Donner une autre vie à vos besoins fonctionnels : une approche dirigée par l'entreposage et l'analyse en ligne

Zouhir Djilani

▶ **To cite this version:**

Zouhir Djilani. Donner une autre vie à vos besoins fonctionnels : une approche dirigée par l'entreposage et l'analyse en ligne. Autre [cs.OH]. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers, 2017. Français. ⟨NNT : 2017ESMA0012⟩. ⟨tel-01591845⟩

HAL Id: tel-01591845

https://theses.hal.science/tel-01591845v1

Submitted on 22 Sep 2017

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

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

HAL Authorization

Ecole Nationale Supérieure de Mécanique et d'Aérotechnique

THESE

pour l'obtention du Grade de

DOCTEUR DE L'ÉCOLE NATIONALE SUPÉRIEURE DE MÉCANIQUE ET D'AEROTECHNIQUE

(Diplôme National — Arrêté du 25 mai 2016)

Ecole Doctorale : Sciences et Ingénierie pour l'Information, Mathématiques
Secteur de Recherche : INFORMATIQUE ET APPLICATIONS

Présentée par :

Zouhir DJILANI

****

Donner une autre vie à vos besoins fonctionnels : une approche dirigée par l'entreposage et l'analyse en ligne

****

Directeur de Thèse : **Ladjel BELLATRECHE**
Co-encadrant de Thèse : **Selma KHOURI**
Soutenu le 12 07 2017,
devant la Commission d'Examen

JURY

Rapporteurs :	**Dominique MERY**	**Professeur, LORIA, Nancy, France**
	Parisa GHODOUS	**Professeur, Université de Lyon 1, France**
Membres de jury :	**Yamine AIT AMEUR**	**Professeur, ENSEEIHT, Toulouse, France**
	Abderrafiaa KOUKAM	**Professeur, UTBM, Belfort-Montbéliard, France**
	Ladjel BELLATRECHE	**Professeur, ISAE-ENSMA, Poitiers, France**
	Selma KHOURI	**Maître de conférences: ESI Algérie, ISAE-ENSMA France**

Remerciements

Ce mémoire de thèse est le résultat d'une période de travail acharné qui n'aurait pas pu être achevé sans le soutien, les conseils et l'aide de nombreuses personnes.

Je tiens ainsi à remercier,

Ladjel BELLATRECHE, mon directeur de thèse, professeur des universités à l'Ecole Nationale Supérieure de Mécanique et d'Aérotechnique ENSMA de Poitiers, à la fois chef d'équipe d'Ingénierie des Données et des Modèles, pour avoir accepté de diriger ma thèse et pour m'avoir accueillie dans son équipe, au sein du Laboratoire d'Informatique et d'Automatique pour les Systèmes LIAS. Je le remercie pour ses conseils et ses recommandations, pour m'avoir guidé tout au long de cette thèse tout en m'accordant la liberté et la confiance nécessaire pour faire évoluer mon travail. Je le remercie énormément pour les relectures multiples de ce mémoire et pour ces corrections minutieuses. C'est vraiment un honneur et un véritable plaisir de l'avoir comme directeur de thèse. J'en profite pour lui exprimer ma plus profonde gratitude.

Selma Khouri, Maître de Conférences à l'ESI Algérie et chercheuse associée au LIAS-ENSMA France, mon co-encadrant, pour l'intérêt et la disponibilité qu'elle a manifestés à l'égard de mes recherches ainsi que pour son soutien, sa patience, sa coopération et toutes ces discussions très enrichissantes. Ses lectures, ses suggestions, ses commentaires et ses critiques ont contribué à l'aboutissement de ce travail et qui m'a beaucoup aidé. Qu'elle soit ici assurée de mon très grand respect et du plaisir que j'ai à travailler avec elle.

Dominique MERY et **Parisa GHODOUS**, pour l'honneur qu'ils me font en acceptant cette lourde tâche et d'être les rapporteurs de ce mémoire ; leur lecture attentive et leurs remarques ont permis d'en améliorer la rédaction.

Yamine AIT AMEUR et **Abderrafiaa KOUKAM** d'avoir accepté d'examiner mon travail.

Emmanuel GROLLEAU, le directeur du laboratoire LIAS-ENSMA, pour sa gentillesse, son respect et sa sympathie.

le personnel du laboratoire.

