.w3.org/TR/owl-semantics/; W3C OWL, *Ontology Web Language*: http://www.w3.org/2004/OWL/

[65] *Ontolingua Home Page*: http://www.ksl.stanford.edu/software/ontolingua/

[66] *Loom Project Home Page*: http://www.isi.edu/isd/LOOM/

[67] Meštrović et Čubrilo, *F-Logic Data and Knowledge Reasoning in the Semantic Web Context*

[68] *Open Knowledge Base Connectivity (OKBC)*: http://www.ksl.stanford.edu/software/OKBC/

[69] *Operational Conceptual Modelling Language (OCML)*: http://technologies.kmi.open.ac.uk/ocml//

informatiques (voir la section 3.1.5 p.134) notamment associés au développement du Web sémantique.

3.1.3 Choix du langage source: le langage de modélisation par objets typés MOT

Issu du domaine de l'ingénierie pédagogique, le langage de modélisation par objets typés MOT, de degré semi-formel, est celui qui a été utilisé pour développer la méthodologie proposée. Le langage et le logiciel qui l'implémente (MOT*Plus*) ont été conçus par une équipe sous la direction de Paquette (2002b). Une version détaillée du langage est présentée à l'appendice A de cette thèse. Notons que notre but est de faire en sorte qu'OntoCASE puisse s'appliquer à d'autres langages semi-formels tels que ceux utilisés pour les cartes conceptuelles, les réseaux sémantiques ou les diagrammes UML.

Une caractéristique importante d'un langage semi-formel, est qu'il existe plus d'une façon de représenter un même objet. C'est ce que nous avons appelé la synonymie (voir la section 1.1.6 p.16). Une conséquence directe de la synonymie est que pour une primitive du langage, il existe plusieurs significations possibles. C'est ce que nous avons appelé la polysémie. Les tableaux qui suivent présentent quelques situations de synonymies de MOT. De même, le tableau 3.6 présente la polysémie de MOT.

3.1.3.1 Étude de la synonymie de MOT[70]

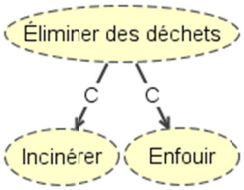
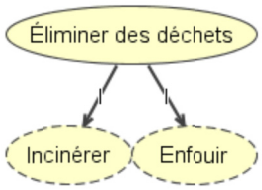
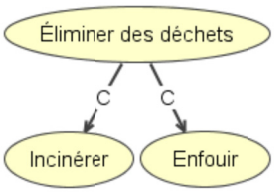
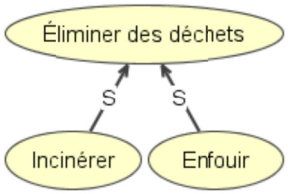
La synonymie d'agrégation intervient lorsque qu'une situation transporte l'idée d'agglomérat ou de collection d'objets. L'idée d'agrégation peut s'exprimer de plusieurs façons en langage MOT. Illustré au tableau 3.1, le texte en en-tête du tableau se représente selon l'un des quatre modèles, soit par une composition selon le niveau d'abstraction factuelle (voir le tableau 3.1**a**), soit par le changement de niveau

[70] Cette section est extraite de l'appendice E

d'abstraction (voir le tableau 3.1**b**), soit par l'utilisation de la composition au niveau d'abstraction conceptuel (voir le tableau 3.1**c**), ou encore par l'utilisation de la spécialisation (voir le tableau 3.1**d**).

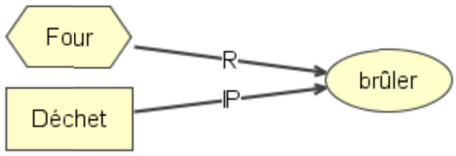
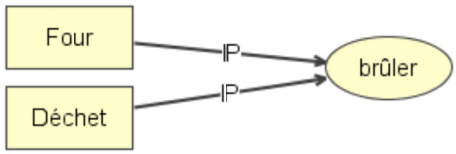
Tableau 3.1
Représenter une agrégation en MOT

L'élimination des déchets se fait de deux façons principales: l'incinération et l'enfouissement

Représentations possibles en MOT	
a) 	b) 
c) 	d) 

L'exemple du tableau 3.2 présente une situation où un processus est réalisé par un instrument. L'instrument peut être un objet ou une personne. Dans le premier cas **a**, l'instrument, qui est représenté par un principe, est considéré dans ce contexte comme un agent. Selon la définition du langage MOT (Paquette, 2002b), l'agent est l'une des significations possibles d'un principe (voir le tableau 3.6). Dans cette représentation, l'agent « Four » régit le processus de « brûler ». Dans la deuxième représentation (en **b)**, l'instrument « Four » est un concept qui est uni au processus « brûler » par un lien I/P. Dans ce contexte, le lien I/P doit être interprété par « est-l'instrument-qui-réalise » et non comme l'intrant du processus « brûler » comme c'est normalement le cas.

Tableau 3.2
Représenter l'utilisation d'un instrument en MOT

	brûler les déchets dans un four
Représentations possibles en MOT	a) Le four (qui est un agent) régit le processus de brûler 
	b) Le four est l'instrument qui réalise le processus de brûler 

Le tableau 3.3 présente un modèle selon le niveau d'abstraction conceptuel et factuel. Lors d'une activité de modélisation, le choix du niveau d'abstraction pour représenter une connaissance dépend des réponses qui sont attendues par l'interprétation du modèle (Allemang et Hendler, 2008) surtout, lorsqu'il s'agit d'une modélisation en vue de produire une ontologie interprétable par un agent cognitif formel. Afin d'assurer la complétude[71] et la décidabilité[72] du modèle, trois questions servent de guide quand vient le choix du niveau d'abstraction pour représenter une connaissance (Mariot *et al.*, 2008): la question de la *satisfiabilité* (soit C un concept quelconque, est-ce qu'il existe des instances de C?); la question de la *vérification d'instances* (est-ce que *a* est une instance de C?) et la question de *subsomption* (est-ce que C est plus général de D?). Si l'on se réfère au tableau 3.4 **a** qui est l'interprétation formelle[73] du modèle conceptuel du tableau 3.3 **a,** on constate que toutes les connaissances et les

[71] Nous entendons par complétude la capacité qu'un modèle possède à représenter sans contradiction l'ensemble de la sémantique d'un système.

[72] Nous entendons par décidabilité la propriété que possède un modèle à être interprété par un agent cognitif formel afin de produire des conclusions en un nombre fini d'opérations et dans un temps fini.

[73] L'interprétation formelle du modèle a été réalisée par le module de validation sémantique d'OntoCASE.

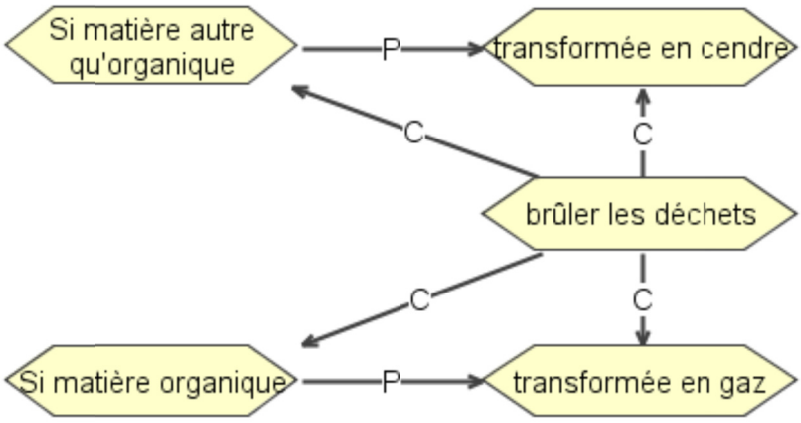
relations y sont interprétées. Cependant, aucune déduction automatique n'a été produite. En revanche, le modèle factuel du tableau 3.3 **b** a permis la production d'une quantité importante de nouvelles connaissances (voir le tableau 3.4 **b**). Ceci établi, cette démonstration n'invalide en rien la modélisation selon le niveau conceptuel. Cependant, quand on modélise, il faut garder à l'esprit le degré de complétude que l'on désire maintenir dans le modèle et faire le choix de la représentation selon le niveau d'abstraction adéquat.

Tableau 3.3
Représenter selon le niveau d'abstraction en MOT

Brûler les déchets ... la matière organique est alors transformée en gaz tandis que le reste des déchets devient des cendres

Représentations possibles en MOT

a) représentation de niveau conceptuel



b) représentation de niveau factuel

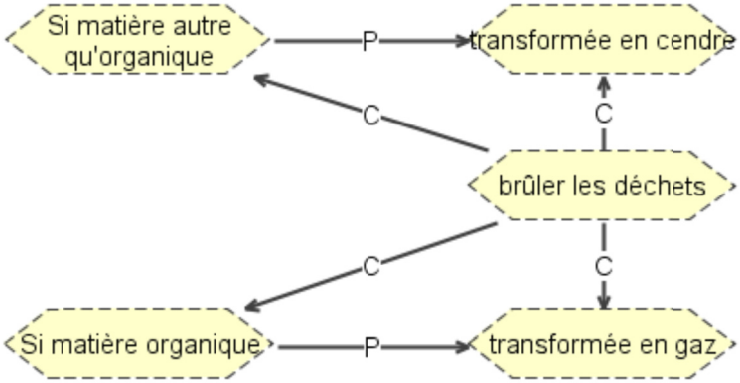


Tableau 3.4
Interprétation formelle de la règle brûler des déchets

a) interprétation du niveau conceptuel

```
----------Début du processus de validation sémantique-------------
Si matière autre qu'organique a pour conclusion transformée en
cendre
Si matière organique a pour conclusion transformée en gaz
bruler les déchets se compose de Si matière autre qu'organique
bruler les déchets se compose de Si matière organique
bruler les déchets se compose de transformée en cendre
bruler les déchets se compose de transformée en gaz
------------------Résultats après inférence----------------------
```

b) interprétation du niveau factuel

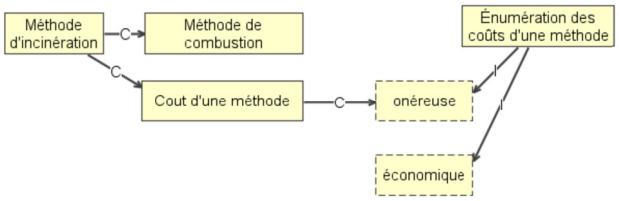
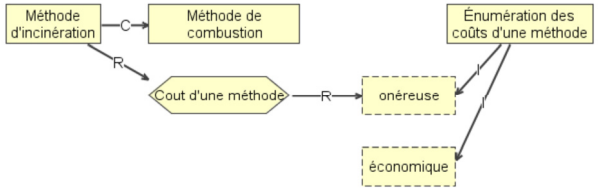
```
----------Début du processus de validation sémantique-------------
[Si matière autre qu'organique] a pour conclusion [transformée en
cendre]
[Si matière autre qu'organique] est de catégorie
(metaDom:MD_Strategique_Entite_Regle_Antecedent)
[Si matière organique] a pour conclusion [transformée en gaz]
[bruler les déchets] est un ( :Nom d'une règle )
[bruler les déchets] se compose de [Si matière autre qu'organique]
[bruler les déchets] se compose de [Si matière organique]
[bruler les déchets] se compose de [transformée en cendre]
[bruler les déchets] se compose de [transformée en gaz]
------------------Résultats après inférence----------------------
[Si matière autre qu'organique] alors [transformée en cendre]
[Si matière autre qu'organique] est une partie de [bruler les
déchets]
[Si matière autre qu'organique] permet [transformée en cendre]
[Si matière organique] alors [transformée en gaz]
[Si matière organique] est une partie de [bruler les déchets]
[Si matière organique] permet [transformée en gaz]
[transformée en cendre] a pour dépendance [Si matière autre
qu'organique]
[transformée en cendre] est la conclusion de [Si matière autre
qu'organique]
[transformée en cendre] est une partie de [bruler les déchets]
[transformée en gaz] a pour dépendance [Si matière organique]
[transformée en gaz] est la conclusion de [Si matière organique]
[transformée en gaz] est une partie de [bruler les déchets]
```

En langage MOT, l'association d'un *attribut* à un concept ne fait pas partie des primitives atomiques de MOT. Il existe tout de même deux façons semi-formelles pour représenter un attribut. La première, qui est présentée au tableau 3.5 **a** utilise un concept associé par un lien C à un concept symbolisant l'attribut. Dans l'exemple (en **a)** le concept « coût de la méthode » est un attribut du concept « Méthode

d'incinération » alors que le concept « Méthode de combustion » symbolise une composante de la « Méthode d'incinération ». Dans cette façon de représenter l'attribut, il n'existe aucune manière formelle de distinguer la composition et l'attribut.

L'exemple du tableau 3.5 **b** présente une alternative un peu plus formelle pour représenter un attribut. L'utilisation du principe en tant que propriété de type "dataType" permet de faire une distinction claire entre la composition de concepts et l'affectation d'un attribut à un concept.

Tableau 3.5
Représenter un attribut en MOT

L'incinération, qui comprend une méthode de combustion, est la méthode la plus onéreuse	
Représentations possibles en MOT	a) attribut par l'utilisation d'un concept lié par un lien C 
	b) attribut par l'utilisation d'un principe 

3.1.3.2 Étude de la polysémie de MOT

Une conséquence très importante de l'utilisation d'un même symbole dans une multitude de contextes est qu'il y a surcharge de la sémantique symbolisée par le symbole (ce que nous avons appelé la *polysémie*). Le tableau 3.6 présente la polysémie que nous avons identifiée pour chacune des primitives atomiques du langage MOT.

Tableau 3.6
Polysémie de MOT

<table>
<tr><th colspan="3">ENTITÉ</th><th colspan="2">RELATION</th></tr>
<tr><th colspan="2">Représentation en MOT</th><th>Signification (s)</th><th>Représentation en MOT</th><th>Signification (s)</th></tr>
<tr><td rowspan="12">Caonnaissacne abstraite</td><td rowspan="4">concept
Connaissance déclarative</td><td colspan="2">Classe</td><td rowspan="4">--C->
Composition</td><td>Est composé de</td></tr>
<tr><td rowspan="3">Schéma</td><td>Entier</td><td rowspan="3">A pour attribut</td></tr>
<tr><td>Caractère</td></tr>
<tr><td>Naturel</td></tr>
<tr><td rowspan="3">procédure
Connaissance procédurale</td><td colspan="2">Action</td><td rowspan="3">-IP->
Intrant/Produit</td><td>Est l’intrant de</td></tr>
<tr><td colspan="2" rowspan="2">Opération</td><td>A pour produit</td></tr>
<tr><td>Est l'instrument utilisé</td></tr>
<tr><td rowspan="4">principe
Connaissance stratégique</td><td colspan="2">Agent</td><td>--R->
Régulation</td><td>Régit</td></tr>
<tr><td colspan="2">Propriété</td><td rowspan="4">--P->
Précédence</td><td>Puis exécuter</td></tr>
<tr><td colspan="2">Classe de règle</td><td>Puis évaluer</td></tr>
<tr><td colspan="2">Classe de condition</td><td>Permet</td></tr>
<tr><td rowspan="8">Connaissance factuelle</td><td rowspan="2">exemple
Connaissance déclarative</td><td colspan="2">Objet</td><td>Évaluer à partir de</td></tr>
<tr><td colspan="2">Valeur</td><td>A pour dépendance</td></tr>
<tr><td rowspan="2">trace
Connaissance procédurale</td><td colspan="2">Acte</td><td>--I->
Instance</td><td>A pour instance</td></tr>
<tr><td colspan="2">Instruction</td><td>--S->
Spécialisation</td><td>Est une sorte de</td></tr>
<tr><td rowspan="4">énoncé
Connaissance stratégique</td><td colspan="2">Individu</td><td>--A->
Application</td><td>A pour application</td></tr>
<tr><td colspan="2">Assertion</td><td rowspan="3">Icône
englobement</td><td rowspan="3">Association avec les éléments d’un sous-modèle</td></tr>
<tr><td colspan="2">Condition</td></tr>
<tr><td colspan="2">Règle</td></tr>
</table>

Comme nous le verrons un peu plus loin dans la thèse, le rôle du processus de désambiguïsation est de permettre à l'utilisateur de choisir le sens approprié au symbole qu'il utilise dans le modèle semi-formel.

3.1.4 Déterminer les catégories de l'ontologie de référence

La construction de l'ontologie de référence implique une analyse de plusieurs systèmes de représentation. Afin de construire une ontologie de référence la plus générique possible, l'analyse doit inclure l'étude d’un ensemble de langages servant à

représenter des connaissances de type tant déclaratif que procédural et stratégique, qui sont les trois types de connaissances que l'on retrouve dans tout domaine de connaissances. Chacun des éléments de vocabulaire doit être étiqueté afin de faciliter une classification ultérieure dans l'ontologie de référence. La nomenclature des étiquettes divise le nom de l'étiquette en deux champs délimités par le caractère deux-points (:). Le premier champ indique le nom du langage concerné et le deuxième champ indique l'élément du vocabulaire. Ainsi, le lien de composition faisant partie du vocabulaire du langage MOT serait étiqueté par le libellé suivant 'mot:lienC'.

3.1.4.1 Vocabulaire du langage de MOT

Le langage MOT (Paquette, 2002b p. 73) a pour vocabulaire les éléments présentés au tableau 3.7. MOT est un langage important en représentation de connaissances, car la modalité de ce langage permet la représentation de connaissances de type procédural, déclaratif, stratégique et factuel. De ce fait, la structure de MOT s'impose d'elle-même en tant que prototype de structure pour la construction de l'ontologie de référence.

Tableau 3.7
Vocabulaire du langage MOT

Entité		**Relation**
Connaissances abstraites	Concept (mot:concept)	Composition/Composition multiple (mot:lienC)
	Procédure (mot:procedure)	Spécialisation (mot:lienS)
	Principe (mot:principe)	Instanciation (mot:lienI)
Connaissance factuelle	Exemple (mot:exemple)	Application (mot:lienA)
	Trace (mot:trace)	Régulation (mot:lienR)
	Énoncé (mot:enonce)	Précédence (mot:lienP)
		Intrant/Produit (mot:lienIP)
		Englobement (mot:lienE)

3.1.4.2 Vocabulaire du langage UML

Le tableau 3.8, le tableau 3.9 et le tableau 3.10 présentent respectivement le vocabulaire des diagrammes UML de classes, de cas d'utilisation et du diagramme d'état.

Tableau 3.8
Vocabulaire du diagramme UML de classes (tiré de Rhem, 2006)

Entité	Relation
Classe (uml_class:classe)	Héritage (uml_class:relHerit)
Attribut (uml_class:attrib)	Composition (uml_class:relComp)
Opération (uml_class:oper)	Agrégation (uml_class:relAgr)
Interface (uml_class:interface)	Association (uml_class:relAss)
	Dépendance (uml_class:relDep)

Tableau 3.9
Vocabulaire du diagramme UML de cas d'utilisation (tiré de Rhem, 2006)

Entité	Relation
Acteur (uml_cu:acteur)	Communication (uni et bi)directionnelle (uml_cu:relComm)
Cas d'utilisation (uml_cu:cu)	Inclusion (uml_cu:relIncl)
	Extension (uml_cu:relExt)
	Généralisation (uml_cu:relGen)

Tableau 3.10
Vocabulaire du diagramme UML d'état (tiré de Rhem, 2006)

Entité	Relation
Action (uml_etat:action)	Flux de contrôle (uml_etat:fluxControle)
État (uml_etat:etat)	Flux d'objets (uml_etat:fluxObj)
Synchronisation de flux (uml_etat:sync)	
État de départ (uml_etat:depart)	
État de fin (uml_etat:fin)	
Décision (uml_etat:decision)	
Objet (uml_etat:objet)	

3.1.4.3 Vocabulaire du langage BPMN

Le tableau 3.11 présente le vocabulaire du *Business Process Management Notation* (BPMN). Chacune des entités d'un modèle BPMN peut représenter une entité abstraite ou concrète.

Tableau 3.11
Vocabulaire du diagramme BPMN (tiré de OMG BPDM, 2007 ; Weske, 2007)

Entité	Relation
Activité (bpmn:act)	Flux de séquence (bpmn:relSeq)
Événement (bpmn:even)	Flux de message (bpmn:relMess)
Commutateur [Décision] (bpmn:commut)	Association (bpmn:relAss)
Couloir d'activité (bpmn:coul_act)	Flux d'événement (bpmn:relEven)
Participant (bpmn:parti)	
Artefact (bpmn:arte)	

3.1.4.4 Classification du vocabulaire des langages dans l'ontologie de référence

Chacun des langages présentés ci-haut possède un vocabulaire qui lui est propre, dont plusieurs propriétés peuvent être partagées par des éléments de vocabulaire appartenant à d'autres langages. Cataloguées au tableau 3.12 et au tableau 3.13, les entités et relations de chacun des langages sont classées dans une ou plusieurs catégories qui, une fois synthétisés, contribuent à définir l'ontologie de référence.

Tableau 3.12
Classification des entités des divers langages pour chacune des catégories d'entités de l'ontologie de référence

Catégories des entités de la réalité abstraite		**Catégories des entités de la réalité concrète**	
Classe	mot:concept; uml_class:classe; uml_class:interface; bpmn:even; bpmn:coul_act; bpmn:arité	Objet	mot:exemple; uml_class:classe; uml_etat:objet; bpmn:even; bpmn:coul_act; bpmn:arité
Schéma	mot:concept; uml_class:attrib	Attribut	mot:exemple; uml_class:attrib
Opération	mot:procedure; uml_class:oper; uml_cu:cu	Opération	mot:trace; uml_class:oper; uml_cu:cu; uml_etat:etat; uml_etat:sync; uml_etat:depart; uml_etat:fin
Procédure	mot:procedure; bpmn:act	Procédure	mot:trace; uml_etat:action; bpmn:act
Agent	mot:principe; uml_cu:acteur; bpmn:parti	Agent	mot:enonce; uml_cu:acteur; bpmn:parti
Condition	mot:principe; bpmn:commut	Condition	mot:enonce; uml_etat:decision; bpmn:commut
Règle nom	mot:principe	Règle nom	mot:enonce
Règle antécédent	mot:principe	Règle antécédent	mot:enonce
Règle conclusion	mot:principe	Règle conclusion	mot:enonce
Règle non-décomposée	mot:principe	Règle non-décomposée	mot:enonce
Propriété	mot:principe; uml_class:attrib; uml_class:relAss; bpmn:relAss	Assertion	mot:enonce

Certains éléments de vocabulaire sont catalogués dans plus d'une catégorie. Il s'agit alors d'éléments comportant une sémantique ambigüe et qui devront subir une désambiguïsation.

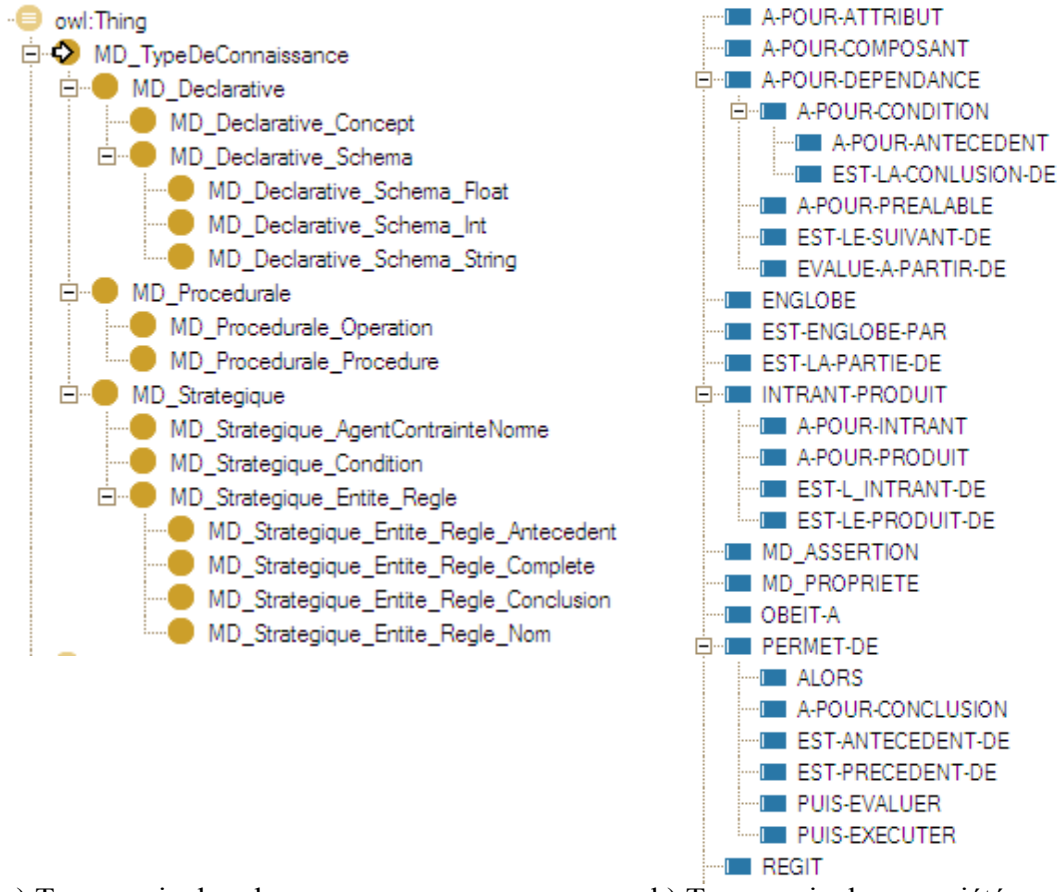
Tableau 3.13
Classification des relations de divers langages pour chacune des catégories de langage de l'ontologie de référence

Catégorie de relations	
Intrant	mot:lienIP; uml_etat:fluxObj; bpmn:relMess
Produit	mot:lienIP; uml_etat:fluxObj
Précédence	mot:lienP; uml_etat:fluxControle; bpmn:relSeq; bpmn:relEven
Agrégation	uml_class:relAgr
Composition	mot:lienC; uml_class:relComp
Englobe	mot:lienE; uml_class:relDep; uml_cu:relIncl
Généralisation	uml_class:relHerit; uml_cu:relGen
Spécialisation	mot:lienS
Définit	mot:lienA
Instance	mot:lienI
Régulation	mot:lienR; uml_cu:relExt

3.1.5 Déterminer la structure de l'ontologie cadre

Tel que déjà mentionné, l'*ontologie cadre* est l'ontologie qui structure la représentation des connaissances du domaine. Elle est utilisée en tant que composante de l'ontologie de transformation pour le processus de transformation. Lors de l'utilisation de l'ontologie du domaine, l'ontologie cadre est intégrée à l'ontologie du domaine par importation (`owl:imports`). Présentée à la figure 3.2**a**, la taxonomie des classes correspond à la structure du niveau subjectif de l'ontologie de référence (voir le tableau 4-1). La figure 3.2**b** présente la structure hiérarchique des propriétés de

l'ontologie cadre. Certaines propriétés sont définies en tant que propriétés inverses. Par exemple, la propriété EST-LA-PARTIE-DE est définie comme propriété inverse de A-POUR-COMPOSANT.



a) Taxonomie des classes b) Taxonomie des propriétés

Figure 3.2: Représentation taxonomique des classes et propriétés de l'ontologie cadre.

3.1.6 Représenter de manière formelle le vocabulaire de MOT

Grâce à l'ontologie de référence et à l'ontologie cadre, il est alors possible de représenter de manière formelle le vocabulaire du langage MOT. Ainsi, la correspondance formelle de la représentation de connaissances déclaratives, procédurales, stratégiques et mixtes (qui intègre plusieurs types de connaissances) est respectivement présentée au tableau 3.15, au tableau 3.16, ainsi qu'au tableau 3.17.

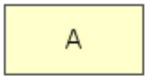
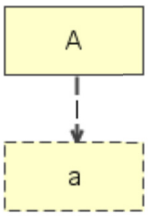
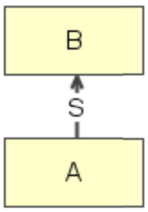
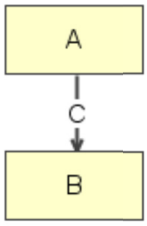
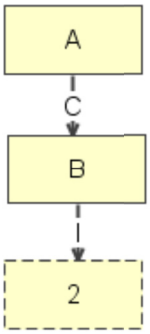
On retrouve à l'appendice B la référence complète de la représentation formelle du vocabulaire de MOT ainsi que les règles de désambiguïsation et de conversion permettant cette formalisation.

3.1.6.1 Représentation formelle de connaissances déclaratives

Le tableau 3.14**a** présente la représentation formelle d'un `mot:Concept`. La ligne 01 indique que l'entité est de type `owl:class` et que son nom est `A_0`. Chacun des objets du modèle semi-formel d'origine est représenté de manière unique dans l'ontologie cible. Ainsi, pour plusieurs objets possédant le même étiquette, le processus de transformation créera le même nombre de classes dans l'ontologie cible avec chacune leur *nom* unique composé d'une première partie par l'étiquette de l'objet et d'une deuxième partie par un nombre entier unique à l'ontologie cible. Les deux parties sont séparées par le caractère "_". Pendant la transformation, l'étiquette du `mot:Concept` est transposée dans la propriété `rdfs:label` de la classe cible (voir la ligne 02). Finalement, la ligne 03 indique que la classe `A_0` fait partie de la catégorie déclarative (`metaDom:MD_Declarative_Concept`).

Pour la formalisation d'une relation d'instanciation (voir le tableau 3.14**b**), le processus de transformation traduit le `mot:LienI` en `rdf:type`. De même, la transformation du `mot:LienS` est traduit par `rdfs:subClassOf` (voir le tableau 3.14**c** ligne 03). Grâce à la propriété de transitivité de `rdfs:subClassOf`, il n'est pas nécessaire de déclarer `A_4` en tant que sous-classe de `metaDom:MD_Declarative_Concept`.

Tableau 3.14
Correspondance formelle de représentations MOT des connaissances déclaratives

Représentation MOT	Représentation formelle
a) 	01. :A_0 a owl:Class ; 02. rdfs:label "A"^^xsd:string ; 03. rdfs:subClassOf metaDom:MD_Declarative_Concept .
b) 	01. :A_1 a owl:Class ; 02. rdfs:label "A"^^xsd:string ; 03. rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept . 04. :a_2 a :A_1 ; 05. rdfs:label "a"^^xsd:string .
c) 	01. :A_4 a owl:Class ; 02. rdfs:label "A"^^xsd:string ; 03. rdfs:subClassOf :B_3. 04. :B_3 a owl:Class ; 05. rdfs:label "B"^^xsd:string ; 06. rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
d) 	01. :A_5 a owl:Class ; 02. rdfs:label "A"^^xsd:string ; 03. rdfs:subClassOf metaDom:MD_Declarative_Concept . 04. :B_6 a owl:Class ; 05. rdfs:label "B"^^xsd:string ; 06. rdfs:subClassOf metaDom:MD_Declarative_Concept . 07. :A_5_aPourComposant_B_6 08. a owl:ObjectProperty ; 09. rdfs:domain :A_5 ; 10. rdfs:range :B_6 ; 11. rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
e) 	01. :A_7 a owl:Class ; 02. rdfs:label "A"^^xsd:string ; 03. rdfs:subClassOf metaDom:MD_Declarative_Concept . 04. :B_8 a owl:Class ; 05. rdfs:label "B"^^xsd:string ; 06. rdfs:subClassOf metaDom:MD_Declarative_Schema_Int 07. metaDom:MD_isString "2"^^xsd:int . 08. :A_7_aPourAttribut_B_8 09. a owl:ObjectProperty ; 10. rdfs:domain :A_7 ; 11. rdfs:label "B"^^xsd:string ; 12. rdfs:range :B_8 ; 13. rdfs:subPropertyOf metaDom:A-POUR-ATTRIBUT . 14. :_2_9 15. a :B_8 ; 16. rdfs:label "2"^^xsd:string .

La relation de composition (`mot:LienC`) est un premier cas d'ambiguïté de la sémantique du langage MOT. Le `mot:LienC` est utilisé: soit pour représenter une relation d'holonymie[74] entre deux connaissances (voir le tableau 3.14**d**), soit pour représenter une relation d'attribution (voir le tableau 3.14**e**). Lorsque désambiguïsé en holonyme (voir le tableau 3.14**e**), le `mot:LienC` est interprété, dans l'ontologie cible, par une propriété dont le libellé est constitué d'une partie contenant le nom de la connaissance d'origine, le libellé "`aPourComposant`", suivi du libellé de la connaissance cible (voir la ligne 07). Le domaine et l'image de la propriété sont respectivement attribués à la connaissance source et à la connaissance cible de la relation (voir la ligne 09 et la ligne 11). La propriété est une sous-propriété de la méta-propriété de l'ontologie cadre (`metaDom:A-POUR-COMPOSANT`).

Lorsque la relation de composition est désambiguïsée en attribut (voir le tableau 3.14**f**), la propriété d'attribution est classée sous la méta-propriété `metaDom:-A-POUR-ATTRIBUT` (voir la ligne 13) et le concept définissant le type de l'attribut[75] (entier, réel, chaîne de caractères) est classé sous la méta-classe associée à son type (voir la ligne 06). En MOT, la valeur associée à l'attribut est représentée par la relation d'instanciation entre le concept et l'exemple. La représentation dans l'ontologie cible est assumée par un individu dont le type correspond à l'attribut.

3.1.6.2 Représentation formelle de connaissances procédurales

La représentation formelle de connaissances procédurales est présentée au tableau 3.15. En a), on observe que la classe associée à la procédure est une sous-classe de `metaDom:MD_Procedurale_Procedure` (voir la ligne 04). La relation d'instanciation (en **b**), la relation de spécialisation (en **c**), et la relation de composition

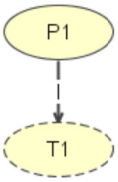
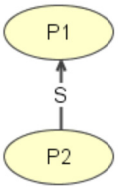
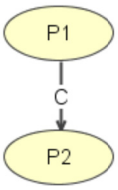
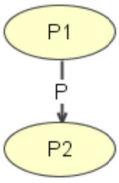
---

[74] *Terme lié à un autre de la même langue par une relation d'holonymie, c'est-à-dire de tout à partie. Un holonyme A d'un mot B est un mot dont le signifié désigne un ensemble comprenant le signifié de B. Corps est un holonyme de bras, de même que maison est un holonyme de toit. (*extrait de http://fr.wiktionary.org/wiki/holonyme, récupérée le 15 mars 2010).

[75] Aussi appelé schéma dans l'ontologie de référence et l'ontologie cadre.

(en **d**) sont traitées de la même manière que les relations associées aux connaissances déclaratives. Pour ce qui est de la relation de précédence (voir le tableau 3.15 **e**), celle-ci est formalisée par la propriété "`puisExecuter`" (voir la ligne 08) et classée sous la méta-propriété `metaDom:PUIS-EXECUTER` (voir la ligne 10).

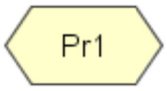
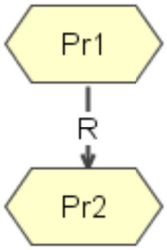
Tableau 3.15
Correspondance formelle de représentations MOT des connaissances procédurales

Représentation MOT	Représentation formelle
a) 	01. :P1_0 02. a owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure.
b) 	01. :P1_1 02. a owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure . 05. :T1_2 06. a :P1_1 ; 07. rdfs:label "T1"^^xsd:string .
c) 	01. :P1_5 02. a owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure , 05. :P2_6 06. a owl:Class ; 07. rdfs:label "P2"^^xsd:string ; 08. rdfs:subClassOf metaDom:MD_Procedurale_Procedure, :P1_5.
d) 	01. :P1_7 02. a owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure. 05. :P1_7_aPourComposant_P2_8 06. a owl:ObjectProperty ; 07. rdfs:domain :P1_7 ; 08. rdfs:range :P2_8 ; 09. rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT. 10. :P2_8 11. a owl:Class ; 12. rdfs:label "P2"^^xsd:string ; 13. rdfs:subClassOf metaDom:MD_Procedurale_Procedure.
e) 	01. :P1_3 02. a owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure . 05. :P2_4 06. a owl:Class ; 07. rdfs:label "P2"^^xsd:string ; 08. rdfs:subClassOf metaDom:MD_Procedurale_Procedure. 09. :P1_3_puisExecuter_P2_4 10. a owl:ObjectProperty ; 11. rdfs:domain :P1_3 ; 12. rdfs:range :P2_4 ; 13. rdfs:subPropertyOf metaDom:PUIS-EXECUTER .

3.1.6.3 Représentation formelle de connaissances stratégiques

Représenté sous la méta-classe `metaDom:MD_Strategique_AgentContrainteNorme` (voir le tableau 3.16**a**) ligne 04, le `mot:Principe` de type *agent/norme/contrainte* associé par un lien d'instanciation, de généralisation ou de composition utilise les mêmes principes de représentation formelle que vu précédemment. La relation de régulation (`mot:LienR`) utilisée avec le principe se formalise par la construction de la propriété "`regit`" (voir le tableau 3.16**b** ligne 09), classée sous la méta-propriété `metaDom:REGIT` (voir la ligne 14).

Tableau 3.16
Correspondance formelle de représentations MOT des connaissances stratégiques

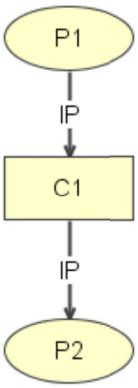
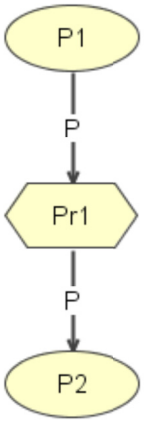
Représentation MOT	Représentation formelle
a) 	01. :Pr1_0 02. rdf:type owl:Class ; 03. rdfs:label "Pr1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Strategique_AgentContrainteNorme .
b) 	01. :Pr1_3 02. rdf:type owl:Class ; 03. rdfs:label "Pr1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Strategique_AgentContrainteNorme . 05. :Pr2_4 06. rdf:type owl:Class ; 07. rdfs:label "Pr2"^^xsd:string ; 08. rdfs:subClassOf metaDom:MD_Strategique_AgentContrainteNorme 09. :Pr1_3_regit_Pr2_4 10. rdf:type owl:ObjectProperty ; 11. rdfs:domain :Pr1_3 ; 12. rdfs:label ""^^xsd:string ; 13. rdfs:range :Pr2_4 ; 14. rdfs:subPropertyOf metaDom:REGIT .

3.1.6.4 Représentation formelle de connaissances mixtes

Certaines relations unissent des connaissances de type divers. Par exemple, un lien intrant/produit peut unir une connaissance procédurale à une connaissance déclarative, d'où le terme de représentation de connaissances mixtes. Le tableau 3.17 présente les principaux scénarios de représentation de connaissances mixtes que l'on retrouve en langage MOT. En **a**, le `mot:LienIP` est une relation qui unit une connaissance procédurale à une connaissance déclarative. Formalisé, le `mot:LienIP` qui

unit `P1` et `C1`, s'interprète par l'énoncé: `P1` produit `C1` (voir les lignes 05 et 09). Alors qu'entre `C1` et `P2`, la relation `mot:LienIP` s'interprète par l'énoncé `P2` a pour intrant `C1` (voir les lignes 14 et 15). Pour un modèle procédural, le lien de précédence s'utilise parfois entre une procédure et un principe (voir tableau 3.17**b**). Dans ce cas, le principe est interprété comme une condition à résoudre avant de poursuivre le flux d'exécution. Le `mot:LienP` entre `P1` et `Pr1` est formellement interprété par: *P1 puis évaluer Pr1* (voir les lignes 13 et 17); alors qu'entre `Pr1` et `P2`, le `mot:LienP` est interprété par: *Pr1 puis exécuter P1* (voir les lignes 18 et 23).

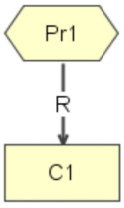
Tableau 3.17
Correspondance formelle de représentations MOT des connaissances mixtes

Représentation MOT	Représentation formelle
a) 	01. :P1_0 02. rdf:type owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure. 05. :P1_0_aPourProduit_C1_2 06. rdf:type owl:ObjectProperty ; 07. rdfs:domain :P1_0 ; 08. rdfs:range :C1_2 ; 09. rdfs:subPropertyOf metaDom:A-POUR-PRODUIT 10. :P2_1 11. rdf:type owl:Class ; 12. rdfs:label "P2"^^xsd:string ; 13. rdfs:subClassOf metaDom:MD_Procedurale_Procedure. 14. :P2_1_aPourIntrant_C1_2 15. rdf:type owl:ObjectProperty ; 16. rdfs:domain :P2_1 ; 17. rdfs:range :C1_2 ; 18. rdfs:subPropertyOf metaDom:A-POUR-INTRANT
b) 	01. :P1_0 02. rdf:type owl:Class ; 03. rdfs:label "P1"^^xsd:string ; 04. rdfs:subClassOf metaDom:MD_Procedurale_Procedure . 05. :P2_2 06. rdf:type owl:Class ; 07. rdfs:label "P2"^^xsd:string ; 08. rdfs:subClassOf metaDom:MD_Procedurale_Procedure. 09. :Pr1_1 10. rdf:type owl:Class ; 11. rdfs:label "Pr1"^^xsd:string ; 12. rdfs:subClassOf metaDom:MD_Strategique_Condition . 13. :P1_0_puisEvaluer_Pr1_1 14. rdf:type owl:ObjectProperty ; 15. rdfs:domain :P1_0 ; 16. rdfs:range :Pr1_1 ; 17. rdfs:subPropertyOf metaDom:PUIS-EVALUER . 18. :Pr1_1_puisExecuter_P2_2 19. rdf:type owl:ObjectProperty ; 20. rdfs:domain :Pr1_1 ; 21. rdfs:label ""^^xsd:string ; 22. rdfs:range :P2_2 ; 23. rdfs:subPropertyOf metaDom:PUIS-EXECUTER .

La relation `mot:LienR` s'utilise aussi avec une connaissance déclarative ou procédurale (voir le tableau 3.18).

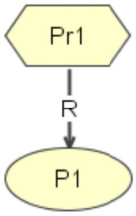
Tableau 3.18
Correspondance formelle de représentations MOT des connaissances mixtes associées par un lien de régulation

a)



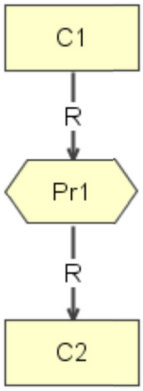

```
01.     :Pr1_0
02.           rdf:type owl:Class ;
03.           rdfs:label "Pr1"^^xsd:string ;
04.           rdfs:subClassOf owl:Thing ,
        metaDom:MD_Strategique_AgentContrainteNorme .
05.     :C1_1
06.           rdf:type owl:Class ;
07.           rdfs:label "C1"^^xsd:string ;
08.           rdfs:subClassOf owl:Thing ,
        metaDom:MD_Declarative_Concept .
09.     :Pr1_0_regit_C1_1
10.           rdf:type owl:ObjectProperty ;
11.           rdfs:domain :Pr1_0 ;
12.           rdfs:range :C1_1 ;
13.           rdfs:subPropertyOf metaDom:REGIT
```

b)




```
01.     :P1_1
02.           rdf:type owl:Class ;
03.           rdfs:label "P1"^^xsd:string ;
04.           rdfs:subClassOf metaDom:MD_Procedurale_Procedure ,
        owl:Thing .
05.     :Pr1_0
06.           rdf:type owl:Class ;
07.           rdfs:label "Pr1"^^xsd:string ;
08.           rdfs:subClassOf owl:Thing ,
        metaDom:MD_Strategique_AgentContrainteNorme .
09.     :Pr1_0_regit_P1_1
10.           rdf:type owl:ObjectProperty ;
11.           rdfs:domain :Pr1_0 ;
12.           rdfs:range :P1_1 ;
13.           rdfs:subPropertyOf metaDom:REGIT .
```

c)




```
01.    :C1_6
02.        rdf:type owl:Class ;
03.        rdfs:label "C1"^^xsd:string ;
04.        rdfs:subClassOf metaDom:MD_Declarative_Concept .
05.    :C2_19
06.        rdf:type owl:Class ;
07.        rdfs:label "C2"^^xsd:string ;
08.        rdfs:subClassOf metaDom:MD_Declarative_Concept .
09.    :Pr1_7
10.        rdf:type owl:ObjectProperty ;
11.        rdfs:label "Pr1"^^xsd:string ;
12.        rdfs:subPropertyOf metaDom:MD_PROPRIETE .
13.    :C1_6_Pr1_7_C2_19
14.        rdf:type owl:ObjectProperty ;
15.        rdfs:domain :C1_6 ;
16.        rdfs:label "Pr1"^^xsd:string ;
17.        rdfs:range :C2_19 ;
18.        rdfs:subPropertyOf :Pr1_7 .
```

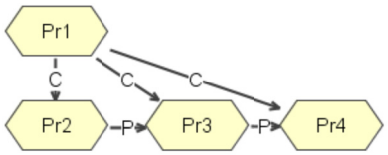
Au tableau 3.18 **a**, le `mot:Principe` est formalisé en une `owl:Class` de type `metaDom:MD_Strategique_AgentContrainteNorme` (voir la ligne 04). Cette `owl:Class Pr1` est reliée à la `owl:Class` procédurale `P1` et à la `owl:Class` déclarative `C1` par la propriété de

régulation `régit` (voir la ligne 13). En **c**, la situation est totalement différente. Le positionnement du `mot:Principe` entre deux `mot:Concept` réunis par des relations de régulation s'interprète par la formalisation du `mot:Principe` en une représentation `owl:ObjectProperty` (voir la ligne 09), qui elle-même est une sous-propriété de la méta-propriété `metaDom:MD_PROPRIETE` (voir la ligne 12) dont l'utilité est de regrouper les propriétés du domaine. À la propriété `Pr1_7`, une sous-propriété (`C1_6_Pr1_7_C2_19`) associée à la description de la situation est formalisée (voir la ligne 13). Cette formalisation assure la spécificité de la représentation par la définition du domaine (voir la ligne 15) et de l'image (voir la ligne 17) correspondant au `mot:Concept` d'origine et de destination.

Le tableau 3.19 et le tableau 3.20 présentent un scénario d'utilisation du `mot:LienC` et du `mot:LienP` en relation avec des `mot:Principe` et `mot:Procedure` qui permet la représentation de règles. Au tableau 3.19, le `mot:Principe Pr1` est formalisé en `Regle_Nom` (voir la ligne 04), alors que les `mot:Principe Pr2` et `Pr3` sont formalisés en `Antécédent` (voir les lignes 08 et 12). Pour sa part, `Pr4` est formalisé en `Conclusion` (voir la ligne 16). Chacun des `mot:LienC` est formalisé en propriété `aPourComposant` (voir les lignes 22, 28 et 34). Quant au lien de précédence `mot:LienP`, il est formalisé d'une part, en propriété `estAntécédentDe` pour le `mot:LienP` entre `Pr2` et `Pr3` (voir la ligne 40) et d'autre part, en propriété `aPourConclusion` pour le `mot:LienP` entre `Pr3` et `Pr4` (voir la ligne 46). En langage naturel, la règle représentée au tableau 3.19 s'interprète de la façon suivante: *la règle du nom Pr1 a pour antécédent Pr2 et Pr3; et pour conclusion Pr4*[76].

[76] Dans cette représentation, nous avons déterminé de manière arbitraire que le dernier élément de la chaîne de principes est une conclusion. Au besoin, Pr3 pourrait aussi représenter une conclusion.

Tableau 3.19
Correspondance formelle de représentations MOT des connaissances mixtes pour la formation d'une règle produisant une conclusion

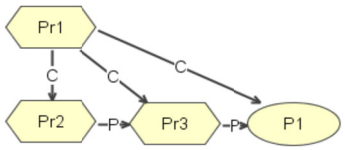



```
:Pr1_0
    rdf:type owl:Class ;
    rdfs:label "Pr1"^^xsd:string ;
    rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Nom .
:Pr2_1
    rdf:type owl:Class ;
    rdfs:label "Pr2"^^xsd:string ;
    rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent.
:Pr3_2
    rdf:type owl:Class ;
    rdfs:label "Pr3"^^xsd:string ;
    rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent .
:Pr4_3
    rdf:type owl:Class ;
    rdfs:label "Pr4"^^xsd:string ;
    rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Conclusion .
:Pr1_0_aPourComposant_Pr2_1
    rdf:type owl:ObjectProperty ;
    rdfs:domain :Pr1_0 ;
    rdfs:label ""^^xsd:string ;
    rdfs:range :Pr2_1 ;
    rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Pr1_0_aPourComposant_Pr3_2
    rdf:type owl:ObjectProperty ;
    rdfs:domain :Pr1_0 ;
    rdfs:label ""^^xsd:string ;
    rdfs:range :Pr3_2 ;
    rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Pr1_0_aPourComposant_Pr4_3
    rdf:type owl:ObjectProperty ;
    rdfs:domain :Pr1_0 ;
    rdfs:label ""^^xsd:string ;
    rdfs:range :Pr4_3 ;
    rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Pr2_1_estAntecedentDe_Pr3_2
    rdf:type owl:ObjectProperty ;
    rdfs:domain :Pr2_1 ;
    rdfs:label ""^^xsd:string ;
    rdfs:range :Pr3_2 ;
    rdfs:subPropertyOf metaDom:EST-ANTECEDENT-DE .
:Pr3_2_aPourConclusion_Pr4_3
    rdf:type owl:ObjectProperty ;
    rdfs:domain :Pr3_2 ;
    rdfs:label ""^^xsd:string ;
    rdfs:range :Pr4_3 ;
    rdfs:subPropertyOf metaDom:A-POUR-CONCLUSION .
```

Le traitement de la représentation de la règle du tableau 3.20 diffère peu de la précédente à l'exception du traitement de la représentation du résultat. Ici, le résultat est représenté par une connaissance procédurale qui est interprétée en tant

qu'opération[77] (voir la ligne 16). Le `mot:LienP` qui unit `Pr3` et `P1` est formalisé en propriété "`Alors`" (voir la ligne 37) sous la méta-propriété `metaDom:ALORS` (voir la ligne 41). En langage naturel, la représentation de cette règle s'interprète ainsi: *la règle du nom Pr1: si Pr2 et Pr3, alors P1.*

Tableau 3.20
Correspondance formelle de représentations MOT des connaissances mixtes pour la formation d'une règle résultant de l'exécution d'une opération




```
01.    :Pr1_0
02.        rdf:type owl:Class ;
03.        rdfs:label "Pr1"^^xsd:string ;
04.        rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Nom.
05.    :Pr2_1
06.        rdf:type owl:Class ;
07.        rdfs:label "Pr2"^^xsd:string ;
08.        rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent
09.    :Pr3_2
10.        rdf:type owl:Class ;
11.        rdfs:label "Pr3"^^xsd:string ;
12.        rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent .
13.    :P1_3
14.        rdf:type owl:Class ;
15.        rdfs:label "P1"^^xsd:string ;
16.        rdfs:subClassOf owl:Thing , metaDom:MD_Procedurale_Operation.
17.    :Pr1_0_aPourComposant_P1_3
18.        rdf:type owl:ObjectProperty ;
19.        rdfs:domain :Pr1_0 ;
20.        rdfs:range :P1_3 ;
21.        rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
22.    :Pr1_0_aPourComposant_Pr2_1
23.        rdf:type owl:ObjectProperty ;
24.        rdfs:domain :Pr1_0 ;
25.        rdfs:range :Pr2_1 ;
26.        rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
27.    :Pr1_0_aPourComposant_Pr3_2
28.        rdf:type owl:ObjectProperty ;
29.        rdfs:domain :Pr1_0 ;
30.        rdfs:range :Pr3_2 ;
31.        rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
32.    :Pr2_1_estAntecedentDe_Pr3_2
33.        rdf:type owl:ObjectProperty ;
34.        rdfs:domain :Pr2_1 ;
35.        rdfs:range :Pr3_2 ;
36.        rdfs:subPropertyOf metaDom:EST-ANTECEDENT-DE .
37.    :Pr3_2_alors_P1_3
38.        rdf:type owl:ObjectProperty ;
39.        rdfs:domain :Pr3_2 ;
40.        rdfs:range :P1_3 ;
41.        rdfs:subPropertyOf metaDom:ALORS .
```

[77] Ici encore, le terme *opération* a été arbitrairement choisi pour distinguer la procédure qui est exécutée en tant que conclusion d'une règle de la procédure qui est exécutée dans une séquence de procédures.

3.1.7 Choix de l'environnement informatique

Le volet informatique est une composante importante d'OntoCASE. Pour le développement de l'assistant informatique, nous avons choisi l'environnement de développement *Eclipse*[78] qui intègre plusieurs outils à base de développement conduit par les modèles tels que: l'*Eclipse Modeling Framework* (EMF[79]) pour la gestion de la pérennisation des données et le *Graphical Modeling Framework* (GMF[80]) pour la conception d'éditeurs graphiques.

3.1.7.1 Environnement *Eclipse*

Plusieurs applications concernant le Web sémantique sont développées avec la plateforme *Eclipse*, notamment: *Protégé 4.0*[81]*, TopBraid Composer*[82]*, NeOn Toolkit*[83] et *Eclipse Ontology Definition Metamodel (EODM) Workbench*[84].

Eclipse est une fondation à source ouverte, parrainée par IBM, qui regroupe un ensemble d'acteurs segmentant leurs activités en projets. Un projet majeur d'*Eclipse* est la conception d'environnements de développement d'applications écrites en langage *java, C, C++, Python, PHP,* etc. L'architecture d'*Eclipse* est construite sur le concept de *plug-in.* Le *plug-in* est un module greffé à un noyau. Le *plug-in* est conçu pour accomplir une tâche de manière autonome ou en collaboration avec d'autres *plug-in.* Le noyau d'*Eclipse* est une implémentation d'*Equinox* compatible avec la

---

[78] The Eclipse Foundation, *The Eclipse home page*: http://www.eclipse.org/

[79] *Eclipse Modeling Framework Project (EMF)*: http://www.eclipse.org/modeling/emf/

[80] *Eclipse Graphical Modeling Framework (GMF)*: http://www.eclipse.org/modeling/gmf

[81] Protégé Home Site, *Welcome to protégé*: http://protege.stanford.edu/

[82] TopQuadrant, *TopBraid Composer (TM)*: http://www.topquadrant.com/products/TB_Composer.html

[83] NeOn project, *Neon toolkit*: http://neon-toolkit.org/wiki/Main_Page

[84] IBM, *IBM Integrated Ontology Development Toolkit: An ontology toolkit for storage, manipulation, query, and inference of ontologies and corresponding instances.*: http://www.alphaworks.ibm.com/tech/semanticstk

standardisation OSGi. À sa base, l'environnement de développement d'*Eclipse* est divisé en trois couches (Gamma et Beck, 2003), soit la couche *plateforme* qui fournit un accès de base aux fonctionnalités du noyau, la couche *Java Development Tools (JDT)* qui intègre les fonctionnalités d'édition et d'exécution d'applications *Java* et la couche *Plug-in Development Environment (PDE)* qui est une extension du JDT dédiée au développement de *Plug-In.* Comme nous l'avons dit plus haut, la couche plateforme offre des services de base au développeur.

L'architecture de ces services est divisée en une catégorie *noyau* et une catégorie *interface utilisateur (IU).* La catégorie noyau comporte un composant *runtime* qui définit l'infrastructure des *plug-in* et en gère l'instanciation au démarrage de l'application. L'espace de travail (*workspace*) est aussi un composant du noyau. L'espace de travail est un gestionnaire de haut niveau qui permet la gestion de projet. Un projet est un objet *Eclipse* qui contient des fichiers et des dossiers et qui est synchronisé avec le système de gestion de fichiers du système d'exploitation hôte. Quant à la catégorie IU, elle comporte trois composants, soit le plan de travail (*workbench*), le *JFace* et le *Standard Widget Toolkit (SWT).* Le SWT fournit les éléments graphiques de base pour la construction de fenêtres. Le *JFace* fournit des éléments graphiques de plus haut niveau dans lesquels sont implantées des utilisations standardisées. Finalement, le plan de travail fournit les éléments graphiques associés aux paradigmes de l'environnement *Eclipse*. Il offre les primitives nécessaires à la gestion de perspectives, de vues et d'éditeurs. Les ouvrages suivants sont particulièrement instructifs pour le développement de *plug-in:* (Clayberg et Ribel, 2006 ; Daum, 2004 ; Gamma et Beck, 2003 ; McAffer et Lemieux, 2006).

3.1.7.2 *Eclipse Modeling Framework*

Basé sur l'architecture conduite par les modèles (OMG-MDA[85]), le projet *Eclipse Modeling Framework* (EMF) (Steinberg *et al.*, 2008) est un projet *Eclipse* basé sur la génération automatique de code à partir de modèles. À partir d'un modèle *ecore* de format *XML Metadata Interchange* (XMI), l'EMF fournit les outils nécessaires à la transformation du modèle en code source *Java.* Le code *Java* généré fournit les interfaces nécessaires à la manipulation de données dans le respect de la structure exprimée dans le modèle. Les utilitaires assurent aussi la génération automatique de code d'éléments graphiques essentiel à la manipulation des données. L'EMF est l'un des projets d'envergure d'*Eclipse* et il sert de base à de nombreux autres projets. La schématisation détaillée à la figure 3.3 indique qu'à partir d'une conceptualisation et d'un éditeur de modèle UML (Ex: Rational Rose®), il est possible de concevoir un modèle UML qui sert d'intrant au processus de génération d'un modèle *ecore*. C'est à partir du modèle *ecore* que le processus de génération du code *Java* peut produire le code source *Java* nécessaire à la manipulation des données. Le processus de génération des modèles *ecore* et le processus de génération du code source utilisent les outils de génération d'EMF.

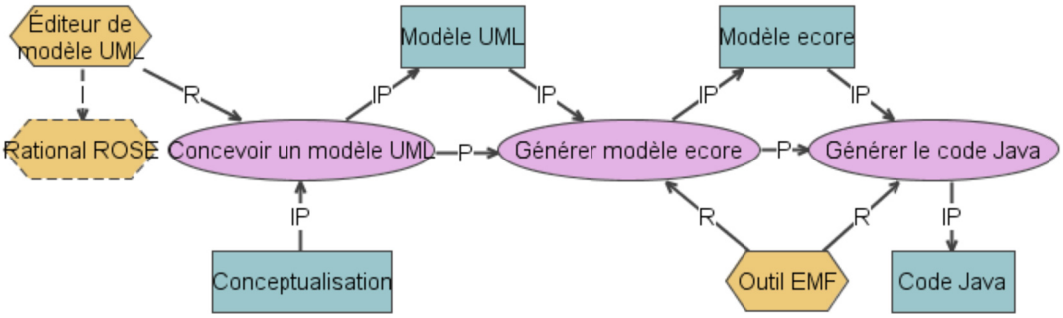


Figure 3.3: Modèle procédural EMF de production de code source *Java* à partir d'une conceptualisation.

[85] OMG MDA, *OMG Model Driven Architecture*: http://www.omg.org/mda/

L'EMF est à la base du mécanisme de pérennisation des données de l'éditeur de modèles semi-formels, que nous appelons eLi. À partir d'une version UML du métamodèle de MOT présenté à la section 3.1.3 p.124, nous avons généré le modèle *ecore* (voir la figure 3.4) qui est à la base du code source *Java* de la couche de pérennisation d'eLi.

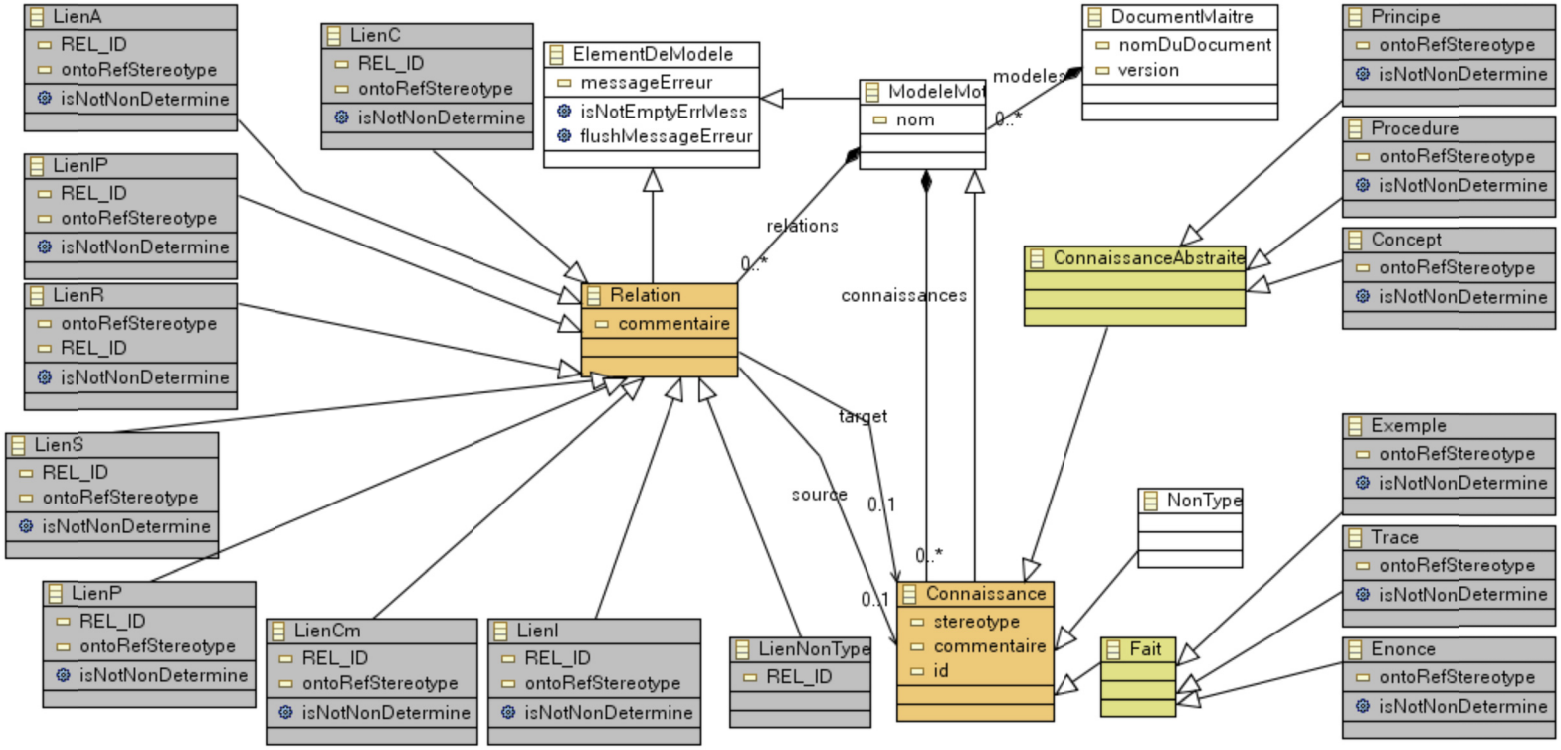


Figure 3.4: Modèle *ecore* de la structure de donnée d'eLi pour la modélisation en langage MOT[86].

[86] Bien qu'eLi permette la représentation d'objets et de liens non-typés ainsi que la composition multiple (lienC**m**), notre travail ne considère pas le traitement de ces primitives.

3.1.7.3 *Graphical Modeling Framework*

Aussi basé sur l'architecture conduite par le modèle, le *Graphical Modeling Framework* (GMF) (Gronback, 2008) fournit les composants nécessaires à la construction d'éditeurs graphiques et schématiques à partir de modèles. Le GMF intègre les fonctionnalités d'EMF et offre une représentation schématisée des données contenues dans les documents XMI produits par EMF. Extrait du Tutoriel GMF[87], la figure 3.5 présente le processus de construction d'un éditeur conçu avec le GMF. Cinq fichiers modèles sont nécessaires à la production du code *Java* pour l'utilisation de GMF. Le modèle du domaine (fichier *ecore*) est produit par le processus EMF. La définition des composants graphiques, ellipse, rectangle, polygone, etc, est modélisée dans le fichier *gmfgraph.* La définition des outils, des actions à réaliser dans l'éditeur à venir, est modélisée dans le fichier *gmftool.* L'étape de développement des correspondances (*mapping*) entre le modèle du domaine, le modèle de définition graphique et le modèle de définition des outils est représentée dans le fichier *gmfmap.* Pour le générateur de code *Java,* le dernier fichier, le *gmfgen,* est le modèle complet de la définition de l'éditeur.

Le GMF est utilisé dans cette thèse pour la conception de l'éditeur de modèles semi-formels MOT (eLi). Cette plateforme de conception d'environnement graphique est une composante importante de la méthodologie puisqu'elle offre les fonctionnalités nécessaires au développement rapide et standardisé d'applications graphiques.

---

[87] *GMF Tutorial*: http://wiki.eclipse.org/index.php/GMF_Tutorial

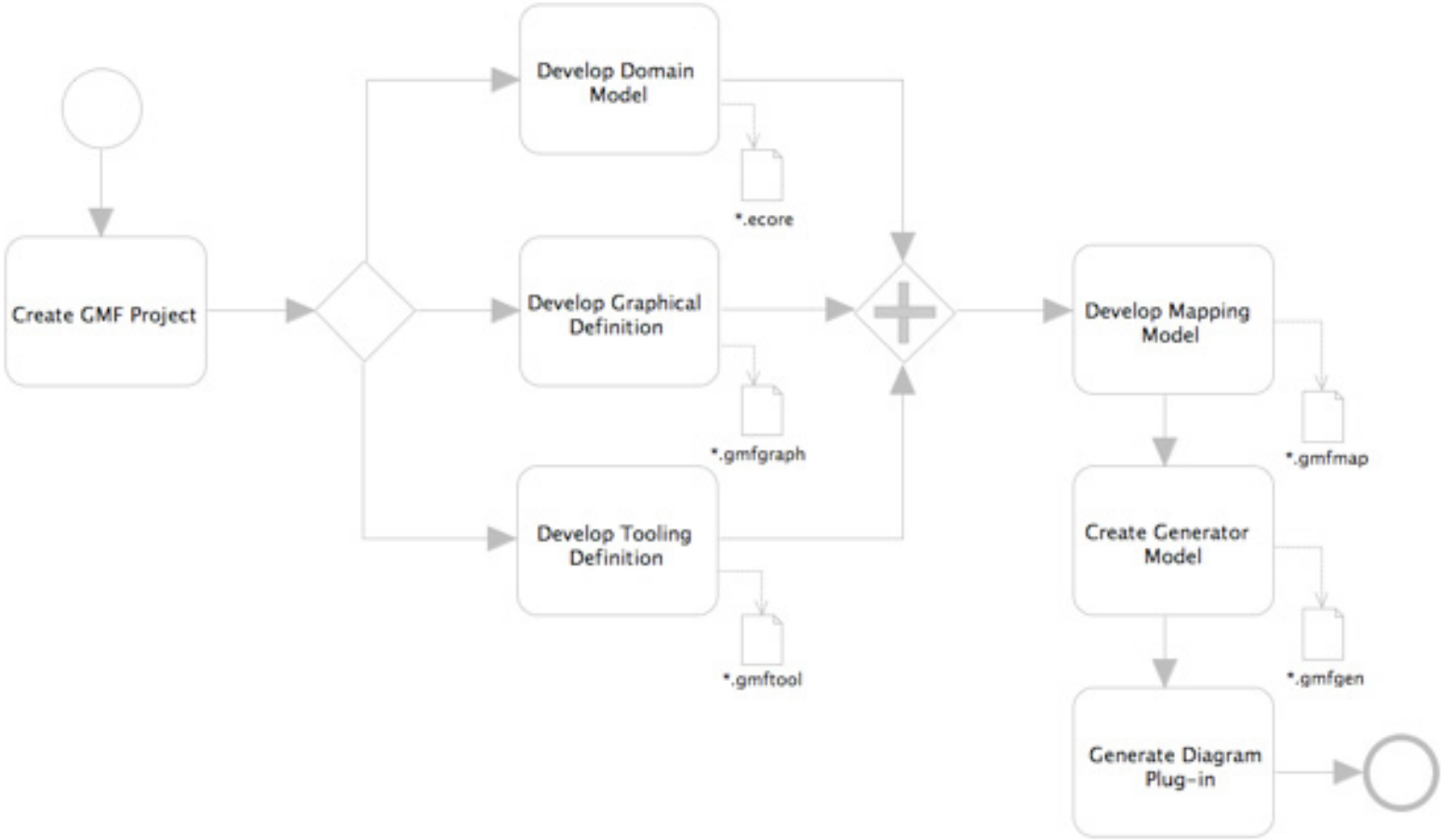


Figure 3.5: Processus de construction d'un éditeur graphique GMF. (Tirée du GMF Tutorial 2009d)

3.1.8 Choix des outils informatiques associés à l'édition des ontologies

La réalisation des ontologies et de la base de règles qui les accompagnent nécessite l'utilisation d'outils d'édition permettant leur manipulation. Spécifiquement, les éditeurs sélectionnés doivent être en mesure d'éditer les ontologies de langage OWL et SWRL. De plus, une représentation *Java* de l'ontologie OWL doit être générable.

3.1.8.1 Éditeurs d'ontologies OWL

Protégé[88] est une application de l'Université de Stanford qui est régie par la convention de codes sources libres. *Protégé* est dédié à l'édition d'ontologies et il offre une plateforme de développement de systèmes à base de connaissances. Plusieurs composants de l'application OntoCASE sont construits à partir de *Protégé*. Le *Protégé-OWL API programmers guide*[89] est un constituant important du *Protégé Programming Development Kit* (PDK)[90] pour le développement d'applications *Java* intégrant la manipulation d'ontologies. Le *Protégé-OWL API* fournit les classes et les méthodes qui permettent la manipulation de fichiers OWL/RDFS. De plus, cette *API (Application Programming Interface)* fournit les interfaces nécessaires à l'appel de requêtes de recherche, à la manipulation de *datamodel* et assure l'appel à des raisonneurs à base de logique de descriptions et à base de règles. Le code source de cette *API* est facilement encapsulable dans un *plug-in Eclipse* afin de construire un environnement autonome et indépendant dans l'éditeur de *Protégé*. De même, le *Protégé-OWL Reasoner API*[91] offre un ensemble d'interfaces permettant d'interagir

---

[88] Protégé Home Site, *Welcome to protégé*: http://protege.stanford.edu/

[89] *protégé-owl api programmer's guide*: http://protege.stanford.edu/plugins/owl/api/guide.html

[90] *Protégé Programming Development Kit (PDK)*: http://protege.stanford.edu/doc/dev.html

[91] *using the protégé-owl reasoner api*: http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html

avec les principaux moteurs d'inférences (Racer[92], Fact[93], Fact++[94], Pellet[95]) compatibles avec la logique de descriptions et le langage OWL.

TopBraid Composer ™[96] *(TBC)* est un éditeur d'ontologie OWL disponible en trois versions : la version gratuite, la version standard et la version Maestro. TBC est un outil de modélisation graphique d'ontologies OWL. Il comprend principalement un éditeur de classes, de propriétés, d'individus et de règles SWRL. TBC permet aussi l'édition d'ontologies dans le formalisme N3, RDF/XML, N-TRIPLE, TURLE ainsi qu'en représentation graphique UML. Il facilite le développement de multiples ontologies par la synchronisation inter-ontologies des données ce qui est une fonctionnalité importante en développement d'ontologies. De plus, TBC intègre les principaux moteurs d'inférences disponibles. La version TBC standard est utilisée pendant les différentes phases de la démarche de cette recherche puisqu'elle s'intègre parfaitement à un environnement *Eclipse* déjà établi. Quant à la version gratuite, nous avons intégré ses *plug-ins* à l'application OntoCASE fournissant ainsi un éditeur d'ontologies à notre méthodologie.

3.1.8.2 *Semantic Web Rule Language*

Le *Semantic Web Rule Language* (SWRL)[97] qui, au moment d'écrire ces lignes, en est à l'étape de proposition auprès du W3C, est le langage proposé ici pour supporter le paradigme à base de règles pour les ontologies OWL. L'éditeur SWRL de *Protégé*

---

[92] *Renamed Abox and Concept Expression Reasoner (RACER)*: http://www.sts.tu-harburg.de/~r.f.moeller/racer/

[93] Horrocks, *The FaCT System*: The FaCT System

[94] Horrocks, *FaCT++*: http://owl.man.ac.uk/factplusplus/

[95] *Pellet: The Open Source OWL Reasoner*: http://clarkparsia.com/pellet

[96] TopQuadrant, *TopBraid Composer (TM)*: http://www.topquadrant.com/products/TB_Composer.html

[97] Horrocks, Boley, Tabet, Grosof et Dean, *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*: http://www.w3.org/Submission/SWRL/

(SWRLTab[98]) est un environnement complet de développement de systèmes à base de règles SWRL. En plus d'offrir un éditeur de règles, *Protégé* offre un *API* complet de développement d'applications *Java* (Golbreich et Imai, 2004 ; Mei et Paslaru, 2005 ; O'Connor *et al.*, 2008a ; O'Connor *et al.*, 2008b ; O'Connor *et al.*, 2005a ; O'Connor *et al.*, 2005b). Du point de vue du moteur d'inférences SWRL, *Protégé* utilise le langage à base de règles *Jess*[99](Golbreich et Imai, 2004 ; O'Connor *et al.*, 2005b). Deux interfaces du SWRLTab retiendront particulièrement notre attention: le *SWRL Factory*[100] et le *SWRL Built-in Bridge*[101]. Le *SWRL Factory* fournit une *API Java* de haut niveau qui permet la création et la modification de règles SWRL à l'intérieur d'une ontologie. Le *SWRL Built-in Bridge* est un mécanisme permettant l'implantation *Java* de prédicats *SWRL Built-in*[102] utilisateurs. Cet environnement jouera un rôle important dans la conception du module de transformation du modèle semi-formel en ontologie du domaine (voir la section 3.2.4.2). TBC possède aussi un éditeur de règles SWRL. Beaucoup moins sophistiqué que celui de *Protégé*, l'éditeur de TBC ne permet pas l'intégration de *Built-in commands.* En revanche, TBC utilise le moteur d'inférence *Pellet*.

3.2 Phase 2: Agrégation des composants ontologiques, procéduraux et informatiques d'OntoCASE

Au sein de la démarche de recherche, la phase 2 est l'étape de construction des objets de la recherche. La conception de l'assistant informatique servira de fil conducteur à la construction de l'ontologie de transformation ainsi que de ses composants et à la consolidation des éléments procéduraux de la méthodologie. Schématisée à la

---

[98] O'Connor, *SWRLTab*: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab

[99] *Jess: the Rule Engine for the Java Platform*: http://www.jessrules.com/jess/index.shtml

[100] O'Connor, *SWRL Factory FAQ*: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLFactoryFAQ

[101] O'Connor, *SWRLBuilt In Bridge*: http://protege.cim3.net/cgi-bin/wiki.pl?SWRLBuiltInBridge

[102] Horrocks, Patel-Schneider, Boley, Tabet, Grosof et Dean, *SWRL Section 8. Built-Ins*: http://www.daml.org/rules/proposal/builtins.html

figure 3.6, l'architecture de la plateforme de support à la formalisation d'OntoCASE se décompose en quatre thèmes: l'*édition*, l'*importation*, le *traitement* et la *communication*. L'édition concerne les modules dédiés à l'édition d'un modèle semi-formel ou d'une ontologie. Les modules associés à la conversion ont pour fonction d'assurer la traduction des modèles d'une notation à une autre. Quant aux modules associés au traitement, ils ont pour fonction de réaliser la transformation des éléments de modèles. Finalement, les modules associés à la communication servent à assurer la communication de l'information entre les différents modules ou entre les modules et l'utilisateur.

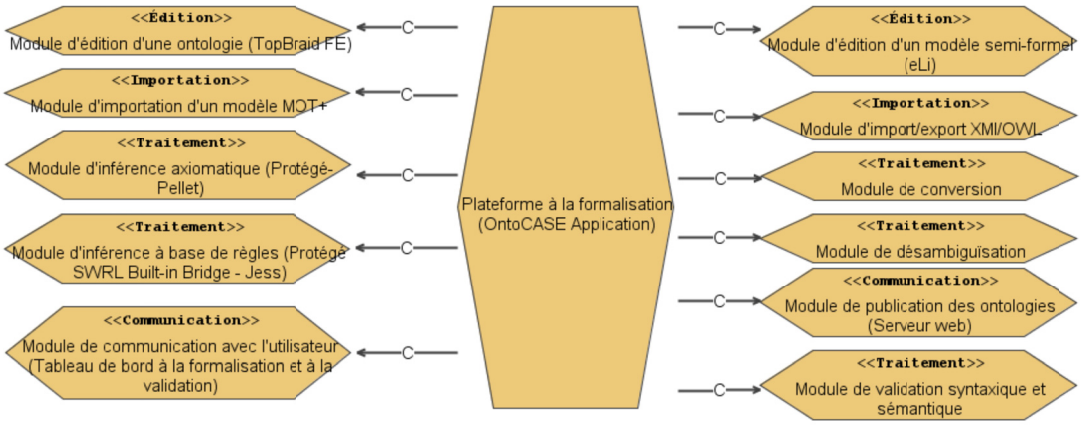


Figure 3.6: Les composants de l'application OntoCASE.

Dans ce modèle, nous avons représenté les modules sous forme de principes puisque, conformément au langage MOT, ces modules sont considérés comme des acteurs informatiques qui régissent les diverses procédures de la méthodologie.

La figure 3.7 présente les modules de l'application OntoCASE mis en relation avec les méthodes de la méthodologie. La méthode de modélisation semi-formelle utilise le module d'édition eLi. Le processus d'importation qui fait partie de la méthode de formalisation utilise le module d'importation *XMI/OWL* ou le module d'importation de fichiers MOT+. Le processus de désambiguïsation nécessite des modules qui sont associés au traitement et à l'édition. Pour l'édition, eLi et TBC permettent la

présentation des informations de manière semi-formelle et ontologique en vue de solliciter d'éventuelles interventions de l'utilisateur. Le module de désambiguïsation, responsable du traitement des ambiguïtés, se compose des modules d'inférence axiomatique et à base de règles SWRL, qui, eux-mêmes, utilisent le module de communication avec le serveur Web pour la publication des ontologies.

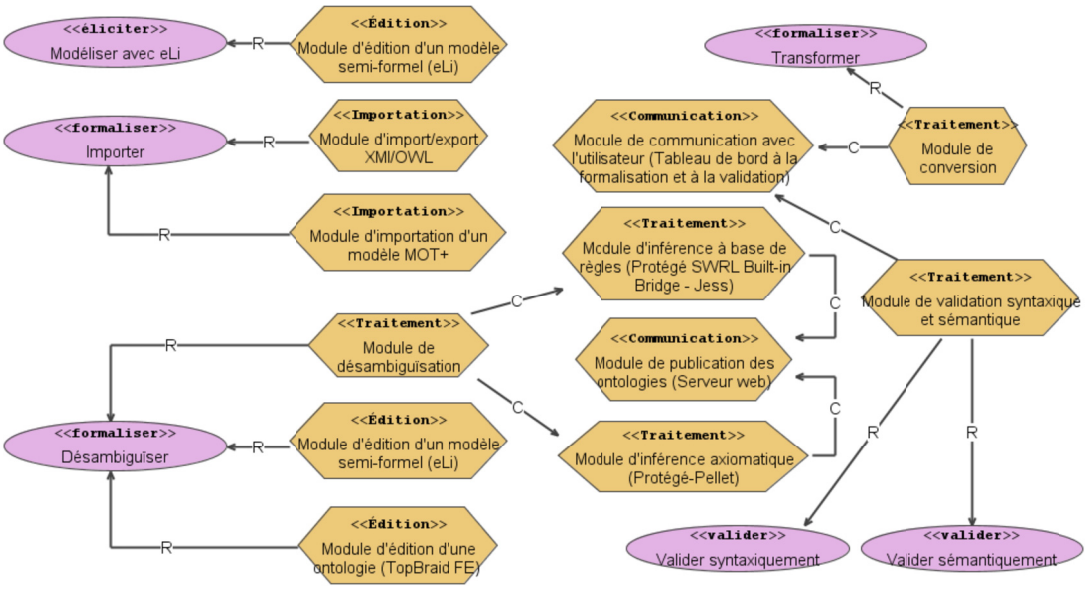


Figure 3.7: Relations entre les modules de l'application et les méthodes d'OntoCASE.

La figure 3.8 présente l'interface utilisateur de l'application OntoCASE. Au centre de la moitié supérieure, on trouve le schéma du modèle semi-formel, alors qu'à gauche, la vue en graphe de composites en présente la taxonomie. La moitié inférieure droite présente un tableau de bord (sous forme d'un graphe de processus) permettant à l'ingénieur de lancer les processus nécessaires à la formalisation (importer, désambiguïser et convertir) et d'éditer au besoin chaque modèle après le déclenchement du processus. Une vue permettant le contrôle de la validation ainsi qu'une console de commandes exécutées, un volet des propriétés des objets et un calepin des messages d'erreurs sont aussi accessibles. Dans la section inférieure gauche, l'utilisateur a accès à des utilitaires tels qu'un calepin de tâches à faire, une

structure *SVN*[103] pour le contrôle de version des ontologies, une vue de manipulation du serveur Web ainsi qu'une vue de structure facilitant la navigation dans de grands modèles.

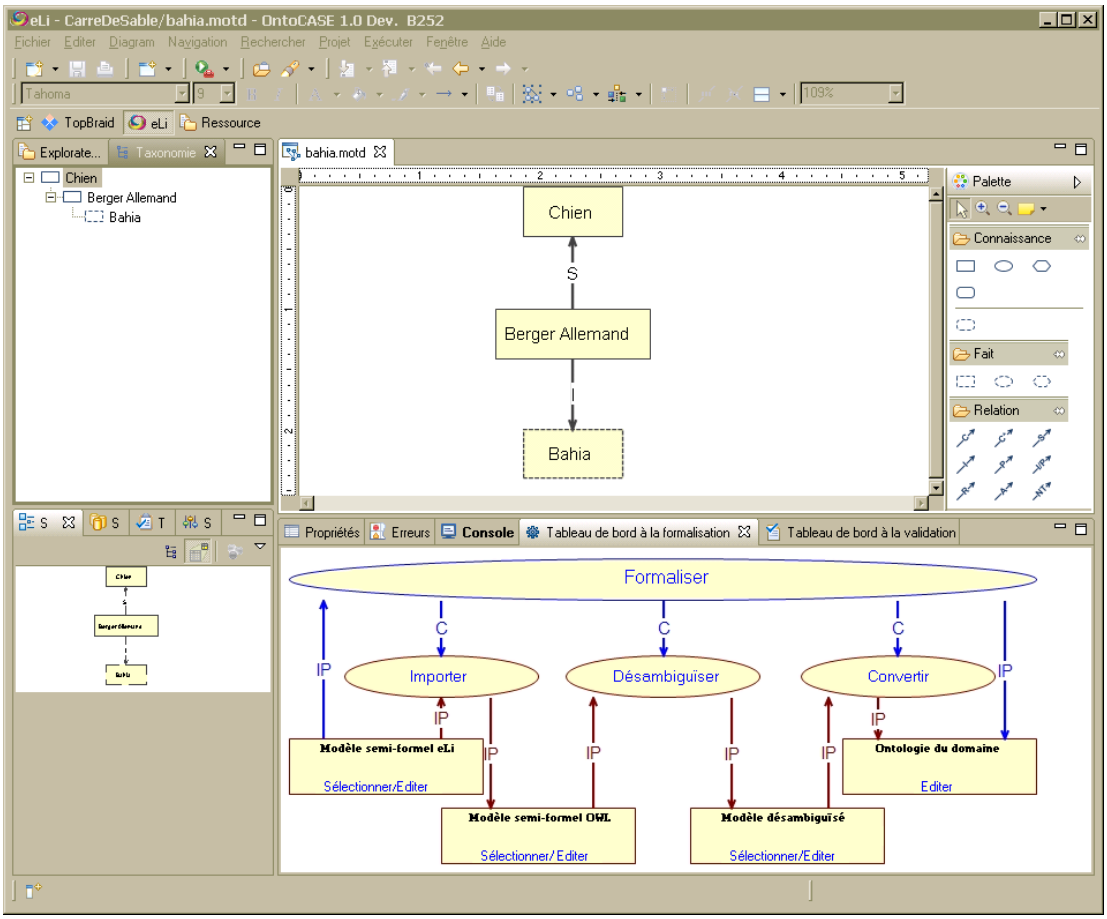


Figure 3.8: Interface utilisateur de l'application OntoCASE.

3.2.1 Développement du module d'édition

L'édition est assurée par deux modules, le module d'édition de modèles semi-formels (eLi) et le module d'édition d'ontologies OWL (*TopBraid Composer* ™) qui constituent l'instrumentation de la méthode de modélisation. *TopBraid Composer* ™

[103] SubVersion, version control system, http://subversion.tigris.org/

free edition est une application *Eclipse* d'où il est possible d'extraire les *Plug-in* spécifiques à l'édition d'ontologies pour les intégrer à l'application OntoCASE. Quant à eLi, il s'agit d'une application de type EMF-GMF (voir la section 3.1.7.2 et la section 3.1.7.3). Le tableau 3.21 présente trois représentations supportées par l'application OntoCASE.

Tableau 3.21
Représentation MOT, XMI et OWL-N3 de *Bahia est un Berger Allemand qui est une sorte de Chien.*

a) modèle en MOT



b) modèle en XMI

```
01.  <?xml version="1.0" encoding="UTF-8"?>
02.  <mot:ModeleMot xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:mot="www.cotechnoe.com/mot">
03.  <connaissances xsi:type="mot:Exemple" nom="Bahia" id="Bahia_0"/>
04.  <connaissances xsi:type="mot:Concept" nom="Berger Allemand" id="Berger-
     Allemand_1">
05.  <relations xsi:type="mot:LienS" source="//@connaissances.1"
     target="//@connaissances.2"/>
06.  <relations xsi:type="mot:LienI" source="//@connaissances.1"
     target="//@connaissances.0"/>
07.  </connaissances>
08.  <connaissances xsi:type="mot:Concept" nom="Chien" id="Chien_2"/>
09.  </mot:ModeleMot>
```

c) modèle en OWL-N3

```
01.  @prefix :        <http://localhost/Ontologies/bahia.owl#> .
02.  @prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
03.  @prefix metaDom:
     <http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl#> .
04.  @prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
05.  @prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
06.  @prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
07.  @prefix owl:     <http://www.w3.org/2002/07/owl#> .
08.  @prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
09.  @prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
10.  @prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
11.  <http://localhost/Ontologies/bahia.owl>  a  owl:Ontology ;
12.  owl:imports <http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl> .
13.  :Bahia_0  a  :Berger-Allemand_1 ;
14.      rdfs:label "Bahia"^^xsd:string .
15.  :Berger-Allemand_1  a  owl:Class ;
16.      rdfs:label "Berger Allemand"^^xsd:string ;
17.      rdfs:subClassOf :Chien_2 .
18.  :Chien_2  a  owl:Class ;
19.      rdfs:label "Chien"^^xsd:string ;
20.      rdfs:subClassOf owl:Thing , metaDom:MD Declarative Concept .
```

De l'exemple de la conceptualisation *Le Chien Bahia est un Berger Allemand,* le schéma en **a** en représente le modèle en langage MOT, alors que **b** et **c** représentent

respectivement le modèle dans le formalisme XMI et OWL-N3. Au tableau 3.21**b,** on remarque que chacun des éléments du modèle présenté en **a** est traduit par un énoncé dans le fichier *XMI* (voir les lignes 03 à 08). Ainsi, cinq énoncés XMI sont nécessaires pour représenter les deux concepts (Chien et Berger Allemand), l'exemple (Bahia) ainsi que le `mot:LienS` et le `mot:LienI`. L'ontologie de **c** contient les énoncés formalisés. On y retrouve les deux classes *Chien* et *Berger Allemand* (voir les lignes 15 et 18). L'exemple *Bahia* et le `mot:LienI` sont transformés en individu (voir la ligne 23) dont le type est *Berger Allemand* (voir la ligne 24). Quant au `mot:LienS`, il est transformé en énoncé `rdfs:subClassOf` (voir la ligne 29).

3.2.2 Développement du module d'importation

Les processus de formalisation et de validation sont réalisés dans l'espace de modélisation OWL alors que l'édition est réalisée dans un autre espace de modélisation (l'espace de modélisation *XMI*). Le premier processus de la formalisation est le processus d'importation (voir la figure 2.9, p.101). Le programme `Eli2OWL.java`[104] est celui qui importe le modèle de format *XMI* pour produire un modèle dans le format OWL, dont le processus de conception est schématisé à la figure 3.9. La première étape est la génération du code *Java* de l'ontologie du langage MOT en utilisant le générateur de code de *Protégé* (2009h). L'intrant de cette étape est l'ontologie du langage semi-formel MOT qui est détaillée au tableau 3.22. L'étape suivante est la génération du code *Java* correspondant au modèle EMF de MOT en conformité avec le processus EMF détaillé à la section 3.1.7.2 p. 148.

La programmation du module d'importation qui produit le programme `Eli2OWL.java`, utilise en intrant le code *java* de l'ontologie de MOT et le code *java* du modèle EMF de MOT. Le programme réalise la traduction un à un des éléments et des relations de

---

[104] Le détail du code d'Eli2OWL.java est présenté à la section 1 de l'appendice F

formalisme *XMI* vers le formalisme OWL (voir la représentation *XMI* tableau 3.21**b** et sa correspondance OWL au tableau 3.22**c**).

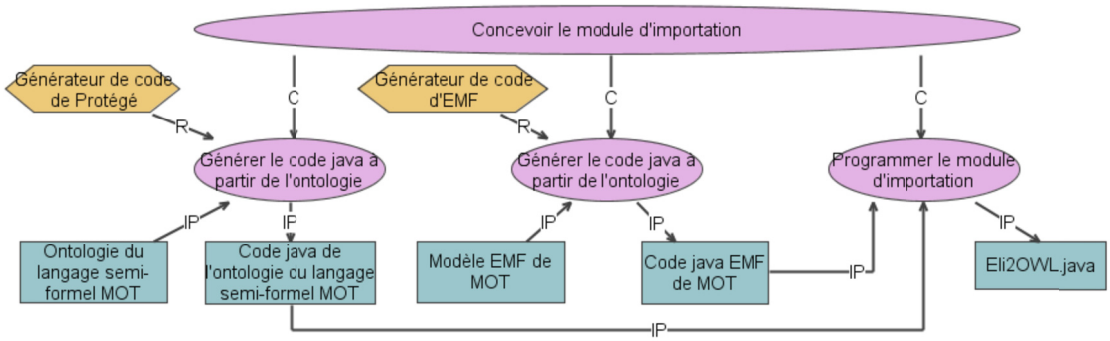
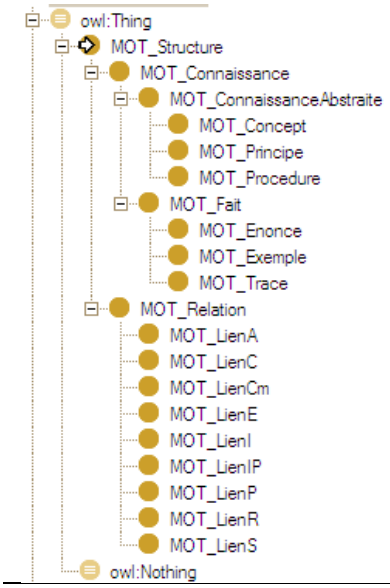


Figure 3.9: Processus de conception du programme *Eli2OWL.java.*

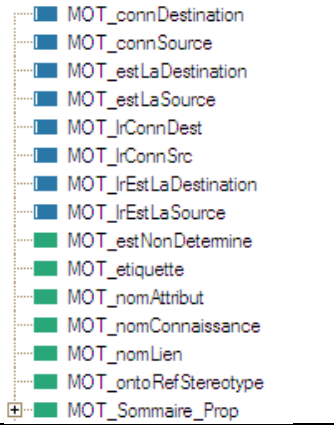
Le tableau 3.22**a** et **b** présente la structure détaillée des classes et propriétés de l'ontologie du vocabulaire de MOT. La taxonomie des classes présentée en **a** reflète le métamodèle de MOT tel que défini à la section 2.4 p.86. Les propriétés `MOT_connSource` et `MOT_connDestination` servent à faire référence aux connaissances source et destination d'un lien (voir en **c**, les lignes 22 et 23). Leurs propriétés inverses, `MOT_estLaSource` et `MOT_estLaDestination`, sont utilisées par les entités pour identifier les relations qui les pointent (voir en **c,** les lignes 10 et 16). Les propriétés dont le nom commence par `MOT_lr`(…) sont spécifiquement dédiées à la manipulation des `mot:LienR`, notamment dans le cas où le `mot:Principe` est formalisé en propriété. L'attribut booléen `metaMot:MOT_estNonDetermine` **(**voir **c** ligne 5) est un indicateur qui informe l'assistant à la désambiguïsation à savoir s'il doit désambiguïser l'élément du modèle. Si l'utilisateur désambiguïse préalablement l'élément du modèle (`metaMot:MOT_estNonDetermine = false`), alors l'assistant à la désambiguïsation sera désactivé pour cet élément.

Tableau 3.22
Ontologie cadre (OWL) du vocabulaire de MOT [a) et b)], et exemple d'importation en OWL-N3 d'un modèle *MOT-XMI* [c)]

a)



b)



c)

```
01.   :Bahia_0
02.   rdf:type metaMot:MOT_Exemple ;
03.   metaMot:MOT_estLaDestination
04.   :LienI_Berger-Allemand_1_Bahia_0 ;
05.   metaMot:MOT_estNonDetermine "true"^^xsd:boolean ;
06.   metaMot:MOT_etiquette "Bahia"^^xsd:string ;
07.   metaMot:MOT_nomConnaissance "Bahia_0"^^xsd:string
      .
08.   :Berger-Allemand_1
09.   rdf:type metaMot:MOT_Concept ;
10.   metaMot:MOT_estLaSource :LienS_Berger-
      Allemand_1_Chien_2 , :LienI_Berger-
      Allemand_1_Bahia_0 ;
11.   metaMot:MOT_estNonDetermine "true"^^xsd:boolean ;
12.   metaMot:MOT_etiquette "Berger
      Allemand"^^xsd:string;
13.   metaMot:MOT_nomConnaissance "Berger-
      Allemand_1"^^xsd:string .
14.   :Chien_2
15.   rdf:type metaMot:MOT_Concept ;
16.   metaMot:MOT_estLaDestination :LienS_Berger-
      Allemand_1_Chien_2 ;
17.   metaMot:MOT_estNonDetermine "true"^^xsd:boolean ;
18.   metaMot:MOT_etiquette "Chien"^^xsd:string ;
19.   metaMot:MOT_nomConnaissance "Chien_2"^^xsd:string
      .
20.   :LienI_Berger-Allemand_1_Bahia_0
21.   rdf:type metaMot:MOT_LienI ;
22.   metaMot:MOT_connDestination :Bahia_0 ;
23.   metaMot:MOT_connSource :Berger-Allemand_1 ;
24.   metaMot:MOT_estNonDetermine "true"^^xsd:boolean ;
25.   metaMot:MOT_nomLien "LienI_Berger-
      Allemand_1_Bahia_0"^^xsd:string .
26.   :LienS_Berger-Allemand_1_Chien_2
27.   rdf:type metaMot:MOT_LienS ;
28.   metaMot:MOT_connDestination :Chien_2 ;
29.   metaMot:MOT_connSource :Berger-Allemand_1 ;
30.   metaMot:MOT_estNonDetermine "true"^^xsd:boolean ;
31.   metaMot:MOT_nomLien "LienS_Berger-
      Allemand_1_Chien_2"^^xsd:string .
```

3.2.3 Développement du module de traitement

Le traitement concerne les modules responsables de la manipulation de l'information, soit la désambiguïsation du modèle semi-formel en modèle semi-formel désambiguïsé, la conversion du modèle semi-formel désambiguïsé en ontologie du domaine et la validation syntaxique et sémantique de l'ontologie du domaine. Les

trois sections suivantes présentent le modèle procédural de développement de chacun de ces modules.

3.2.4 Développement du module de désambiguïsation

Le processus de *conception du module de désambiguïsation* (voir la figure 3.10) consiste en : la création du programme `ReglesDesamb.java` (voir appendice F.2.), l'élaboration du catalogue de la sémantique formelle de MOT (voir l'appendice B), le raffinement de l'ontologie de référence et de l'ontologie du traitement des ambiguïtés ainsi que la réalisation des bases de règles à la désambiguïsation typologique (voir l'appendice F.3) et topologique (voir l'appendice F.4). Le développement de chacun de ces cinq artéfacts est interrelié au développement des quatre autres; c'est pourquoi la conception de ce module implique un développement de type itératif et évolutif[105] dont les composants de l'ontologie de transformation sont le centre.

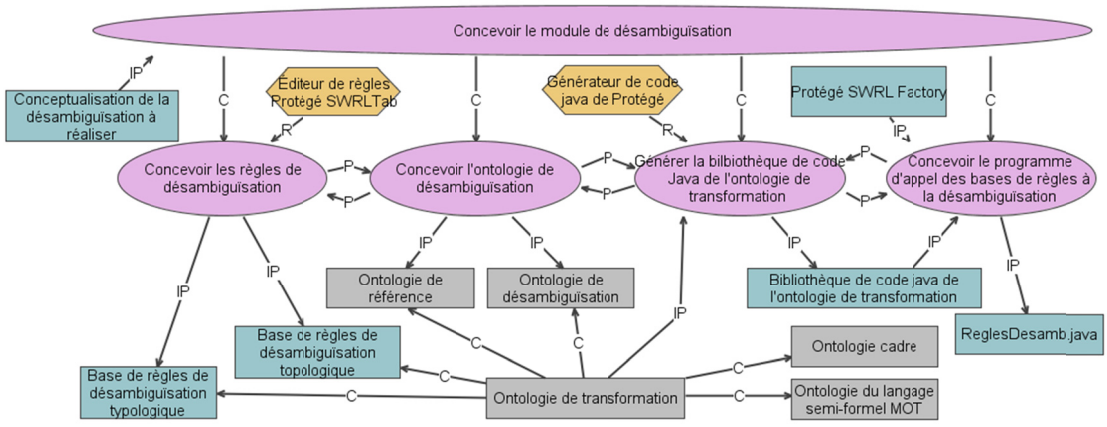


Figure 3.10: Modèle du processus de conception du module de désambiguïsation.

3.2.4.1 Ontologie de traitement des ambiguïtés et règles de désambiguïsation

Le développement des bases de règles de désambiguïsation et le développement de l'ontologie de traitement des ambiguïtés sont deux processus intrinsèquement reliés.

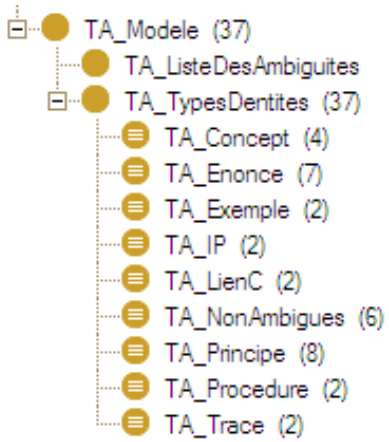
---

[105] Représentée par les liens P entre chacun de sous-processus.

Présentée de manière partielle au tableau 3.23 et de manière complète à l'appendice F.11, l'ontologie du traitement des ambiguïtés (TA) est une composante de l'ontologie de transformation. Le rôle de cette ontologie est d'offrir une structure de classification des éléments du modèle semi-formel pour l'étape de désambiguïsation assurée par les bases de règles de désambiguïsation. La structure de classes de l'ontologie du traitement des ambiguïtés (voir la classe `TA_Modele` du tableau 3.23**a**) englobe deux classes. La classe contenant la liste des ambiguïtés (`TA_ListeDesAmbiguites`) est le contenant des éléments du modèle semi-formel qui n'auront pas été désambiguïsés et qui seront identifiés par l'application de la restriction représentée en **b**. Une action de l'expert de contenu sera nécessaire pour désambiguïser les éléments du modèle semi-formel contenus dans cette classe. Quant à la classe des types d'entités (`TA_TypesDentites`), elle regroupe un ensemble de classes représentant les différentes entités semi-formelles du langage MOT. Les individus associés à ces classes (voir le tableau 3.23**c**) représentent les interprétations possibles de chacune des entités semi-formelles telles que définies au tableau des *catégories des entités du modèle de référence* (voir le tableau 2.3 et le tableau 2.6 p.89). Chacune des entités du modèle semi-formel est associée, par une règle de désambiguïsation, à l'un de ces éléments par la propriété `ot:OT_EntiteEstDeType` de l'ontologie de transformation (voir l'exemple de la ligne 17 du tableau 3.24). Dans ce cas précis, le *Principe ?p* est désambiguïsé en *Propriété*, le *Concept* à la source *?connSrc* et le *Concept* à la destination *?connDest* sont désambiguïsés en *Concept_Classe*, et finalement, les *LienR* qui relient les *Concepts* au *Principe* sont désambiguïsés en *Relation_Regulation.*

Tableau 3.23
Structure de l'ontologie de traitement des ambiguïtés

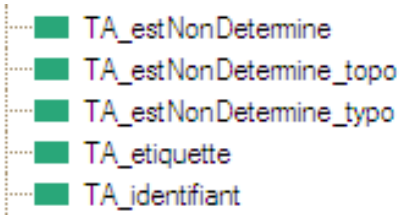
a)



b)

```
oAmbig:TA_ListeDesAmbiguites
  owl:equivalentClass
   [ a  owl:Restriction ;
     owl:hasValue oAmbig:ENTITE_AMBIGUE ;
     owl:onProperty :OT_EntiteEstDeType
   ].
```

d)



c)

[Resource]	rdf:type
Agent_Contrainte_Norme	TA_Principe
Attribut	TA_LienC
Attribut_Valeur	TA_Exemple
Concept_Classe	TA_Concept
Condition	TA_Principe
ENTITE_AMBIGUE	TA_TypesDentites
Entite_Schema_Float	TA_Concept
Entite_Schema_Int	TA_Concept
Entite_Schema_String	TA_Concept
Holonyme	TA_LienC
Intrant	TA_IP
NON_DETERMINE	TA_TypesDentites
Observable_AgentNormeContrainte	TA_Enonce
Observable_Condition	TA_Enonce
Observable_Objet	TA_Exemple
Observable_Operation	TA_Trace
Observable_Procedure	TA_Trace
Observable_Propriete	TA_Enonce
Observable_Regle_Antecedent	TA_Enonce
Observable_Regle_Complete	TA_Enonce
Observable_Regle_Conclusion	TA_Enonce
Observable_Regle_Nom	TA_Enonce
Operation	TA_Procedure
Procedure	TA_Procedure
Produit	TA_IP
Propriete	TA_Principe
Propriete_Unaire	TA_Principe
Regle_Antecedent	TA_Principe
Regle_Complete	TA_Principe
Regle_Conclusion	TA_Principe
Regle_Nom	TA_Principe
Relation_Application	TA_NonAmbigues
Relation_Englobe	TA_NonAmbigues
Relation_Instanciation	TA_NonAmbigues
Relation_Precedence	TA_NonAmbigues
Relation_Regulation	TA_NonAmbigues
Relation_Specialisation	TA_NonAmbigues

Les propriétés, de l'ontologie de traitement des ambiguïtés, présentées en **d** (`TA_estNonDetermine(_type,_topo)`) sont respectivement des `owl:DatatypeProperty` de type booléen qui déterminent le type de désambiguïsation réalisé sur l'entité semi-formelle. `TA_etiquette` et `TA_identifiant` permettant d'assigner des valeurs d'appellation sur l'entité semi-formelle.

En ce qui concerne les règles de désambiguïsation typologique et topologique, les appendices F.3 et F.4 présentent l'ensemble des règles utilisées par OntoCASE pour réaliser la désambiguïsation. À titre d'exemple, le tableau 3.24 présente la règle de désambiguïsation topologique de même que son interprétation associée à la désambiguïsation d'un principe entre deux concepts (voir le scénario présenté au tableau 3.18**c,** p. 142).

Tableau 3.24
Règle de désambiguïsation topologique dans le cas d'un principe lié par lien R d'un concept (domaine) à un autre (codomaine)

a) Règle de désambigüisation

```
01.   #Rule-08-a_desamb_Principe_entre_concept
02.   metaMot:MOT_Principe(?p)  ^
03.   metaMot:MOT_lrEstLaSource(?p, ?lrSrc)  ^
04.   metaMot:MOT_lrEstLaDestination(?p, ?lrDest)  ^
05.   metaMot:MOT_lrConnSrc(?lrDest, ?connSrc)  ^
06.   metaMot:MOT_lrConnDest(?lrSrc, ?connDest)  ^
07.   metaMot:MOT_LienR(?lrSrc)  ^
08.   metaMot:MOT_LienR(?lrDest)  ^
09.   metaMot:MOT_Concept(?connSrc)  ^
10.   metaMot:MOT_Concept(?connDest)  ^
11.   metaMot:MOT_nomConnaissance(?connSrc, ?nomSrc)  ^
12.   metaMot:MOT_nomConnaissance(?p, ?nomProp)  ^
13.   metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
14.   oAmbig:TA_estNonDetermine(?p, true)  ^
15.   oAmbig:TA_estNonDetermine(?connSrc, true)  ^
16.   oAmbig:TA_estNonDetermine(?connDest, true)  ^
          ➔  ot:OT_EntiteEstDeType(?p, oAmbig:Propriete)  ^
17.   ot:OT_EntiteEstDeType(?connSrc, oAmbig:Concept_Classe)  ^
18.   ot:OT_EntiteEstDeType(?connDest, oAmbig:Concept_Classe)  ^
19.   ot:OT_EntiteEstDeType(?lrSrc, oAmbig:Relation_Regulation)  ^
20.   ot:OT_EntiteEstDeType(?lrDest, oAmbig:Relation_Regulation)  ^
21.   oAmbig:TA_estNonDetermine_topo(?p, false)  ^
22.   oAmbig:TA_estNonDetermine_topo(?connSrc, false)  ^
23.   oAmbig:TA_estNonDetermine_topo(?connDest, false)  ^
24.   oAmbig:TA_estNonDetermine_topo(?lrSrc, false)  ^
25.   oAmbig:TA_estNonDetermine_topo(?lrDest, false)  ^
26.   swrlbi:invoker("printf", "OBJET(%s) -R-> PROPRIETE(%s) -R-> OBJET(%s) )",
      ?nomSrc, ?nomProp, ?nomDest)
```

b) interprétation

Le nom de la règle est représenté à la ligne 1. Les lignes 2 à 13 représentent le scénario topologique déclenchant l'exécution de la règle. Elles indiquent que le scénario consiste en un principe ?p qui est la source et la destination de `lienR` *et que les connaissances aux extrémités de ces liens sont des concepts. Les lignes 14, 15 et 16 assurent que les entités concernées par l'exécution de cette règle n'ont pas été préalablement déterminées par l'utilisateur. Les lignes 17 à 20 déterminent le type de chacune des entités impliquées dans le scénario, soit une propriété entre deux classes. Les lignes 21 à 25 spécifient que les entités du scénario ont été désambiguïsées de manière topologique. Finalement, la ligne 26 envoie un message au gestionnaire d'impression à la console.*

3.2.4.2 Gestion de la désambiguïsation

L'objectif de la conception du module de désambiguïsation est de produire une application informatique intelligente dont les inférences, orientées vers la désambiguïsation, sont guidées par l'ontologie de désambiguïsation. Le contrôle des éléments d'entrées/sorties de l'application intelligente, ainsi que l'ordre de déclenchement des inférences, sont régis par le module `ReglesDesamb.java`, dont le code complet est présenté à l'appendice F.2. `ReglesDesamb.java` importe la bibliothèque de codes sources de l'ontologie de transformation produite à l'aide du générateur de codes de *Protégé*. Cette bibliothèque offre les fonctionnalités de manipulation d'ontologies de faits, qui est structurée en conformité avec l'ontologie de transformation.

Les bases de règles de désambiguïsation typologique et topologique sont structurées selon l'ontologie de transformation. L'exécution de `ReglesDesamb.java` se divise en quatre phases. La première phase est l'initialisation de la structure de l'ontologie résultant du processus de désambiguïsation. La deuxième phase est le déclenchement des règles de désambiguïsation typologique et topologique sur l'ontologie résultant de l'importation du modèle semi-formel. Lorsque la désambiguïsation est terminée, la phase 3 déclenche une inférence fondée sur les bases de règles afin d'identifier les éléments du modèle semi-formel qui n'ont pas été désambiguïsés et sauvegarde les données dans l'ontologie du modèle semi-formel désambiguïsé. Finalement, la phase 4 met à jour le diagramme du modèle semi-formel qui est présenté à l'utilisateur afin de marquer sur le diagramme les éléments du modèle qui sont encore ambigus.

3.2.5 Développement du module de conversion

Schématisé à la figure 3.11, le développement du module de conversion implique l'implantation des patrons de conception *Commande* et *Invocateur* (Buschmann *et al.*, 2007 ; Gamma *et al.*, 1999), le développement du gestionnaire d'ontologies `ExecuteRegles.java`, du développement conjoint des règles de conversion, des commandes *Built-ins* et de l'ajustement de l'ontologie de référence et de l'ontologie cadre.

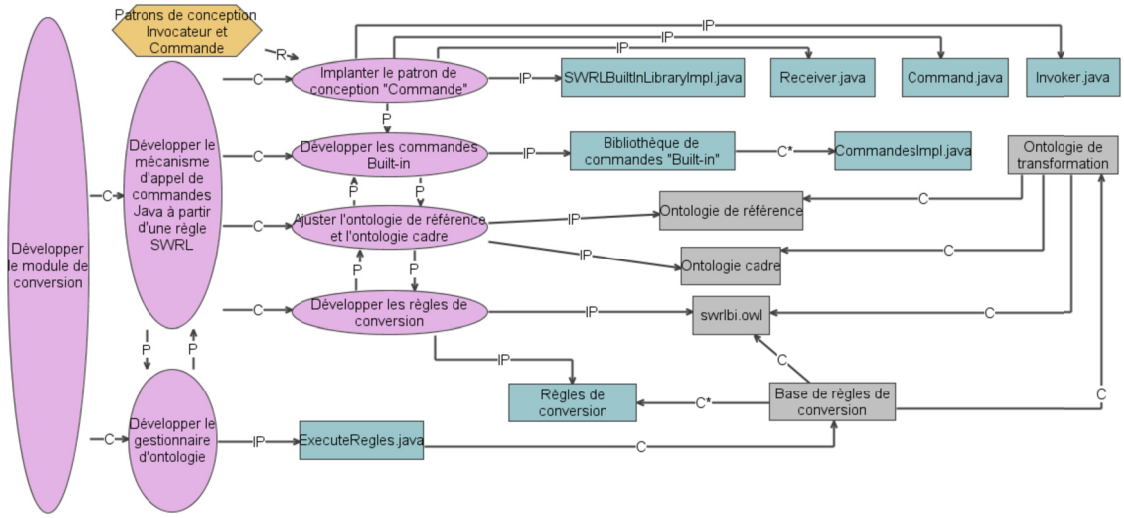


Figure 3.11: Développer le module de conversion de l'ontologie du modèle semi-formel désambiguïsé en ontologie du domaine.

3.2.5.1 Patrons de conception *Invocateur* et *Commande*

Les patrons de conception *Invocateur* et *Commande* ont pour intention d'offrir à un client une méthode standardisée et unique permettant l'appel de commandes. On considère la commande comme une action offerte par une boîte à outils en vue de réaliser une tâche. Jumelé à l'*invocateur*, le patron de conception *Commande* permet le développement itératif et évolutif de commandes sans que la signature des appels ne soit affectée. La dynamique d'utilisation de ce patron de conception, schématisée à

la figure 3.12, démarre avec l'appel de la méthode[106] *invoquer()* de l'invocateur par le client. Le nom de la commande à exécuter ainsi que les arguments de la commande sont passés en paramètre à la méthode. L'invocateur a la responsabilité de créer la commande, puis de lancer l'exécution de celle-ci. Le receveur, qui est un objet dynamique de la boîte à outils, reçoit la requête d'action de la commande, ce qui permet au receveur de produire le résultat. L'invocateur retourne au client le résultat de l'exécution de la commande.

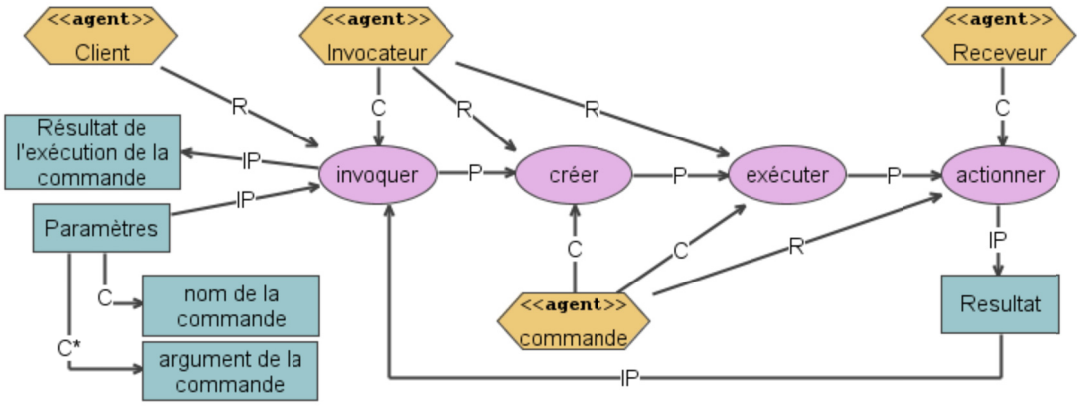


Figure 3.12: Relations[107] et utilisations d'un invocateur, d'une commande et d'un receveur.

3.2.5.2 Développement du mécanisme d'appel de commande *Java* à partir de règles SWRL.

Un aspect innovateur important d'OntoCASE est sa boîte à outils permettant la construction d'une ontologie cible résultant du déclenchement de règles d'inférence d'une ontologie de traitement. Pour ce faire, le module de conversion exploite la

---

[106] Ici, le terme méthode est utilisé dans le contexte informatique du développement orienté objets et il fait référence à l'action associée à une classe.

[107] Le lien de composition entre un agent et une procédure indique que la procédure est exécutée par l'agent alors que le lien R est utilisé pour indiquer que l'agent déclenche l'exécution de la procédure. Par exemple, la procédure *créer* est exécutée par l'agent *commande* lorsque l'agent *invocateur* en fait la demande

propriété de modularité du *SWRL Built-in*[108] jumelé avec *SWRLBuiltInBridge*[109] qui sert d'API entre la déclaration de l'atome SWRL et son implantation Java. Pour l'exécution de règles SWRL, OntoCASE utilise l'interface *SWRLRuleEnginBridge*[110] (Golbreich, 2004 ; O'Connor *et al.*, 2008a ; O'Connor *et al.*, 2008b ; O'Connor *et al.*, 2005b) qui est un *pont*, au sens de Gamma *et al.* (1999), entre un modèle OWL contenant des règles SWRL et le moteur d'inférences à base de règles. C'est ce pont qui encapsule l'invocateur de notre architecture (voir le tableau 3.25**a**) dont le code est présenté à l'appendice F.16.3.

Comme présenté au tableau 3.25, le raccord d'une action Java à sa représentation en SWRL se divise en trois parties. La première partie (voir le tableau 3.25**a**) est la déclaration d'un *invoker*. La deuxième partie (voir le tableau 3.25**b**) présente un appel en SWRL par l'invocateur de la commande `OWLCreerUneClasseCmd` implantée en Java. La partie **c** du tableau 3.25 présente l'implantation *Java* de la commande `OWLCreerUneClasseCmd` dont la partie fonctionnelle de la commande est encapsulée dans la méthode `action()`. Pour cette commande, la méthode `action()` charge le singleton `owlModel` qui est l'objet contenant l'ontologie cible. Lorsque l'ontologie cible est disponible, la méthode associée `createOWLNamedClass` permet de créer une nouvelle classe dont le nom est `nomClasse` qui a préalablement été passée en argument lors de l'appel de l'*invoker* par la commande SWRL. Dans OntoCASE, des commandes telles que `CreerUneOntologieCmd`, `OWLCreerUneProprieteCmd`, `OWLCreerUneProprieteDatatypeCmd`, `OWLProprieteSetFonctionelleCmd`, `OWLProprieteSetInverseOfCmd`, `OWLEstUneInstanceDeCmd`, `OWLProprieteAjoutDunDomaineEtDuneImageCmd` et plusieurs autres (voir le tableau 3.25**d**), permettent la construction d'une ontologie à partir d'inférences sur des règles SWRL. On trouvera à l'appendice F.16 le code source détaillé nécessaire à l'implantation des commandes.

---

[108] http://www.daml.org/rules/proposal/builtins.html#8.1

[109] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLBuiltInBridge

[110] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLRuleEngineBridgeFAQ#nid6PM

Tableau 3.25
Exemple d'implantation d'une commande Java appelée lors du déclenchement d'une règle SWRL

a) Déclaration de l'atome *invoke* dans le fichier **`swrlbi.owl`**

```
<swrl:Builtin rdf:ID="invoker">
  <swrlb:minArgs rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >1</swrlb:minArgs>
</swrl:Builtin>
```

b) Appel de la commande Java **`OWLCreerUneClasseCmd`** à partir d'une règle SWRL de création d'une classe dans une ontologie de domaine

```
oRef:OR_Entite_Abstraite_Declaratif(?c)  ∘
oRef:OR_identifiant(?c, ?nc)
  → swrlbi:invoker("OWLCreerUneClasseCmd", ?nc)
```

c) Implantation de l'action de créer une classe OWL de la commande Java **`OWLCreerUneClasseCmd`**

```
public Boolean action() {
    owlModel = ontologieRegister.getOwlDomainModel();
    OWLNamedClass sousClasse =
        owlModel.createOWLNamedClass(nomClasse);
    return new Boolean(true);
}
```

d) Liste des commandes de la boite à outils pour la construction d'une ontologie cible

ImprimeMessageCmd.java	OWLCreerUneClasseCmd.java
OWLAjoutDuneProprieteEntreClasseEtIndividuCmd.java	OWLCreerUneClasseSousCmd.java
OWLAjoutDuneProprieteEntreIndividuCmd.java	OWLCreerUneInstanceSousCmd.java
OWLAjoutDuneProprieteEntreResourceCmd.java	OWLCreerUneProprieteCmd.java
OWLAssignerRestrictionAPourValeur_FloatCmd.java	OWLEstDisjointDeCmd.java
OWLAssignerRestrictionAPourValeur_IndividuCmd.java	OWLEstUneInstanceDeCmd.java
OWLAssignerRestrictionAPourValeur_IntegerCmd.java	OWLEstUneSousClasseDeCmd.java
OWLAssignerRestrictionAPourValeur_StringCmd.java	OWLEstUneSousProprieteDeCmd.java
OWLAssignerRestrictionExistencielCmd.java	OWLSupprimerSuperClasseDeCmd.java
OWLAssignerRestrictionUniverselCmd.java	printf.java
OWLAssignerUneValeurAUnAttributCmd.java	SauvegarderOntologieCmd.java
OWLProprieteDatatype_AjoutDunDomaineEtDunRangeType Cmd.java	EnvoyerMessageDerreurCmd.java
OWLProprieteSetFonctionelleCmd.java	OWLProprieteSetInverseOfCmd.java
OWLPropriete_AjoutDunDomaineCmd.java	OWLProprieteSetSymetricCmd.java
OWLPropriete_AjoutDunDomaineEtDuneImageCmd.java	OWLProprieteSetTransitiveCmd.java
OWLSupprimerLeTypeDeLinstanceCmd.java	
OWLSupprimerSuperProprieteDeCmd.java	
OWLCreerUneProprieteDatatypeCmd.java	
OWLProprieteSetInvFonctionelleCmd.java	

3.2.5.3 Développer le module de conversion

Le processus de conversion du modèle semi-formel désambiguïsé en ontologie formelle est réalisé par un système expert (`ExecuteRegles.java`) utilisant des axiomes issus de la définition de restrictions dans l'ontologie de transformation et de règles SWRL contenues dans les bases de règles à la transformation. Schématisé à la figure 3.13, le processus de conversion se décompose en cinq sous-processus.

Le sous-processus *Inférer*, dont les intrants sont l'ontologie de transformation et l'ontologie du modèle semi-formel désambigüisé, réalise une classification des entités du modèle semi-formel en vue du traitement ultérieur par les autres sous-processus de la transformation. Le tableau 3.26**a)** présente, à titre d'exemple, les axiomes de classification nécessaires au traitement d'une spécialisation entre deux concepts.

Les sous-processus *Appliquer la base de règles d'initialisation* et *Appliquer la base de règles de finalisation* concrétisent la structure de l'ontologie du domaine. Ils servent, d'une part, à créer le fichier ontologique et, d'autre part, à permettre la sauvegarde de l'ontologie du domaine sur disque.

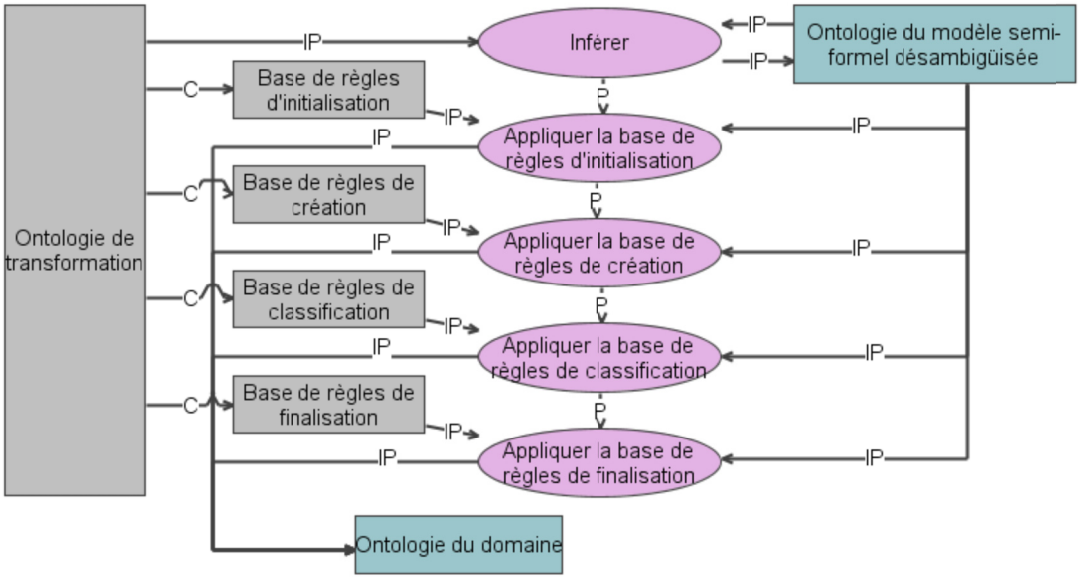


Figure 3.13: Sous-processus et ontologies impliqués dans la création de l'ontologie du domaine.

Le sous-processus *Appliquer la base de règles de création* accomplit la création des entités de l'ontologie. Qu'il s'agisse des classes, des propriétés ou des individus, ce sous-processus crée chacune des entités de l'ontologie du domaine. Toujours dans l'exemple de la création d'une spécialisation entre deux concepts, le tableau 3.26**b** présente la règle SWRL pour la création des classes associées aux concepts à représenter.

Tableau 3.26
Axiomes et règles impliqués dans la conversion d'une spécialisation entre deux concepts

a) Axiome (OWL-N3)	`oRef:OR_Entite_Abstraite_Declaratif owl:equivalentClass [ a owl:Restriction ; owl:hasValue oAmbig:Entite_Declarative ; owl:onProperty :OT_EntiteEstDeType ] .`
	`oRef:OR_Relation_Hyponyme owl:equivalentClass [ a owl:Restriction ; owl:hasValue oAmbig:Relation_Specialisation ; owl:onProperty :OT_EntiteEstDeType ] .`
	Interprétation de l'axiome :
	Toutes les entités qui ont la propriété OT_EntiteEstDeType assignée à oAmbig:Entite_Declarative sont des oRef:OR_Entite_Abstraite_Declaratif.
	De plus, toutes les relations qui ont la propriété OT_EntiteEstDeType assignée à oAmbig:Relation_Specialisation sont des oRef:OR_Relation_Hyponyme.

b) règle SWRL de création	`oRef:OR_Entite_Abstraite_Declaratif(?c) ^`
	`oRef:OR_identifiant(?c, ?nc)`
	`→ swrlbi:invoker("OWLCreerUneClasseCmd", ?nc) ^`
	`swrlbi:invoker("OWLEstUneSousClasseDeCmd", ?nc, "metaDom:MD_Declarative")`
	Interprétation de la règle :
	Soit une connaissance abstraite et déclarative (?c) ET son identifiant (?nc) ALORS : Il faut créer dans l'ontologie du domaine une classe du nom de ?nc et classer cette classe sous la classe metaDom:MD_Declarative

c) Règle SWRL de classification des entités	`oRef:OR_Relation_Hyponyme(?ls) ^`
	`oRef:OR_connSource(?ls, ?src) ^`
	`oRef:OR_connDestination(?ls, ?dest) ^`
	`oRef:OR_Entite_Abstraite_Declaratif(?src) ^`
	`oRef:OR_Entite_Abstraite_Declaratif(?dest) ^`
	`oRef:OR_identifiant(?src, ?nomSrc) ^`
	`oRef:OR_identifiant(?dest, ?nomDest)`
	`→ swrlbi:invoker("OWLEstUneSousClasseDeCmd", ?nomSrc, ?nomDest) ^`
	`swrlbi:invoker("OWLSupprimerSuperClasseDeCmd", "owl:Thing", ?nomSrc) ^`
	`swrlbi:invoker("OWLSupprimerSuperClasseDeCmd", "metaDom:MD_Declarative", ?nomSrc)`
	Interprétation de la règle :
	Soit un lienS (?ls) qui est relié à sa source par une connaissance source (?src) et relié à sa destination par une connaissance destination (?dest) ET ?src et ?dest sont des connaissances abstraites et déclarative dont les identifiants sont respectivement ?nomSrc et ?nomDest ALORS : il faut indiquer dans l'ontologie du domaine que la classe du nom ?nomSrc est une sous classe de ?nomDest ET indiquer que ?nomSrc n'est plus une sous classe directe de owlThing ni une sous classe directe de metaDom:MD_Declarative

Finalement, le sous-processus *Appliquer la base de règles de classification* permet de classer adéquatement les entités qui ont été préalablement créées en fonction de

l'ontologie cadre. C'est à cette étape que les domaines et codomaines seront associés aux propriétés. De même, les classes et les propriétés seront ordonnées selon la structure taxonomique représentée dans le modèle semi-formel. La règle SWRL du tableau 3.26**c** représente la règle de classification taxonomique entre deux concepts qui ont été unis par la relation de spécialisation.

3.2.6 Développement du module de validation

Tel que vu à la figure 2.10, p.110, la validation de l'ontologie du domaine comporte les deux sous-processus de validation syntaxique et sémantique. L'assistant comporte trois artéfacts à développer, dont les processus de développement sont schématisés à la figure 3.14. Au cœur de la validation sémantique, la classe `OntologyToInterpretation.java` est le programme qui produit le rapport d'interprétation sémantique. La construction de cette classe nécessite l'utilisation de l'ontologie cadre ainsi que la bibliothèque Java de manipulation de l'ontologie cadre précédemment conçue (voir la section 3.2.3 p.162). Deux classes Java sont nécessaires à la réalisation d'une validation syntaxique. La première classe `OwlToMot.java` reconstruit un modèle semi-formel à partir de l'ontologie du domaine. Pour ce faire, la classe utilise la bibliothèque Java de manipulation de l'ontologie cadre pour la lecture de l'ontologie du domaine, et la bibliothèque de code Java EMF de MOT pour la génération du modèle semi-formel reconstruit. À la fin de la reconstruction, la classe réalise une inférence sur l'ontologie cadre et inclut les résultats dans le modèle semi-formel reconstruit pour une consultation ultérieure par l'expert et l'ingénieur. L'appendice C présente le catalogue des axiomes et des règles qui permettent la génération des erreurs de cohérence.

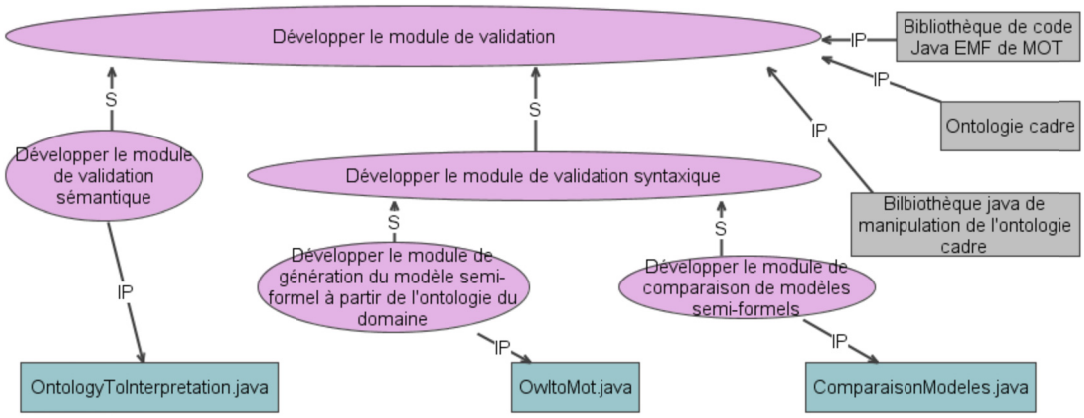


Figure 3.14: Modèle procédural du développement du module de validation.

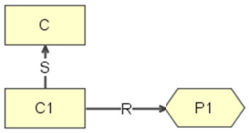
Finalement, la classe `ComparaisonModeles.java`, qui sert à produire le rapport de validation syntaxique, réalise la comparaison entre le modèle semi-formel source et le modèle semi-formel reconstruit. Les résultats de la comparaison (à savoir, l'identification des composants de modèles non équivalents) sont entreposés dans le rapport de validation syntaxique.

À titre d'exemple de rapport de validation syntaxique, le tableau 3.27 présente le rapport de validation syntaxique de deux modèles différents (voir le tableau 3.27 **a** et **b**). Les modèles sont volontairement différents afin de mettre en évidence les différentes sections du rapport de validation syntaxique. Les lignes 02 et 03 indiquent le nom des fichiers concernés dans la comparaison. La première section a pour sujet la comparaison des connaissances entre les deux modèles (voir les lignes 05 à 17 inclusivement). Il présente respectivement les éléments du modèle qui sont partagés par les deux modèles (voir la ligne 09), qui sont uniques au modèle d'origine (voir les lignes 12 et 13) et qui sont uniques au modèle reconstitué (voir les lignes 16 et 17). La deuxième section présente la comparaison des relations entre les deux modèles (voir les lignes 19 à 29). Elle présente respectivement les relations communes aux deux modèles (qui pour ce rapport spécifique ne possède aucune relation commune aux deux modèles), les relations

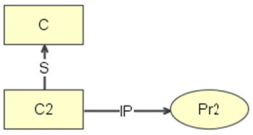
uniques au modèle d'origine (voir les lignes 24 et 25) et les relations uniques au modèle reconstitué (voir les lignes 28 et 29). Finalement, la dernière partie du rapport (voir les lignes 31 à 47) présente le bilan de la comparaison entre les deux modèles.

Tableau 3.27
Exemple de rapport complet de validation syntaxique

a)



b)



c)

```
01.     Comparaison des documents :
02.     Orig   = C:/Developpement/OntoCASE_these_260/runtime-OntoCASE_these.product/tst/mod2.mot
03.     Recons = C:/Developpement/OntoCASE_these_260/runtime-OntoCASE_these.product/tst/mod1_rest.mot
04.     Comparaison des documents :
05.     ------------------------------------------------------------------
06.     --------------------- Les connaissances  ----------------------
07.     ------------------------------------------------------------------
08.     ------------------- Commun aux deux modèles ---------------------
09.           Concept   : C_2
10.     ------------------------------------------------------------------
11.     ----------------Unique au modèle d'origine--------------------
12.           Concept   : C2_0
13.           Procédure : Pr2_1
14.     ------------------------------------------------------------------
15.     ----------------Unique au modèle reconstitué--------------------
16.           Concept   : C1_0
17.           Principe  : P1_1
18.     ------------------------------------------------------------------
19.     ---------------------- Les relations -------------------------
20.     ------------------------------------------------------------------
21.     ------------------- Commun aux deux modèles ---------------------
22.     ------------------------------------------------------------------
23.     ----------------Unique au modèle d'origine--------------------
24.           LienIP: source C2_0 : dest Pr2_1
25.           LienS: source C2_0 : dest C_2
26.     ------------------------------------------------------------------
27.     ----------------Unique au modèle reconstitué--------------------
28.           LienR: source C1_0 : dest P1_1
29.           LienS: source C1_0 : dest C_2
30.     ------------------------------------------------------------------
31.     ---------------------- BILAN -------------------------------
32.     ------------------------------------------------------------------
33.     ---------Totaux des divers composants des modèles --------------
34.     Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
35.     Compte des Procedures: Original = 1, Reconstruit = 0, Différence = 1
36.     Compte des Principes : Original = 0, Reconstruit = 1, Différence = 1
37.     Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
38.     Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
39.     Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
40.     Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
41.     Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
42.     Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
43.     Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
44.     Compte des LienIP    : Original = 1, Reconstruit = 0, Différence = 1
45.     Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
46.     Compte des LienR     : Original = 0, Reconstruit = 1, Différence = 1
47.     Compte des LienS     : Original = 1, Reconstruit = 1, Différence = 0
```

3.2.7 Développement des interfaces utilisateurs

Le dernier module d'importance d'OntoCASE est le module de communication avec l'utilisateur. En plus d'informer l'utilisateur sur l'état du déroulement des applications, le module de communication offre les interfaces nécessaires au pilotage de la méthodologie (les tableaux de bord). Les tableaux de bord sont directement connectés aux fonctionnalités d'édition et de traitement des modèles et ils sont les représentés fonctionnels du processus de formalisation et de validation de la méthodologie.

3.2.7.1 Fonctionnalités de communication avec l'utilisateur

L'assistant informatique d'OntoCASE offre plusieurs fonctionnalités de communication avec l'utilisateur. Certaines d'entres elles utilisent des modèles graphiques comme médium de communication (voir la figure 3.15 **a**) alors que d'autres utilisent le mode caractère (voir la figure 3.15 **b** et **c**).

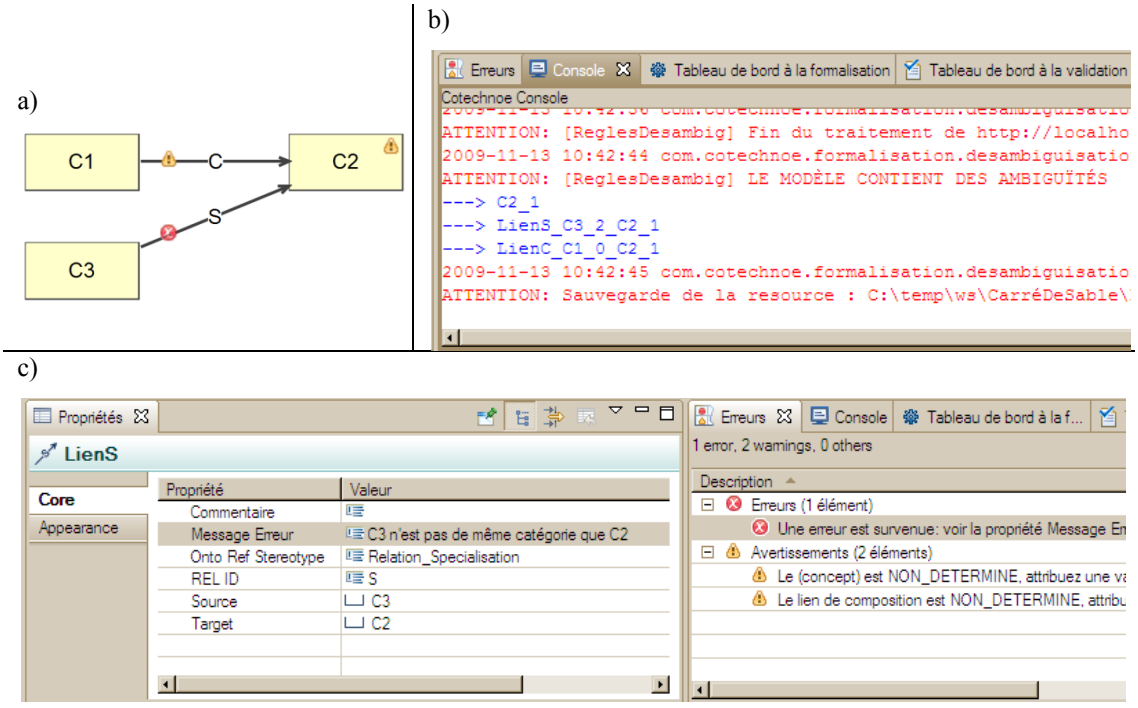


Figure 3.15: Interfaces de communication avec l'utilisateur.

L'exemple schématisé à la figure 3.15**a** illustre une situation d'ambiguïté et une situation d'erreur. Indiqué par l'icône , la situation d'ambiguïté survient lorsque l'assistant à la désambiguïsation ne peut pas désambiguïser l'élément de manière automatique. Dans cet exemple, l'assistant à la désambiguïsation ne peut pas identifier si le `LienC` est un attribut ou une composition. C'est alors à l'utilisateur de désambiguïser le cas. Les situations d'erreurs, qui sont indiquées par l'icône , surviennent lorsqu'une situation, apparemment valide selon le langage semi-formel, ne l'est pas selon la sémantique du langage représenté par l'ontologie cadre. Dans l'exemple, cette situation peut survenir si `C3` est désambigüisé en classe et que `C2` est désambigüisé en schéma. Il n'est donc pas permis de mettre un `LienS` entre ces deux concepts puisqu'une relation de spécialisation ne peut pas unir une classe et un schéma[111].

Tel que présenté à la figure 3.15**b** et **c**, l'assistant informatique utilise aussi la communication en mode caractère pour informer l'utilisateur. La *console* (en **b**) informe l'utilisateur pendant l'exécution du module d'importation, de traitement et de validation. Le journal d'exécution des modules ainsi que la trace d'exécution des règles de désambiguïsation et de formalisation y figurent aussi. Après l'exécution d'un module, certains composants du modèle peuvent être altérés. Les onglets Erreur et Propriété, présentés en **c**, permettent de consulter les altérations. Un clic sur un élément d'erreur permet au système de focaliser sur l'élément du modèle concerné et de faire apparaître la propriété correspondante. Les ajustements au modèle sont directement réalisés sur le modèle ou dans l'onglet des propriétés.

3.2.7.2 Tableau de bord à la formalisation

Le module de formalisation, dont l'interface est un tableau de bord (voir la figure 3.16), est un module qui assiste l'ingénieur dans la tâche de formalisation du

---

[111] Bien que la classe et le schéma soient tous les deux représentés par un concept, ceux-ci sont définis dans deux catégories distinctes dans l'ontologie cadre (voir le tableau 2.2, p.86).

modèle semi-formel en ontologie selon les étapes présentées à la section 4.4.6. Les trois étapes d'importation, de désambiguïsation et de conversion qui composent la réalisation de cette tâche y sont représentées. À chacune des étapes, l'ingénieur a la possibilité d'éditer le modèle résultant. La sélection d'*Importer,* de *Désambiguïser* et *Convertir* active directement les modules de traitements associés à l'importation (voir la section 3.2.2), de la désambiguïsation (voir la section 3.2.4) à la conversion (voir la section 3.2.4.2).

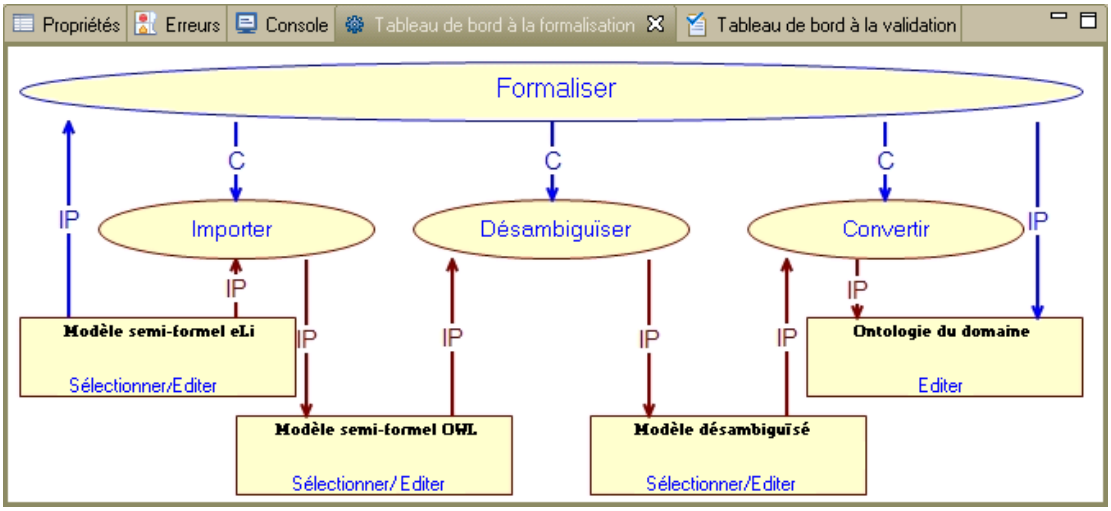


Figure 3.16: Le tableau de bord à la formalisation.

3.2.7.3 Tableau de bord à la validation

Le module de validation permet à l'ingénieur et à l'expert de contenu d'accéder aux outils nécessaires à la réalisation d'une validation syntaxique et sémantique telle que décrite à la section 3.2.6. Ainsi, les acteurs de cette activité ont accès à des outils permettant la production de conclusions automatiques nécessaires à l'évaluation sémantique de l'ontologie. Ils ont aussi accès à des outils de génération d'un modèle semi-formel à partir de l'ontologie de domaine ainsi qu'à des outils de comparaison entre le modèle d'origine et le modèle généré afin d'assumer la validation syntaxique de l'ontologie du domaine. L'appel à ces outils est assuré par le tableau de bord à la validation (voir la figure 3.17) qui guide l'utilisateur dans la méthode de validation.

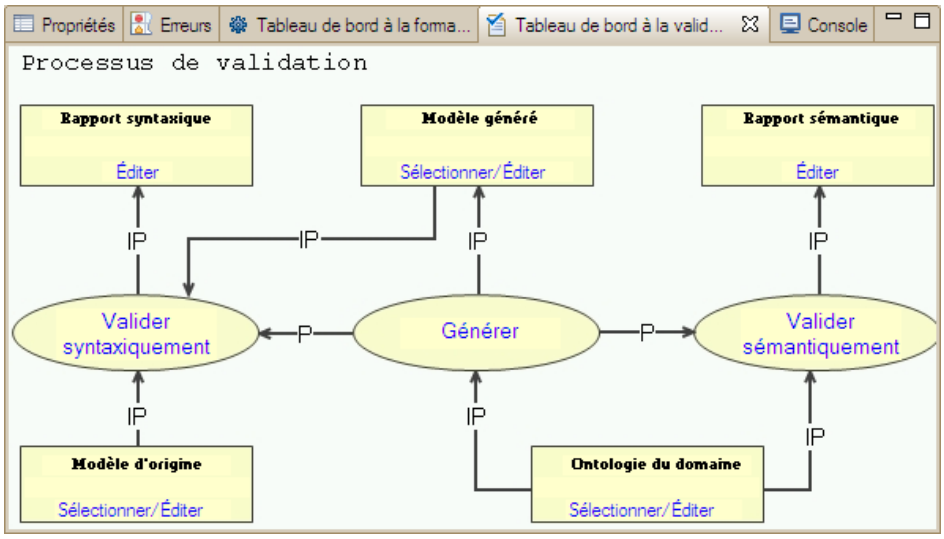


Figure 3.17: Le tableau de bord à la validation.

3.3 Phase 3: Consolidation

La phase de consolidation a pour sujet la validation de la démarche de recherche. La première étape de cette phase est la détermination des facettes d'OntoCASE qui doivent être validées par des cas de test. L'exécution automatique des cas de test est assurée par la mise en place d'un banc d'essais. Finalement, en fonction des facettes à évaluer, une méthode de conception et d'exécution des tests est élaborée.

3.3.1 Cas de test

Couramment utilisés en ingénierie logicielle, les *cas de test* "*testCase*" (Burnstein, 2003 ; Jacobson *et al.*, 1999) sont des applications informatiques qui mesurent les réponses d'un système par la comparaison des résultats obtenus *versus* les résultats attendus. Les cas de tests sont regroupés en *scénarios de test* pour évaluer des systèmes selon des perspectives contextuelles particulières. Les tests sont appliqués à des niveaux spécifiques des applications.

Le test de niveau unitaire (*unit test*) s'adresse au premier niveau des applications, c'est-à-dire aux fonctions, procédures, classes et méthodes. Il a pour objectif d'évaluer les réponses des entités de base d'un système et de signaler tout changement dans ses fonctionnalités élémentaires. Le test de niveau intégration (*integration test*) mesure la réponse d'un système lorsque ses divers composants sont en interrelation les uns avec les autres. Finalement, le test de niveau systémique (*system test*) évalue le système avec des cas de test se rapprochant davantage des situations d'usage de l'application. Les tests de ce niveau sont utilisés pour mesurer les fonctions d'usage du système, évaluer ses performances, sa capacité à réagir au stress ou à toute autre caractéristique de niveau système telles que la sécurité, la configuration ou la restauration d'informations.

3.3.2 Concevoir le banc d'essais

Le banc d'essais d'OntoCASE est une application *JUnit*[112] intégrée à *Eclipse* qui permet l'exécution de *scénarios de test*. Composé de plusieurs *cas de test*, le *scénario de test* permet de valider de manière systématique et automatique les fonctionnalités d'un système. Pour chacune des fonctionnalités à tester, l'ingénieur responsable du test produit un *résultat attendu* qui servira de point de comparaison au *résultat produit* par l'appel de la fonction à tester. Les dispositions prises à la suite de la comparaison sont programmées dans l'application qui exécute le cas de test. La figure 3.18 présente le modèle procédural associé à l'exécution des scénarios de test réalisé pour valider OntoCASE. Dans cette optique, le scénario de cas de test se compose de plusieurs cas de test dont la première étape d'exécution est la formalisation d'un modèle semi-formel *attendu* en ontologie de domaine. Selon les spécificités à évaluer par le scénario de test, le modèle semi-formel *attendu* pourra avoir subi une désambiguïsation manuelle si nécessaire. En revanche, dans le cas d'ambiguïté sémantique, le modèle semi-formel *attendu* devra être systématiquement

---

[112] JUnit.org, *Resources for Test Driven Development*: http://www.junit.org/about

désambiguïsé manuellement avant le déclenchement du cas de test correspondant. La deuxième étape consiste à utiliser le module de validation syntaxique d'OntoCASE pour générer, à partir de l'ontologie de domaine, le modèle semi-formel *produit*.

La comparaison des modèles semi-formels *attendu* et *produit* (réalisée à l'étape 3) s'effectue par une comparaison croisée des entités et des relations contenus dans les modèles. La comparaison croisée consiste à faire une recherche de tous les éléments du modèle attendu dans le modèle produit et *vice-versa*. De cette manière, il est assuré que les deux modèles sont identiques en nombre et en équivalence pour chacun des éléments. C'est après l'exécution des comparaisons croisées que la production des rapports de totaux est réalisée. Ce rapport indique les entités de modèles qui diffèrent et permet la rédaction du bilan de comparaison entre les modèles.

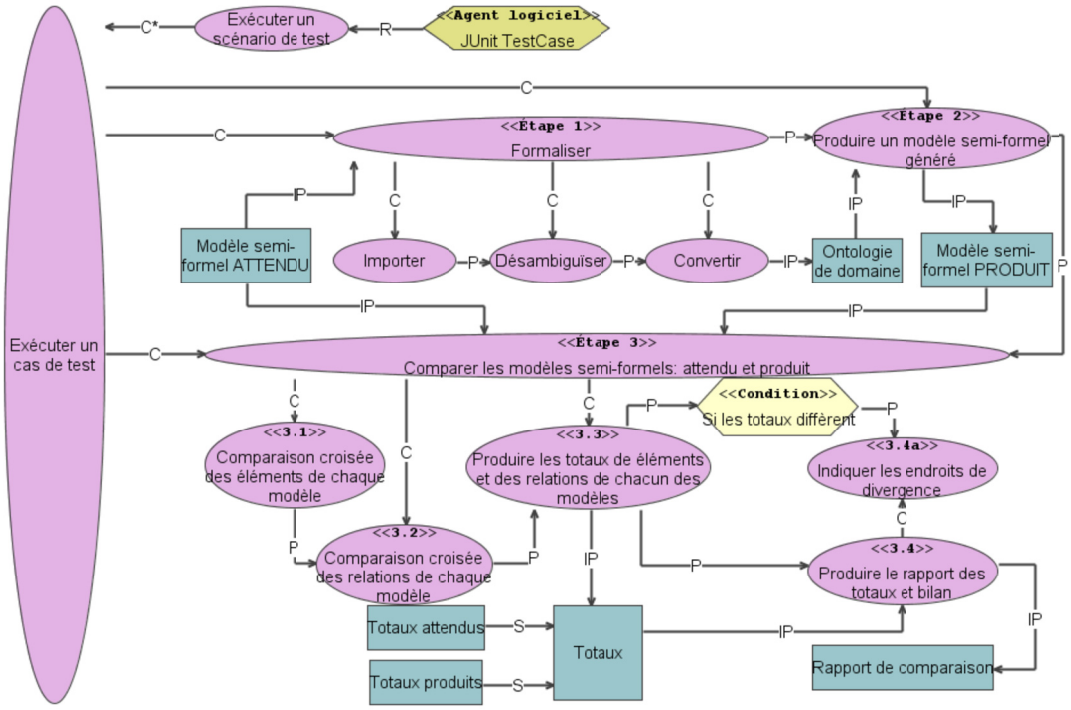


Figure 3.18: Exécution d'un cas de test pour la mesure d'efficacité d'OntoCASE.

3.3.3 Concevoir et exécuter les cas de test

Afin de consolider OntoCASE, nous avons conçu une série de scénarios de test que nous avons catégorisés en vue de tester des aspects spécifiques. La première catégorie, le scénario de cas de test *unitaire*, est conçue pour valider l'étape de conversion du processus de formalisation. Pour ce faire, chacun des modèles du cas de test est préalablement désambiguïsé par le concepteur. Le scénario de test d'*intégration* vise la validation de l'étape de désambiguïsation topologique et typologique. Le scénario de test *système* est conçu afin de valider la capacité d'OntoCASE à formaliser des modèles complexes contenant des représentations de type mixte. Finalement, le scénario de test d'*incohérence* est conçu pour évaluer la capacité d'OntoCASE à identifier et communiquer des erreurs de modélisation à l'utilisateur. L'appendice C présente le catalogue des axiomes et des règles qui permettent la génération des erreurs de cohérence. Au chapitre suivant, nous verrons plus en détail les scénarios de test qui ont servi à valider OntoCASE.

CHAPITRE 4

VALIDATION D'ONTOCASE

Tel que vu précédemment, la méthodologie d'OntoCASE se compose de trois volets : un volet méthodologique, un volet représentationnel et un volet computationnel. Ce chapitre présente les résultats des activités réalisées en vue de vérifier l'hypothèse présentée au chapitre 2, à savoir qu'il est possible de soutenir la formalisation de connaissances procédurales, stratégiques et déclaratives, exprimées de manière semi-formelle, en connaissances déclaratives dans un formalisme formel, et plus spécifiquement en ontologie, et ce, par l'exécution d'un ensemble de méthodes soutenues de manière automatique ou semi-automatique.

Pour vérifier cette hypothèse, nous avons procédé à une validation d'OntoCASE selon les trois axes suivants:

- Généralité des types de connaissances à formaliser

  Il s'agit de démontrer que les connaissances de divers types (déclaratives, procédurales, stratégiques et factuelles) présentes dans le modèle semi-formel sont représentées dans l'ontologie produite.

- Ergonomie d'OntoCASE

  La validation ergonomique a pour objectif de valider la capacité d'OntoCASE à être utilisé par des utilisateurs autres que son concepteur. L'ergonomie d'OntoCASE sera démontrée par une expérimentation en laboratoire où des

utilisateurs seront mis en situation de formalisation ontologique de modèles semi-formels.

- Généricité des langages semi-formels utilisés pour la formalisation

  Cette dimension sera démontrée en appliquant le scénario de transformation d'OntoCASE à partir d'un modèle semi-formel construit à l'aide d'un autre langage que MOT, à savoir le langage *MindMap* (Buzan et Buzan, 1994).

Nous présentons les résultats relatifs à chacun de ces axes dans ce chapitre selon le modèle de l'utilisabilité tel que défini par Brangier et Barcenilla (2003).

4.1 Modèle d'utilisabilité

Pour Brangier et Barcenilla (2003), l'*utilité* d'un dispositif technique concerne sa capacité à répondre aux besoins réels des utilisateurs alors que l'*utilisabilité* concerne sa facilité à être utilisé par une personne donnée de façon à accomplir la tâche pour laquelle cet objet a été conçu. Trois critères sont proposés pour évaluer l'ergonomie d'un logiciel:

- l'*efficacité*, qui est le degré de réalisation des objectifs visés par le logiciel;
- l'*efficience*, qui est la capacité de réaliser une tâche donnée avec le minimum d'effort;
- la *satisfaction*, qui est le niveau de confort ressenti par l'usager du logiciel.

Nous allons appliquer ces trois critères à la validation d'OntoCASE.

La mesure de l'efficacité est le plus mécanique des mesures. Elle doit déterminer si le dispositif accomplit la tâche pour laquelle il a été conçu. Nous appliquons cette mesure aux premier et troisième axes, *généralité des types de connaissances à formaliser* et *généricité des langages semi-formels utilisés pour la formalisation* de la validation d'OntoCASE. Dans chacun de ces deux axes, il s'agit de mesurer la capacité d'OntoCASE à formaliser en ontologie un modèle semi-formel contenant

divers types de connaissances par la mesure de l'efficacité du logiciel à exécuter cette tâche. Pour ce faire, une série de cas de tests unitaires, d'intégration et de système sont développés et mécaniquement appliqués. Un premier ensemble de scénarios de test (les tests de véracité) démontre la capacité du logiciel à désambiguïser et à transformer correctement des modèles qui sont syntaxiquement valides. Le deuxième ensemble de scénarios de test (les tests d'erreurs) valide le logiciel dans son efficacité à identifier des situations de modélisation erronées.

La mesure de l'efficience est une évaluation plus subjective que la mesure de l'efficacité, qui comporte tout de même des indicateurs objectifs tels que « *[...] le taux et la nature des erreurs d'utilisation; le temps d'exécution d'une tâche donnée; le nombre d'opérations requises [...]; la charge de travail* » [113]. Des techniques d'évaluation telles que l'analyse en situation d'utilisation, les groupes de discussions, l'entretien structuré, le questionnaire, l'observation, le monitorage de l'utilisation et les retours d'expériences permettent d'obtenir des indicateurs de mesure de l'efficience.

La satisfaction est la composante de l'utilisabilité qui fait référence à la dimension du confort d'utilisation du dispositif, à la réaction affective d'agrément ou de désagrément d'utilisation. Les aspects émotionnels de l'utilisation sont modulés par la stabilité de l'application, la facilité d'apprentissage, le sentiment que l'application permet de sauver du temps de travail et même la stimulation de la créativité de l'utilisateur. Les techniques d'évaluation décrites dans la mesure de l'efficience peuvent aussi servir à la mesure de la satisfaction.

L'efficacité, l'efficience et la satisfaction sont des mesures importantes de l'ergonomie d'OntoCASE

---

[113] Brangier et Barcenilla, *Concevoir un produit facile à utiliser* p. 47.

4.2 Généralité des types de connaissances à formaliser et mesure de l'efficacité d'OntoCASE

L'évaluation de la généralité des types de connaissances à formaliser est réalisée par l'implantation de cas de tests unitaires, d'intégration et de système (Burnstein, 2003 ; Dirk, 2008). D'un point de vue méthodologique, ces cas de tests valident de manière systématique chacune des règles de formalisation appliquée à chacun des scénarios possibles de modélisation. Du point de vue applicatif, les cas de test valident la mécanique informatique qui réalise la formalisation de chacun des scénarios de modélisation. Les scénarios et cas de tests décrits plus bas sont réalisés grâce au banc d'essai réalisé à la phase 3 de la démarche de construction d'OntoCASE décrite au chapitre 3 et les résultats détaillés sont présentés à l'appendice D.

4.2.1 Scénario de tests unitaires

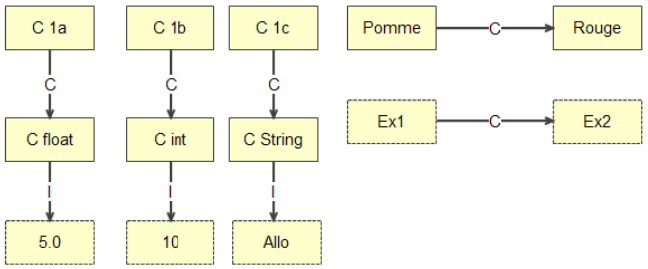
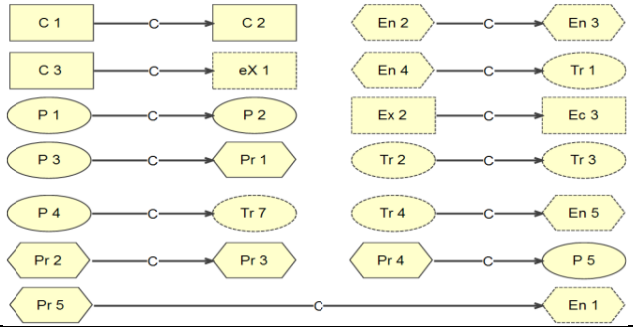
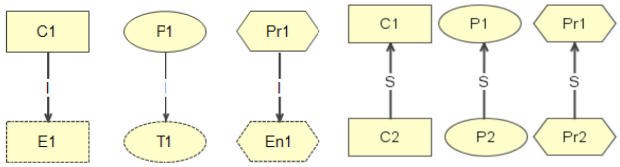
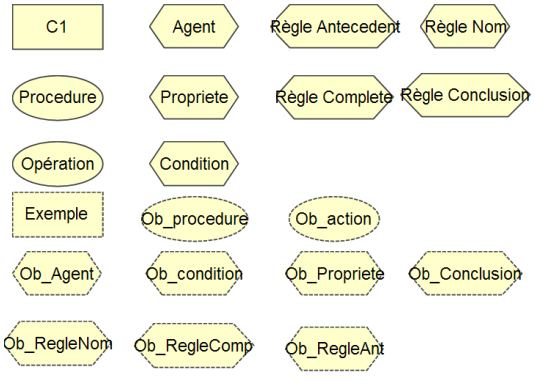
Le test unitaire consiste à introduire dans le processus de test des modèles préalablement désambiguïsés. Cette caractéristique des modèles permet de focaliser les tests sur la validation du processus de conversion. Le tableau 4.2 (dont la structure est présentée au tableau 4.1) présente quelques-uns des cas de tests unitaires qui ont été appliqués à OntoCASE.

Tableau 4.1
Structure de présentation des divers cas de test unitaires.

Nom du test	Représentation en MOT
Description	

En plus de ceux-ci, le scénario de tests unitaires comporte des tests pour l'évaluation des liens de MOT ainsi que pour la construction d'entités ontologiques de type propriétés, de classes et d'individus dans le contexte de l'ontologie cadre. Le scénario de tests unitaires comporte dix-sept cas de tests différents (voir l'appendice D.1).

Tableau 4.2
Quelques cas de test unitaire

a) Lien d'attribution Ce test valide l'utilisation d'un `mot:LienC` pour désigner qu'un schéma ou un concept est attribut d'un concept. Ce test valide aussi l'assignation d'une valeur à un attribut.	
b) L'holonyme Ce test valide l'utilisation du `mot:LienC` pour la désignation d'une composition. La composition est validée pour plusieurs types de connaissances de niveaux de représentation abstrait et concret.	
c) L'instanciation et hyponyme Ce test valide l'utilisation du `mot:LienI` et du `mot:lienS` pour l'instanciation d'une connaissance abstraite en connaissance concrète et pour la subsomption entre connaissances abstraites.	
d) Les entités Ce test valide l'utilisation des entités abstraites et concrètes de MOT pour désigner les différentes entités de l'ontologie cadre.	

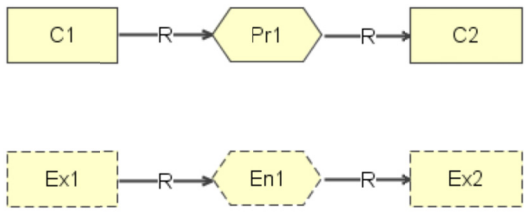
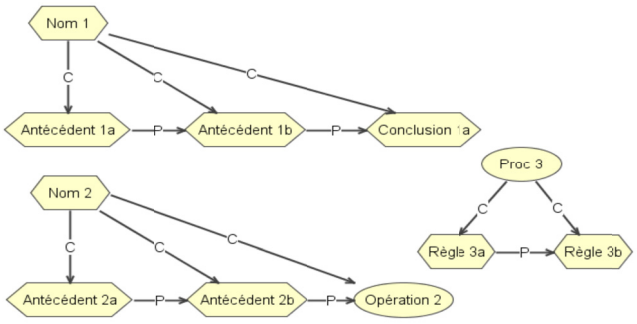
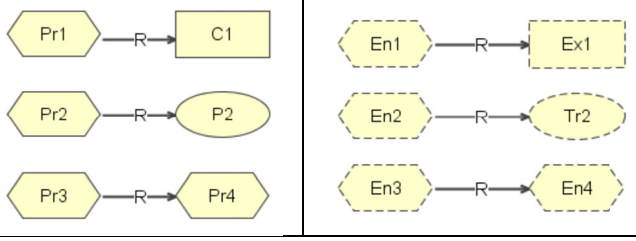
Les tests unitaires ainsi que les tests d'intégration qui suivent s'adressent à la partie vocabulaire et à celle de la grammaire du langage. C'est pourquoi les libellés de chacun des éléments sont sémantiquement peu significatifs.

4.2.2 Scénario de tests d'intégration

Les modèles utilisés pour les tests d'intégration sont identiques à ceux utilisés pour les tests unitaires, à ceci près qu'aucune désambiguïsation n'est réalisée *a priori* sur les modèles à tester. Ce scénario de test met donc l'accent sur l'évaluation de l'efficacité du système (méthode et application) à désambiguïser typologiquement et topologiquement les modèles semi-formels.

Le tableau 4.3 présente quelques cas de tests d'intégration nécessitant (dans les deux premiers cas) une désambiguïsation topologique.

Tableau 4.3
Quelques cas de tests d'intégration

a) Propriété et assertion Ce test d'intégration valide l'utilisation d'une connaissance stratégique entre deux connaissances déclaratives pour la représentation d'une propriété qui unit deux classes et l'assertion d'une propriété entre deux individus. La désambiguïsation de ce modèle est de type topologique.	
b) Construction d'une règle à conclusion unique Ce test valide la représentation d'une règle de type *si 'antécédent' alors 'conclusion unique', si 'antécédent' alors 'opération'* et *la procédure se compose de règle.*	
c) Régulation entre entités abstraites et concrètes Ce test valide la représentation d'une propriété de régulation entre connaissances abstraites et concrètes.	

Notons que les cas de tests nécessitant une désambiguïsation sémantique (par exemple, les représentations utilisant le `mot:LienC`) sont inclus dans le scénario des

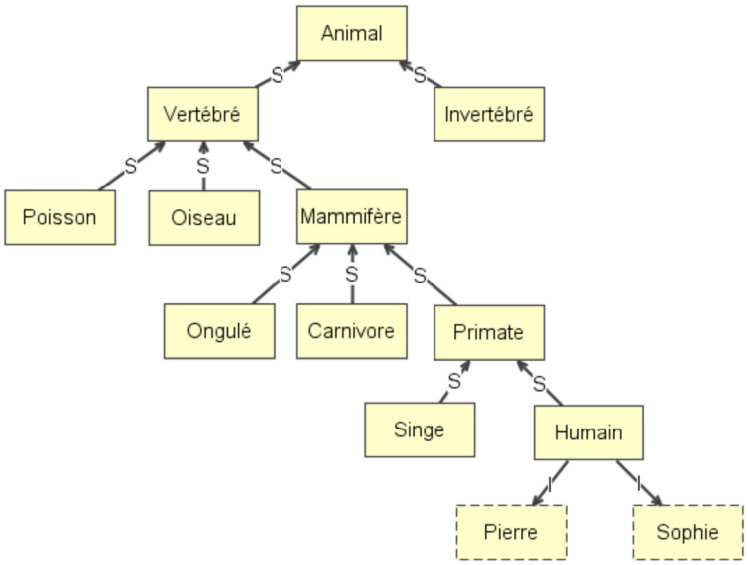
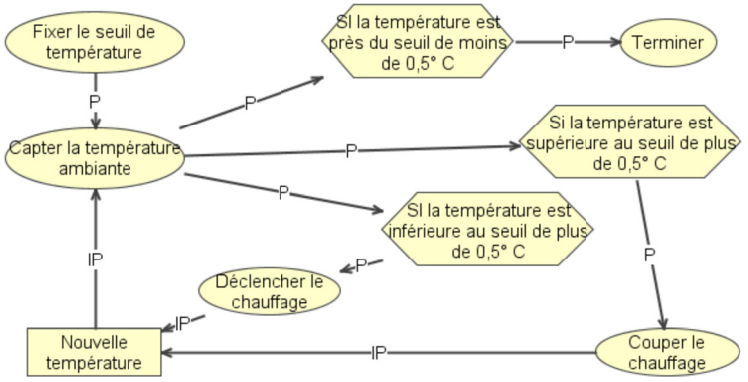
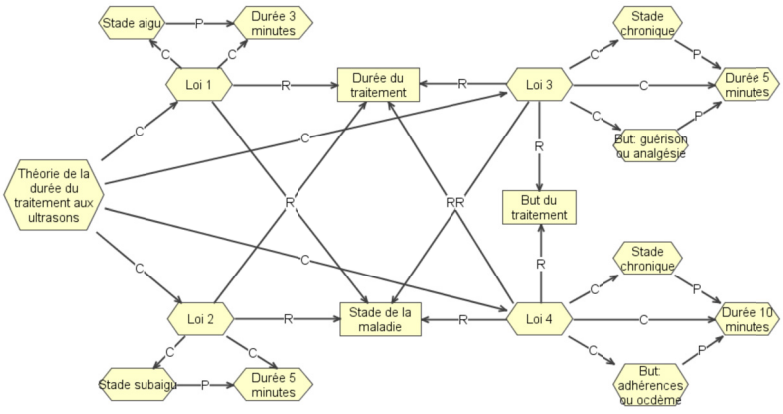
tests unitaires. Le scénario de cas de tests d'intégration comporte au total dix-neuf cas de tests différents (voir appendice D.2).

4.2.3 Scénario de tests systèmes

Principalement orienté sur la sémantique des modèles, ce scénario de test met l'accent sur la formalisation de modèles sémantiquement significatifs. Pour ce faire, nous avons utilisé la majeure partie des exemples de modèles MOT présentés aux chapitres 2 et 3 de l'ouvrage de Paquette (2002b). Une désambiguïsation sémantique a été réalisée pour les modèles dont la sémantique l'exigeait. Pour tous les autres modèles, aucune désambiguïsation sémantique n'a été réalisée laissant ainsi la réalisation de cette tâche au système.

Le tableau 4.4 présente quelques cas de test systèmes. En plus de ceux présentés (taxonomie, procédure et définition d'une loi), le scénario teste des cas tels que contrôle de procédure; arbre de décision, procédure séquentielle. Au total, le scénario de tests systèmes comporte vingt-et-un cas de tests différents (voir l'appendice D.3).

Tableau 4.4
Quelques cas de tests systèmes

a) Taxonomie Ce cas de test permet de valider la représentation d'une taxonomie complétée par l'instanciation de deux individus.	
b) Procédure Ce cas de test permet de valider la représentation de la relation de précédence mise en contexte avec des procédures et des conditions. De plus, il valide l'utilisation de la relation d'intrant et de produit entre un objet et une procédure.	
c) Définition d'une loi Ce cas de test valide l'utilisation du principe dans le contexte de définition d'une règle. L'utilisation des divers `mot:LienC` et `mot:LienP` en conjonction avec les `mot:Principe` adjacents permettent au module de désambiguïsation d'identifier des patrons de définition de règles.	

4.2.4 Scénario de tests du mécanisme de détection d'incohérence

Les scénarios de tests unitaires, d'intégration et de systèmes utilisent des modèles qui sont sémantiquement et syntaxiquement valides. Ils sont spécialement conçus pour valider le processus de désambiguïsation et de conversion des modèles semi-formels. En revanche, les scénarios de tests d'incohérence ont pour objectif de tester le système de détection des incohérences d'OntoCASE. Pour OntoCASE, l'incohérence est une erreur qui survient lorsque certains éléments de l'ontologie cible ne respectent pas la grammaire de l'ontologie cadre. Ces erreurs sont souvent engendrées par une mauvaise désambiguïsation des éléments du modèle semi-formel. Par exemple, dans un modèle semi-formel, la représentation d'une relation d'instanciation entre un principe et un énoncé pourrait être désambiguïsée par erreur en instanciation entre un `oRef:OR_Entite_Principe_Agent` et un `oRef:OR_Entite_Principe_Condition`, ce qui est une incohérence puisque dans la grammaire de l'ontologie cadre, une condition ne peut pas être l'instance d'un agent. Ces erreurs ne peuvent pas être identifiées par l'éditeur de modèles semi-formels puisque la catégorisation des éléments du modèle semi-formel est traitée à l'étape de la formalisation. Il importe donc d'implanter un mécanisme d'identification d'erreurs qui soit actif pendant l'étape de formalisation. C'est la mesure de l'efficacité de ce mécanisme qui est prise par le scénario de tests d'incohérence. L'appendice D.4 présente les quatre scénarios de tests d'incohérence qui sont appliqués pour valider la cohérence de l'ontologie cible. L'appendice C présente le catalogue des axiomes et des règles qui permettent la génération des erreurs de cohérence.

4.3 Ergonomie d'OntoCASE

L'évaluation de l'ergonomie d'un dispositif est une activité complexe puisqu'elle doit considérer des aspects objectifs et subjectifs de l'utilisation d'un dispositif. La mesure de l'efficience et de la satisfaction est une façon d'obtenir des informations sur la qualité de l'ergonomie. Dans cette section, nous présentons d'abord les mécanismes

qui sont mis en place dans OntoCASE dans le but d'offrir un logiciel efficient à son utilisateur. Quant à la mesure de l'efficience et de la satisfaction, elle est réalisée à partir d'une évaluation en laboratoire dont nous présenterons le protocole et le résultat à la section 4.3.4.

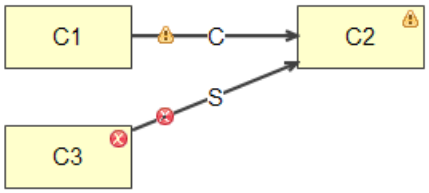
4.3.1 Mécanismes visant à assurer l'efficience d'OntoCASE et la satisfaction de l'usager

La volonté de maintenir l'efficience et la satisfaction à un haut niveau est une qualité importante d'OntoCASE . Pour ce faire, nous proposons une application informatique intelligente, soutenant la méthodologie de formalisation, puis l'implantation de mécanismes de rétroaction conviviaux qui informent l'ingénieur sur l'état de la formalisation pendant son déroulement.