Mes remerciements vont également :

À tous mes ami(e)s et collègues sans exception, pour leur présence, leur soutien et leur respect toute la période de ce mémoire. Je tiens à partager le plaisir de présenter ce mémoire avec eux. Mes pensées vont d'abord à: Ahcène, Abdelkader, Géraud, Guillaume, Okba, Nadir, Selma, Lahcen, Nabila, Abderahmane, ...

Mes très chers parents à qui je dédie ce titre en témoignage de gratitude. Aucune dédicace ne saurait exprimer mon grand amour éternel pour vous et ma grande reconnaissance pour les sacrifices que vous avez consentis pour mon instruction et mon bien être.

Bien évidemment, je remercie profondément mes chères frères, sœurs, beaux frères et belles sœurs qui ont toujours été d'un soutien inconditionnel et sans limites dans les moments difficiles, pour leurs encouragements et pour leurs conseils pendant cette long période, qui ont toujours cru en moi, qui m'ont soutenue et m'ont poussé vers le mieux tout ou long de ce travail . Il m'a été simplement très précieux.

Enfin, Je ne saurais finir sans exprimer mes remerciements à ma chère femme qui m'a supporté, et qui a toujours su me redonner le sourire dans les tous moments. Je remercie bien évidement mon ange Iyad, dont l'existence donne un sens à ma vie, qui à vécu avec moi des bons et mauvais moments de cette thèse.

Je clos ces remerciements par une pensée spéciale pleine d'affection et de nostalgie en mémoire de mon chére frère "SALAH" qui nous a malheureusement quittés, qui, même loin des yeux, fait toujours partie de ma vie et à tout jamais ..., trouvez en ce titre un petit geste de reconnaissance et que dieu - que je remercie pour tout - nous réunisse au paradis.

Que tous ceux et celles que je n'ai pas mentionnés n'en reçoivent pas moins ma gratitude.

Table des matières

Chapitre 1 Introduction Générale **1**

1 Contexte 3

2 Bilan 8

3 Problématique 9

4 Nos Objectifs et Contributions 12

4.1 Les scénarios d'intégration 12

4.2 Définition du schéma de l'entrepôt 13

4.3 L'intégration des instances 14

4.4 Déploiement et exploitation de l'entrepôt 14

5 Structure du mémoire 15

5.1 Partie état de l'art 15

5.2 Partie Contributions 15

6 Publications 16

Partie I Background & État de l'art

Chapitre 2 Notions de base **21**

1 Introduction 24

2 L'ingénierie des besoins 25

2.1 Définitions 25

2.2 Le processus de l'ingénierie des besoins 26

2.3 Approches d'expression des besoins 27

2.3.1 Approche orienté buts 28

2.3.2 Approche à base de scénarii 28

2.3.3 Approches basées sur les automates et réseaux de Petri . 28

2.4 Techniques de modélisation des besoins 29

2.4.1 Définitions 30

2.4.2 Les techniques informelles (Langage naturel) 31

2.4.3 Les techniques semi-formelles 32

2.4.4 Les techniques formelles 32

3 Les solutions d'entreposage 32

3.1 Les ED : des systèmes d'intégration matérialisés 33

3.2 La modélisation multidimensionnelle 35

3.3 L'étape d'intégration de données (ETL) 36

3.4 Déploiement de l'ED 36

3.5 La place des besoins utilisateurs dans les solutions d'entreposage . 38

4 Les ontologies pour la gestion des données et des besoins 39

4.1 Définitions 39

4.2 Taxonomie des ontologies 40

4.2.1 Classes d'ontologies selon le type des concepts 40

4.2.2 Classes d'ontologies selon l'objet de conceptualisation . 41

4.3 Les formalismes ontologiques 42

4.4 Représentation formelle des ontologies 42

4.5 Les ontologies pour la gestion des besoins 43
4.6 Les ontologies dans les systèmes de gestion des données 44
4.6.1 Les bases de données à base ontologique (BDBO) 44
4.6.2 Les ontologies dans le processus d'intégration 46
5 Le matching comme solution d'intégration d'ontologies 48
5.1 Définitions 49
5.1.1 Terminologie 49
5.1.2 Méthodes de comparaison et mesures de similarité 50
5.2 Les étapes du processus du matching 51
5.3 Classification des techniques de matching d'ontologies 52
5.3.1 Techniques terminologiques 53
5.3.2 Techniques structurelles 53
5.3.3 Les techniques extensionnelles 54
5.3.4 Les techniques sémantiques 54
6 Conclusion 55

Chapitre 3 État de l'art sur l'unification, la gestion et l'exploitation des besoins utilisateurs 57

1 Introduction 60
2 Présentation des travaux : état de l'art 61
2.1 L'intégration des besoins utilisateurs 61
2.1.1 L'intégration des langages de modélisation 61
2.1.2 L'intégration des terminologies utilisées 65
2.1.3 L'intégration de l'univers du discours 66
2.2 La modélisation et l'analyse des BU 67
2.3 La matérialisation et l'exploitation des BU 70
3 Nos critères de classification 72
3.1 L'intégration des BU 72

3.2 Modélisation et analyse des BU 73
3.3 Matérialisation et exploitation des BU 73
4 Synthèse et positionnement de nos contributions 74
5 Conclusion 77

Partie II Contributions: Construction d'un entrepôt sémantique de besoins fonctionnels

Chapitre 4 Construction de l'Ontologie Intégrante 81
1 Introduction 84
2 Architecture générale de l'approche d'entreposage 86
3 Scénarios d'intégration 87
4 Approche de construction de l'ontologie intégrante 88
4.1 Scénario d'une ontologie partagée 88
4.1.1 Hypothèse 1 88
4.1.2 Hypothèse 2 92
4.2 Scénario d'intégration "ontologies multiples" 93
4.2.1 Matching ontologique 94
4.2.2 Algorithme de génération de l'ontologie intégrante . . . 99
5 Validation de l'approche : études de cas 99
5.1 Étude de cas 1 : ontologie partagée 101
5.2 Étude de cas 2 : ontologies multiples 103
5.2.1 Performances du mécanisme de matching 105
6 Conclusion 106

Chapitre 5 Un Modèle Multidimensionnel pour l'Entrepôt des Besoins Fonctionnels **107**

1 Introduction . . . 109

2 Présentation des sources . . . 110

2.1 Formalismes sources et ontologies locales . . . 110

2.1.1 Connexion du formalisme MCT avec l'ontologie local (OntoMCT) . . . 111

2.1.2 Connexion du formalisme des buts avec l'ontologie locale (OntoGoal) . . . 111

2.1.3 Connexion du formalisme des cas d'utilisation avec l'ontologie locale (OntoUseCase) . . . 111

3 Le modèle pivot proposé . . . 114

4 Le modèle conceptuel de l'ESBF . . . 114

4.1 Révision du modèle pivot . . . 114

4.2 Modélisation multidimensionnelle du schéma de l'entrepôt . . . 119

4.2.1 Étendre le MPE avec les dimensions temps et localisation 119

4.2.2 Définition du schéma de l'entrepôt de besoins . . . 121

5 Transformation des besoins sources vers le modèle de l'entrepôt . . . 123

6 Connexion des modèles de besoin et ontologiques . . . 125

7 Conclusion . . . 127

Chapitre 6 Un Processus ETL pour les Besoins Fonctionnels **129**

1 Introduction . . . 132

2 Composantes de la phase ETL . . . 133

2.1 Les mappings M . . . 134

2.2 Les opérateurs ETL . . . 134

2.2.1 Les opérateurs de base . . . 135

2.2.2 Les opérateurs dédiés aux besoins . . . 136

2.2.3 Les opérateurs d'inférence . . . 141

2.2.4 Les opérateurs de gestion 143
2.3 Le processus ETL de besoins 145
2.4 L'algorithme ETL 146
3 Raisonnement sur les besoins 147
3.1 Relations sémantiques entre les besoins : définitions 149
3.2 Mécanisme de raisonnement 150
3.2.1 Les règles d'identification 150
3.2.2 Les règles d'inférence 153
3.2.3 Les règles de vérification de la cohérence 153
4 Implémentation et Expérimentation 154
4.1 Performances du raisonnement 155
4.2 Complexité de l'algorithme ETL de besoins 160
5 Conclusion 161