4.3.2 Mécanisme de rétroaction d'OntoCASE

Le mécanisme de rétroaction est un ensemble de dispositifs qui informent l'utilisateur sur l'état des processus en cours à l'aide d'une application qui se veut conviviale. Deux dispositifs visuels sont présentés à l'utilisateur d'OntoCASE. Une console de présentation des messages générés lors de la phase de désambiguïsation et de formalisation constitue le premier dispositif de rétroaction. La lecture des messages fournit notamment des informations sur les règles qui ont été déclenchées pour réaliser la formalisation ainsi que sur leur contenu. Le deuxième dispositif de rétroaction est la fenêtre d'édition du modèle semi-formel. En plus de présenter le modèle sous une forme graphique, l'éditeur identifie les ambiguïtés directement dans le modèle au moyen d'icônes spécifiques (voir le tableau 4.5).

Tableau 4.5
Icônes de rétroaction des ambiguïtés et des erreurs d'incohérence

Signification d'ambiguïtés et d'erreurs et leur représentation	
Dans cet exemple, l'assistant indique que le `mot:lienC` et le `mot:Concept(C2)` sont en ambiguïtés et que le `mot:Concept(C3)` et le `mot: Liens` sont en erreur.	

4.3.3 Mécanisme de pilotage de l'ingénieur

Un autre dispositif que nous estimons important dans le but d'accroître l'efficience d'OntoCASE est l'utilisation de tableaux de bord qui guident l'utilisateur dans la méthodologie. L'assistant d'OntoCASE comporte deux tableaux de bord, soit le *tableau de bord à la formalisation* (voir la figure 3.16 p.179) et le *tableau de bord à la validation* (voir la figure 3.17, p.180). Il est à remarquer que les fonctionnalités des tableaux de bord sont présentées dans le formalisme graphique et semi-formel MOT du fait de sa simplicité et de sa facilité d'interprétation. Le tableau de bord à la formalisation guide l'ingénieur dans le processus de formalisation aux étapes d'importation, de désambiguïsation et de transformation. À chacune des étapes, l'utilisateur peut éditer les fichiers concernés ou sélectionner d'autres fichiers en intrants aux processus. Puis, il n'a qu'à cliquer sur chacun des processus représentés pour en déclencher le déploiement. L'utilisateur suit alors l'évolution de la formalisation par l'intermédiaire des mécanismes de rétroaction.

Un aspect important de l'efficience est la possibilité pour l'ingénieur et l'expert de valider le résultat produit. Le tableau de bord à la validation vise à les assister pour ce faire. La génération des rapports de validation syntaxique et sémantique peut y être produite et éditée de façon intuitive.

4.3.4 Évaluation expérimentale de l’ergonomie d’OntoCASE

Il est difficile d'établir des indicateurs objectifs sans point de comparaison avec des données préalablement établies. Cependant, il est possible d'obtenir des indicateurs de temps de réalisation d'une tâche, de temps d'apprentissage, de calcul d'erreur et de charge de travail pour réaliser une tâche ainsi qu’une appréciation globale sur l’utilisation. Ces indicateurs ont été vérifiés par une expérimentation en laboratoire. Dans cette section, nous décrivons d’abord le protocole expérimental utilisé et brossons ensuite un bilan des résultats de cette expérimentation.

4.3.4.1 Objectif général de l'évaluation expérimentale

L'évaluation expérimentale a donc pour objectif de valider l’ergonomie d'OntoCASE en termes d'efficacité (degré de réalisation des objectifs poursuivis en matière d'utilisation), d'efficience (capacité de produire une tâche donnée avec le minimum d'effort) et de satisfaction (niveau de confort ressenti en utilisant le logiciel).

De plus, l’expérimentation vise à évaluer la capacité d'OntoCASE à résister à des situations non prévues qui surviennent en situation réelle de formalisation.

4.3.4.2 Objectifs secondaires

Du point de vue de l'efficacité, l'évaluation d'OntoCASE vérifie que l'utilisateur peut :

- éditer (créer, modifier, sauvegarder) des modèles semi-formels dans le formalisme du langage MOT;
- éditer (créer, modifier, sauvegarder) des ontologies en OWL;
- formaliser en ontologie des connaissances déclaratives, procédurales et stratégiques à partir d'un modèle semi-formel;
- identifier les connaissances ambiguës et assister l’usager pour la désambiguïsation;
- réaliser une validation syntaxique et sémantique de l'ontologie cible.

Du point de vue de l'efficience, l'évaluation d'OntoCASE doit permettre de:

- mesurer le temps nécessaire à:
    - réaliser une formalisation ne contenant aucune ambiguïté;
    - réaliser une formalisation contenant des ambiguïtés;
    - réaliser une validation syntaxique et sémantique;
- quantifier le nombre d'opérations nécessaires pour réaliser des opérations de formalisation et de validation;
- mesurer la capacité à être découvert de manière intuitive;
- quantifier les bonnes et les mauvaises utilisations.

Du point de vue de la satisfaction, l'évaluation d'OntoCASE doit permettre d'identifier :

- des aspects d'OntoCASE qui peuvent être frustrants pour l'usager ;
- les aspects d'OntoCASE jugés satisfaisants du point de vue de représentants d'usagers cibles ;
- d'autres champs potentiels d'utilisation.

4.3.4.3 Méthode d'évaluation

En ce qui concerne la méthode d'évaluation, nous ferons appel à l'*analyse en situation d'utilisation* ainsi qu'au *monitorage de l'utilisation.* Quatre participants familiers avec la modélisation par objets typés ont été sollicités pour participer, de manière individuelle, à l'expérimentation. Chaque séance d'expérimentation a été enregistrée en audio. Les principaux éléments du verbatim de chacune des séances, le détail du protocole d'expérimentation ainsi qu'une copie du certificat d'éthique sont présentés à l'appendice G.

4.3.5 Déroulement du test d'utilisabilité

Le test d'utilisabilité consiste à mettre des participants en situation d'utilisation plus ou moins contrôlée selon des scénarios prédéfinis afin de mesurer l'utilisabilité du produit (Brangier et Barcenilla, 2003 p. 218). L'exécution du test d'utilisabilité est guidée par les objectifs d'évaluation de l'efficacité, de l'efficience et de la satisfaction d'OntoCASE.

La séance d'expérimentation se divise en quatre phases:

- la première phase, la *familiarisation*, a pour objectif d'initier le participant aux aspects méthodologiques et informatiques d'OntoCASE. Cette initiation a été faite par l'expérimentateur;
- la deuxième phase, la *formalisation*, a pour objectif d'initier le participant au processus de formalisation et de validation de modèles et de permettre à celui-ci de réaliser une première manipulation. Un modèle semi-formel d'expérimentation déjà conçu est alors présenté au participant (voir la figure 4.1) afin qu'il soit formalisé en ontologie. Cette phase est aussi animée par l'expérimentateur et les manipulations sont réalisées par le participant;
- au cours de la troisième phase, la *modélisation et la formalisation libres*, le participant est invité à modéliser un court texte en langage semi-formel puis à le formaliser en ontologie à l'aide d'OntoCASE, et ce, avec guidage uniquement sur demande. Cette phase permet de mesurer la capacité du logiciel et de la méthodologie à résister aux différentes approches des participants;
- finalement, la quatrième phase, le *debriefing*, est consacrée au recueil des commentaires du participant après son expérimentation d'OntoCASE au cours d'une courte entrevue semi dirigée.

4.3.6 Modèle semi-formel d'expérimentation

Le modèle semi-formel d'expérimentation de la figure 4.1, portant sur le thème du système solaire, présente le modèle employé lors de la phase 2 de l'expérimentation. Certaines erreurs de modélisation ainsi que quelques particularités de modélisation ont été volontairement introduites dans le modèle afin de mesurer l'efficience d'OntoCASE.

- L'étiquette **a** illustre un exemple d'erreur de modélisation couramment observée. Il s'agit d'une erreur de classification du niveau d'abstraction des "choses" représentées. Il arrive qu'un modélisateur confonde la représentation d'un objet concret avec l'abstraction de l'objet. Il est vrai que le concept de *Lune* est une sorte de *Satellite naturel de la Terre*, mais, de manière formelle, ce qui est représenté ici devrait se lire *l'objet de type Lune est un exemple (une instance par le lien I) des "satellites naturels" de la Terre.*
- L'étiquette **b** illustre une autre erreur de représentation, soit la confusion dans l'utilisation du *LienS* (sorte de [is-a]) et du *LienC* (composé ou [part-of]). Nous pourrions être tentés d'exprimer dans le modèle la proposition que les *Satellites naturels* se composent de *Satellite naturel de Saturne* pour indiquer la relation de composition entre ces deux concepts, alors qu'il s'agit plutôt d'une relation de spécialisation, car les satellites naturels de Saturne forment un sous-ensemble de la classe de tous les satellites naturels.
- Finalement, une autre erreur commise est l'inversion de l'orientation du *LienS* (voir l'étiquette **c**). Les étiquettes **d** à **g** soulignent, quant à elles, des points de précisions qui seront traités plus loin.

Les particularités de ce modèle permettront de mesurer l'efficience d'OntoCASE à:

- repérer les erreurs de subsomption (étiquette **c**);
- générer de nouvelles connaissances (étiquette **f** et **g**);

- interpréter différemment les liens de composition [part-of] et de classification [is-a] (étiquette **b**);
- sélectionner les niveaux d'abstraction (étiquette **a**);
- désambiguïser le langage (étiquette **c**);
- construire des propriétés et des normes (étiquette **e**).

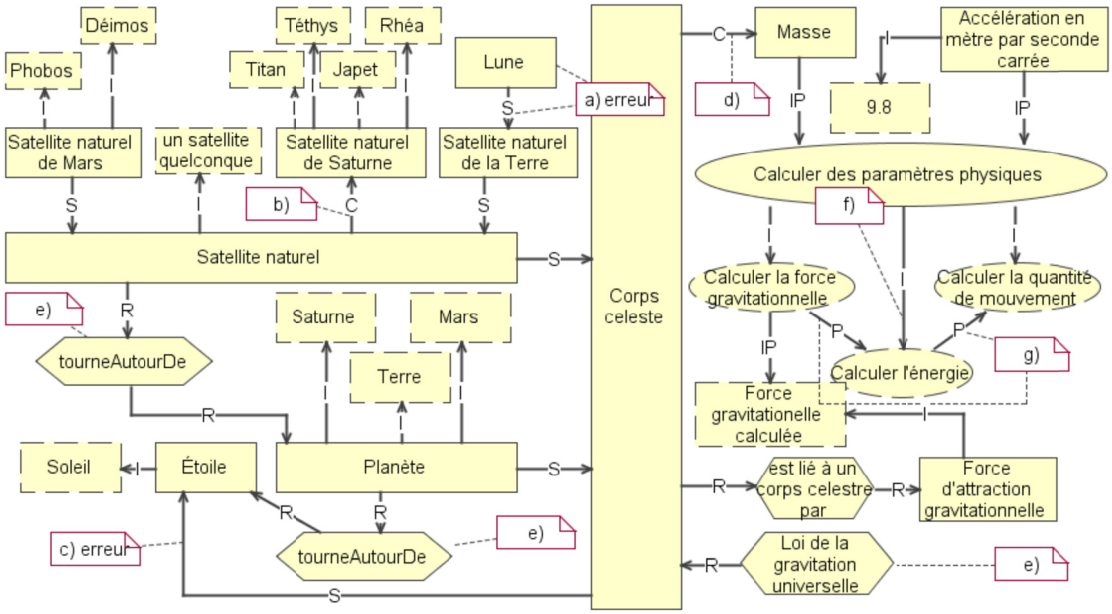


Figure 4.1: Modèle semi-formel d'expérimentation: l'*Objet céleste.*

4.3.7 Texte à modéliser en MOT et à formaliser

Le texte à modéliser de la figure 4.2 contient des énoncés faisant référence à des connaissances déclaratives, procédurales et stratégiques. Ce texte est un extrait tiré du matériel d'un cours en sciences de l'environnement offert à la Télé-université et a été utilisé dans le cadre d'autres recherches portant sur la modélisation semi-formelle des connaissances (Basque et Pudelko, 2004). Simple, précis et court, il nous a semblé que des participants convenablement formés à OntoCASE pourraient mettre à l'épreuve les différents composants procéduraux et d'assistance d'OntoCASE à partir de ce texte.

Élimination des déchets

L'élimination des déchets se fait de deux façons principales: l'incinération et l'enfouissement. L'incinération, qui est la méthode la plus onéreuse, consiste à brûler les déchets dans un four à des températures de 500 à 1000 degrés Celsius. La matière organique est alors transformée en gaz tandis que le reste des déchets devient un résidu (cendres). Cette technique permet d'éliminer entre 85 et 90 % du volume initial des déchets, mais les résidus doivent obligatoirement être éliminés dans un lieu d'enfouissement sanitaire.

Figure 4.2: Texte présenté lors de la phase 3 de l'expérimentation.

4.3.8 Bilan et commentaires

En moyenne, les expérimentations ont été d'une durée de deux à trois heures chacune. Après la séance, chacun des participants a déclaré qu'il serait apte à utiliser OntoCASE avec un minimum de soutien de l'expérimentateur. Il est à noter que les principaux points de difficultés dans l'utilisation d'OntoCASE sont associés à l'ergonomie d'*Eclipse.* Il est à supposer qu'un utilisateur déjà familier avec l'environnement *Eclipse* acquerrait plus aisément les réflexes nécessaires à l'utilisation d'OntoCASE. Par exemple, les utilisateurs surtout familiers avec le logiciel MotPlus avaient tendance à réaliser des manipulations d'éléments graphiques selon l'ergonomie d'utilisation prescrit par le logiciel MotPlus (façon de déplacer les objets, de varier leur taille) alors que dans OntoCASE ces manipulations doivent être réalisées de manière différente ce qui pouvait générer une certaine confision chez l'utilisateur.

Du point de vue de la convivialité, les participants ont particulièrement apprécié le tableau de bord à la formalisation et à la validation en tant que support au déploiement de la méthodologie. Ils ont aussi apprécié le mécanisme de rétroaction des erreurs. Suite aux remarques du premier participant, l'expérimentateur a ajusté l'étape de formation afin d'y inclure une explication détaillée de la structure de l'ontologie de référence et la structure de l'ontologie cadre. Cet ajout a été très

apprécié des participants subséquents, lesquels ont largement accru leur rapidité à acquérir de l'autonomie face à l'usage de l'assistant, notamment à l'étape de désambiguïsation du modèle semi-formel.

Tous les participants ont réussi à formaliser le modèle étalon. Cependant, aucun d'entre eux n'a pu compléter la modélisation du texte sur l'élimination des déchets. Bien que court, ce texte contient beaucoup de « pièges » de modélisation, ce qui exige une longue réflexion. Devant ce constat et pour ne pas retarder indûment la séance d'expérimentation, nous avons demandé aux participants de formaliser leur modèle incomplet afin d'utiliser OntoCASE pour produire des interprétations transitoires de leur modèle. Cet exercice fut très apprécié des utilisateurs puisqu'il faisait déjà ressortir des erreurs de modélisation, ce qui a servi de guide pour la poursuite de la modélisation. Par exemple, à l'énoncé "*L'élimination des déchets se fait de deux façons principales: l'incinération et l'enfouissement*", trois participants ont modélisé cet énoncé par l'utilisation de la composition (LienC) entre la procédure *élimination des déchets* et les procédures *incinération* et *enfouissement*. La formalisation du modèle et l'interprétation du rapport de validation sémantique ont permis de déterminer que ces procédures doivent être unies par des liens de spécialisation (*lienS*). En effet, l'incinération et l'enfouissement sont bel et bien des sortes de procédures pour éliminer des déchets et non des sous-procédures.

Un aspect important a été soulevé par l'un des participants concernant l'une des fonctionnalités de l'assistant. Pendant la modélisation, il est possible de dérouler un menu à partir d'un objet du modèle afin d'en déterminer le type pour une désambiguïsation manuelle. Pour ce participant, cette fonctionnalité est très instructive dans la démarche de réflexion sur l'utilisation de cet objet. Par exemple, le *principe* est un objet qui peut se désambiguïser en *agent* ou en *condition.* En déroulant le menu associé, le participant peut désambiguïser cet objet par la sélection de l'un au l'autre de ces types. La réflexion menant à la sélection du type génère une foule de questionnements sur le modèle lui-même et sur le contexte d'utilisation de

cet objet. Pour ce participant, cette réflexion est une occasion de pousser plus loin l'expressivité du langage pour ainsi créer de nouvelles façons d'exprimer une situation. Par exemple, l'association d'un principe à une procédure incite le participant à se questionner pour déterminer si le principe agit en tant que *contrainte* ou en tant que *condition.*

Cette étude a démontré qu'après une brève introduction, un utilisateur déjà familier avec la modélisation par objets typés arrivait en une dizaine de minutes à utiliser l'éditeur de modèles. De plus, tous les participants ont su utiliser correctement les tableaux de bord à la formalisation et à la validation. Pendant les séances, le logiciel a réalisé les tâches de manières attendues et aucun imprévu, ou défaillance logiciels n'est survenu. Pendant les séances, les commentaires des participants étaient plutôt positifs et plusieurs des commentaires concernaient les différents contextes d'utilisation potentielle du logiciel.

4.4 Généricité de la méthode de formalisation des systèmes semi-formels

Le dernier volet de la validation est le test de généricité de la formalisation des systèmes semi-formels. Il s'agit d'un test qui permet de valider le fait que la méthodologie et l'architecture informatique de l'assistant permettent la formalisation en ontologies de modèles semi-formels autres que des modèles exprimés en langage MOT. Nous comptons démontrer la généricité d'OntoCASE par l'intégration du langage *MindMap* (Buzan et Buzan, 1994) à OntoCASE.

4.4.1 Métamodèle de *MindMap*

Tel que déjà mentionné au chapitre 1, le langage *MindMap* est un langage graphique semi-formel qui combine des lignes, des images et du texte afin de représenter des idées et des concepts en vue d'élaborer un discours par l'éclatement d'un *thème* en *sous-thèmes*. Un métamodèle possible de ce langage est présenté à la figure 4.3. Pour le déroulement de la validation de la généricité, et pour ne pas alourdir initilement

l’implantation, seuls les éléments en gris foncé du métamodèle seront traités. À partir de cette contrainte, la définition d'un modèle (*Map*) se compose d'un thème (*Topic*) qui peut se relier à d'autres thèmes par une association de type sous-thème (*subtopic*).

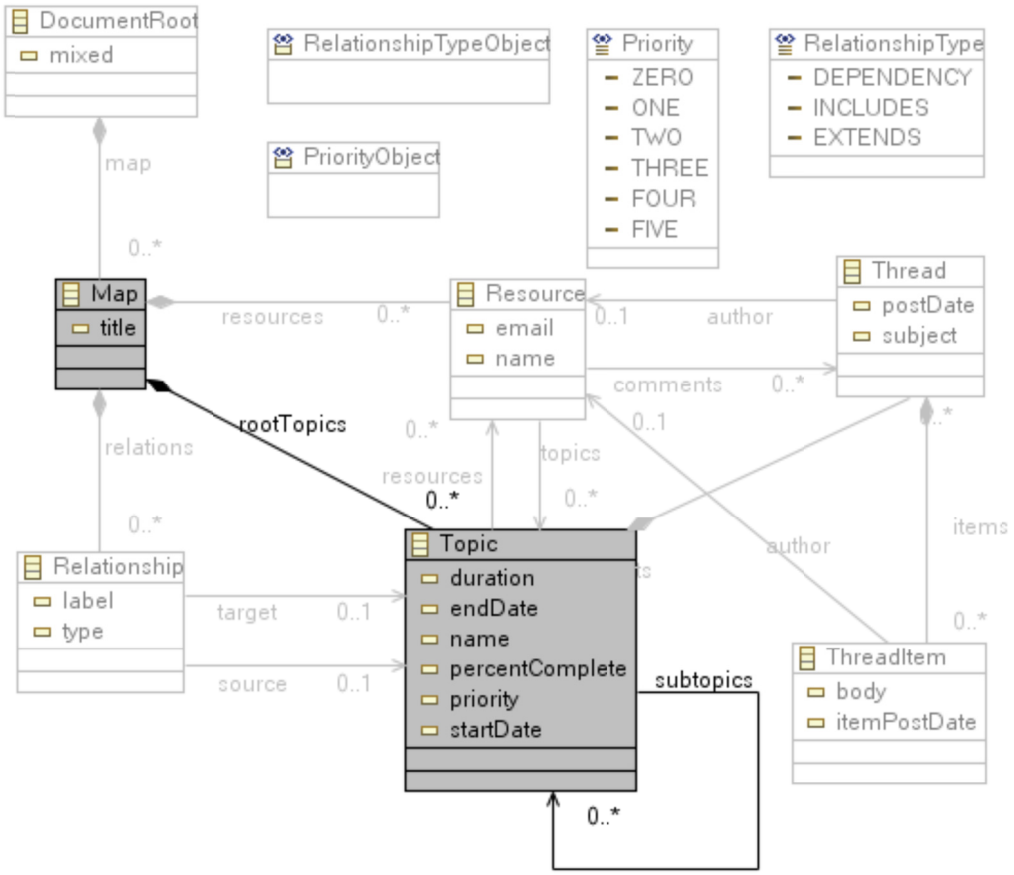


Figure 4.3: Métamodèle *ecore* du langage *MindMap. (Tirée de Reitsma et al., 2008).*

4.4.2 Méthode d'intégration du langage *MindMap* à OntoCASE

Le modèle procédural de la figure 4.4 représente la méthode par laquelle un langage semi-formel est intégré à OntoCASE.

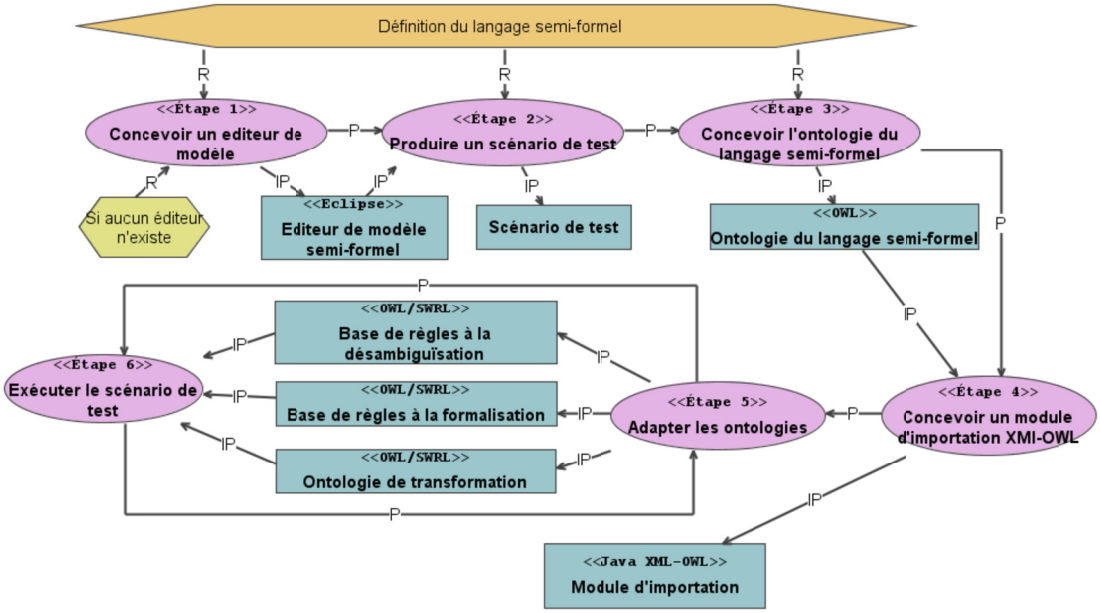


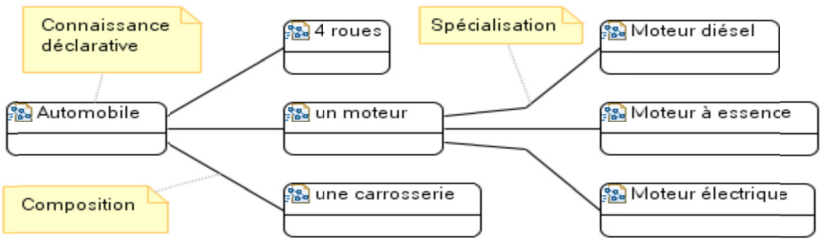
Figure 4.4: Méthode d'intégration d'un nouveau langage source à OntoCASE.

L'étape 1, la conception d'un éditeur de modèle, est réalisée s'il n'existe aucun éditeur du modèle semi-formel produisant des documents interopérables. À partir de la définition du langage, l'étape 2 sert à produire un scénario de test qui guidera le futur développement. Le scénario de test peut être utilisé ou modifié à l'une ou l'autre des étapes à venir. L'étape 3, concevoir l'ontologie du langage semi-formel, est l'étape de représentation de la définition du langage semi-formel en ontologie. La définition du langage semi-formel représentée dans l'ontologie du langage semi-formel sert d'intrant au processus de l'étape 4, lequel consiste à concevoir un module d'importation XML-OWL servant à la production du code *Java* dédié à l'import/export du modèle semi-formel du formalisme XML au formalisme OWL. L'étape 5 est consacrée à la conception des règles de désambiguïsation, de

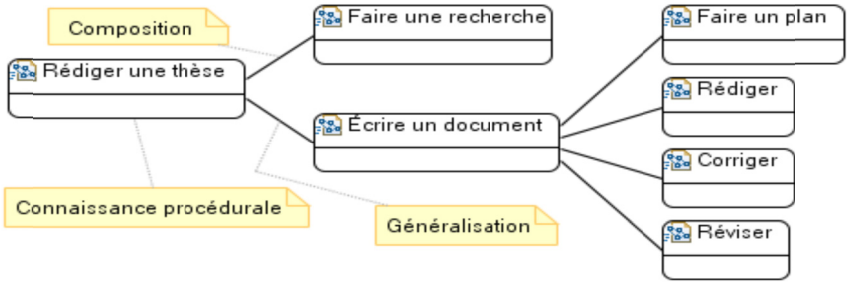
formalisation ainsi que de l'ontologie de transformation selon les définitions du langage semi-formel. Finalement, l'étape 6, la validation du processus d'intégration, est assurée par l'exécution du scénario de tests.

4.4.3 Définition du langage semi-formel et production du scénario de test

Nous avons vu que le vocabulaire du langage *MindMap* utilise le *thème* et l'association de *sous-thèmes* pour représenter des concepts et les relations qui les unissent. Sa grammaire impose une lecture du modèle de la gauche vers la droite avec un éclatement du thème principal en sous-thèmes. Le langage *MindMap* ne possède pas de symbole particulier faisant référence à des connaissances de type déclaratif ou procédural (voir la figure 4.5). De même, aucune symbolique particulière ne permet la distinction entre une association d'hyponymie (spécialisation), d'hyperonymie (généralisation) ou d'holonymie (composition). En ce sens, le langage *MindMap* comporte un plus grand niveau d'ambiguïté que le langage MOT.



a) Une automobile se compose de 4 roues, d'une carrosserie et d'un moteur. Un moteur peut être diesel, à essence ou électrique.



b) Pour écrire une thèse, il faut faire une recherche et aussi faire des activités comprises dans la procédure d'écriture d'un document, soit faire un plan, rédiger, corriger et réviser.

Figure 4.5: Modèles *MindMap* sur le thème de l'automobile et le thème de la rédaction d'une thèse.

Dans le contexte de la présente démonstration, les exemples de la figure 4.5 servent de scénarios de tests pour l'intégration du langage *MindMap* dans OntoCASE. Les composants d'OntoCASE devront être en mesure d'assister l'utilisateur dans la désambiguïsation des connaissances procédurales et déclaratives associées à un thème ou un sous-thème, ainsi qu'à la désambiguïsation des associations d'hyponymie, d'hyperonymie et d'holonymie.

4.4.4 Concevoir l'ontologie du langage semi-formel

Nous avons vu dans la définition du langage que le métamodèle comporte un *Thème* pouvant se développer en *Sous-thèmes*. En termes de métamodèle d'entité-relation, la *structure* d'un modèle comporte le *thème* (l'entité) et le sous-thème lié par l'*association* (la relation) (voir le tableau 4.6 lignes 1, 4 et 7).

Tableau 4.6
Ontologie du langage *MindMap* dans le formalisme OWL-N3

```
01.  :MM_Structure
02.        rdf:type owl:Class ;
03.        rdfs:subClassOf owl:Thing .
04.  :MM_Association
05.        rdf:type owl:Class ;
06.        rdfs:subClassOf :MM_Structure.
07.  :MM_Theme
08.        rdf:type owl:Class ;
09.        rdfs:subClassOf :MM_Structure.
10.  :MM_estLaDestination
11.        rdf:type owl:ObjectProperty ;
12.        rdfs:domain :MM_Theme ;
13.        rdfs:range :MM_Association ;
14.        owl:inverseOf
     :MM_themeDestination.
15.  :MM_estLaSource
16.        rdf:type owl:ObjectProperty ;
17.        rdfs:domain :MM_Theme ;
18.        rdfs:range :MM_Association ;
19.        owl:inverseOf :MM_themeSource.
20.  :MM_estNonDetermine
21.        rdf:type owl:FunctionalProperty
     , owl:DatatypeProperty ;
22.        rdfs:domain :MM_Structure ;
23.        rdfs:range xsd:boolean .
```

```
24.  :MM_etiquette
25.        rdf:type owl:FunctionalProperty
     , owl:DatatypeProperty ;
26.        rdfs:domain :MM_Structure ;
27.        rdfs:range xsd:string .
28.  :MM_identifiant
29.        rdf:type owl:FunctionalProperty
     , owl:DatatypeProperty ;
30.        rdfs:domain :MM_Structure ;
31.        rdfs:range xsd:string .
32.  :MM_ontoRefStereotype
33.        rdf:type owl:DatatypeProperty;
34.        rdfs:domain :MM_Structure ;
35.        rdfs:range xsd:string .
36.  :MM_themeSource
37.        rdf:type owl:FunctionalProperty
     , owl:ObjectProperty ;
38.        rdfs:domain :MM_Association ;
39.        rdfs:range :MM_Theme ;
     owl:inverseOf :MM_estLaSource .
40.  :MM_themeDestination
41.        rdf:type owl:FunctionalProperty
     , owl:ObjectProperty ;
42.        rdfs:domain :MM_Association ;
43.        rdfs:range :MM_Theme ;
44.        owl:inverseOf
     :MM estLaDestination.
```

En plus de ces trois classes, l'ontologie comporte un ensemble de propriétés définissant les relations existantes entre le *thème* et l'*association.* Ainsi les propriétés `MM_estLaDestination`, `MM_estLaSource`, `MM_themeSource` `MM_themeDestination` (voir les lignes 10, 15, 36, 40) définissent les thèmes qui sont à la source et à la destination d'une association. Les propriétés `MM_etiquette` et `MM_identifiant` (voir les lignes 24 et 28) sont de type *chaîne de caractères* et permettent d'entreposer le nom et l'identifiant unique associé à chacun des thèmes et associations. Les propriétés `MM_estNonDetermine` et `MM_ontoRefStereotype` (voir les lignes 20 et 32) sont utilisées à l'étape de désambiguïsation.

4.4.5 Conception du module d'import/export

Tel qu'indiqué à la section 3.2.2, p.160, la conception du module d'importation implique l'utilisation du code *Java* de l'ontologie du langage semi-formel ainsi que le code *Java* EMF généré à partir du modèle *ecore* du langage semi-formel. Le tableau 4.7 présente un exemple de conversion XMI à OWL des modèles de la figure 4.5. La ligne 3 de **a** représente la définition du thème `automobile` associé aux sous-thèmes; `4 roues` (à la ligne 4), `un moteur` (à la ligne 5) et `une carrosserie` (à la ligne 6). Converti en OWL (voir le tableau 4.7**b**), le triplet Automobile/subtopics/4_roues est traduit par la création des deux classes `Automobile_1` (voir la ligne 1) et `_4-roues-2` (voir la ligne 7) qui sont de type `MM_Theme`, puis en une classe `Automobile-1_association__4-roues-2` (voir la ligne 13) de type `MM_Association`. Il en va de même pour le reste des composants du modèle.

Tableau 4.7
Représentation *XMI* (a) et *OWL* (b) du modèle *MindMap* sur le thème de l'automobile

a)

```
01.   <?xml version="1.0" encoding="UTF-8"?>
02.   <mindmap:map xmlns:mindmap="http://www.example.org/mindmap" title="un test">
03.   <rootTopics name="Automobile" subtopics="#//@map/@rootTopics.1
      #//@map/@rootTopics.2 #//@map/@rootTopics.6"/>
04.   <rootTopics name="4 roues"/>
05.   <rootTopics name="un moteur" subtopics="#//@map/@rootTopics.4
      #//@map/@rootTopics.3 #//@map/@rootTopics.5"/>
06.   <rootTopics name="Moteur diÃ©sel"/>
07.   <rootTopics name="Moteur Ã  essence"/>
08.   <rootTopics name="Moteur Ã©lectrique"/>
09.   <rootTopics name="une carrosserie"/>
10.   ...
11.   </mindmap:map>
```

b)

```
01.   :Automobile-1
02.     rdf:type metaMM:MM_Theme ;
03.     metaMM:MM_estLaSource :Automobile-1_association_un-moteur-3 , :Automobile-
      1_association_une-carrosserie-7 , :Automobile-1_association__4-roues-2 ;
04.     metaMM:MM_estNonDetermine "true"^^xsd:boolean ;
05.     metaMM:MM_etiquette "Automobile"^^xsd:string ;
06.     metaMM:MM_identifiant "Automobile-1"^^xsd:string .
07.   :_4-roues-2
08.     rdf:type metaMM:MM_Theme ;
09.     metaMM:MM_estLaDestination :Automobile-1_association__4-roues-2 ;
10.     metaMM:MM_estNonDetermine "true"^^xsd:boolean ;
11.     metaMM:MM_etiquette "4 roues"^^xsd:string ;
12.     metaMM:MM_identifiant "_4-roues-2"^^xsd:string .
13.   :Automobile-1_association__4-roues-2
14.     rdf:type metaMM:MM_Association ;
15.     metaMM:MM_estNonDetermine "true"^^xsd:boolean ;
16.     metaMM:MM_etiquette "Automobile est associÃ©(e) Ã  4 roues"^^xsd:string ;
17.     metaMM:MM_identifiant "Automobile-1_association__4-roues-2"^^xsd:string ;
18.     metaMM:MM_themeDestination :_4-roues-2 ;
19.     metaMM:MM_themeSource :Automobile-1 .
```

4.4.6 Adapter les ontologies

L'ajout d'un nouveau langage semi-formel à traiter dans OntoCASE implique une adaptation de l'ontologie de transformation et de ses sous-ontologies. Présentée à la figure 4.6, l'intégration du nouveau langage impose des changements structurels de classes (voir en **a** les classes qui sont surlignées), l'ajout de nouveaux stéréotypes de désambiguïsation (voir en **b**) dans l'ontologie de traitement des ambiguïtés, ainsi que l'ajout de nouvelles propriétés (voir en **c**).

Pour terminer l'intégration du formalisme de *MindMap*, il est nécessaire d'ajuster les règles de conversion à la sémantique du langage. Dans le formalisme de *MindMap*, une association entre deux thèmes peut représenter une composition (holonyme), une

spécialisation (hyponyme) ou encore une généralisation (hyperonyme). Les relations d'holonymie et d'hyponymie ont déjà été définies à l'étape de définition du langage MOT. Il reste donc à intégrer les règles nécessaires à la conversion de l'aspect sémantique de l'hyperonyme. Le tableau 4.8 présente la règle SWRL de conversion, entre deux connaissances déclaratives, d'un hyperonyme en propriété `owl:subClassOf` d'OWL. Une règle du même genre a été conçue pour le traitement d'une connaissance procédurale.

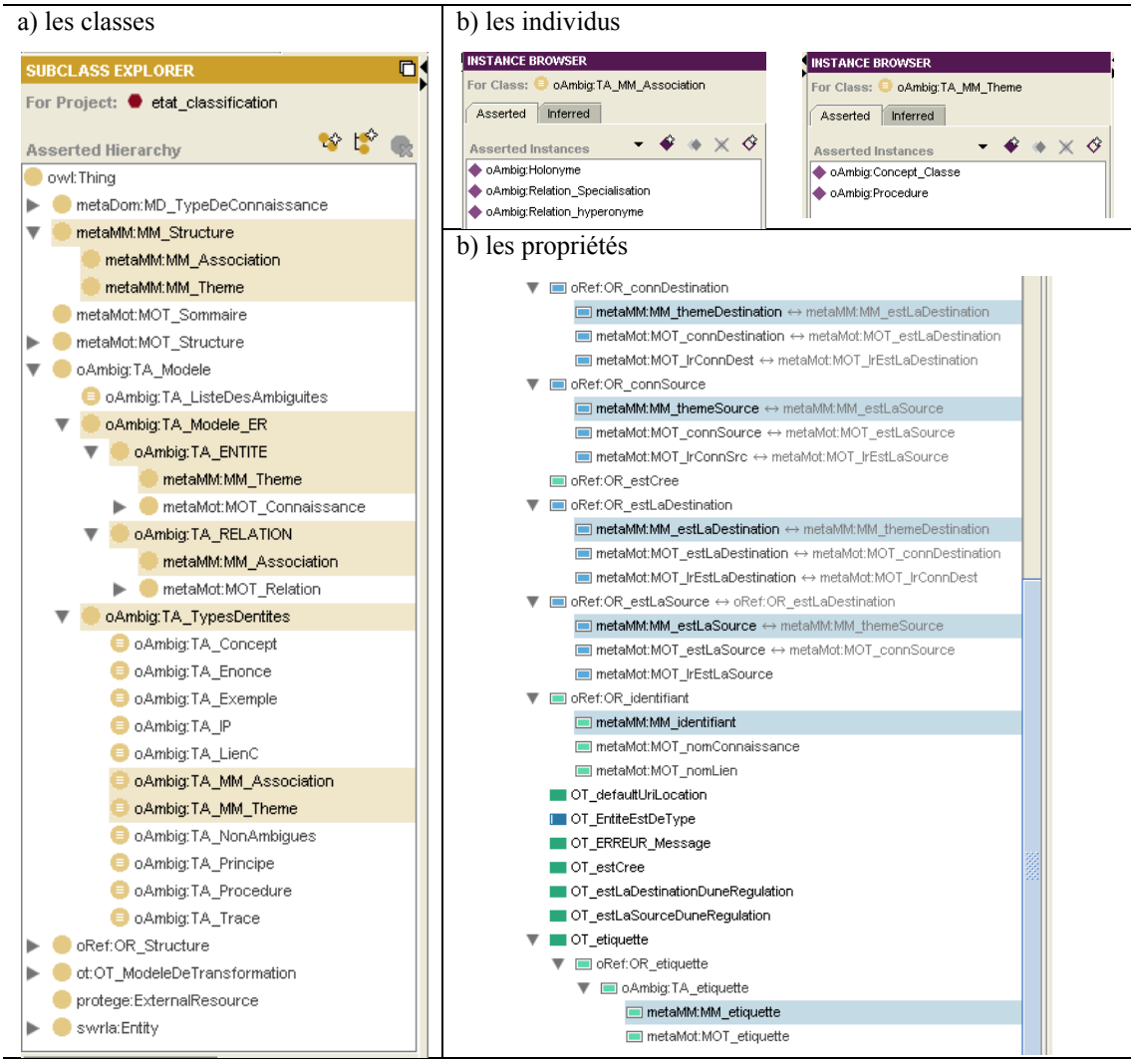


Figure 4.6: Structure de l'ontologie de transformation après l'intégration des éléments structurels nécessaires à la transformation d'un modèle *MindMap*.

Tableau 4.8
Règles de conversion pour un hyperonyme

```
01.    Rule-02-d_Creation_subClassOf_INV_Declaratif
02.    oRef:OR_Relation_Hyperonyme(?lg)  ^
03.    oRef:OR_connSource(?lg, ?src)  ^
04.    oRef:OR_connDestination(?lg, ?dest)  ^
05.    oRef:OR_Entite_Concept_Classe(?src)  ^
06.    oRef:OR_Entite_Concept_Classe(?dest)  ^
07.    oRef:OR_identifiant(?src, ?nomSrc)  ^
08.    oRef:OR_identifiant(?dest, ?nomDest) ->
09.    swrlbi:invoker("OWLEstUneSousClasseDeCmd", ?nomDest, ?nomSrc)  ^
10.    swrlbi:invoker("OWLSupprimerSuperClasseDeCmd", "owl:Thing", ?nomDest)  ^
11.      swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",
       "metaDom:MD_Declarative_Concept", ?nomDest)
```

4.4.7 Exécuter le scénario de test

La validation est la dernière étape de l'intégration d'un nouveau formalisme. La conception d'un scénario de test permet de mesurer l'efficacité du dispositif à réaliser la tâche. Bien que normalement beaucoup plus élaboré que dans une modélisation en situation réelle, le scénario proposé ici permet de tirer plusieurs conclusions. Tout d'abord, à la lecture du tableau 4.9 et de la figure 4.7, il est constaté que la sémantique du modèle d'origine (voir la figure 4.5) est préservée. Ensuite, l'intégration d'une nouvelle sémantique (l'hyperonyme pour le présent cas) à traiter par l'ontologie de transformation s'implante assez facilement par la conception d'une nouvelle règle SWRL et par quelques ajouts aux ontologies associées à l'ontologie de transformation.

Tableau 4.9
Rapport de validation sémantique

a) validation sémantique du modèle descriptif

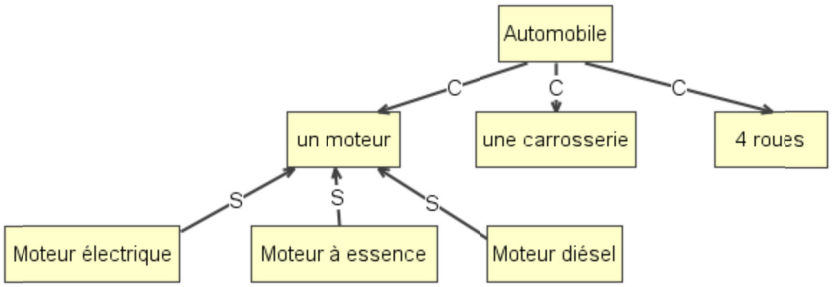
```
----------Début du processus de validation sémantique-------------
4 roues est une sorte de (metaDom:MD_Declarative_Concept)
Automobile est une sorte de (metaDom:MD_Declarative_Concept)
Automobile se compose de 4 roues
Automobile se compose de un moteur
Automobile se compose de une carrosserie
Moteur diésel est une sorte de un moteur
Moteur à essence est une sorte de un moteur
Moteur électrique est une sorte de un moteur
un moteur est une sorte de (metaDom:MD_Declarative_Concept)
une carrosserie est une sorte de (metaDom:MD_Declarative_Concept)
```

b) validation sémantique du modèle procédural

```
----------Début du processus de validation sémantique-------------
Corriger est une sorte de (metaDom:MD_Procedurale_Procedure)
Faire un plan est une sorte de (metaDom:MD_Procedurale_Procedure)
Faire une recherche est une sorte de (metaDom:MD_Procedurale_Procedure)
Faire une recherche est une sorte de Rédiger une thèse
Rédiger est une sorte de (metaDom:MD_Procedurale_Procedure)
Rédiger une thèse est une sorte de (metaDom:MD_Procedurale_Procedure)
Rédiger une thèse est une sorte de Écrire un document
Réviser est une sorte de (metaDom:MD_Procedurale_Procedure)
Écrire un document est une sorte de (metaDom:MD_Procedurale_Procedure)
Écrire un document se compose de Corriger
Écrire un document se compose de Faire un plan
Écrire un document se compose de Rédiger
Écrire un document se compose de Réviser
```

Dans le présent contexte, le module de validation syntaxique ne peut pas être intégralement utilisé, car il a été conçu pour la validation de modèle MOT. Cependant, le module de validation peut être utilisé pour restaurer le modèle source dans le formalisme de MOT (voir la figure 4.7) offrant ainsi une interprétation en langage MOT du modèle d'origine.

a) restauration du modèle descriptif



b) restauration du modèle procédural

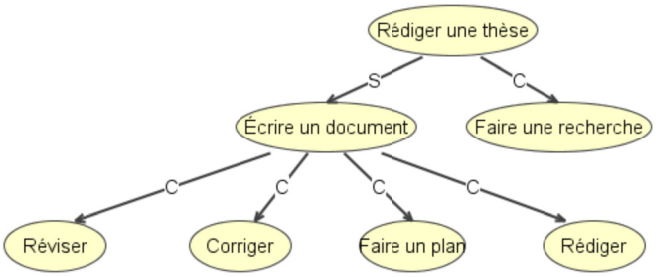


Figure 4.7: Restauration dans le formalisme de MOT du thème de l'automobile (a) et de la rédaction d'une thèse (b).

4.5 Conclusion

Dans ce chapitre, nous avons voulu démontrer que l'hypothèse de cette thèse a été vérifiée en soumettant OntoCASE à une validation selon :

- La généralité des types de connaissances à formaliser, afin de vérifier qu'OntoCASE permet la formalisation de connaissances procédurales, stratégiques, déclaratives et factuelles. Pour effectuer la validation, des métriques d'efficacité ont été matérialisés par la production de scénarios de tests unitaires, de tests d'intégrations, des tests systémiques et de tests de cohérence ;
- L'ergonomie, afin de valider que la méthodologie soit exploitable par des utilisateurs autres que le concepteur de la méthodologie et que l'assistant informatique offre toutes les fonctionnalités nécessaires à la réalisation d'une formalisation, et ce, de manière conviviale. La validation de l'ergonomie a été

réalisée par la mesure de l'efficience et de la satisfaction lors d'une expérimentation en laboratoire ;

- La généricité des langages semi-formels utilisés pour la formalisation, afin de valider que la méthodologie permet le passage du degré semi-formel au degré formel d'une représentation sans altérer la sémantique qui lui est associée. Cette validation a été établie en intégrant un autre langage semi-formel, le langage *MindMap,* à OntoCASE.

Chapitre 5

CONCLUSION ET DISCUSSION

Le but de cette thèse était de produire une méthodologie d'ingénierie ontologique se composant de méthodes et d'outils informatiques intelligents afin d'assister l'ingénieur de la connaissance et l'expert de contenu à concevoir une ontologie à partir d'une conceptualisation semi-formelle d'un domaine contenant des représentations de connaissances déclaratives, procédurales, stratégiques et factuelles.

Dans ce chapitre, nous rappelons d'abord les principales réalisations issues de notre travail afin d'atteindre ce but. Nous poursuivons par la présentation des contributions scientifiques et des limites de cette recherche. Nous concluons ce chapitre par une discussion sur les perspectives de recherche entrouvertes par cette thèse.

5.1 Réalisations

Les principaux concepts nécessaires à la réalisation de la thèse ont été présentés au premier chapitre. Ceux-ci concernent les fondements théoriques en représentation des connaissances, en modélisation des connaissances, en classification des ontologies et en assistance logicielle. Nous avons notamment présenté quelques langages semi-formels et formels utilisés en représentation des connaissances et avons discuté de concepts liés à la métamodélisation et à la transformation de modèles, de méthodes d'ingénierie ontologique ainsi que de principes de conception d'un système d'assistance informatique.

La méthodologie d'OntoCASE développée dans le cadre de la thèse inclut des constituants méthodologiques, représentationnels et computationnels. Au cœur du

volet représentationnel de la transformation des modèles, réside l'ontologie de transformation, composée elle-même de deux ontologies, soit l'ontologie de référence responsable de la conversion des modèles semi-formels en ontologie et l'ontologie cadre qui élargit l'expressivité de l'ontologie du domaine pour y inclure la représentation de connaissances déclaratives, procédurales et stratégiques. D'autres ontologies de même que des bases de règles composent l'ontologie de transformation, telles que l'ontologie du langage semi-formel, la base de règles de transformation, la base de règles à la désambiguïsation et la base de règles à l'identification des erreurs. Quant au volet méthodologique d'OntoCASE, il comprend quatre méthodes : une *méthode d'intégration d'un nouveau langage semi-formel à la méthodologie*, une *méthode de modélisation* d'un modèle semi-formel, une *méthode de transformation* du modèle semi-formel en ontologie et une *méthode de validation* syntaxique et sémantique de l'ontologie cible. Chacune de ces méthodes, sauf la première, est assistée par une application informatique et représentée par l'ontologie de transformation.

La démarche de *construction d'OntoCASE* que nous avons adoptée a été divisée en trois phases. La phase 1, *la mise en place des composants d'OntoCASE*, avait pour objectif l'élaboration des outils, processus et ontologies nécessaires au fonctionnement d'OntoCASE. La phase 2, *l'agrégation des composants*, a consisté à construire l'architecture du système de support à la méthodologie d'OntoCASE. Finalement, la phase 3, *la consolidation,* a été marquée par la construction de bancs d'essais permettant l'exécution de scénarios de tests unitaires, d'intégration, de systèmes et d'incohérences nécessaires à la vérification du bon fonctionnement des composants informatiques. Ces tests ont aussi été utilisés pour la validation de l'efficacité, de l'efficience et de la satisfaction d'OntoCASE.

Les résultats concernant la méthodologie d'OntoCASE ont été présentés au chapitre 2, alors que ceux relatifs à la validation d'OntoCASE l'ont été au chapitre 4. Ces résultats ont permis de confirmer l'hypothèse de cette thèse à savoir qu'*à partir d'un*

modèle de connaissances exprimé dans un langage semi-formel représentant des connaissances déclaratives, procédurales, stratégiques, au niveau abstrait et concret, il est possible de formaliser ce modèle en ontologie de manière automatique ou semi-automatique d'une manière syntaxiquement et sémantiquement valide. La confirmation de cette hypothèse a été rendue possible grâce à l'implantation de scénarios de tests visant à évaluer l'efficacité d'OntoCASE et à une expérimentation en laboratoire ayant permis d'évaluer l'efficience et la satisfaction auprès d'usagers représentant le public cible. L'intégration d'un autre langage semi-formel (*MindMap*) à OntoCASE tend à démontrer que la méthodologie comporte une certaine généricité.

Trois outils documentaires ont été produits par ces travaux. Un *guide du langage de modélisation par objets typés* (voir l'appendice A) apporte un support à l'activité de modélisation semi-formelle MOT pour l'ingénieur de la connaissance et pour l'expert de domaine. Le deuxième document, le *catalogue de la sémantique formelle* (voir l'appendice B), sert de guide de désambigüisation et de formalisation d'un modèle MOT en ontologie OWL. Il permet donc à l'ingénieur de documenter la façon dont cette étape permet de codifier le modèle semi-formel en ontologie et peut servir, le cas échéant, à supporter une codification manuelle du modèle semi-formel advenant une indisponibilité de l'assistant informatique. Finalement, les *éléments pour guider la modélisation semi-formelle à l'aide de concepts ontologiques* (voir l'appendice E) est un outil mis à la disposition de l'ingénieur et de l'expert afin de les guider à l'étape de la conception du modèle semi-formel en vue d'en faire une ontologie représentative et opérationnelle du domaine conceptualisé.

En résumé, du point de vue des sciences cognitives, les réalisations de cette thèse sont les suivantes :

- une ontologie représentationnelle et opérationnelle du domaine de la représentation des connaissances;

- une méthode de transformation de modèles de connaissances semi-formels en ontologies;
- une méthode de représentation de connaissances tant déclaratives, procédurales que stratégiques en ontologie selon le niveau abstrait et le niveau factuel;
- une méthode qui, lors du processus de formalisation, gère les ambiguïtés intrinsèquement contenues dans la sémantique du modèle semi-formel;
- un ensemble de documents qui servent de guide à la conception d'un modèle semi-formel dans la perspective de sa transformation subséquente en ontologie.

Du point de vue informatique, les réalisations de cette thèse sont les suivantes:

- un outil intelligent et intégré dans l'environnement *Eclipse* apte à assister un ingénieur ontologique et un ou des experts du domaine dans la conception d'une ontologie à partir d'un modèle semi-formel;
- un mécanisme de construction automatique d'ontologies à partir d'une base de connaissances de type OWL et SWRL;
- un éditeur de modèles MOT (eLi) dans l'environnement *Eclipse* selon les principes de l'*architecture conduite par les modèles* et d'EMF/GMF;
- l'intégration dans *Eclipse* des bibliothèques de codes sources de *Protégé* pour la manipulation d'ontologies, de règles SWRL et d'interfaces aux engins d'inférences *Pellet* et *Jess*;
- des outils d'importation/exportation de modèles de l'espace de modélisation MOF dans l'espace de modélisation OWL;
- la substitution avec succès du MOF par OWL/SWRL en tant que méta-métamodèle dans l'utilisation de l'architecture conduite par les modèles pour la transformation de modèles semi-formels;

- un outil intelligent qui automatise la conversion et la validation du modèle semi-formel en ontologie et qui semi-automatise le processus de désambiguïsation des connaissances exprimées de manière semi-formelle;
- un outil intelligent de production de texte informel à partir d'une conceptualisation exprimée de manière semi-formelle.

5.2 Contributions

Nous allons maintenant présenter les contributions de cette thèse aux domaines suivants: la représentation des connaissances, l'ingénierie ontologique et la gestion des connaissances.

5.2.1 Apports en représentation des connaissances

Notre recherche a permis de mettre au point une ontologie de transformation qui, sur le plan représentationnel, présente une conceptualisation possible du domaine de la représentation de la connaissance et qui, sur le plan opérationnel, permet la réalisation d'inférences automatiques dédiées à la formalisation en ontologie de divers types de connaissances. La structure de l'ontologie de transformation, quoique développée de manière empirique, révèle par son opérationnalité confirmée (voir les mesures de l'efficacité d'OntoCASE à la section 4.2 et à l'appendice D), que les éléments qui la composent sont une représentation possible du domaine de la représentation des connaissances. La multitude et la variété de scénarios de tests réalisés ont permis de démontrer que l'ontologie de transformation est opérationnellement valide.

Basée sur un métamodèle de type *entité-relation*, l'ontologie de référence, qui est une composante de l'ontologie de transformation, a une structure qui représente la réalité selon les niveaux objectif (concret) et subjectif (abstrait), chacun catégorisé selon les trois types de connaissances: déclaratif, procédural et stratégique. Il a été démontré par l'intégration d'un formalisme de type *MindMap* que la structure interne de

l'ontologie de référence résiste bien à l'incorporation d'un nouveau langage semi-formel à la méthodologie.

En outre, notre recherche a permis de mettre au point une ontologie de transformation permettant d'opérationnaliser l'activité de désambiguïsation, de classification et de formalisation en ontologie d'un modèle exprimé en langage semi-formel. La structure interne de l'ontologie de référence et de l'ontologie cible, composants de l'ontologie de transformation, s'inspire des travaux réalisés pour la conception du langage MOT dont l'un des objectifs est d'intégrer en un seul langage la possibilité de modéliser divers types de modèles, tels que des modèles factuels, conceptuels, procéduraux, prescriptifs ainsi que des modèles de type processus et méthodes. Ainsi, la classification du vocabulaire à la base de la modélisation de modèles UML (de type diagrammes de classe, de cas d'utilisation, d'état) ou encore, la classification du vocabulaire du langage BPMN (voir la section 3.1.4.3), ainsi que l'implantation du langage *MindMap* dans la méthodologie OntoCASE (voir la section 4.4), constituent une validation supplémentaire, et ce de manière formelle, que le langage MOT possède réellement les qualités d'intégration de plusieurs types de modèles dont la représentation est assurée par un langage unifié. Cette conclusion est un apport important à la représentation des connaissances puisqu'elle garantit que MOT tient une place privilégiée dans l'arsenal des outils disponibles pour la représentation unifiée de connaissances de type hétérogène.

En ce qui a trait à l'activité de modélisation des connaissances, OntoCASE est un outil d'assistance à la modélisation et à la réflexion sur le domaine modélisé. Il permet de réaliser une validation de nature sémantique et syntaxique du modèle semi-formel et facilite ainsi la détection d'ambigüités ou d'erreurs de modélisation. La rétro-génération du modèle semi-formel et la génération du contenu du modèle sous forme textuelle à partir de l'ontologie sont des caractéristiques de rétroaction qui sont innovantes en termes d'instrumentation de l'activité de modélisation des connaissances. Finalement, la méthodologie permet à un concepteur de modèles

semi-formels, qui serait peu familier avec OWL et les représentations de type ontologiques, de contribuer activement à la construction d'ontologies.

5.2.2 Apports en ingénierie ontologique et aux applications du Web sémantique

Du point de vue de l'ingénierie ontologique, ce travail de thèse aura permis de produire une méthodologie permettant d'exprimer, dans un langage déclaratif et formel comme OWL, des connaissances déclaratives, procédurales, stratégiques et factuelles issues d'une conceptualisation exprimée en langage semi-formel. Rappelons qu'un *gain de qualité* provient de la méthode d'élicitation privilégiée dans OntoCASE, qui utilise une représentation semi-formelle de la connaissance, et qui, comme il a été démontré dans d'autres recherches, stimule la créativité et libère la charge cognitive pour l'expression de l'expertise du domaine plutôt que de l'occuper à formaliser la connaissance. De plus, un *gain de productivité* nous semble acquis par l'automatisation de la formalisation et du mécanisme de validation d'OntoCASE. Assurément, l'automatisation simplifie le processus de formalisation qui, exécuté en quelques secondes ou minutes, facilite le travail de l'ingénieur de la connaissance. Cette qualité est importante pour un ingénieur qui aurait à formaliser plusieurs dizaines de modèles semi-formels. Comme il a été mentionné par un des participants à l'expérimentation *« Pour avoir des ontologies de domaines, on doit soit les faire nous-mêmes, soit les transformer de modèles semi-formels déjà existants. Et c'est là qu'OntoCASE se positionne. »*

En outre, basé sur une architecture conduite par les modèles, OntoCASE offre une méthode pour intégrer des formalismes autres que celui de MOT à la méthodologie. Bien sûr, l'intégration d'un nouveau formalisme implique toutefois une connaissance approfondie en ingénierie ontologique, notamment dans la conception de systèmes à base de règles SWRL ainsi que dans la programmation de l'environnement de développement *Eclipse* demandant l'intervention d'un spécialiste en développement informatique.

La manière dont est exécutée la création automatique d'ontologies est aussi une caractéristique innovante d'OntoCASE. L'architecture de commandes de type *builtIn* permet l'utilisation de commandes SWRL aptes à créer des ontologies, ce qui d'après notre revue de littérature, n'était pas disponible avant OntoCASE. Ce catalogue de commandes interopérables peut être réutilisé et complété dans un contexte autre que celui d'OntoCASE. En effet, l'interopérabilité de SWRL et du langage *Java* qui sert à implanter les commandes *builtIn* font de ce catalogue un outil pouvant être réutilisé dans n'importe quel environnement utilisant le langage SWRL.

Du point des vues des applications pour le Web sémantique, notre travail nous semble également apporter une contribution significative. Par exemple, pour l'environnement *TeleLearning Operating System* TELOS (www.lornet.ca), qui est un système d'exploitation à base d'ontologies, OntoCASE pourrait permettre de produire des ontologies dans le cadre de la conception de cours, notamment pour faire le référencement sémantique des ressources d'enseignement et d'apprentissage (REA) par une modélisation semi-formelle. Il pourrait aussi permettre la récupération de modèles semi-formels déjà existants afin d'assurer une formalisation et une pérennisation des connaissances déjà élicitées

5.2.3 Apports en gestion des connaissances

En gestion des connaissances, la semi-automatisation du processus de transformation de modèles semi-formels de connaissances et le mécanisme de désambiguïsation intégrés dans OntoCASE permettent à l'ingénieur ontologique d'offrir aux experts de domaine un outil d'élicitation des connaissances qui possède une expressivité souple, intuitive et adaptée à une réalité de terrain parfois complexe, telle que peut l'être un contexte d'entreprise par exemple. De plus, cette méthodologie permet le traitement de modèles semi-formels qui seraient déjà existants dans une organisation, permettant ainsi une incorporation aux systèmes de gestion de connaissances déjà en place.

Le mécanisme de validation d'OntoCASE apporte également une contribution significative au domaine de la gestion des connaissances. Il permet de pré-valider de manière opérationnelle l'ontologie conçue. Dans le cadre d'une gestion de projets de systèmes à base de connaissances, par exemple, cette pré-validation permettrait ainsi d'entamer de manière plus précoce l'étape d'élicitation de la connaissance et de repousser à plus tard l'étape d'implantation informatique de l'ontologie de domaine, sachant que les ontologies produites sont déjà pré-validées de manière opérationnelle, minimisant ainsi les risques liés à l'opérationnalisation des ontologies. Cela permet une distribution plus équilibrée dans le temps des ressources humaines et matérielles, ce qui, en termes de gestion de projets, est un réel gain.

En élicitation des connaissances, le mécanisme de validation peut s'avérer aussi très utile comme assistant à la validation du contenu du domaine auprès des experts, notamment par le réaménagement par l'ingénieur du modèle semi-formel sur la base des résultats du processus de validation, ce qui lui permet de mieux cibler les questions à poser aux experts en entrevue.

5.3 Limites de la recherche

Nous discutons de trois limites de la recherche : des limites en représentation des connaissances, des limites informatiques et des limites liées au volet méthodologique d'OntoCASE.

5.3.1 Limites en représentation des connaissances

Il est à anticiper que la structure interne de l'ontologie de référence subirait des changements plus importants advenant l'intégration d'un langage semi-formel plus complexe, par exemple le *BPMN*, ou encore d'un langage qui serait construit selon un métamodèle autre que l'*entité-relation* comme pourraient l'être des systèmes de représentation de connaissance à base de règles, de schémas ou de scripts. De même, un langage qui représenterait des connaissances en utilisant des logiques d'ordre

supérieur, modales, probabilistes ou floues imposerait de sortir du cadre de la logique de descriptions et donc de la cible OWL-DL développée dans cette thèse.

L'opérationnalité des ontologies constituantes de l'ontologie de transformation a été démontrée. Cependant, la terminologie employée pour l'identification des classes et des propriétés de l'ontologie de référence et de l'ontologie cadre gagnerait à être comparée à d'autres ontologies de haut niveau telles que SUMO, CYC ou l'ontologie KR de Sowa, ce qui aurait dépassé le cadre de nos travaux.

Certaines règles de « bonnes pratiques » de la DL ne sont pas respectées dans la construction de l'ontologie cible. Pour l'harmoniser aux bonnes pratiques, il faudrait utiliser `rdf:typeOf` au lieu de `rdf:subClassOf` pour relier les classes de l'ontologie cible aux classes l'ontologie cadre et utiliser, pour des cas précis, l'`owl:Restriction` au lieu de `rdfs:subPropertyOf` pour définir le domaine et l'image des sous-propriétés.

5.3.2 Limites informatiques du prototype

Certaines limites d'OntoCASE ont été observées. Bien qu'il ait été validé par une expérimentation en laboratoire, le logiciel n'a pas subi l'épreuve d'un déploiement à large échelle. Pour l'heure, nous pouvons le considérer comme un prototype stable et opérationnel. Même s'il a été utilisé et amélioré durant toute la période de la recherche, certains aspects ergonomiques, notamment pour l'édition des modèles, sont à améliorer. L'ergonomie de l'édition des modèles, si elle était améliorée, offrirait une réelle valeur ajoutée à eLi, notamment en ce qui concerne la manipulation des fichiers modèles, l'accès au contenu du modèle afin qu'il soit réutilisé dans de multiples diagrammes, l'édition des propriétés associées à un objet du modèle, la manipulation des objets dans les sous-modèles et l'accès au référencement de ressources externes en lien avec un objet du modèle.

Concernant le volet computationnel, une certaine lenteur de l'application à exécuter les règles de désambiguïsation et de conversion a été observée qui pourrait sans nul doute être améliorée. De plus, des fuites de mémoire (*memory leak*) dans l'utilisation des bibliothèques de code source de *Protégé* nécessitent que l'application soit occasionnellement redémarrée. L'ajout de fonctionnalités permettant de corriger ces problèmes nécessiterait une connaissance approfondie de l'environnement de développement *Eclipse* et des produits *PDE, EMF* et *GMF,* dont la courbe d'apprentissage est assez lente.

5.3.3 Limites du volet méthodologique d'OntoCASE

Le volet méthodologique est difficilement exécutable sans l'utilisation de l'assistant informatique. Bien que conçues dans cette optique et soutenues par la rédaction des divers guides de la méthodologie, les règles et les situations de transformation et de désambiguïsation qui doivent être éclaircies par l'assistant logiciel sont nombreuses et nécessitent un traitement informatique important pour minimiser les erreurs de transformation. Un tel traitement manuel est tout à fait réalisable, mais il nécessite un temps et des efforts non négligeables.

5.4 Perspectives

Dans cette section, nous identifions les perpectives qui nous semblent les plus prometteuses pour OntoCASE.

5.4.1 Perspectives de développement et d'utilisation

Dans le cadre de projets industriels associés au Web sémantique, OntoCASE pourrait être utilisé pour formaliser des processus d'affaires ou encore être intégré à des projets de référencement sémantique de documents. Une évaluation des fonctionnalités et des aspects ergonomiques d'OntoCASE serait à réaliser dans cette perspective.

5.4.2 Perspectives en recherche sur la modélisation

Le mécanisme de validation par l'interprétation en langage naturel de l'ontologie cible est un mécanisme qui pourrait facilement être exploité pour la modélisation semi-formelle autre que celle de MOT comme par exemple le langage UML, et ce, non seulement pour le diagramme de classes, mais aussi pour le diagramme d'états, de séquences ou encore, de cas d'utilisation. Il pourrait aussi être exploité pour l'interprétation de modèle *BPMN, MindMap, CMap* et autres. L'avantage notoire de l'utilisation des ontologies dans une telle perspective est que le volet opérationnel des ontologies permet la production de conclusions de domaine, qui peuvent être exploitées pour produire du texte ou encore des modèles semi-formels dans un langage différent de celui du modèle semi-formel source. Ainsi, un modèle semi-formel, qui est à l'origine écrit en MOT, pourrait être traduit en modèle semi-formel cible de type UML. OntoCASE pourrait donc être exploité comme traducteur de modèles.

Pour l'heure, le module de validation génère un texte de la forme: sujet prédicat objet. Des améliorations pourraient être développées afin d'offrir des fonctionnalités de tri selon le sujet, le prédicat ou l'objet. De plus, si un sujet et un objet sont identiques, ils pourraient être concaténés pour former une phrase plus complète du type : sujet1→prédicat1→(objet1/sujet2)→prédicat2→(objet2/sujet3)→prédicat3→objet3, ainsi par exemple, trois énoncés du style : *Bahia est un chien*; *chien est un canin*; *canin est un carnivore* serait formé par *Bahia est un chien qui est un canin qui est un carnivore*. Autrement, des techniques de génération de texte pourraient offrir une meilleure présentation des résultats générés par l'assistant à la validation.

5.4.3 Perspectives en représentation des connaissances

Dans le domaine de la représentation des connaissances, certains éléments d'expressivité des ontologies ou d'autres types de langages ne peuvent pas être

représentés dans le langage MOT. Par exemple, la propriété `owl:inverseOf` n'a pas de correspondant en langage MOT. Certaines fonctionnalités de "*stéréotypage*" pourraient être implantées dans OntoCASE pour permettre la représentation de l'ensemble des éléments d'expressivité d'OWL à partir de l'édition semi-formelle de la conceptualisation d'origine.

En revanche, l'intégration de nouveaux formalismes à la méthodologie permettrait de généraliser davantage l'ontologie de transformation ce qui, à une ultime limite, pourrait être unifiant pour le domaine de la représentation des connaissances.

5.4.4 Perspectives en gestion des connaissances

En élicitation de connaissances, la capacité que possède OntoCASE de produire une interprétation objective et normalisée en langage naturel d'un modèle semi-formel pourrait être exploitée pour stimuler la créativité des experts à exprimer leurs connaissances. Une amélioration de l'assistant informatique à produire une interprétation textuelle du modèle graphique en temps réel pendant l'activité de modélisation pourrait être une contribution intéressante à ce type de recherche. En science cognitive, l'apprentissage des mécanismes structurants de la pensée est parfois ardu. Des recherches sur l'utilisation d'OntoCASE en tant qu'EIAH seraient à explorer dans cette perspective.

Bien qu'il faille valider ce qui suit par une expérimentation, nous pouvons tout de même présumer qu'à la lumière des résultats obtenus, l'assistant informatique d'OntoCASE peut favoriser le passage de connaissances tacites en représentations déclaratives de degré formel, soit une ontologie.

5.4.5 Perspective de recherche pour les Environnements Informatiques pour l'Apprentissage Humain (EIAH)

En EIAH, OntoCASE pourrait fournir un outil d'ingénierie ontologique convivial pour la production d'ontologies dans des domaines d'enseignement variés (par exemple en mathématique, français, philosophie, etc.). Comme indiqué par l'un des participants à l'expérimentation, l'utilisation d'un langage de modélisation semi-formel pour la conception d'ontologies serait un réel gain pour le concepteur pédagogique responsable de la conception et de la diffusion de ce contenu pour les EIAH à base d'ontologies. Par ailleurs, pour la rédaction d'un texte didactique, la modélisation ontologique et sa validation avec OntoCASE pourraient permettre de relever des contradictions dans le texte. Des recherches en ce sens pourraient aboutir à la mise en place d'une méthodologie de développement de contenu pédagogique dont la représentation serait formelle et qui serait utilisable par les enseignants.

En considérant OntoCASE comme EIAH pour l'apprentissage de la modélisation, les dispositifs de rétroaction, qu'il s'agisse de menus contextuels qui limitent le choix des objets de modélisation à des possibilités valides, ou encore des messages liés à la désambiguïsation ou à la gestion des erreurs de modélisation, ou même de l'utilisation des outils de validation syntaxique et sémantique, offrent à l'apprenant en modélisation un outil efficace d'interprétation automatique du modèle semi-formel initialement produit. L'interprétation automatique de ce modèle à travers un processus de désambigüisation et de formalisation ontologique procure les avantages suivants: elle guide l'activité de modélisation pendant l'étape d'explicitation de la connaissance; elle permet une auto-évaluation de la démarche de modélisation; elle permet une validation objective du modèle conçu. L'usage d'OntoCASE en tant qu'EIAH pourrait avoir des effets sur la conceptualisation interne (mentale) des sujets, mais cela serait à vérifier avec des recherches faisant appel à des mesures cognitives.

Par ailleurs, dans une perspective pédagogique, la production automatique d'une validation à la suite de la production d'un modèle semi-formel est une propriété d'OntoCASE qui pourrait être utilisée par un enseignant pour évaluer la performance d'un étudiant. Enfin, pour l’apprentissage de la compréhension de texte, la structure logique du langage de modélisation et les fonctionnalités de validation du modèle semi-formel permettraient une auto-évaluation de l'étudiant et amènerait celui-ci à se poser les bonnes questions pour s'auto corriger.

APPENDICE A

GUIDE DU LANGAGE DE MODÉLISATION PAR OBJETS TYPÉS MOT

Le langage de Modélisation par Objets Typés (MOT) conçu par Paquette (2002b, 2010) est un langage de représentation graphique des connaissances. Nous décrivons d'abord la structure de ce langage et son alphabet. Par la suite, nous présentons les types de connaissances et les relations qui sont exprimables en MOT. Finalement, nous présentons la sémantique de chacune des primitives du langage.

A.1 Structure du langage MOT

Comme la plupart des langages, la structure de MOT se compose d'un alphabet, d'une grammaire et d'une sémantique (voir la figure a.1).

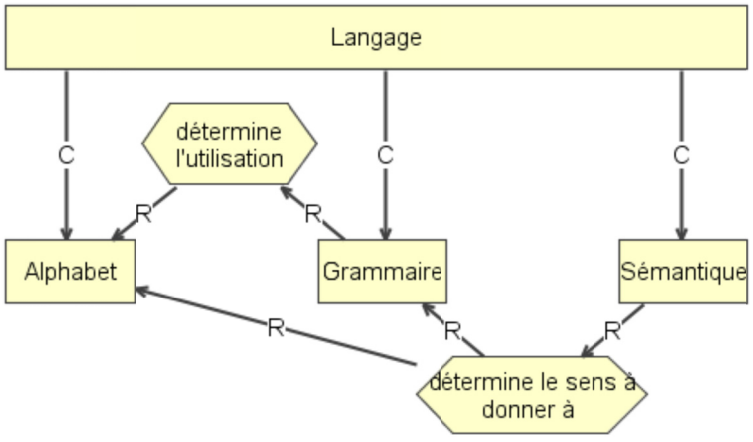


Figure A.1 : Structure générale d'un langage.

L'alphabet est constitué de symboles, d'icônes ou de la représentation de base du langage (ce que l'on appelle parfois les primitives du langage). Par exemple, en langue française, l'alphabet se compose de caractères regroupés en voyelles et en consonnes. La *grammaire,* quant à elle, sert à définir les règles d'utilisation des symboles. L'application des règles est indépendante du sens que représentent les symboles. La *sémantique* est la définition du sens qui est donné aux symboles. Par exemple, dans la figure a.1, deux symboles sont utilisés soit le rectangle représentant un *concept* et la flèche traversée d'un « C » qui indique un *lien de composition*. La règle de grammaire utilisée ici s'énonce comme suit : « *un lien de composition qui a comme origine un concept relie à sa destination un autre concept ».* La sémantique associée à cette règle est : *un concept A est composé d'un concept B*. Par exemple, dans la figure a.1 on peut lire : « le concept *langage* se compose des concepts : *Alphabet, Grammaire* et *Sémantique ».*

A.2 Alphabet du langage MOT

L'alphabet du langage MOT inclut deux types d'entités (comme le sont les voyelles et les consonnes dans le langage naturel) qui sont les *connaissances* et les *relations.* Les *connaissances* peuvent être « abstraites » ou « factuelles ». Ainsi, chaque symbole de MOT est soit une relation, soit une connaissance abstraite ou une connaissance factuelle (voir la figure a.2[114]).

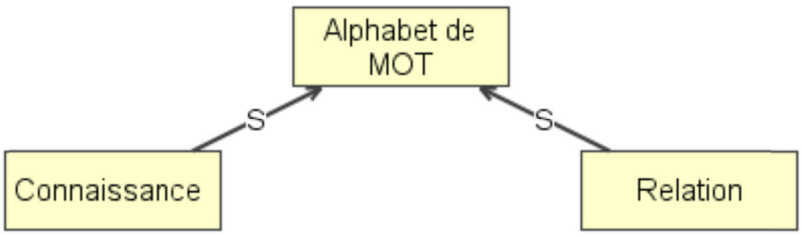


Figure A.2 : Structure de l'alphabet de MOT.

[114] Le lien S qui relie des *Concepts* en est un de spécialisation qui se lit « *sorte de ».* Le lien S est aussi une façon de désigner un sous-ensemble. On peut donc lire que *Relation* est un sous-ensemble de l'*Alphabet de MOT* ou encore « la relation est une sorte d'alphabet de MOT »

La *relation* unit deux connaissances. La *connaissance abstraite* représente quelque chose ressemblant à une idée. Par exemple, dans la phrase « le *chien* est le meilleur ami de l'Homme », le mot « chien » fait référence à un concept, à une idée que l'on se représente de ce qu'est un « chien ». C'est ce qui est appelé une « connaissance abstraite ». En contrepartie, s'il est dit « *Fido* est le meilleur ami de l'Homme », alors le mot « Fido » fait référence à quelque chose qui existe, qu'il est possible de toucher. On dira alors que le mot « Fido » est une connaissance factuelle. La *connaissance factuelle* fait référence à une entité tangible, qu'on peut aussi nommer « objet concret ».

A.3 Types des connaissances en MOT

A.3.1 Alphabet de MOT associé aux types de connaissances

Le langage semi-formel MOT différencie les types de connaissances au moyen de symboles graphiques (voir le tableau a.1 et le tableau a.2). Les connaissances peuvent être combinées au sein d'un même schéma de manière à produire des modèles mixtes de connaissances. Tel que déjà mentionné, à l'instar des théories sur la représentation des connaissances, le langage MOT offre la possibilité de représenter des connaissances selon deux niveaux d'abstraction: conceptuel (*ConnaissanceAbstraite*) et factuel (*Fait*). La figure a.3 présente le vocabulaire de MOT sous une forme taxonomique mettant ainsi en relation les différents éléments du vocabulaire de MOT.

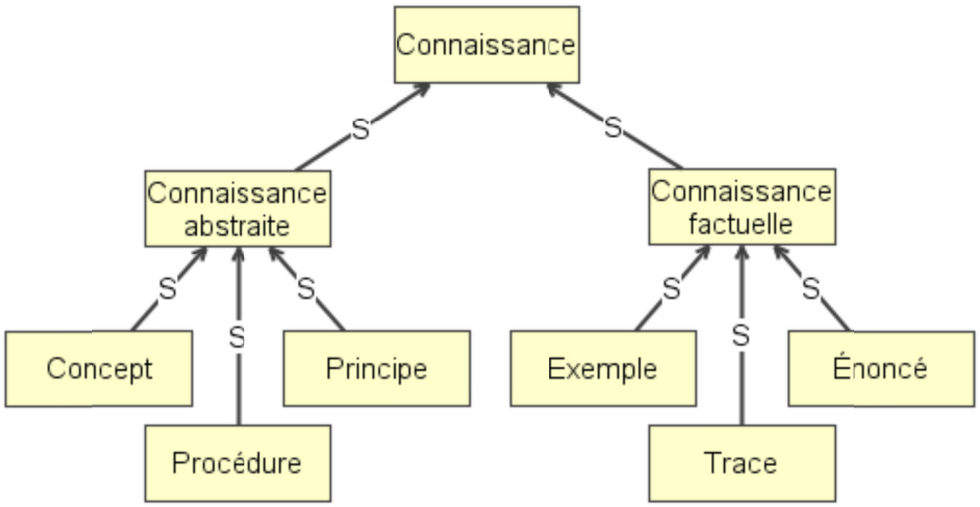


Figure A.3: Représentation des connaissances en langage MOT.

A.3.2 Sémantique de MOT associée aux types de connaissances

Du point de vue de la sémantique, le *concept* représente « le quoi » des choses (voir le tableau a.2). Il sert à décrire l'essence d'un objet concret. Il peut être associé à l'idée de classe ou de catégorie. En ce sens, il est l'abstraction d'un objet concret. L'*exemple* représente l'un de ces objets en énonçant un certain nombre de faits qui le décrivent. La *procédure* permet de décrire « le comment » des choses. Elle désigne des opérations, des actions pouvant être accomplies. La *trace* représente l'ensemble des faits concrets obtenus lors de l'exécution d'une procédure. Le *principe* désigne « le pourquoi », « le quand » ou le « qui » associé à une chose. Il est une connaissance stratégique qui permet de nommer une relation qui existe entre des objets, que ce soit des concepts, des procédures ou d'autres principes. Il sert notamment à représenter une condition pouvant s'appliquer à l'exécution d'une action. L'*énoncé* représente l'instanciation d'un principe à propos d'objets concrets.

Tableau A.1
Type de connaissances dans le langage MOT et leur symbole associé.

Type de connaissance	Connaissance abstraite		Connaissance factuelle	
Déclarative *Le quoi des choses*	Concept		Exemple	
Action *Le comment de choses*	Procédure		Trace	
Stratégique *Le pourquoi, le quand, le qui*	Principe		Énoncé	

A.3.3 Stéréotype

Le stéréotype ne fait pas partie de la définition de base du langage MOT, tel que défini par Paquette (2002b). Cependant, nous l'avons intégré dans la présente définition du langage MOT. Largement utilisé et standardisé en modélisation UML (Alhir, 2002), le stéréotype est une extension du vocabulaire qui permet à un modélisateur d'associer un élément d'un modèle à un autre domaine de connaissances. Par exemple, une connaissance procédurale P pourrait être stéréotypée par une tâche, une procédure ou encore une méthode (voir la figure a.4). Le stéréotype est encapsulé par les symboles « ».

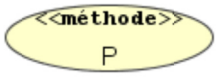


Figure A.4: Représentation de la procédure P dont le stéréotype une méthode.

A.4 Type de relations dans MOT

A.4.1 Alphabet des relations

La *relation* est un lien directionnel qui unit des connaissances. Le langage MOT offre un ensemble de liens qui sont typés (voir la figure a.5). Le lienI représente une relation d'instanciation, le lienS représente une relation de spécialisation, le lienR représente une relation de régulation, le lienA représente une association d'application, le lienP représente la préséance, le lienIP représente une association

d'intrant/produit, le lienC/C* représente l'association de composition et de composition multiple et, enfin, le lienE représente une relation d'englobement.

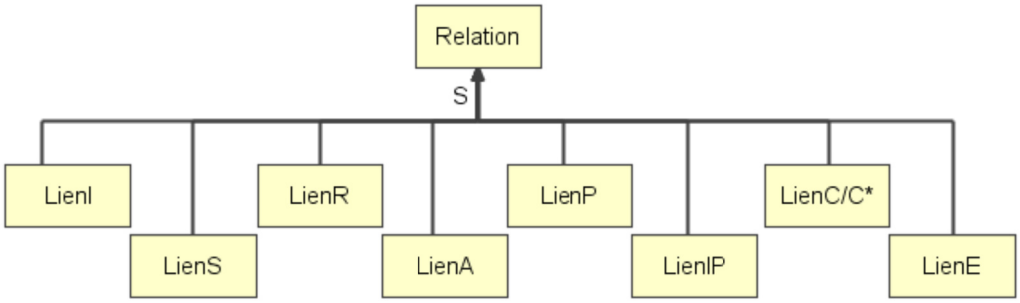


Figure A.5: Hiérarchie des relations typées utilisées en langage MOT et leur représentation dans l'ontologie du langage semi-formel.

A.4.2 Sémantique des relations

Chaque type de lien possède une sémantique propre (voir le tableau a.2) qui respecte des règles d'intégrité (voir le tableau a.3). Par exemple, un lien de spécialisation (lien S) unit deux connaissances abstraites, qui doivent être de même nature. L'ensemble de ces règles d'intégrité sont décrites dans Paquette (2002b).

Tableau A.2
Sémantique des relations typées dans MOT (adapté de Paquette 2002b, 2010)

Type de lien	Signification
S	Le lien de spécialisation associe deux connaissances abstraites de même type dont la première est une spécialisation de la seconde. Ce lien est notamment utile dans la description des taxonomies. Le lien de spécialisation est une relation transitive.
I	Le lien d'instanciation associe à une connaissance abstraite un fait qui caractérise une instance de cette connaissance. Le lien d'instanciation n'est pas une relation transitive.
I/P	Le lien intrant/produit sert à associer une connaissance procédurale à une connaissance conceptuelle afin de représenter l'intrant ou le produit d'une procédure. Ce lien est notamment utile dans la description des algorithmes, des processus et des méthodes. Le lien intrant/produit n'est pas une relation transitive.
P	Le lien de précédence associe une connaissance à une autre qui la suit dans une séquence temporelle de procédures ou de règle de décision (principes). Le lien de précédence est une relation transitive.
R	Le lien de régulation associe une connaissance stratégique (un Principe ou un Énoncé) à une autre connaissance afin de préciser une contrainte, une restriction ou une règle qui régit la connaissance. Le lien de régulation est une relation non-transitive.
C, C*	Les liens de composition et de composition multiple permettent de représenter l'association entre une connaissance et des connaissances qui la composent. Le lien de composition est une relation transitive.
E	Le lien englobe ne possède pas de symbolique particulière dans le langage MOT original. Il s'agit d'une relation qui unit l'élément d'un modèle aux éléments d'un sous-modèle. Le lien englobe est une relation transitive.

Certaines règles d'association entre des connaissances sources et des connaissances destinations sont appliquées à chacun des types de relation. Ces règles définissent les relations considérées valides entre les différents types de connaissances du point de vue de la sémantique MOT.

Voici les quelques règles générales d'utilisation :

- Règle 1 : une relation ne peut pas exister seule; elle doit, à son origine et à sa destination, référer à une connaissance (factuelle et/ou abstraite selon le cas).
- Règle 2 : il est possible qu'une relation possède la même connaissance d'origine et de destination.
- Règle 3 : une connaissance peut exister seule, sans qu'elle soit l'origine ou la destination d'une relation.

Plus spécifiquement, il existe un ensemble de règles secondaires qui régissent chacune des relations en fonction de la nature des connaissances d'origine et de destination qu'elles associent. Le tableau a.3 présente, en format condensé, l'ensemble des règles d'union des relations et des connaissances de MOT.

Tableau A.3
Grammaire des relations MOT (adapté de Paquette 2002b)

Destination / *Origine*	**Connaissances abstraites**			**Connaissances factuelles**		
	Concept	**Procédure**	**Principe**	**Exemple**	**Trace**	**Énoncé**
Concept	C, S	I/P	R[115]	I, C		
Procédure	I/P	C, S, P	C, P		I, C	
Principe	R	C, R, P	C, S, P, R			I, C
Exemple	A	A	A	A, C	A, I/P	A
Trace	A	A	A	A, I/P	A, C, P	A, C, P
Énoncé	A	A	A	A, R	A, C, R, P	A, C, R, P

Le tableau a.3 s'interprète de la façon suivante : prenons par exemple la première case du haut à gauche. On y lit « C, S ». L'interprétation, sous forme de règle, s'inscrit comme suit :

- Règle C1 : si l'origine d'un lien est un concept et que la destination est un concept, alors la relation peut être de type « C » (qui est un lien de composition);
- Règle S1 : si l'origine d'un lien est un concept et que la destination est un concept, alors la relation peut être de type « S » (qui est un lien de spécialisation);

Chacune des cases du tableau s'interprète selon la même lecture impliquant un ensemble de cas d'utilisation pour chacun des liens.

[115] La relation « R » entre un Concept et un Principe est un ajout de notre part. Elle ne fait pas pas partie de la grammaire originale de MOT proposée par Paquette (2002).

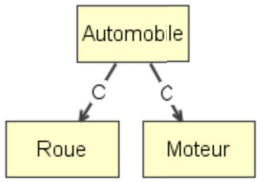
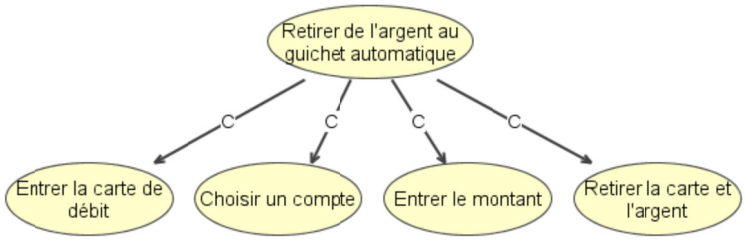
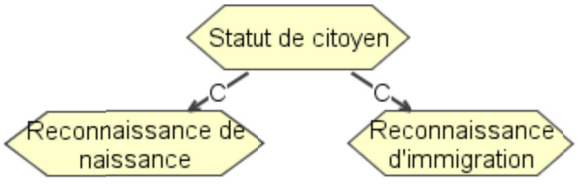
A.5 Sémantique des éléments grammaticaux du langage MOT

Nous venons de définir les constituants du vocabulaire, de la sémantique du chacun des éléments de vocabulaire ainsi que de la grammaire de MOT. Maintenant, nous nous attardons sur la sémantique des éléments grammaticaux de MOT.

A.5.1 Composition

Le lien « C », qui se lit : « lien de composition », sert à représenter les composants, les constituants d'une connaissance (voir l'exemple du tableau a.4). Il permet d'indiquer qu'une connaissance se compose d'une ou plusieurs autres connaissances.

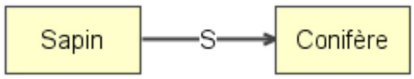
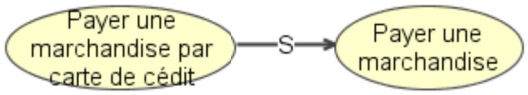
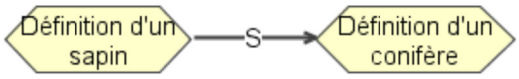
Tableau A.4
Exemple de représentation d'une composition entre connaissances

Origine/ Destination	**Exemple/ Représentation en MOT**	
Concept / Concept	Une « Automobile » se compose de « Moteur », de « Roue »	
Procédure/ Procédure	« retirer de l'argent au guichet automatique » se compose de : » entrer la carte de débit », « entrer le NIP », « choisir un compte », « entrer le montant », « retirer la carte et l'argent ». 	
Principe/ Principe	Le « statut de citoyen » se compose des règles de « reconnaissance de naissance », de « reconnaissance d'immigration ». 	

A.5.2 Spécialisation

Le lien « S », qui se lit : « lien de spécialisation », sert à représenter la spécialisation d'une connaissance par rapport à une autre (voir l'exemple du tableau a.5). Il permet de désigner des cas particuliers de connaissances conceptuelles. Les connaissances liées par un lien de spécialisation sont des connaissances de même type.

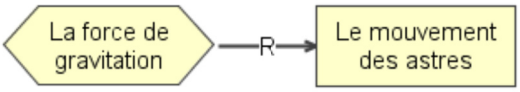
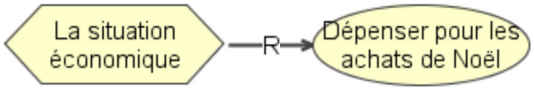
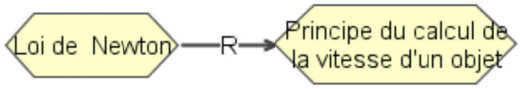
Tableau A.5
Exemple de représentation de la spécialisation de connaissances

Origine/ Destination	**Exemple/ Représentation en MOT**
Concept / Concept	Un « Sapin » est une sorte de « Conifère » 
Procédure/ Procédure	« Payer une marchandise par carte de crédit » est une sorte de façon de « payer une marchandise » 
Principe/ Principe	La « définition d'un sapin » est une sorte de « définition de conifère ». 

A.5.3 Régulation

Le lien « R », qui se lit : « lien de régulation », est une relation qui met en jeu un principe et l'une ou l'autre des connaissances abstraites (voir l'exemple du tableau a.6). En tant qu'agent, norme ou contrainte, le principe est utilisé en conjonction avec un lien de régulation pour indiquer une situation de régulation d'un objet par un autre.

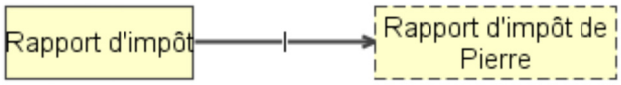
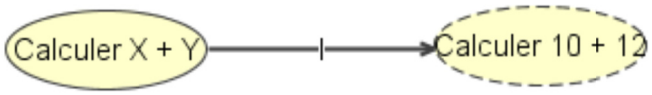
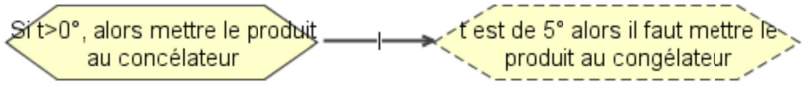
Tableau A.6
Exemple de représentation d'une régulation entre des connaissances

Origine/ Destination	**Exemple/ Représentation en MOT**
Principe / Concept	« La force gravitation » régit « le mouvement des astres ». 
Principe/ Procédure	« La situation économique » régit l'action « de dépenser pour les achats de Noël ». 
Principe/ Principe	La « Loi de Newton » régit le « principe du calcul de la vitesse d'un objet ». 

A.5.4 Instanciation

Le lien « I », qui se lit : « lien d'instanciation », met en relation une connaissance abstraite et la connaissance factuelle de même type (voir le tableau a.7). Il sert à représenter la relation entre un objet concret et l'abstraction qui lui est associée.

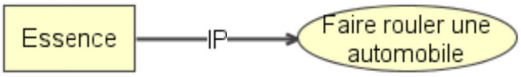
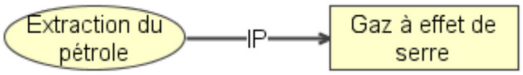
Tableau A.7
Exemple de représentation de l'instanciation entre une connaissance abstraite et un connaissance factuelle

Origine/ Destination	**Exemple/ Représentation en MOT**
Concept / Exemple	Le « Rapport d'impôt » a pour instance « le rapport d'impôt de Pierre » 
Procédure / Trace	« Calculer 10 + 12 » est un calcul de type « calculer X + Y » 
Principe / Énoncé	L'énoncé « t est de 5° alors il faut mettre le produit au congélateur » est une instanciation de la règle « Si t>0° alors mettre le produit au congélateur » 

A.5.5 Intrant et le produit

Le lien « I/P », qui se lit : « lien intrant/produit », met en relation une connaissance de type procédure et de type concept (voir l'exemple du tableau a.8). Il sert à désigner les composants nécessaires à la réalisation de la procédure ainsi que les objets produits par la procédure.

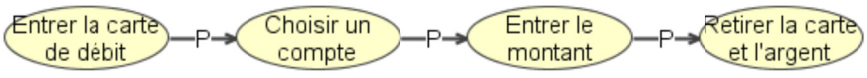
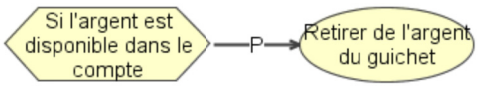
Tableau A.8
Exemple représentation d'un intrant et d'un produit entre une connaissance procédurale et une connaissance conceptuelle

Origine/ Destination	**Exemple/ Représentation en MOT**
Concept / Procédure	L' » Essence » est l'intrant du processus de « faire rouler une automobile » 
Procédure/ Concept	Le processus d' » extraction du pétrole » produit des « Gaz à effet de serre » 

A.5.6 Précédence

Le lien « P », qui se lit « lien de précédence » met en relation des procédures ou des principes (voir l'exemple du tableau a.9). Il sert à ordonner la séquence d'exécution des procédures ou l'ordonnancement de l'application de principes.

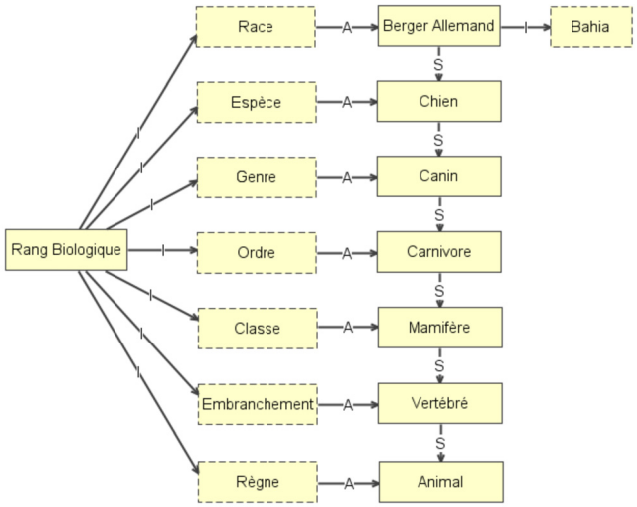
Tableau A.9
Exemple de représentation de la préséance

Origine/ Destination	**Exemple/ Représentation en MOT**
Procédure/ Procédure	le processus : » entrer la carte » précède » choisir un compte » qui précède » entrer le montant » qui précède « retirer la carte et l'argent » 
Principe/ Procédure	« On retire l'argent du guichet » à condition que « l'argent soit disponible dans le compte » 

A.5.7 Lien d'application

Le lien « A », qui se lit « lien d'application », met en relation une connaissance factuelle avec une connaissance abstraite (voir l'exemple du tableau a.10). Dans la majorité des situations, un modèle MOT sert à représenter le vocabulaire ainsi que les entités qui composent un domaine d'application spécifique. Dans certains cas, le vocabulaire d'un domaine d'application sert à la représentation d'un autre domaine. On dira alors que le vocabulaire du premier domaine d'application est la métaconnaissance du vocabulaire du deuxième domaine d'application.

Tableau A.10
Exemple de représentation d'un lien d'application

Origine/ Destination	**Exemple/ Représentation en MOT**
Connaissance factuelle/ Connaissance Abstraite	Dans cet exemple, « Bahia » qui est de la « Race » des « Berger », qui est de l' « Espèce » des « Chien » … qui est du « Règne » « Animal ». La « Race », l' « Espèce », le « Genre », …, le « Règne » sont des entités du concept « Rang Biologique ». 

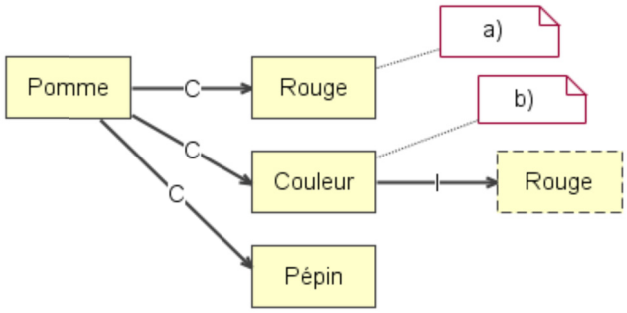
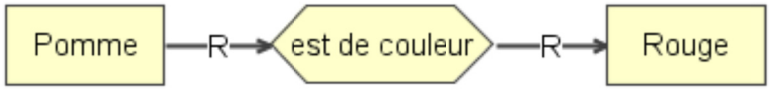
A.5.8 Propriété et l'attribut

L'attribut et la propriété sont deux associations qui mettent en relation deux connaissances abstraites (voir l'exemple au tableau a.11). La représentation d'une

relation d'attribut est exprimée par le LienC entre deux concepts. Cette utilisation du LienC crée une ambiguïté avec l'interprétation de composition. C'est la sémantique du domaine représenté, qui permet de déterminer l'interprétation adéquate du l'utilisation du LienC.

Par ailleurs, la représentation de la propriété nécessite une utilisation particulière du LienR, qui ne fait pas partie de la définition initiale du langage MOT. Il s'agit d'inclure un principe entre deux concepts reliés par des LienR. Le principe prend alors le rôle de propriété qui unit deux concepts. Cette combinaison peut aussi s'appliquer pour la définition de propriétés entre des connaissances procédurales.

Tableau A.11
Exemple de représentation d'un attribut et d'une propriété

Exemple/ **Représentation en MOT**
« La pomme » se compose de « pépin » et a) « La pomme » a pour attribut « Rouge » ; b) *de manière plus formelle:* » la pomme » a pour attribut « couleur » qui a la valeur d'être « rouge » 
« la pomme » « est de couleur » « rouge » 

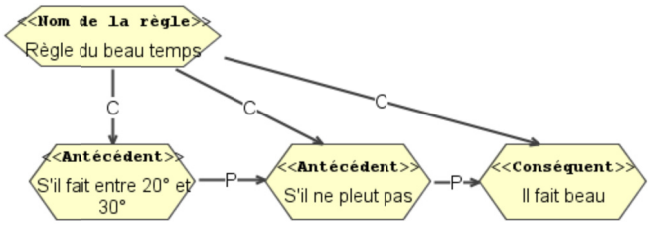
A.5.9 Règle

La règle est un énoncé qui se compose du nom de la règle et d'un ensemble d'antécédents mis en conjonction pour produire une conclusion ou une opération. En

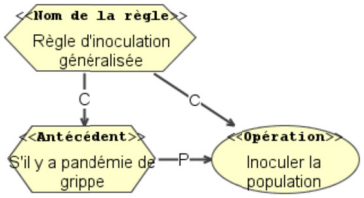
MOT, il est possible de représenter une règle (voir l'exemple du tableau a.12) par une topologie particulière dans l'utilisation de principes, de LienC, de LienP et de procédures. La signification de chacun de ces éléments du modèle est mise en stéréotype.

Tableau A.12
Exemple de représentation d'une règle

« La règle du beau temps »: « S'il fait entre 20° et 30° » et « s'il ne pleut pas » alors « il fait beau »



« La règle d'inoculation généralisée »: « S'il y a pandémie de grippe » alors « inoculer la population »



A.6 Résumé

Nous avons vu que le langage de Modélisation par Objet Typé permet la représentation de connaissances déclaratives (Concept et Exemple), procédurales (Procédure et Trace) et stratégiques (Principe et Énoncé). Les connaissances sont mises en relation par l'utilisation des liens typés : composition, spécialisation, précédence, intrant/produit, d'instance et de régulation. Nous avons aussi vu que chaque relation est régie par des règles d'utilisation et que ces règles sont associées à une sémantique propre.

APPENDICE B

CATALOGUE DE LA SÉMANTIQUE FORMELLE DE MOT

B.1 Utilisation du catalogue

Le catalogue de la sémantique formelle de MOT présente la signification formelle de l'utilisation de MOT. Chaque section est divisée selon la structure présentée au tableau b.1. Le titre du l'élément du catalogue concerne l'interprétation possible d'un contexte d'utilisation des éléments du langage. Certaines sections présentent le métamodèle de l'utilisation de l'élément présenté. Pour chaque section, est présenté un exemple de représentation en langage MOT. Cet exemple est présenté de manière formelle en langage OWL-N3. Le nom des règles de désambiguïsation et de conversion utilisées fait référence aux règles détaillées à l'appendice F. Finalement, les rapports de validation syntaxique et sémantique concluent la section.

Tableau B.1
Structure de chaque élément du catalogue

Titre de l'élément du catalogue
Méta-modèle de la règle d'utilisation
Exemple de représentation en langage MOT
Sémantique formelle
Règles de désambiguïsation
Règles de conversion
Rapport de validation syntaxique
Rapport de validation sémantique

Le tableau b.2 a la fonctionnalité de servir d'index pour les éléments du catalogue. Chaque règle valide de liaison entre connaissances MOT est accompagnée du numéro de page correspondant à son interprétation.

Tableau B.2:
Tableau des règles concernant chaque relation légale en MOT combiné au numéro de page de l'interprétation

Destination / *Origine*	**Connaissances abstraites**			**Connaissances factuelles**		
	Concept	**Procédure**	**Principe**	**Exemple**	**Trace**	**Énoncé**
Concept	C(265, 271), S(251)	I/P(255)	R(258, 262, 264)	I(253, 271), C(269)		
Procédure	I/P(255)	C(266), S(251), P(274)	C(278), P(275)		I(253), C(269)	
Principe	R(258, 262)	C(278), R(258), P(275, 278)	C(265, 278), S(251), P(275, 278), R(258)			I(253), C(269)
Exemple				C(267)	I/P(257)	R(262)
Trace				I/P(257)	C(267, 281), P(274)	C(), P(275)
Énoncé				R(260, 262)	C(281), R(260), P(275, 281)	C(267, 281), R(260), P(275, 281)

B.2 Entités atomiques d'OntoCASE

Chaque élément du vocabulaire de MOT comporte une sémantique dont certains sont ambigües. La figure b. présente l'interprétation possible de chacun de ces éléments de vocabulaire et la figure b.2 présente un exemple préalablement désambiguïsé de ces éléments de vocabulaire.

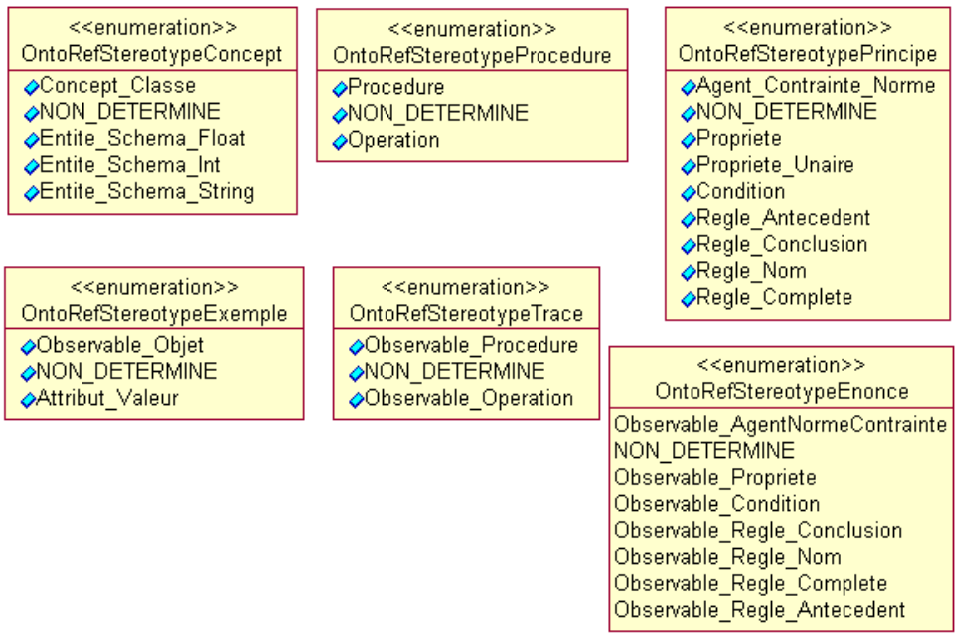


Figure B.1:La représentation en UML des différentes valeurs possibles des entités d'OntoCASE.

Exemple de représentation en langage MOT

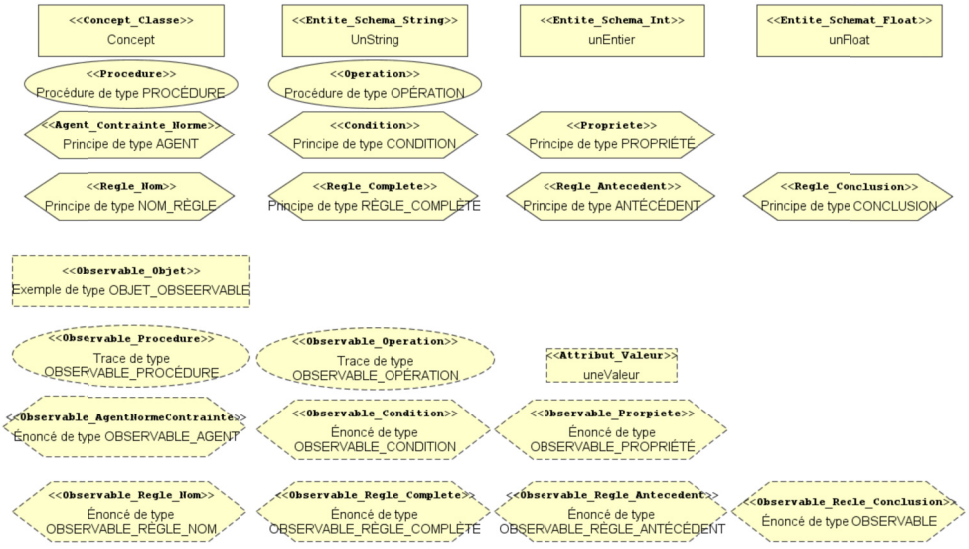


Figure B.2 : Exemple de définition des entités atomiques de MOT et leur stéréotype de désambiguïsation possible.

Sémantique formelle après l'étape de formalisation

```
:Concept_0
      rdf:type owl:Class ;
      rdfs:label "Concept"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:Enonce-de-type-OBSERVABLE_16
      rdf:type metaDom:MD_Strategique_Entite_Regle_Conclusion ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE"^^xsd:string .
:Enonce-de-type-OBSERVABLE_AGENT_13
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE_AGENT"^^xsd:string .
:Enonce-de-type-OBSERVABLE_CONDITION_14
      rdf:type metaDom:MD_Strategique_Condition ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE_CONDITION"^^xsd:string .
:Enonce-de-type-OBSERVABLE_PROPRIETE_15
      rdf:type owl:ObjectProperty ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE_PROPRIÃ‰TÃ‰"^^xsd:string ;
      rdfs:subPropertyOf metaDom:MD_ASSERTION .
:Enonce-de-type-OBSERVABLE_REGLE_ANTECEDENT_19
      rdf:type metaDom:MD_Strategique_Entite_Regle_Antecedent ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE_RÃ^GLE_ANTÃ‰CÃ‰DENT"^^xsd:string .
:Enonce-de-type-OBSERVABLE_REGLE_COMPLETE_18
      rdf:type metaDom:MD_Strategique_Entite_Regle_Complete ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE_RÃ^GLE_COMPLÃ^TE"^^xsd:string .
:Enonce-de-type-OBSERVABLE_REGLE_NOM_17
      rdf:type metaDom:MD_Strategique_Entite_Regle_Nom ;
      rdfs:label "Ã‰noncÃ© de type OBSERVABLE_RÃ^GLE_NOM"^^xsd:string .
:Exemple-de-type-OBJET_OBSEERVABLE_10
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Exemple de type OBJET_OBSEERVABLE"^^xsd:string .
:Principe-de-type-AGENT_3
      rdf:type owl:Class ;
      rdfs:label "Principe de type AGENT"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Principe-de-type-ANTECEDENT_6
      rdf:type owl:Class ;
      rdfs:label "Principe de type ANTÃ‰CÃ‰DENT"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent , owl:Thing .
:Principe-de-type-CONCLUSION_7
      rdf:type owl:Class ;
      rdfs:label "Principe de type CONCLUSION"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Entite_Regle_Conclusion .
:Principe-de-type-CONDITION_5
      rdf:type owl:Class ;
      rdfs:label "Principe de type CONDITION"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Condition .
:Principe-de-type-NOM_REGLE_8
      rdf:type owl:Class ;
      rdfs:label "Principe de type NOM_RÃ^GLE"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Nom , owl:Thing .
:Principe-de-type-PROPRIETE_4
      rdf:type owl:ObjectProperty ;
      rdfs:label "Principe de type PROPRIÃ‰TÃ‰"^^xsd:string ;
      rdfs:subPropertyOf metaDom:MD_PROPRIETE .
:Principe-de-type-REGLE_COMPLETE_9
      rdf:type owl:Class ;
      rdfs:label "Principe de type RÃ^GLE_COMPLÃ^TE"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Complete , owl:Thing .
:Procedure-de-type-OPERATION_2
      rdf:type owl:Class ;
      rdfs:label "ProcÃ©dure de type OPÃ‰RATION"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Procedurale_Operation .
:Procedure-de-type-PROCEDURE_1
      rdf:type owl:Class ;
      rdfs:label "ProcÃ©dure de type PROCÃ‰DURE"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Trace-de-type-OBSERVABLE_OPERATION_12
      rdf:type metaDom:MD_Procedurale_Operation ;
      rdfs:label "Trace de type OBSERVABLE_OPÃ‰RATION"^^xsd:string .
:Trace-de-type-OBSERVABLE_PROCEDURE_11
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Trace de type OBSERVABLE_PROCÃ‰DURE"^^xsd:string .
:UnString_20
      rdf:type owl:Class ;
      rdfs:label "UnString"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_String .
:unEntier_21
      rdf:type owl:Class ;
      rdfs:label "unEntier"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_Int .
:unFloat_22
      rdf:type owl:Class ;
      rdfs:label "unFloat"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_Float .
:uneValeur_23
      rdf:type metaDom:MD_Declarative_Schema ;
      rdfs:label "uneValeur"^^xsd:string .
```

Liste des règles de conversion utilisée

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2-5k_Creation_AssertionDeDomaine]
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2c_Creation_Individu_operation]
 [Rule-01-2d_Creation_Individu_Principe_Agent]
 [Rule-01-2e_Creation_Individu_Principe_Regle_Nom]
 [Rule-01-2f_Creation_Individu_Principe_Regle_Complete]
 [Rule-01-2g_Creation_Individu_Principe_Antecedent]
 [Rule-01-2h_Creation_Individu_Principe_Conclusion]
 [Rule-01-2i_Creation_Individu_Principe_Condition]
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-c_Creation_class_Action_Operation]
 [Rule-01-d_Creation_class_agent]
 [Rule-01-e_Creation_class_condition]
 [Rule-01-f_Creation_class_regle_nom]
 [Rule-01-g_Creation_class_regle_complete]
 [Rule-01-h_Creation_class_regle_antecedent]
 [Rule-01-i_Creation_class_regle_conclusion]
 [Rule-01-k_Creation_ProprieteDeDomaine]
 [Rule-13-b_Creation_class_Schema_Int]
 [Rule-13-c_Creation_class_Schema_Float]
 [Rule-13-d_Creation_class_Schema_String]
 [Rule-13-e-Creation_Dun_Individu_dattribut]
etat_classification.owl
etat_final.owl
 [Rule-00_Sauvegarde]
etat_validation.owl
```

Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
Concept est une sorte de (metaDom:MD_Declarative_Concept)
Enonce-de-type-OBSERVABLE_PROPRIETE_15([],[]) est une assertion de la propriété (metaDom:MD_ASSERTION)
Principe de type AGENT est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Principe de type ANTÉCÉDENT est une sorte de (metaDom:MD_Strategique_Entite_Regle_Antecedent)
Principe de type CONCLUSION est une sorte de (metaDom:MD_Strategique_Entite_Regle_Conclusion)
Principe de type CONDITION est une sorte de (metaDom:MD_Strategique_Condition)
Principe de type NOM_RÈGLE est une sorte de (metaDom:MD_Strategique_Entite_Regle_Nom)
Principe de type RÈGLE_COMPLÈTE est une sorte de (metaDom:MD_Strategique_Entite_Regle_Complete)
Procédure de type OPÉRATION est une sorte de (metaDom:MD_Procedurale_Operation)
Procédure de type PROCÉDURE est une sorte de (metaDom:MD_Procedurale_Procedure)
UnString est une sorte de (metaDom:MD_Declarative_Schema_String)
[Exemple de type OBJET_OBSEERVABLE] est de catégorie (metaDom:MD_Declarative_Concept)
[Exemple de type OBJET_OBSEERVABLE] est un ( :Connaissance d'objet )
[Trace de type OBSERVABLE_OPÉRATION] est de catégorie (metaDom:MD_Procedurale_Operation)
[Trace de type OBSERVABLE_OPÉRATION] est un ( :Opération )
[Trace de type OBSERVABLE_PROCÉDURE] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Trace de type OBSERVABLE_PROCÉDURE] est un ( :Procédure )
[uneValeur] est un ( :Connaissance constantes (Schéma) )
[Énoncé de type OBSERVABLE] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Conclusion)
[Énoncé de type OBSERVABLE] est un ( :Conclusion d'une règle )
[Énoncé de type OBSERVABLE_AGENT] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[Énoncé de type OBSERVABLE_AGENT] est un ( :Agent-Contrainte-Norme )
[Énoncé de type OBSERVABLE_CONDITION] est de catégorie (metaDom:MD_Strategique_Condition)
[Énoncé de type OBSERVABLE_CONDITION] est un ( :Condition )
[Énoncé de type OBSERVABLE_RÈGLE_ANTÉCÉDENT] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Antecedent)
[Énoncé de type OBSERVABLE_RÈGLE_ANTÉCÉDENT] est un ( :Antécédent d'une règle )
[Énoncé de type OBSERVABLE_RÈGLE_COMPLÈTE] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Complete)
[Énoncé de type OBSERVABLE_RÈGLE_COMPLÈTE] est un ( :Règle complète )
[Énoncé de type OBSERVABLE_RÈGLE_NOM] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Nom)
[Énoncé de type OBSERVABLE_RÈGLE_NOM] est un ( :Nom d'une règle )
unEntier est une sorte de (metaDom:MD_Declarative_Schema_Int)
unFloat est une sorte de (metaDom:MD_Declarative_Schema_Float)

------------------Résultats après inférence-----------------------
[Exemple de type OBJET_OBSEERVABLE] est un ( :Connaissance d'objet :Type de connaissances :Connaissance
déclarative )
[Trace de type OBSERVABLE_OPÉRATION] est un ( :Opération :Type de connaissances :Connaissance d'actions )
[Trace de type OBSERVABLE_PROCÉDURE] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[uneValeur] est un ( :Connaissance constantes (Schéma) :Type de connaissances :Connaissance déclarative )
[Énoncé de type OBSERVABLE] est un ( :Conclusion d'une règle :Type de connaissances :Connaissance stratégique
:Élément d'une règle )
[Énoncé de type OBSERVABLE_AGENT] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance
stratégique )
[Énoncé de type OBSERVABLE_CONDITION] est un ( :Condition :Type de connaissances :Connaissance stratégique )
[Énoncé de type OBSERVABLE_RÈGLE_ANTÉCÉDENT] est un ( :Antécédent d'une règle :Type de connaissances
:Connaissance stratégique :Élément d'une règle )
[Énoncé de type OBSERVABLE_RÈGLE_COMPLÈTE] est un ( :Règle complète :Type de connaissances :Connaissance
stratégique :Élément d'une règle )
[Énoncé de type OBSERVABLE_RÈGLE_NOM] est un ( :Nom d'une règle :Type de connaissances :Connaissance
stratégique :Élément d'une règle )
```

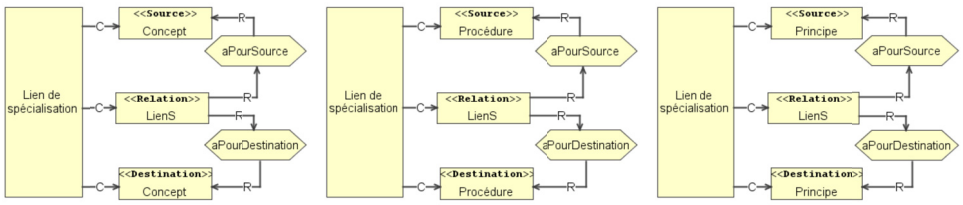
Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/ENTITE_ATOMIQUE_ONTOCASE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/ENTITE_ATOMIQUE_ONTOCASE_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
-------------------- Les connaissances  -----------------------
-----------------------------------------------------------------
------------------ Commun aux deux modèles ---------------------
      Concept   : Concept_0
      Concept   : UnString_20
      Concept   : unEntier_21
      Concept   : unFloat_22
      Enonce    : Enonce-de-type-OBSERVABLE_16
      Enonce    : Enonce-de-type-OBSERVABLE_AGENT_13
      Enonce    : Enonce-de-type-OBSERVABLE_CONDITION_14
      Enonce    : Enonce-de-type-OBSERVABLE_PROPRIETE_15
      Enonce    : Enonce-de-type-OBSERVABLE_REGLE_ANTECEDENT_19
      Enonce    : Enonce-de-type-OBSERVABLE_REGLE_COMPLETE_18
      Enonce    : Enonce-de-type-OBSERVABLE_REGLE_NOM_17
      Exemple   : Exemple-de-type-OBJET_OBSEERVABLE_10
      Exemple   : uneValeur_23
      Principe  : Principe-de-type-AGENT_3
      Principe  : Principe-de-type-ANTECEDENT_6
      Principe  : Principe-de-type-CONCLUSION_7
      Principe  : Principe-de-type-CONDITION_5
      Principe  : Principe-de-type-NOM_REGLE_8
      Principe  : Principe-de-type-PROPRIETE_4
      Principe  : Principe-de-type-REGLE_COMPLETE_9
      Procédure : Procedure-de-type-OPERATION_2
      Procédure : Procedure-de-type-PROCEDURE_1
      Trace     : Trace-de-type-OBSERVABLE_OPERATION_12
      Trace     : Trace-de-type-OBSERVABLE_PROCEDURE_11
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------ Commun aux deux modèles ---------------------
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 4, Reconstruit = 4, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 7, Reconstruit = 7, Différence = 0
Compte des Enonces   : Original = 7, Reconstruit = 7, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

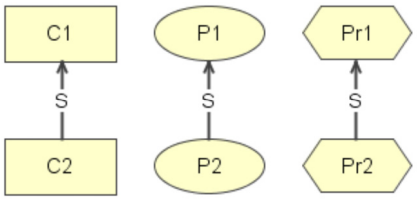
B.3 Spécialisation et l'instance

B.3.1 Spécialisation entre deux concepts, deux procédures et deux principes

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_0
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:C2_1
      rdf:type owl:Class ;
      rdfs:label "C2"^^xsd:string ;
      rdfs:subClassOf :C1_0 .
:P1_2
      rdf:type owl:Class ;
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:P2_3
      rdf:type owl:Class ;
      rdfs:label "P2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , :P1_2 .
:Pr1_4
      rdf:type owl:Class ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Pr2_5
      rdf:type owl:Class ;
      rdfs:label "Pr2"^^xsd:string ;
      rdfs:subClassOf :Pr1_4 , metaDom:MD_Strategique_AgentContrainteNorme .
```

Règles de désambiguïsation typologique

```
[Rule-02-a_desamb_lienS-entre_Concepts]
 [Rule-02-b_desamb_lienS_entre_Procedures]
 [Rule-02-c_desamb_lienS_entre_Principes]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-d_Creation_class_agent]
etat_classification.owl
 [Rule-02-a_Creation_subClassOf_Declaratif]
 [Rule-02-b_Creation_subClassOf_Procedurale]
 [Rule-02-c_Creation_subClassOf_Strategique]
etat_final.owl
 [Rule-00_Sauvegarde]
etat_validation.owl
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/SPECIALISATION_ENTRE_CONCEPTS_PROCEDURE_PRINCIPE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/SPECIALISATION_ENTRE_CONCEPTS_PROCEDURE_PRINCIPE_rest.mot
Comparaison des documents :
------------------------------------------------------------------
---------------------- Les connaissances  ------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      Concept   : C1_0
      Concept   : C2_1
      Principe  : Pr1_4
      Principe  : Pr2_5
      Procédure : P1_2
      Procédure : P2_3
------------------------------------------------------------------
----------------------- Les relations ----------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      LienS: source C2_1 : dest C1_0
      LienS: source P2_3 : dest P1_2
```

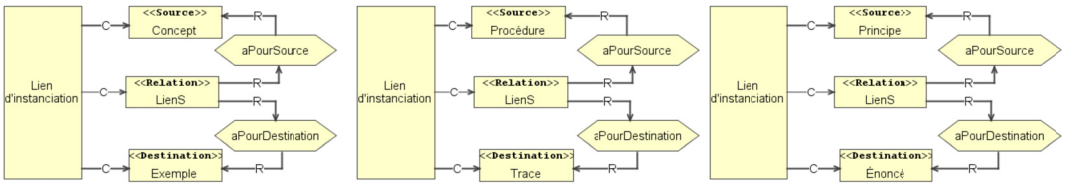
```
      LienS: source Pr2_5 : dest Pr1_4
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 3, Reconstruit = 3, Différence = 0
```

Rapport de validation sémantique

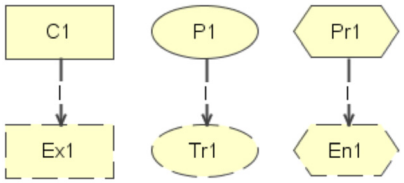
```
----------Début du processus de validation sémantique-------------
C1 est une sorte de (metaDom:MD_Declarative_Concept)
C2 est une sorte de C1
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
P2 est une sorte de (metaDom:MD_Procedurale_Procedure)
P2 est une sorte de P1
Pr1 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Pr2 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Pr2 est une sorte de Pr1
------------------Résultats après inférence-----------------------
```

B.3.2 Instanciation entre une abstraction et son observable

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_0
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:En1_5
      rdf:type :Pr1_2 ;
      rdfs:label "En1"^^xsd:string .
:Ex1_3
      rdf:type :C1_0 ;
      rdfs:label "Ex1"^^xsd:string .
:P1_1
      rdf:type owl:Class ;
```

```
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Pr1_2
      rdf:type owl:Class ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Tr1_4
      rdf:type :P1_1 ;
      rdfs:label "Tr1"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-00-b-desamb_LienI]
 [Rule-03-a_desamb_LienI_Concept_Exemple-OBJET]
 [Rule-03-aa_desamb_exemple_Concept_Instance]
 [Rule-03-b_desamb_LienI_Principe_Enonce]
 [Rule-03-c_desamb_LienI_Procedure_Trace-PROCEDURE]
```

Le principe et l'énoncé sont désambiguïsés de manière manuelle

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2d_Creation_Individu_Principe_Agent]
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-d_Creation_class_agent]
etat_classification.owl
 [Rule-03-a_Creation_typeOf_OBJET]
 [Rule-03-b_Creation_typeOf_ACTION-PROCEDURE]
 [Rule-03-c_Creation_typeOf_PRINCIPE-AGENT]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/INSTANCE_ENTRE_CONN_ABSTRAITES_CONN_FACTUELS.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/INSTANCE_ENTRE_CONN_ABSTRAITES_CONN_FACTUELS_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
---------------------- Les connaissances  ----------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
Concept   : C1_0
Enonce    : En1_5
Exemple   : Ex1_3
Principe  : Pr1_2
Procédure : P1_1
Trace     : Tr1_4
-----------------------------------------------------------------
----------------------- Les relations -------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
LienI: source C1_0 : dest Ex1_3
LienI: source P1_1 : dest Tr1_4
LienI: source Pr1_2 : dest En1_5
-----------------------------------------------------------------
----------------------- BILAN ---------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

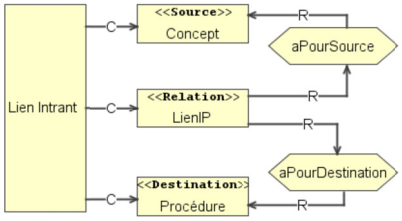
Rapport de validation sémantique

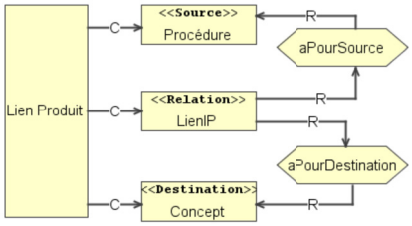
```
----------Début du processus de validation sémantique-------------
C1 est une sorte de (metaDom:MD_Declarative_Concept)
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
Pr1 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
[En1] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En1] est un ( :Pr1 )
[Ex1] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex1] est un ( :C1 )
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :P1 )
------------------Résultats après inférence-----------------------
[En1] est un ( :Pr1 :Type de connaissances :Connaissance stratégique :Agent-Contrainte-Norme )
[Ex1] est un ( :C1 :Type de connaissances :Connaissance déclarative :Connaissance d'objet )
[Tr1] est un ( :P1 :Type de connaissances :Connaissance d'actions :Procédure )
```

B.4 Intrant et le produit

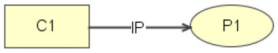
B.4.1 Intrant-produit entre connaissances d'action et d'objet de niveau abstrait

Méta-modèle de la règle d'utilisation

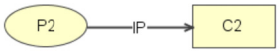




Exemple de représentation en langage MOT



Les observables de C1 sont les *intrants* des observables procédurales de P1



Les observables de C1 sont les *produits* des observables procédurales de P1

Sémantique formelle

```
:C1_0
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:C2_1
      rdf:type owl:Class ;
      rdfs:label "C2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:P1_2
      rdf:type owl:Class ;
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:P1_2_aPourIntrant_C1_0
      rdf:type owl:ObjectProperty ;
      rdfs:domain :P1_2 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :C1_0 ;
      rdfs:subPropertyOf metaDom:A-POUR-INTRANT .
:P2_3
      rdf:type owl:Class ;
      rdfs:label "P2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:P2_3_aPourProduit_C2_1
```

```
      rdf:type owl:ObjectProperty ;
      rdfs:domain :P2_3 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :C2_1 ;
      rdfs:subPropertyOf metaDom:A-POUR-PRODUIT .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-04-a_desamb_Relation_A-POUR-INTRANT]
 [Rule-04-b_desamb_Relation_A-POUR-PRODUIT]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-03-a_Creation_Propriete_A-POUR-INTRANT]
 [Rule-03-b_Creation_Propriete_A-POUR-PRODUIT]
etat_classification.owl
 [Rule-04-a_Ajout_Image_Domaine_A-POUR-INTRANT]
 [Rule-04-b_Ajout_Image_Domaine_A-POUR-PRODUIT]
etat_final.owl
 [Rule-00_Sauvegarde]
etat_validation.owl
```

Rapport de validation syntaxique

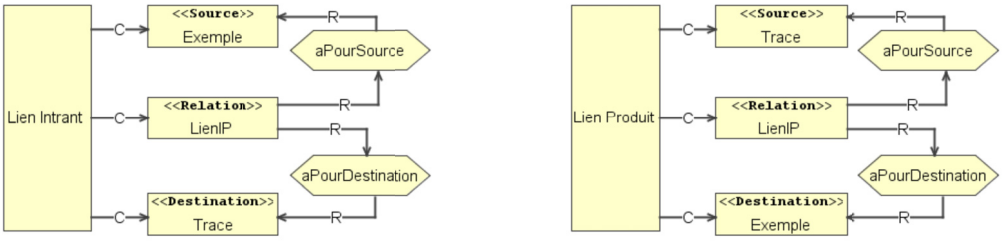
```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/INTRANT_PRODUIT_CONCEPT_PROCEDURE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/INTRANT_PRODUIT_CONCEPT_PROCEDURE_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
-------------------- Les connaissances  ------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Concept   : C1_0
      Concept   : C2_1
      Procédure : P1_2
      Procédure : P2_3
-----------------------------------------------------------------
----------------------- Les relations ---------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienIP: source C1_0 : dest P1_2
      LienIP: source P2_3 : dest C2_1
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
C1 est un intrant de P1
C1 est une sorte de (metaDom:MD_Declarative_Concept)
C2 est une sorte de (metaDom:MD_Declarative_Concept)
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
P2 a pour produit C2
P2 est une sorte de (metaDom:MD_Procedurale_Procedure)
------------------Résultats après inférence-----------------------
```

B.4.2 Intrant-produit entre des connaissances d'action et d'objet observables

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:Ex1_0
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex1"^^xsd:string .
:Ex2_3
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex2"^^xsd:string .
:Tr1_1
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr1"^^xsd:string ;
      metaDom:A-POUR-INTRANT :Ex1_0 .
:Tr2_2
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr2"^^xsd:string ;
      metaDom:A-POUR-PRODUIT  :Ex2_3 .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-05-a_desamb_Relation_A-POUR-INTRANT_observables]
 [Rule-05-b_desamb_Relation_A-POUR-PRODUIT_observables]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
etat_classification.owl
 [Rule-05-a_Creation_propriete_A-POUR-PRODUIT_assertion]
 [Rule-05-b_Creation_propriete_A-POUR-INTRANT_assertion]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/INTRANT_PRODUIT_TRACE_EXEMPLE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/INTRANT_PRODUIT_TRACE_EXEMPLE_rest.mot
Comparaison des documents :
------------------------------------------------------------------
---------------------- Les connaissances  ----------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Exemple   : Ex1_0
```

```
      Exemple   : Ex2_3
      Trace     : Tr1_1
      Trace     : Tr2_2
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienIP: source Ex1_0 : dest Tr1_1
      LienIP: source Tr2_2 : dest Ex2_3
```

```
---------------------- BILAN ---------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

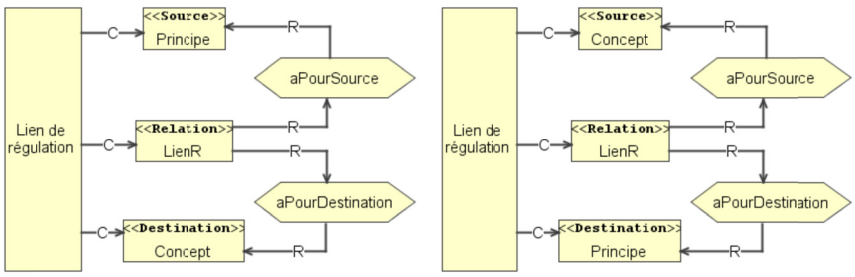
```
----------Début du processus de validation sémantique-------------
[Ex1] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex1] est un ( :Connaissance d'objet )
[Ex2] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex2] est un ( :Connaissance d'objet )
[Tr1] a pour intrant [Ex1]
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :Procédure )
[Tr2] a pour produit [Ex2]
[Tr2] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr2] est un ( :Procédure )

------------------Résultats après inférence-----------------------
[Ex1] est l'intrant de [Tr1]
[Ex1] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex2] est le produit de [Tr2]
[Ex2] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Tr1] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr2] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
```

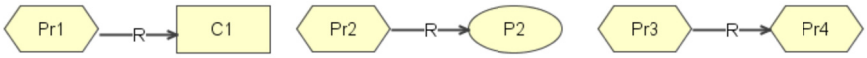
B.5 Régulation

B.5.1 Régulation entre connaissances abstraites

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_1
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:P2_3
      rdf:type owl:Class ;
      rdfs:label "P2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Pr1_0
      rdf:type owl:Class ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Pr1_0_regit_C1_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :C1_1 ;
      rdfs:subPropertyOf metaDom:REGIT .
:Pr2_2
      rdf:type owl:Class ;
      rdfs:label "Pr2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Pr2_2_regit_P2_3
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr2_2 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :P2_3 ;
      rdfs:subPropertyOf metaDom:REGIT .
:Pr3_4
      rdf:type owl:Class ;
      rdfs:label "Pr3"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Pr3_4_regit_Pr4_5
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr3_4 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Pr4_5 ;
      rdfs:subPropertyOf metaDom:REGIT .
:Pr4_5
      rdf:type owl:Class ;
      rdfs:label "Pr4"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-06-a_desamb_LienR_Principe_Concept]
 [Rule-06-b_desamb_LienR_Principe_Procedure]
 [Rule-06-c_desamb_LienR_Principe_Principe]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-d_Creation_class_agent]
 [Rule-06-a_Creation_Propriete_Regit]
 [Rule-06-b_Creation_Prop_Regir_une_procedure]
 [Rule-06-c_Creation_Prop_Regir_un_autre_principe]
etat_classification.owl
 [Rule-06-a_Ajout_ImgDom_Prop_REGIT]
 [Rule-06_b_Ajout_ImgDom_Prop_Regit_Procedure]
 [Rule-06_c_Ajout_ImgDom_Prop_Regit_Principes]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/REGULATION_PRINCIPE_CONCEPT.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/REGULATION_PRINCIPE_CONCEPT_rest.mot
```

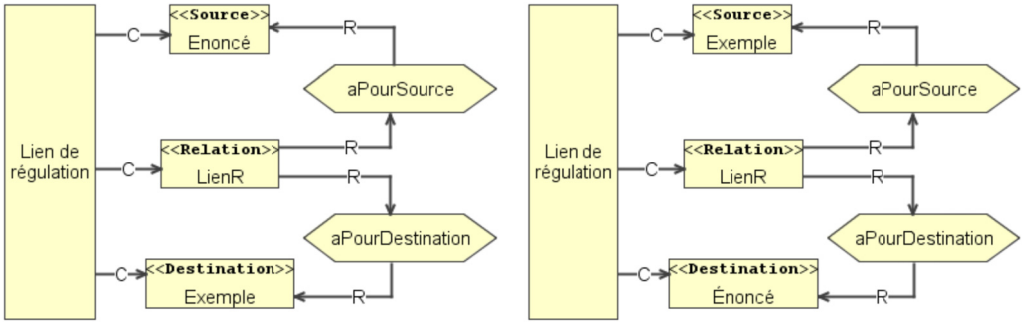
```
Comparaison des documents :
------------------------------------------------------------------
--------------------- Les connaissances  ----------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      Concept   : C1_1
      Principe  : Pr1_0
      Principe  : Pr2_2
      Principe  : Pr3_4
      Principe  : Pr4_5
      Procédure : P2_3
------------------------------------------------------------------
---------------------- Les relations --------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      LienR: source Pr1_0 : dest C1_1
      LienR: source Pr2_2 : dest P2_3
      LienR: source Pr3_4 : dest Pr4_5
------------------------------------------------------------------
---------------------- BILAN -----------------------------------
------------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 4, Reconstruit = 4, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

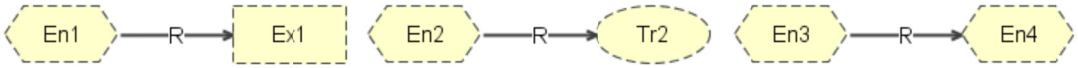
```
----------Début du processus de validation sémantique-------------
C1 est une sorte de (metaDom:MD_Declarative_Concept)
P2 est une sorte de (metaDom:MD_Procedurale_Procedure)
Pr1 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Pr1 régit C1
Pr2 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Pr2 régit P2
Pr3 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Pr3 régit Pr4
Pr4 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
------------------Résultats après inférence-----------------------
```

B.5.2 Interprétation de la régulation entre un énoncé et des observables d'objets, de procédures et de principes.

Métamodèle de l'utilisation de la règle



Exemple de représentation en langage MOT



Sémantique formelle

```
:En1_0
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En1"^^xsd:string ;
      metaDom:REGIT :Ex1_3 .
:En2_1
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En2"^^xsd:string ;
      metaDom:REGIT :Tr2_4 .
:En3_2
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En3"^^xsd:string ;
      metaDom:REGIT :En4_5 .
:En4_5
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En4"^^xsd:string .
:Ex1_3
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex1"^^xsd:string .
:Tr2_4
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr2"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-07-a_desamb_LienR_Enonce_Exemple]
 [Rule-07-b_desamb_LienR_Enonce_Trace]
 [Rule-07-c_desamb_LienR_Enonce_Enonce]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2d_Creation_Individu_Principe_Agent]
etat_classification.owl
 [Rule-07-a_Creation_propriete_regit_entre_enonce_exemple]
 [Rule-07-b_Creation_Regit_enonce_trace]
 [Rule-07-c_Creation_Regit_entre_deux_enonces]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/REGULATION_ENONCE_OBSERVABLES.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/REGULATION_ENONCE_OBSERVABLES_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
--------------------- Les connaissances  ------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Enonce    : En1_0
      Enonce    : En2_1
      Enonce    : En3_2
      Enonce    : En4_5
      Exemple   : Ex1_3
      Trace     : Tr2_4
-----------------------------------------------------------------
----------------------- Les relations -------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienR: source En1_0 : dest Ex1_3
      LienR: source En2_1 : dest Tr2_4
      LienR: source En3_2 : dest En4_5
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
```

```
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 4, Reconstruit = 4, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

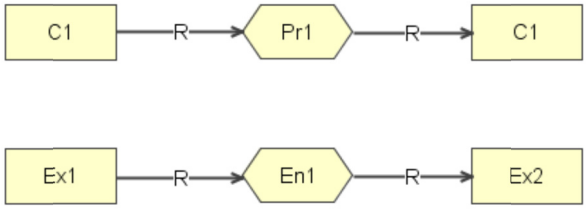
Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
[En1] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En1] est un ( :Agent-Contrainte-Norme )
[En1] régit [Ex1]
[En2] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En2] est un ( :Agent-Contrainte-Norme )
[En2] régit [Tr2]
[En3] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En3] est un ( :Agent-Contrainte-Norme )
[En3] régit [En4]
[En4] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En4] est un ( :Agent-Contrainte-Norme )
[Ex1] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex1] est un ( :Connaissance d'objet )
[Tr2] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr2] est un ( :Procédure )
------------------Résultats après inférence----------------------
[En1] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En2] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En3] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En4] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En4] obéit à [En3]
[Ex1] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex1] obéit à [En1]
[Tr2] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr2] obéit à [En2]
```

B.6 Propriété

B.6.1 Définition d'un domaine et d'une image à une propriété entre des objets

Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_0
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:C1_0_Pr1_1_C2_2
      rdf:type owl:ObjectProperty ;
      rdfs:domain :C1_0 ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:range :C2_2 ;
      rdfs:subPropertyOf :Pr1_1 .
:C2_2
      rdf:type owl:Class ;
```

```
      rdfs:label "C2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:En1_4
      rdf:type owl:ObjectProperty ;
      rdfs:label "En1"^^xsd:string ;
      rdfs:subPropertyOf metaDom:MD_ASSERTION .
:Ex1_3
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex1"^^xsd:string ;
      :En1_4  :Ex2_5 .
:Ex2_5
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex2"^^xsd:string .
:Pr1_1
      rdf:type owl:ObjectProperty ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subPropertyOf metaDom:MD_PROPRIETE .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
 [Rule-08-a_desamb_Principe_entre_concept]
 [Rule-08-b_desamb_Enonces_entre_exemple]
regles_desambig_typologique.owl
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2-5k_Creation_AssertionDeDomaine]
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-k_Creation_ProprieteDeDomaine]
 [Rule-08_Creation_dune_propriete_objet_DECLARATIF]
etat_classification.owl
 [Rule-08-a_Ajout_ImgDom_a_une_propriete_objet]
 [Rule-08-b_Creation_propriete_entre_deux_individus]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/PROPRIETE-ENTRE-OBJETS.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/PROPRIETE-ENTRE-OBJETS_rest.mot
Comparaison des documents :
------------------------------------------------------------------
---------------------- Les connaissances  ------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      Concept   : C1_0
      Concept   : C2_2
      Enonce    : En1_4
      Exemple   : Ex1_3
      Exemple   : Ex2_5
      Principe  : Pr1_1
------------------------------------------------------------------
------------------------ Les relations ---------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      LienR: source C1_0 : dest Pr1_1
      LienR: source En1_4 : dest Ex2_5
      LienR: source Ex1_3 : dest En1_4
      LienR: source Pr1_1 : dest C2_2
------------------------------------------------------------------
----------------------- BILAN ------------------------------------
------------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

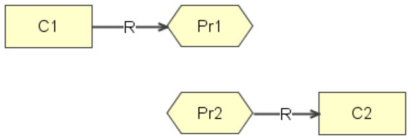
Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
C1 Pr1 C2
C1 est une sorte de (metaDom:MD_Declarative_Concept)
C2 est une sorte de (metaDom:MD_Declarative_Concept)
En1_4([],[]) est une assertion de la propriété (metaDom:MD_ASSERTION)
[Ex1] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex1] est un ( :Connaissance d'objet )
[Ex2] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex2] est un ( :Connaissance d'objet )

------------------Résultats après inférence-----------------------
[Ex1] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex2] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
```

B.6.2 Propriétés unaires

B.6.3 Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_2
      rdf:type metaMot:MOT_Concept ;
      metaMot:MOT_estNonDetermine  "false"^^xsd:boolean ;
      metaMot:MOT_etiquette        "C1"^^xsd:string ;
      metaMot:MOT_lrEstLaSource    :LienR_C1_2_Pr1_0 ;
      metaMot:MOT_nomConnaissance  "C1_2"^^xsd:string ;
      ot:OT_EntiteEstDeType        oAmbig:Concept_Classe .
:C2_3
      rdf:type metaMot:MOT_Concept ;
      metaMot:MOT_estNonDetermine   "false"^^xsd:boolean ;
      metaMot:MOT_etiquette         "C2"^^xsd:string ;
      metaMot:MOT_lrEstLaDestination :LienR_Pr2_1_C2_3 ;
      metaMot:MOT_nomConnaissance    "C2_3"^^xsd:string ;
      ot:OT_EntiteEstDeType          oAmbig:Concept_Classe .
:LienR_C1_2_Pr1_0
      rdf:type metaMot:MOT_LienR ;
      metaMot:MOT_estNonDetermine     "false"^^xsd:boolean ;
      metaMot:MOT_lrConnDest          :Pr1_0 ;
      metaMot:MOT_lrConnSrc           :C1_2 ;
      metaMot:MOT_nomLien      "LienR_C1_2_Pr1_0"^^xsd:string ;
      ot:OT_EntiteEstDeType    oAmbig:Relation_Regulation .

:LienR_Pr2_1_C2_3
      rdf:type metaMot:MOT_LienR ;
      metaMot:MOT_estNonDetermine    "false"^^xsd:boolean ;
      metaMot:MOT_lrConnDest        :C2_3 ;
      metaMot:MOT_lrConnSrc         :Pr2_1 ;
      metaMot:MOT_nomLien "LienR_Pr2_1_C2_3"^^xsd:string ;
      ot:OT_EntiteEstDeType     oAmbig:Relation_Regulation .
:Pr1_0
      rdf:type metaMot:MOT_Principe ;
      metaMot:MOT_estNonDetermine   "false"^^xsd:boolean ;
      metaMot:MOT_etiquette         "Pr1"^^xsd:string ;
      metaMot:MOT_lrEstLaDestination  :LienR_C1_2_Pr1_0 ;
      metaMot:MOT_nomConnaissance     "Pr1_0"^^xsd:string ;
      ot:OT_EntiteEstDeType     oAmbig:Propriete_Unaire ;
      ot:OT_estLaDestinationDuneRegulation  "true"^^xsd:boolean ;
      ot:OT_nomEntiteSource    "C1_2"^^xsd:string .
:Pr2_1
      rdf:type metaMot:MOT_Principe ;
      metaMot:MOT_estNonDetermine   "false"^^xsd:boolean ;
      metaMot:MOT_etiquette         "Pr2"^^xsd:string ;
      metaMot:MOT_lrEstLaSource     :LienR_Pr2_1_C2_3 ;
      metaMot:MOT_nomConnaissance   "Pr2_1"^^xsd:string ;
      ot:OT_EntiteEstDeType         oAmbig:Propriete_Unaire ;
      ot:OT_estLaSourceDuneRegulation "true"^^xsd:boolean ;
      ot:OT_nomEntiteDestination  "C2_3"^^xsd:string .
```

Règles de désambiguïsation

Une désambiguïsation manuelle doit être réalisée pour que ceux-ci soient interprétés en tant que propriété

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-09-a_Creation_SuperPropriete_unaire]
 [Rule-09-b_Creation_propriete_unaire_Image]
 [Rule-09-c_Creation_propriete_unaire_Domaine]
etat_classification.owl
 [Rule-09-a_Ajout_Image_Propriete_Unaire]
 [Rule-09-b_Ajout_Domaine_Propriete_Unaire]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/PROPRIÉTE_UNAIRE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/PROPRIÉTE_UNAIRE_rest.mot
Comparaison des documents :
------------------------------------------------------------------
-------------------- Les connaissances  ----------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Concept   : C1_2
      Concept   : C2_3
      Principe  : Pr1_0
      Principe  : Pr2_1
------------------------------------------------------------------
---------------------- Les relations --------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienR: source C1_2 : dest Pr1_0
      LienR: source Pr2_1 : dest C2_3
------------------------------------------------------------------
---------------------- BILAN ---------------------------------
------------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

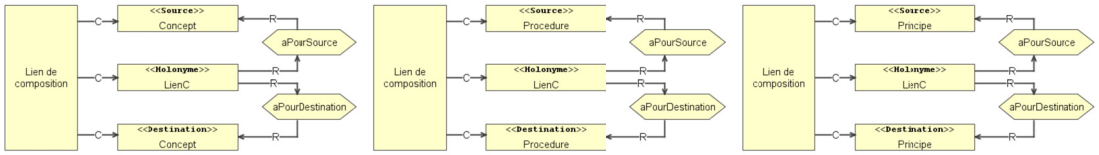
Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
C1 est une sorte de (metaDom:MD_Declarative_Concept)
C2 est une sorte de (metaDom:MD_Declarative_Concept)
------------------Résultats après inférence-----------------------
```

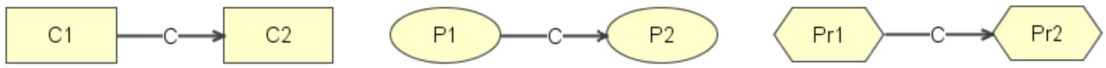
B.7 Composition entre connaissances

B.7.1 Holonyme entre deux abstractions

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_0
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:C1_0_aPourComposant_C2_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :C1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :C2_1 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:C2_1
      rdf:type owl:Class ;
      rdfs:label "C2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:P1_2
      rdf:type owl:Class ;
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:P1_2_aPourComposant_P2_3
      rdf:type owl:ObjectProperty ;
      rdfs:domain :P1_2 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :P2_3 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:P2_3
      rdf:type owl:Class ;
      rdfs:label "P2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Pr1_4
      rdf:type owl:Class ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:Pr1_4_aPourComposant_Pr2_5
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr1_4 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Pr2_5 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Pr2_5
      rdf:type owl:Class ;
      rdfs:label "Pr2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Condition .
```

Règles de désambiguïsation

Le lien de composition entre les connaissances déclaratives doit être désambiguïsé de manière manuelle.

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-10-a_holonyme_entre_deux_concepts]
 [Rule-10-b_holonyme_entre_deux_procedures]
 [Rule-10-c_holonyme_entre_un_agents_condition]
 [Rule-16-a_Desamb_LienC_Entre_Principe-et-inconnu]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-d_Creation_class_agent]
 [Rule-01-e_Creation_class_condition]
 [Rule-10-a_Creation_Propriete_aPourComposant]
 [Rule-10-b_Creation_Propriete_aPourComposant_Procedurale]
 [Rule-10-d_Creation_Propriete_aPourComposant_Strategique_Condition]
etat_classification.owl
 [Rule-10-a_Ajoute_Image_Domaine_a_la_Propriete_aPourComposant]
 [Rule-10-b_Ajoute_Image_Domaine_a_la_Propriete_aPourComposant_ConnProcedurale]
 [Rule-10-d_Ajoute_ImgDom_aPourComposant_Agent_Condition]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

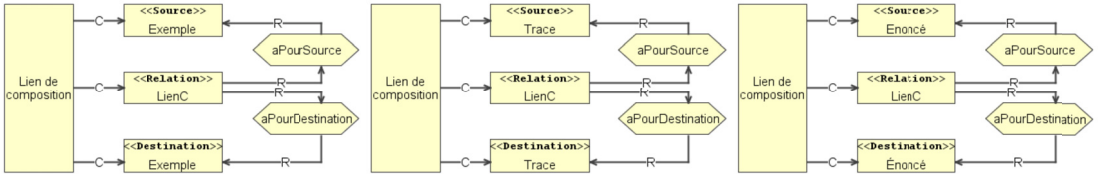
```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/COMPOSITION_ABSTRACTION.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/COMPOSITION_ABSTRACTION_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
---------------------- Les connaissances  ------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      Concept   : C1_0
      Concept   : C2_1
      Principe  : Pr1_4
      Principe  : Pr2_5
      Procédure : P1_2
      Procédure : P2_3
-----------------------------------------------------------------
----------------------- Les relations ---------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      LienC: source C1_0 : dest C2_1
      LienC: source P1_2 : dest P2_3
      LienC: source Pr1_4 : dest Pr2_5
-----------------------------------------------------------------
----------------------- BILAN -----------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

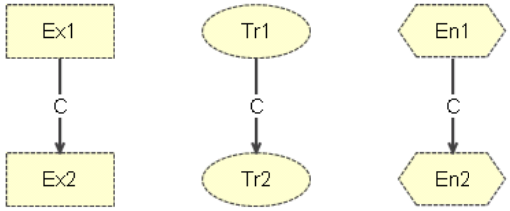
```
----------Début du processus de validation sémantique-------------
C1 est une sorte de (metaDom:MD_Declarative_Concept)
C1 se compose de C2
C2 est une sorte de (metaDom:MD_Declarative_Concept)
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
P1 se compose de P2
P2 est une sorte de (metaDom:MD_Procedurale_Procedure)
Pr1 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
Pr1 se compose de Pr2
Pr2 est une sorte de (metaDom:MD_Strategique_Condition)
------------------Résultats après inférence-----------------------
```

B.7.2 Composition entre deux connaissances observables

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:En1_4
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En1"^^xsd:string ;
      metaDom:A-POUR-COMPOSANT   :En2_5 .
:En2_5
      rdf:type metaDom:MD_Strategique_Condition ;
      rdfs:label "En2"^^xsd:string .
:Ex1_0
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex1"^^xsd:string ;
      metaDom:A-POUR-COMPOSANT   :Ex2_1 .
:Ex2_1
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex2"^^xsd:string .
:Tr1_2
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr1"^^xsd:string ;
      metaDom:A-POUR-COMPOSANT  :Tr2_3 .
:Tr2_3
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr2"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-11-a_holonyme_entre_deux_enonces_Agents_Condition]
 [Rule-11-b_holonyme_entre_deux_traces]
 [Rule-11-c_holonyme_entre_deux_exemples]
 [Rule-17-b_LienC_entre_deux_enonces]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2d_Creation_Individu_Principe_Agent]
 [Rule-01-2i_Creation_Individu_Principe_Condition]
etat_classification.owl
 [Rule-11_Creation_propriete_aPourComposant_individu]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/HOLONYME_ENTRE_OBSERVABLE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/HOLONYME_ENTRE_OBSERVABLE_rest.mot
Comparaison des documents :
---------------------------------------------------------------
```

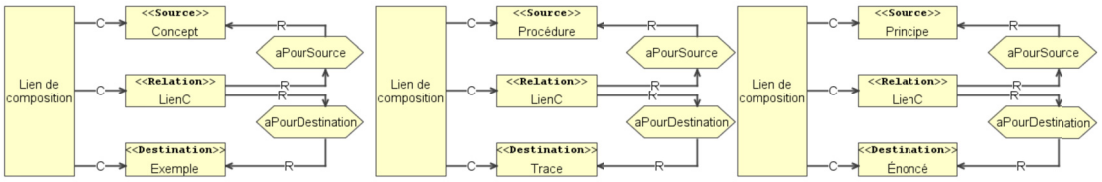
```
-------------------- Les connaissances  ----------------------
-------------------------------------------------------------
------------------ Commun aux deux modèles ------------------
      Enonce    : En1_4
      Enonce    : En2_5
      Exemple   : Ex1_0
      Exemple   : Ex2_1
      Trace     : Tr1_2
      Trace     : Tr2_3
-------------------------------------------------------------
----------------------- Les relations -----------------------
-------------------------------------------------------------
------------------ Commun aux deux modèles ------------------
      LienC: source En1_4 : dest En2_5
      LienC: source Ex1_0 : dest Ex2_1
      LienC: source Tr1_2 : dest Tr2_3
-------------------------------------------------------------
----------------------- BILAN -------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles -----------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 2, Reconstruit = 2, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

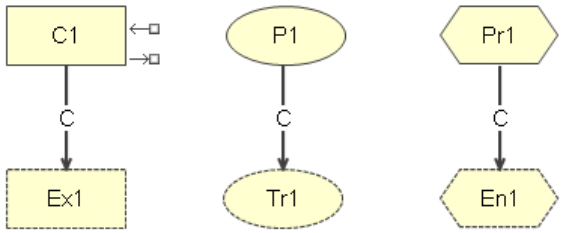
```
----------Début du processus de validation sémantique-------------
[En1] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En1] est un ( :Agent-Contrainte-Norme )
[En1] se compose de [En2]
[En2] est de catégorie (metaDom:MD_Strategique_Condition)
[En2] est un ( :Condition )
[Ex1] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex1] est un ( :Connaissance d'objet )
[Ex1] se compose de [Ex2]
[Ex2] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex2] est un ( :Connaissance d'objet )
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :Procédure )
[Tr1] se compose de [Tr2]
[Tr2] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr2] est un ( :Procédure )
-----------------Résultats après inférence----------------------
[En1] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En2] est un ( :Condition :Type de connaissances :Connaissance stratégique )
[En2] est une partie de [En1]
[Ex1] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex2] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex2] est une partie de [Ex1]
[Tr1] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr2] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr2] est une partie de [Tr1]
```

B.7.3 Composition entre des connaissances de niveau d'abstraction différent

Méta-modèle de la règle d'utilisation



Exemple de représentation en langage MOT



Sémantique formelle

```
:C1_0
      rdf:type owl:Class ;
      rdfs:label "C1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept ;
      metaDom:A-POUR-COMPOSANT  :Ex1_3 .
:En1_5
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En1"^^xsd:string .
:Ex1_3
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex1"^^xsd:string .
:P1_1
      rdf:type owl:Class ;
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing ;
      metaDom:A-POUR-COMPOSANT    :Tr1_4 .
:Pr1_2
      rdf:type owl:Class ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme ;
      metaDom:A-POUR-COMPOSANT      :En1_5 .
:Tr1_4
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr1"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-16-a_Desamb_LienC_Entre_Principe-et-inconnu]
 [Rule-XX-05-2-3_holonyme_concept_exemple]
 [Rule-XX-06-2-3_holonyme_procedure_trace]
 [Rule-XX-07-2-3_holonyme_principe_enonce]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2d_Creation_Individu_Principe_Agent]
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-d_Creation_class_agent]
etat_classification.owl
 [Rule-12-a_Creation_propriete_aPourComposant_entre_Concept_Exemple]
 [Rule-12-b_Creation_propriete_aPourComposant_entre_Procedure_Trace]
 [Rule-12-c_Creation_propriete_aPourComposant_entre_principe_enonce]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/HOLONYME_ABS_OBS.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/HOLONYME_ABS_OBS_rest.mot
Comparaison des documents :
------------------------------------------------------------------
---------------------- Les connaissances  ------------------------
------------------------------------------------------------------
-------------------- Commun aux deux modèles ---------------------
      Concept   : C1_0
      Enonce    : En1_5
      Exemple   : Ex1_3
```

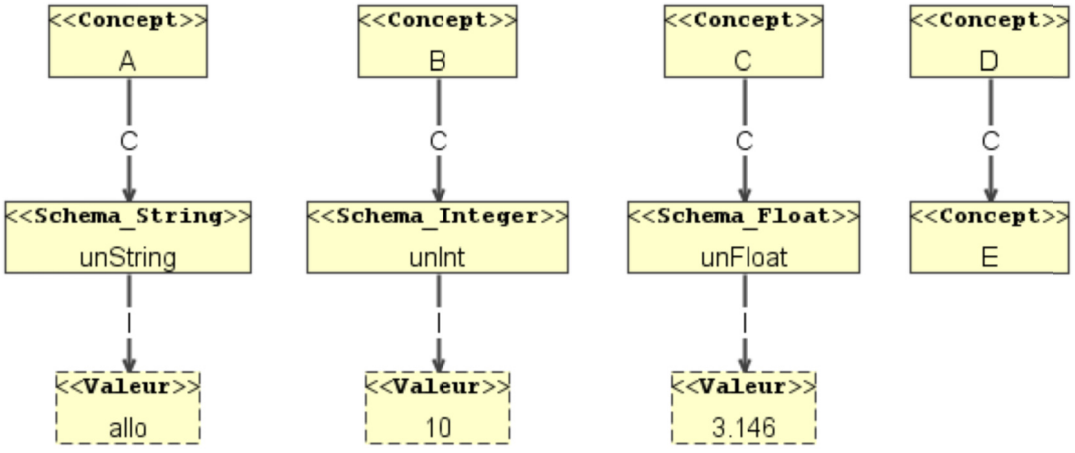
```
      Principe  : Pr1_2
      Procédure : P1_1
      Trace     : Tr1_4
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienC: source C1_0 : dest Ex1_3
      LienC: source P1_1 : dest Tr1_4
      LienC: source Pr1_2 : dest En1_5
-----------------------------------------------------------------
----------------------- BILAN -----------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
C1 est une sorte de (metaDom:MD_Declarative_Concept)
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
Pr1 est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
[En1] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En1] est un ( :Agent-Contrainte-Norme )
[Ex1] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex1] est un ( :Connaissance d'objet )
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :Procédure )
------------------Résultats après inférence-----------------------
[En1] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En1] est une partie de [Pr1]
[Ex1] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex1] est une partie de [C1]
[Tr1] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr1] est une partie de [P1]
```

B.8 Attribut de connaissances déclaratives

Exemple de représentation en langage MOT



Sémantique formelle

```
:A_6  rdf:type owl:Class ;
      rdfs:label "A"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:A_6_aPourAttribut_unString_0
      rdf:type owl:ObjectProperty ;
      rdfs:domain :A_6 ;
      rdfs:label "unString"^^xsd:string ;
      rdfs:range :unString_0 ;
      rdfs:subPropertyOf metaDom:A-POUR-ATTRIBUT .
:B_7  rdf:type owl:Class ;
      rdfs:label "B"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:B_7_aPourAttribut_unInt_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :B_7 ;
      rdfs:label "unInt"^^xsd:string ;
      rdfs:range :unInt_1 ;
      rdfs:subPropertyOf metaDom:A-POUR-ATTRIBUT .
:C_8  rdf:type owl:Class ;
      rdfs:label "C"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:C_8_aPourAttribut_unFloat_2
      rdf:type owl:ObjectProperty ;
      rdfs:domain :C_8 ;
      rdfs:label "unFloat"^^xsd:string ;
      rdfs:range :unFloat_2 ;
      rdfs:subPropertyOf metaDom:A-POUR-ATTRIBUT .
:D_9  rdf:type owl:Class ;
      rdfs:label "D"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:D_9_aPourAttribut_E_10
      rdf:type owl:ObjectProperty ;
      rdfs:domain :D_9 ;
      rdfs:label "E"^^xsd:string ;
      rdfs:range :E_10 ;
      rdfs:subPropertyOf metaDom:A-POUR-ATTRIBUT .
:E_10
      rdf:type owl:Class ;
      rdfs:label "E"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:_10_4
      rdf:type :unInt_1 ;
      rdfs:label "10"^^xsd:string .
:_3-146_5
      rdf:type :unFloat_2 ;
      rdfs:label "3.146"^^xsd:string .
:allo_3
      rdf:type :unString_0 ;
      rdfs:label "allo"^^xsd:string .
:unFloat_2
      rdf:type owl:Class ;
      rdfs:label "unFloat"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_Float ;
      metaDom:MD_isString "3.146"^^xsd:float .
:unInt_1
      rdf:type owl:Class ;
      rdfs:label "unInt"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_Int ;
      metaDom:MD_isString "10"^^xsd:int .
:unString_0
      rdf:type owl:Class ;
      rdfs:label "unString"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Schema_String ;
      metaDom:MD_isString "allo"^^xsd:string .
```

Règles de désambiguïsation

Tous les éléments doivent êtres désambiguïsés de manière manuelle

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-13-a_Creation_Propriete_aPourAttribut_Schema]
 [Rule-13-b_Creation_class_Schema_Int]
 [Rule-13-c_Creation_class_Schema_Float]
 [Rule-13-d_Creation_class_Schema_String]
 [Rule-13-e-Creation_Dun_Individu_dattribut]
```

```
 [Rule-13-f_Creation_Propriete_aPourAttribut]
etat_classification.owl
 [Rule-01-d_Creation_subClassOf_SchemaString]
 [Rule-01-e_Creation_subClassOf_SchemaInt]
 [Rule-01-f_Creation_subClassOf_SchemaFloat]
 [Rule-13-a_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut]
 [Rule-13-e_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut_Int]
 [Rule-13-f_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut_Float]
 [Rule-13-g_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut_String]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/13_CompositionAPourAttribut.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/13_CompositionAPourAttribut_rest.mot
Comparaison des documents :
----------------------------------------------------------------
---------------------- Les connaissances  ----------------------
----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Concept   : A_6
      Concept   : B_7
      Concept   : C_8
      Concept   : D_9
      Concept   : E_10
      Concept   : unFloat_2
      Concept   : unInt_1
      Concept   : unString_0
      Exemple   : _10_4
      Exemple   : _3-146_5
      Exemple   : allo_3
----------------------------------------------------------------
---------------------- Les relations -------------------------
----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienC: source A_6 : dest unString_0
      LienC: source B_7 : dest unInt_1
      LienC: source C_8 : dest unFloat_2
      LienC: source D_9 : dest E_10
      LienI: source unFloat_2 : dest _3-146_5
      LienI: source unInt_1 : dest _10_4
      LienI: source unString_0 : dest allo_3
----------------------------------------------------------------
---------------------- BILAN ---------------------------------
----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 8, Reconstruit = 8, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

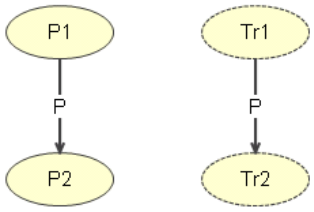
Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
A a pour attribut unString
A est une sorte de (metaDom:MD_Declarative_Concept)
B a pour attribut unInt
B est une sorte de (metaDom:MD_Declarative_Concept)
C a pour attribut unFloat
C est une sorte de (metaDom:MD_Declarative_Concept)
D a pour attribut E
D est une sorte de (metaDom:MD_Declarative_Concept)
E est une sorte de (metaDom:MD_Declarative_Concept)
[10] est de catégorie (metaDom:MD_Declarative_Schema_Int)
[10] est un ( :unInt )
[3.146] est de catégorie (metaDom:MD_Declarative_Schema_Float)
[3.146] est un ( :unFloat )
[allo] est de catégorie (metaDom:MD_Declarative_Schema_String)
[allo] est un ( :unString )
unFloat est une sorte de (metaDom:MD_Declarative_Schema_Float)
unInt est une sorte de (metaDom:MD_Declarative_Schema_Int)
unString est une sorte de (metaDom:MD_Declarative_Schema_String)
------------------Résultats après inférence-----------------------
!!! Erreur :Cannot do reasoning with inconsistent ontologies!
```

B.9 Lien de précédence

B.9.1 Relation de précédence entre connaissances d'actions

Exemple de représentation en langage MOT



Sémantique formelle

```
:P1_0
      rdf:type owl:Class ;
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:P1_0_puisExecuter_P2_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :P1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :P2_1 ;
      rdfs:subPropertyOf metaDom:PUIS-EXECUTER .
:P2_1
      rdf:type owl:Class ;
      rdfs:label "P2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Tr1_2
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr1"^^xsd:string ;
      metaDom:PUIS-EXECUTER
              :Tr2_3 .
:Tr2_3
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr2"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-00_ConnaissancesNonAmbigue_LienP]
 [Rule-14-a_Procedure-P-Procedure_Procedure-Procedure]
 [Rule-14-b_Trace-P-Trace_Procedure-Procedure]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-14_Creation_Propriete_puisExecuter_Procedure_Procedure]
etat_classification.owl
 [Rule-14-a_Ajout_ImgDom_Prop_puisExecuter_Procedure_Procedure]
 [Rule-14-b_Creation_Individu_Procedural_puisExecuter_Procedural]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE_ENTRE_ACTION.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE_ENTRE_ACTION_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
-------------------- Les connaissances  ------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Procédure : P1_0
```

```
      Procédure : P2_1
      Trace     : Tr1_2
      Trace     : Tr2_3
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienP: source P1_0 : dest P2_1
      LienP: source Tr1_2 : dest Tr2_3
-----------------------------------------------------------------
----------------------- BILAN -----------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

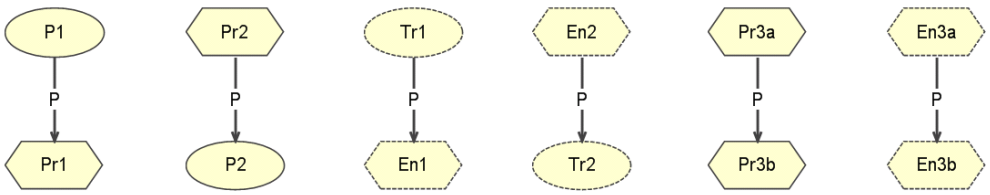
Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
P1 puis exécuter P2
P2 est une sorte de (metaDom:MD_Procedurale_Procedure)
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :Procédure )
[Tr1] puis exécuter [Tr2]
[Tr2] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr2] est un ( :Procédure )

------------------Résultats après inférence-----------------------
[Tr1] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr1] permet [Tr2]
[Tr2] a pour dépendance [Tr1]
[Tr2] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
```

B.9.2 Précédence entre connaissances d'actions et stratégique

Exemple de représentation en langage MOT



Sémantique formelle

```
:En1_6
      rdf:type metaDom:MD_Strategique_Condition ;
      rdfs:label "En1"^^xsd:string .
:En2_7
      rdf:type metaDom:MD_Strategique_Condition ;
      rdfs:label "En2"^^xsd:string ;
      metaDom:PUIS-EXECUTER   :Tr2_5 .
:En3a_10
      rdf:type metaDom:MD_Strategique_Condition ;
      rdfs:label "En3a"^^xsd:string ;
      metaDom:PUIS-EVALUER    :En3b_11 .
:En3b_11
      rdf:type metaDom:MD_Strategique_Condition ;
```

```
      rdfs:label "En3b"^^xsd:string .
:P1_0
      rdf:type owl:Class ;
      rdfs:label "P1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:P1_0_puisEvaluer_Pr1_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :P1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Pr1_1 ;
      rdfs:subPropertyOf metaDom:PUIS-EVALUER .
:P2_3
      rdf:type owl:Class ;
      rdfs:label "P2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Pr1_1
      rdf:type owl:Class ;
      rdfs:label "Pr1"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Condition .
:Pr2_2
      rdf:type owl:Class ;
      rdfs:label "Pr2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Condition .
:Pr2_2_puisExecuter_P2_3
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr2_2 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :P2_3 ;
      rdfs:subPropertyOf metaDom:PUIS-EXECUTER .
:Pr3a_8
      rdf:type owl:Class ;
      rdfs:label "Pr3a"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Condition .
:Pr3a_8_puisEvaluer_Pr3b_9
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr3a_8 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Pr3b_9 ;
      rdfs:subPropertyOf metaDom:PUIS-EVALUER .
:Pr3b_9
      rdf:type owl:Class ;
      rdfs:label "Pr3b"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Condition .
:Tr1_4
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr1"^^xsd:string ;
      metaDom:PUIS-EVALUER   :En1_6 .
:Tr2_5
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr2"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
 [Rule-00_ConnaissancesNonAmbigue_LienP]
 [Rule-15-a_Procedure-P-Principe_Procedure-Condition]
 [Rule-15-b_Principe-P-Procedure_Condition-Procedure]
 [Rule-15-c_Principe-P-Principe_Condition-Condition]
 [Rule-15-c_Trace-P-Enonce_Procedure-Condition]
 [Rule-15-d_Enonce-P-Enonce_Condition-Condition]
 [Rule-15-e_Enonce-P-Trace_Condition-Procedure]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2i_Creation_Individu_Principe_Condition]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-e_Creation_class_condition]
 [Rule-15-a_Creation_Propriete_puisEvaluer_Procedure_Condition]
 [Rule-15-b_Creation_Propriete_puisExecuter_Condition_Procedure]
 [Rule-15-c_Creation_Propriete_puisEvaluer_Condition_Condition]
etat_classification.owl
 [Rule-15-a_Ajout_ImgDom_Prop_puisEvaluer_Procedure_Condition]
 [Rule-15-b_Ajout_ImgDom_Prop_puisExecuter_Condition_Procedure]
 [Rule-15-c_Ajout_ImgDom_Prop_puisEvaluer_Condition_Condition]
 [Rule-15-d_Creation_Individu_Procedural_puisEvaluer_Condition]
 [Rule-15-e_Creation_Individu_Condition_puisExecuter_Procedural]
 [Rule-15-f_Creation_Individu_Condition_puisEvaluer_Condition]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE_ACTION_STRATEGIQUE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE_ACTION_STRATEGIQUE_rest.mot
Comparaison des documents :
----------------------------------------------------------------
--------------------- Les connaissances  ------------------------
----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Enonce    : En1_6
      Enonce    : En2_7
      Enonce    : En3a_10
      Enonce    : En3b_11
      Principe  : Pr1_1
      Principe  : Pr2_2
      Principe  : Pr3a_8
      Principe  : Pr3b_9
      Procédure : P1_0
      Procédure : P2_3
      Trace     : Tr1_4
      Trace     : Tr2_5
----------------------------------------------------------------
---------------------- Les relations -------------------------
----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienP: source En2_7 : dest Tr2_5
      LienP: source En3a_10 : dest En3b_11
      LienP: source P1_0 : dest Pr1_1
      LienP: source Pr2_2 : dest P2_3
      LienP: source Pr3a_8 : dest Pr3b_9
      LienP: source Tr1_4 : dest En1_6
----------------------------------------------------------------
----------------------- BILAN ------------------------------------
----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 4, Reconstruit = 4, Différence = 0
Compte des Enonces   : Original = 4, Reconstruit = 4, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
P1 est une sorte de (metaDom:MD_Procedurale_Procedure)
P1 puis évaluer Pr1
P2 est une sorte de (metaDom:MD_Procedurale_Procedure)
Pr1 est une sorte de (metaDom:MD_Strategique_Condition)
Pr2 est une sorte de (metaDom:MD_Strategique_Condition)
Pr2 puis exécuter P2
Pr3a est une sorte de (metaDom:MD_Strategique_Condition)
Pr3a puis évaluer Pr3b
Pr3b est une sorte de (metaDom:MD_Strategique_Condition)
[En1] est de catégorie (metaDom:MD_Strategique_Condition)
[En1] est un ( :Condition )
[En2] est de catégorie (metaDom:MD_Strategique_Condition)
[En2] est un ( :Condition )
[En2] puis exécuter [Tr2]
[En3a] est de catégorie (metaDom:MD_Strategique_Condition)
[En3a] est un ( :Condition )
[En3a] puis évaluer [En3b]
[En3b] est de catégorie (metaDom:MD_Strategique_Condition)
[En3b] est un ( :Condition )
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :Procédure )
[Tr1] puis évaluer [En1]
[Tr2] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr2] est un ( :Procédure )

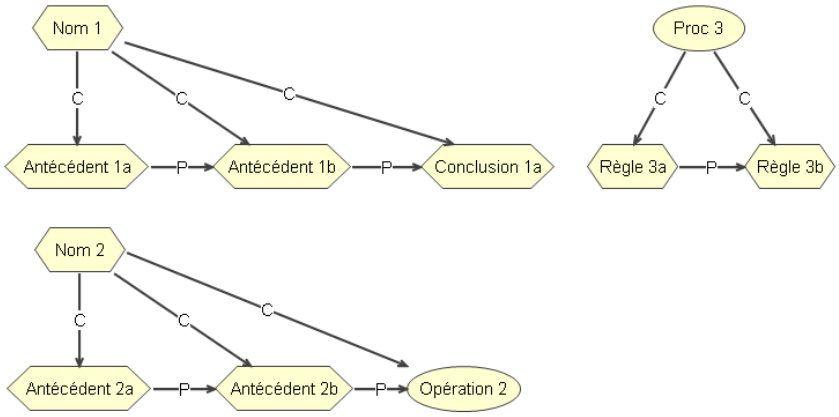
------------------Résultats après inférence-----------------------
[En1] a pour dépendance [Tr1]
[En1] est un ( :Condition :Type de connaissances :Connaissance stratégique )
[En1] évalué à partir de [Tr1]
[En2] est un ( :Condition :Type de connaissances :Connaissance stratégique )
[En2] permet [Tr2]
[En3a] est un ( :Condition :Type de connaissances :Connaissance stratégique )
[En3a] permet [En3b]
```

```
[En3b] a pour dépendance [En3a]
[En3b] est un ( :Condition :Type de connaissances :Connaissance stratégique )
[En3b] évalué à partir de [En3a]
[Tr1] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr1] permet [En1]
[Tr2] a pour dépendance [En2]
[Tr2] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
```