Chapitre 7 Déploiement de l'Entrepôt Sémantique de Besoins Fonctionnels 163

1 Introduction 165
2 La conception logique de l'entrepôt 166
3 La conception physique de l'entrepôt 167
3.1 Déploiement de l'entrepôt de besoins sur une BDBO Oracle 168
3.1.1 Construction du schéma de l'entrepôt 169
3.1.2 Traduction des opérateurs ETL 170
3.2 Déploiement de l'entrepôt sur une BDBO OntoDB 175
4 Exploration et analyse des besoins entreposés 176
5 Evaluation : ETL et raisonnement 178
5.1 Scénario 1 : ontologie partagée 179
5.1.1 Les besoins sources 179
5.1.2 Moteur d'inférence 179
5.1.3 Machine physique: 180

5.1.4 Performances de l'ETL et du raisonnement 180
5.2 Scénario 2 : ontologies multiples 181
5.2.1 Performance de l'ETL 181
5.2.2 Performance du mécanisme de raisonnement 183
6 Présentation de l'outil d'unification des besoins 184
6.1 Architecture fonctionnelle de l'outil 184
6.2 Fonctionnement de l'outil 186
6.2.1 Chargement et visualisation des entrées de notre système 186
6.2.2 Mécanisme de matching 188
6.2.3 Processus ETL 189
6.2.4 Mécanisme de raisonnement 190
7 Conclusion 191

Chapitre 8 Conclusion et perspectives **193**

1 Conclusion & Perspectives 195
1.1 Synthèse du travail 195
1.2 Contributions 195
1.2.1 Construction de l'ontologie intégrante 196
1.2.2 Définition du schéma de l'entrepôt de besoins 196
1.2.3 Définition d'un ETL pour l'alimentation de l'entrepôt . . 197
1.2.4 Déploiement et exploitation de l'entrepôt de besoins . . . 197
1.3 Perspectives 197
1.3.1 Intégration des bases de connaissances 198
1.3.2 Validation de l'approche sur un projet réel 198
1.3.3 D'autres mécanismes de raisonnement 198
1.3.4 Stratégie d'évolution des sources 199
1.3.5 Davantage de décisionnel sur les besoins 199
1.3.6 Vers un entrepôt générique 199

Bibliographie **201**

Annexes **213**

1 Les besoins du CMS document 217

2 Les besoins du systèmes de gestion des conférences: 225

Table des figures **233**

Liste des tableaux **237**

Glossaire **239**

Chapitre 1

Introduction Générale

Sommaire

1	**Contexte**	3
2	**Bilan**	8
3	**Problématique**	9
4	**Nos Objectifs et Contributions**	12
5	**Structure du mémoire**	15
6	**Publications**	16

1 Contexte

Depuis deux décennies, les organisations et les entreprises sont confrontées à un nombre important de problèmes informatiques liés à la variété des objets qu'elles manipulent. Les éléments suivants représentent des exemples des objets : (a) les sources de données de différents types (par ex. relationnelles, XML, documents, graphes, etc.), (b) les processus/services exploitant les données, (c) les bases de connaissances (par ex. *Yago, DBPedia, Nell, DeepDive, Google's Knowledge Vault* et *Freebase*) explicitant le sens de certains objets comme les données et les documents, (d) les plateformes de déploiement dédiées à stocker les objets (par ex. les machines centralisées, parallèles, Cloud, etc.), (e) le matériel de traitement des objets (CPU, GPU, etc.), (f) les formalismes de modélisation des objets (par ex. UML, Entité-Association, SysML, BPMN, etc.), les langages de définition/manipulation/configuration des objets (par ex. SQL, Sparql, BPEL, etc.), (g) les langages de programmation (par ex. Java, Python, Scala, etc.), etc. Le rapport Beckman [1] édité en 2016 [6] a souligné l'importance et l'enjeu de la variété pour la communauté de bases de données. En même temps, la variété peut représenter un atout pour les entreprises car en l'exploitant elle peut générer de la valeur ajoutée. La variété est toujours implicitement liée à la valeur [23]. En d'autres termes, la variété des objets pousse les entreprises à les analyser afin d'extraire une valeur ajoutée. L'arrivée de l'ère de Données Massives (Big Data) a rendu le lien entre la valeur et la variété *plus explicite* (les deux Vs).