B.10 Règle

B.10.1 Règle à partir de connaissances abstraites

Exemple de représentation en langage MOT



Sémantique formelle

```
:Antecedent-1a_1
      rdf:type owl:Class ;
      rdfs:label "AntÃ©cÃ©dent 1a"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent , owl:Thing .
:Antecedent-1a_1_estAntecedentDe_Antecedent-1b_2
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Antecedent-1a_1 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Antecedent-1b_2 ;
      rdfs:subPropertyOf metaDom:EST-ANTECEDENT-DE .
:Antecedent-1b_2
      rdf:type owl:Class ;
      rdfs:label "AntÃ©cÃ©dent 1b"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent , owl:Thing .
:Antecedent-1b_2_aPourConclusion_Conclusion-1a_3
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Antecedent-1b_2 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Conclusion-1a_3 ;
      rdfs:subPropertyOf metaDom:A-POUR-CONCLUSION .
:Antecedent-2a_4
      rdf:type owl:Class ;
      rdfs:label "AntÃ©cÃ©dent 2a"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent , owl:Thing .
:Antecedent-2a_4_estAntecedentDe_Antecedent-2b_6
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Antecedent-2a_4 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Antecedent-2b_6 ;
      rdfs:subPropertyOf metaDom:EST-ANTECEDENT-DE .
:Antecedent-2b_6
      rdf:type owl:Class ;
      rdfs:label "AntÃ©cÃ©dent 2b"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Antecedent , owl:Thing .
:Antecedent-2b_6_alors_Operation-2_7
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Antecedent-2b_6 ;
```

```
      rdfs:label ""^^xsd:string ;
      rdfs:range :Operation-2_7 ;
      rdfs:subPropertyOf metaDom:ALORS .
:Conclusion-1a_3
      rdf:type owl:Class ;
      rdfs:label "Conclusion 1a"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_Entite_Regle_Conclusion .
:Nom-1_0
      rdf:type owl:Class ;
      rdfs:label "Nom 1"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Nom , owl:Thing .
:Nom-1_0_aPourComposant_Antecedent-1a_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Nom-1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Antecedent-1a_1 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Nom-1_0_aPourComposant_Antecedent-1b_2
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Nom-1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Antecedent-1b_2 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Nom-1_0_aPourComposant_Conclusion-1a_3
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Nom-1_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Conclusion-1a_3 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Nom-2_5
      rdf:type owl:Class ;
      rdfs:label "Nom 2"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Nom , owl:Thing .
:Nom-2_5_aPourComposant_Antecedent-2a_4
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Nom-2_5 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Antecedent-2a_4 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Nom-2_5_aPourComposant_Antecedent-2b_6
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Nom-2_5 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Antecedent-2b_6 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Nom-2_5_aPourComposant_Operation-2_7
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Nom-2_5 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Operation-2_7 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Operation-2_7
      rdf:type owl:Class ;
      rdfs:label "OpÃ©ration 2"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Procedurale_Operation .
:Proc-3_8
      rdf:type owl:Class ;
      rdfs:label "Proc 3"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Proc-3_8_aPourComposant_Regle-3a_9
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Proc-3_8 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Regle-3a_9 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Proc-3_8_aPourComposant_Regle-3b_10
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Proc-3_8 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Regle-3b_10 ;
      rdfs:subPropertyOf metaDom:A-POUR-COMPOSANT .
:Regle-3a_9
      rdf:type owl:Class ;
      rdfs:label "RÃ¨gle 3a"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Complete , owl:Thing .
:Regle-3a_9_estPrecedentDe_Regle-3b_10
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Regle-3a_9 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :Regle-3b_10 ;
      rdfs:subPropertyOf metaDom:EST-PRECEDENT-DE .
:Regle-3b_10
      rdf:type owl:Class ;
      rdfs:label "RÃ¨gle 3b"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Strategique_Entite_Regle_Complete , owl:Thing .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
 [Rule-16-a_RegleNom-Antecedent_Principe-Principe-Principe]
 [Rule-16-b_RegleNom-Antecedent_Principe-Principe-Procedure]
 [Rule-19-c_Procedure-LienC-Principe-Procedure_RegleComplete]
regles_desambig_typologique.owl
 [Rule-00_ConnaissancesNonAmbigue_LienP]
 [Rule-16-a_Desamb_LienC_Entre_Principe-et-inconnu]
 [Rule-16-b_Principe-P-Principe_Antecedent-Conclusion]
 [Rule-16-c_Principe-P-Procedure_Antecedent-Operation]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-c_Creation_class_Action_Operation]
 [Rule-01-f_Creation_class_regle_nom]
 [Rule-01-g_Creation_class_regle_complete]
 [Rule-01-h_Creation_class_regle_antecedent]
 [Rule-01-i_Creation_class_regle_conclusion]
 [Rule-16-b_Creation_Propriete_aPourComposant_Regles_Nom-Antécédent]
 [Rule-16-c_Creation_Propriete_aPourComposant_Regles_Nom-Consequent]
 [Rule-16-d_Creation_Propriete_aPourComposant_Regles_Nom-Operation]
 [Rule-16-e_Creation_Propriete_aPourComposant_Procedurale_Strategique]
 [Rule-16-f_Creation_Propriete_EstPrecedentDe_Entre_RegleComplete]
 [Rule-16-g_Creation_Propriete_estAntecedentDe_Antecedent_Antecedent]
 [Rule-16-h_Creation_Propriete_ALORS_Antecedent_Operation]
 [Rule-16_i_Creation_Propriete_aPourConclusion_Antecedent_Conclusion]
etat_classification.owl
 [Rule-16-a_Ajoute_Image_Domaine_aPourComposant_entre_RegleNom_Consequence]
 [Rule-16-b_Ajoute_Image_Domaine_aPourComposant_entre_RegleNom_Antecedent]
 [Rule-16-c_Ajoute_Image_Domaine_aPourComposant_entre_RegleNom_Operation]
 [Rule-16-d_Ajoute_Image_Domaine_a_la_Propriete_aPourComosant_ConnProceduraleEtConnStrag]
 [Rule-16-e_Ajout_ImgDom_Prop_estAntecedentDe_Entre_Principes]
 [Rule-16-f_Ajout_ImgDom_Prop_alors_Condition_Procedure]
 [Rule-16-g_Ajout_ImgDom_Prop_estPrecedentDe_RegleNom_RegleNom]
 [Rule-16-h_Ajout_ImgDom_Prop_aPourConclusion]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE_ET_COMPOSITION_REGLE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE_ET_COMPOSITION_REGLE_rest.mot
Comparaison des documents :
-----------------------------------------------------------------
--------------------- Les connaissances  ------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      Principe  : Antecedent-1a_1
      Principe  : Antecedent-1b_2
      Principe  : Antecedent-2a_4
      Principe  : Antecedent-2b_6
      Principe  : Conclusion-1a_3
      Principe  : Nom-1_0
      Principe  : Nom-2_5
      Principe  : Regle-3a_9
      Principe  : Regle-3b_10
      Procédure : Operation-2_7
      Procédure : Proc-3_8
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienC: source Nom-1_0 : dest Antecedent-1a_1
      LienC: source Nom-1_0 : dest Antecedent-1b_2
      LienC: source Nom-1_0 : dest Conclusion-1a_3
      LienC: source Nom-2_5 : dest Antecedent-2a_4
      LienC: source Nom-2_5 : dest Antecedent-2b_6
      LienC: source Nom-2_5 : dest Operation-2_7
      LienC: source Proc-3_8 : dest Regle-3a_9
      LienC: source Proc-3_8 : dest Regle-3b_10
      LienP: source Antecedent-1a_1 : dest Antecedent-1b_2
      LienP: source Antecedent-1b_2 : dest Conclusion-1a_3
      LienP: source Antecedent-2a_4 : dest Antecedent-2b_6
      LienP: source Antecedent-2b_6 : dest Operation-2_7
      LienP: source Regle-3a_9 : dest Regle-3b_10
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 9, Reconstruit = 9, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
```

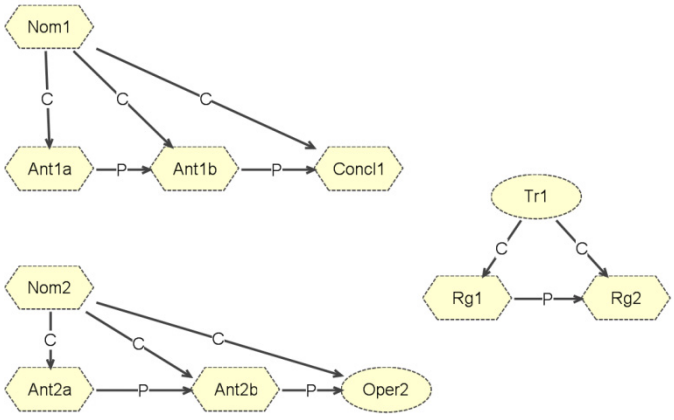
```
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 5, Reconstruit = 5, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
Antécédent 1a est antécédent de Antécédent 1b
Antécédent 1a est une sorte de (metaDom:MD_Strategique_Entite_Regle_Antecedent)
Antécédent 1b a pour conclusion Conclusion 1a
Antécédent 1b est une sorte de (metaDom:MD_Strategique_Entite_Regle_Antecedent)
Antécédent 2a est antécédent de Antécédent 2b
Antécédent 2a est une sorte de (metaDom:MD_Strategique_Entite_Regle_Antecedent)
Antécédent 2b alors Opération 2
Antécédent 2b est une sorte de (metaDom:MD_Strategique_Entite_Regle_Antecedent)
Conclusion 1a est une sorte de (metaDom:MD_Strategique_Entite_Regle_Conclusion)
Nom 1 est une sorte de (metaDom:MD_Strategique_Entite_Regle_Nom)
Nom 1 se compose de Antécédent 1a
Nom 1 se compose de Antécédent 1b
Nom 1 se compose de Conclusion 1a
Nom 2 est une sorte de (metaDom:MD_Strategique_Entite_Regle_Nom)
Nom 2 se compose de Antécédent 2a
Nom 2 se compose de Antécédent 2b
Nom 2 se compose de Opération 2
Opération 2 est une sorte de (metaDom:MD_Procedurale_Operation)
Proc 3 est une sorte de (metaDom:MD_Procedurale_Procedure)
Proc 3 se compose de Règle 3a
Proc 3 se compose de Règle 3b
Règle 3a est précédent de Règle 3b
Règle 3a est une sorte de (metaDom:MD_Strategique_Entite_Regle_Complete)
Règle 3b est une sorte de (metaDom:MD_Strategique_Entite_Regle_Complete)
------------------Résultats après inférence-----------------------
```

B.10.2 Règle à partir de connaissances factuelles

Exemple de représentation en langage MOT



Sémantique formelle

```
:Ant1a_1
      rdf:type metaDom:MD_Strategique_Entite_Regle_Antecedent ;
      rdfs:label "Ant1a"^^xsd:string ;
      metaDom:EST-ANTECEDENT-DE  :Ant1b_2 .
:Ant1b_2
      rdf:type metaDom:MD_Strategique_Entite_Regle_Antecedent ;
      rdfs:label "Ant1b"^^xsd:string ;
      metaDom:A-POUR-CONCLUSION   :Concl1_3 .
:Ant2a_5
```

```
      rdf:type metaDom:MD_Strategique_Entite_Regle_Antecedent ;
      rdfs:label "Ant2a"^^xsd:string ;
      metaDom:EST-ANTECEDENT-DE   :Ant2b_6 .
:Ant2b_6
      rdf:type metaDom:MD_Strategique_Entite_Regle_Antecedent ;
      rdfs:label "Ant2b"^^xsd:string ;
      metaDom:ALORS :Oper2_8 .
:Concl1_3
      rdf:type metaDom:MD_Strategique_Entite_Regle_Conclusion ;
      rdfs:label "Concl1"^^xsd:string .
:Nom1_0
      rdf:type metaDom:MD_Strategique_Entite_Regle_Nom ;
      rdfs:label "Nom1"^^xsd:string ;
      metaDom:A-POUR-COMPOSANT   :Ant1a_1 , :Concl1_3 , :Ant1b_2 .
:Nom2_4
      rdf:type metaDom:MD_Strategique_Entite_Regle_Nom ;
      rdfs:label "Nom2"^^xsd:string ;
      metaDom:A-POUR-COMPOSANT   :Ant2b_6 , :Ant2a_5 , :Oper2_8 .
:Oper2_8
      rdf:type metaDom:MD_Procedurale_Operation ;
      rdfs:label "Oper2"^^xsd:string .
:Rg1_10
      rdf:type metaDom:MD_Strategique_Entite_Regle_Complete ;
      rdfs:label "Rg1"^^xsd:string ;
      metaDom:EST-PRECEDENT-DE  :Rg2_7 .
:Rg2_7
      rdf:type metaDom:MD_Strategique_Entite_Regle_Complete ;
      rdfs:label "Rg2"^^xsd:string .
:Tr1_9
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr1"^^xsd:string ;
      metaDom:A-POUR-COMPOSANT     :Rg2_7 , :Rg1_10 .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
 [Rule-17-a_RegleNom-Antecedent_Enonce-Enonce-Enonce]
 [Rule-17-b_RegleNom-Antecedent_Enonce-Enonce-Trace]
 [Rule-17-d_Enonce-LienC-Trace-RegleComplete_Procedure]
 [Rule-17-e_RegleNom-Antecedent_Enonce-Enonce-trace]
regles_desambig_typologique.owl
 [Rule-00_ConnaissancesNonAmbigue_LienP]
 [Rule-17-b_LienC_entre_deux_enonces]
 [Rule-17-c_Enonce-P-Trace_Antecedent-Operation]
 [Rule-17-d_LienC_entre_EnonceEtTrace]
 [Rule-17-e_Enonce-P-Enonce_Antecedent-Conclusion]
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2c_Creation_Individu_operation]
 [Rule-01-2e_Creation_Individu_Principe_Regle_Nom]
 [Rule-01-2f_Creation_Individu_Principe_Regle_Complete]
 [Rule-01-2g_Creation_Individu_Principe_Antecedent]
 [Rule-01-2h_Creation_Individu_Principe_Conclusion]
etat_classification.owl
 [Rule-11_Creation_propriete_aPourComposant_individu]
 [Rule-17-a_Creation_Individu_Antecedent_aPourConclusion_Conclusion]
 [Rule-17-b_Creation_Individu_Antecedent_estAntecedentDe_Antecedent]
 [Rule-17-c_Creation_Individu_EstPrecedentDe_Entre_RegleComplete]
 [Rule-17-d_Creation_Individu_Antecedent_Alors_Action]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE-ET_COMPOSITION_REGLE_OBSERVABLES.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/PRECEDENCE-ET_COMPOSITION_REGLE_OBSERVABLES_rest.mot
Comparaison des documents :
------------------------------------------------------------------
--------------------- Les connaissances  -----------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      Enonce    : Ant1a_1
      Enonce    : Ant1b_2
      Enonce    : Ant2a_5
      Enonce    : Ant2b_6
      Enonce    : Concl1_3
      Enonce    : Nom1_0
      Enonce    : Nom2_4
      Enonce    : Rg1_10
      Enonce    : Rg2_7
      Trace     : Oper2_8
```

```
      Trace     : Tr1_9
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------- Commun aux deux modèles ---------------------
      LienC: source Nom1_0 : dest Ant1a_1
      LienC: source Nom1_0 : dest Ant1b_2
      LienC: source Nom1_0 : dest Concl1_3
      LienC: source Nom2_4 : dest Ant2a_5
      LienC: source Nom2_4 : dest Ant2b_6
      LienC: source Nom2_4 : dest Oper2_8
      LienC: source Tr1_9 : dest Rg1_10
      LienC: source Tr1_9 : dest Rg2_7
      LienP: source Ant1a_1 : dest Ant1b_2
      LienP: source Ant1b_2 : dest Concl1_3
      LienP: source Ant2a_5 : dest Ant2b_6
      LienP: source Ant2b_6 : dest Oper2_8
      LienP: source Rg1_10 : dest Rg2_7
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 9, Reconstruit = 9, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 5, Reconstruit = 5, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

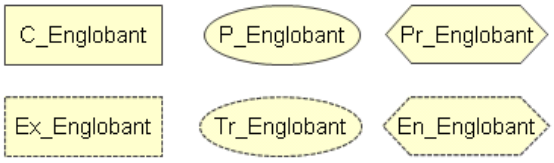
```
----------Début du processus de validation sémantique-------------
[Ant1a] est antécédent de [Ant1b]
[Ant1a] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Antecedent)
[Ant1a] est un ( :Antécédent d'une règle )
[Ant1b] a pour conclusion [Concl1]
[Ant1b] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Antecedent)
[Ant1b] est un ( :Antécédent d'une règle )
[Ant2a] est antécédent de [Ant2b]
[Ant2a] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Antecedent)
[Ant2a] est un ( :Antécédent d'une règle )
[Ant2b] alors [Oper2]
[Ant2b] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Antecedent)
[Ant2b] est un ( :Antécédent d'une règle )
[Concl1] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Conclusion)
[Concl1] est un ( :Conclusion d'une règle )
[Nom1] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Nom)
[Nom1] est un ( :Nom d'une règle )
[Nom1] se compose de [Ant1a]
[Nom1] se compose de [Ant1b]
[Nom1] se compose de [Concl1]
[Nom2] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Nom)
[Nom2] est un ( :Nom d'une règle )
[Nom2] se compose de [Ant2a]
[Nom2] se compose de [Ant2b]
[Nom2] se compose de [Oper2]
[Oper2] est de catégorie (metaDom:MD_Procedurale_Operation)
[Oper2] est un ( :Opération )
[Rg1] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Complete)
[Rg1] est précédent de [Rg2]
[Rg1] est un ( :Règle complète )
[Rg2] est de catégorie (metaDom:MD_Strategique_Entite_Regle_Complete)
[Rg2] est un ( :Règle complète )
[Tr1] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr1] est un ( :Procédure )
[Tr1] se compose de [Rg1]
[Tr1] se compose de [Rg2]

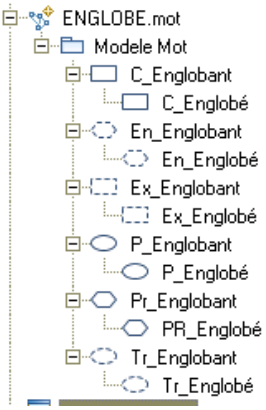
------------------Résultats après inférence-----------------------
[Ant1a] est un ( :Antécédent d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle
)
[Ant1a] est une partie de [Nom1]
[Ant1a] permet [Ant1b]
[Ant1a] permet [Concl1]
[Ant1b] a pour antécédent [Ant1a]
[Ant1b] a pour dépendance [Ant1a]
[Ant1b] alors [Concl1]
[Ant1b] est un ( :Antécédent d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle
)
```

```
[Ant1b] est une partie de [Nom1]
[Ant1b] permet [Concl1]
[Ant2a] est un ( :Antécédent d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle
)
[Ant2a] est une partie de [Nom2]
[Ant2a] permet [Ant2b]
[Ant2a] permet [Oper2]
[Ant2b] a pour antécédent [Ant2a]
[Ant2b] a pour conclusion [Oper2]
[Ant2b] a pour dépendance [Ant2a]
[Ant2b] est un ( :Antécédent d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle
)
[Ant2b] est une partie de [Nom2]
[Ant2b] permet [Oper2]
[Concl1] a pour dépendance [Ant1a]
[Concl1] a pour dépendance [Ant1b]
[Concl1] est la conclusion de [Ant1b]
[Concl1] est un ( :Conclusion d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle
)
[Concl1] est une partie de [Nom1]
[Nom1] est un ( :Nom d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle )
[Nom2] est un ( :Nom d'une règle :Type de connaissances :Connaissance stratégique :Élément d'une règle )
[Oper2] a pour dépendance [Ant2a]
[Oper2] a pour dépendance [Ant2b]
[Oper2] est la conclusion de [Ant2b]
[Oper2] est un ( :Opération :Type de connaissances :Connaissance d'actions )
[Oper2] est une partie de [Nom2]
[Rg1] est un ( :Règle complète :Type de connaissances :Connaissance stratégique :Élément d'une règle )
[Rg1] est une partie de [Tr1]
[Rg1] permet [Rg2]
[Rg2] a pour dépendance [Rg1]
[Rg2] est la suivant de [Rg1]
[Rg2] est un ( :Règle complète :Type de connaissances :Connaissance stratégique :Élément d'une règle )
[Rg2] est une partie de [Tr1]
[Tr1] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
```

B.11 Connaissance qui englobe des connaissances

Exemple de représentation en langage MOT





Sémantique formelle

```
:C_Englobant_0
      rdf:type owl:Class ;
      rdfs:label "C_Englobant"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept ;
      metaDom:ENGLOBE :C_Englobe_1 .
:C_Englobant_0_englobe_C_Englobe_1
      rdf:type owl:ObjectProperty ;
      rdfs:domain :C_Englobant_0 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :C_Englobe_1 ;
      rdfs:subPropertyOf metaDom:ENGLOBE .
:C_Englobe_1
      rdf:type owl:Class ;
      rdfs:label "C_EnglobÃ©"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Declarative_Concept .
:En_Englobant_10
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En_Englobant"^^xsd:string ;
      metaDom:ENGLOBE :En_Englobe_11 .
```

```
:En_Englobe_11
      rdf:type metaDom:MD_Strategique_AgentContrainteNorme ;
      rdfs:label "En_EnglobÃ©"^^xsd:string .
:Ex_Englobant_6
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex_Englobant"^^xsd:string ;
      metaDom:ENGLOBE :Ex_Englobe_7 .
:Ex_Englobe_7
      rdf:type metaDom:MD_Declarative_Concept ;
      rdfs:label "Ex_EnglobÃ©"^^xsd:string .
:PR_Englobe_5
      rdf:type owl:Class ;
      rdfs:label "PR_EnglobÃ©"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme .
:P_Englobant_2
      rdf:type owl:Class ;
      rdfs:label "P_Englobant"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing ;
      metaDom:ENGLOBE :P_Englobe_3 .
:P_Englobant_2_englobe_P_Englobe_3
      rdf:type owl:ObjectProperty ;
      rdfs:domain :P_Englobant_2 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :P_Englobe_3 ;
      rdfs:subPropertyOf metaDom:ENGLOBE .
:P_Englobe_3
      rdf:type owl:Class ;
      rdfs:label "P_EnglobÃ©"^^xsd:string ;
      rdfs:subClassOf metaDom:MD_Procedurale_Procedure , owl:Thing .
:Pr_Englobant_4
      rdf:type owl:Class ;
      rdfs:label "Pr_Englobant"^^xsd:string ;
      rdfs:subClassOf owl:Thing , metaDom:MD_Strategique_AgentContrainteNorme ;
      metaDom:ENGLOBE :PR_Englobe_5 .
:Pr_Englobant_4_englobe_PR_Englobe_5
      rdf:type owl:ObjectProperty ;
      rdfs:domain :Pr_Englobant_4 ;
      rdfs:label ""^^xsd:string ;
      rdfs:range :PR_Englobe_5 ;
      rdfs:subPropertyOf metaDom:ENGLOBE .
:Tr_Englobant_8
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr_Englobant"^^xsd:string ;
      metaDom:ENGLOBE :Tr_Englobe_9 .
:Tr_Englobe_9
      rdf:type metaDom:MD_Procedurale_Procedure ;
      rdfs:label "Tr_EnglobÃ©"^^xsd:string .
```

Règles de désambiguïsation

```
regles_desambig_topologique.owl
regles_desambig_typologique.owl
```

Règles de conversion

```
etat_init.owl
 [Rule-00_Creer_ontologie_du_domaine]
 [Rule-00_Importer_metaOnto_du_domaine]
etat_creation.owl
 [Rule-01-2a_Creation_Individu_objet]
 [Rule-01-2b_Creation_Individu_procedural]
 [Rule-01-2d_Creation_Individu_Principe_Agent]
 [Rule-01-a_Creation_class_objet_Concept]
 [Rule-01-b_Creation_class_Action_Procedure]
 [Rule-01-d_Creation_class_agent]
 [Rule-18-a_Creation_Propriete_Englobe_EntAbstraite_EntAbstraite]
etat_classification.owl
 [Rule-18-b_Englobe-Declaratif-Declaratif]
 [Rule-18_Ajout_ImgDom_Prop_englobe]
etat_final.owl
 [Rule-00_Sauvegarde]
```

Rapport de validation syntaxique

```
Comparaison des documents :
Orig   = C:/Developpement/DATA/workspace/CarréDeSable/ENGLOBE.mot
Recons = C:/Developpement/DATA/workspace/CarréDeSable/ENGLOBE_rest.mot
Comparaison des documents :
------------------------------------------------------------------
---------------------- Les connaissances  ------------------------
------------------------------------------------------------------
------------------- Commun aux deux modèles ----------------------
      Concept   : C_Englobant_0
      Concept   : C_Englobe_1
      Enonce    : En_Englobant_10
      Enonce    : En_Englobe_11
      Exemple   : Ex_Englobant_6
      Exemple   : Ex_Englobe_7
```

```
      Principe  : PR_Englobe_5
      Principe  : Pr_Englobant_4
      Procédure : P_Englobant_2
      Procédure : P_Englobe_3
      Trace     : Tr_Englobant_8
      Trace     : Tr_Englobe_9
-----------------------------------------------------------------
---------------------- Les relations ----------------------------
-----------------------------------------------------------------
------------------ Commun aux deux modèles ----------------------
-----------------------------------------------------------------
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 2, Reconstruit = 2, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

Rapport de validation sémantique

```
----------Début du processus de validation sémantique-------------
C_Englobant est une sorte de (metaDom:MD_Declarative_Concept)
C_Englobé englobe C_Englobant
C_Englobé est une sorte de (metaDom:MD_Declarative_Concept)
PR_Englobé englobe Pr_Englobant
PR_Englobé est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
P_Englobant est une sorte de (metaDom:MD_Procedurale_Procedure)
P_Englobé englobe P_Englobant
P_Englobé est une sorte de (metaDom:MD_Procedurale_Procedure)
Pr_Englobant est une sorte de (metaDom:MD_Strategique_AgentContrainteNorme)
[En_Englobant] englobe [En_Englobé]
[En_Englobant] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En_Englobant] est un ( :Agent-Contrainte-Norme )
[En_Englobé] est de catégorie (metaDom:MD_Strategique_AgentContrainteNorme)
[En_Englobé] est un ( :Agent-Contrainte-Norme )
[Ex_Englobant] englobe [Ex_Englobé]
[Ex_Englobant] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex_Englobant] est un ( :Connaissance d'objet )
[Ex_Englobé] est de catégorie (metaDom:MD_Declarative_Concept)
[Ex_Englobé] est un ( :Connaissance d'objet )
[Tr_Englobant] englobe [Tr_Englobé]
[Tr_Englobant] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr_Englobant] est un ( :Procédure )
[Tr_Englobé] est de catégorie (metaDom:MD_Procedurale_Procedure)
[Tr_Englobé] est un ( :Procédure )
------------------Résultats après inférence----------------------
[En_Englobant] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[En_Englobé] est englobé par [En_Englobant]
[En_Englobé] est un ( :Agent-Contrainte-Norme :Type de connaissances :Connaissance stratégique )
[Ex_Englobant] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Ex_Englobé] est englobé par [Ex_Englobant]
[Ex_Englobé] est un ( :Connaissance d'objet :Type de connaissances :Connaissance déclarative )
[Tr_Englobant] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
[Tr_Englobé] est englobé par [Tr_Englobant]
[Tr_Englobé] est un ( :Procédure :Type de connaissances :Connaissance d'actions )
```

APPENDICE C

CATALOGUE DES COHÉRENCES DE L'ONTOLOGIE CADRE D'ONTOCASE

C.1 Axiomes d'identification des erreurs de cohérences

L'ontologie cadre d'OntoCASE est une métastructure qui permet de représenter les connaissances d'un domaine. Certains axiomes régissent la représentation du domaine afin d'assurer la cohérence. La stratégie employée pour assurer la cohérence en est une de type contrainte. Plutôt que d'identifier les conditions qui sont correctes, la stratégie par contrainte détermine les conditions qui sont incorrectes. Par exemple, l'axiome[116] **a** du tableau c.1 identifie qu'il y a une erreur d'hyponymie entre la catégorie *primaire* si une relation est classée comme hyponyme et que la connaissance source de la relation est de catégorie *primaire* et que la connaissance destination est de catégorie *règle*, *secondaire* ou *tertiaire*.

[116] Les axiomes sont dans la notation Manchester OWL (Horridge, Drummond, Goodwin, Rector, Stevens et Wang, *The Manchester OWL Syntax*)

Tableau C.1
Axiomes d'identification des incohérences

a) ot:OT_ERREUR_HYPONYME_CategoriePrimaire

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire)))
```

b) ot:OT_ERREUR_HYPONYME_CategorieRegle_ANTECEDENT

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Complete)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Conclusion)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Nom)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
```

c) ot:OT_ERREUR_HYPONYME_CategorieRegle_COMPLETE

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Antecedent)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Conclusion)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Nom)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
```

d) ot:OT_ERREUR_HYPONYME_CategorieRegle_CONCLUSION

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Antecedent)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Complete)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Nom)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
```

e) ot:OT_ERREUR_HYPONYME_CategorieRegle_NOM

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Antecedent)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Complete)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Conclusion)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
```

f) ot:OT_ERREUR_HYPONYME_CategorieSecondaire

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire)))
```

g) ot:OT_ERREUR_HYPONYME_CategorieTertiaire

```
(oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Tertiaire)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Tertiaire)))
 or (oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Tertiaire)))
```

h) ot:OT_ERREUR_HYPONYME_Schema

```
oRef:OR_Relation_Hyponyme
 and ((oRef:OR_connDestination some oRef:OR_Entite_Schema)
 and (oRef:OR_connSource some oRef:OR_Entite_Schema))
```

i) ot:OT_ERREUR_INSTANCE_Agent

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire_Declaratif)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire_Procedural)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Entite_FacteurInfluence_Condition)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Strategique)))
```

j) ot:OT_ERREUR_INSTANCE_CategoriePrimaire

```
oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire))
```

k) ot:OT_ERREUR_INSTANCE_CategorieSecondaire

```
oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire))
```

l) ot:OT_ERREUR_INSTANCE_Complete

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Antecedent)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Conclusion)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Nom)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Complete)))
```

m) ot:OT_ERREUR_INSTANCE_Condition

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire_Declaratif)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire_Procedurale)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Strategique)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Strategique)))
```

n) ot:OT_ERREUR_INSTANCE_Operation

```
(oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Primaire_Declaratif)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Procedural))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Primaire_Strategique)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Procedural))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Procedural))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Procedural))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Primaire_Procedural))
```

o) ot:OT_ERREUR_INSTANCE_Procedure

```
(oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Procedurale))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Procedurale))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Secondaire_Declaratif)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Procedurale))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Secondaire_Strategique)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Procedurale))
 or (oRef:OR_Relation_Instance
 and (oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Procedurale))
```

p) ot:OT_ERREUR_INSTANCE_Propriete

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Tertiaire)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Categorie_Tertiaire)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Tertiaire)))
```

q) ot:OT_ERREUR_INSTANCE_RegleAntecedent

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Complete)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Conclusion)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Nom)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Antecedent)))
```

r) ot:OT_ERREUR_INSTANCE_RegleConclusion

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Antecedent)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Complete)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Nom)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Conclusion)))
```

s) ot:OT_ERREUR_INSTANCE_RegleNom

```
(oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Antecedent)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Complete)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle_Conclusion)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
 or (oRef:OR_Relation_Instance
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Tertiaire)
 and (oRef:OR_connSource some oRef:OR_Categorie_Regle_Nom)))
```

t) ot:OT_ERREUR_REGULATION_ConnaissanceDeclarative

```
(oRef:OR_Relation_Regulation
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Declarative)))
 or (oRef:OR_Relation_Regulation
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Declarative)))
 or (oRef:OR_Relation_Regulation
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Declarative)))
```

u) ot:OT_ERREUR_REGULATION_ConnaissanceProcedurale

```
(oRef:OR_Relation_Regulation
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Primaire)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Procedurale)))
 or (oRef:OR_Relation_Regulation
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Regle)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Procedurale)))
 or (oRef:OR_Relation_Regulation
 and ((oRef:OR_connDestination some oRef:OR_Categorie_Secondaire)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Procedurale)))
```

v) ot:OT_ERREUR_REL_ATTRIBUT_PROCEDURAL

```
(oRef:OR_Relation_Attribut
 and (oRef:OR_connDestination some oRef:OR_Categorie_Primaire_Declaratif)
 and (oRef:OR_connSource some oRef:OR_Categorie_Secondaire_Declaratif))
 or (oRef:OR_Relation_Attribut
 and (oRef:OR_connDestination some oRef:OR_Connaissance_Procedurale)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Procedurale))
 or (oRef:OR_Relation_Attribut
 and (oRef:OR_connDestination some oRef:OR_Connaissance_Procedurale)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Strategique))
 or (oRef:OR_Relation_Attribut
 and (oRef:OR_connDestination some oRef:OR_Connaissance_Strategique)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Procedurale))
 or (oRef:OR_Relation_Attribut
 and (oRef:OR_connDestination some oRef:OR_Connaissance_Strategique)
 and (oRef:OR_connSource some oRef:OR_Connaissance_Strategique))
```

C.2 Règles de génération des erreurs

Après l'identification des situations en incohérence, le déclenchement des règles présenté au tableau c.2 permet l'exécution de la commande de gestion des erreurs, notamment l'envoi d'un message au mécanisme d'impression des erreurs dans l'environnement informatique d'OntoCASE.

Tableau C.2
Règles de génération des erreurs

a) Rule-ERREUR_HYPONYME

```
ot:OT_ERREUR_HYPONYME_Categorie(?err_ins)  ∧
ot:OT_ERREUR_Message(ot:OT_MessageErreur_HYPONYME, ?err_msg)  ∧
oRef:OR_connDestination(?err_ins, ?connDest)  ∧
oRef:OR_identifiant(?err_ins, ?ident)
  → swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ∧
swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
```

b) Rule-ERREUR_HYPONYME_SCHEMA

```
ot:OT_ERREUR_HYPONYME_Schema(?err_ins)  ∧
ot:OT_ERREUR_Message(ot:OT_MessageErreur_HYPONYME_Schema, ?err_msg)  ∧
oRef:OR_connDestination(?err_ins, ?connDest)  ∧
oRef:OR_identifiant(?err_ins, ?ident)
  → swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ∧
swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
```

c) Rule-ERREUR_INSTANCE

```
ot:OT_ERREUR_INSTANCE(?err_ins)  ∧
ot:OT_ERREUR_Message(ot:OT_MessageErreur_INSTANCE, ?err_msg)  ∧
oRef:OR_connDestination(?err_ins, ?connDest)  ∧
oRef:OR_identifiant(?err_ins, ?ident)
  → swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ∧
swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
```

d) Rule-ERREUR_REGULATION

```
ot:OT_ERREUR_REGULATION(?err_ins)  ∧
ot:OT_ERREUR_Message(ot:OT_MessageErreur_REGULATION, ?err_msg)  ∧
oRef:OR_connDestination(?err_ins, ?connDest)  ∧
oRef:OR_identifiant(?err_ins, ?ident)
  → swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ∧
swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
```

e) Rule-ERREUR_REL_ATTRIBUT

```
ot:OT_ERREUR_REL_ATTRIBUT(?err_ins)  ∧
ot:OT_ERREUR_Message(ot:OT_MessageErreur_REL_ATTRIBUT, ?err_msg)  ∧
oRef:OR_connDestination(?err_ins, ?connDest)  ∧
oRef:OR_identifiant(?err_ins, ?ident)
  → swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ∧
swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
```

APPENDICE D

SCÉNARIOS DE CAS DE TESTS

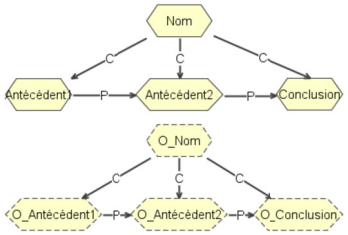
Quatre scénarios de tests sont présentés dans cet appendice : le scénario de test unitaire qui évalue l'étape de conversion par la formalisation des éléments de base d'un modèle, le scénario de test fonctionnel qui évalue l'étape de désambiguïsation par la formalisation d'une multitude de combinaisons de modélisation possibles, le scénario de tests systémiques qui évalue l'efficacité du système à modéliser des modèles complexes contenant une sémantique de domaine réel, et finalement, le scénario de test d'incohérence qui permet d'évaluer l'efficacité du système à identifier des erreurs de modélisation. Pour chacun des tests, nous présentons le modèle soumis au processus de formalisation ainsi que le bilan de la formalisation qui comprend un énoncé des éléments du modèle ainsi que le nombre total d'éléments contenus dans le modèle d'origine et le nombre total d'éléments dans le modèle reconstruit. Le détail du processus d'exécution du test est présenté à la section 3.3.3, p. 183

D.1 Scénario de tests unitaires

Les cas de tests unitaires sont désambiguïsés de manière manuelle afin de concentrer l'exécution de la formalisation à l'étape de conversion.

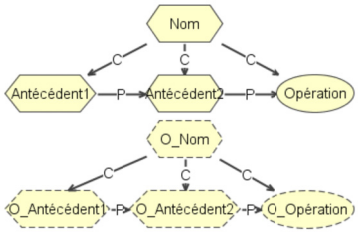
Tableau D.1
nom du test, modèle d'origine et bilan du scénario de test unitaire

a) À pour conclusion



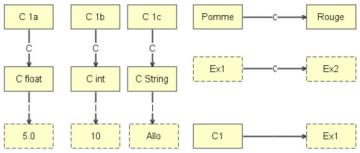

```
----------------------- BILAN ----------------------------------
----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 4, Reconstruit = 4, Différence = 0
Compte des Enonces   : Original = 4, Reconstruit = 4, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

b) alors



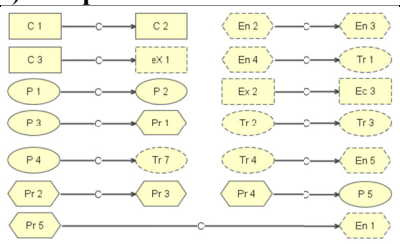

```
----------------------- BILAN ----------------------------------
----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 3, Reconstruit = 3, Différence = 0
Compte des Enonces   : Original = 3, Reconstruit = 3, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

c) Attribut



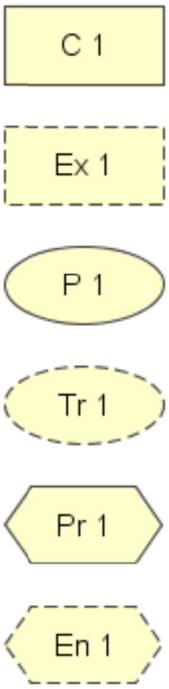
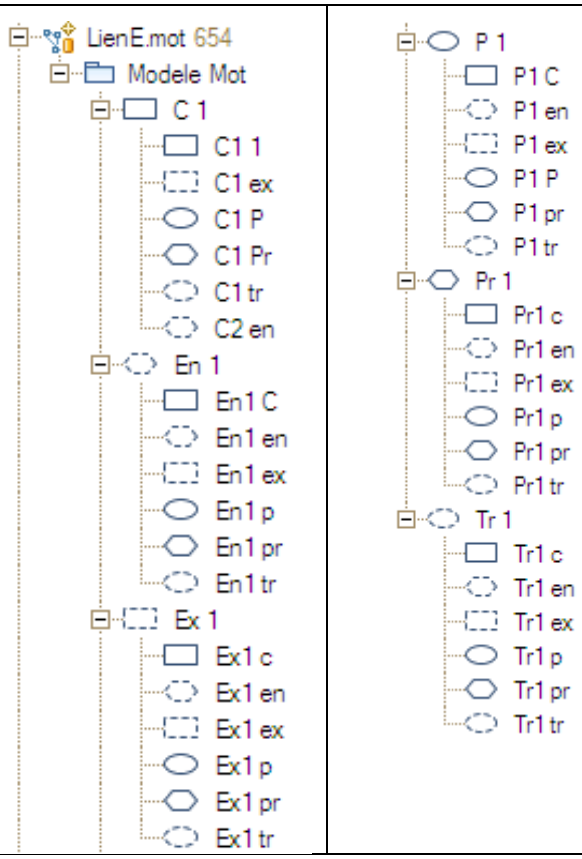

```
----------------------- BILAN ----------------------------------
----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 9, Reconstruit = 9, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 5, Reconstruit = 5, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

d) Composition



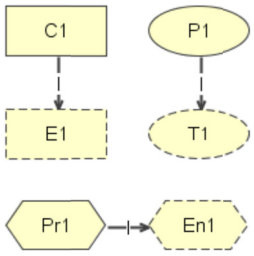

```
----------------------- BILAN ----------------------------------
----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 3, Reconstruit = 3, Différence = 0
Compte des Procedures: Original = 5, Reconstruit = 5, Différence = 0
Compte des Principes : Original = 5, Reconstruit = 5, Différence = 0
Compte des Enonces   : Original = 5, Reconstruit = 5, Différence = 0
Compte des Traces    : Original = 5, Reconstruit = 5, Différence = 0
Compte des Exemples  : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC   : Original = 13, Reconstruit = 13, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

e) Englobe

Représentation graphique	Structure de données
	

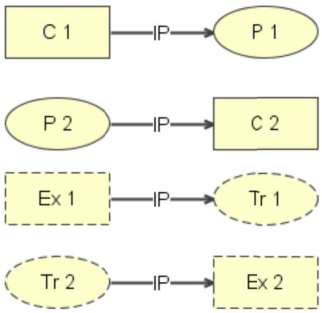

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles -------------
Compte des Concepts  : Original = 7, Reconstruit = 7, Différence = 0
Compte des Procedures: Original = 7, Reconstruit = 7, Différence = 0
Compte des Principes : Original = 7, Reconstruit = 7, Différence = 0
Compte des Enonces   : Original = 7, Reconstruit = 7, Différence = 0
Compte des Traces    : Original = 7, Reconstruit = 7, Différence = 0
Compte des Exemples  : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

f)Instanciation



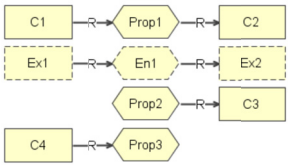

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles -------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

g) Intrant et Produit



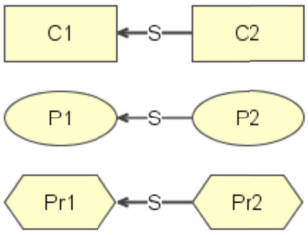

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

h) Propriété et Propiété-Unaire



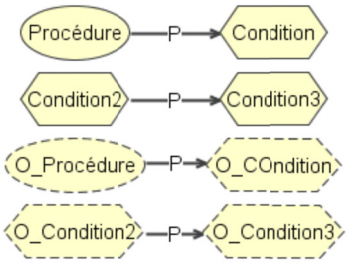

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 4, Reconstruit = 4, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 3, Reconstruit = 3, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 2, Différence = 1
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 6, Reconstruit = 8, Différence = 2
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

i) Spécialisation



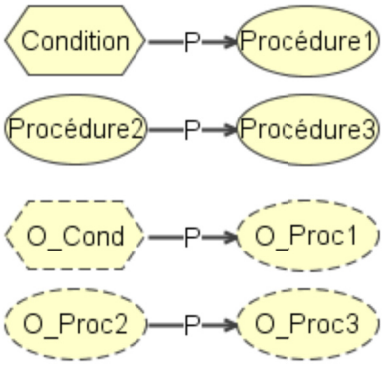

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 3, Reconstruit = 3, Différence = 0
```

j) Puis évaluer



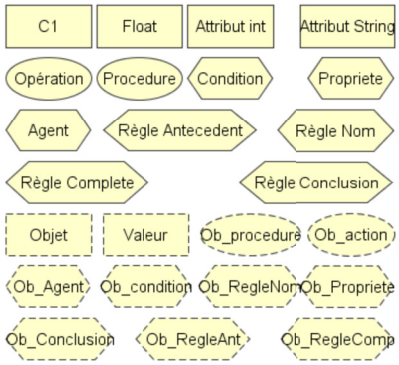

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 3, Reconstruit = 3, Différence = 0
Compte des Enonces   : Original = 3, Reconstruit = 3, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

k) Puis-exécuter




```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 3, Reconstruit = 3, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 3, Reconstruit = 3, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

l) Les éléments de modèle



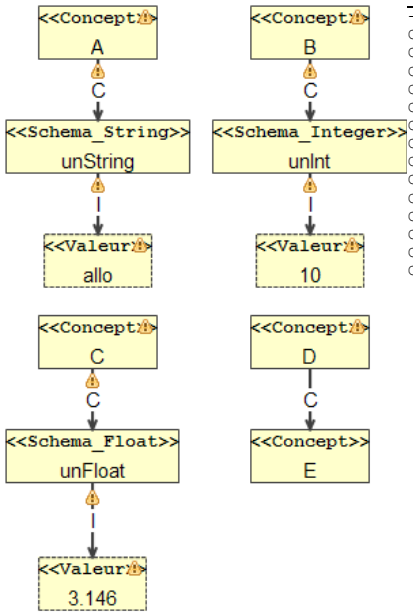

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 4, Reconstruit = 4, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 7, Reconstruit = 8, Différence = 1
Compte des Enonces   : Original = 7, Reconstruit = 8, Différence = 1
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

D.2 Scénario de tests fonctionnels

Ce scénario de test assure la validation de l'étape de conversion et de désambiguïsation. Les éléments contenant des ambiguïtés de nature sémantique sont désambiguïsés de manière manuelle.

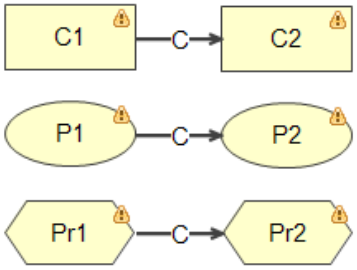
Tableau D.2
nom du test, modèle d'origine et bilan du scénario de test fonctionnel

a) A pour attribut



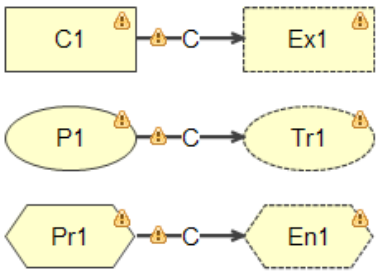

```
---------------------- BILAN ------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 8, Reconstruit = 8, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

b) Composition d'ojets abstraits



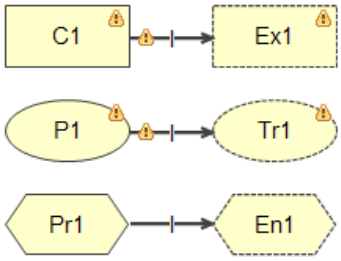

```
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

c) Composition d'observables



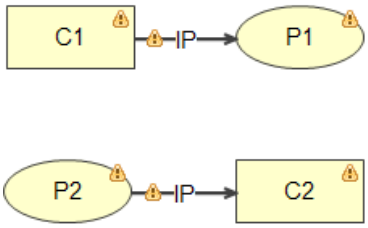

```
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

d) Instanciation



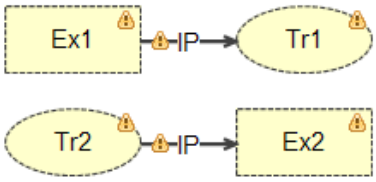

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 1, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

e) Intrant/produit d'objets abstraits



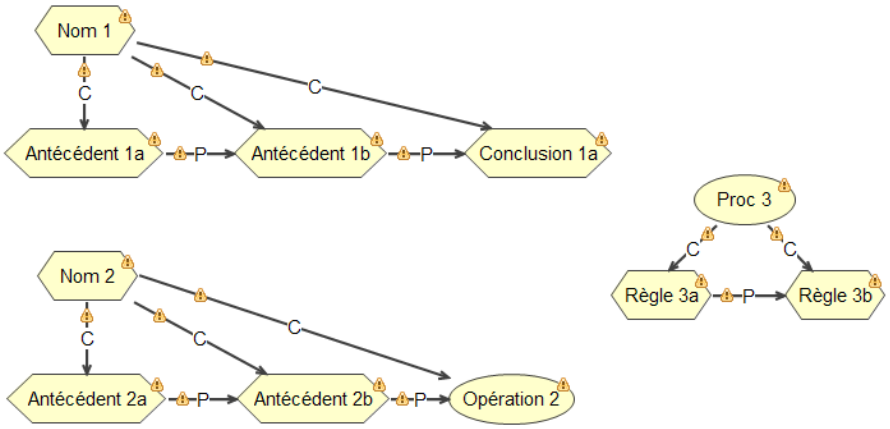

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

f) Intrant/Produit d'objets concrèts



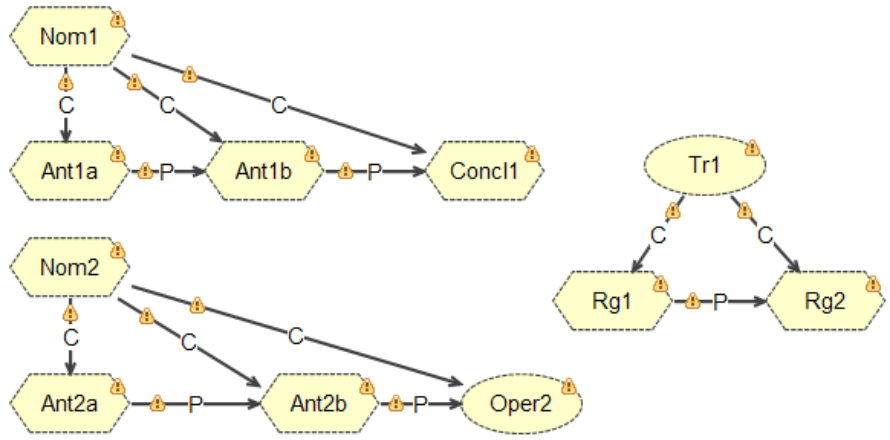

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

g) Règles, procédures et opérations



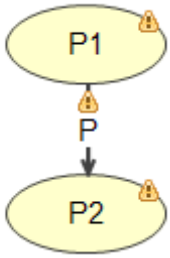

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 9, Reconstruit = 9, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 5, Reconstruit = 5, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

h) Règles, procédures et opérations d'observables



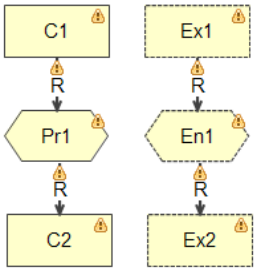

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 9, Reconstruit = 9, Différence = 0
Compte des Traces    : Original = 2, Reconstruit = 2, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 5, Reconstruit = 5, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

i) Précédence de procédures



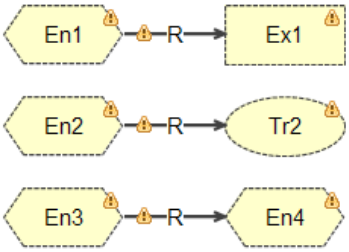

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

j) Propriétés et assertions



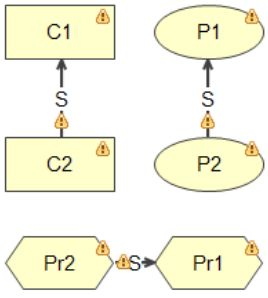

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 1, Reconstruit = 1, Différence = 0
Compte des Enonces   : Original = 1, Reconstruit = 2, Différence = 1
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 4, Reconstruit = 6, Différence = 2
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

k) Régulation entre observables




```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 4, Reconstruit = 4, Différence = 0
Compte des Traces    : Original = 1, Reconstruit = 1, Différence = 0
Compte des Exemples  : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

l)La spécialisation




```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 2, Reconstruit = 2, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 2, Reconstruit = 2, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 3, Reconstruit = 3, Différence = 0
```

D.3 Scénario de tests systémiques

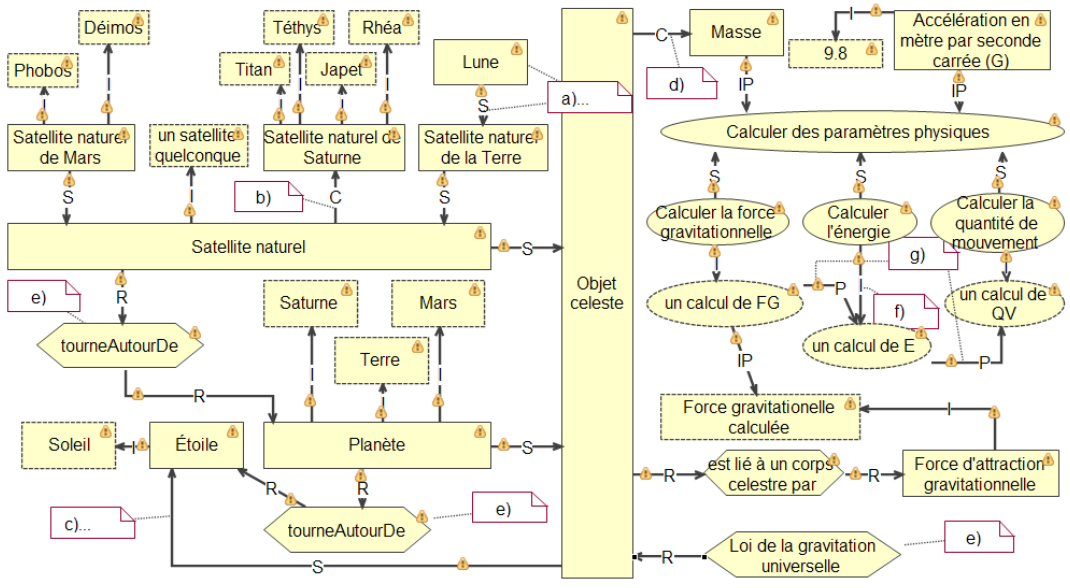
Le scénario de tests systémiques mesure la capacité du volet computationnel d'OntoCASE de formaliser des modèles complexes qui font intervenir des

combinaisons de représentation diverses. Les éléments dont l'ambiguïté sémantique le nécessite sont préalablement désambiguïsés. Le test valide l'étape de désambiguïsation et l'étape de conversion.

D.3.1 Scénarios de tests systémiques issus de l'expérimentation

La première catégorie de scénarios de tests systémiques provient de l'expérimentation réalisée dans cette thèse. Il s'agit du modèle *Le système solaire* présenté au tableau d.3 ainsi que le modèle (voir la tableau d.4) d'une interprétation possible du texte *l : gestion des déchets* (présenté à la figure 4.2, p. 200).

Tableau D.3
Scénario de test systémique: *Le système solaire*

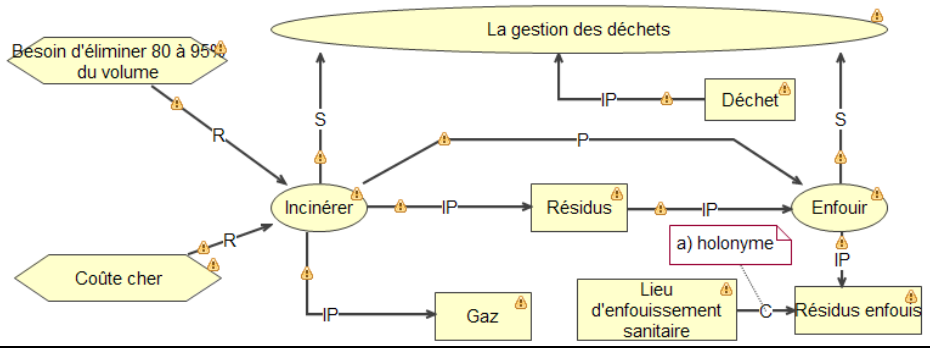



```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles -----------
Compte des Concepts  : Original = 11, Reconstruit = 11, Différence = 0
Compte des Procedures: Original = 4, Reconstruit = 4, Différence = 0
Compte des Principes : Original = 4, Reconstruit = 4, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 3, Reconstruit = 3, Différence = 0
Compte des Exemples  : Original = 13, Reconstruit = 13, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 16, Reconstruit = 16, Différence = 0
Compte des LienIP    : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienP     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienR     : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienS     : Original = 9, Reconstruit = 9, Différence = 0
```

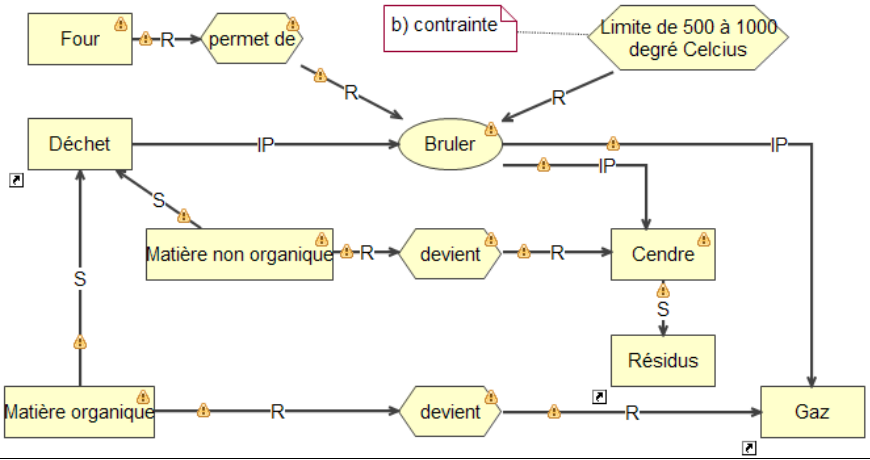
Le modèle du tableau d.4 contient certains éléments, l'holonyme de l'étiquette **a** et la contrainte de l'étiquette **b** sont désambiguïsés de manière manuelle.

Tableau D.4
Scénario de test systémique: *La gestion des déchets*

a) Le modèle de haut niveau



b) Sous-modèle du processus *incinérer*

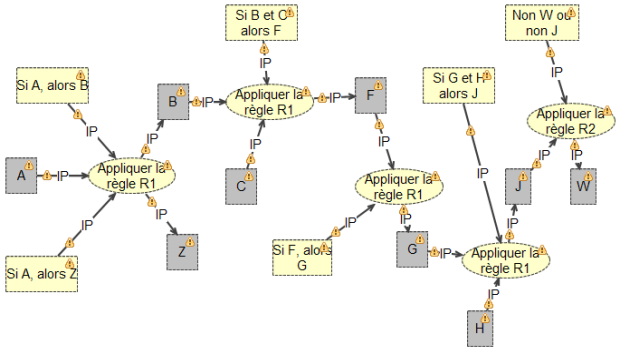


a) le bilan après formalisation

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles -------------
Compte des Concepts  : Original = 9, Reconstruit = 9, Différence = 0
Compte des Procedures: Original = 4, Reconstruit = 4, Différence = 0
Compte des Principes : Original = 6, Reconstruit = 6, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienP     : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienR     : Original = 9, Reconstruit = 9, Différence = 0
Compte des LienS     : Original = 5, Reconstruit = 5, Différence = 0
```

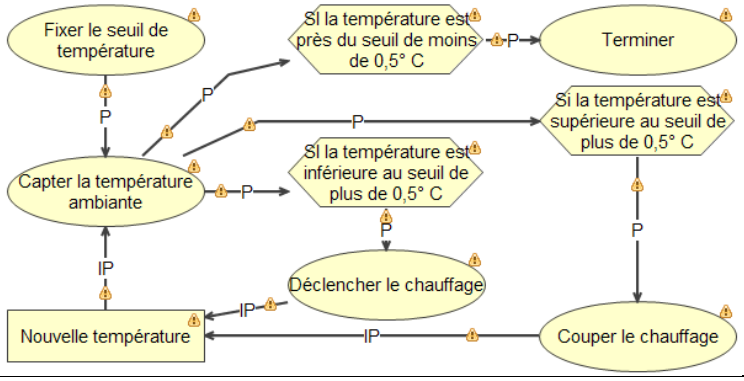
D.3.2 Scénarios de tests systémiques issus d'ouvrage en modélisation

a) Arbre de déduction



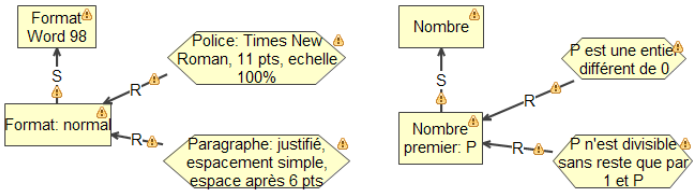

```
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 5, Reconstruit = 5, Différence = 0
Compte des Exemples  : Original = 15, Reconstruit = 15, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 19, Reconstruit = 19, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

b) Procédure itérative



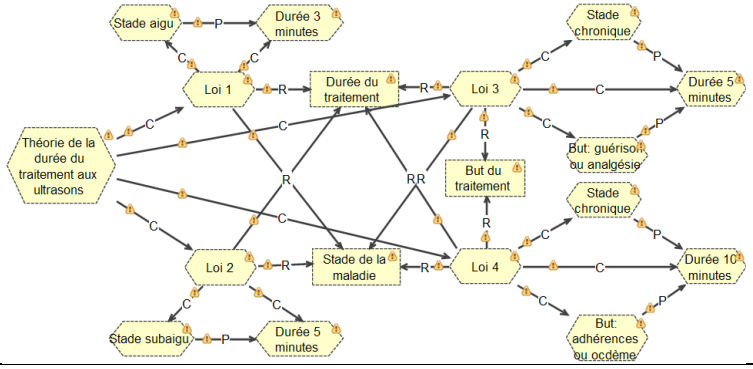

```
---------------------- BILAN ------------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 1, Reconstruit = 1, Différence = 0
Compte des Procedures: Original = 5, Reconstruit = 5, Différence = 0
Compte des Principes : Original = 3, Reconstruit = 3, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienP     : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

c) Normes et connaissances prescriptives



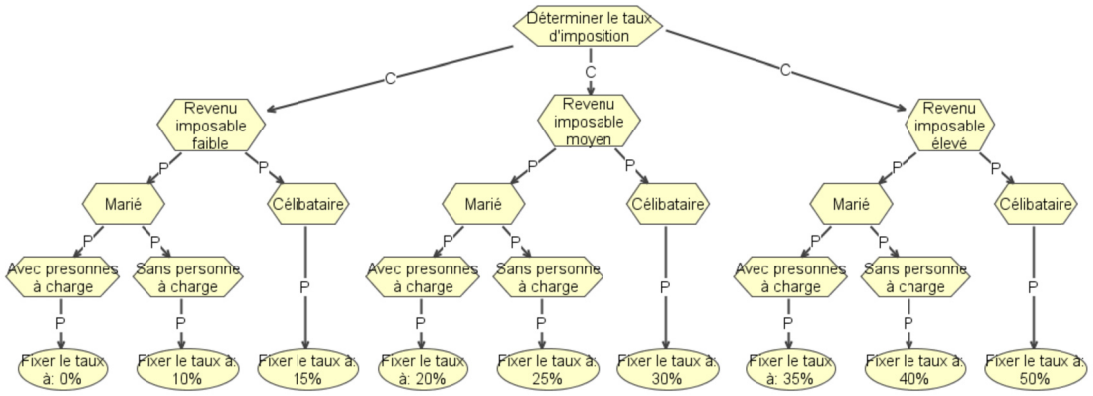

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 4, Reconstruit = 4, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 4, Reconstruit = 4, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienS     : Original = 2, Reconstruit = 2, Différence = 0
```

d) Lois sur la durée d'un traitement



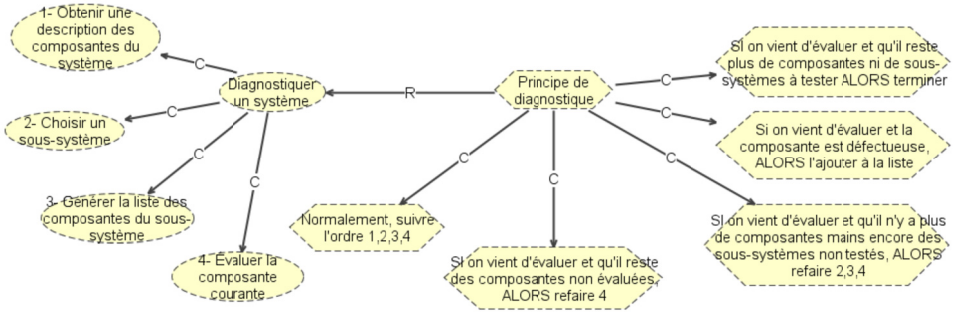

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 15, Reconstruit = 15, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 14, Reconstruit = 14, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienR     : Original = 10, Reconstruit = 10, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

e) Arbre de décision, le calcul de l'impôt



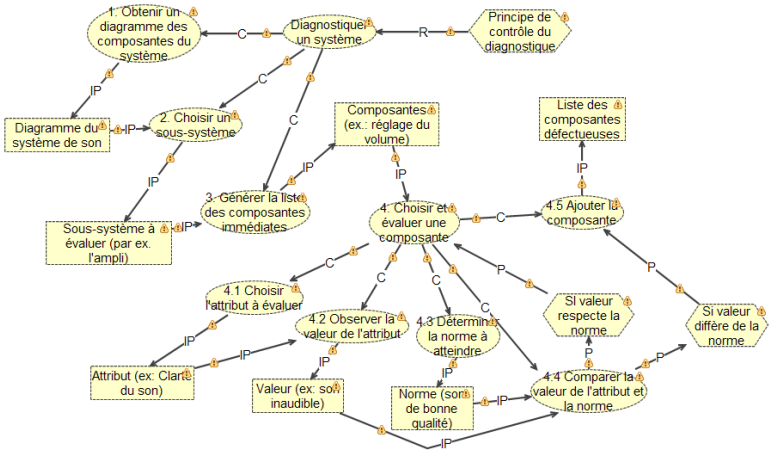

```
---------- BILAN ----------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 9, Reconstruit = 9, Différence = 0
Compte des Principes : Original = 16, Reconstruit = 16, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 21, Reconstruit = 21, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

f) Contrôle et procédures



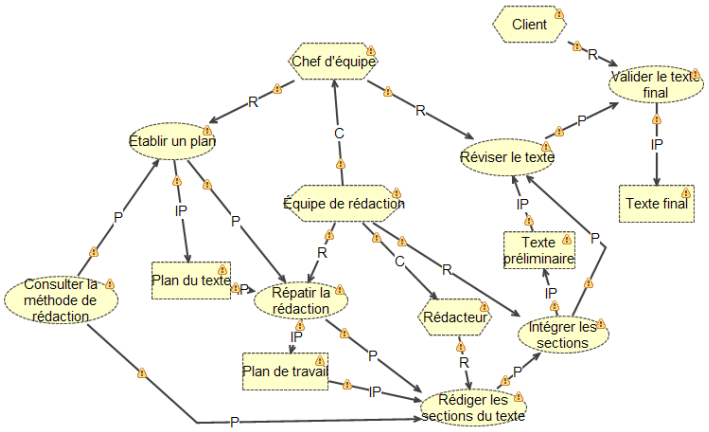

```
---------- BILAN ----------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 6, Reconstruit = 6, Différence = 0
Compte des Traces    : Original = 5, Reconstruit = 5, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 9, Reconstruit = 9, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

g) Processus de diagnostic



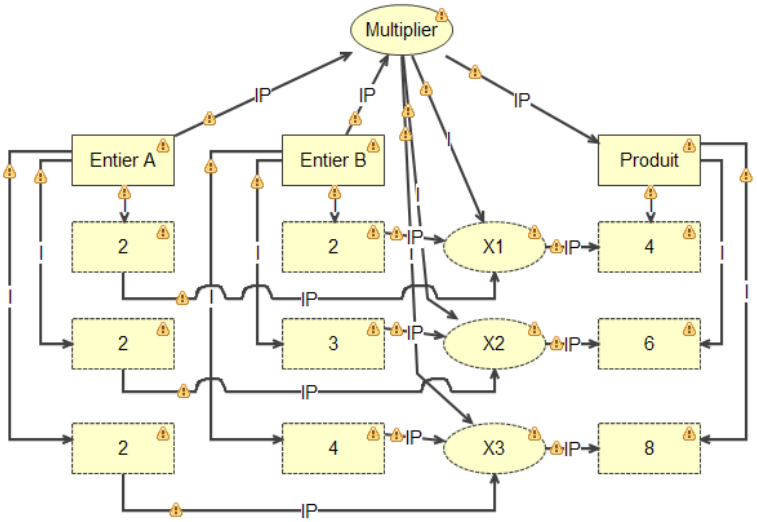

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 3, Reconstruit = 3, Différence = 0
Compte des Traces    : Original = 10, Reconstruit = 10, Différence = 0
Compte des Exemples  : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 13, Reconstruit = 13, Différence = 0
Compte des LienP     : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienR     : Original = 1, Reconstruit = 1, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

h) Processus collaboratif de rédaction




```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 4, Reconstruit = 4, Différence = 0
Compte des Traces    : Original = 7, Reconstruit = 7, Différence = 0
Compte des Exemples  : Original = 4, Reconstruit = 4, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienP     : Original = 7, Reconstruit = 7, Différence = 0
Compte des LienR     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

i) Systèmes factuels



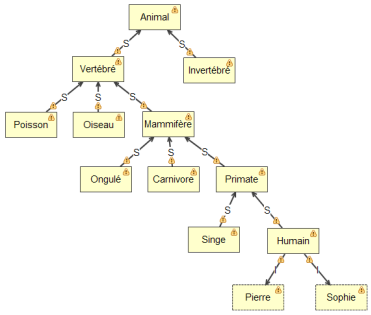

```
---------------------- BILAN --------------------------------
-----------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 3, Reconstruit = 3, Différence = 0
Compte des Procedures: Original = 1, Reconstruit = 1, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 3, Reconstruit = 3, Différence = 0
Compte des Exemples  : Original = 9, Reconstruit = 9, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 12, Reconstruit = 12, Différence = 0
Compte des LienIP    : Original = 12, Reconstruit = 12, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

j) Réseau d'énoncés



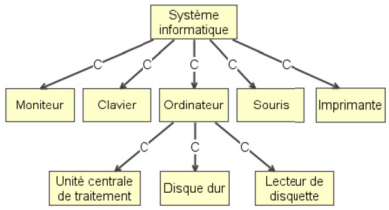

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 6, Reconstruit = 6, Différence = 0
Compte des Enonces   : Original = 36, Reconstruit = 36, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 35, Reconstruit = 35, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

k) Taxonomie animale



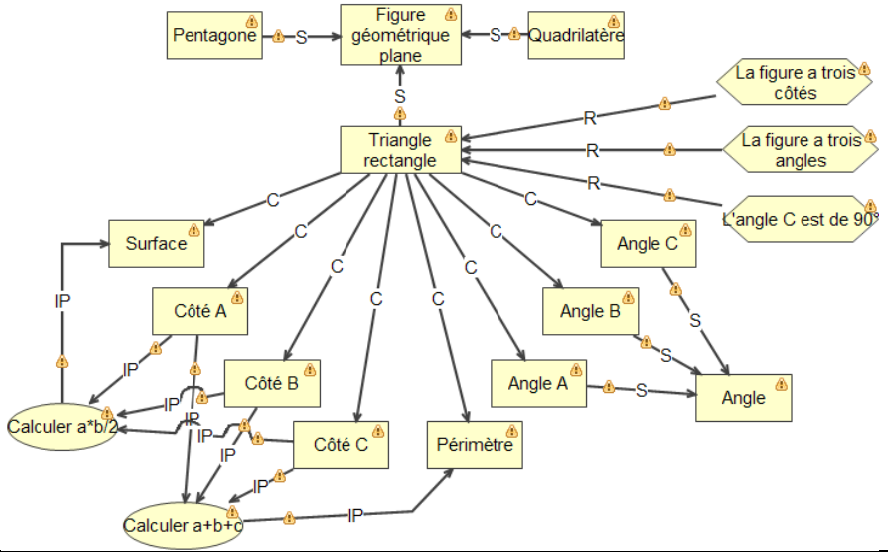

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 11, Reconstruit = 11, Différence =
0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 10, Reconstruit = 10, Différence =
0
```

l) Système à composants



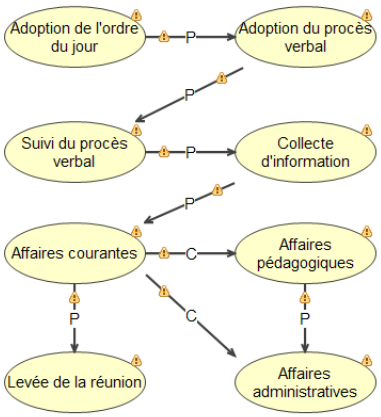

```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 9, Reconstruit = 9, Différence = 0
Compte des Procedures: Original = 0, Reconstruit = 0, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

m) Système conceptuel hybride




```
---------------------- BILAN ----------------------------------
---------------------------------------------------------------
---------Totaux des divers composants des modèles --------------
Compte des Concepts  : Original = 13, Reconstruit = 13, Différence = 0
Compte des Procedures: Original = 2, Reconstruit = 2, Différence = 0
Compte des Principes : Original = 3, Reconstruit = 3, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 8, Reconstruit = 0, Différence = 8
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 8, Reconstruit = 8, Différence = 0
Compte des LienP     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienR     : Original = 3, Reconstruit = 3, Différence = 0
Compte des LienS     : Original = 6, Reconstruit = 6, Différence = 0
```

n) Procédure séquentielle



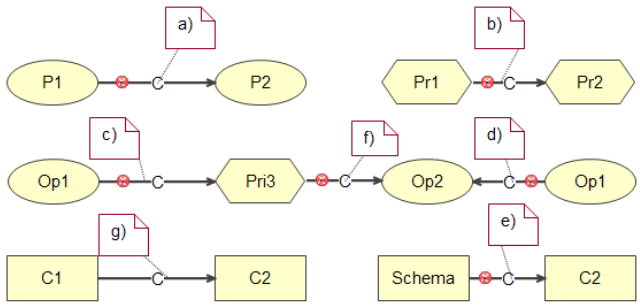

```
---------------------- BILAN --------------------------------
-------------------------------------------------------------
---------Totaux des divers composants des modèles ----------
Compte des Concepts  : Original = 0, Reconstruit = 0, Différence = 0
Compte des Procedures: Original = 8, Reconstruit = 8, Différence = 0
Compte des Principes : Original = 0, Reconstruit = 0, Différence = 0
Compte des Enonces   : Original = 0, Reconstruit = 0, Différence = 0
Compte des Traces    : Original = 0, Reconstruit = 0, Différence = 0
Compte des Exemples  : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienA     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienC     : Original = 2, Reconstruit = 2, Différence = 0
Compte des LienCm    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienI     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienIP    : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienP     : Original = 6, Reconstruit = 6, Différence = 0
Compte des LienR     : Original = 0, Reconstruit = 0, Différence = 0
Compte des LienS     : Original = 0, Reconstruit = 0, Différence = 0
```

D.4 Scénarios de tests d'incohérences

Le scénario de tests d'incohérences a pour objectif de valider l'efficacité des axiomes et les règles d'identification des incohérences qui sont détaillées à l'appendice C. Des tests cohérents et incohérents sont soumis au processus de formalisation afin d'en détecter les erreurs. Le tableau d.5 présente l'ensemble des scénarios de tests d'incohérences. Certains tests sont valides et d'autres sont non valides afin de démontrer que la détection des erreurs se réalise dans les situations appropriées. Les situations qui sont en erreur sont identifiées par des points rouges.

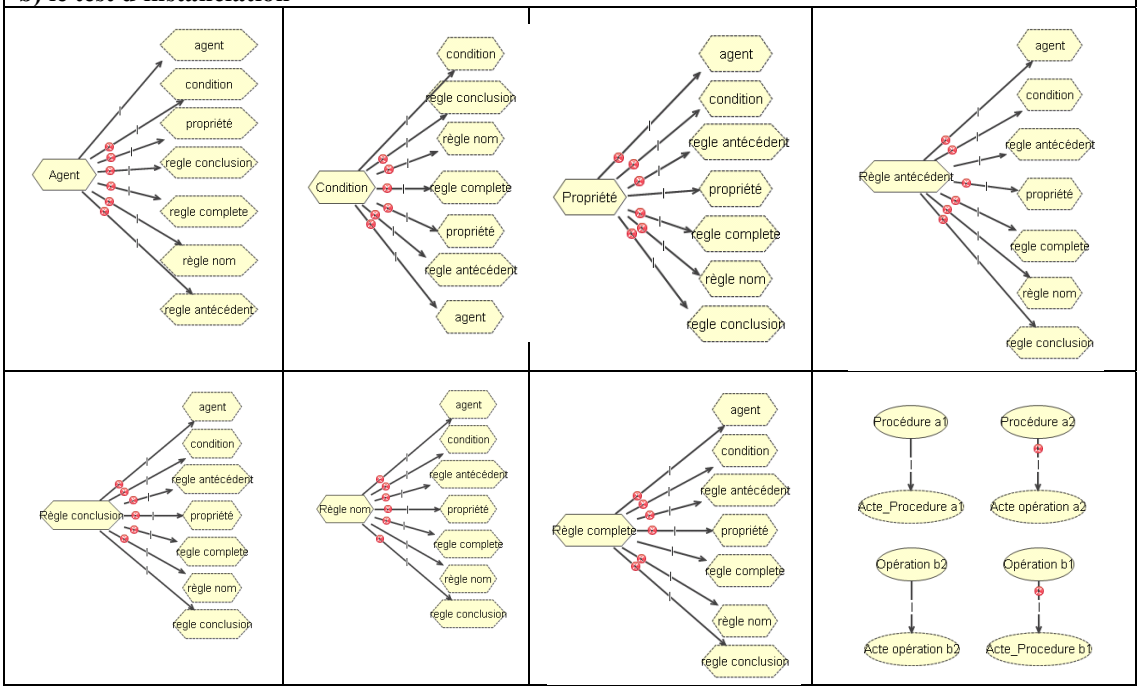
Tableau D.5
Scénarios de test d'incohérences

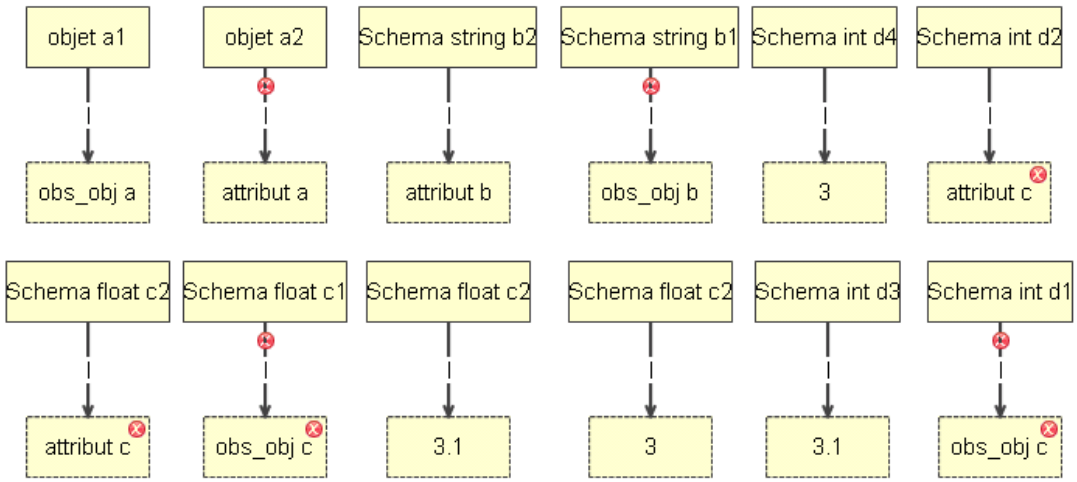
a) le test de l'attribut



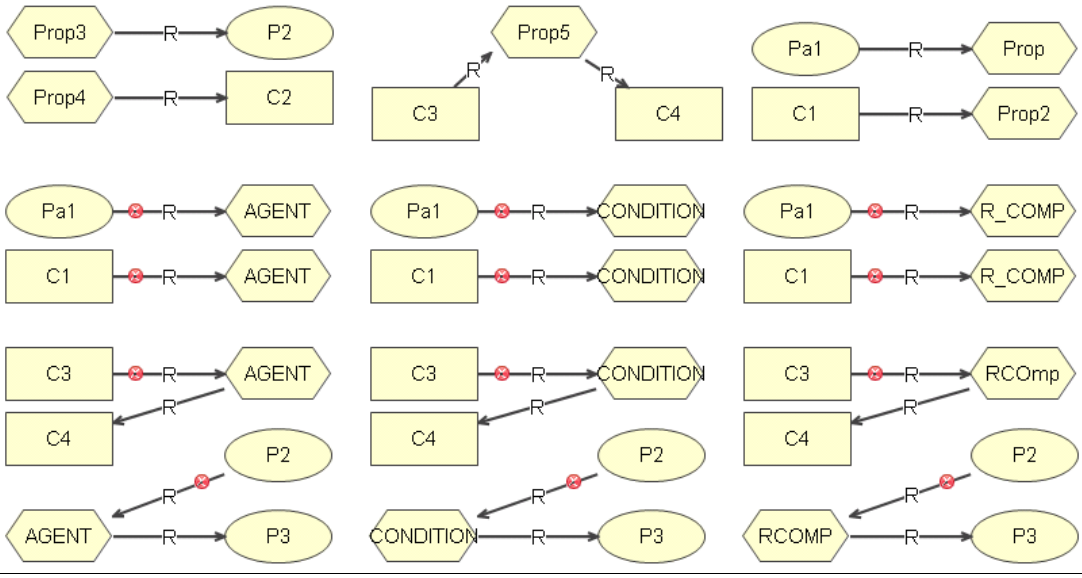
Tous les liens de composition ont été préalablement déterminés comme étant des relations d'attributions. Seul le lienC indiqué par l'étiquette **g** est valide puisque l'attribution ne peut être déterminée qu'entre deux classes.

b) le test d'instanciation

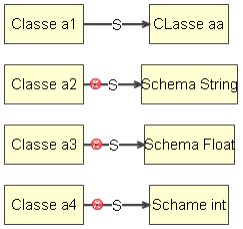


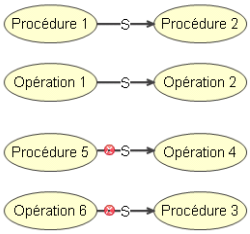


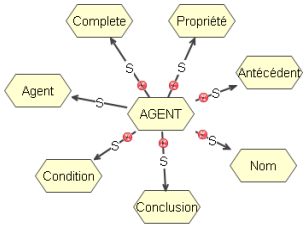
c) le test de régulation de connaissances abstraites

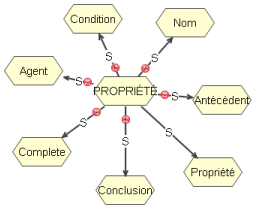


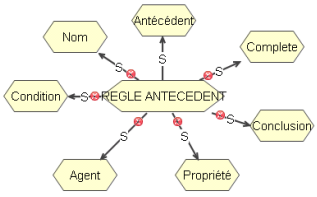
d) le test de spécialisation

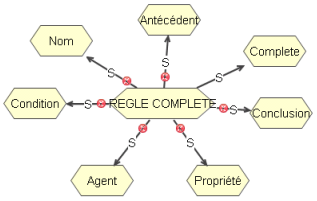


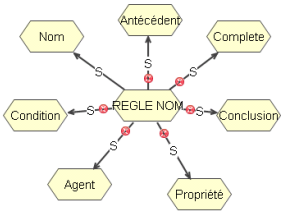


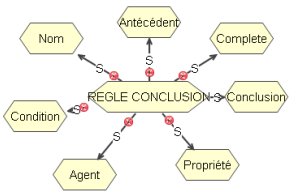


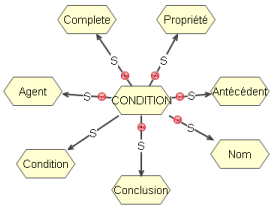












APPENDICE E

ÉLÉMENTS POUR GUIDER LA MODÉLISATION SEMI-FORMELLE À L'AIDE DE CONCEPTS ONTOLOGIQUES

Cet appendice que nous savons incomplet et qui, en soi, pourrait constituer une recherche, se veut le point départ d'une réflexion concernant les pratiques de modélisation semi-formelle pour la construction d'une ontologie. Les ouvrages suivants (Allemang et Hendler, 2008 ; Brachman et Levesque, 2004 ; Gašević *et al.*, 2006d ; Mariot *et al.*, 2008 ; Olivé, 2007 ; Paquette, 2002b) ont servi d'inspiration à la conception de ce guide. La première section expose le métamodèle du langage OWL. Les représentations graphiques de ce métamodèle utilisent le langage MOT. La présentation des métamodèles d'OWL est issue d'un exercice dont l'objectif est de présenter le vocabulaire et la grammaire du langage OWL qui nous apparaît important afin de fixer les bonnes pratiques de modélisation. Dans le cadre de ce guide, le métamodèle OWL sert de guide de référence concernant le langage ontologique utilisé par OntoCASE. La section 2 présente un compte-rendu de la notation 3 d'OWL qui est utilisée dans la thèse et la méthodologie. La section 3 présente quelques heuristiques de modélisation semi-formelle en vue de produire une ontologie que nous avons identifiées au cours de notre recherche.

E.1 Métamodèle MOT d'OWL

En ingénierie ontologique, l'*Object Management Group* (OMG) préconise l'utilisation du *Ontology Definition Metamodel* (ODM[117]) pour la définition du langage OWL.

---

[117] OMG ODM, *Ontology Definition Metamodel: OMG Adopted Specification*: http://www.omg.org/spec/ODM/1.0/Beta2/PDF/

Extrait de Đurić (2004) et Gašević *et al.*(2006a, 2006c), les figures 5.3 à 5.6 présentent le métamodèle d'OWL dans le formalisme de MOT pour la définition d'une *ressource*, d'une *classe,* d'une *propriété*, ainsi que la définition des relations entre les différents objets du métamodèle.

E.1.1 Ressource

La *ressource* « `rdfs:Resource` » est le concept de haut niveau de toute ontologie OWL. Elle représente toute chose décrite par RDFS et OWL (voir la figure e.1). Une ressource se compose d'un identifiant unique « `rdf:ID` », d'une étiquette « `rdfs:Label` » et d'un champ commentaire « `rdfs:comment` ». Le concept d'*ontologie* « `owl:Ontology` » regroupe tous les éléments de gestion de l'ontologie, tels que la version de l'ontologie « `owl:versionInfo` », la liste des imports « `owl:imports` » d'ontologies externes et leurs préfixes. La *classe* « `rdfs:Class` » regroupe deux sous-concepts: le *type de donnée* « `rdfs:datatype` » et le *concept de classe* qui est une abstraction sans représentation `rdfs`. Le *concept de propriété* est aussi une abstraction qui regroupe la catégorie d'objets servant à décrire les caractéristiques des classes. Finalement, l'*instance* est une ressource `rdfs` qui est l'objet du prédicat [`rdf:type`], dont le sujet est un [`rdfs:Class`]. Cette ressource est soit la *valeur* associée à un type de donnée ou encore l'*individu* d'une classe.

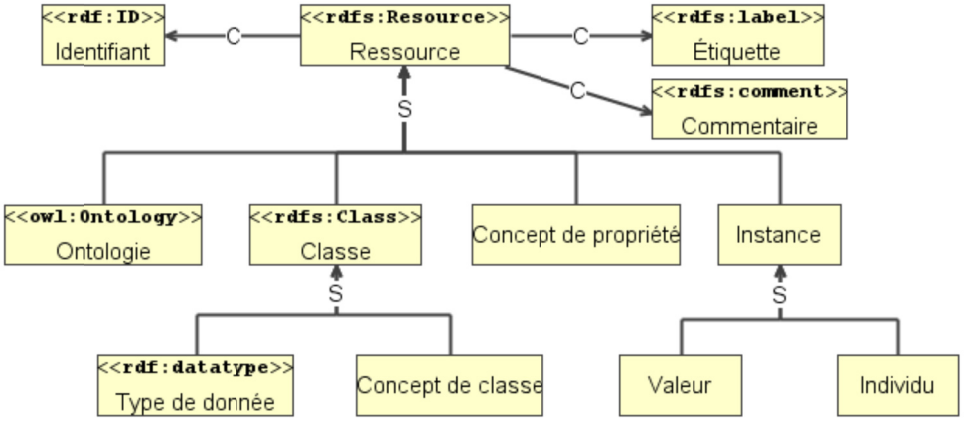


Figure E.1: Hiérarchie des concepts de haut niveau d'OWL selon ODM.

E.1.2 Concept de classe

Le concept de *classe,* utilisé en modélisation UML et en programmation orientée objets, diffère du concept de *classe* utilisé en OWL (voir la figure e.2).

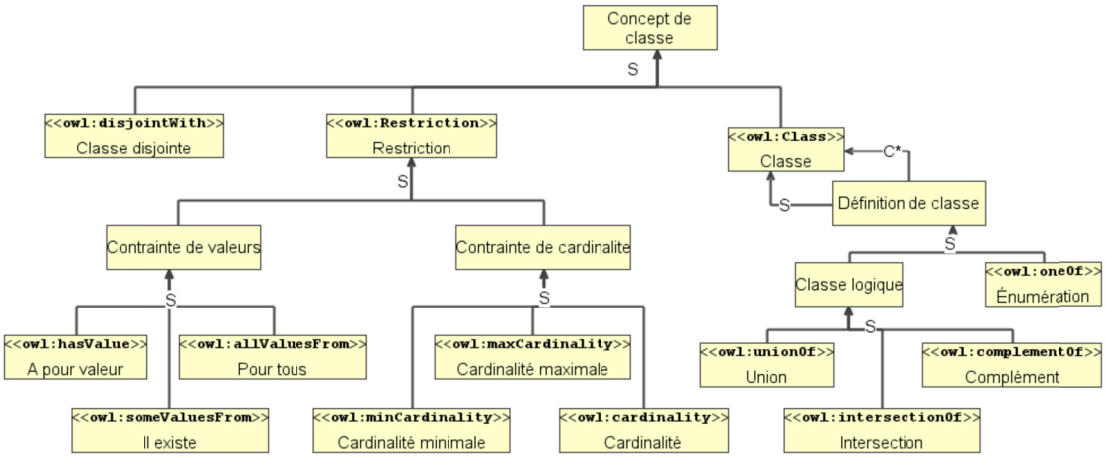


Figure E.2: Hiérarchie des classes d'OWL.

Contrairement à la classe UML, la classe OWL ne permet pas la définition d'attributs ou de relations (ces associations sont décrites par l'utilisation de la *propriété*). En OWL, une classe « `owl:Class` » se forme par le regroupement d'un ensemble d'individus (appelé l'extensionalité) ou par la création d'un identifiant (appelé intentionnalité). La disjonction « `owl:disjointWith` » entre deux classes indique qu'aucun individu ne peut être commun aux deux classes concernées. La sélection des individus d'une classe peut être réalisée par l'application d'une *restriction* « `owl:Restriction` » selon ses caractéristiques. Les restrictions sont appliquées à partir de *contraintes de valeurs* (« `owl:someValuesFrom` », « `owl:allValuesFrom` », « `owl:hasValue` ») que peuvent prendre les individus, ou encore, de la *contrainte de cardinalité* (« `owl:cardinality` », « `owl:minCardinality` », « `owl:maxCardinality` ») associée à leurs propriétés. Les individus peuvent être aussi sélectionnés à partir de *l'union* « `owl:unionOf` », ou de *l'intersection* « `owl:intersectionOf` » de deux classes. Finalement, une classe peut se former en

complémentarité « `owl:complementOf` »*:* par exemple, la classe de toutes les `Personnes` qui ne sont pas des `Universitaires`.

E.1.3 Propriété

La propriété est un objet qui sert à déterminer une association ou un attribut entre deux classes (voir la figure e.3). Contrairement à la relation, la propriété possède la caractéristique d'existence propre. En effet, la relation est un objet qui doit son existence à la classe qui en est la source alors que la propriété possède une existence indépendante de toutes classes. En OWL et en RDFS, la propriété se divise en deux classes soit: la propriété d'objet « `owl:ObjectProperty` » et la propriété de données « `owl:DatatypeProperty` ». Des *contraintes de propriété* telles que: la transitivité « `owl:TransitiveProperty` », la symétrie « `owl:SymetricProperty` », la fonctionnalité « `owl:FunctionalProperty` » et la fonctionnalité inverse « `owl:InverseFunctionalProperty` » s'appliquent aux propriétés.

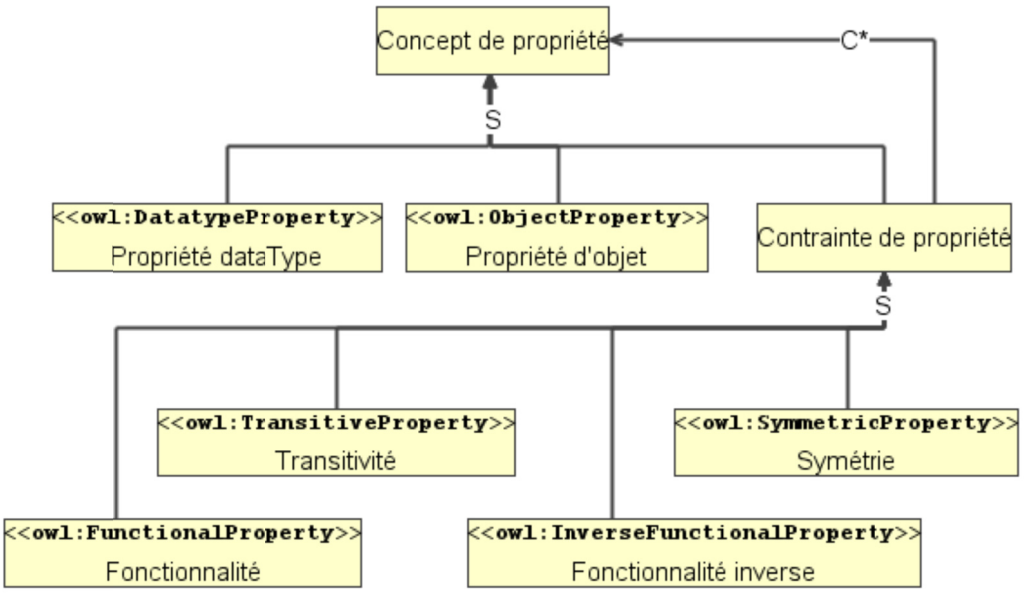


Figure E.3: Hiérarchie des propriétés d'OWL.

E.1.4 Relations

En OWL, les concepts de *propriété,* de *classe,* d'*énoncé* et d'*instances* sont reliés entre eux par des propriétés diverses. Schématisé à la figure e.4, le concept de classe peut se lier à un autre concept de classe par la relation de généralisation « `owl:subClassOf` » ou d'équivalence « `owl:equivalentClass` ». De même, pour le concept de propriété, les relations de généralisation de propriété « `owl:subPropertyOf` », d'équivalence de propriétés « `owl:equivalentProperty` » et de propriété inverse « `owl:inverseProperty` » permettent de définir des propriétés, l'une par apport à l'autre. À l'instar d'une relation qui possède un attribut source et un attribut destination qui la lie à un concept, le concept de propriété possède un *domaine* « `rdfs:domain` » et une image « `rdfs:range` » qui le lient à un concept de classe. Cependant, et contrairement à la relation, la définition d'une propriété ne nécessite pas obligatoirement l'instanciation d'une image et d'un domaine. L'*instance* est un objet `rdf` lié à un concept de classe par la propriété « `rdf:type` ». La structure d'un énoncé `rdf` est de la forme sujet/prédicat/objet dont le sujet et l'objet sont des instances.

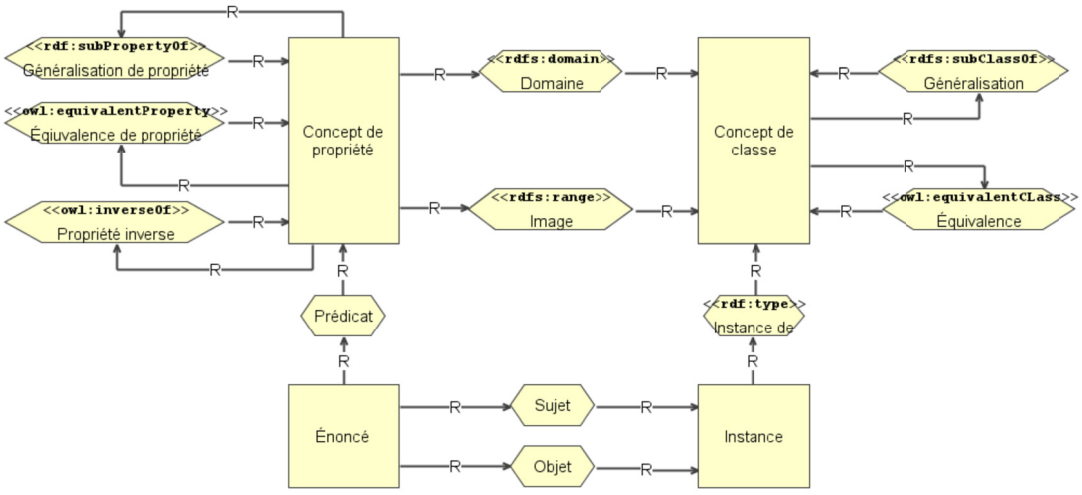


Figure E.4: Relations unissant les divers concepts d'OWL.

La figure e.5 présente un exemple en OWL-N3[118] (*Ontology Web Language Notation 3)* d'une ontologie de l'énoncé: *la pomme, qui est un fruit, est de couleur rouge.* Dans la forme sujet/prédicat/objet, la première assertion N3 de l'exemple s'interprète de la façon suivante: *le Fruit est une classe* « `Fruit/rdf:type/owl:Class` » et *le fruit est une sorte de choses* « `Fruit/rdfs:subClassOf/owl:Thing` ». La deuxième assertion N3 définit la propriété *couleur* et il s'interprète de la façon suivante: *la couleur est une propriété de type donnée* « `couleur/rdf:type/owl:DatatypeProperty` »; *la couleur a pour domaine la classe Fruit* « `couleur/rdfs:domain/Fruit` »; et *la couleur a pour image (le datatype) la chaîne de caractères* « `couleur/rdfs:range/xsd:string` ». Finalement, la troisième assertion N3 nous indique que: *la pomme est un Fruit* « `la_pomme/rdf:type/Fruit` » et que *la pomme est de couleur rouge* « `la_pomme/couleur/"rouge"` ».

Assertion 1

```
:Fruit
      a       owl:Class ;
      rdfs:subClassOf owl:Thing .
```

Assertion 2

```
:couleur
      a       owl:DatatypeProperty ;
      rdfs:domain :Fruit ;
      rdfs:range xsd:string .
```

Assertion 3

```
:la_pomme
      a       :Fruit ;
      :couleur "rouge"^^xsd:string .
```

Figure E.5: Représentation en OWL-N3 de l'énoncé: *la pomme, qui est un Fruit, est de couleur rouge.*

E.2 La notation N3

L'OWL Notation-3 (Berners-Lee, 1998) s'avère efficace et concis pour représenter un énoncé OWL de la forme sujet/prédicat/objet. Le '.' s'utilise pour délimiter les

---

[118] Berners-Lee, *Notation 3*: http://www.w3.org/DesignIssues/Notation3.html

énoncés, alors que le ';' indique un nouveau triplet contenant le même sujet/prédicat que le précédent. Quant au caractère ':' il permet de délimiter le domaine et le qualificatif d'un nom. Ainsi, `owl:Thing` désigne la classe `Thing` du domaine `owl`. Les prédicats dédiés: 'a', '=' et '=>' sont interprétés par les expressions suivantes: `rdf:type, owl:sameAs, log:implies.`

E.3 Heuristiques de modélisation

Une heuristique de modélisation est une règle empirique non formelle qui est appliquée à l'activité de modélisation. Pour l'élaboration des heuristiques de modélisation, nous avons répertorié les éléments de la polysémie de MOT, présentée en première sous-section. Par la suite, est présentée l'heuristique de modélisation selon le bon niveau d'abstraction, suivie de la représentation d'un attribut en MOT, puis de l'heuristique de la synonymie de l'agrégation.

E.3.1 Polysémie

Une conséquence très importante de l'utilisation d'un même symbole dans une multitude de contextes est qu'il y a surcharge de la sémantique symbolisée par le symbole (ce que nous avons appelé la *polysémie,* voir la section 1.1.6 p.16). Le tableau e.1 présente la polysémie que nous avons identifiée pour chacune des primitives atomiques du langage MOT, résumant ainsi les différentes significations pouvant être associées aux entités et aux relations constituant les primitives du langage MOT. Le langage semi-formel MOT différencie les types de connaissances au moyen de symboles graphiques catégorisés en entités et en relations. Les connaissances peuvent être combinées au sein d'un même diagramme de manière à produire des modèles représentant les relations entre connaissances de divers types. À l'instar des théories sur la représentation des connaissances, le langage MOT offre la possibilité de représenter des connaissances selon deux niveaux d'abstraction: conceptuel (*ConnaissanceAbstraite*) et factuel (*Fait*), ce qui, en langage des descriptions, correspond à la TBOX et la ABOX (Baader *et al.*, 2007).

Tableau E.1
Polysémie de MOT

ENTITÉ				RELATION	
Représentation en MOT		Polysémie		Représentation en MOT	Polysémie
Caonnaissacne abstraite	concept Connaissance déclarative	Classe		--C-> Composition	Est composé de
		Schéma	Entier		A pour attribution
			Caractère		
			Naturel		
	procédure Connaissance procédurale	Action		-IP-> Intrant/Produit	A pour intrant
		Opération			A pour produit
					Est l'instrument qui réalise
	principe Connaissance stratégique	Agent		--R-> Régulation	Régulation
		Propriété		--P-> Précédence	Puis exécuter
		Règle			Puis évaluer
		Condition			Permet
Connaissance factuelle	exemple Connaissance déclarative	Objet			Évaluer à partir de
		Valeur			A pour dépendance
	trace Connaissance procédurale	Acte		--I-> Instance	A pour instance
		Instruction		--S-> Spécialisation	Est une sorte de
	énoncé Connaissance stratégique	Agent		--A-> Application	A pour application
		Assertion		Icône englobement	Association avec les éléments d'un sous-modèle
		Condition			
		Règle			

E.3.2 Modéliser selon le bon niveau d'abstraction

Le tableau e.2 présente un modèle selon les deux niveaux d'abstraction, soit les niveaux conceptuel et factuel. Lors d'une activité de modélisation, le choix du niveau d'abstraction pour représenter une connaissance dépend des réponses qui sont attendues par l'interprétation du modèle (Allemang et Hendler, 2008) surtout, lorsqu'il s'agit d'une modélisation en vue de produire une ontologie interprétable par un agent

cognitif formel. Afin d'assurer la complétude[119] et la décidabilité[120] du modèle, trois questions servent de guide quand vient le choix du niveau d'abstraction pour représenter une connaissance (Mariot *et al.*, 2008): la question de la *satisfiabilité* (soit C un concept quelconque, est-ce qu'il existe des instances de C?); la question de la *vérification d'instances* (est-ce que *a* est une instance de C?) et la question de *subsomption* (est-ce que C est plus général que D?). Si l'on se réfère au tableau e.3 **a** qui est l'interprétation formelle[121] du modèle conceptuel du tableau e.2 **a**, on constate que toutes les connaissances et les relations y sont interprétées. Cependant, aucune nouvelle connaissance n'a été produite. En revanche, le modèle factuel du tableau e.2 **b** a permis la production d'une quantité importante de nouvelles connaissances (voir le tableau e.3 **b**). Ceci établi, cette démonstration n'invalide en rien la modélisation selon le niveau conceptuel. Cependant, quand on modélise, il faut garder à l'esprit que le degré de complétude que l'on désire maintenir dans le modèle et faire le choix du niveau d'abstraction en conséquence.

---

[119] Nous entendons par complétude la capacité que possède un modèle de représenter sans contradiction l'ensemble de la sémantique d'un système.

[120] Nous entendons par décidabilité la propriété que possède un modèle d'être interprété par un agent cognitif formel afin de produire des conclusions en un nombre fini d'opérations et dans un temps déterminé.

[121] L'interprétation formelle du modèle a été réalisée par le module de validation sémantique d'OntoCASE

Tableau E.2
Représenter selon le niveau d'abstraction en MOT

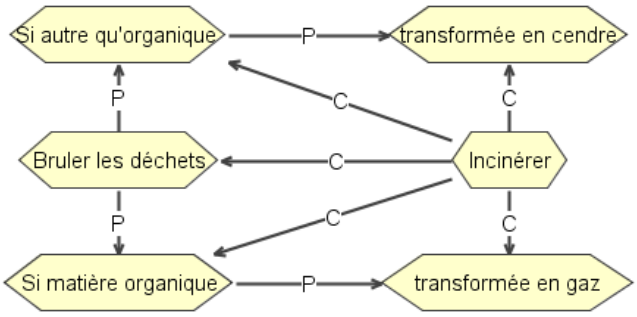
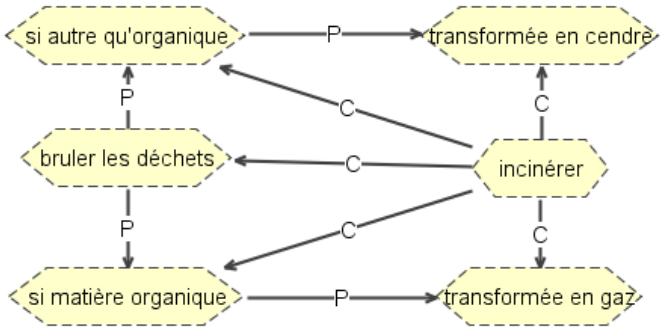
L'incinération consiste à brûler les déchets … La matière organique est alors transformée en gaz tandis que le reste des déchets devient des cendres	
Représentations possibles en MOT	a) représentation de niveau conceptuel 
	b) représentation de niveau factuel 

Tableau E.3
Interprétation formelle de la règle d'incinération des déchets

a) interprétation du niveau conceptuel

```
----------Début du processus de validation sémantique-------------
Bruler les déchets est antécédent de Si autre qu'organique
Bruler les déchets est antécédent de Si matière organique
Incinérer se compose de Bruler les déchets
Incinérer se compose de Si autre qu'organique
Incinérer se compose de Si matière organique
Incinérer se compose de transformée en cendre
Incinérer se compose de transformée en gaz
Si autre qu'organique  a pour conclusion transformée en cendre
Si matière organique a pour conclusion transformée en gaz
------------------Résultats après inférence-----------------------
```

b) interprétation du niveau factuel

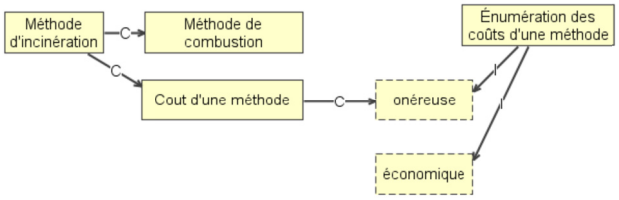
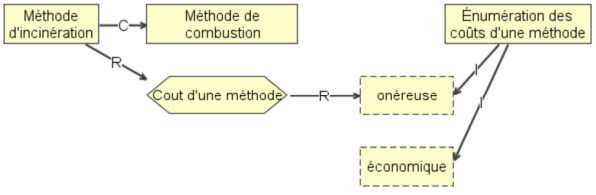
```
[bruler les déchets] est une partie de [incinérer]
[bruler les déchets] permet [si autre qu'organique]
[bruler les déchets] permet [si matière organique]
[bruler les déchets] permet [transformée en cendre]
[bruler les déchets] permet [transformée en gaz]
[si autre qu'organique] a pour antécédent [bruler les déchets]
[si autre qu'organique] a pour dépendance [bruler les déchets]
[si autre qu'organique] alors [transformée en cendre]
[si autre qu'organique] est une partie de [incinérer]
[si autre qu'organique] permet [transformée en cendre]
[si matière organique] a pour antécédent [bruler les déchets]
[si matière organique] a pour dépendance [bruler les déchets]
[si matière organique] alors [transformée en gaz]
[si matière organique] est une partie de [incinérer]
[si matière organique] permet [transformée en gaz]
[transformée en cendre] a pour dépendance [bruler les déchets]
[transformée en cendre] a pour dépendance [si autre qu'organique]
[transformée en cendre] est la conclusion de [si autre qu'organique]
[transformée en cendre] est une partie de [incinérer]
[transformée en gaz] a pour dépendance [bruler les déchets]
[transformée en gaz] a pour dépendance [si matière organique]
[transformée en gaz] est la conclusion de [si matière organique]
[transformée en gaz] est une partie de [incinérer]
```

E.3.3 Représentation d'un attribut

En langage MOT, l'association d'un *attribut* à un concept ne fait pas partie des primitives atomiques de MOT. Il existe tout de même deux façons semi-formelles de représenter un attribut. La première, qui est présentée au tableau e.4 **a**, utilise un concept associé par un lien C à un concept symbolisant l'attribut. Dans l'exemple en **a,** le concept « coût de la méthode » est un attribut du concept « Méthode d'incinération » alors que le concept « Méthode de combustion » symbolise une composante de la « Méthode d'incinération ». Dans cette façon de représenter l'attribut, il n'existe aucune manière formelle de distinguer la composition et l'attribut.

L'exemple du tableau e.4 **b** présente une alternative un peu plus formelle pour représenter un attribut. L'utilisation du principe en tant que propriété de type "dataType" permet de faire une distinction claire entre la composition de concepts et l'assignation d'un attribut à un concept.

Tableau E.4
Représenter un attribut en MOT

L'incinération, qui comprend une méthode de combustion, est la méthode la plus onéreuse	
Représentations possibles en MOT	a) attribut par l'utilisation d'un concept lié par un lien C 
	b) attribut par l'utilisation d'un principe 

E.3.4 Désambiguïser la synonymie d'agrégation

La synonymie d'agrégation intervient lorsqu'une situation transporte l'idée d'agglomérat ou de collection d'objets qui se représente de quatre façons: soit par une composition selon le niveau d'abstraction factuelle (voir le tableau e.5 **a**), soit par le changement de niveau d'abstraction (voir le tableau e.5 **b**), soit par l'utilisation de la composition au niveau d'abstraction conceptuel (voir le tableau e.5 **c**) ou encore par l'utilisation de la spécialisation. L'interprétation automatique du modèle (voir le tableau e.6) et la validation par la complétude (voir la section E.3.2) sont des moyens mis à la disposition du concepteur afin qu'il ajuste convenablement le modèle pour que sa sémantique soit conforme à la réalité à représenter.

Tableau E.5
Représenter une agrégation en MOT

L'élimination des déchets se fait de deux façons principales: l'incinération et l'enfouissement

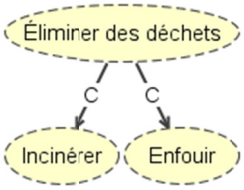
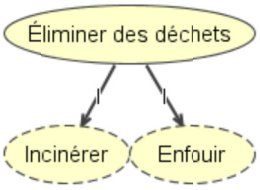
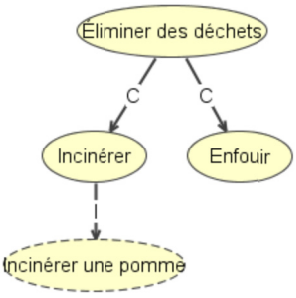
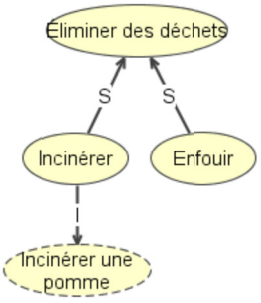
Représentations possibles en MOT	
a) 	b) 
c) 	d) 

Tableau E.6
Interprétation formelle des modèles semi-formels du tableau e.5

a) [Bruler] est un (:Éliminer de déchets) [Incinérer] est un (:Éliminer de déchets)
b) ----------Début du processus de validation sémantique---------- [Éliminer de déchets] se compose de [enfouir] [Éliminer de déchets] se compose de [incinérer] ------------------Résultats après inférence---------- [enfouir] est une partie de [Éliminer de déchets] [incinérer] est une partie de [Éliminer de déchets]
c) ----------Début du processus de validation sémantique---------- [incinérer une pomme] est un (:Incinérer) Éliminer des déchets se compose de Incinérer Éliminer des déchets se compose de Enfouir ------------------Résultats après inférence---------- [incinérer une pomme] est un (:Incinérer)
d) ----------Début du processus de validation sémantique---------- Incinérer est une sorte de Éliminer des déchets [Incinérer une pomme] est un (:Incinérer) Enfouir est une sorte de Éliminer des déchets ------------------Résultats après inférence---------- [Incinérer une pomme] est un (:Incinérer :Éliminer des déchets)

Voici comment appliquer le test de complétude par le questionnement s'y rapportant pour chacun des cas (voir le tableau e.7). On applique à chaque cas, les tests de satisfiabilité, de vérification d'instance et de subsomption. À chacun de ces tests, on tire un ensemble de conclusions en comparant l'interprétation automatique à la sémantique que le modèle doit représenter. On choisira le modèle en fonction de la conclusion qui démontre le plus de vraisemblance entre l'interprétation formelle du modèle et la sémantique du domaine que le modèle doit représenter. Pour le présent exemple, le modèle du cas **d** est celui que nous identifions comme étant le plus représentatif de la sémantique du domaine.

Tableau E.7
Test de complétude pour les modèles du tableau e.5

	satisfiabilité est-ce qu'il existe des instances de C?	*vérification d'instances* est-ce que *a* est une instance de C?	*subsomption* est-ce que C est plus général que D?
Cas **a**	Ce cas est non-satisfait puisqu'il n'y a aucun concept de déclaré.	Ce cas est non-satisfait puisqu'il n'y a aucun concept de déclaré.	Cette question est non applicable pour un modèle factuel (qui ne contient que des faits).
Cas **b**	Ce cas est satisfait puisque *incinérer* et *enfouir* sont des instances *d'éliminer les déchets.*	Toutes les instances du modèle sont des instances d'une classe	N'est pas applicable car il n'y a qu'un seul concept.
Cas **c**	A priori, non. Cependant, l'ajout du fait *incinérer une pomme* permet de valider la fiabilité du concept *incinérer.*	À la lecture de l'interprétation automatique on détecte un problème avec l'interprétation selon la sémantique du domaine. Le fait *incinérer une pomme*, selon la sémantique du domaine, est une façon d'éliminer un déchet. Or, l'interprétation automatique n'indique pas de relation d'instanciation entre *incinérer une pomme* et *éliminer des déchets,* ce qui est une erreur.	N'est pas applicable car il n'y a qu'un seul concept.
Cas **d**	Ici aussi on ajoute un fait pour assurer la satisfiabilité.	Pour ce cas, la lecture de l'interprétation automatique est en concordance avec la sémantique du domaine.	N'est pas applicable car il n'y a qu'un seule concept.

E.3.5 Résumé

Nous avons vu dans cet appendice quelques heuristiques de modélisation semi-formelle en vue de produire une ontologie. Après avoir assimilé le métamodèle d'OWL et la polysémie du langage MOT, les heuristiques guident le concepteur dans le choix du bon niveau d'abstraction pour la modélisation d'un domaine, dans la façon de représenter un attribut et dans le choix du meilleur modèle pour désambiguïser la synonymie d'agrégation selon le test de complétude.

APPENDICE F

CODE SOURCE

F.1 Base de règles de la désambiguïsation typologique

```
#Rule-11-a_holonyme_entre_deux_enonces_Agents_Condition
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Enonce(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Condition)  ^
          ot:OT_EntiteEstDeType(?src,
oAmbig:Observable_AgentNormeContrainte)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",       "OBS_AGENT(%s)       -HOLONYME->
OBS_CONDITION(%s)", ?nomSrc, ?nomDest)
#Rule-00-k_desamb_lienA_ND_Concept
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "ND(%s)   -A->OBJET(%s)",   ?nomSrc,
?nomDest)
#Rule-17-d_LienC_entre_EnonceEtTrace
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Enonce(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          swrlbi:invoker("printf", "ND_ENONCE(%s) -HOLONYME> ND_TRACE(%s)",
?nomSrc, ?nomDest)
#Rule-03-aa_desamb_exemple_Concept_Instance
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",       "OBJET(%s)       -INSTANCIATION->
OBSERVABLE_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-02-c_desamb_lienS_entre_Principes
      metaMot:MOT_LienS(?ls)  ^
      metaMot:MOT_connSource(?ls, ?src)  ^
      metaMot:MOT_connDestination(?ls, ?dest)  ^
      oAmbig:TA_estNonDetermine(?ls, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?ls, oAmbig:Relation_Specialisation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Agent_Contrainte_Norme)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
          oAmbig:TA_estNonDetermine_typo(?ls, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",       "AGENT(%s)       -SPECIALISTATION->
AGENT(%s)", ?nomSrc, ?nomDest)
#Rule-14-a_Procedure-P-Procedure_Procedure-Procedure
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
```

```
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "PROCEDURE(%s)   -P->   PROCEDURE(%s)",
?nomSrc, ?nomDest)
#Rule-15-c_Principe-P-Principe_Condition-Condition
     metaMot:MOT_LienP(?lp)  ^
     metaMot:MOT_connSource(?lp, ?src)  ^
     metaMot:MOT_connDestination(?lp, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lp, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     metaMot:MOT_Principe(?src)  ^
     metaMot:MOT_Principe(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Condition)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Condition)  ^
         oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",   "CONDITION(%s)   -P->   CONDITION(%s)",
?nomSrc, ?nomDest)
#Rule-XX-00_Assignation_des_connSource_et_connDest_a_une_Propriete
     metaMot:MOT_Principe(?p)  ^
     ot:OT_EntiteEstDeType(?p, oAmbig:Propriete)  ^
     metaMot:MOT_estLaSource(?p, ?lrSrc)  ^
     metaMot:MOT_estLaDestination(?p, ?lrDest)  ^
     metaMot:MOT_connSource(?lrDest, ?connSrc)  ^
     metaMot:MOT_connDestination(?lrSrc, ?connDest)  ^
     ot:OT_EntiteEstDeType(?connSrc, oAmbig:Concept_Classe)  ^
     ot:OT_EntiteEstDeType(?connDest, oAmbig:Concept_Classe)
     ->  oRef:OR_connSource(?p, ?connSrc)  ^
         oRef:OR_connDestination(?p, ?connDest)  ^
         swrlbi:invoker("printf", " ")
#Rule-17-b_LienC_entre_deux_enonces
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, true)  ^
     oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Enonce(?src)  ^
     metaMot:MOT_Enonce(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
         oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
         swrlbi:invoker("printf",        "ND_Enonce(%s)        -HOLONYME->
ND_Enonce(%s)", ?nomSrc, ?nomDest)
#Rule-10-ca_holonyme_entre_un_agents_condition
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, false)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
     metaMot:MOT_Principe(?src)  ^
     metaMot:MOT_Principe(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Condition)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",  "AGENT(%s)  -HOLONYME->  CONDITION(%s)",
?nomSrc, ?nomDest)
#Rule-XX-00_Assignation_de_connSource_a_une_Propriete
     metaMot:MOT_Principe(?p)  ^
     ot:OT_EntiteEstDeType(?p, oAmbig:Propriete_Unaire)  ^
     metaMot:MOT_estLaDestination(?p, ?lrDest)  ^
     metaMot:MOT_connSource(?lrDest, ?connSrc)  ^
     ot:OT_EntiteEstDeType(?connSrc, oAmbig:Concept_Classe)
     ->  oRef:OR_connSource(?p, ?connSrc)  ^
         swrlbi:invoker("printf", " ")
#Rule-03-J_desamb_LienI_ND_Condition_OBS_CONDITION
     metaMot:MOT_LienI(?li)  ^
     metaMot:MOT_connSource(?li, ?src)  ^
     metaMot:MOT_connDestination(?li, ?dest)  ^
     oAmbig:TA_estNonDetermine(?li, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Principe(?src)  ^
     metaMot:MOT_Enonce(?dest)  ^
     ot:OT_EntiteEstDeType(?src, oAmbig:Condition)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Condition)  ^
         oAmbig:TA_estNonDetermine_typo(?li, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",    "ND_CONDITION(%s)    -INSTANCIATION->
OBS_CONDITION(%s)", ?nomSrc, ?nomDest)
#Rule-17-a_Enonce-LienC-Trace-RegleComplete_Procedure
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     metaMot:MOT_Enonce(?src)  ^
     metaMot:MOT_Trace(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Regle_Complete)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
         oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",       "OB_REGLE_COMPLETE(%s)       -C->
OB_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-05-b_desamb_Relation_A-POUR-PRODUIT_observables
     metaMot:MOT_LienIP(?l_ip)  ^
     metaMot:MOT_connSource(?l_ip, ?src)  ^
     metaMot:MOT_connDestination(?l_ip, ?dest)  ^
     oAmbig:TA_estNonDetermine(?l_ip, true)  ^
```

```
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Trace(?src)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?l_ip, oAmbig:Produit)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?l_ip, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",    "OBSERVABLE_PROCEDURE(%s)    -A-POUR-
PRODUIT-> OBSERVABLE_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-00-e_desamb_lienA_Exemple_ND
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      metaMot:MOT_Exemple(?src)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?la, oAmbig:Relation_Application)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?la, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",  "OBS_OBJET(%s)  -A->  ND_ABSTRAIT(%s)",
?nomSrc, ?nomDest)
#Rule-11-e_Attribut_Concept_Schema_String
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, false)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      ot:OT_EntiteEstDeType(?dest, oAmbig:Entite_Schema_String)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Attribut)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",   "OBJET(%s)   -HOLONYME->   OBJET(%s)",
?nomSrc, ?nomDest)
#Rule-03-g_desamb_LienI_Principe_OBS_AGENT
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      oAmbig:TA_estNonDetermine(?li, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          ot:OT_EntiteEstDeType(?dest,
oAmbig:Observable_AgentNormeContrainte)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "ND_STRATEGIQUE(%s)   -INSTANCIATION->
OBS_AGENT(%s)", ?nomSrc, ?nomDest)
#Rule-00_ConnaissancesNonAmbigue_LienP
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          swrlbi:invoker("printf",          "ND_PROCEDURE(%s)          -P->
ND_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-03-f_desamb_LienI_Schema_String_Valeur
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Entite_Schema_String)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Attribut_Valeur)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "SCHEMA_STRING(%s)   -A   POUR   VALEUR-
ATTRIBUT_VALEUR(%s)", ?nomSrc, ?nomDest)
#Rule-10-c_holonyme_entre_un_agents_condition
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Condition)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "AGENT(%s)  -HOLONYME->  CONDITION(%s)",
?nomSrc, ?nomDest)
#Rule-14-b_Trace-P-Trace_Procedure-Procedure
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Trace(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
```

```
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",          "OB_PROCEDURE(%s)          -P->
OB_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-07-a_desamb_LienR_Enonce_Exemple
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Enonce(?src)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?src,
oAmbig:Observable_AgentNormeContrainte)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",      "OBSERVABLE_AGENT(%s)      -REGIT->
OBSERVABLE_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-03-b_desamb_LienI_Principe_Enonce
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      oAmbig:TA_estNonDetermine(?li, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          swrlbi:invoker("printf",   "ND_STRATEGIQUE(%s)   -INSTANCIATION->
ND_STRAG_OBS(%s)", ?nomSrc, ?nomDest)
#Rule-00-i_desamb_lienA_ND_Procedure
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "ND(%s)  -A->PROCEDURE(%s)",  ?nomSrc,
?nomDest)
#Rule-XX-22-2-3_desamb_LienR_Exemple_Enonce
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Exemple(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?dest,
oAmbig:Observable_AgentNormeContrainte)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", "Source=%s Dest=%s", ?nomSrc, ?nomDest)
#Rule-03-d_desamb_LienI_Schema_Int_Valeur
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Entite_Schema_Int)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Attribut_Valeur)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "SCHEMA_INT(%s)   -A   POUR   VALEUR-
ATTRIBUT_VALEUR(%s)", ?nomSrc, ?nomDest)
#Rule-17-e_Enonce-P-Enonce_Antecedent-Conclusion
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Regle_Antecedent)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?dest,  oAmbig:Observable_Regle_Conclusion)
^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",          "OB_ANTECEDENT(%s)          -P->
OB_CONCLUSION(%s)", ?nomSrc, ?nomDest)
#Rule-17-c_Enonce-P-Trace_Antecedent-Operation
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Regle_Antecedent)  ^
      metaMot:MOT_Trace(?dest)  ^
```

```
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Operation)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",          "OB_ANTECEDENT(%s)          -P->
OB_OPERATION(%s)", ?nomSrc, ?nomDest)
#Rule-20-a_desambig_RegleNom_LienR_Concept
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Regle_Nom)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "REGLE_NOM(%s)   -REGIT->   OBJET(%s)",
?nomSrc, ?nomDest)
#Rule-10-ba_holonyme_entre_deux_procedures
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, false)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",        "PROCEDURE(%s)        -HOLONYME->
PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-05-a_desamb_Relation_A-POUR-INTRANT_observables
      metaMot:MOT_LienIP(?l_ip)  ^
      metaMot:MOT_connSource(?l_ip, ?src)  ^
      metaMot:MOT_connDestination(?l_ip, ?dest)  ^
      oAmbig:TA_estNonDetermine(?l_ip, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Exemple(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?l_ip, oAmbig:Intrant)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Objet)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?l_ip, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",    "OBSERVABLE_PROCEDURE(%s)    -A-POUR-
INTRANT-> OBSERVABLE_OBJET(%s)", ?nomDest, ?nomSrc)
#Rule-06-a_desamb_LienR_Principe_Concept
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", "AGENT(%s) -REGIT-> OBJET(%s)", ?nomSrc,
?nomDest)
#Rule-11-c_holonyme_entre_deux_exemples
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Exemple(?src)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",        "OBS_OBJET(%s)        -HOLONYME->
OBS_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-00-h_desamb_lienA_ND_Trace
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", "ND(%s) -A->OBS_PROCEDURE(%s)", ?nomSrc,
?nomDest)
#Rule-00-d_desamb_lienA_Enonce_ND
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
```

```
     oAmbig:TA_estNonDetermine(?la, true)  ^
     metaMot:MOT_Enonce(?src)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?la, oAmbig:Relation_Application)  ^
         ot:OT_EntiteEstDeType(?src,
oAmbig:Observable_AgentNormeContrainte)  ^
         oAmbig:TA_estNonDetermine_typo(?la, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         swrlbi:invoker("printf",  "OBD_AGENT(%s)  -A->  ND_ABSTRAIT(%s)",
?nomSrc, ?nomDest)
#Rule-10-aa_holonyme_entre_deux_concepts
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, false)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
     metaMot:MOT_Concept(?src)  ^
     metaMot:MOT_Concept(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",   "OBJET(%s)   -HOLONYME->   OBJET(%s)",
?nomSrc, ?nomDest)
#Rule-XX-05-2-3_holonyme_concept_exemple
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Concept(?src)  ^
     metaMot:MOT_Exemple(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
         oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf", "Source=%s Dest=%s", ?nomSrc, ?nomDest)
#Rule-00-j_desamb_lienA_ND_Enonce
     metaMot:MOT_LienA(?la)  ^
     metaMot:MOT_connSource(?la, ?src)  ^
     metaMot:MOT_connDestination(?la, ?dest)  ^
     oAmbig:TA_estNonDetermine(?la, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Enonce(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?dest,
oAmbig:Observable_AgentNormeContrainte)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",  "ND(%s)  -A->OBS_AGENT(%s)",  ?nomSrc,
?nomDest)
#Rule-02-a_desamb_lienS-entre_Concepts
     metaMot:MOT_LienS(?ls)  ^
     metaMot:MOT_connSource(?ls, ?src)  ^
     metaMot:MOT_connDestination(?ls, ?dest)  ^
     oAmbig:TA_estNonDetermine(?ls, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Concept(?src)  ^
     metaMot:MOT_Concept(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?ls, oAmbig:Relation_Specialisation)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
         oAmbig:TA_estNonDetermine_typo(?ls, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",      "OBJET(%s)      -SPECIALISTATION->
OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-XX-07-2-1_Principe-LienC-Procedure-RegleComplete_Procedure
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     metaMot:MOT_Principe(?src)  ^
     metaMot:MOT_Procedure(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Regle_Complete)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
         oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf", "REGLE_COMPLETE(%s) -C-> PROCEDURE(%s)",
?nomSrc, ?nomDest)
#Rule-07-c_desamb_LienR_Enonce_Enonce
     metaMot:MOT_LienR(?lr)  ^
     metaMot:MOT_lrConnSrc(?lr, ?src)  ^
     metaMot:MOT_lrConnDest(?lr, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lr, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     metaMot:MOT_Enonce(?src)  ^
     metaMot:MOT_Enonce(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
         ot:OT_EntiteEstDeType(?dest,
oAmbig:Observable_AgentNormeContrainte)  ^
         ot:OT_EntiteEstDeType(?src,
oAmbig:Observable_AgentNormeContrainte)  ^
         oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
```

```
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",      "OBSERVABLE_AGENT(%s)      -REGIT->
OBSERVABLE_AGENT(%s)", ?nomSrc, ?nomDest)
#Rule-02-b_desamb_lienS_entre_Procedures
     metaMot:MOT_LienS(?ls)  ^
     metaMot:MOT_connSource(?ls, ?src)  ^
     metaMot:MOT_connDestination(?ls, ?dest)  ^
     oAmbig:TA_estNonDetermine(?ls, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Procedure(?src)  ^
     metaMot:MOT_Procedure(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?ls, oAmbig:Relation_Specialisation)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
         oAmbig:TA_estNonDetermine_typo(?ls, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",    "PROCEDURE(%s)     -SPECIALISTATION->
PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-03-e_desamb_LienI_Schema_Float_Valeur
     metaMot:MOT_LienI(?li)  ^
     metaMot:MOT_connSource(?li, ?src)  ^
     metaMot:MOT_connDestination(?li, ?dest)  ^
     ot:OT_EntiteEstDeType(?src, oAmbig:Entite_Schema_Float)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Attribut_Valeur)  ^
         oAmbig:TA_estNonDetermine_typo(?li, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",   "SCHEMA_FLOAT(%s)   -A   POUR   VALEUR-
ATTRIBUT_VALEUR(%s)", ?nomSrc, ?nomDest)
#Rule-XX-04-2-3c_holonyme_entre_un_enonces_etUneRegleNom
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?src)  ^
     metaMot:MOT_connDestination(?lc, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lc, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, false)  ^
     metaMot:MOT_Enonce(?src)  ^
     metaMot:MOT_Enonce(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
         ot:OT_EntiteEstDeType(?src,
oAmbig:Observable_AgentNormeContrainte)  ^
         oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         swrlbi:invoker("printf",  "L'agent  %s  composé  d'une  regle  %s
(Contexte %s -C-> %s)", ?nomSrc, ?nomDest, ?nomSrc, ?nomDest)
#Rule-03-h_desamb_LienI_AGENT_OBS_ND
     metaMot:MOT_LienI(?li)  ^
     metaMot:MOT_connSource(?li, ?src)  ^
     metaMot:MOT_connDestination(?li, ?dest)  ^
     oAmbig:TA_estNonDetermine(?li, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     metaMot:MOT_Principe(?src)  ^
     metaMot:MOT_Enonce(?dest)  ^
     ot:OT_EntiteEstDeType(?dest,   oAmbig:Observable_AgentNormeContrainte)
^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
         oAmbig:TA_estNonDetermine_typo(?li, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         swrlbi:invoker("printf",      "ND_AGENT(%s)      -INSTANCIATION->
OBS_AGENT(%s)", ?nomSrc, ?nomDest)
#Rule-07-b_desamb_LienR_Enonce_Trace
     metaMot:MOT_LienR(?lr)  ^
     metaMot:MOT_lrConnSrc(?lr, ?src)  ^
     metaMot:MOT_lrConnDest(?lr, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lr, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     metaMot:MOT_Enonce(?src)  ^
     metaMot:MOT_Trace(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
         ot:OT_EntiteEstDeType(?src,
oAmbig:Observable_AgentNormeContrainte)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
         swrlbi:invoker("printf", "Source=%s Dest=%s", ?nomSrc, ?nomDest)
^
         oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf",      "OBSERVABLE_AGENT(%s)      -REGIT->
OBSERVABLE_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-XX-06-b_desamb_LienR_Concept_Principe
     metaMot:MOT_LienR(?lr)  ^
     metaMot:MOT_lrConnSrc(?lr, ?src)  ^
     metaMot:MOT_lrConnDest(?lr, ?dest)  ^
     oAmbig:TA_estNonDetermine(?lr, true)  ^
     oAmbig:TA_estNonDetermine(?src, true)  ^
     oAmbig:TA_estNonDetermine(?dest, true)  ^
     oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
     oAmbig:TA_estNonDetermine_topo(?src, true)  ^
     oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
     metaMot:MOT_Concept(?src)  ^
     metaMot:MOT_Principe(?dest)  ^
     metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?dest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
         ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?dest, oAmbig:Propriete_Unaire)  ^
         oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
         oAmbig:TA_estNonDetermine_typo(?src, false)  ^
         oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
         swrlbi:invoker("printf", "Source=%s Dest=%s", ?nomSrc, ?nomDest)
#Rule-15-b_Principe-P-Procedure_Condition-Procedure
     metaMot:MOT_LienP(?lp)  ^
     metaMot:MOT_connSource(?lp, ?src)  ^
```

```
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Condition)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "CONDITION(%s)  -P->  PROCEDURE(%s)",
?nomSrc, ?nomDest)
#Rule-21-a_desamb_LienR_RegleNomObs_Exemple
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Regle_Nom)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "OBSERVABLE_REGLE_NOM(%s)  -REGIT->
OBSERVABLE_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-06-c_desamb_LienR_Principe_Principe
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Agent_Contrainte_Norme)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", "AGENT(%s) -REGIT-> AGENT(%s)", ?nomSrc,
?nomDest)
#Rule-15-e_Enonce-P-Trace_Condition-Procedure
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      metaMot:MOT_Enonce(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Condition)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "OB_CONDITION(%s)  -P->
OB_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-11-e_Attribut_Concept_Schema_Float
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, false)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      ot:OT_EntiteEstDeType(?dest, oAmbig:Entite_Schema_Float)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Attribut)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",  "OBJET(%s)  -HOLONYME->  OBJET(%s)",
?nomSrc, ?nomDest)
#Rule-00-b-desamb_LienI
      metaMot:MOT_LienI(?li)  ^
      oAmbig:TA_estNonDetermine(?li, true)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          swrlbi:invoker("printf",  "ND_OBJET(%s)  -I->  ND_OBSERVABLE(%s)",
?nomSrc, ?nomDest)
#Rule-00-k_desamb_lienA_ND_Principe
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Agent_Contrainte_Norme)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "ND(%s)  -A->AGENT(%s)",  ?nomSrc,
?nomDest)
#Rule-11-d_Attribut_Concept_Schema_Int
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
```

```
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, false)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      ot:OT_EntiteEstDeType(?dest, oAmbig:Entite_Schema_Int)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Attribut)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",   "OBJET(%s)   -HOLONYME->   OBJET(%s)",
?nomSrc, ?nomDest)
#Rule-16-a_Desamb_LienC_Entre_Principe-et-inconnu
      metaMot:MOT_LienC(?lc)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          swrlbi:invoker("printf",       "MD_Principe(%s)       -HOLONYME->
ENTITE_ND(%s)", ?nomSrc, ?nomDest)
#Rule-03-a_desamb_LienI_Concept_Exemple-OBJET
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      oAmbig:TA_estNonDetermine(?li, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",       "OBJET(%s)       -INSTANCIATION->
OBSERVABLE_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-16-c_Principe-P-Procedure_Antecedent-Operation
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Regle_Antecedent)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Operation)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "ANTECEDENT(%s)  -P->  OPERATION(%s)",
?nomSrc, ?nomDest)
#Rule-03-c_desamb_LienI_Procedure_Trace-PROCEDURE
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      oAmbig:TA_estNonDetermine(?li, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",     "PROCEDURE(%s)      -INSTANCIATION->
OBSERVABLE_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-00-f_desamb_lienA_Trace_ND
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      metaMot:MOT_Trace(?src)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?la, oAmbig:Relation_Application)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?la, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",         "OBS_PROCEDURE(%s)         -A->
ND_ABSTRAIT(%s)", ?nomSrc, ?nomDest)
#Rule-20-b_desambig_RegleNom_LienR_Procedure
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Regle_Nom)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "REGLE_NOM(%s)  -REGIT->  PROCEDURE(%s)",
?nomSrc, ?nomDest)
#Rule-04-a_desamb_Relation_A-POUR-INTRANT
      metaMot:MOT_LienIP(?l_ip)  ^
      metaMot:MOT_connSource(?l_ip, ?src)  ^
      metaMot:MOT_connDestination(?l_ip, ?dest)  ^
      oAmbig:TA_estNonDetermine(?l_ip, true)  ^
```

```
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?l_ip, oAmbig:Intrant)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?l_ip, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",     "PROCEDURE(%s)     -A-POUR-INTRANT->
OBJET(%s)", ?nomDest, ?nomSrc)
#Rule-15-d_Enonce-P-Enonce_Condition-Condition
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      metaMot:MOT_Enonce(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Condition)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Condition)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",          "OB_CONDITION(%s)          -P->
OB_CONDITION(%s)", ?nomSrc, ?nomDest)
#Rule-11-d_Attribut_entre_Enonces
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, false)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?lc, oAmbig:Attribut)  ^
      metaMot:MOT_Exemple(?src)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",     "OBS_OBJET(%s)     -A_POUR_ATTRIBUR->
OBS_OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-04-b_desamb_Relation_A-POUR-PRODUIT
      metaMot:MOT_LienIP(?l_ip)  ^
      metaMot:MOT_connSource(?l_ip, ?src)  ^
      metaMot:MOT_connDestination(?l_ip, ?dest)  ^
      oAmbig:TA_estNonDetermine(?l_ip, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?l_ip, oAmbig:Produit)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?l_ip, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",     "PROCEDURE(%s)     -A-POUR-PRODUIT->
OBJET(%s)", ?nomSrc, ?nomDest)
#Rule-10-a_holonyme_entre_deux_concepts
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Concept_Classe)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", "OBJET(%s) -LienC-> OBJET(%s)", ?nomSrc,
?nomDest)
#Rule-06-b_desamb_LienR_Principe_Procedure
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lr, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "AGENT(%s)   -REGIT->   PROCEDURE(%s)",
?nomSrc, ?nomDest)
#Rule-15-a_Procedure-P-Principe_Procedure-Condition
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
```

```
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Condition)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",   "PROCEDURE(%s)   -P->   CONDITION(%s)",
?nomSrc, ?nomDest)
#Rule-21-b_desamb_LienR_RegleNomObs_Trace
      metaMot:MOT_LienR(?lr)  ^
      metaMot:MOT_lrConnSrc(?lr, ?src)  ^
      metaMot:MOT_lrConnDest(?lr, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lr, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Regle_Nom)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lr, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?lr, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",    "OBSERVABLE_REGLE_NOM(%s)    -REGIT->
OBSERVABLE_ACTION(%s)", ?nomSrc, ?nomDest)
#Rule-00-c_desamb_lienA
      metaMot:MOT_LienA(?la)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?la, oAmbig:Relation_Application)  ^
          oAmbig:TA_estNonDetermine_typo(?la, false)  ^
          swrlbi:invoker("printf",         "ND_OBSERVABLE(%s)         -A->
ND_ABSTRAIT(%s)", ?nomSrc, ?nomDest)
#Rule-10-b_holonyme_entre_deux_procedures
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?lc, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Procedure(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",        "PROCEDURE(%s)        -HOLONYME->
PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-11-b_holonyme_entre_deux_traces
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Trace(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",      "OBS_PROCEDURE(%s)      -HOLONYME->
OBS_PROCEDURE(%s)", ?nomSrc, ?nomDest)
#Rule-15-c_Trace-P-Enonce_Procedure-Condition
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?src, true)  ^
      metaMot:MOT_Trace(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Condition)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",          "OB_PROCEDURE(%s)          -P->
OB_CONDITION(%s)", ?nomSrc, ?nomDest)
#Rule-03-i_desamb_LienI_Condition_OBS_ND
      metaMot:MOT_LienI(?li)  ^
      metaMot:MOT_connSource(?li, ?src)  ^
      metaMot:MOT_connDestination(?li, ?dest)  ^
      oAmbig:TA_estNonDetermine(?li, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Condition)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?li, oAmbig:Relation_Instanciation)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Condition)  ^
          oAmbig:TA_estNonDetermine_typo(?li, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          swrlbi:invoker("printf",     "CONDITION(%s)      -INSTANCIATION->
OBS_CONDITION_ND(%s)", ?nomSrc, ?nomDest)
#Rule-16-b_Principe-P-Principe_Antecedent-Conclusion
      metaMot:MOT_LienP(?lp)  ^
      metaMot:MOT_connSource(?lp, ?src)  ^
      metaMot:MOT_connDestination(?lp, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lp, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      oAmbig:TA_estNonDetermine_topo(?dest, true)  ^
```

```
      ot:OT_EntiteEstDeType(?src, oAmbig:Regle_Antecedent)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lp, oAmbig:Relation_Precedence)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Regle_Conclusion)  ^
          oAmbig:TA_estNonDetermine_typo(?lp, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "ANTECEDENT(%s)  -P->  CONCLUSION(%s)",
?nomSrc, ?nomDest)
#Rule-00-g_desamb_lienA_ND_Exemple
      metaMot:MOT_LienA(?la)  ^
      metaMot:MOT_connSource(?la, ?src)  ^
      metaMot:MOT_connDestination(?la, ?dest)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?la, true)  ^
      metaMot:MOT_Exemple(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Objet)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf",  "ND(%s)  -A->  OBJET(%s)",  ?nomSrc,
?nomDest)
#Rule-XX-06-2-3_holonyme_procedure_trace
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Trace(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Procedure)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", "Source=%s Dest=%s", ?nomSrc, ?nomDest)
#Rule-XX-07-2-3_holonyme_principe_enonce
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Principe(?src)  ^
      metaMot:MOT_Enonce(?dest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Agent_Contrainte_Norme)  ^
          ot:OT_EntiteEstDeType(?dest,
oAmbig:Observable_AgentNormeContrainte)  ^
          oAmbig:TA_estNonDetermine_typo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_typo(?src, false)  ^
          oAmbig:TA_estNonDetermine_typo(?dest, false)  ^
          swrlbi:invoker("printf", " ")
```

F.2 Base de règles de la désambiguïsation topologique

```
#Rule-10_a_desam_Attribut_Abstrait
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, false)  ^
      ot:OT_EntiteEstDeType(?lc, oAmbig:Attribut)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      metaMot:MOT_Concept(?src)  ^
      metaMot:MOT_Concept(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?src, oAmbig:Concept_Classe)  ^
          oAmbig:TA_estNonDetermine_topo(?src, false)  ^
          swrlbi:invoker("printf",  "OBJET(%s)  -Attribut->  OBJET_ND(%s)",
?nomSrc, ?nomDest)
#Rule-16-a_RegleNom-Antecedent_Principe-Principe-Principe
      metaMot:MOT_Principe(?p)  ^
      metaMot:MOT_estLaSource(?p, ?lpSrc)  ^
      metaMot:MOT_LienP(?lpSrc)  ^
      metaMot:MOT_connDestination(?lpSrc, ?connDest)  ^
      metaMot:MOT_Principe(?connDest)  ^
      metaMot:MOT_estLaDestination(?p, ?lc)  ^
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?lcSrc)  ^
      metaMot:MOT_Principe(?lcSrc)  ^
      metaMot:MOT_estLaDestination(?connDest, ?lcConnDest)  ^
      metaMot:MOT_LienC(?lcConnDest)  ^
      metaMot:MOT_connSource(?lcConnDest, ?lcSrcConnDest)  ^
      sameAs(?lcSrc, ?lcSrcConnDest)  ^
      oAmbig:TA_estNonDetermine(?p, true)  ^
      oAmbig:TA_estNonDetermine(?lpSrc, true)  ^
      oAmbig:TA_estNonDetermine(?connDest, true)  ^
      metaMot:MOT_nomConnaissance(?p, ?nomP)  ^
      metaMot:MOT_nomConnaissance(?lcSrc, ?nomLcSrc)  ^
      metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?p, oAmbig:Regle_Antecedent)  ^
          ot:OT_EntiteEstDeType(?lcSrc, oAmbig:Regle_Nom)  ^
          ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?lpSrc, oAmbig:Relation_Precedence)  ^
          oAmbig:TA_estNonDetermine_topo(?lpSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?p, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lcSrc, false)  ^
          swrlbi:invoker("printf", "REGLE_NOM(%s)  -C->  ANTECEDENT(%s)  -P->
PRINCIPE_ND(%s) )", ?nomLcSrc, ?nomP, ?nomDest)
#Rule-19-c_Procedure-LienC-Principe-Procedure_RegleComplete
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Procedure(?src)  ^
      metaMot:MOT_Principe(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Regle_Complete)  ^
          oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?src, false)  ^
          oAmbig:TA_estNonDetermine_topo(?dest, false)  ^
```

```
          swrlbi:invoker("printf", "PROCEDURE(%s) -C-> REGLE_COMPLETE(%s)",
?nomSrc, ?nomDest)
#Rule-17-b_RegleNom-Antecedent_Enonce-Enonce-Trace
      metaMot:MOT_Enonce(?p)  ^
      metaMot:MOT_estLaSource(?p, ?lpSrc)  ^
      metaMot:MOT_LienP(?lpSrc)  ^
      metaMot:MOT_connDestination(?lpSrc, ?connDest)  ^
      metaMot:MOT_Trace(?connDest)  ^
      metaMot:MOT_estLaDestination(?p, ?lc)  ^
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?lcSrc)  ^
      metaMot:MOT_Enonce(?lcSrc)  ^
      oAmbig:TA_estNonDetermine(?p, true)  ^
      oAmbig:TA_estNonDetermine(?lpSrc, true)  ^
      oAmbig:TA_estNonDetermine(?connDest, true)  ^
      metaMot:MOT_nomConnaissance(?p, ?nomP)  ^
      metaMot:MOT_nomConnaissance(?lcSrc, ?nomLcSrc)  ^
      metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?p, oAmbig:Observable_Regle_Antecedent)  ^
          ot:OT_EntiteEstDeType(?lcSrc, oAmbig:Observable_Regle_Nom)  ^
          ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?lpSrc, oAmbig:Relation_Precedence)  ^
          oAmbig:TA_estNonDetermine_topo(?lpSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?p, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lcSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
          swrlbi:invoker("printf", "REGLE_NOM(%s) -C-> ANTECEDENT(%s) -P->
TRACE_ND(%s) )", ?nomLcSrc, ?nomP, ?nomDest)
#Rule-16-b_RegleNom-Antecedent_Principe-Principe-Procedure
      metaMot:MOT_Principe(?p)  ^
      metaMot:MOT_estLaSource(?p, ?lpSrc)  ^
      metaMot:MOT_LienP(?lpSrc)  ^
      metaMot:MOT_connDestination(?lpSrc, ?connDest)  ^
      metaMot:MOT_Procedure(?connDest)  ^
      metaMot:MOT_estLaDestination(?p, ?lc)  ^
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?lcSrc)  ^
      metaMot:MOT_Principe(?lcSrc)  ^
      oAmbig:TA_estNonDetermine(?p, true)  ^
      oAmbig:TA_estNonDetermine(?lpSrc, true)  ^
      oAmbig:TA_estNonDetermine(?connDest, true)  ^
      metaMot:MOT_nomConnaissance(?p, ?nomP)  ^
      metaMot:MOT_nomConnaissance(?lcSrc, ?nomLcSrc)  ^
      metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?p, oAmbig:Regle_Antecedent)  ^
          ot:OT_EntiteEstDeType(?lcSrc, oAmbig:Regle_Nom)  ^
          ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?lpSrc, oAmbig:Relation_Precedence)  ^
          oAmbig:TA_estNonDetermine_topo(?lpSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?p, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lcSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
          swrlbi:invoker("printf", "REGLE_NOM(%s) -C-> ANTECEDENT(%s) -P->
PROCEDURE_ND(%s) )", ?nomLcSrc, ?nomP, ?nomDest)
#Rule-17-e_RegleNom-Antecedent_Enonce-Enonce-trace
      metaMot:MOT_Enonce(?p)  ^
      metaMot:MOT_estLaSource(?p, ?lpSrc)  ^
      metaMot:MOT_LienP(?lpSrc)  ^
      metaMot:MOT_connDestination(?lpSrc, ?connDest)  ^
      metaMot:MOT_Trace(?connDest)  ^
      metaMot:MOT_estLaDestination(?p, ?lc)  ^
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?lcSrc)  ^
      metaMot:MOT_Enonce(?lcSrc)  ^
      oAmbig:TA_estNonDetermine(?p, true)  ^
      oAmbig:TA_estNonDetermine(?lpSrc, true)  ^
      oAmbig:TA_estNonDetermine(?connDest, true)  ^
      metaMot:MOT_nomConnaissance(?p, ?nomP)  ^
      metaMot:MOT_nomConnaissance(?lcSrc, ?nomLcSrc)  ^
      metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?p, oAmbig:Observable_Regle_Antecedent)  ^
          ot:OT_EntiteEstDeType(?lcSrc, oAmbig:Observable_Regle_Nom)  ^
          ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?lpSrc, oAmbig:Relation_Precedence)  ^
          oAmbig:TA_estNonDetermine_topo(?lpSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?p, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lcSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
          swrlbi:invoker("printf", "OB_REGLE_NOM(%s) -C-> OB_ANTECEDENT(%s)
-P-> TRACE_ND(%s) )", ?nomLcSrc, ?nomP, ?nomDest)
#Rule-08-b_desamb_Enonces_entre_exemple
      metaMot:MOT_Enonce(?p)  ^
      metaMot:MOT_lrEstLaSource(?p, ?lrSrc)  ^
      metaMot:MOT_lrEstLaDestination(?p, ?lrDest)  ^
      metaMot:MOT_LienR(?lrSrc)  ^
      metaMot:MOT_LienR(?lrDest)  ^
      metaMot:MOT_lrConnSrc(?lrDest, ?connSrc)  ^
      metaMot:MOT_lrConnDest(?lrSrc, ?connDest)  ^
      oAmbig:TA_estNonDetermine(?p, true)  ^
      oAmbig:TA_estNonDetermine(?connSrc, true)  ^
      oAmbig:TA_estNonDetermine(?connDest, true)  ^
      metaMot:MOT_Exemple(?connSrc)  ^
      metaMot:MOT_Exemple(?connDest)  ^
      metaMot:MOT_nomConnaissance(?connSrc, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?p, ?nomProp)  ^
      metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?p, oAmbig:Observable_Propriete)  ^
          ot:OT_EntiteEstDeType(?connSrc, oAmbig:Observable_Objet)  ^
          ot:OT_EntiteEstDeType(?connDest, oAmbig:Observable_Objet)  ^
          ot:OT_EntiteEstDeType(?lrSrc, oAmbig:Relation_Regulation)  ^
          ot:OT_EntiteEstDeType(?lrDest, oAmbig:Relation_Regulation)  ^
          oAmbig:TA_estNonDetermine_topo(?p, false)  ^
          oAmbig:TA_estNonDetermine_topo(?connSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?connDest, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lrSrc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?lrDest, false)  ^
          swrlbi:invoker("printf", "OBJET_OBSERVABLE(%s) -R-> ASSERTION(%s)
-R-> OBJET_OBSERVABLE(%s) )", ?nomSrc, ?nomProp, ?nomDest)
#Rule-17-d_Enonce-LienC-Trace-RegleComplete_Procedure
      metaMot:MOT_LienC(?lc)  ^
      metaMot:MOT_connSource(?lc, ?src)  ^
      metaMot:MOT_connDestination(?lc, ?dest)  ^
      oAmbig:TA_estNonDetermine(?lc, true)  ^
      oAmbig:TA_estNonDetermine(?src, true)  ^
      oAmbig:TA_estNonDetermine(?dest, true)  ^
      metaMot:MOT_Trace(?src)  ^
      metaMot:MOT_Enonce(?dest)  ^
      metaMot:MOT_nomConnaissance(?src, ?nomSrc)  ^
      metaMot:MOT_nomConnaissance(?dest, ?nomDest)
      ->  ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
          ot:OT_EntiteEstDeType(?src, oAmbig:Observable_Procedure)  ^
          ot:OT_EntiteEstDeType(?dest, oAmbig:Observable_Regle_Complete)  ^
          oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
          oAmbig:TA_estNonDetermine_topo(?src, false)  ^
```

```
          oAmbig:TA_estNonDetermine_topo(?dest, false)  ^
          swrlbi:invoker("printf",          "OB_PROCEDURE(%s)          -C->
OB_REGLE_COMPLETE(%s)", ?nomSrc, ?nomDest)
#Rule-XX-19-4-3_b_desamb_Principe_entre_concept_procedure
     metaMot:MOT_Principe(?p)  ^
     metaMot:MOT_lrEstLaSource(?p, ?lrSrc)  ^
     metaMot:MOT_lrEstLaDestination(?p, ?lrDest)  ^
     metaMot:MOT_LienR(?lrSrc)  ^
     metaMot:MOT_LienR(?lrDest)  ^
     metaMot:MOT_lrConnSrc(?lrDest, ?connSrc)  ^
     metaMot:MOT_lrConnDest(?lrSrc, ?connDest)  ^
     oAmbig:TA_estNonDetermine(?p, true)  ^
     oAmbig:TA_estNonDetermine(?connSrc, true)  ^
     oAmbig:TA_estNonDetermine(?connDest, true)  ^
     metaMot:MOT_Concept(?connSrc)  ^
     metaMot:MOT_Procedure(?connDest)  ^
     metaMot:MOT_nomConnaissance(?connSrc, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?p, ?nomProp)  ^
     metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?p, oAmbig:Propriete)  ^
         ot:OT_EntiteEstDeType(?connSrc, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?connDest, oAmbig:Procedure)  ^
         ot:OT_EntiteEstDeType(?lrSrc, oAmbig:Relation_Regulation)  ^
         ot:OT_EntiteEstDeType(?lrDest, oAmbig:Relation_Regulation)  ^
         oAmbig:TA_estNonDetermine_topo(?p, false)  ^
         oAmbig:TA_estNonDetermine_topo(?connSrc, false)  ^
         oAmbig:TA_estNonDetermine_topo(?connDest, false)  ^
         oAmbig:TA_estNonDetermine_topo(?lrSrc, false)  ^
         oAmbig:TA_estNonDetermine_topo(?lrDest, false)  ^
         swrlbi:invoker("printf",    "Source=%s    Propriete=%s    Dest=%s",
?nomSrc, ?nomProp, ?nomDest)
#Rule-17-a_RegleNom-Antecedent_Enonce-Enonce-Enonce
     metaMot:MOT_Enonce(?p)  ^
     metaMot:MOT_estLaSource(?p, ?lpSrc)  ^
     metaMot:MOT_LienP(?lpSrc)  ^
     metaMot:MOT_connDestination(?lpSrc, ?connDest)  ^
     metaMot:MOT_Enonce(?connDest)  ^
     metaMot:MOT_estLaDestination(?p, ?lc)  ^
     metaMot:MOT_LienC(?lc)  ^
     metaMot:MOT_connSource(?lc, ?lcSrc)  ^
     metaMot:MOT_Enonce(?lcSrc)  ^
     metaMot:MOT_estLaDestination(?connDest, ?lcConnDest)  ^
     metaMot:MOT_LienC(?lcConnDest)  ^
     metaMot:MOT_connSource(?lcConnDest, ?lcSrcConnDest)  ^
     sameAs(?lcSrc, ?lcSrcConnDest)  ^
     oAmbig:TA_estNonDetermine(?p, true)  ^
     oAmbig:TA_estNonDetermine(?lpSrc, true)  ^
     oAmbig:TA_estNonDetermine(?connDest, true)  ^
     metaMot:MOT_nomConnaissance(?p, ?nomP)  ^
     metaMot:MOT_nomConnaissance(?lcSrc, ?nomLcSrc)  ^
     metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?p, oAmbig:Observable_Regle_Antecedent)  ^
         ot:OT_EntiteEstDeType(?lcSrc, oAmbig:Observable_Regle_Nom)  ^
         ot:OT_EntiteEstDeType(?lc, oAmbig:Holonyme)  ^
         ot:OT_EntiteEstDeType(?lpSrc, oAmbig:Relation_Precedence)  ^
         oAmbig:TA_estNonDetermine_topo(?lpSrc, false)  ^
         oAmbig:TA_estNonDetermine_topo(?p, false)  ^
         oAmbig:TA_estNonDetermine_topo(?lc, false)  ^
         oAmbig:TA_estNonDetermine_topo(?lcSrc, false)  ^
         swrlbi:invoker("printf", "REGLE_NOM(%s)  -C->  ANTECEDENT(%s)  -P->
ENONCE_ND(%s) )", ?nomLcSrc, ?nomP, ?nomDest)
#Rule-08-a_desamb_Principe_entre_concept
     metaMot:MOT_Principe(?p)  ^
     metaMot:MOT_lrEstLaSource(?p, ?lrSrc)  ^
     metaMot:MOT_lrEstLaDestination(?p, ?lrDest)  ^
     metaMot:MOT_LienR(?lrSrc)  ^
     metaMot:MOT_LienR(?lrDest)  ^
     metaMot:MOT_lrConnSrc(?lrDest, ?connSrc)  ^
     metaMot:MOT_lrConnDest(?lrSrc, ?connDest)  ^
     oAmbig:TA_estNonDetermine(?p, true)  ^
     oAmbig:TA_estNonDetermine(?connSrc, true)  ^
     oAmbig:TA_estNonDetermine(?connDest, true)  ^
     metaMot:MOT_Concept(?connSrc)  ^
     metaMot:MOT_Concept(?connDest)  ^
     metaMot:MOT_nomConnaissance(?connSrc, ?nomSrc)  ^
     metaMot:MOT_nomConnaissance(?p, ?nomProp)  ^
     metaMot:MOT_nomConnaissance(?connDest, ?nomDest)
     ->  ot:OT_EntiteEstDeType(?p, oAmbig:Propriete)  ^
         ot:OT_EntiteEstDeType(?connSrc, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?connDest, oAmbig:Concept_Classe)  ^
         ot:OT_EntiteEstDeType(?lrSrc, oAmbig:Relation_Regulation)  ^
         ot:OT_EntiteEstDeType(?lrDest, oAmbig:Relation_Regulation)  ^
         oAmbig:TA_estNonDetermine_topo(?p, false)  ^
         oAmbig:TA_estNonDetermine_topo(?connSrc, false)  ^
         oAmbig:TA_estNonDetermine_topo(?connDest, false)  ^
         oAmbig:TA_estNonDetermine_topo(?lrSrc, false)  ^
         oAmbig:TA_estNonDetermine_topo(?lrDest, false)  ^
         swrlbi:invoker("printf",  "OBJET(%s)  -R->  PROPRIETE(%s)  -R->
OBJET(%s) )", ?nomSrc, ?nomProp, ?nomDest)
```

F.3 Base de règles de transformation: état initial

```
#Rule-00_Importer_metaOnto_du_domaine
      ot:OT_DataOntologieDeDomaine(?o)  ^
      ot:OT_estCree(?o, true)  ^
      ot:OT_DonneesDeTravail(?dt)  ^
      sameAs(?dt, ot:infoOntologieDeDomaine)  ^
      ot:OT_prefix(?dt, ?prefix)  ^
      ot:OT_URL(?dt, ?url)
      ->  swrlbi:invoker("ImporterOntologieCmd", ?prefix, ?url)
#Rule-00_Creer_ontologie_du_domaine
      ot:OT_DataOntologieDeDomaine(?o)  ^
      ot:OT_nameSpace(?o, ?nd)
      ->  swrlbi:invoker("CreerUneOntologieCmd", ?nd)  ^
          swrlbi:invoker("ConnectToMessageServerCmd")  ^
          ot:OT_estCree(?o, true)
```

F.4 Base de règles de transformation: état création

```
#Rule-01-2h_Creation_Individu_Principe_Conclusion
      oRef:OR_Entite_FacteurInfluence_Conclusion(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneInstanceSousCmd",              ?ident,
"metaDom:MD_Strategique_Entite_Regle_Conclusion", ?etiquette)
#Rule-09-e_Creation_propriete_unaire_Domaine_ConnProc
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomP)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-13-c_Creation_class_Schema_Float
      oRef:OR_Entite_Schema_Float(?c)  ^
      oRef:OR_identifiant(?c, ?nc)  ^
      oRef:OR_etiquette(?c, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd", ?nc, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",                ?nc,
"metaDom:MD_Declarative_Schema_Float")
#Rule-01-2c_Creation_Individu_operation
      oRef:OR_Entite_Acte_Operation(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneInstanceSousCmd",              ?ident,
"metaDom:MD_Procedurale_Operation", ?etiquette)
#Rule-01-h_Creation_class_regle_antecedent
      oRef:OR_Entite_Principe_Regle_Antecedent(?antecedent)  ^
      oRef:OR_identifiant(?antecedent, ?nomAntecedent)  ^
      oRef:OR_etiquette(?antecedent, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd",            ?nomAntecedent,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",        ?nomAntecedent,
"metaDom:MD_Strategique_Entite_Regle_Antecedent")
#Rule-18-a_Creation_Propriete_Englobe_EntAbstraite_EntAbstraite
      oRef:OR_Relation_Holonyme_Englobe(?le)  ^
      oRef:OR_connSource(?le, ?src)  ^
      oRef:OR_connDestination(?le, ?dest)  ^
      oRef:OR_Representation_Abstraite(?src)  ^
      oRef:OR_Representation_Abstraite(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_englobe_", ?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomProp)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",            ?nomProp,
"metaDom:ENGLOBE")
#Rule-16_i_Creation_Propriete_aPourConclusion_Antecedent_Conclusion
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?src)  ^
      oRef:OR_Entite_Principe_Regle_Consequent(?dest)  ^
      oRef:OR_identifiant(?src, ?condition)  ^
      oRef:OR_identifiant(?dest, ?conclusion)  ^
      swrlb:stringConcat(?nomPropPreced,  ?condition,  "_aPourConclusion_",
?conclusion)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:A-POUR-CONCLUSION")  ^
          oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-XX-53_Creation_DataTypeProp_aPourIntrant
      oRef:OR_Relation_Flux_Intrant(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_Entite_Schema(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropIP,  ?nomDest,  "_aPourIntrant_",  ?nomSrc)
^
      oRef:OR_etiquette(?src, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteDatatypeCmd",     ?nomPropIP,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",         ?nomPropIP,
"metaDom:A-POUR-INTRANT")
#Rule-01-2d_Creation_Individu_Principe_Agent
      oRef:OR_Entite_FacteurInfluence_Agent_Norme_Contrainte(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneInstanceSousCmd",              ?ident,
"metaDom:MD_Strategique_AgentContrainteNorme", ?etiquette)
#Rule-01-f_Creation_class_regle_nom
      oRef:OR_Entite_Principe_Regle_Nom(?rNom)  ^
      oRef:OR_identifiant(?rNom, ?nomConsequent)  ^
      oRef:OR_etiquette(?rNom, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd",            ?nomConsequent,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",        ?nomConsequent,
"metaDom:MD_Strategique_Entite_Regle_Nom")
#Rule-10-c_Creation_Propriete_aPourComposant_Strategique
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Principe_Agent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
```

```
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-16-g_Creation_Propriete_estAntecedentDe_Antecedent_Antecedent
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?src)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?dest)  ^
      oRef:OR_identifiant(?src, ?condition)  ^
      oRef:OR_identifiant(?dest, ?condition_suite)  ^
      swrlb:stringConcat(?nomPropPreced,  ?condition,  "_estAntecedentDe_",
?condition_suite)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:EST-ANTECEDENT-DE")  ^
          oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-01-a_Creation_class_objet_Concept
      oRef:OR_Entite_Concept_Classe(?c)  ^
      oRef:OR_identifiant(?c, ?nc)  ^
      oRef:OR_etiquette(?c, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd", ?nc, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",                  ?nc,
"metaDom:MD_Declarative_Concept")
#Rule-01-i_Creation_class_regle_conclusion
      oRef:OR_Entite_Principe_Regle_Consequent(?consequent)  ^
      oRef:OR_identifiant(?consequent, ?nomConsequent)  ^
      oRef:OR_etiquette(?consequent, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd",            ?nomConsequent,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",        ?nomConsequent,
"metaDom:MD_Strategique_Entite_Regle_Conclusion")
#Rule-16-b_Creation_Propriete_aPourComposant_Regles_Nom-Antécédent
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Nom(?src)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-09-a_Creation_SuperPropriete_unaire
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomP, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",             ?nomP,
"metaDom:MD_PROPRIETE")  ^
          oRef:OR_estCree(?p, true)
#Rule-XX-19-4-4_b_Creation_dune_propriete_objet_entre_Decl_Proc
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomP, "_", ?nomDest)
^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-06-c_Creation_Prop_Regir_un_autre_principe
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Principe_Agent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomProp)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",          ?nomProp,
"metaDom:REGIT")
#Rule-10-d_Creation_Propriete_aPourComposant_Strategique_Condition
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Principe_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-16-a_Creation_Propriete_aPourComposant_Principe_Procedurale
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Regle_Nom(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-08b_Creation_dune_propriete_objet_PROCEDURAL
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
```

```
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomP, "_", ?nomDest)
^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-14_Creation_Propriete_puisExecuter_Procedure_Procedure
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,     ?nomSrc,     "_puisExecuter_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:PUIS-EXECUTER")  ^
          oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-01-2-5k_Creation_AssertionDeDomaine
      oRef:OR_Entite_Propriete_Objet_Observable(?p)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomP, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",              ?nomP,
"metaDom:MD_ASSERTION")  ^
          oRef:OR_estCree(?p, true)
#Rule-01-d_Creation_class_agent
      oRef:OR_Entite_Principe_Agent(?agent)  ^
      oRef:OR_identifiant(?agent, ?nom_agent)  ^
      oRef:OR_etiquette(?agent, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd", ?nom_agent, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",            ?nom_agent,
"metaDom:MD_Strategique_AgentContrainteNorme")
#Rule-09-c_Creation_propriete_unaire_Domaine
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomP)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-09-b_Creation_propriete_unaire_Image
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomP, "_", ?nomDest)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-15-a_Creation_Propriete_puisEvaluer_Procedure_Condition
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Principe_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,     ?nomSrc,     "_puisEvaluer_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:PUIS-EVALUER")  ^
          oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-01-k_Creation_ProprieteDeDomaine
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomP, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",             ?nomP,
"metaDom:MD_PROPRIETE")  ^
          oRef:OR_estCree(?p, true)
#Rule-06-a_Creation_Propriete_Regit
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomProp)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",           ?nomProp,
"metaDom:REGIT")
#Rule-13-b_Creation_class_Schema_Int
      oRef:OR_Entite_Schema_Int(?c)  ^
      oRef:OR_identifiant(?c, ?nc)  ^
      oRef:OR_etiquette(?c, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd", ?nc, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",                  ?nc,
"metaDom:MD_Declarative_Schema_Int")
#Rule-10-a_Creation_Propriete_aPourComposant
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-15-b_Creation_Propriete_puisExecuter_Condition_Procedure
```

```
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Condition(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,     ?nomSrc,     "_puisExecuter_",
?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:PUIS-EXECUTER")  ^
           oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-03-a_Creation_Propriete_A-POUR-INTRANT
      oRef:OR_Relation_Flux_Intrant(?lIP)  ^
      oRef:OR_connSource(?lIP, ?src)  ^
      oRef:OR_connDestination(?lIP, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropIP, ?nomDest, "_aPourIntrant_", ?nomSrc)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropIP)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",         ?nomPropIP,
"metaDom:A-POUR-INTRANT")
#Rule-03-b_Creation_Propriete_A-POUR-PRODUIT
      oRef:OR_Relation_Flux_Produit(?lIP)  ^
      oRef:OR_connSource(?lIP, ?src)  ^
      oRef:OR_connDestination(?lIP, ?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropIP, ?nomSrc, "_aPourProduit_", ?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropIP)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",         ?nomPropIP,
"metaDom:A-POUR-PRODUIT")
#Rule-08_Creation_dune_propriete_objet_DECLARATIF
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomP, "_", ?nomDest)
^
      oRef:OR_etiquette(?p, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-13-f_Creation_Propriete_aPourAttribut
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropAtttribut,    ?nomSrc,    "_aPourAttribut_",
?nomDest)  ^
      oRef:OR_etiquette(?dest, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd",      ?nomPropAtttribut,
?etiquette)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",   ?nomPropAtttribut,
"metaDom:A-POUR-ATTRIBUT")
#Rule-06-b_Creation_Prop_Regir_une_procedure
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomProp)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",           ?nomProp,
"metaDom:REGIT")
#Rule-10-b_Creation_Propriete_aPourComposant_Procedurale
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-16-d_Creation_Propriete_aPourComposant_Regles_Nom-Operation
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Nom(?src)  ^
      oRef:OR_Entite_Action_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-01-c_Creation_class_Action_Operation
      oRef:OR_Entite_Action_Operation(?p)  ^
      oRef:OR_identifiant(?p, ?np)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneClasseCmd", ?np, ?etiquette)  ^
           swrlbi:invoker("OWLEstUneSousClasseDeCmd",                 ?np,
"metaDom:MD_Procedurale_Operation")
#Rule-16-e_Creation_Propriete_aPourComposant_Procedurale_Strategique
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Principe_Regle_Complete(?dest)  ^
```

```
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-XX-52_Creation_DataTypeProp_aPourProduit
      oRef:OR_Relation_Flux_Produit(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Schema(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropIP,  ?nomSrc,  "_aPourProduit_",  ?nomDest)
^
      oRef:OR_etiquette(?dest, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteDatatypeCmd",      ?nomPropIP,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",          ?nomPropIP,
"metaDom:A-POUR-PRODUIT")
#Rule-20-b_Creation_Prop_Regir_Entre_Regle_Procedure
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_Principe_Regle(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomProp)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",           ?nomProp,
"metaDom:REGIT")
#Rule-16-h_Creation_Propriete_ALORS_Antecedent_Operation
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?src)  ^
      oRef:OR_Entite_Action_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?condition)  ^
      oRef:OR_identifiant(?dest, ?operation)  ^
      swrlb:stringConcat(?nomPropPreced, ?condition, "_alors_", ?operation)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropPreced,
"metaDom:ALORS")  ^
          oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-01-g_Creation_class_regle_complete
      oRef:OR_Entite_Principe_Regle_Complete(?rNom)  ^
      oRef:OR_identifiant(?rNom, ?nomConsequent)  ^
      oRef:OR_etiquette(?rNom, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd",           ?nomConsequent,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",        ?nomConsequent,
"metaDom:MD_Strategique_Entite_Regle_Complete")
#Rule-20_a_Creation_Propriete_Regit_Entre_Regle_Concept
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_Principe_Regle(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomProp)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",           ?nomProp,
"metaDom:REGIT")
#Rule-16-f_Creation_Propriete_EstPrecedentDe_Entre_RegleComplete
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Complete(?src)  ^
      oRef:OR_Entite_Principe_Regle_Complete(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,    ?nomSrc,    "_estPrecedentDe_",
?nomDest)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:EST-PRECEDENT-DE")  ^
          oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-01-2f_Creation_Individu_Principe_Regle_Complete
      oRef:OR_Entite_FacteurInfluence_Regle_Complete(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Strategique_Entite_Regle_Complete", ?etiquette)
#Rule-01-b_Creation_class_Action_Procedure
      oRef:OR_Entite_Action_Procedure(?p)  ^
      oRef:OR_identifiant(?p, ?np)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd", ?np, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",                    ?np,
"metaDom:MD_Procedurale_Procedure")
#Rule-13-d_Creation_class_Schema_String
      oRef:OR_Entite_Schema_String(?c)  ^
      oRef:OR_identifiant(?c, ?nc)  ^
      oRef:OR_etiquette(?c, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneClasseCmd", ?nc, ?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousClasseDeCmd",                    ?nc,
"metaDom:MD_Declarative_Schema_String")
#Rule-13-a_Creation_Propriete_aPourAttribut_Schema
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Schema(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropAtttribut,   ?nomSrc,   "_aPourAttribut_",
?nomDest)  ^
      oRef:OR_etiquette(?dest, ?etiquette)
      ->  swrlbi:invoker("OWLCreerUneProprieteCmd",      ?nomPropAtttribut,
?etiquette)  ^
          swrlbi:invoker("OWLEstUneSousProprieteDeCmd",  ?nomPropAtttribut,
"metaDom:A-POUR-ATTRIBUT")
#Rule-16-c_Creation_Propriete_aPourComposant_Regles_Nom-Consequent
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Nom(?src)  ^
      oRef:OR_Entite_Principe_Regle_Consequent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
```

```
      swrlb:stringConcat(?nomPropCompo,     ?nomSrc,     "_aPourComposant_",
?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropCompo)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
"metaDom:A-POUR-COMPOSANT")
#Rule-09-d_Creation_propriete_unaire_Image_ConnProc
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estCree(?p, true)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomP, "_", ?nomDest)  ^
      oRef:OR_etiquette(?p, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd",          ?nomPropCompo,
?etiquette)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",      ?nomPropCompo,
?nomP)
#Rule-01-e_Creation_class_condition
      oRef:OR_Entite_Principe_Condition(?decision)  ^
      oRef:OR_identifiant(?decision, ?nomDecision)  ^
      oRef:OR_etiquette(?decision, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneClasseCmd",  ?nomDecision,  ?etiquette)
^
           swrlbi:invoker("OWLEstUneSousClasseDeCmd",          ?nomDecision,
"metaDom:MD_Strategique_Condition")
#Rule-15-c_Creation_Propriete_puisEvaluer_Condition_Condition
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Condition(?src)  ^
      oRef:OR_Entite_Principe_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,     ?nomSrc,     "_puisEvaluer_",
?nomDest)
      ->   swrlbi:invoker("OWLCreerUneProprieteCmd", ?nomPropPreced)  ^
           swrlbi:invoker("OWLEstUneSousProprieteDeCmd",     ?nomPropPreced,
"metaDom:PUIS-EVALUER")  ^
           oRef:OR_identifiant(?lp, ?nomPropPreced)
#Rule-01-2b_Creation_Individu_procedural
      oRef:OR_Entite_Acte_Procedure(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Procedurale_Procedure", ?etiquette)
#Rule-01-2e_Creation_Individu_Principe_Regle_Nom
      oRef:OR_Entite_FacteurInfluence_Regle_Nom(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Strategique_Entite_Regle_Nom", ?etiquette)
#Rule-13-e-Creation_Dun_Individu_dattribut
      oRef:OR_Entite_Manifestation_Attribut(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Declarative_Schema", ?etiquette)
#Rule-01-2i_Creation_Individu_Principe_Condition
      oRef:OR_Entite_FacteurInfluence_Condition(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Strategique_Condition", ?etiquette)
#Rule-01-2a_Creation_Individu_objet
      oRef:OR_Entite_Manifestation_Objet(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Declarative_Concept", ?etiquette)
#Rule-XX-32_Creation_dune_propriete_objet_dataType
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Schema(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomP, "_", ?nomDest)
^
      oRef:OR_etiquette(?dest, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneProprieteDatatypeCmd",  ?nomPropCompo,
?etiquette)
#Rule-01-2g_Creation_Individu_Principe_Antecedent
      oRef:OR_Entite_FacteurInfluence_Antecedent(?o)  ^
      oRef:OR_identifiant(?o, ?ident)  ^
      oRef:OR_etiquette(?o, ?etiquette)
      ->   swrlbi:invoker("OWLCreerUneInstanceSousCmd",               ?ident,
"metaDom:MD_Strategique_Entite_Regle_Antecedent", ?etiquette)
```