La communauté de bases de données est considérée comme celle la plus confrontée au phénomène de la variété. Cette situation est motivée par les besoins d'intégration, de partage et d'analyse de la mine d'or des données. En conséquence, des systèmes d'intégration de données, souvent associés à des outils commerciaux et académiques (par. Informatica, Pantaho, Talend, etc.) ont été largement développés [49]. Le processus de construction d'un tel système doit prendre en compte le phénomène de la variété, ce qui complexifie leur tâche [49]. Cette difficulté est due principalement aux facteurs suivants [17] : **(a)** l'autonomie des sources, **(b)** l'hétérogénéité des sources, **(c)** l'évolution des sources et **(d)** le fait que la sémantique des sources soit peu explicitée.

- *L'autonomie des sources*. La plupart des sources participant à l'intégration de données fonctionnent en mode totalement autonome. Autrement dit, les propriétaires de ces sources sont tout à fait libres d'utiliser leur propre langage, vocabulaire, et plateforme de déploiement pour modéliser, définir et exploiter leurs sources. Ces propriétaires sont également libres de modifier leurs schémas ou mettre à jour leurs contenus, sans en faire état préalablement aux utilisateurs. Cette autonomie est une des causes de la variété.
- *L'hétérogénéité des données*. Ce facteur est une conséquence de l'autonomie. L'hétérogénéité concerne à la fois la structure et la sémantique. L'hétérogénéité structurelle provient du fait que les sources de données peuvent avoir différentes structures et/ou de différents formats pour stocker leurs données. De nombreux travaux concernant l'hété-

1. Un groupe de chercheurs de base de données se réunit périodiquement pour discuter de l'état des lieux et les orientations clés à l'avenir

rogénéité structurelle ont été proposés dans les contextes des bases de données fédérées et des multi-bases de données [17]. L'hétérogénéité sémantique, par contre, présente un défi majeur dans le processus d'élaboration d'un système d'intégration. Elle est due aux différentes interprétations des objets du monde réel [58, 162].

- *Dynamicité des sources*. Les fournisseurs des sources de données étant différents, chaque source doit pouvoir, en plus, se comporter indépendamment des autres. En conséquence, la relation entre le système intégré et ses sources est faiblement couplée. Une source doit pouvoir modifier sa structure et sa population sans en informer les autres et sans que cela engendre des anomalies de maintenance du système d'intégration. Dans un tel contexte asynchrone, l'évolution du système d'intégration concerne à la fois les schémas et les populations intégrées [20].
- *La sémantique de sources de données est peu explicitée*. La plupart des sources de données candidates à l'intégration n'ont pas été conçues pour être intégrées dans le futur mais pour satisfaire des applications de tous les jours [17, 18]. Souvent, le peu de sémantique contenue dans leurs modèles conceptuels (dans le cas, où chaque source est une base de données) est perdue par le fait que leurs modèles physiques sont utilisés par les applications. Le modèle conceptuel permet d'exprimer à la fois les besoins applicatifs et la connaissance du domaine sous une forme intelligible pour un utilisateur ultérieur. L'absence de représentation du modèle conceptuel ou de toute autre représentation sémantique dans les bases de données rend l'interprétation et la compréhension de celle-ci très difficile même pour les concepteurs ayant une bonne connaissance du domaine d'application.

En analysant de près les systèmes d'intégration de données, nous remarquons qu'ils utilisent deux principales architectures : **(i)** une architecture à base de médiation et **(ii)** une architecture à base de dépôt. Les systèmes à base de médiation consistent à développer une application chargée de jouer le rôle d'interface entre les sources de données locales et les applications d'utilisateurs. Cette architecture a été utilisée par un nombre important de projets [58, 102, 12]. Elle repose sur deux composants essentiels (Figure 1.1): le médiateur et l'adaptateur. Le médiateur se charge de la localisation des sources de données et des données pertinentes par rapport à une requête, il résout de manière transparente les conflits des données qui pouvaient être causés par la variété. Un ensemble de connaissances sur les sources permet au médiateur de générer un plan d'exécution pour traiter les requêtes d'utilisateurs. L'adaptateur permet à un (ou plusieurs) médiateur(s) d'accéder au contenu des sources d'informations dans un langage uniforme. Il fait le lien entre la représentation locale des informations et leur représentation dans le modèle de médiation. Il a le rôle d'extraction, de transformation, de nettoyage, de transformation des données locales et leur préparation pour les requêtes du médiateur. Dans cette architecture, les données *ne sont pas stockées au niveau du médiateur*. Elles restent dans les sources de données et ne sont accessibles qu'à ce niveau. L'intégration d'information est fondée sur l'exploitation de vues abstraites décrivant de façon homogène et uniforme le contenu des sources d'information dans les termes du médiateur.