F.5 Base de règles de transformation: état classification

```
#Rule-XX-03-2-
4_Ajoute_Image_Domaine_a_la_Propeiete_aPour_Composant_entre_AgentEnConnProc
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->   swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-16-e_Ajout_ImgDom_Prop_estAntecedentDe_Entre_Principes
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?src)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,    ?nomSrc,    "_estAntecedentDe_",
?nomDest)
      ->   swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-13-b_Creation_propriete_aPourAttribut_entre_Concept_Exemple
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
```

```
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreClasseEtIndividuCmd",
?nomSrc, "metaDom:A-POUR-ATTRIBUT", ?nomDest)
#Rule-16-
d_Ajoute_Image_Domaine_a_la_Propriete_aPourComosant_ConnProceduraleEtConnSt
rag
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Principe_Regle_Complete(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-20-a_Ajout_ImgDom_Prop_REGIT_Entre_Regle_Procedure
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_Entite_Principe_Regle(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-15-b_Ajout_ImgDom_Prop_puisExecuter_Condition_Procedure
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Condition(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,     ?nomSrc,     "_puisExecuter_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-06_b_Ajout_ImgDom_Prop_Regit_Procedure
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-XX-99-Assignation_de_disjonction
      oRef:OR_Relation_Operateur_Binaire_Disjoint(?d)  ^
      oRef:OR_estLaSource(?d, ?lrSource)  ^
      oRef:OR_estLaDestination(?d, ?lrDest)  ^
      oRef:OR_connSource(?lrDest, ?src)  ^
      oRef:OR_connDestination(?lrSource, ?dest)  ^
      oRef:OR_identifiant(?src, ?nomSource)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLEstDisjointDeCmd", ?nomSource, ?nomDest)
#Rule-17-d_Creation_Individu_Antecedent_Alors_Action
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Antecedent(?src)  ^
      oRef:OR_Entite_Acte_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:ALORS", ?nomDest)
#Rule-XX-22-2-4_Creation_propriete_regit_entre_exemple_enonce
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_Manifestation_Objet(?src)  ^
      oRef:OR_Entite_FacteurInfluence(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:REGIT", ?nomDest)
#Rule-05-b_Creation_propriete_A-POUR-INTRANT_assertion
      oRef:OR_Relation_Flux_Intrant(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Manifestation_Objet(?src)  ^
      oRef:OR_Entite_Acte(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd", ?nomDest,
"metaDom:A-POUR-INTRANT", ?nomSrc)
#Rule-01-f_Creation_subClassOf_SchemaFloat
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Schema_Float(?src)  ^
      oRef:OR_Entite_Manifestation_Attribut(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_etiquette(?dest, ?valeur)
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Declarative_Schema", ?nomDest)  ^
          swrlbi:invoker("OWLAssignerUneValeurAUnAttributCmd",     ?nomSrc,
"metaDom:MD_isString", "XsdFloat", ?valeur, ?nomDest)
#Rule-11_Creation_propriete_aPourComposant_individu
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Representation_Concrete(?src)  ^
      oRef:OR_Representation_Concrete(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:A-POUR-COMPOSANT", ?nomDest)
#Rule-03-b_Creation_typeOf_ACTION-PROCEDURE
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Acte_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
```

```
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Procedurale_Procedure", ?nomDest)
#Rule-XX-31_Ajout_Domaine_DataType_a_une_propriete
      oRef:OR_Entite_Propriete_Objet(?prop)  ^
      oRef:OR_connSource(?prop, ?src)  ^
      oRef:OR_connDestination(?prop, ?dest)  ^
      oRef:OR_identifiant(?prop, ?nomProp)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Schema(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,   ?nomSrc,   "_",   ?nomProp,   "_",
?nomDest)  ^
      ot:OT_EntiteEstDeType(?dest, ?nomType)  ^
      oAmbig:TA_identifiant(?nomType, ?type)
      ->
      swrlbi:invoker("OWLProprieteDatatype_AjoutDunDomaineEtDunRangeTypeCmd
", ?nomSrc, ?nomPropCompo, ?type)
#Rule-10-
b_Ajoute_Image_Domaine_a_la_Propriete_aPourComposant_ConnProcedurale
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-18-b_Englobe-Declaratif-Declaratif
      oRef:OR_Relation_Holonyme_Englobe(?le)  ^
      oRef:OR_connSource(?le, ?src)  ^
      oRef:OR_connDestination(?le, ?dest)  ^
      oRef:OR_Entite(?src)  ^
      oRef:OR_Entite(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreResourceCmd",  ?nomSrc,
"metaDom:ENGLOBE", ?nomDest)
#Rule-14-b_Creation_Individu_Procedural_puisExecuter_Procedural
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Acte_Procedure(?src)  ^
      oRef:OR_Entite_Acte_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:PUIS-EXECUTER", ?nomDest)
#Rule-04-a_Ajout_Image_Domaine_A-POUR-INTRANT
      oRef:OR_Relation_Flux_Intrant(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomDest,     "_aPourIntrant_",
?nomSrc)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomDest, ?nomPropCompo, ?nomSrc)
#Rule-10-a_Ajoute_Image_Domaine_a_la_Propriete_aPourComposant
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-XX_99_Englobe_declaratif_propriete
      oRef:OR_Relation_Holonyme_Englobe(?le)  ^
      oRef:OR_connSource(?le, ?src)  ^
      oRef:OR_connDestination(?le, ?dest)  ^
      oRef:OR_Entite(?src)  ^
      oRef:OR_Entite_Propriete_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_estLaSource(?dest, ?lrSrc)  ^
      oRef:OR_estLaDestination(?dest, ?lrDest)  ^
      oRef:OR_Relation_Regulation(?lrSrc)  ^
      oRef:OR_Relation_Regulation(?lrDest)  ^
      oRef:OR_connSource(?lrDest, ?connSrc)  ^
      oRef:OR_connDestination(?lrSrc, ?connDest)  ^
      oRef:OR_identifiant(?connSrc, ?nomConnSrc)  ^
      oRef:OR_identifiant(?connDest, ?nomConnDest)  ^
      swrlb:stringConcat(?nomPropCompo,  ?nomConnSrc,  "_",  ?nomDest,  "_",
?nomConnDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreResourceCmd",  ?nomSrc,
"metaDom:ENGLOBE", ?nomPropCompo)
#Rule-03-c_Creation_typeOf_PRINCIPE-AGENT
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Agent_Norme_Contrainte(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Strategique_AgentContrainteNorme", ?nomDest)
#Rule-12-b_Creation_propriete_aPourComposant_entre_Procedure_Trace
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Acte_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreClasseEtIndividuCmd",
?nomSrc, "metaDom:A-POUR-COMPOSANT", ?nomDest)
#Rule-08-b_Ajout_ImgDom_Prop_ConnProcedurale
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
```

```
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,   ?nomSrc,   "_",   ?nomProp,   "_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropCompo, ?nomDest)
#Rule-13-f_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut_Float
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Schema_Float(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourAttribut_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-15-e_Creation_Individu_Condition_puisExecuter_Procedural
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Condition(?src)  ^
      oRef:OR_Entite_Acte_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:PUIS-EXECUTER", ?nomDest)
#Rule-10-c_Ajoute_Image_Domaine_a_la_Propeiete_aPour_Composant_entre_Agent
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Principe_Agent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-16-f_Ajout_ImgDom_Prop_alors_Condition_Procedure
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?src)  ^
      oRef:OR_Entite_Action_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced, ?nomSrc, "_alors_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-17-c_Creation_Individu_EstPrecedentDe_Entre_RegleComplete
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Regle_Complete(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Regle_Complete(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:EST-PRECEDENT-DE", ?nomDest)
#Rule-16-b_Ajoute_Image_Domaine_aPourComposant_entre_RegleNom_Antecedent
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Nom(?src)  ^
      oRef:OR_Entite_Principe_Regle_Consequent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-04-b_Ajout_Image_Domaine_A-POUR-PRODUIT
      oRef:OR_Relation_Flux_Produit(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,      ?nomSrc,      "_aPourProduit_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropCompo, ?nomDest)
#Rule-XX-17-2-7_Creation_Individu_Condition_alors_Procedure
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Condition(?src)  ^
      oRef:OR_Entite_Action_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:ALORS", ?nomDest)
#Rule-01-d_Creation_subClassOf_SchemaString
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Schema_String(?src)  ^
      oRef:OR_Entite_Manifestation_Attribut(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_etiquette(?dest, ?valeur)
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Declarative_Schema", ?nomDest)  ^
          swrlbi:invoker("OWLAssignerUneValeurAUnAttributCmd",     ?nomSrc,
"metaDom:MD_isString", "XsdString", ?valeur, ?nomDest)
#Rule-03-b_Creation_typeOf_ACTION-OPERATION
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Action_Operation(?src)  ^
      oRef:OR_Entite_Acte_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Procedurale_Operation", ?nomDest)
#Rule-06-a_Ajout_ImgDom_Prop_REGIT
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
```

```
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-XX-52_Ajout_d_un_domaine_et_type_Prop_Intrants
      oRef:OR_Relation_Flux_Intrant(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Schema(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,     ?nomDest,     "_aPourIntrant_",
?nomSrc)  ^
      ot:OT_EntiteEstDeType(?src, ?nomType)  ^
      oAmbig:TA_identifiant(?nomType, ?type)
      ->
      swrlbi:invoker("OWLProprieteDatatype_AjoutDunDomaineEtDunRangeTypeCmd
", ?nomDest, ?nomPropCompo, ?type)
#Rule-08-b_Creation_propriete_entre_deux_ConnProceduraux
      oRef:OR_Entite_Propriete_Objet_Observable(?p)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
?nomP, ?nomDest)
#Rule-16-g_Ajout_ImgDom_Prop_estPrecedentDe_RegleNom_RegleNom
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Complete(?src)  ^
      oRef:OR_Entite_Principe_Regle_Complete(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,    ?nomSrc,    "_estPrecedentDe_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-05-a_Creation_propriete_A-POUR-PRODUIT_assertion
      oRef:OR_Relation_Flux_Produit(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Acte(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:A-POUR-PRODUIT", ?nomDest)
#Rule-14-d_Creation_propriete_estPrecedentDe_entre_Trace_Procedure
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Acte_Procedure(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreClasseEtIndividuCmd",
?nomDest, "metaDom:EST-LE-SUIVANT-DE", ?nomSrc)
#Rule-16-h_Ajout_ImgDom_Prop_aPourConclusion
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?src)  ^
      oRef:OR_Entite_Principe_Regle_Consequent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,    ?nomSrc,    "_aPourConclusion_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-17-b_Creation_Individu_Antecedent_estAntecedentDe_Antecedent
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Antecedent(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Antecedent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:EST-ANTECEDENT-DE", ?nomDest)
#Rule-12-a_Creation_propriete_aPourComposant_entre_Concept_Exemple
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreClasseEtIndividuCmd",
?nomSrc, "metaDom:A-POUR-COMPOSANT", ?nomDest)
#Rule-13-e_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut_Int
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Schema_Int(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourAttribut_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-18_Ajout_ImgDom_Prop_englobe
      oRef:OR_Relation_Holonyme_Englobe(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Representation_Abstraite(?src)  ^
      oRef:OR_Representation_Abstraite(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_englobe_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-XX-30_Association_des_individus_a_une_propriete
```

```
      oRef:OR_Entite_Propriete_Objet(?prop)  ^
      oRef:OR_connSource(?prop, ?src)  ^
      oRef:OR_connDestination(?prop, ?dest)  ^
      oRef:OR_identifiant(?prop, ?nomProp)  ^
      oRef:OR_Entite_Manifestation_Objet(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
?nomProp, ?nomDest)
#Rule-06_c_Ajout_ImgDom_Prop_Regit_Principes
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Principe_Agent(?dest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-16-a_Ajoute_Image_Domaine_aPourComposant_entre_RegleNom_Consequence
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Nom(?src)  ^
      oRef:OR_Entite_Principe_Regle_Antecedent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-02-c_Creation_subClassOf_Strategique
      oRef:OR_Relation_Hyponyme(?ls)  ^
      oRef:OR_connSource(?ls, ?src)  ^
      oRef:OR_connDestination(?ls, ?dest)  ^
      oRef:OR_Entite_Principe(?src)  ^
      oRef:OR_Entite_Principe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlbi:invoker("OWLEstUneSousClasseDeCmd", ?nomSrc, ?nomDest)
      ->  swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",       "owl:Thing",
?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",
"metaDom:MD_Agent_Contrainte_Norme", ?nomSrc)
#Rule-20-a_Ajout_ImgDom_Prop_REGIT_Entre_Regle_Concept
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_Entite_Principe_Regle(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_regit_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-09-b_Ajout_Domaine_Propriete_Unaire_ConnProc
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomProp)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropCompo, "null")
#Rule-03-d_Creation_typeOf_CONDITION
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Principe_Condition(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Strategique_AgentContrainteNorme", ?nomDest)
#Rule-14-c_Creation_propriete_estPrecedentDe_entre_Procedure_Trace
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Acte_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreClasseEtIndividuCmd",
?nomSrc, "metaDom:EST-PRECEDENT-DE", ?nomDest)
#Rule_51_Ajout_d_un_domaine_et_type_Prop_Produit
      oRef:OR_Relation_Flux_Produit(?lip)  ^
      oRef:OR_connSource(?lip, ?src)  ^
      oRef:OR_connDestination(?lip, ?dest)  ^
      oRef:OR_Entite_Schema(?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,      ?nomSrc,      "_aPourProduit_",
?nomDest)  ^
      ot:OT_EntiteEstDeType(?dest, ?nomType)  ^
      oAmbig:TA_identifiant(?nomType, ?type)
      ->
      swrlbi:invoker("OWLProprieteDatatype_AjoutDunDomaineEtDunRangeTypeCmd
", ?nomSrc, ?nomPropCompo, ?type)
#Rule-13-a_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourAttribut_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-17-a_Creation_Individu_Antecedent_aPourConclusion_Conclusion
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Antecedent(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Conclusion(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:A-POUR-CONCLUSION", ?nomDest)
```

```
#Rule-09-a_Ajout_Image_Propriete_Unaire
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomProp, "_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
"null", ?nomPropCompo, ?nomDest)
#Rule-09-b_Ajout_Domaine_Propriete_Unaire
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomProp)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropCompo, "null")
#Rule-08-b_Creation_propriete_entre_deux_individus
      oRef:OR_Entite_Propriete_Objet_Observable(?p)  ^
      oRef:OR_identifiant(?p, ?nomP)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_Entite_Manifestation_Objet(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
?nomP, ?nomDest)
#Rule-08-a_Ajout_ImgDom_a_une_propriete_objet
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,  ?nomSrc,  "_",  ?nomProp,  "_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropCompo, ?nomDest)
#Rule-07-a_Creation_propriete_regit_entre_enonce_exemple
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:REGIT", ?nomDest)
#Rule-15-a_Ajout_ImgDom_Prop_puisEvaluer_Procedure_Condition
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Principe_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,      ?nomSrc,      "_puisEvaluer_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-15-c_Ajout_ImgDom_Prop_puisEvaluer_Condition_Condition
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Principe_Condition(?src)  ^
      oRef:OR_Entite_Principe_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,      ?nomSrc,      "_puisEvaluer_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-15-f_Creation_Individu_Condition_puisEvaluer_Condition
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence_Condition(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:PUIS-EVALUER", ?nomDest)
#Rule-07-c_Creation_Regit_entre_deux_enonces
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence(?src)  ^
      oRef:OR_Entite_FacteurInfluence(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:REGIT", ?nomDest)
#Rule-XX-19-4-4b_Ajout_ImgDom_a_une_propriete_objet_ConnDecl_Proc
      oRef:OR_Entite_Propriete_Objet(?p)  ^
      oRef:OR_estLaDestination(?p, ?lr_dest)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_dest)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connSource(?lr_dest, ?src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo,  ?nomSrc,  "_",  ?nomProp,  "_",
?nomDest)
```

```
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropCompo, ?nomDest)
#Rule-XX-30_Ajout_dun_domaine_a_un_propriete_unaire
      oRef:OR_Entite_Propriete_Objet_Unaire(?prop)  ^
      oRef:OR_connSource(?prop, ?src)  ^
      oRef:OR_identifiant(?prop, ?nomProp)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomSrc, "_", ?nomProp)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineCmd",        ?nomSrc,
?nomPropCompo)
#Rule-02-a_Creation_subClassOf_OBJET
      oRef:OR_Relation_Hyponyme(?ls)  ^
      oRef:OR_connSource(?ls, ?src)  ^
      oRef:OR_connDestination(?ls, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Concept_Classe(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlbi:invoker("OWLEstUneSousClasseDeCmd", ?nomSrc, ?nomDest)
      ->  swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",        "owl:Thing",
?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",
"metaDom:MD_Declarative_Concept", ?nomSrc)
#Rule-10-d_Ajoute_ImgDom_aPourComposant_Agent_Condition
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_Principe_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-11_b_Creation_Propriete_aPourAttribut_individu
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Representation_Concrete(?src)  ^
      oRef:OR_Representation_Concrete(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:A-POUR-ATTRIBUT", ?nomDest)
#Rule-XX-40_Creation_Individu_Procedural_puisEvaluer_Antecedent
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Antecedent(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->  swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",  ?nomSrc,
"metaDom:PUIS-EVALUER", ?nomDest)
#Rule-16-c_Ajoute_Image_Domaine_aPourComposant_entre_RegleNom_Operation
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Regle_Nom(?src)  ^
      oRef:OR_Entite_Action_Operation(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourComposant_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-01-e_Creation_subClassOf_SchemaInt
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Schema_Int(?src)  ^
      oRef:OR_Entite_Manifestation_Attribut(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      oRef:OR_etiquette(?dest, ?valeur)
      ->  swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Declarative_Schema", ?nomDest)  ^
          swrlbi:invoker("OWLAssignerUneValeurAUnAttributCmd",     ?nomSrc,
"metaDom:MD_isString", "XsdInt", ?valeur, ?nomDest)
#Rule-02-b_Creation_subClassOf_Procedurale
      oRef:OR_Relation_Hyponyme(?ls)  ^
      oRef:OR_connSource(?ls, ?src)  ^
      oRef:OR_connDestination(?ls, ?dest)  ^
      oRef:OR_Entite_Action(?src)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlbi:invoker("OWLEstUneSousClasseDeCmd", ?nomSrc, ?nomDest)
      ->  swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",        "owl:Thing",
?nomSrc)  ^
          swrlbi:invoker("OWLSupprimerSuperClasseDeCmd",
"metaDom:MD_Procedurale", ?nomSrc)
#Rule-13-g_Ajoute_Image_Domaine_a_la_Propriete_aPourAttribut_String
      oRef:OR_Relation_Attribut(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Schema_String(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomProp, ?nomSrc, "_aPourAttribut_", ?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomProp, ?nomDest)
#Rule-14-a_Ajout_ImgDom_Prop_puisExecuter_Procedure_Procedure
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Action_Procedure(?src)  ^
      oRef:OR_Entite_Action_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropPreced,     ?nomSrc,     "_puisExecuter_",
?nomDest)
      ->  swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
?nomSrc, ?nomPropPreced, ?nomDest)
#Rule-12-c_Creation_propriete_aPourComposant_entre_principe_enonce
      oRef:OR_Relation_Holonyme_Composition(?lc)  ^
      oRef:OR_connSource(?lc, ?src)  ^
      oRef:OR_connDestination(?lc, ?dest)  ^
      oRef:OR_Entite_Principe_Agent(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Agent_Norme_Contrainte(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
```

```
      ->   swrlbi:invoker("OWLAjoutDuneProprieteEntreClasseEtIndividuCmd",
?nomSrc, "metaDom:A-POUR-COMPOSANT", ?nomDest)
#Rule-03-a_Creation_typeOf_OBJET
      oRef:OR_Relation_Instance(?li)  ^
      oRef:OR_connSource(?li, ?src)  ^
      oRef:OR_connDestination(?li, ?dest)  ^
      oRef:OR_Entite_Concept_Classe(?src)  ^
      oRef:OR_Entite_Manifestation_Objet(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->   swrlbi:invoker("OWLEstUneInstanceDeCmd", ?nomDest, ?nomSrc)  ^
           swrlbi:invoker("OWLSupprimerLeTypeDeLinstanceCmd",
"metaDom:MD_Declarative_Concept", ?nomDest)
#Rule-07-b_Creation_Regit_enonce_trace
      oRef:OR_Relation_Regulation(?lr)  ^
      oRef:OR_connSource(?lr, ?src)  ^
      oRef:OR_connDestination(?lr, ?dest)  ^
      oRef:OR_Entite_FacteurInfluence(?src)  ^
      oRef:OR_Entite_Acte_Procedure(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->   swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",   ?nomSrc,
"metaDom:REGIT", ?nomDest)
#Rule-15-d_Creation_Individu_Procedural_puisEvaluer_Condition
      oRef:OR_Relation_Flux_Precedence(?lp)  ^
      oRef:OR_connSource(?lp, ?src)  ^
      oRef:OR_connDestination(?lp, ?dest)  ^
      oRef:OR_Entite_Acte_Procedure(?src)  ^
      oRef:OR_Entite_FacteurInfluence_Condition(?dest)  ^
      oRef:OR_identifiant(?src, ?nomSrc)  ^
      oRef:OR_identifiant(?dest, ?nomDest)
      ->   swrlbi:invoker("OWLAjoutDuneProprieteEntreIndividuCmd",   ?nomSrc,
"metaDom:PUIS-EVALUER", ?nomDest)
#Rule-09-a_Ajout_Image_Propriete_Unaire_ConnProc
      oRef:OR_Entite_Propriete_Objet_Unaire(?p)  ^
      oRef:OR_estLaSource(?p, ?lr_src)  ^
      oRef:OR_Relation_Regulation(?lr_src)  ^
      oRef:OR_connDestination(?lr_src, ?dest)  ^
      oRef:OR_identifiant(?p, ?nomProp)  ^
      oRef:OR_Entite_Action(?dest)  ^
      oRef:OR_identifiant(?dest, ?nomDest)  ^
      swrlb:stringConcat(?nomPropCompo, ?nomProp, "_", ?nomDest)
      ->   swrlbi:invoker("OWLPropriete_AjoutDunDomaineEtDuneImageCmd",
"null", ?nomPropCompo, ?nomDest)
```

F.6 Base de règles de transformation: état final

```
#Rule-00_Sauvegarde
      ot:OT_DataOntologieDeDomaine(?o)  ^
      ot:OT_nameSpace(?o, ?ns)  ^
      ot:OT_defaultUriLocation(?o, ?nf)
      ->   swrlbi:invoker("SauvegarderOntologieCmd", ?nf, ?ns)
```

F.7 Base de règles de transformation: état validation

```
#Rule-ERREUR_HYPONYME_SCHEMA
      ot:OT_ERREUR_HYPONYME_Schema(?err_ins)  ^
      ot:OT_ERREUR_Message(ot:OT_MessageErreur_HYPONYME_Schema,    ?err_msg)
^
      oRef:OR_connDestination(?err_ins, ?connDest)  ^
      oRef:OR_identifiant(?err_ins, ?ident)
      ->   swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ^
           swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
#Rule-ERREUR_REL_ATTRIBUT
      ot:OT_ERREUR_REL_ATTRIBUT(?err_ins)  ^
      ot:OT_ERREUR_Message(ot:OT_MessageErreur_REL_ATTRIBUT, ?err_msg)  ^
      oRef:OR_connDestination(?err_ins, ?connDest)  ^
      oRef:OR_identifiant(?err_ins, ?ident)
      ->   swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ^
           swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
#Rule-ERREUR_REGULATION
      ot:OT_ERREUR_REGULATION(?err_ins)  ^
      ot:OT_ERREUR_Message(ot:OT_MessageErreur_REGULATION, ?err_msg)  ^
      oRef:OR_connDestination(?err_ins, ?connDest)  ^
      oRef:OR_identifiant(?err_ins, ?ident)
      ->   swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ^
           swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
#Rule-ERREUR_INSTANCE
      ot:OT_ERREUR_INSTANCE(?err_ins)  ^
      ot:OT_ERREUR_Message(ot:OT_MessageErreur_INSTANCE, ?err_msg)  ^
      oRef:OR_connDestination(?err_ins, ?connDest)  ^
      oRef:OR_identifiant(?err_ins, ?ident)
      ->   swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ^
           swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
#Rule-ERREUR_HYPONYME
      ot:OT_ERREUR_HYPONYME_Categorie(?err_ins)  ^
      ot:OT_ERREUR_Message(ot:OT_MessageErreur_HYPONYME, ?err_msg)  ^
      oRef:OR_connDestination(?err_ins, ?connDest)  ^
      oRef:OR_identifiant(?err_ins, ?ident)
      ->   swrlbi:invoker("SendMessageDerreurCmd", ?ident, ?err_msg)  ^
           swrlbi:invoker("printf", "(%s): [%s]", ?ident, ?err_msg)
```

F.8 Ontologie du langage semi-formel MOT

```
# Saved by TopBraid on Mon Nov 09 17:22:45 EST 2009
# baseURI: http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl

@prefix                                                              :
<http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .

<http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl>
      rdf:type owl:Ontology ;
      rdfs:comment "Copyright Cotechnoe inc 2009"^^xsd:string ;
      rdfs:label "EPackage_mot"^^xsd:string .
:AUTEUR
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:CATEGORIE
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:COMMENTAIRE
```

```
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:COPYRIGHT
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:DATECREATION
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:DERNIERMODIFICATION
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:LICENSE
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:MOTCLE
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:MOT_Concept
      rdf:type owl:Class ;
      rdfs:subClassOf :MOT_ConnaissanceAbstraite .
:MOT_Connaissance
      rdf:type owl:Class ;
      rdfs:label "EClass_Connaissance"^^xsd:string ;
      rdfs:subClassOf :MOT_Structure .
:MOT_ConnaissanceAbstraite
      rdf:type owl:Class ;
      rdfs:label "EClass_Connaissanceabstraite"^^xsd:string ;
      rdfs:subClassOf :MOT_Connaissance .
:MOT_Decision
      rdf:type owl:Class ;
      rdfs:label "EClass_Decision"^^xsd:string ;
      rdfs:subClassOf :MOT_Fait .
:MOT_Enonce
      rdf:type owl:Class ;
      rdfs:comment "Les Ã©noncÃ©s sont obtenus en spÃ©cifiant les variables dâ€™un principe, obtenant ainsi un lien de cause Ã  effet entre les propriÃ©tÃ©s dâ€™objets particuliers ou entre des propriÃ©tÃ©s dâ€™un objet particulier et une action prÃ©cise Ã  effectuer. (Ref GP)"@fr ;
      rdfs:label "EClass_Enonce"^^xsd:string ;
      rdfs:subClassOf :MOT_Fait .
:MOT_Exemple
      rdf:type owl:Class ;
      rdfs:comment "Les Ã©noncÃ©s sont obtenus en spÃ©cifiant les variables dâ€™un principe, obtenant ainsi un lien de cause Ã  effet entre les propriÃ©tÃ©s dâ€™objets particuliers ou entre des propriÃ©tÃ©s dâ€™un objet particulier et une action prÃ©cise Ã  effectuer."@fr ;
      rdfs:label "EClass_Exemple"^^xsd:string ;
      rdfs:subClassOf :MOT_Fait .
:MOT_Fait
      rdf:type owl:Class ;
      rdfs:label "EClass_Fait"^^xsd:string ;
      rdfs:subClassOf :MOT_Connaissance .
:MOT_LienA
      rdf:type owl:Class ;
      rdfs:label "EClass_Application"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienC
      rdf:type owl:Class ;
      rdfs:label "EClass_Composition"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienCm
      rdf:type owl:Class ;
      rdfs:label "EClass_CompositionMultiple"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienE
      rdf:type owl:Class ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienI
      rdf:type owl:Class ;
      rdfs:label "EClass_Instanciation"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienIP
      rdf:type owl:Class ;
      rdfs:label "EClass_Intrant_Produit"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienNT
      rdf:type owl:Class ;
      rdfs:label "EClass_LienNonType"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienP
      rdf:type owl:Class ;
      rdfs:label "EClass_Precedence"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienR
      rdf:type owl:Class ;
      rdfs:label "EClass_Regulation"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_LienS
      rdf:type owl:Class ;
      rdfs:label "EClass_Specialisation"^^xsd:string ;
      rdfs:subClassOf :MOT_Relation .
:MOT_NonType
      rdf:type owl:Class ;
      rdfs:label "EClass_NonType"^^xsd:string ;
      rdfs:subClassOf :MOT_Connaissance .
:MOT_Principe
      rdf:type owl:Class ;
      rdfs:comment "Les principes ou connaissances stratÃ©giques sont des Ã©noncÃ©s permettant de dÃ©crire les propriÃ©tÃ©s des objets, dâ€™Ã©tablir des liens de cause Ã  effet entre des objets ( le pourquoi ), ou de dÃ©terminer dans quelles conditions appliquer une procÃ©dure ( le quand ); les principes prennent le plus souvent la forme Â »  Si telle condition Alors, telle condition ou telle action Â ». (ref: GP)"@fr ;
      rdfs:label "EClass_Principe"^^xsd:string ;
      rdfs:subClassOf :MOT_ConnaissanceAbstraite .
:MOT_Procedure
      rdf:type owl:Class ;
      rdfs:comment "Les procÃ©dures ou connaissances procÃ©durales dÃ©crivent des ensembles dâ€™opÃ©rations permettant dâ€™agir sur les objets (le comment); elles sâ€™intÃ©ressent aux combinaisons dâ€™actions sâ€™appliquant dans plusieurs cas, chaque cas se distinguant des autres par
```

```
les objets auxquels les actions peuvent sâ€™appliquer et les
transformations quâ€™on leur fait subir. (Ref: GP)"@fr ;
      rdfs:label "EClass_Procedure"^^xsd:string ;
      rdfs:subClassOf :MOT_ConnaissanceAbstraite .
:MOT_Relation
      rdf:type owl:Class ;
      rdfs:label "EClass_Lien"^^xsd:string ;
      rdfs:subClassOf :MOT_Structure .
:MOT_Sommaire
      rdf:type owl:Class ;
      rdfs:label "EClass_Sommaire"^^xsd:string ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :VERSION
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :AUTEUR
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :TITRE
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :COMMENTAIRE
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :COPYRIGHT
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :SUJET
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :LICENSE
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :DERNIERMODIFICATION
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :NOMFICHIER
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :DATECREATION
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :MOTCLE
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "1"^^xsd:int ;
                owl:onProperty :CATEGORIE
              ] .
:MOT_Sommaire_Prop
      rdf:type owl:DatatypeProperty .
:MOT_Structure
      rdf:type owl:Class ;
      rdfs:label "EClass_RepresentationMOT"^^xsd:string .
:MOT_Trace
      rdf:type owl:Class ;
      rdfs:comment "Les traces sont obtenues en spÃ©cifiant les variables
de chacune des actions qui composent une procÃ©dure, obtenant un ensemble
dâ€™actions particuliÃ¨res bien prÃ©cises, une trace dâ€™exÃ©cution."@fr ;
      rdfs:label "EClass_Trace"^^xsd:string ;
      rdfs:subClassOf :MOT_Fait .
:MOT_connDestination
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain :MOT_Relation ;
      rdfs:range :MOT_Connaissance ;
      owl:inverseOf :MOT_estLaDestination .
:MOT_connSource
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain :MOT_Relation ;
      rdfs:range :MOT_Connaissance ;
      owl:inverseOf :MOT_estLaSource .
:MOT_estLaDestination
      rdf:type owl:ObjectProperty ;
      rdfs:domain :MOT_Connaissance ;
      rdfs:range :MOT_Relation ;
      owl:inverseOf :MOT_connDestination .
:MOT_estLaSource
      rdf:type owl:ObjectProperty ;
      rdfs:domain :MOT_Connaissance ;
      rdfs:range :MOT_Relation ;
      owl:inverseOf :MOT_connSource .
:MOT_estNonDetermine
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:range xsd:boolean .

:MOT_etiquette
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:domain :MOT_Structure ;
      rdfs:range xsd:string .
:MOT_lrConnDest
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain :MOT_LienR ;
      rdfs:range :MOT_Connaissance ;
      owl:inverseOf :MOT_lrEstLaDestination .
:MOT_lrConnSrc
      rdf:type owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain :MOT_LienR ;
      rdfs:range :MOT_Connaissance ;
      owl:inverseOf :MOT_lrEstLaSource .
:MOT_lrEstLaDestination
      rdf:type owl:ObjectProperty ;
      rdfs:domain :MOT_Connaissance ;
```

```
      rdfs:range :MOT_LienR ;
      owl:inverseOf :MOT_lrConnDest .
:MOT_lrEstLaSource
      rdf:type owl:ObjectProperty ;
      rdfs:domain :MOT_Connaissance ;
      rdfs:range :MOT_LienR .
:MOT_nomAttribut
      rdf:type owl:DatatypeProperty ;
      rdfs:range xsd:string .
:MOT_nomConnaissance
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:domain :MOT_Connaissance ;
      rdfs:range xsd:string .
:MOT_nomLien
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:domain :MOT_Relation ;
      rdfs:range xsd:string .
:MOT_ontoRefStereotype
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Structure ;
      rdfs:range xsd:string .
:NOMFICHIER
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:SUJET
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:TITRE
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
:VERSION
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :MOT_Sommaire ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MOT_Sommaire_Prop .
```

F.9 Ontologie du traitement des ambiguïtés

```
# Saved by TopBraid on Mon Nov 09 17:24:42 EST 2009
#                                                              baseURI:
http://localhost/Ontologies/OntoCASE/ontologie_traitement_ambiguite.owl
# imports: http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl
# imports: http://www.w3.org/2003/11/swrl
# imports: http://www.w3.org/2003/11/swrlb

@prefix                                                          :
<http://localhost/Ontologies/OntoCASE/ontologie_traitement_ambiguite.owl#>
.
@prefix                                                      metaMot:
<http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .

<http://localhost/Ontologies/OntoCASE/ontologie_traitement_ambiguite.owl>
      rdf:type owl:Ontology ;
      rdfs:comment "Ontologie crÃ©Ã©e dans le cadre de la thÃ¨se doctorale
de Michel HÃ©on"@fr ;
      owl:imports            <http://www.w3.org/2003/11/swrl>            ,
<http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl>              ,
<http://www.w3.org/2003/11/swrlb> ;
      owl:versionInfo "Copyrigth (c) Cotechnoe inc."@fr .
:Agent_Contrainte_Norme
      rdf:type :TA_Principe .
:Attribut
      rdf:type :TA_LienC .
:Attribut_Valeur
      rdf:type :TA_Exemple .
:Concept_Classe
      rdf:type :TA_Concept .
:Condition
      rdf:type :TA_Principe .
:ENTITE_AMBIGUE
      rdf:type :TA_TypesDentites .
:Entite_Schema_Float
      rdf:type :TA_Concept ;
      :TA_identifiant "XsdFloat"^^xsd:string .
:Entite_Schema_Int
      rdf:type :TA_Concept ;
      :TA_identifiant "XsdInt"^^xsd:string .
:Entite_Schema_String
      rdf:type :TA_Concept ;
      :TA_identifiant "XsdString"^^xsd:string .
:Holonyme
      rdf:type :TA_LienC .
:Intrant
      rdf:type :TA_IP .
:NON_DETERMINE
      rdf:type :TA_TypesDentites .
:Observable_AgentNormeContrainte
      rdf:type :TA_Enonce .
:Observable_Condition
      rdf:type :TA_Enonce .
:Observable_Objet
      rdf:type :TA_Exemple .
:Observable_Operation
      rdf:type :TA_Trace .
:Observable_Procedure
      rdf:type :TA_Trace .
:Observable_Propriete
      rdf:type :TA_Enonce .
:Observable_Regle_Antecedent
      rdf:type :TA_Enonce .
:Observable_Regle_Complete
      rdf:type :TA_Enonce .
:Observable_Regle_Conclusion
      rdf:type :TA_Enonce .
:Observable_Regle_Nom
      rdf:type :TA_Enonce .
:Operation
      rdf:type :TA_Procedure .
:Procedure
      rdf:type :TA_Procedure .
```

```
:Produit
      rdf:type :TA_IP .
:Propriete
      rdf:type :TA_Principe .
:Propriete_Unaire
      rdf:type :TA_Principe .
:Regle_Antecedent
      rdf:type :TA_Principe .
:Regle_Complete
      rdf:type :TA_Principe .
:Regle_Conclusion
      rdf:type :TA_Principe .
:Regle_Nom
      rdf:type :TA_Principe .
:Relation_Application
      rdf:type :TA_NonAmbigues .
:Relation_Englobe
      rdf:type :TA_NonAmbigues .
:Relation_Instanciation
      rdf:type :TA_NonAmbigues .
:Relation_Precedence
      rdf:type :TA_NonAmbigues .
:Relation_Regulation
      rdf:type :TA_NonAmbigues .
:Relation_Specialisation
      rdf:type :TA_NonAmbigues .
:TA_Concept
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf        (:Concept_Classe        :Entite_Schema_Int
:Entite_Schema_Float :Entite_Schema_String)
              ] .
:TA_Enonce
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf                       (:Observable_Regle_Antecedent
:Observable_Propriete                                   :Observable_Condition
:Observable_AgentNormeContrainte                        :Observable_Regle_Nom
:Observable_Regle_Conclusion :Observable_Regle_Complete)
              ] .
:TA_Exemple
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:Observable_Objet :Attribut_Valeur)
              ] .
:TA_IP
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:Intrant :Produit)
              ] .
:TA_LienC
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:Holonyme :Attribut)
              ] .
:TA_ListeDesAmbiguites
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_Modele .
:TA_Modele
      rdf:type owl:Class .
:TA_NonAmbigues
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf    (:Relation_Englobe    :Relation_Specialisation
:Relation_Instanciation     :Relation_Application     :Relation_Precedence
:Relation_Regulation)
              ] .
:TA_Principe
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf     (:Propriete     :Regle_Complete     :Condition
:Agent_Contrainte_Norme   :Regle_Conclusion   :Regle_Nom   :Propriete_Unaire
:Regle_Antecedent)
              ] .
:TA_Procedure
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:Operation :Procedure)
              ] .
:TA_Trace
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_TypesDentites ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:Observable_Operation :Observable_Procedure)
              ] .
:TA_TypesDentites
      rdf:type owl:Class ;
      rdfs:subClassOf :TA_Modele .
:TA_estNonDetermine
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:comment "Ce flag est assignÃ© pour indiquer que le type est
dÃ©terminÃ© Ã  l'Ã©tape de la crÃ©ation du modÃ¨le"^^xsd:string ;
      rdfs:domain metaMot:MOT_Structure ;
      rdfs:range xsd:boolean .
:TA_estNonDetermine_topo
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:comment "Ce flag est assignÃ© pour indiquer que le type est
dÃ©terminÃ© Ã  l'Ã©tape de la dÃ©sambiguisation topographique"^^xsd:string
;
      rdfs:domain metaMot:MOT_Structure ;
      rdfs:range xsd:boolean .
:TA_estNonDetermine_typo
      rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
      rdfs:comment "Ce flag est assignÃ© pour indiquer que le type est
dÃ©terminÃ© Ã  l'Ã©tape de la dÃ©sambiguisation typographique"^^xsd:string
;
      rdfs:domain metaMot:MOT_Structure ;
```

```
      rdfs:range xsd:boolean .
:TA_etiquette
      rdf:type owl:DatatypeProperty ;
      rdfs:range xsd:string .
:TA_identifiant
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :TA_Modele ;
      rdfs:range xsd:string .
```

F.10 Ontologie de référence

```
# Saved by TopBraid on Mon Nov 09 17:32:14 EST 2009
# baseURI: http://localhost/Ontologies/OntoCASE/ontologie_de_reference.owl

@prefix                                                                 :
<http://localhost/Ontologies/OntoCASE/ontologie_de_reference.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .

<http://localhost/Ontologies/OntoCASE/ontologie_de_reference.owl>
      rdf:type owl:Ontology ;
      rdfs:comment "Ontologie crÃ©Ã©e dans le cadre de la thÃ¨se doctorale
de Michel HÃ©on"@fr ;
      owl:versionInfo "Copyrigth (c) Cotechnoe inc."@fr .

:OR_Categorie
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Structure .
:OR_Categorie_Primaire
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie .
:OR_Categorie_Primaire_Declaratif
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Primaire .
:OR_Categorie_Primaire_Procedural
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Primaire .
:OR_Categorie_Primaire_Strategique
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Primaire .
:OR_Categorie_Regle
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie .
:OR_Categorie_Regle_Antecedent
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Regle .
:OR_Categorie_Regle_Complete
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Regle .
:OR_Categorie_Regle_Conclusion
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Regle .
:OR_Categorie_Regle_Nom
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Regle .
:OR_Categorie_Secondaire
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie .
:OR_Categorie_Secondaire_Declaratif
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Secondaire .
:OR_Categorie_Secondaire_Procedurale
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Secondaire .
:OR_Categorie_Secondaire_Strategique
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Secondaire .
:OR_Categorie_Tertiaire
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie .
:OR_Connaissance
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Structure .
:OR_Connaissance_Declarative
      rdf:type owl:Class ;
      rdfs:label "Connaissance dÃ©clarative"^^xsd:string ;
      rdfs:subClassOf :OR_Connaissance .
:OR_Connaissance_Procedurale
      rdf:type owl:Class ;
      rdfs:label "Connaissance procÃ©durale"^^xsd:string ;
      rdfs:subClassOf :OR_Connaissance .
:OR_Connaissance_Strategique
      rdf:type owl:Class ;
      rdfs:label "Connaissance statÃ©gique"^^xsd:string ;
      rdfs:subClassOf :OR_Connaissance .
:OR_Entite
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Structure .
:OR_Entite_Acte
      rdf:type owl:Class ;
      rdfs:subClassOf    :OR_Representation_Concrete    ,    :OR_Entite    ,
:OR_Connaissance_Procedurale .
:OR_Entite_Acte_Operation
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_Acte , :OR_Categorie_Primaire_Procedural .
:OR_Entite_Acte_Procedure
      rdf:type owl:Class ;
      rdfs:subClassOf         :OR_Categorie_Secondaire_Procedurale         ,
:OR_Entite_Acte .
:OR_Entite_Action
      rdf:type owl:Class ;
      rdfs:subClassOf    :OR_Representation_Abstraite    ,    :OR_Entite    ,
:OR_Connaissance_Procedurale .
:OR_Entite_Action_Operation
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Primaire_Procedural , :OR_Entite_Action
.
:OR_Entite_Action_Procedure
      rdf:type owl:Class ;
      rdfs:subClassOf         :OR_Categorie_Secondaire_Procedurale         ,
:OR_Entite_Action .
:OR_Entite_Concept
      rdf:type owl:Class ;
      rdfs:subClassOf           :OR_Representation_Abstraite            ,
:OR_Connaissance_Declarative , :OR_Entite .
:OR_Entite_Concept_Classe
      rdf:type owl:Class ;
```

```
      rdfs:subClassOf                :OR_Entite_Concept                ,
:OR_Categorie_Primaire_Declaratif .
:OR_Entite_FacteurInfluence
      rdf:type owl:Class ;
      rdfs:label "Un facteur d'influence"^^xsd:string ;
      rdfs:subClassOf    :OR_Representation_Concrete    ,    :OR_Entite    ,
:OR_Connaissance_Strategique .
:OR_Entite_FacteurInfluence_Agent_Norme_Contrainte
      rdf:type owl:Class ;
      rdfs:subClassOf             :OR_Entite_FacteurInfluence            ,
:OR_Categorie_Primaire_Strategique .
:OR_Entite_FacteurInfluence_Antecedent
      rdf:type owl:Class ;
      rdfs:subClassOf           :OR_Categorie_Regle_Antecedent           ,
:OR_Entite_FacteurInfluence_Regle .
:OR_Entite_FacteurInfluence_Conclusion
      rdf:type owl:Class ;
      rdfs:subClassOf           :OR_Categorie_Regle_Conclusion           ,
:OR_Entite_FacteurInfluence_Regle .
:OR_Entite_FacteurInfluence_Condition
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_FacteurInfluence .
:OR_Entite_FacteurInfluence_Regle
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_FacteurInfluence .
:OR_Entite_FacteurInfluence_Regle_Complete
      rdf:type owl:Class ;
      rdfs:subClassOf            :OR_Categorie_Regle_Complete            ,
:OR_Entite_FacteurInfluence_Regle .
:OR_Entite_FacteurInfluence_Regle_Nom
      rdf:type owl:Class ;
      rdfs:subClassOf              :OR_Categorie_Regle_Nom              ,
:OR_Entite_FacteurInfluence_Regle .
:OR_Entite_Manifestation
      rdf:type owl:Class ;
      rdfs:subClassOf    :OR_Connaissance_Declarative    ,    :OR_Entite    ,
:OR_Representation_Concrete .
:OR_Entite_Manifestation_Attribut
      rdf:type owl:Class ;
      rdfs:subClassOf              :OR_Entite_Manifestation              ,
:OR_Categorie_Secondaire_Declaratif .
:OR_Entite_Manifestation_Objet
      rdf:type owl:Class ;
      rdfs:subClassOf              :OR_Entite_Manifestation              ,
:OR_Categorie_Primaire_Declaratif .
:OR_Entite_Principe
      rdf:type owl:Class ;
      rdfs:comment "Les principes ou connaissances stratÃ©giques sont des
Ã©noncÃ©s permettant de dÃ©crire les propriÃ©tÃ©s des objets, dâ€™Ã©tablir
des liens de cause Ã  effet entre des objets ( le pourquoi ), ou de
dÃ©terminer dans quelles conditions appliquer une procÃ©dure ( le quand );
les principes prennent le plus souvent la forme Â »  Si telle condition
Alors, telle condition ou telle action Â ». (ref: GP)"@fr ;
      rdfs:subClassOf    :OR_Representation_Abstraite    ,    :OR_Entite    ,
:OR_Connaissance_Strategique .
:OR_Entite_Principe_Agent
      rdf:type owl:Class ;
      rdfs:subClassOf                :OR_Entite_Principe                ,
:OR_Categorie_Primaire_Strategique .
:OR_Entite_Principe_Condition
      rdf:type owl:Class ;
      rdfs:subClassOf                :OR_Entite_Principe                ,
:OR_Categorie_Secondaire_Strategique .
:OR_Entite_Principe_Regle
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_Principe .
:OR_Entite_Principe_Regle_Antecedent
      rdf:type owl:Class ;
      rdfs:subClassOf            :OR_Categorie_Regle_Antecedent           ,
:OR_Entite_Principe_Regle .
:OR_Entite_Principe_Regle_Complete
      rdf:type owl:Class ;
      rdfs:subClassOf             :OR_Categorie_Regle_Complete            ,
:OR_Entite_Principe_Regle .
:OR_Entite_Principe_Regle_Consequent
      rdf:type owl:Class ;
      rdfs:subClassOf            :OR_Categorie_Regle_Conclusion           ,
:OR_Entite_Principe_Regle .
:OR_Entite_Principe_Regle_Nom
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Regle_Nom , :OR_Entite_Principe_Regle .
:OR_Entite_Propriete
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_Principe .
:OR_Entite_Propriete_Objet
      rdf:type owl:Class ;
      rdfs:comment "Cette propriÃ©tÃ© est de type binaire, elle possÃ¨de un
domaire et une image"^^xsd:string ;
      rdfs:subClassOf :OR_Categorie_Tertiaire , :OR_Entite_Propriete .
:OR_Entite_Propriete_Objet_Observable
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Categorie_Tertiaire , :OR_Entite_Propriete .
:OR_Entite_Propriete_Objet_Unaire
      rdf:type owl:Class ;
      rdfs:comment  "Â€ la diffÃ©rence de la Propriete_Objet,  cette
propriÃ©tÃ© ne possÃ¨de qu'un domaine"^^xsd:string ;
      rdfs:subClassOf :OR_Categorie_Tertiaire , :OR_Entite_Propriete .
:OR_Entite_Schema
      rdf:type owl:Class ;
      rdfs:subClassOf                :OR_Entite_Concept                ,
:OR_Categorie_Secondaire_Declaratif .
:OR_Entite_Schema_Float
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_Schema .
:OR_Entite_Schema_Int
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_Schema .
:OR_Entite_Schema_String
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Entite_Schema .
:OR_Niveau_Representation
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Structure .
:OR_Relation
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Structure .
:OR_Relation_Antonyme
      rdf:type owl:Class ;
      rdfs:comment  "Mot  qui  a  un  sens  opposÃ©  Ã  un  autre.
Ref:http://fr.wiktionary.org/wiki/antonyme"@fr ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Attribut
      rdf:type owl:Class ;
```

```
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Defini
      rdf:type owl:Class ;
      rdfs:comment "Relation qui fait rÃ©fÃ©rence Ã  un changement de
niveau d'abstraction"^^xsd:string ;
      rdfs:subClassOf :OR_Relation_Meta .
:OR_Relation_Flux
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Flux_Intrant
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Flux .
:OR_Relation_Flux_Precedence
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Flux .
:OR_Relation_Flux_Produit
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Flux .
:OR_Relation_Holonyme
      rdf:type owl:Class ;
      rdfs:comment """L'holonymie est une relation sÃ©mantique entre mots
d'une mÃªme langue. Des termes liÃ©s par holonymie sont des holonymes.
L'holonymie est une relation partitive hiÃ©rarchisÃ©e : un holonyme A d'un
mot B est un mot dont le signifiÃ© dÃ©signe un ensemble comprenant le
signifiÃ© de B. La relation inverse est la mÃ©ronymie.
Par exemple, corps est un holonyme de bras, de mÃªme que maison est un
holonyme de toit.
RÃ©cupÃ©rÃ©e de Â »  http://fr.wikipedia.org/wiki/Holonymie Â »"""@fr ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Holonyme_Agregation
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Holonyme .
:OR_Relation_Holonyme_Composition
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Holonyme .
:OR_Relation_Holonyme_Englobe
      rdf:type owl:Class ;
      rdfs:label "relation entre un schÃ©ma et une connaissance qui lla
compose"^^xsd:string ;
      rdfs:subClassOf :OR_Relation_Holonyme .
:OR_Relation_Hyponyme
      rdf:type owl:Class ;
      rdfs:comment "DÃ©signation d'un concept subordonnÃ©."@fr ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Instance
      rdf:type owl:Class ;
      rdfs:comment "Relation qui existe en un objet de la rÃ©alitÃ©
objective et sa conceptualisation subjective (Ref: MH)"^^xsd:string ,
"""anglicisme informatique (programmation orientÃ©e objet), de instance qui
signifie Â »  cas, exemple Â » : objet appartenant Ã  une classe donnÃ©e
Ref: http://fr.wiktionary.org/wiki/instance"""@fr ;
      rdfs:subClassOf :OR_Relation_Meta .
:OR_Relation_Meta
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Operateur
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Operateur_Binaire
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur .
:OR_Relation_Operateur_Binaire_Disjoint
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Binaire .
:OR_Relation_Operateur_Binaire_Equivalent
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Binaire .
:OR_Relation_Operateur_Binaire_Inverse
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Binaire .
:OR_Relation_Operateur_Unaire
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur .
:OR_Relation_Operateur_Unaire_Condition
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire .
:OR_Relation_Operateur_Unaire_Condition_Necessaire
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Condition .
:OR_Relation_Operateur_Unaire_Condition_NecessaireEtSuffisant
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Condition .
:OR_Relation_Operateur_Unaire_Restriction
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire .
:OR_Relation_Operateur_Unaire_Restriction_Cardinalite
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Restriction .
:OR_Relation_Operateur_Unaire_Restriction_CardinaliteMax
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Restriction .
:OR_Relation_Operateur_Unaire_Restriction_CardinaliteMin
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Restriction .
:OR_Relation_Operateur_Unaire_Restriction_Existenciel
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Restriction .
:OR_Relation_Operateur_Unaire_Restriction_Universel
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Restriction .
:OR_Relation_Operateur_Unaire_Restriction_Valeur
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation_Operateur_Unaire_Restriction .
:OR_Relation_Regulation
      rdf:type owl:Class ;
      rdfs:subClassOf :OR_Relation .
:OR_Relation_Synonyme
      rdf:type owl:Class ;
      rdfs:comment "Un des termes qui dÃ©signent le mÃªme concept et qui
sont interchangeables dans tous les contextes. Aussi appelÃ© synonyme
absolu.                                                                Ref:
http://www.translationbureau.gc.ca/index.php?cont=700&lang=francais"@fr ;
      rdfs:subClassOf :OR_Relation .
:OR_Representation_Abstraite
      rdf:type owl:Class ;
      rdfs:label "ReprÃ©sentation de choses abstraites"^^xsd:string ;
      rdfs:subClassOf :OR_Niveau_Representation .
:OR_Representation_Concrete
      rdf:type owl:Class ;
      rdfs:label "ReprÃ©sentation de choses concrÃ¨tes"^^xsd:string ;
      rdfs:subClassOf :OR_Niveau_Representation .
:OR_Structure
      rdf:type owl:Class .
:OR_connDestination
```

```
      rdf:type owl:FunctionalProperty , owl:ObjectProperty .
:OR_connSource
      rdf:type owl:FunctionalProperty , owl:ObjectProperty .
:OR_estCree
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :OR_Structure ;
      rdfs:range xsd:boolean .
:OR_estLaDestination
      rdf:type owl:ObjectProperty .
:OR_estLaSource
      rdf:type owl:ObjectProperty ;
      owl:inverseOf :OR_estLaDestination .
:OR_etiquette
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :OR_Structure ;
      rdfs:range xsd:string .
:OR_identifiant
      rdf:type owl:DatatypeProperty ;
      rdfs:domain :OR_Structure ;
      rdfs:range xsd:string .
```

F.11 Ontologie cadre en représentation des connaissances d'OntoCASE

```
# Saved by TopBraid on Thu Oct 22 19:36:30 EDT 2009
# baseURI: http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl

@prefix                                                                 :
<http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl>
      a       owl:Ontology ;
      owl:versionInfo "CrÃ©e par Michel HÃ©on"^^xsd:string .

:A-POUR-ANTECEDENT
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "a pour antÃ©cÃ©dent"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :A-POUR-CONDITION ;
      owl:inverseOf :EST-ANTECEDENT-DE .
:A-POUR-ATTRIBUT
      a       owl:ObjectProperty ;
      rdfs:domain :MD_Declarative_Concept ;
      rdfs:label "a pour attribut" ;
      rdfs:range :MD_TypeDeConnaissance .
:A-POUR-COMPOSANT
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:comment "HAS-PART"^^xsd:string ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "a pour composant"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance .
:A-POUR-CONCLUSION
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "a pour conclusion"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :PERMET-DE .
:A-POUR-CONDITION
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "a pour condition"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :A-POUR-DEPENDANCE .
:A-POUR-DEPENDANCE
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "a pour dÃ©pendance"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      owl:inverseOf :PERMET-DE .
:A-POUR-INTRANT
      a       owl:ObjectProperty ;
      rdfs:label "a pour intrant"^^xsd:string ;
      rdfs:subPropertyOf :INTRANT-PRODUIT .
:A-POUR-PREALABLE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "a pour prÃ©alable"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :A-POUR-DEPENDANCE .
:A-POUR-PRODUIT
      a       owl:ObjectProperty ;
      rdfs:domain :MD_Procedurale ;
      rdfs:label "a pour produit"^^xsd:string ;
      rdfs:range :MD_Declarative_Concept ;
      rdfs:subPropertyOf :INTRANT-PRODUIT ;
      :MD_identifiant "est le produit"^^xsd:string .
:ALORS
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "alors"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :PERMET-DE .
:ENGLOBE
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "englobe"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance .
:EST-ANTECEDENT-DE
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "est antÃ©cÃ©dent de"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :PERMET-DE .
:EST-ENGLOBE-PAR
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:label "est englobÃ© par" ;
      owl:inverseOf :ENGLOBE .
:EST-LA-CONLUSION-DE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "est la conclusion de"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :A-POUR-CONDITION ;
      owl:inverseOf :A-POUR-CONCLUSION , :ALORS .
```

```
:EST-LA-PARTIE-DE
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:comment "PART-OF"^^xsd:string ;
      rdfs:label "est la partie de"^^xsd:string ;
      owl:inverseOf :A-POUR-COMPOSANT .
:EST-LE-PRODUIT-DE
      a       owl:ObjectProperty ;
      rdfs:label "est le produit de"^^xsd:string ;
      rdfs:subPropertyOf :INTRANT-PRODUIT ;
      owl:inverseOf :A-POUR-PRODUIT .
:EST-LE-SUIVANT-DE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "est la suivant de"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :A-POUR-DEPENDANCE ;
      owl:inverseOf :EST-PRECEDENT-DE .
:EST-L_INTRANT-DE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_Declarative_Concept ;
      rdfs:label "est l'intrant de"^^xsd:string ;
      rdfs:range :MD_Procedurale ;
      rdfs:subPropertyOf :INTRANT-PRODUIT ;
      :MD_identifiant "a pour intrant"^^xsd:string ;
      owl:inverseOf :A-POUR-INTRANT .
:EST-PRECEDENT-DE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "est prÃ©cÃ©dent de"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :PERMET-DE .
:EVALUE-A-PARTIR-DE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "Ã©valuÃ© Ã  partir de"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :A-POUR-DEPENDANCE ;
      owl:inverseOf :PUIS-EVALUER .
:INTRANT-PRODUIT
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "intrant/produit de"^^xsd:string ;
      rdfs:range :MD_TypeDeConnaissance .
:MD_ASSERTION
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "Les assertions du domaine" ;
      rdfs:range :MD_TypeDeConnaissance .
:MD_DataTypeValue
      a       owl:DatatypeProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "Type de donnÃ©es" .
:MD_Declarative
      a       owl:Class ;
      rdfs:label "Connaissance dÃ©clarative" ;
      rdfs:subClassOf :MD_TypeDeConnaissance .
:MD_Declarative_Concept
      a       owl:Class ;
      rdfs:label "Connaissance d'objet" ;
      rdfs:subClassOf :MD_Declarative ;
      :MD_identifiant "Connaissance dÃ©clarative"^^xsd:string .
:MD_Declarative_Schema
      a       owl:Class ;
      rdfs:label "Connaissance constantes (SchÃ©ma)" ;
      rdfs:subClassOf :MD_Declarative .
:MD_Declarative_Schema_Float
      a       owl:Class ;
      rdfs:subClassOf :MD_Declarative_Schema .
:MD_Declarative_Schema_Int
      a       owl:Class ;
      rdfs:subClassOf :MD_Declarative_Schema .
:MD_Declarative_Schema_String
      a       owl:Class ;
      rdfs:subClassOf :MD_Declarative_Schema .
:MD_PROPRIETE
      a       owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "PropriÃ©tÃ© de domaine" ;
      rdfs:range :MD_TypeDeConnaissance .
:MD_Procedurale
      a       owl:Class ;
      rdfs:label "Connaissance d'actions" ;
      rdfs:subClassOf :MD_TypeDeConnaissance ;
      :MD_identifiant "Connaissance procÃ©durale"^^xsd:string .
:MD_Procedurale_Operation
      a       owl:Class ;
      rdfs:comment "L'action est la consÃ©quence de la vÃ©recitÃ© d'un
ensemble d'antÃ©cÃ©dent"^^xsd:string ;
      rdfs:label "OpÃ©ration" ;
      rdfs:subClassOf :MD_Procedurale ;
      owl:disjointWith :MD_Procedurale_Procedure .
:MD_Procedurale_Procedure
      a       owl:Class ;
      rdfs:label "ProcÃ©dure" ;
      rdfs:subClassOf :MD_Procedurale ;
      owl:disjointWith :MD_Procedurale_Operation .
:MD_Strategique
      a       owl:Class ;
      rdfs:label "Connaissance stratÃ©gique" ;
      rdfs:subClassOf :MD_TypeDeConnaissance ;
      :MD_identifiant "Connaissance stratÃ©gique"^^xsd:string ;
      owl:disjointWith   :MD_Declarative_Schema   ,   :MD_Procedurale   ,
:MD_Declarative_Concept .
:MD_Strategique_AgentContrainteNorme
      a       owl:Class ;
      rdfs:label "Agent-Contrainte-Norme" ;
      rdfs:subClassOf :MD_Strategique ;
      :MD_identifiant "Agent, Contrainte ou Norme"^^xsd:string ;
      owl:disjointWith            :MD_Strategique_Entite_Regle           ,
:MD_Strategique_Condition .
:MD_Strategique_Condition
      a       owl:Class ;
      rdfs:label "Condition" ;
      rdfs:subClassOf :MD_Strategique ;
      owl:disjointWith            :MD_Strategique_Entite_Regle           ,
:MD_Strategique_AgentContrainteNorme .
:MD_Strategique_Entite_Regle
      a       owl:Class ;
      rdfs:comment  "Contient  les  entitÃ©s  associÃ©s  Ã  une  rÃ¨gle
(consÃ©quent et conclusion)"^^xsd:string ;
      rdfs:label "Ã‰lÃ©ment d'une rÃ¨gle" ;
      rdfs:subClassOf :MD_Strategique ;
      owl:disjointWith        :MD_Strategique_AgentContrainteNorme       ,
:MD_Strategique_Condition .
```

```
:MD_Strategique_Entite_Regle_Antecedent
      a       owl:Class ;
      rdfs:label "AntÃ©cÃ©dent d'une rÃ¨gle" ;
      rdfs:subClassOf :MD_Strategique_Entite_Regle ;
      owl:disjointWith          :MD_Strategique_Entite_Regle_Nom         ,
:MD_Strategique_Entite_Regle_Complete                                    ,
:MD_Strategique_Entite_Regle_Conclusion .
:MD_Strategique_Entite_Regle_Complete
      a       owl:Class ;
      rdfs:label "RÃ¨gle complÃ¨te" ;
      rdfs:subClassOf :MD_Strategique_Entite_Regle ;
      owl:disjointWith          :MD_Strategique_Entite_Regle_Nom         ,
:MD_Strategique_Entite_Regle_Antecedent                                  ,
:MD_Strategique_Entite_Regle_Conclusion .
:MD_Strategique_Entite_Regle_Conclusion
      a       owl:Class ;
      rdfs:label "Conclusion d'une rÃ¨gle" ;
      rdfs:subClassOf :MD_Strategique_Entite_Regle ;
      owl:disjointWith          :MD_Strategique_Entite_Regle_Nom         ,
:MD_Strategique_Entite_Regle_Complete                                    ,
:MD_Strategique_Entite_Regle_Antecedent .
:MD_Strategique_Entite_Regle_Nom
      a       owl:Class ;
      rdfs:label "Nom d'une rÃ¨gle" ;
      rdfs:subClassOf :MD_Strategique_Entite_Regle ;
      owl:disjointWith       :MD_Strategique_Entite_Regle_Complete      ,
:MD_Strategique_Entite_Regle_Antecedent                                  ,
:MD_Strategique_Entite_Regle_Conclusion .
:MD_TypeDeConnaissance
      a       owl:Class ;
      rdfs:label "Type de connaissances" .
:MD_identifiant
      a       owl:DatatypeProperty ;
      rdfs:range xsd:string .
:MD_isFloat
      a       owl:DatatypeProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "Point flottant" ;
      rdfs:range xsd:float ;
      rdfs:subPropertyOf :MD_DataTypeValue .
:MD_isInteger
      a       owl:DatatypeProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "Nombre entier" ;
      rdfs:range xsd:int ;
      rdfs:subPropertyOf :MD_DataTypeValue .
:MD_isString
      a       owl:DatatypeProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "ChaÃ®ne de caractÃ¨res" ;
      rdfs:range xsd:string ;
      rdfs:subPropertyOf :MD_DataTypeValue .
:OBEIT-A
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "obÃ©it Ã  "^^xsd:string ;
      owl:inverseOf :REGIT .
:PERMET-DE
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:label "permet"^^xsd:string .
:PUIS-EVALUER
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:label "puis Ã©valuer"^^xsd:string ;
      rdfs:subPropertyOf :PERMET-DE .
:PUIS-EXECUTER
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_TypeDeConnaissance ;
      rdfs:label "puis exÃ©cuter" ;
      rdfs:range :MD_TypeDeConnaissance ;
      rdfs:subPropertyOf :PERMET-DE .
:REGIT
      a       owl:TransitiveProperty , owl:ObjectProperty ;
      rdfs:domain :MD_Strategique ;
      rdfs:label "rÃ©git"^^xsd:string .
```

F.12 Ontologie de transformation

```
# Saved by TopBraid on Mon Nov 09 18:01:20 EST 2009
#                                                                  baseURI:
http://localhost/Ontologies/OntoCASE/ontologie_de_transformation.owl
# imports: http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl
# imports: http://localhost/Ontologies/OntoCASE/ontologie_de_reference.owl
#                                                                  imports:
http://localhost/Ontologies/OntoCASE/ontologie_traitement_ambiguite.owl
# imports: http://www.w3.org/2003/11/swrl
# imports: http://www.w3.org/2003/11/swrlb
# imports: http://localhost/Ontologies/SWRL/swrlbi.owl
# imports: http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl

@prefix                                                              :
<http://localhost/Ontologies/OntoCASE/ontologie_de_transformation.owl#> .
@prefix                                                      metaDom:
<http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl#> .
@prefix                                                      metaMot:
<http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl#> .
@prefix                                                       oAmbig:
<http://localhost/Ontologies/OntoCASE/ontologie_traitement_ambiguite.owl#>
.
@prefix                                                         oRef:
<http://localhost/Ontologies/OntoCASE/ontologie_de_reference.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix protege:  <http://protege.stanford.edu/plugins/owl/protege#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix swrlbi:  <http://localhost/Ontologies/SWRL/swrlbi.owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
metaMot:MOT_connDestination
      rdfs:subPropertyOf oRef:OR_connDestination .
metaMot:MOT_connSource
    rdfs:subPropertyOf oRef:OR_connSource .
metaMot:MOT_estLaDestination
    rdfs:subPropertyOf oRef:OR_estLaDestination .
metaMot:MOT_estLaSource
    rdfs:subPropertyOf oRef:OR_estLaSource .
metaMot:MOT_estNonDetermine
    rdfs:subPropertyOf oAmbig:TA_estNonDetermine .
metaMot:MOT_etiquette
    rdfs:subPropertyOf oAmbig:TA_etiquette .
metaMot:MOT_lrConnDest
    rdfs:subPropertyOf oRef:OR_connDestination .
metaMot:MOT_lrConnSrc
```

```
    rdfs:subPropertyOf oRef:OR_connSource .
metaMot:MOT_lrEstLaDestination
    rdfs:subPropertyOf oRef:OR_estLaDestination .
metaMot:MOT_lrEstLaSource
    rdfs:subPropertyOf oRef:OR_estLaSource .
metaMot:MOT_nomConnaissance
    rdfs:subPropertyOf oRef:OR_identifiant .
metaMot:MOT_nomLien
    rdfs:subPropertyOf oRef:OR_identifiant .
oRef:OR_Entite_Acte_Operation
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Operation ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Acte_Procedure
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Procedure ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Action_Operation
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Operation ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Action_Procedure
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Procedure ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Concept_Classe
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Concept_Classe ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_FacteurInfluence_Agent_Norme_Contrainte
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_AgentNormeContrainte ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_FacteurInfluence_Antecedent
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Regle_Antecedent ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_FacteurInfluence_Conclusion
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Regle_Conclusion ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_FacteurInfluence_Condition
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Condition ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_FacteurInfluence_Regle_Complete
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Regle_Complete ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_FacteurInfluence_Regle_Nom
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Regle_Nom ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Manifestation_Attribut
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Attribut_Valeur ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Manifestation_Objet
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Objet ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Principe_Agent
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Agent_Contrainte_Norme ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Principe_Condition
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Condition ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Principe_Regle_Antecedent
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Regle_Antecedent ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Principe_Regle_Complete
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Regle_Complete ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Principe_Regle_Consequent
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Regle_Conclusion ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Principe_Regle_Nom
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Regle_Nom ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Propriete_Objet
```

```
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Propriete ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Propriete_Objet_Observable
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Observable_Propriete ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Propriete_Objet_Unaire
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Propriete_Unaire ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Schema_Float
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Entite_Schema_Float ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Schema_Int
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Entite_Schema_Int ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Entite_Schema_String
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Entite_Schema_String ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Attribut
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Attribut ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Defini
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Relation_Application ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Flux_Intrant
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Intrant ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Flux_Precedence
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Relation_Precedence ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Flux_Produit
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Produit ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Holonyme_Composition
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Holonyme ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Holonyme_Englobe
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Relation_Englobe ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Hyponyme
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Relation_Specialisation ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Instance
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Relation_Instanciation ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_Relation_Regulation
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:Relation_Regulation ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oRef:OR_etiquette
    rdfs:subPropertyOf :OT_etiquette .
<http://localhost/Ontologies/OntoCASE/ontologie_de_transformation.owl>
    rdf:type owl:Ontology ;
    rdfs:comment "Ontologie crÃ©Ã©e dans le cadre de la thÃ¨se doctorale de
Michel HÃ©on"@fr ;
    owl:imports
<http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl>             ,
<http://www.w3.org/2003/11/swrl>                                          ,
<http://localhost/Ontologies/OntoCASE/meta_langage/mot.owl>               ,
<http://localhost/Ontologies/SWRL/swrlbi.owl>                             ,
<http://localhost/Ontologies/OntoCASE/ontologie_traitement_ambiguite.owl> ,
<http://www.w3.org/2003/11/swrlb>                                         ,
<http://localhost/Ontologies/OntoCASE/ontologie_de_reference.owl> ;
    owl:versionInfo "Copyrigth (c) Cotechnoe inc."@fr .
:OT_DataOntologieDeDomaine
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ModeleDeTransformation .
:OT_DonneesDeTravail
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ModeleDeTransformation .
:OT_ERREUR_HYPONYME
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_Element_Gestion_Erreur .
:OT_ERREUR_HYPONYME_Categorie
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME .
:OT_ERREUR_HYPONYME_CategoriePrimaire
    rdf:type owl:Class ;
```

```
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Secondaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Regle
                          ])
                    ])
              ])
        ] .

:OT_ERREUR_HYPONYME_CategorieRegle_ANTECEDENT
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Secondaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                          ])
                    ])
              ])
        ] .
:OT_ERREUR_HYPONYME_CategorieRegle_COMPLETE
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
```

```
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Primaire
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Secondaire
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                            ])
                      ])
                ])
        ] .

:OT_ERREUR_HYPONYME_CategorieRegle_CONCLUSION
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Primaire
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Secondaire
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connSource ;
                            owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                            ] [ rdf:type owl:Restriction ;
                            owl:onProperty oRef:OR_connDestination ;
                            owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                            ])
                      ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                      owl:intersectionOf ([ rdf:type owl:Restriction ;
```

```
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ])
                  ])
            ])
      ] .

:OT_ERREUR_HYPONYME_CategorieRegle_NOM
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ])
                  ])
            ])
      ] .

:OT_ERREUR_HYPONYME_CategorieSecondaire
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
```

```
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_HYPONYME_CategorieTertiaire
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME_Categorie ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Hyponyme   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_HYPONYME_Schema
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_HYPONYME ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:intersectionOf  (oRef:OR_Relation_Hyponyme [ rdf:type owl:Class
;
              owl:intersectionOf ([ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom oRef:OR_Entite_Schema
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Entite_Schema
                    ])
              ])
        ] .
:OT_ERREUR_INSTANCE
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_Element_Gestion_Erreur .
:OT_ERREUR_INSTANCE_Agent
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Declaratif
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
```

```
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Procedural
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Entite_FacteurInfluence_Condition
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_INSTANCE_CategoriePrimaire
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:intersectionOf  (oRef:OR_Relation_Instance [ rdf:type owl:Class
;
            owl:intersectionOf ([ rdf:type owl:Restriction ;
                  owl:onProperty oRef:OR_connSource ;
                  owl:someValuesFrom oRef:OR_Categorie_Primaire
                  ] [ rdf:type owl:Restriction ;
                  owl:onProperty oRef:OR_connDestination ;
                  owl:someValuesFrom oRef:OR_Categorie_Secondaire
                  ])
            ])
      ] .

:OT_ERREUR_INSTANCE_CategorieSecondaire
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:intersectionOf  (oRef:OR_Relation_Instance [ rdf:type owl:Class
;
            owl:intersectionOf ([ rdf:type owl:Restriction ;
                  owl:onProperty oRef:OR_connSource ;
                  owl:someValuesFrom oRef:OR_Categorie_Secondaire
                  ] [ rdf:type owl:Restriction ;
                  owl:onProperty oRef:OR_connDestination ;
                  owl:someValuesFrom oRef:OR_Categorie_Primaire
                  ])
            ])
      ] .

:OT_ERREUR_INSTANCE_Complete
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
```

```
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_INSTANCE_Condition
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Declaratif
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Procedurale
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Strategique
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_INSTANCE_Operation
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                  owl:onProperty oRef:OR_connSource ;
                  owl:someValuesFrom
oRef:OR_Categorie_Primaire_Procedural
                  ] [ rdf:type owl:Restriction ;
                  owl:onProperty oRef:OR_connDestination ;
                  owl:someValuesFrom
oRef:OR_Categorie_Primaire_Strategique
```

```
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Primaire_Procedural
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom
oRef:OR_Categorie_Primaire_Declaratif
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Primaire_Procedural
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Categorie_Secondaire
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Primaire_Procedural
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Primaire_Procedural
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Categorie_Regle
                    ])
              ])
        ] .
:OT_ERREUR_INSTANCE_Procedure
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Strategique
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Declaratif
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Categorie_Primaire
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Categorie_Regle
                    ])
              ])
        ] .
:OT_ERREUR_INSTANCE_Propriete
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
```

```
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_INSTANCE_RegleAntecedent
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_INSTANCE_RegleConclusion
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
```

```
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ])
                  ])
            ])
      ] .
:OT_ERREUR_INSTANCE_RegleNom
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_INSTANCE ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Primaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Secondaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom oRef:OR_Categorie_Tertiaire
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Antecedent
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
                  owl:intersectionOf ([ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connSource ;
                        owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                        ] [ rdf:type owl:Restriction ;
                        owl:onProperty oRef:OR_connDestination ;
                        owl:someValuesFrom
oRef:OR_Categorie_Regle_Complete
                        ])
                  ])
            ] [ rdf:type owl:Class ;
            owl:intersectionOf   (oRef:OR_Relation_Instance   [   rdf:type
owl:Class ;
```

```
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom oRef:OR_Categorie_Regle_Nom
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom
oRef:OR_Categorie_Regle_Conclusion
                          ])
                    ])
              ])
        ] .
:OT_ERREUR_Message
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:range xsd:string .
:OT_ERREUR_REGULATION
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_Element_Gestion_Erreur .
:OT_ERREUR_REGULATION_ConnaissanceDeclarative
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_REGULATION ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf  (oRef:OR_Relation_Regulation  [  rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Connaissance_Declarative
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf  (oRef:OR_Relation_Regulation  [  rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Connaissance_Declarative
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Secondaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf  (oRef:OR_Relation_Regulation  [  rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Connaissance_Declarative
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Regle
                          ])
                    ])
              ])
        ] .
:OT_ERREUR_REGULATION_ConnaissanceProcedurale
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_REGULATION ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf  (oRef:OR_Relation_Regulation  [  rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Connaissance_Procedurale
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Primaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf  (oRef:OR_Relation_Regulation  [  rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Connaissance_Procedurale
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Secondaire
                          ])
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf  (oRef:OR_Relation_Regulation  [  rdf:type
owl:Class ;
                    owl:intersectionOf ([ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connSource ;
                          owl:someValuesFrom
oRef:OR_Connaissance_Procedurale
                          ] [ rdf:type owl:Restriction ;
                          owl:onProperty oRef:OR_connDestination ;
                          owl:someValuesFrom oRef:OR_Categorie_Regle
                          ])
                    ])
              ])
        ] .
:OT_ERREUR_REL_ATTRIBUT
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_Element_Gestion_Erreur .
:OT_ERREUR_REL_ATTRIBUT_PROCEDURAL
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ERREUR_REL_ATTRIBUT ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:unionOf ([ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Attribut   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom oRef:OR_Connaissance_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Connaissance_Procedurale
                    ])
              ] [ rdf:type owl:Class ;
              owl:intersectionOf   (oRef:OR_Relation_Attribut   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
```

```
                    owl:someValuesFrom oRef:OR_Connaissance_Strategique
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Connaissance_Strategique
                    ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Attribut   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom oRef:OR_Connaissance_Strategique
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Connaissance_Procedurale
                    ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Attribut   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom oRef:OR_Connaissance_Procedurale
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom oRef:OR_Connaissance_Strategique
                    ])
                ] [ rdf:type owl:Class ;
                owl:intersectionOf   (oRef:OR_Relation_Attribut   [   rdf:type
owl:Restriction ;
                    owl:onProperty oRef:OR_connSource ;
                    owl:someValuesFrom
oRef:OR_Categorie_Secondaire_Declaratif
                    ] [ rdf:type owl:Restriction ;
                    owl:onProperty oRef:OR_connDestination ;
                    owl:someValuesFrom
oRef:OR_Categorie_Primaire_Declaratif
                    ])
                ])
        ] .
:OT_Element_Gestion_Erreur
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ModeleDeTransformation .

:OT_EntiteEstDeType
    rdf:type owl:ObjectProperty ;
    rdfs:comment "Permet d'identifier le type pour la dÃ©sambiguisation"@fr
;
    rdfs:domain metaMot:MOT_Structure ;
    rdfs:range oAmbig:TA_TypesDentites .

:OT_Etat_execution
    rdf:type owl:Class ;
    rdfs:comment "Cette classe permet de contrÃ´ler le cycle d'exÃ©cution
des rÃ¨gles de transformation"^^xsd:string ;
    rdfs:subClassOf :OT_ModeleDeTransformation ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:oneOf   (:initial   :creation_class   :classification   :final
:validation)
        ] .
:OT_LienRDouble
    rdf:type owl:Class ;
    rdfs:comment "Cette classe permet d'identifier les principes qui sont
entre deux liens R. Elle s'insÃ¨re dans le processus de classification des
principes en entite de domaine qui est soit une entitePrescriptive ou un
attribut"@fr ;
    rdfs:subClassOf :OT_DonneesDeTravail .
:OT_MessageErreur
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_Element_Gestion_Erreur .
:OT_MessageErreur_HYPONYME
    rdf:type :OT_MessageErreur ;
    :OT_ERREUR_Message "La relation d'hyponymie doit unir des connaissances
abstraites de mÃªme catÃ©gorie et de mÃªme type"^^xsd:string .
:OT_MessageErreur_HYPONYME_Schema
    rdf:type :OT_MessageErreur ;
    :OT_ERREUR_Message  "La  relation  d'hyponymie  ne  peut  pas  unir  deux
schÃ©mas"^^xsd:string .
:OT_MessageErreur_INSTANCE
    rdf:type :OT_MessageErreur ;
    :OT_ERREUR_Message "Une abstraction est son instance doivent Ãªtre de
mÃªme catÃ©gorie"^^xsd:string .
:OT_MessageErreur_REGULATION
    rdf:type :OT_MessageErreur ;
    :OT_ERREUR_Message "La relation de rÃ©gulation n'est pas adaptÃ© pour
cette situation"^^xsd:string .
:OT_MessageErreur_REL_ATTRIBUT
    rdf:type :OT_MessageErreur ;
    :OT_ERREUR_Message "La relation d'attribution n'est pas valide pour la
catÃ©gorie des entitÃ©s concernÃ©es."^^xsd:string .
:OT_ModeleDeTransformation
    rdf:type owl:Class .
:OT_Sequence_Desambiguisation
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ModeleDeTransformation ;
    owl:equivalentClass
        [ rdf:type owl:Class ;
        owl:oneOf                          (:desambiguisation_topologique
:desambiguisation_typologique)
        ] .
:OT_TypeDesObjets
    rdf:type owl:Class ;
    rdfs:subClassOf :OT_ModeleDeTransformation .
:OT_URL
    rdf:type owl:DatatypeProperty ;
    rdfs:domain :OT_Etat_execution ;
    rdfs:range xsd:string .
:OT_defaultUriLocation
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain :OT_DataOntologieDeDomaine ;
    rdfs:range xsd:string .
:OT_estCree
    rdf:type owl:DatatypeProperty ;
    rdfs:domain :OT_DataOntologieDeDomaine ;
    rdfs:range xsd:boolean .
:OT_estLaDestinationDuneRegulation
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain metaMot:MOT_Connaissance ;
    rdfs:range xsd:boolean .
:OT_estLaSourceDuneRegulation
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain metaMot:MOT_Connaissance ;
    rdfs:range xsd:boolean .
:OT_etiquette
    rdf:type owl:DatatypeProperty ;
    rdfs:range xsd:string .
```

```
:OT_identifiant
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain :OT_ModeleDeTransformation ;
    rdfs:range xsd:string .
:OT_nameSpace
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain :OT_DataOntologieDeDomaine ;
    rdfs:range xsd:string .
:OT_nomEntiteDestination
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:comment "Identifiant de l'entite qui est pointÃ©e par un lienR"@fr
;
    rdfs:domain metaMot:MOT_Connaissance ;
    rdfs:range xsd:string .
:OT_nomEntiteSource
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:comment "Identifiant de l'entite source qui pointe par le lienR le
principe {@fr}"^^xsd:string ;
    rdfs:domain metaMot:MOT_Connaissance ;
    rdfs:range xsd:string .
:OT_nomEtat
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain :OT_Etat_execution ;
    rdfs:range xsd:string .
:OT_nomOntologieEtat
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain :OT_Etat_execution ;
    rdfs:range xsd:string .
:OT_nomRepertoireDeTravail
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:domain :OT_Etat_execution ;
    rdfs:range xsd:string .
:OT_prefix
    rdf:type owl:FunctionalProperty , owl:DatatypeProperty ;
    rdfs:comment "unitisÃ© par: infoOntologieDeDomaine"^^xsd:string ;
    rdfs:domain :OT_DonneesDeTravail ;
    rdfs:range xsd:string .
:Type_Classe
    rdf:type :OT_TypeDesObjets .
:Type_DataTypeProperty
    rdf:type :OT_TypeDesObjets .
:classification
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_clas
sification.owl"^^xsd:string .
:creation_class
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_crea
tion.owl"^^xsd:string .
:creation_patron
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_patr
on.owl"^^xsd:string .
:creation_prop_objet
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_prop
objet.owl"^^xsd:string .
:desambiguisation_topologique
    rdf:type :OT_Sequence_Desambiguisation ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regle_desambig/regles_desambig_
topologique.owl"^^xsd:string .
:desambiguisation_typologique
    rdf:type :OT_Sequence_Desambiguisation ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regle_desambig/regles_desambig_
typologique.owl"^^xsd:string .
:final
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_fina
l.owl"^^xsd:string .
:infoOntologieDeDomaine
    rdf:type :OT_DonneesDeTravail ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontocasemetamodele.owl"^^xsd:string ;
    :OT_prefix "metaDom"^^xsd:string .
:initial
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_init
.owl"^^xsd:string .
:validation
    rdf:type :OT_Etat_execution ;
    :OT_URL
"http://localhost/Ontologies/OntoCASE/ontos_regles_transformation/etat_vali
dation.owl"^^xsd:string .
oAmbig:TA_ListeDesAmbiguites
    owl:equivalentClass
        [ rdf:type owl:Restriction ;
        owl:hasValue oAmbig:ENTITE_AMBIGUE ;
        owl:onProperty :OT_EntiteEstDeType
        ] .
oAmbig:TA_etiquette
    rdfs:subPropertyOf oRef:OR_etiquette .
```