Les dépôts de données ont été hérités de la communauté de base de données [24]. Dans

ce contexte, le mot "dépôt" fait référence à une extension d'un système de gestion de base de données avec une couche de contrôle explicite mettant l'accent sur la gestion des méta-données. Un dépôt peut ensuite être défini comme une "base de données partagée d'informations sur les artefacts fabriqués et utilisés par une entreprise" [24].

Dans le cadre d'une intégration de données, un dépôt intégré consiste à persister l'ensemble de données des sources. Les systèmes d'intégration matérialisés et les entrepôts décisionnels de données sont des exemples des dépôts intégrés. Dans ces derniers, des processus similaires à ETL (Extract, Transform, Load) sont nécessaires. Ils effectuent certaines tâches similaires de l'adaptateur pour prendre en charge la variété, en extrayant les données des sources, les nettoyant et les chargeant dans l'entrepôt. L'une des caractéristiques de ces dépôts est qu'ils offrent une vue historique sur les données. Une fois construit, un dépôt intégré peut être exploité comme un dépôt traditionnel avec des opérations transactionnelles, des techniques de fouille de données, l'analyse multicritère comme le Skyline [25], ou par des outils d'analyse de type OLAP (On-Line Analytical Processing). Les outils décisionnels représentent un atout considérable pour les entreprises. Ils ont contribue largement au succès de plusieurs entreprises comme Coca-Cola [2], Wal-Mart [3] [156] et le groupe français Casino [4] [27].

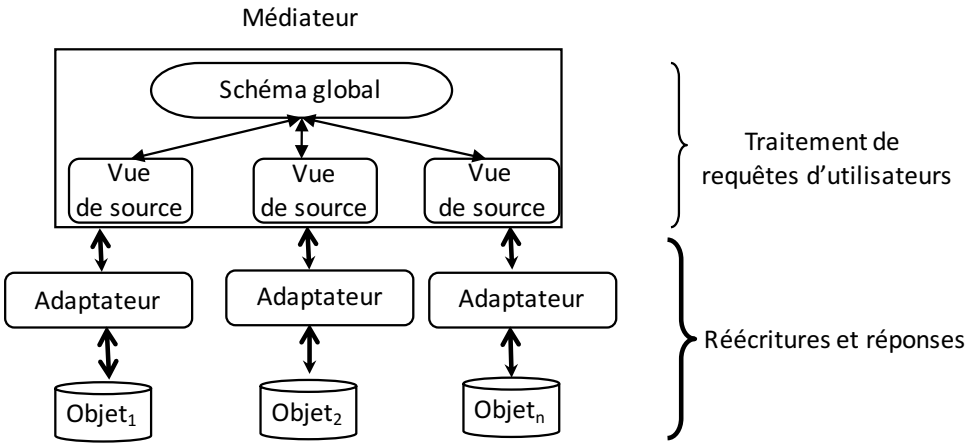

FIGURE 1.1 – Architecture de médiation

La notion de dépôt n'est pas spécifique à la données, mais elle a été également utilisée par d'autres communautés. Par exemple, la communauté de services/processus a élaboré des dépôts dédiés aux services/processus afin de fournir un annuaire élaboré de services supportant la description, la recherche et la découverte, la persistance et l'évolution de services/processus. Nous pouvons citer l'exemple du dépôt APROMORE (An advanced PROcess MOdel REpository) [131]. Récemment, dans [16], les auteurs recommandent l'association de la dimension d'analyse d'OLAP et les dépôts de services/processus pour augmenter le pouvoir décisionnel au niveau des processus. D'autres exemples de dépôts de brevets et de traitements de maladies

2. http://www.coca-colacompany.com/
3. www.walmart.com
4. https://www.groupe-casino.fr/fr/

(comme *PatientsLikeMe* [5]) existent et permettent la réutilisation et le partage de la connaissance. Pour résumer, nous pouvons avancer que les systèmes de dépôts des objets se déclinent en deux