F.13 Bibliothèque de codes *Java* nécessaire à l'implantation du modèle de conception *commande*

F.13.1 Structure de la *built-in command* swrlbi.owl

```
# Saved by TopBraid on Tue Nov 10 11:15:46 EST 2009
# baseURI: http://localhost/Ontologies/SWRL/swrlbi.owl
# imports: http://www.w3.org/2003/11/swrlb
# imports: http://www.w3.org/2003/11/swrl

@prefix :        <http://localhost/Ontologies/SWRL/swrlbi.owl#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrl:    <http://www.w3.org/2003/11/swrl#> .
@prefix swrlb:   <http://www.w3.org/2003/11/swrlb#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix xsp:     <http://www.owl-ontologies.com/2005/08/07/xsp.owl#> .
<http://localhost/Ontologies/SWRL/swrlbi.owl>
```

```
      rdf:type owl:Ontology ;
      rdfs:comment "Ontologie crÃ©Ã©e dans le cadre de la thÃ¨se doctorale
de Michel HÃ©on"@fr ;
      owl:imports             <http://www.w3.org/2003/11/swrl>            ,
<http://www.w3.org/2003/11/swrlb> ;
      owl:versionInfo "Copyrigth (c) 2009  Cotechnoe inc."@fr .
:invoker
      rdf:type swrl:Builtin ;
      swrlb:minArgs "1"^^xsd:int .

swrl:argument2
      rdf:type owl:ObjectProperty .
```

F.13.2 Implantation de SWRLBuiltInLibraryImpl.java

```
package edu.stanford.smi.protegex.owl.swrl.bridge.builtins.swrlbi;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Logger;

import com.cotechnoe.util.command.Command;
import com.cotechnoe.util.command.InvokerProxy;

import edu.stanford.smi.protegex.owl.swrl.bridge.BuiltInArgument;
import edu.stanford.smi.protegex.owl.swrl.bridge.SWRLRuleEngineBridge;
import
edu.stanford.smi.protegex.owl.swrl.bridge.builtins.AbstractSWRLBuiltInLibra
ry;
import edu.stanford.smi.protegex.owl.swrl.bridge.builtins.SWRLBuiltInUtil;
import
edu.stanford.smi.protegex.owl.swrl.bridge.exceptions.BuiltInException;
import
edu.stanford.smi.protegex.owl.swrl.bridge.exceptions.InvalidBuiltInArgument
Exception;

/**
 * <p>Titre : SWRLBuiltInLibraryImpl </p>
 * <p>Description : Prototype de thèse doctorale</p>
 * <p>Copyright : Copyright (c) 2008</p>
 * <p>Société : Cotechnoe inc.</p>
 * @author Michel Héon
 * @version 1.0
 */
public class SWRLBuiltInLibraryImpl extends AbstractSWRLBuiltInLibrary {
    public SWRLBuiltInLibraryImpl() { super("SWRLCoreBuiltIns"); }

      private           static           Logger           logger          =
Logger.getLogger(SWRLBuiltInLibraryImpl.class.getName());
    private String className = "["+this.getClass().getSimpleName()+ "] " ;
    private InvokerProxy _invoker;
    private SWRLRuleEngineBridge bridge;

    public boolean imprimeAlloWorld(List<BuiltInArgument> arguments) throws
BuiltInException {
        logger.warning("Allo World de SwrlBuiltIn");
        String arg1;
        try {
            arg1 = SWRLBuiltInUtil.getArgumentAsAString(0, arguments);
            logger.warning("L'argument est:" + arg1);
        } catch (Exception ex) {
            logger.warning("oups! ça pas marché" + arguments.toString());
        }
        return true;
    }
    public boolean ifNotFalseThenTrue(List<BuiltInArgument> arguments)
throws BuiltInException {
        SWRLBuiltInUtil.checkNumberOfArgumentsEqualTo(1, arguments.size());
        if (!SWRLBuiltInUtil.areAllArgumentsBooleans(arguments))
            throw new InvalidBuiltInArgumentException(1, "expecting a
Boolean");
        boolean operationResult = !SWRLBuiltInUtil.getArgumentAsABoolean(0,
arguments);
        if (operationResult == false )return false;
        else return true;
    }
    public boolean ifNotTrueThenFalse(List<BuiltInArgument> arguments)
throws BuiltInException {
        SWRLBuiltInUtil.checkNumberOfArgumentsEqualTo(1, arguments.size());
        if (!SWRLBuiltInUtil.areAllArgumentsBooleans(arguments))
            throw new InvalidBuiltInArgumentException(1, "expecting a
Boolean");
        boolean operationResult = !SWRLBuiltInUtil.getArgumentAsABoolean(0,
arguments);
        System.out.println("La valeur est:" +operationResult);
        if (operationResult == true )return true;
        else return false;
    }
    public boolean invoker(List<BuiltInArgument> arguments) throws
BuiltInException {
        String ruleName = getInvokingRuleName();
        System.out.println("Nom de la règle invoquée: ["+ ruleName+"]");
        boolean returnValue = true;
        String commandeName;
        try {
            commandeName      =      SWRLBuiltInUtil.getArgumentAsAString(0,
arguments);
            Command                   commande         =          (Command)
Class.forName("com.cotechnoe.noyau.command.concrete."+commandeName).newInst
ance();
          List<String> cmdArg = new ArrayList<String>();
            for (int i = 1; i < arguments.size(); i++) {
              String value = null;
              try {
              value = SWRLBuiltInUtil.getArgumentAsAString(i, arguments);
              } catch ( InvalidBuiltInArgumentException e){
                  value                                                  =
Boolean.toString(SWRLBuiltInUtil.getArgumentAsABoolean(i, arguments));
              }
              cmdArg.add(value);
              }
            logger.warning(className+"Appel de la commande:"+commandeName +
cmdArg.toString());
            commande.setArgument(cmdArg);
            returnValue = commande.invoke();
        } catch (Exception ex) {
            logger.severe("oups!  ça  pas  marché"  +  arguments.toString()+
"("+ex.getMessage()+")");
            ex.printStackTrace();
            returnValue = false;
        }
```

```
        return returnValue;
    }

    public void initialize(SWRLRuleEngineBridge sWRLRuleEngineBridge) {
        this.bridge = sWRLRuleEngineBridge;
        _invoker = InvokerProxy.getInstance();
    }

     @Override
     public void reset() throws BuiltInException {
     }

}
```

F.13.3 Interface Command.java

```
package com.cotechnoe.util.command;
import java.util.List;
/**
 * <p>Titre : Command</p>
 * <p>Copyright : Copyright (c) 2006</p>
 * <p>Société : Cotechnoe inc.</p>
 * @author Michel Héon
 * @version 1.0
 */
public interface Command {
     public void setArgument (List<String> arguments);

    /**
     * La methode execute() est appelé par l'invoker afin d'exécuter
l'opération
    public Boolean execute();
    public Boolean invoke();
}
```

F.13.4 Interface Receiver.java

```
package com.cotechnoe.util.command;
/**
 * <p>Titre : Receiver</p>
 * <p>Description: knows how to perform the operations associated with
carrying
 * out a request. Any class may serve as a Receiver. </p>
 * <p>Copyright : Copyright (c) 2006</p>
 * <p>Société : Cotechnoe inc.</p>
 * @author Michel Héon
 * @version 1.0
 */
public interface Receiver{
    public Boolean action();
}
```

F.13.5 Interface Invoker.java

```
package com.cotechnoe.util.command;
/**
 * <p>Titre : Invoker</p>
 * <p>Copyright : Copyright (c) 2006</p>
 * <p>Société : Cotechnoe inc.</p>
 * @author Michel Héon
 * @version 1.0
 */

public interface Invoker {
    public Boolean stocke(Command command) ;
}
```

F.13.6 Implantation InvokerImpl.java

```
package com.cotechnoe.util.command;

import java.rmi.RemoteException;

public class InvokerImpl  implements Invoker {
    Command _command = null;

    public InvokerImpl() {
    }

    /**
     * Méthode permettant la soumission d'une commande
     * @param command
     * @return
     * @throws RemoteException
     */
    public Boolean stocke(Command command) {
        _command = command;
        return _command.execute();
    }
}
```

F.14 Quelques implantations de *Receiver*

F.14.1 Implantation de CreerUneOntologieCmdReceiver.java

```
package com.cotechnoe.noyau.command.concrete.receiver;

import java.util.ArrayList;
import java.util.Collection;
import java.util.logging.Logger;
```

```
import com.cotechnoe.noyau.OntologieRegister;
import com.cotechnoe.util.command.Receiver;

import edu.stanford.smi.protege.exception.OntologyLoadException;
import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.jena.creator.NewOwlProjectCreator;

/* <p>Titre : CreerUneOntologieCmdReceiver</p>
*
* <p>Copyright : Copyright (c) 2008</p>
*
* <p>Société : Cotechnoe inc.</p>
*
* @author Michel Héon
* @version 1.0
*/

public class CreerUneOntologieCmdReceiver implements Receiver {
    private           static           Logger           logger           =
Logger.getLogger(CreerUneOntologieCmdReceiver.class.getName());
     private String loggerLabel =   "["+this.getClass().getSimpleName()+"]
";
    private JenaOWLModel owlModel;
    private String nomDuDomaine = null;
    private OntologieRegister ontologieRegister;

    public CreerUneOntologieCmdReceiver() {
    }
    public Boolean action() {
          try {
              ontologieRegister = OntologieRegister.getInstance();
               Collection errors = new ArrayList();
               NewOwlProjectCreator creator = new NewOwlProjectCreator();
               creator.setOntologyName("");
               creator.create(errors);
               owlModel =creator.getOwlModel();

     owlModel.getNamespaceManager().setDefaultNamespace(nomDuDomaine+"#");
          ontologieRegister.setOwlDomainModel(owlModel);

          } catch (OntologyLoadException e) {
              e.printStackTrace();
          }
        logger.warning("["+this.getClass().getSimpleName()+"]  "  +"Création
du domaine: "+nomDuDomaine );
        return new Boolean(true);
    }
     public void setNomDuDomaine(String nomDuDomaine) {
          this.nomDuDomaine = nomDuDomaine;
     }
}
```

F.14.2 Implantation de OWLCreerUneClasseCmdReceiver.java

```
package com.cotechnoe.noyau.command.concrete.receiver;
import java.util.logging.Logger;
import com.cotechnoe.noyau.OntologieRegister;
import com.cotechnoe.util.command.Receiver;
import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.model.OWLNamedClass;
/* <p>Titre : OWLCreerUneClasseCmdReceiver</p>
* <p>Description : Création d'une classe OWL</p>
* <p>Copyright : Copyright (c) 2008</p>
* <p>Société : Cotechnoe inc.</p>
* @author Michel Héon
* @version 1.0
*/
public class OWLCreerUneClasseCmdReceiver implements Receiver {
    private           static           Logger           logger           =
Logger.getLogger(OWLCreerUneClasseCmdReceiver.class.getName());
     private String loggerLabel =   "["+this.getClass().getSimpleName()+"]
";
    private String nomClasse = null;
    private String etiquette = "";
     private JenaOWLModel owlModel;
    private OntologieRegister ontologieRegister;
    public OWLCreerUneClasseCmdReceiver() {
        ontologieRegister = OntologieRegister.getInstance();
    }
    public Boolean action() {
     owlModel = ontologieRegister.getOwlDomainModel();
       logger.info(loggerLabel +"Création de la classe: "+ nomClasse);
         try {
             OWLNamedClass                sousClasse                   =
owlModel.createOWLNamedClass(nomClasse);
              sousClasse.addLabel(etiquette, "");
          } catch (Exception e) {
             logger.severe(loggerLabel   +"Création   de   la   classe:   "+
nomClasse+ " n'est pas possible :"+ e.getMessage());
              return new Boolean(false);
          }
     return new Boolean(true);
    }
     public void setNomClasse(String nomClasse) {
         this.nomClasse = nomClasse;
     }
     public void setEtiquette(String etiquette) {
         this.etiquette = etiquette;
     }
}
```

F.14.3 Implantation de OWLAjoutDuneProprieteEntreResourceCmdReceiver.java

```
package com.cotechnoe.noyau.command.concrete.receiver;

import java.util.logging.Logger;

import com.cotechnoe.noyau.OntologieRegister;
import com.cotechnoe.util.command.Receiver;
```

```
import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.model.OWLProperty;
import edu.stanford.smi.protegex.owl.model.RDFResource;
/* <p>Titre : OWLAjoutDuneProprieteEntreResourceCmdReceiver </p>
 * <p>Copyright : Copyright (c) 2008</p>
 * <p>Société : Cotechnoe inc.</p>
 * @author Michel Héon
 * @version 1.0
 */
public   class   OWLAjoutDuneProprieteEntreResourceCmdReceiver   implements
Receiver {
     private           static           Logger           logger           =
Logger.getLogger(OWLAjoutDuneProprieteEntreResourceCmdReceiver.class.getNam
e());
     private String loggerLabel =   "["+this.getClass().getSimpleName()+"]
";
     private String nomIndvOrigine;
     private String nomPropriete;
     private String nomIndvDestination;
     private JenaOWLModel owlModel;
     private OntologieRegister ontologieRegister;
     private RDFResource indvOrigine;
     private RDFResource indVDest;
     private OWLProperty propriete;
     public OWLAjoutDuneProprieteEntreResourceCmdReceiver() {
          logger.info(loggerLabel +"Création ");
          ontologieRegister = OntologieRegister.getInstance();
     }
     public Boolean action() {
          logger.info("["+this.getClass().getSimpleName()+"]Ajoute   de   la
propriété     ("     +nomIndvOrigine     +")     "+     nomPropriete     +"
("+nomIndvDestination+")");
          try {
          owlModel = ontologieRegister.getOwlDomainModel();
          indvOrigine = owlModel.getRDFResource(nomIndvOrigine);
          indVDest = owlModel.getRDFResource(nomIndvDestination);
          propriete = owlModel.getOWLProperty(nomPropriete);
          indvOrigine.addPropertyValue(propriete, indVDest);
          } catch (Exception e){
              e.printStackTrace();
              return new Boolean(false);
          }
          return new Boolean(true);
     }
     public String getNomIndvOrigine() {
          return nomIndvOrigine;
     }
     public void setNomIndvOrigine(String nomIndvOrigine) {
          this.nomIndvOrigine = nomIndvOrigine;
     }
     public String getNomPropriete() {
          return nomPropriete;
     }
     public void setNomPropriete(String nomPropriete) {
          this.nomPropriete = nomPropriete;
     }
     public String getNomIndvDestination() {
          return nomIndvDestination;
     }
     public void setNomIndvDestination(String nomIndvDestination) {
          this.nomIndvDestination = nomIndvDestination;
     }
}Implantation de OWLCreerUneInstanceSousCmdReceiver. java
package com.cotechnoe.noyau.command.concrete.receiver;

import java.util.logging.Logger;

import com.cotechnoe.noyau.OntologieRegister;
import com.cotechnoe.util.command.Receiver;

import edu.stanford.smi.protegex.owl.jena.JenaOWLModel;
import edu.stanford.smi.protegex.owl.model.OWLIndividual;
import edu.stanford.smi.protegex.owl.model.OWLNamedClass;

/* <p>Titre : OWLCreerUneInstanceSousCmdReceiver </p>
* <p>Description : </p>
* <p>Copyright : Copyright (c) 2008</p>
* <p>Société : Cotechnoe inc.</p>
* @author Michel Héon
* @version 1.0
*/
public class OWLCreerUneInstanceSousCmdReceiver implements Receiver {
    private           static           Logger           logger           =
Logger.getLogger(OWLCreerUneInstanceSousCmdReceiver.class.getName());
     private String loggerLabel =   "["+this.getClass().getSimpleName()+"]
";
    private JenaOWLModel owlModel;
    private OntologieRegister ontologieRegister;
     private String nomClasse;
     private String nomInstance;
    private String etiquette = "";
    public OWLCreerUneInstanceSousCmdReceiver() {
        ontologieRegister = OntologieRegister.getInstance();
    }
    public Boolean action() {
          owlModel = ontologieRegister.getOwlDomainModel();
        logger.info(loggerLabel +"Création de l'instance [" + nomInstance +
"] dans le classe ["+nomClasse+"]");
     OWLNamedClass superClasse = owlModel.getOWLNamedClass(nomClasse);
     try {
     OWLIndividual indv = superClasse.createOWLIndividual(nomInstance);
     indv.addLabel(etiquette, "");
     } catch (java.lang.NullPointerException e ){
           logger.severe(loggerLabel  + e.getMessage());
     } catch ( java.lang.IllegalArgumentException e){
           logger.info(loggerLabel +e.getMessage());
     }
       return new Boolean(true);
    }
     public String getNomClasse() {
         return nomClasse; }
     public void setNomClasse(String nomClasse) {
         this.nomClasse = nomClasse; }
     public String getNomInstance() {
         return nomInstance;         }
     public void setNomInstance(String nomInstance) {
         this.nomInstance = nomInstance;       }
     public void setEtiquette(String etiquette) {
         this.etiquette = etiquette;
     }
```

APPENDICE G

ÉVALUATION EXPÉRIMENTALE

Cet appendice présente les composants reliés à l'évaluation expérimentale soit; le protocole d'expérimentation, le certificat d'éthique et le résumé des réponses commentées de chacun des participants.

G.1 Protocole d'expérimentation

PROTOCOLE D'EXPÉRIMENTATION

Titre de la recherche : OntoCASE: Un outil informatique et méthodologique pour la formalisation d'un modèle semi-formel de connaissances sous la forme d'une ontologie

Identification du ou des membres de l'équipe de recherche :
CHERCHEUR PRINCIPAL : Michel Héon; doctorant; (514) 574-7320; michel.heon@licef.ca
DIRECTEUR DE LA THÈSE : Gilbert Paquette, Titulaire de la Chaire de recherche CICE
Directeur de recherches, Centres LICEF et CIRTA.; (514) 843-2015 poste 2818; gilbert.paquette@licef.ca
CODIRECTRICE DE LA THÈSE: Josianne Basque, Professeur, UER Éducation
Chercheure au Centre de recherche LICEF: (514) 843-2015 poste 2826, basque.josianne@teluq.uqam.ca

Les participants sont rencontrés individuellement dans un lieu et à un moment convenus avec chacun d'entre eux.

En première partie, le participant est initié à OntoCASE. Pour ce faire, une formalisation de quelques modèles sera réalisée par le participant, qui sera guidé par le chercheur principal. En deuxième partie, le participant est invité à réaliser des formalisations à partir de modèles de connaissances à partir d'un court texte qui lui sera fourni et sans être guidé. Dans cette partie, le chercheur principal aura un rôle de support et des questions pourront être posées au participant afin de permettre au chercheur de bien comprendre les difficultés que peut éprouver le participant lors de l'usage d'OntoCASE, ses intentions, son appréciation de certaines fonctionnalités, etc.
En dernière partie, un debriefing permettra de recueillir l'opinion du participant après l'usage d'OntoCASE.

Le but:
Le présent protocole d'expérimentation a pour but d'expliquer le déroulement d'une séance d'utilisation d'OntoCASE.

Expérimentation:

Partie 1 (phase préparatoire): aperçu des fonctionnalités du logiciel (20 minutes)
Thèmes abordés:

- Création d'un *espace de travail*
- Création d'un *projet*
- La *perspective* eLi pour l'édition d'un modèle semi-formel et TopBraid pour l'édition d'ontologie
- Concept de *vue*
- Serveur web pour la diffusion de l'ontologie de transformation
- Aspect général du plan de travail
- Modèle et diagramme MOT
- Édition d'un diagramme MOT
- Le tableau de bord à la formalisation
- Le tableau de bord à la validation

Partie 2 (phase de familiarisation): Les modèles semi-formels à formaliser en ontologie (30 minutes)
Cette étape de la séance a pour objectif d'initier le participant au processus de formalisation avec l'utilisation de l'assistant. Pendant cette étape, des opérations telles que la désambiguïsation et la validation seront expliquées. Un modèle semi-formel étalon (voir la figure 1) est fourni au participant. Certains points de modélisation sont portés à l'attention du participant et une instruction particulière sur la régulation de chacun de ces points lui est dispensée.

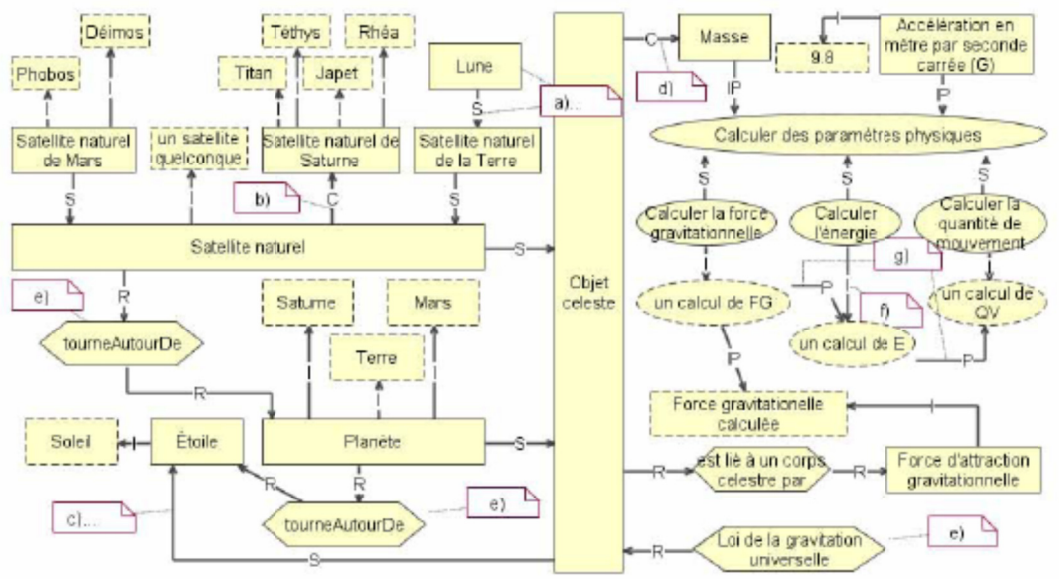


Figure 1: Modèle semi-formel étalon

La signification des points de modélisation mis en évidence par les étiquettes ont la signification suivante:

a) Utilisation inadéquate de la représentation d'un concept
b) Utilisation inadéquate d'un lien de composition
c) Conséquence de l'utilisation inversée d'un lien de spécialisation
d) Utilisation d'un lien C pour désigner un attribut
e) Particularité d'utilisation de la connaissance de type principe pour représenter une propriété ou une connaissance stratégique
f) Effet de l'utilisation d'un lien d'instanciation
g) Particularités du lien de précédence lors de l'interprétation d'un modèle.

Partie 3 (phase d'utilisation): Modélisation libre (45 minutes)
Après s'être initié à l'utilisation d'OntoCASE, on demande au participant de produire un modèle à partir d'un texte précédemment soumis (voir le texte *Élimination de déchets* de la figure 2). Après la modélisation du texte, le participant formalise et valide le modèle avec OntoCASE. Selon les résultats de la validation, plus particulièrement, la validation sémantique, le participant peuvent reprendre la modélisation afin que la sémantique de son modèle corresponde le plus possible au texte initial.

Élimination des déchets

L'élimination des déchets se fait de deux façons principales: l'incinération et l'enfouissement. L'incinération, qui est la méthode la plus onéreuse, consiste à brûler les déchets dans un four à des températures de 500 à 1000 degrés Celsius. La matière organique est alors transformée en gaz tandis que le reste des déchets devient un résidu (cendres). Cette technique permet d'éliminer entre 85 et 90 % du volume initial des déchets, mais les résidus doivent obligatoirement être éliminés dans un lieu d'enfouissement sanitaire.

Figure 2: Texte à soumettre pour modélisation

Partie 4 (phase de discussion): Discussions autour des points suivants et conclusion (25 minutes)

1. Que pensez-vous de :
   - de la méthodologie OntoCASE?
   - de l'assistant informatique OntoCASE?
2. L'utilisation d'OntoCASE accélère-t-elle la production d'une ontologie ?
3. À quel degré d'efficacité (de 1-Faible- à 4-Élevée-) OntoCASE permet-il la production d'ontologies ? et pourquoi?
   a) modèles sans ambigüité
   b) modèles contenant des ambigüités topographiques
   c) modèles contenant des ambigüités sémantiques
   d) modèles de connaissances conçus avec MotPlus
4. Les ontologies produites étaient-elles conformes à vos attentes?
5. Y a-t-il des fonctionnalités d'OntoCASE que :
   - vous n'avez pas comprises ou que vous estimez difficiles à comprendre?
   - qui vous semblent inutiles ou non pertinentes?
   - qui manquent?
6. Les interfaces d'OntoCASE rendent-elles conviviale son utilisation ?
7. Avez-vous d'autres commentaires à faire sur la méthodologie de transformation d'un modèle semi-formel en ontologie ou sur l'outil OntoCASE?

G.2 Certificat d'éthique

TÉLUQ
L'université à distance
de l'UQÀM

CERTIFICAT D'ÉTHIQUE

Le comité d'éthique de la recherche de la Téluq certifie avoir examiné la proposition de recherche soumise par

Michel Héon

intitulée

OntoCASE: Un outil informatique et méthodologique pour la formalisation d'un modèle semi-formel de connaissances sous la forme d'une ontologie

et avoir conclu que la recherche proposée est entièrement conforme aux normes d'éthique en recherche selon la *Politique d'éthique de la recherche avec les êtres humains*.

Valide jusqu'au 22 octobre 2010

Membres du comité

Do, Kim Liên	Cadre-conseil, Téluq	Recherche
Laferté, Sylvie	Professeure, Téluq	Marketing et management
Pichette, François	Professeur, Téluq	Linguistique
Sauvé, Louise	Professeure, Téluq	Technologie éducative

2009-10-22

Date

Sylvie Laferté
Présidente du comité d'éthique

Kim Liên Do
Cadre-conseil à la recherche

G.3 Réponse commentée des participants

Quatre participants ont participé à l'expérimentation en laboratoire. Chacune des réponses commentées qui suivent sont structurées selon le plan suivant

1 Mesure de l'efficience
- 1.1 La modélisation
- 1.2 La formalisation
- 1.3 La validation

2 Mesure de la satisfaction
- 2.1 Que pensez-vous de :
  - la méthodologie OntoCASE?
  - l'assistant informatique OntoCASE?
- 2.2 L'utilisation d'OntoCASE accélère-t-elle la production d'une ontologie ?
- 2.3 À quel degré d'efficacité (de 1-Faible- à 4-Élevée-) OntoCASE permet-il la production d'ontologies ? et pourquoi?
  - a) modèles sans ambigüité
  - b) modèles contenant des ambigüités topographiques
  - c) modèles contenant des ambigüités sémantiques
  - d) modèles de connaissances conçus avec MotPlus
- 2.4 Les ontologies produites étaient-elles conforme à vos attentes
- 2.5 Y a-t-il des fonctionnalités d'OntoCASE que :
  - -vous n'avez pas comprises ou que vous estimez difficiles à comprendre?
  - -qui vous semblent inutiles ou non pertinentes?
  - -qui manquent?
- 2.6 Les interfaces d'OntoCASE rendent-elles conviviales son utilisation?
- 2.7 Avez-vous d'autres commentaires à faire sur la méthodologie de transformation d'un modèle semi-formel en ontologie ou sur l'outil OntoCASE?

Figure G.1 : Structure de chaque réponse commentée.

G.3.1 Réponse commentée du participant 1 (d'après son verbatim)

Profil du participant no 1 :
- Domaine d'expertise : Expert chercheur en pédagogie
- Expertise en modélisation semi-formelle : expert
- Expertise en construction d'ontologie : aucune

1 Mesure de l'efficience

1.1 La modélisation

- « *La modification du texte dans les objets graphiques est difficile »*

- « *L'ergonomie associée à l'ajustement des objets est difficile à saisir »*

1.2 La formalisation
- En trois itérations, l'utilisateur est passé de 12 à 10 puis à 2 minutes pour formaliser un modèle semi-formel en ontologie

1.2.1 La Validation
- Aucune note particulière

2 Mesure de la satisfaction
2.1 Que pensez-vous de :
- la méthodologie OntoCASE?
- l'assistant informatique OntoCASE?
  - *« Couvre l'ensemble du processus de construction d'une ontologie (bien que je sois novice dans le domaine)*
  - *Le tableau de bord supporte bien la méthodologie*
  - *Un expert de contenu pourrait sans doute apprendre à utiliser OntoCASE, mais il y a beaucoup de menus et de fonctions de base… mais pour un expert de contenu qui connait MOT, il pourrait utiliser l'outil*
  - *Pour un modélisateur expert utilisateur du logiciel MOT, il y aurait une transition à faire*
  - *Après l'heure de formation que nous venons de faire, je ne pourrais pas me débrouiller tout seul. J'ai saisi les principales étapes, comment sauver, l'utilisation des menus déroulants. J'ai compris la mécanique générale, mais celle plus spécifique est difficile à retenir. »*

2.2 L'utilisation d'OntoCASE accélère-t-elle la production d'une ontologie ?
- *« Je suppose que oui…. »*

2.3 À quel degré d'efficacité (de 1-Faible- à 4-Élevée-) OntoCASE permet-il la production d'ontologies ? et pourquoi?
a) modèles sans ambigüité
b) modèles contenant des ambigüités topographiques
c) modèles contenant des ambigüités sémantiques
d) modèles de connaissances conçus avec MotPlus
- *« Il me semble que ça facilite ou que ça accélère. Ça aide quand il y a des ambiguïtés pour aller plus loin.»*

2.4 Les ontologies produites étaient-elles conformes à vos attentes?
- N/A

2.5 Y a-t-il des fonctionnalités d'OntoCASE que :
vous n'avez pas comprises ou que vous estimez difficiles à comprendre?
qui vous semblent inutiles ou non pertinentes?
Qui manquent?

- « *Fonctionnalité en tant qu'action à faire : c'est logique*
- *Fonctionnalité en tant que où toucher quel bouton pour faire quelque chose : c'est pas assez clair »*

2.6 Les interfaces d'OntoCASE rendent-elles conviviale son utilisation?

- « *Ça me parait complet*
- *Oui mais c'est conditionnel à une familiarisation*
- *Ça parait assez ergonomique pour quelqu'un qui sait déjà se débrouiller »*

2.7 Avez-vous d'autres commentaires à faire sur la méthodologie de transformation d'un modèle semi-formel en ontologie ou sur l'outil OntoCASE

- « *Très utile comme assistant à la modélisation semi-formel pour valider avec les experts… c'est certain. Notamment pour réaménager le modèle avant une entrevue avec les experts, pour cibler les questions; ça serait très utile*
- *En recherche sur la modélisation :*
    - *On peut se servir de cet outil pour désambigüiser le modèle sans ajouter plus de formalisme*
    - *On peut se servir de cet outil pour concevoir un guide de modélisation qui permet de fixer les questions à se poser lorsqu'on modélise*
    - *Y aurait-il des principes de modélisation qu'il faudrait ajouter compte tenu du fait que la modélisation semi-formelle sert à produire une ontologie?*
    - *Il faudrait être plus familier avec le métamodèle (l'ontologie cadre) car il fournit une structure de modélisation*
    - *Application pédagogique avec ce produit :*
        - *Construction d'un modèle et validation pour évaluation par un enseignant*
        - *Pour apprentissage de compréhension texte validation par l'étudiant lui-même qui permet de générer les questions et surtout de se poser les bonnes questions*
        - *La structuration logique pourrait être faite pour surtout insister sur certains types de relations dans certains domaines*
        - *Apprendre comprendre dans le texte. Ça apprend à poser les bonnes questions indépendamment du contenu*
        - *Texte -> apprendre -> comprendre -> faire une représentation -> représentation en modélisant en respectant les règles -> en validant… on intervient sur le contenu modélisé. L'utilisation des exemples est très intéressante par rapport au concept abstrait versus les concepts concrets. Il est connu en psychologie qu'il est plus facile de raisonner sur les concepts concrets que sur les concepts abstraits. Il s'agit là d'une piste intéressante à explorer avec cet outil.*
- *Aide au concepteur pédagogique.*

    - *On suit un texte didactique pour expliquer quelque chose pour voir toutes les ambiguïtés qu'il pourrait y avoir dans un texte…*
- *La validation permet de fixer la granularité de la modélisation »*

G.3.2 Réponse commentée du participant 2 (d'après son verbatim)

Profil de l'utilisateur no 2 :

Domaine d'expertise : Expert en gestion des ressources humaines
Expertise en modélisation semi-formelle : expert
Expertise en construction d'ontologie : débutant

1 Mesure de l'efficience

Remarque :

- L'activité de modélisation-formalisation-validation est d'une durée d'environ 15 minutes.

1.1 La modélisation

- eLi réagit correctement aux façons de faire de l'utilisateur, mais demande à celui-ci de réadapter certains principes d'utilisation qui sont intrinsèques à l'ergonomie du logiciel MOT, notamment la façon de modifier le nom de la connaissance et la création d'une nouvelle connaissance.

1.2 La formalisation

- La première formalisation est d'une durée de 8 minutes
- « *Les opérations sont convenablement encadrées par le tableau de bord à la validation »*

1.43 La validation

- « *L'utilisation de la validation est plus floue que celle de la validation »*

2 Mesure de la satisfaction

2.1 Que pensez-vous de :

- la méthodologie OntoCASE?
- l'assistant informatique OntoCASE?
    - « *Démarche logique et structurée*
    - *Pour un modèle simple, l'outil n'est pas très utile. Par contre, dans le cas de modèles complexes, l'outil fournit des analyses de validation qui sont intéressantes et sans nul doute utiles*
    - *La disposition à l'écran est assez conviviale et aidante »*

2.2 L'utilisation d'OntoCASE accélère-t-elle la production d'une ontologie ?

- « *Très certainement en OWL* ».

2.3 À quel degré d'efficacité (de 1-Faible- à 4-Élevée-) OntoCASE permet-il la production d'ontologies ? et pourquoi?
a) modèles sans ambigüité
b) modèles contenant des ambigüités topographiques
c) modèles contenant des ambigüités sémantiques
d) modèles de connaissances conçus avec MotPlus

- « *La relecture du texte de validation sémantique est difficile. Distinguer les énoncés qui servent à identifier les connaissances du domaine de celle qui identifient la structure de la connaissance* ».

2.4 Les ontologies produites étaient-elles conformes à vos attentes?
2.5 Y a-t-il des fonctionnalités d'OntoCASE que :
Vous n'avez pas comprises ou que vous estimez difficiles à comprendre?
Qui vous semblent inutiles ou non pertinentes?
Qui manquent?

- « *On se retrouvait dans les fonctionnalités »*

2.6 Les interfaces d'OntoCASE rendent-elles conviviale son utilisation?

- « *L'apprentissage est rapide, quelqu'un pourrait avec les explications de base arriver rapidement à utiliser le logiciel. Avec la séance d'aujourd'hui, je pourrais presque utiliser le logiciel de manière autonome. Moyennant quelques questions…*
- *Dans la restauration du modèle semi-formel, les éléments devraient être à la même place que ceux dans le modèle semi-formel d'origine.* »

2.7 Avez-vous d'autres commentaires à faire sur la méthodologie de transformation d'un modèle semi-formel en ontologie ou sur l'outil OntoCASE

- « *Non »*

G.3.3 Réponse commentée du participant 3 (d'après le verbatim)

Profil du participant no 3 :
Domaine d'expertise : Expert en pédagogie
Expertise en modélisation semi-formelle : expert
Expertise en construction d'ontologie : Avancé

1 Mesure de l'efficience
1.1 La modélisation

- *« Une adaptation est nécessaire*
- *L'édition de texte dans les connaissances n'est pas facile*
- *Beaucoup de la facilité d'édition du logiciel MOT ne se retrouve pas dans eLi*

- *Le choix automatique de lien comme dans MOTplus n'est pas disponible dans eLi*
- *Le logiciel a des difficultés a établir les proportions entre le texte écrit et la grandeur de l'élément graphique »*
- La modélisation a été d'une durée d'environ 10 minutes

1.2 La formalisation
- Le premier cycle complet de formalisation est d'une durée d'environ 10 minutes incluant l'étape de désambiguïsation
- La formalisation et validation se sont déroulées correctement

1.2.1 La validation
- Utilisation de la technique de modélisation par ajout d'un exemple dans le modèle
- La lecture minutieuse du rapport de validation sémantique se déroule bien
- Le texte de validation sémantique amène l'utilisateur à revoir certains points concernant la désambiguïsation de son modèle.

2 Mesure de la satisfaction
2.1 Que pensez-vous de :
- la méthodologie OntoCASE?
- l'assistant informatique OntoCASE?
- « *Pour une personne plus ou moins initiée, ça prendrait un guide.*
- *Assez intuitif pour une personne qui modélise avec les outils MOT moyennant probablement quelques questions.*
- *Ça prend un certain temps d'adaptation entre MOT et eLi*
- *Certaines manipulations de l'éditeur eLi seraient à optimiser ».*

2.2 L'utilisation d'OntoCASE accélère-t-elle la production d'une ontologie ?
- *« Oui, avec un outil comme celui-là, j'aurais sans doute sauvé beaucoup de temps dans la formalisation de modèles de connaissances que j'ai fait dans le passé.* »

2.3 À quel degré d'efficacité (de 1-Faible- à 4-Élevée-) OntoCASE permet-il la production d'ontologies ? et pourquoi?
a) modèles sans ambigüité
b) modèles contenant des ambigüités topographiques
c) modèles contenant des ambigüités sémantiques
d) modèles de connaissances conçus avec MotPlus
- *« On reconnait les ambiguïtés assez facilement »*

2.4 Les ontologies produites étaient-elles conformes à vos attentes?
- « *oui* »

2.5 Y a-t-il des fonctionnalités d'OntoCASE que :
vous n'avez pas comprises ou que vous estimez difficiles à comprendre?
Qui vous semblent inutiles ou non pertinentes?
Qui manquent?

- « *Il a toutes les fonctionnalités qu'il faut, j'ai assez touché à Protégé pour que je me sente à l'aise avec cet outil*. »

2.6 Les interfaces d'OntoCASE rendent-elles conviviales son utilisation?

- « *C'est bien, les tableaux de bord, il y a un petit côté automatique qui me plait.*
- *La liste des messages à la console est rébarbative. Faire ressortir les éléments importants.*
- *Le tableau de bord pourrait être plus compact pour laisser plus de place au modèle.*
- *La disposition de l'interface serait à travailler un peu pour la rendre plus efficace »*

2.7 Avez-vous d'autres commentaires à faire sur la méthodologie de transformation d'un modèle semi-formel en ontologie ou sur l'outil OntoCASE

- « *La partie validation me plait beaucoup*
- *Le texte de validation devrait être centré sur le domaine*
- *Pour l'expert de contenu, le texte produit est un bon moyen pour interpréter ce que l'ontologie dit. »*

G.3.4 Réponse commentée du participant 4 (d'après le verbatim)

Profil du participant no 4 :
Domaine d'expertise : Expert en représentation des connaissances
Expertise en modélisation semi-formel : Expert
Expertise en construction d'ontologie : Expert

1 Mesure de l'efficience

- Après 40 minutes le participant indique qu'il commence à s'habituer à l'utilisation du logiciel

1.1 La modélisation

- L'écriture des libellés dans les objets graphiques n'est pas facile
- L'apprentissage de l'éditeur de modèles progresse rapidement et le participant ??
- *« Ha!, les menus m'aident à modéliser, ils m'ont aidé à prendre une décision »*

1.2 La formalisation

- La première formalisation est réalisée après une trentaine de minutes de modélisation

- La réalisation de la première formalisation contenant des ambiguïtés a duré 10 minutes.
- « *Les menus aident à désambiguïser »*

1.3 La validation

- Un problème a été identifié dans la syntaxe du modèle et le participant a corrigé correctement son erreur
- Le participant s'amuse beaucoup.

2 Mesure de la satisfaction

2.1 Que pensez-vous de :

- la méthodologie OntoCASE?
- l'assistant informatique OntoCASE?
  - « *Le cadre méthodologique est complexe et complet, il peut prendre en compte un spectre assez large de type de modèles.*
  - *Ça reste à valider, mais le fait que l'ontologie de domaine soit structurée par une ontologie cadre permet de représenter des situations complexes.*
  - *La méthodologie est assez riche pour traiter des modèles complexes*
  - *Compte tenu que j'ai pu concevoir un modèle librement comme je le pensais, cela démontre que la méthodologie est valide*
  - *Je pense que l'outil valide la méthodologie*
  - *Concernant l'outil; les deux tableaux de bord (à la formalisation et à la validation) aident vraiment à comprendre l'outil et la méthodologie*
  - *C'est facile à utiliser et en même temps c'est facile à comprendre. L'apprentissage de l'outil et de la méthodologie est facilité par les tableaux de bord »*

2.2 L'utilisation d'OntoCASE accélère-t-elle la production d'une ontologie ?

- « *Je ne pourrais pas formaliser un modèle à la main sans cet outil, comment classifier telle ou telle chose en OWL… c'est trop complexe… en tout les cas, ce serait plus difficile à faire. En tant qu'expert en OWL, je peux y arriver, mais si je me mets à la place d'un concepteur pédagogique, ce ne serait pas possible. Cet outil n'est pas un aide, il est un outil qui permet de faire la formalisation*
- *Si j'étais un concepteur qui ne connaissait rien d'OWL, je ne pourrais pas concevoir une ontologie sans cet outil*
- *Cet outil me serait fort utile pour traduire en OWL des modèles faits par d'autres personnes.*
- *Bien sûr, il faut désambiguïser le modèle, mais même la désambiguïsation est guidée ce qui aide beaucoup. L'outil te donne une palette de réponses possibles et c'est à toi de choisir celle qui est la plus pertinente*
- *Pour faire la formalisation de plusieurs modèles, cet outil permet d'aller beaucoup plus vite*

- *L'utilisation des menus pour la désambiguïsation aide à comprendre le modèle fait par d'autres ou encore, si je suis le concepteur ça m'aide à bien dire ce que j'ai à dire.*
- *Avec sous la main la structure de l'ontologie cadre et le choix que me proposent les menus, l'outil me sert d'aide à l'apprentissage à la conception de modèles semi-formels. »*

2.3 À quel degré d'efficacité (de 1-Faible- à 4-Élevée-) OntoCASE permet-il la production d'ontologies ? Et pourquoi?
Modèles avec ou sans ambigüités

- « *Pour le modèle sans ambiguïté, l'outil est très efficace. Il ne fait pas que seulement transformer le modèle, il t'aide à faire une validation de la conceptualisation d'origine qui est à la base du modèle. Cet outil aide à la modélisation, il valide la cohérence du modèle. La rétroaction est une aide à la découverte des erreurs de modélisation.*
- *Pour les modèles ambigüs, le système m'assiste. J'ai pas à désambiguïser, je n'ai pas à me casser la tête à désambiguïser, je n'ai qu'à choisir ce que le système me propose. Pour moi c'est un système conseil. Au début je ne sais pas comment désambiguïser et lui il me dit comment. Tout ce qui manque, c'est un « help » pour chacune des propositions et dès lors je pourrais facilement choisir. »*

2.4 Les ontologies produites étaient-elles conformes à vos attentes?

- *Observation : L'utilisateur inspecte minutieusement l'ontologie produite (classes, individus, propriétés)*
- *La classification des propriétés c'est très bien.*
- *Il ne faudrait pas voir l'ontologie cadre… juste mon modèle à moi. »* Réponse du chercheur : « *Je devrais passer en OWL-Full et au lieu d'utiliser le subClasseOf entre les éléments de l'ODomaine et l'OCadre je pourrais utiliser un type. »*

2.5 Y a-t-il des fonctionnalités d'OntoCASE que
vous n'avez pas comprises ou que vous estimer difficiles à comprendre?
Qui vous semblent inutiles ou non pertinentes?
Qui manquent?

- « *Ce qui est difficile à comprendre, c'est l'interprétation dans l'ontologie. On ne sait pas si on est dans l'OCadre ou si on est dans ODomaine, ce qui ajoute de la complexité.*
- *Oui, l'OCadre permet de catégoriser les éléments de l'ontologie, mais ça nécessite un apprentissage supplémentaire. Mais c'est vrai, ça apporte une information de plus. Si on la connaît bien, elle apporte de la signification d'une propriété ou d'une classe. Mais c'est sûr ça nécessite un apprentissage, surtout de la structure de l'ontologie cadre. »*

2.6 Les interfaces d'OntoCASE rendent-elles conviviale son utilisation?

- « *Je n'ai pas eu de problèmes à utiliser le logiciel. Ça demande un apprentissage de l'outil, mais une fois que c'est su, c'est très facile à utiliser. L'apprentissage est aussi très facile*. »

2.7 Avez-vous d'autres commentaires à faire sur la méthodologie de transformation d'un modèle semi-formel en ontologie ou sur l'outil OntoCASE

- *« L'exemple que tu as choisi est le plus difficile (représentation de connaissances procédurales et stratégiques dans une ontologie) et la méthodologie fonctionne. Cela veut donc dire que la méthodologie peut traiter correctement beaucoup de types de cas et c'est très bien.*
- *Outils pour le Web sémantique*
- *Créer des ontologies sur les ontologies*
- *Un outil d'aide à la modélisation et d'assistance à la réflexion sur le domaine*
- *Un outil qui aide à trouver des erreurs sur les modèles déjà existants*
- *Un outil de validation de modèles semi-formels*
- *La rétro-génération du modèle semi-formel et la génération du texte à partir de l'ontologie est quelque chose de très innovateur*
- *Le texte généré est de type : sujet prédicat objet.*
    - *Le texte pourrait être trié selon l'un des trois champs,*
    - *Si un sujet et un objet sont identiques, ils pourraient être concaténés pour former une phrase plus complète du type : sujet→prédicat1→(objet/sujet)→prédicat2→(objet/sujet)→prédicat3 →objet*
- *Cela permet de construire des ontologies pour un concepteur de modèles qui est peu familier avec OWL*
- *Plusieurs niveaux de perspective :*
    - *Expert de contenu (permets une validation du contenu du domaine)*
    - *Utilité*
        - *Pour un concepteur qui ne connaît pas OWL*
        - *Pour un IO qui a à formaliser beaucoup de modèles SM*
        - *Pour valider des modèles semi-formels*
        - *Dans TELOS*
            - *Faire ontologie du domaine pour faire le cours*
            - *Récupération de modèle déjà existant*
            - *OWL pour le référencement sémantique*
        - *Dans l'environnement de systèmes de Web sémantique, OntoCASE est un outil de conception d'ontologies à partir d'une représentation semi-formelle*
        - *Scénario dans les cours*

- *Pour avoir des ontologies de domaines, on doit soit les faire nous-mêmes, soit les transformer de modèles SF déjà existants. Et c'est là qu'OntoCASE se positionne. »*

BIBLIOGRAPHIE

«Renamed Abox and Concept Expression Reasoner (RACER)». En ligne. <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>.

1995. «Dublin Core Metadata Initiative (DCMI)». En ligne. <http://www.dublincore.org/>.

1998. «Open Knowledge Base Connectivity (OKBC)». Stanford University. En ligne. <http://www.ksl.stanford.edu/software/OKBC/>. Consulté le 22-10-2009.

2007. «Loom Project Home Page». University of Southern California. En ligne. <http://www.isi.edu/isd/LOOM/>. Consulté le 22-10-2009.

2008a. «Jess: the Rule Engine for the Java Platform». Sandia National Laboratories. En ligne. <http://www.jessrules.com/jess/index.shtml>. Consulté le 27 mars.

2008b. «Ontolingua Home Page». Stanford University. En ligne. <http://www.ksl.stanford.edu/software/ontolingua/>. Consulté le 22-10-2009.

2009a. «Dubling Core - OWL». The National Center for Biomedical Ontology. En ligne. <http://protege.stanford.edu/plugins/owl/dc/protege-dc.owl>.

2009b. «Eclipse Graphical Modeling Framework (GMF)». The Eclipse Foundation. En ligne. <http://www.eclipse.org/modeling/gmf>.

2009c. «Eclipse Modeling Framework Project (EMF)». The Eclipse Foundation. En ligne. <http://www.eclipse.org/modeling/emf/>.

2009d. «GMF Tutorial». The Eclipse Foundation. En ligne. <http://wiki.eclipse.org/index.php/GMF_Tutorial>.

2009e. «NetBeans home page». Sun Microsystem. En ligne. <http://netbeans.org/>.

2009f. «Operational Conceptual Modelling Language (OCML)». En ligne. <http://technologies.kmi.open.ac.uk/ocml//>.

2009g. «Pellet: The Open Source OWL Reasoner». Clark & Parsia. En ligne. <http://clarkparsia.com/pellet>.

2009h. «protégé-owl api programmer's guide». Stanford Center for Biomedical Informatics Research En ligne.

<http://protege.stanford.edu/plugins/owl/api/guide.html>. Consulté le 25-10-2009.

2009i. «Protégé Programming Development Kit (PDK)». Stanford Center for Biomedical Informatics Research. En ligne. <http://protege.stanford.edu/doc/dev.html>. Consulté le 08/01/2009.

2009j. «using the protégé-owl reasoner api». Stanford Center for Biomedical Informatics Research En ligne. <http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html>. Consulté le 25-10-2009.

Alhir, Sinan Si. 2002. *Guide to applying the UML*. Coll. «Computer science». New York: Springer-Verlag.

Allemang, Dean, et Jim Hendler. 2008. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Coll. «Semantic Web / Web programming». Burlington MA: Morgan Kaufmann, 330 p.

An, Yuan, Alexander Borgida et John Mylopoulos. 2005. «Refining Semantic Mappings from Relational Tables to Ontologies». Dans *Semantic Web and Databases,* Springer. Berlin / Heidelberg.

ANSI/NISO-Z39. 2005. «Guidelines for the Construction, Format, and Management of Monolingual Thesauri ». National Information Standards Organization

Apostolou, D, G Mentzas, R Young et A Abecker. 2000 «Consolidating the Product Versus Process Approaches in Knowledge Management: The Know-Net Approach», *Practical application of knowledge management* (Manchester, 2000 Apr). Blackpool, Practical Application Company et 2000, 149-168 p.

Baader, Franz, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi et Peter F. Patel-Schneider. 2007. *The Description Logic Handbook*: Cambridge University Press.

Baader, Franz, Ian Horrocks et Ulrike Sattler. 2004. «Description Logics». Dans *Handbook on ontologies,* S. Staab et R. Studer, p. 1-28. Berlin Heidelberg New York: Springer-Verlag.

Basque, Josianne, Céline Desjardins, Béatrice Pudelko et Michel Léonard. 2008a. «Gérer les connaissances stratégiques dans des entreprises manufacturières de la Montérégie: expérimentation de la co-modélisation des connaissances dans 3 PME.». (November 2008): Rapport de recherche. Montréal: CEFRIO. En ligne.

<https://www.cefrio.qc.ca/upload/1599_rapportcefriovalotechfinal12nov08.pdf>.

Basque, Josianne, Clément Imbeault, Béatrice Pudelko et Michel Léonard. 2004 «Collaborative knowledge modeling between experts and novices: A strategy to support transfer of expertise in a organization», *Conference on Concept Mapping* (September 14-17). J.D. Novak A.J. Canas, F.M. Gonzalez Pamplona, pp. 75-81 p.

Basque, Josianne, Gilbert Paquette, Béatrice Pudelko et Michel Léonard. 2008b. «Collaborative Knowledge Modeling with a Graphical Knowledge Representation Tool: A Strategy to Support the Transfer of Expertise in Organizations». Dans *Knowledge Cartography. Mapping Techniques and Software Tools,* Alexandra Okada, Simon Buckingham Shum et Tony Sherborne, p. 357-382. London: Springer-Verlag.

Basque, Josianne, et Béatrice Pudelko. 2004 «The effect of collaborative knowledge modeling at a distance on performance and on learning», *First International Conference on Concept Mapping* (Pamplona, September 14-17). A. J. Canas, J. D. Novak et F. M. Gonzalez Universidad Publica de Navarra, 67-74 p. En ligne. <http://cmc.ihmc.us/papers/cmc2004-231.pdf >.

Basque, Josianne, et Béatrice Pudelko. 2008. «La co-modélisation des connaissances à l'aide d'un outil informatisé: Une stratégie de transfert d'expertise en milieu de travail - Rapport final des expérimentations menées à la Régie des Rentes du Québec.».

Basque, Josianne, et Béatrice Pudelko. 2010a. «Intersubjective Meaning-Making in Dyads Using Object-Typed Concept Mapping». Dans *In Handbook of Research on Collaborative Learning Using Concept Mapping,* P.L. Torres et R.C.V. Marriott: IGI Global.

Basque, Josianne, et Béatrice Pudelko. 2010b. «La comodélisation des connaissances par objets typés: Une stratégie pour favoriser le transfert d'expertise dans les organisations.». *Revue Télescope*. vol. 16, no Spécial: Le transfert intergénérationnel des connaissances, p. 111-129.

Basque, Josianne, et Béatrice Pudelko. 2010c. «Modeling for Learning». Dans *Visual Knowledge and Competency Modeling - From Informal Learning Models to Semantic Web Ontologies,* Gilbert Paquette. Hershey, New York: IGI Global.

Beckett, Dave, et Brian McBride. 2004. «RDF/XML Syntax Specification (Revised)». W3C. En ligne. <http://www.w3.org/TR/rdf-syntax-grammar/>. Consulté le 30 Mai.

Berardi, Daniela, Diego Calvanese et Giuseppe De Giacomo. 2005. «Reasoning on UML class diagrams». *Artif. Intell.* vol. 168, no 1, p. 70-118.

Berners-Lee, Tim. 1997. «Axioms of Web Architecture: Metadata». En ligne. <http://www.w3.org/DesignIssues/Metadata.html>.

Berners-Lee, Tim. 1998. «Notation 3». En ligne. <http://www.w3.org/DesignIssues/Notation3.html>.

Boley, Harold. «The Rule Markup Initiative». En ligne. <http://ruleml.org/>. Consulté le 29/12/2009.

Borst, Willem Nico. 1997. «Construction of engineering ontologies for knowledge sharing and reuse». Enschede, Université de Twente.

Brachman, Ronald J., et Hector J. Levesque. 2004. *Knowledge representation and reasoning*. San Francisco, CA: Morgan Kaufmann.

Brangier, Éric, et Javier Barcenilla. 2003. *Concevoir un produit facile à utiliser*. Paris: Editions d'Organisation.

Breitman, K.K., M.A.Casanova et W. Truszkowski. 2007. *Semantic Web: Concepts, Technologies and Applications*. Coll. «NASA Monographs in Systems and Software Engineering ». London: Springer.

Brockmans, Saartje, Peter Haase, Pascal Hitzler et Rudi Studer. 2006. «A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies». *Lecture Notes in Computer Science*. vol. 4011, p. 303-316.

Burnstein, Ilene. 2003. *Practical Software Testing: A process-Oriented Approach*. New York: Springer.

Buschmann, Frank, Kevlin Henney et Douglas C. Schmidt. 2007. *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*. Coll. «Software Design Pattern». West Sussex, England: Wiley.

Buzan, Tony, et Barry Buzan. 1994. *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*: E P Dutton.

Calero, Coral, Francisco Ruiz et Mario Piattini. 2006. *Ontologies for Software Engineering and Software Technology*. Coll. «Computer Science». Berlin Heidelberg: Springer. En ligne. <http://dx.doi.org/10.1007/3-540-34518-3_1>.

Castro, Alexander, Philippe Rocca-Serra, Robert Stevens, Chris Taylor, Karim Nashar, Mark Ragan et Susanna-Assunta Sansone. 2006. «The use of concept

maps during knowledge elicitation in ontology development processes - the nutrigenomics use case». *BMC Bioinformatics*. vol. 7, no 1, p. 267. En ligne. <http://www.biomedcentral.com/1471-2105/7/267>.

Chein, Michel, et Marie-Laure Mugnier. 2008. *Graph-based Knowledge Representation* Coll. «Advanced Information and Knowledge Processing». London: Springer.

Chen, Peter Pin-shan. 1976. «The Entity-Relationship Model: Toward a Unified View of Data». *ACM Transactions on Database Systems*. vol. 1, p. 9-36.

Chen, Peter Pin-shan. 2002. «Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned». Dans *Software Pioneers: Contributions to Software Engineering,* p. 297-310: Springer.

Clayberg, Eric, et Dan Ribel. 2006. *Eclipse Building Commercial-Quality Plug-ins*. Coll. «The Eclipse series». Boston: Addison Wesley.

CmapTools. «IHMC CmapTools knowledge modeling kit». En ligne. <http://cmap.ihmc.us/conceptmap.html>.

Comité de travail sur l'intégration des jeunes à la fonction publique québécoise. 2001. «Rapport du comité de travail sur l'intégration des jeunes à la fonction publique québécoise, D'ici 10 ans, 21 000 nouveaux visages, au ministère d'État à l'Administration et à la fonction publique»

Cycorp Inc. 1995. «Formalized Common Knowledge». En ligne. <http://www.opencyc.org/>.

Cycorp Inc. 2002. «What does Cyc know?». En ligne. <http://www.cyc.com/cyc/cyc/technology/whatiscyc_dir/whatdoescycknow>. Consulté le 26/12/2009.

Daconta, Micheal, Leo J Obrst et Kevin T Smith. 2003. *The Semantic Web*: Wiley.

Daum, Berthold. 2004. *Eclipse 3 pour les développeurs Java*. Paris: Dunod.

Davies, John, Deiter Fensel et Frank Van Harmelen. 2003. *Towards The Semantic Web : Ontology-Driven Knowledge Management*: John Wiley & Sons. En ligne. <http://www.ontoknowledge.org/index.shtml>.

Dietz, Jan L.G. 2006. *Entreprise Ontology: Theory and Methodology*. Coll. «Computer Science». Berlin Heidelberg: Springer-Verlag.

Dirk, Riehle. 2008. «JUnit 3.8 documented using collaborations». *SIGSOFT Softw. Eng. Notes*. vol. 33, no 2, p. 1-28.

Đurić, Dragan. 2004. «MDA-based Ontology Infrastructure». *Computer Science and Information Systems*. vol. 01, no 01.

Đurić, Dragan, Dragan Gašević et V. Devedžić. 2005a «Adventures in Modeling Spaces: Close Encounters of the Semantic Web and MDA Kinds», *4th International Semantic Web Conference* (Galway, Ireland). Workshop on Semantic Web Enabled Software Engineering.

Đurić, Dragan, Dragan Gasevic et Vladan Devedzic. 2005b. «Ontology Modeling and MDA». *Journal of Object Technology*. vol. 4, no 1. En ligne. <http://www.jot.fm/issues/issue_2005_01/article3>.

Eclipse foundation. 2010. «Eclipse Modeling Project». En ligne. <http://www.eclipse.org/modeling/>.

Ermine, Jean-Louis, et Nada Matta. 2003. *Initiation à la méthode MASK*: Université de Technologie Troyes.

Eskridge, Thomas, Pat Hayes et Robert Hoffman. 2006 «Formalizing The Informal: A Confluence Of Concept Mapping And The Semantic Web», *Second Int. Conference on Concept Mapping* (San José, Costa Rica). A. J. Cañas et J. D. Novak.

Faucher, Cyril, Frédéric Bertrand et Jean-Yves Lafaye. 2008. «Génération d'ontologie à partir d'un modèle métier UML annoté». Dans *Modélisation des connaissances,* Henri Birand et Stéphanie Loiseau*,* p. 65-84. Vauquelin, France: Cépadues.

Fuggetta, Alfonso. 1993. «A Classification of CASE Technology». *Computer*. vol. 26, no 12, p. 25-38.

Gamma, Erich, et Kent Beck. 2003. *Contributing to Eclipse: Principles, Patterns, and Plug-Ins* Coll. «Eclipse series»: Addison-Wesley 416 p.

Gamma, Erich, Richard Helm, Ralph Johnson et John Vlissides. 1999. *Catalogue de modèles de conception réutilisables*. Jean-Marie Lasvergères: Vuibert.

Gangemi, Aldo, Stefano David, Aguado de Cea, Mari Carmen Suárez-Figueroa, Elena Montiel- Ponsoda et María Poveda. 2008. «A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies». NeOn: Lifecycle Support for Networked Ontologies: 183 p

Gašević, Dragan, Dragan Djurić et Vladan Devedžić. 2006a. «MDA-based Automatic OWL Ontology Development». *International Journal on Software Tools for Technology Transfer (STTT)*. vol. 9, no 2, p. 103-117.

Gašević, Dragan, Dragan Djurić et Vladan Devedžić. 2006b. «The Model Driven Architecture (MDA)». Dans *Model Driven Architecture and Ontology Development,* p. 109-126. New York: Springer-Verlag. En ligne. <http://dx.doi.org/10.1007/3-540-32182-9_4>.

Gašević, Dragan, Dragan Djurić et Vladan Devedžić. 2006c. *Model Driven Architecture and Ontology Development*. New York: Springer-Verlag. En ligne. <http://dx.doi.org/10.1007/3-540-32182-9_4>.

Gašević, Dragan, Dragan Djurić et Vladan Devedžić. 2006d. *Model Driven Architecture and Ontology Development*. New York, Inc.: Springer-Verlag. En ligne. <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/3540321802>.

Gašević, Dragan, Dragan Djurić et Vladan Devedžić. 2006e. «Modeling Spaces». Dans *Model Driven Architecture and Ontology Development,* p. 127-141. En ligne. <http://dx.doi.org/10.1007/3-540-32182-9_5>.

Gerbé, Olivier, Karine Madeleine et Jean Talbot. 2003 «Une méthode pour développer un système de gestion de connaissances en 5 jours», *16th International Conference on Software & Systems Engineering and their Applications* (Paris, France).

Gerbé, Olivier, et Guy Mineau. 2008. «Métamodélisation pour le Web». Dans *Modélisation des connaissances,* Henri Birand et Stéphanie Loiseau, p. 85-104. Vauquelin, France: Cépadues.

Golbreich, Christine. 2004. «Combining Rule and Ontology Reasoners for the Semantic Web». Dans *Rules and Rule Markup Languages for the Semantic Web,* p. 6-22. En ligne. <http://www.springerlink.com/content/mpatx2npbg7yrvw3>.

Golbreich, Christine, et Atsutoshi Imai. 2004. «Combining SWRL rules and OWL ontologies with Protégé OWL Plugin, Jess, and Racer». Laboratoire d'Informatique Médicale, Faculté de Médecine, Université Rennes 1. En ligne. <http://protege.stanford.edu/conference/2004/abstracts/Golbreich.pdf>. Consulté le 29 mai.

Gómez-Pérez, Asunción. 2004. «Ontology Evaluation». Dans *Hanbook on Ontologies,* p. 251-274: Springer.

Gómez-Pérez, Asunción, Mariano Fernández-López et Oscar Corcho. 2003. *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*, First edition. New York: Springer.

Gómez-Pérez, Asunción, Mariano Fernández et Antonio J. de Vincente. 1996 «Towards a Method to Conceptualize Domain Ontologies», *Workshop on Ontological Enginnering, ECIA'96* (Budapest, Hungary). Van der Vet P, 41-52 p.

Gronback, Richard C. 2008. *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. Coll. «Eclipse Series». Boston: Addison-Wesley Professional.

Grosof, Benjamin N, Ian Horrocks, Raphael Volz et Stefan Decker. 2003. «Description Logic Programs: Combining Logic Programs with Description LogicCambridge». Dans *WWW2003*. En ligne. <http://www2003.org/cdrom/papers/refereed/p117/p117-grosof.html>.

Gruber, Thomas. 1993a. «Toward Principles for the Design of Ontologies Used for Knowledge Sharing». *Formal Ontology in Conceptual Analysis and Knowledge Representation*, p. 23.

Gruber, Thomas R. 1993b. «A translation approach to portable ontology specification». *Knowledge Acquisition*. vol. 5, no 2, p. 199-220

Grüninger, Michael, et Mark S. Fox. 1995. «Methodology for the design and evaluation of ontologies». Dans *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing* (13 avril).

Guarino, Nicola. 1998 «Formal Ontology and Information Systems», *1st International Conference on Formal Ontologies in Information Systems* (Trento, Italy). IOS Press, p. 3-15 p.

Hart, Anna. 1986. *Knowledge acquisition for expert systems*. New York: McGraw-Hill.

Helbig, Hermann. 2006. *Knowledge Representation and the Semantics of Natural Language*. Coll. «Cognitive Technologies». New York: Springer.

Héon, Michel. 1998. «Application des réseaux de neurones dans le développement d'algorithmes de restauration d'images appliqués à la tomographie par émission de positrons». Mémoire de maîtrise, Sherbrooke, Département des sciences appliquées, Université de Sherbrooke. En ligne. <http://www.cotechnoe.com/pdf/memoire_doc_maitre.pdf>.

Héon, Michel, et Gilbert Paquette. 2010. «From semi-formal Models to Formal Models». Dans *Visual Knowledge and Competency Modeling - From Informal Learning Models to Semantic Web Ontologies,* Gilbert Paquette. Hershey, New York: IGI Global.

Héon, Michel, Hesham Tolba et Douglas O'Shaughnessy. 1998. «Robust Automatic Speech Recognition by the Application of a Temporal-Correlation-Based Recurrent Multilayer Neural Network to the Mel-Based Cepstral Coefficients». Dans *5th International Conference on Spoken Language Processing* (November 30 - December 4, 1998).

Hepp, Martin, Pieter De Leenheer, Aldo de Moor et York Sure. 2008. *Ontology Management: Semantic Web, Semantic Web Services, and Business Applications*. Coll. «Semantic Web and Beyond», no Semantic Web And Beyond Computing for Human Experience: Springer.

Horridge, Matthew, Nick Drummond, John Goodwin, Alan Rector, Robert Stevens et Hai H Wang. 2006 «The Manchester OWL Syntax», *OWL Experiences and DirectionsWorkshop (OWLED'06) at the ISWC'06*. En ligne. <http://www.webont.org/owled/2006/acceptedLong/submission_9.pdf>.

Horrocks, Ian. 2003. «The FaCT System». En ligne. <The FaCT System>.

Horrocks, Ian. 2005. «OWL: A Description Logic Based Ontology Language». Dans *Logic Programming,* Springer, p. 1-4. Berlin / Heidelberg.

Horrocks, Ian. 2007. «FaCT++». University of Manchester. En ligne. <http://owl.man.ac.uk/factplusplus/>.

Horrocks, Ian, Harold Boley, Said Tabet, Benjamin Grosof et Mike Dean. 2004a. «SWRL: A Semantic Web Rule Language Combining OWL and RuleML». W3C. En ligne. <http://www.w3.org/Submission/SWRL/>. Consulté le 29 mai.

Horrocks, Ian, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof et Mike Dean. 2004b. «SWRL Section 8. Built-Ins». DARPA DAML Program. En ligne. <http://www.daml.org/rules/proposal/builtins.html>.

Hussey, Kenn. 2008. «MDT EODM Termination Review». En ligne. <http://www.eclipse.org/project-slides/EODM%20Termination%20Review.pdf>.

Hyerle, David. 2008. «Thinking Maps®: A Visual Language for Learning». Dans *Knowledge Cartography,* p. 73-88: Springer. En ligne. <http://dx.doi.org/10.1007/978-1-84800-149-7_4>.

IBM. 2007. «IBM Integrated Ontology Development Toolkit: An ontology toolkit for storage, manipulation, query, and inference of ontologies and corresponding instances.». En ligne. <http://www.alphaworks.ibm.com/tech/semanticstk>. Consulté le 19-01-2011.

IEEE. 1990. «IEEE Standard Glossary of Software Engineering Terminology». IEEE Computer Society. New York, IEEE Std 610.121990

--------. 1996. «IEEE Standard for Developing Software Life Cycle Processes ». IEEE Computer Society. New York, IEEE Std 1074-1995

IEEE Standard Upper Ontology Working Group. 2000. «Suggested Upper Merged Ontology (SUMO)». En ligne. <http://www.ontologyportal.org/>.

IHMC. «IHMC CmapTools: Concept Mapping Home Site». Institute for Human and Machine Cognition. En ligne. <http://cmap.ihmc.us/>.

ISO-14977. 1996. «The standard metalanguage Extended BNF». En ligne. <http://www.cl.cam.ac.uk/~mgk25/iso-14977.pdf>.

Jacobson, Ivar, Grady Booch et James Rumbaugh. 1999. *The Unified Software Development Process* Addison-Wesley, 463 pages p.

Jarrosson, Bruno. 1992. *Invitation à la philosophie des sciences*, Édition du seuil. Coll. «Points»: Inédits Sciences.

Johnson-Laird, P.N. 2004. «The history of Mental Model». Dans *Psychology and Reasoning: Theorical and Historical,* K. Manktelow et M.C. Chum, p. 179-212. New York: Psychology Press.

Jonassen, D H, et S Grabinger. 1991. «Instructional Design and Development Adviser: Intelligent Job Aid, Help System, and Intelligent Authoring System.». Denver, Division of Instructional Technology, University on Colorado

JUnit.org. «Resources for Test Driven Development». En ligne. <http://www.junit.org/about>. Consulté le 2009-09-18.

KACTUS. 1996. «The KACTUS Booklet version 1.0. Esprit Project 8145 KACTUS». En ligne. <http://www.swi.psy.uva.nl/projects/NewKACTUS/Reports.html>.

Kadima, Hubert. 2005. *MDA: Conception orientée objet guidée par les modèles*. Coll. «Études & Développement». Paris: Dumond.

Kendal, S.L., et M. Creen. 2007. *An Introduction to Knowledge Engineering*. London: Springer-Verlag.

Kinshuk, Heimo H. Adelsberger, Jan Martin Pawlowski et Demetrios Sampson. 2008. *Handbook for Education and Training on Information Technologies*, Second edition. Coll. «International Handbooks on Information Systems». Heidelberg: Springer-Verlag.

Lacy, Lee W. 2005. *OWL: Representing Information Using the Web Ontology Language*. Victoria, BC, Canada: Trafford.

Lloyd, J. W. 1987. *Foundations of logic programming; (2nd extended ed.)*: Springer-Verlag New York, Inc.

Ma, Li, GuoTong Xie, Yang Yang et Lei Zhang. 2004. «IBM Integrated Ontology Development Toolkit». En ligne. <http://www.alphaworks.ibm.com/tech/semanticstk>.

Mariot, Pierre, Christine Golbreich, JPierre Cotton, François Vexles et Alain Berger. 2008. «Méthode, Modèle et outils Ardams de capitalisation des connaissances». Dans *Modélisation des connaissances,* Cépaduès. Toulouse, France.

Martin, Philippe. 2002. «Knowledge Representation/Translation in RDF+OWL, N3, KIF, UML and the WebKB-2 languages (For-Links, Frame-CG, Formalized English)». En ligne. <http://www.webkb.org/doc/model/comparisons.html>. Consulté le 25-nov.

Matuszek, Cynthia, John Cabral, Michael Witbrock et John Deoliveira. 2006 «An introduction to the syntax and content of Cyc», *2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering* (Stanford, CA, USA). 44-49 p.

McAffer, Jeff, et Jean-Michel Lemieux. 2006. *eclipse Rich Client Platform*. Coll. «Eclipse Series». Boston: Addison-Wesley Professional.

McGuinness, Deborah. 2003. «Ontologies Come of Age». Dans *The Semantic Web: Why, What, and How,* Dieter: MIT Press. En ligne. <http://www.ksl.stanford.edu/people/dlm/papers/ontologies-come-of-age-mit-press-%28with-citation%29.htm>.

McGuinness, Deborah L, et Frank van Harmelen. 2004. «OWL Web Ontology Language Overview ». W3C. En ligne. <http://www.w3.org/TR/owl-features/>. Consulté le 30 mai.

Mei, J., et Paslaru. 2005 «Reasoning paradigms for SWRL-enabled ontologies», *Proceedings of International Workshop on Protege with Rules* (Madrid, Spain). En ligne. <citeulike-article-id:3908631>.

Meštrović, Ana, et Mirko Čubrilo. 2009. «F-Logic Data and Knowledge Reasoning in the Semantic Web Context». Dans *Intelligent Engineering Systems and Computational Cybernetics,* p. 119-136. En ligne. <http://dx.doi.org/10.1007/978-1-4020-8678-6_11>.

Minsky, Marvin. 1985. «A Framework for Representing Knowledge». Dans *Reading in Knowledge representation,* Ronald J. Brachman et Hector J. Levesque, p. 245-262. San Mateo, California: Morgan Kaufmann.

Motta, Enrico. 2006. «NeOn Project». NeOn Project. En ligne. <http://www.neon-project.org/web-content/>.

NeOn project. 2008. «Neon toolkit». En ligne. <http://neon-toolkit.org/wiki/Main_Page>.

Niles, I., et A. Pease. 2001 «Towards a Standard Upper Ontology», *In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)* (Ogunquit, Maine, 17-19 Octobre). Chris Welty et Barry Smith.

Novak, Joseph D, et Alberto J Cañas. 2006. «The origins of the concept mapping tool and the continuing evolution of the tool». *Information Visualization*. vol. 5, p. 175–184.

O'Connor, M. J., C. I. Nyulas, R. D. Shankar, A. K. Das et M. A. Musen. 2008a «The SWRLAPI: A Development Environment for Working with SWRL Rules», *OWL: Experiences and Directions (OWLED), Fifth International Workshop, held with 7th International Semantic Web Conference, Karlsruhe, Germany*. En ligne. <citeulike-article-id:4214309 http://bmir.stanford.edu/file_asset/index.php/1428/BMIR-2008-1336.pdf>.

O'Connor, M. J., R. D. Shankar, S. W. Tu, C. I. Nyulas et A. K. Das. 2008b «Developing a Web-Based Application using OWL and SWRL», *AAAI Spring Symposium* (Stanford, CA). En ligne. <citeulike-article-id:3946260 http://www.aaai.org/Papers/Symposia/Spring/2008/SS-08-01/SS08-01-012.pdf>.

O'Connor, Martin. 2009a. «SWRL Factory FAQ». Stanford Center for Biomedical Informatics Research. En ligne. <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLFactoryFAQ>. Consulté le 25-10-2009.

O'Connor, Martin. 2009b. «SWRLBuilt In Bridge». Stanford Center for Biomedical Informatics Research. En ligne. <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLBuiltInBridge>. Consulté le 2009-01-09.

O'Connor, Martin. 2009c. «SWRLTab». Stanford Center for Biomedical Informatics Research. En ligne. <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>. Consulté le 25-10-2009.

O'Connor, Martin, Holger Knublauch, Samson Tu, Benjamin Grosof, Mike Dean, William Grosso et Mark Musen. 2005a «Supporting Rule System Interoperability on the Semantic Web with SWRL», *Fourth International Semantic Web Conference (ISWC 2005)* (Galway, Ireland). Stanford Medical Informatics, Stanford University School of Medicine. En ligne. <http://smi-web.stanford.edu/auslese/smi-web/reports/SMI-2005-1080.pdf>.

O'Connor, Martin, Holger Knublauch, Samson Tu et Mark Musen. 2005b. «Writing Rules for the Semantic Web Using SWRL and Jess». Dans *8th International Protege Conference, Protege with Rules Workshop*: Stanford Medical Informatics, Stanford University School of Medicine. En ligne. <http://www.med.univ-rennes1.fr/~cgolb/Protege2005/SWRLJessOConnor.pdf>.

Oberle, Daniel. 2006. *Semantic Management of Middleware* Coll. «Semantic Web And Beyond Computing for Human Experience»: Springer US.

ODP.org. 2008. «Ontology Design Pattern Types (OPTypes)». NeOn project. En ligne. <http://ontologydesignpatterns.org/wiki/OPTypes>. Consulté le 26/12/2009.

Olivé, Antoni. 2007. *Conceptual Modeling of Information Systems*: Springer-Verlag New York, Inc., 455 p.

OMG BPDM. 2007. «Business Process Definition Metamodel (BPDM)», Object Management Group

OMG BPMN. 2009. «Object Management Group/Business Process Management Initiative». Object Management Group, Inc. En ligne. <http://www.bpmn.org/>.

OMG MDA. 2007. «OMG Model Driven Architecture». Object Management Group,,. En ligne. <http://www.omg.org/mda/>. Consulté le 30/09/2008.

OMG MOF-QVT. 2008. «Documents associated with Meta Object Facility (MOF) 2.0 Query/View/Transformation, v1.0 ». OMG. En ligne. <http://www.omg.org/spec/QVT/1.0/>.

OMG MOF. 2006. «OMG's MetaObject Facility». Object Management Group,,. En ligne. <http://www.omg.org/mof>. Consulté le 30/09/2008.

OMG OCL. 2006. «Object Constraint Language Specification, version 2.0». En ligne. <http://www.omg.org/technology/documents/formal/ocl.htm>.

OMG ODM. 2007. «Ontology Definition Metamodel: OMG Adopted Specification». Object Management Group,,. En ligne. <http://www.omg.org/spec/ODM/1.0/Beta2/PDF/>. Consulté le 26/05/2008.

OMG UML. 1997-2006. «UML® Resource Page». The Object Management Group (OMG). En ligne. <http://www.uml.org/>.

Paquet, Marcel. 2006. *Magritte: La pensée visible*: Taschen.

Paquette, G., F. Pachet et S. Giroux. 1994. «Épitalk, un outil générique pour la construction de systèmes conseillers». *Sciences et techniques éducatives*.

Paquette, Gilbert. 2002a. *L'ingénierie pédagogique*. Sainte-Foy (Québec): Presses de l'Université du Québec.

Paquette, Gilbert. 2002b. *Modélisation des connaissances et des compétences : un langage graphique pour concevoir et apprendre*. Sainte-Foy: Presses de l'UQ.

Paquette, Gilbert. 2010. *Visual Knowledge and Competency Modeling - From Informal Learning Models to Semantic Web Ontologies*. Hershey, PA: IGI Global.

Paquette, Gilbert, Michel Léonard et Karin Lundgren-Cayrol. 2007. «The MOT+ Visual Language For Knowledge-Based Instructional Design». Télé-université. Montréal, LICEF-CIRTA Research Center and CICE Research Chair: 24 p

Paquette, Gilbert, et Delia Rogozan. 2006. «Correspondance avec le langage graphique MOT-OWL et le langage des prédicats du premier ordre». Montréal, LICEF: 17 p

Paquette, Gilbert, et Lucien Roy. 1990. *Systèmes à base de connaissances*, Université du Québec à Montréal. Montréal: Télé-Université: Beauchemin.

Paris, S., M. Y. Lipson et K. K Wixson. 1983. «Becoming a Strategic Reader». *Contemporary Educational Psychology*. vol. 8, p. 293-316.

Park, Jack, et Sam Hunting. 2003. *XML Topic Maps: Creating and Using Topic Maps for the Web*: Addison Wesley, 605 p.

Patel-Schneider, Peter F, Patrick Hayes et Ian Horrocks. 2004 «OWL Web Ontology Language Semantics and Abstract Syntax ». W3C. En ligne. <http://www.w3.org/TR/owl-semantics/>. Consulté le 29/05/2006.

Pitrat, Jacques. 1990. *Métaconnaissance, futur de l'intelligence artificielle* Trad. de: *Français*, Hermès. Paris.

Pitrat, Jacques. 1993. *Penser autrement l'informatique*, Hermes. Paris.

Prax, Jean-Yves. 2003. *Le manuel du knowledge management* Trad. de: *Français*. Paris: Dumod.

Protégé Home Site. 2009. «Welcome to protégé». Stanford Center for Biomedical Informatics Research En ligne. <http://protege.stanford.edu/>. Consulté le 2009-10-13.

Reitsma, Jaap, David Ing et P. Youmm. 2008. «GMF Tutorial». The Eclipse Foundation. En ligne. <http://wiki.eclipse.org/index.php/GMF_Tutorial>. Consulté le 23/11/2009.

Rhem, Anthony J. 2006. *UML For Developing Knowledge Management Systems*. New York: Auerbach, 269 p.

Rumbaugh, James, Ivar Jackson et Grady Booch. 1999. *The Unified Modeling Language Reference Manual*. Coll. «Object Technology».

SC34/WG3 2008. «Home page of the Topic Maps ISO standards». En ligne. <http://www.isotopicmaps.org/>. Consulté le 2009-09-09.

Sowa, John F. 1976. «Conceptual Graphs for a Data Base Interface». *IBM Journal of Research and Development*. vol. 20, no 4, p. 336–357.

Sowa, John F. 2000a «Ontology, Metadata, and Semiotics», *ICCS'2000* (Darmstadt, Germany, August 14, 2000.). Ganter & G. W. Mineau Springer-Verlag, 55-81 p.

Sowa, John F. 2000b. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Pacific Grove, CA: Brooks Cole Publishing Co., 594 p.

Sowa, John F. 2003. «KR Ontology». En ligne. <http://www.jfsowa.com/ontology/>.

Staab, Steffen, Rudi Studer, Hans-Peter Schnurr et York Sure. 2001. «Knowledge Processes and Ontologies». *IEEE Intelligent Systems*. vol. 16, no 1, p. 26-34.

Steinberg, Dave, Frank Budinsky, Marcelo Paternostro et Ed Merks. 2008. *EMF: Eclipse Modeling Framework, Second EditionEMF: Eclipse Modeling Framework*, Second edition. Coll. «Eclipse series»: Addison Wesley Professional.

Tarski, Alfred. 1944. «The Semantic Conception of Truth: and the Foundations of Semantics». *Philosophy and Phenomenological Research*. vol. 4, no 3, p. 341-376. En ligne. <http://www.jstor.org/stable/2102968>.

The Eclipse Foundation. 2009. «The Eclipse home page». En ligne. <http://www.eclipse.org/>.

ThinkBuzan.com. «iMindMap software». En ligne. <http://www.thinkbuzan.com/uk>.

TopQuadrant. 2009. «TopBraid Composer (TM)». TopQuadrant. En ligne. <http://www.topquadrant.com/products/TB_Composer.html>. Consulté le 27/08/2009.

Uschold, Mike, et Micheal Gruninger. 1996. «Ontologies: Principles, Methods and Applications». *Knowledge Engineering Review*. vol. 11, no 2, p. 69.

W3C OWL. 2004. «Ontology Web Language». World Wide Web Consortium,,. En ligne. <http://www.w3.org/2004/OWL/>. Consulté le 30/09/2008.

W3C PICS. «Platform for Internet Content Selection (PICS)». En ligne. <http://www.w3.org/PICS/>.

Wenger, Etienne. 1987. *Artificial intelligence and Tutoring Systems, Communication and cognitive Approches to the Communication of knowledge*. Los Altos: Morgan Kaufmann.

Weske, Mathias. 2007. *Business Process Management*. Berlin: Springer-Verlag.

Yang, Yang. 2006. «EMF Ontology Definition Metamodel». IBM. En ligne. <http://www.eclipse.org/modeling/mdt/eodm/docs/articles/EODM_Documentation/>.