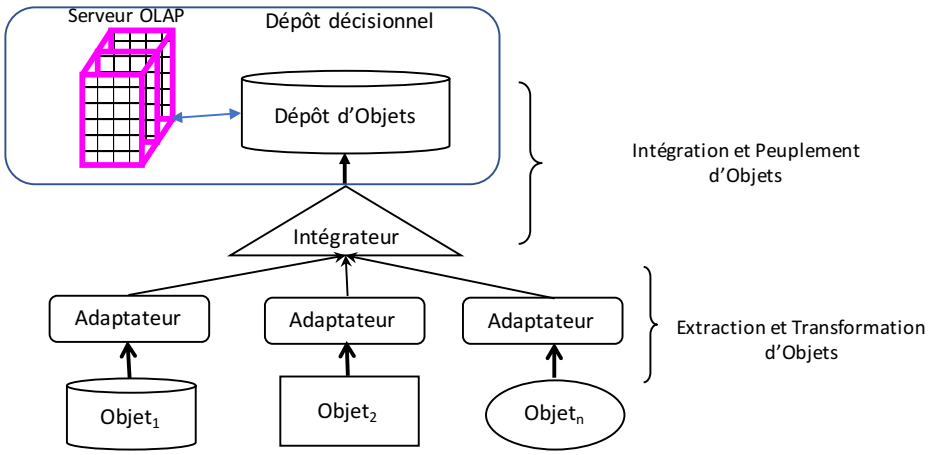

FIGURE 1.2 – Architecture d'un dépôt traditionnel/décisionnel

sous systèmes (Figure 1.2): (i) les dépôts transactionnels des objets (comme le cas des bases de données de type OLTP: On-Line Transaction Processing) et (ii) les dépôts analytiques (comme pour le cas des entrepôts de données décisionnels de type OLAP).

Jusqu'à maintenant, nous avons présenté des exemples de dépôts concernant cinq objets variés, à savoir les données, les processus/services, les brevets, et les traitements. Il existe une dimension de variété citée par le rapport de Beckman, tout aussi importante, et qui commence à attirer l'attention de la communauté de recherche, qui est la variété des utilisateurs et de leurs besoins fonctionnels et non-fonctionnels. Dans cette thèse, nous nous intéressons aux besoins fonctionnels. Ces besoins sont des pré-conditions de tout projet informatique. Notons que tous les niveaux de variété cités des différents objets (sources, systèmes, plateformes, etc.) proviennent eux même d'une variété des besoins à l'origine du développement de ces objets. Avec l'émergence de la technologie des entrepôts de données de type OLAP dans les années 1990, les besoins ont pris une autre dimension, et ont contribué à la naissance d'une approche de leur conception dirigée par les besoins fonctionnels (dite approche de Kimball). Cette approche a été et mise en œuvre dans plusieurs projets.

Comme nous l'avons déjà souligné, l'Ingénierie des Besoins (IB) précède la conception qui précède le codage des applications. Le processus d'IB suit un cycle de vie bien identifié qui comporte les phases suivantes : l'élicitation des besoins, leur modélisation, leur analyse, leur validation et leur gestion. La phase d'*élicitation* consiste à identifier, collecter et développer les besoins à partir d'une variété de sources d'informations comprenant également les parties prenantes humaines. L'identification des parties prenantes, représentées par l'ensemble des personnes ou des groupes qui sont mis en relation avec le système, est donc est étape nécessaire pour mener la phase d'élicitation.

5. https://www.patientslikeme.com/

La phase de *modélisation* des besoins consiste en la construction d'une description abstraite permettant de détailler, structurer, et documenter les besoins en utilisant différents types de langages (informels, semi-formels et formels).

La phase d'*analyse* des besoins consiste à examiner les besoins modélisés à l'étape précédente afin de fournir une compréhension des besoins élicités et afin d'assurer leur vérification en termes d'exactitude, de complétude, de clarté et de consistance. La norme IEEE 830-1998 [6] pour la spécification des besoins liste huit principales caractéristiques que toute spécification des besoins doit assurer : correcte, sans ambiguïté, complète, consistante, stable, vérifiable, modifiable et traçable.

La phase de *validation* des besoins consiste à vérifier la qualité des besoins et leur conformité aux attentes des parties prenantes. Cette phase permet aussi s'assurer que les besoins définissent les fonctionnalités attendues du système.

La phase de *gestion* consiste à gérer l'évolution des besoins (comme l'ajout de besoins, la suppression ou la correction des erreurs) tout en gardant la cohérence des autres besoins touchés.

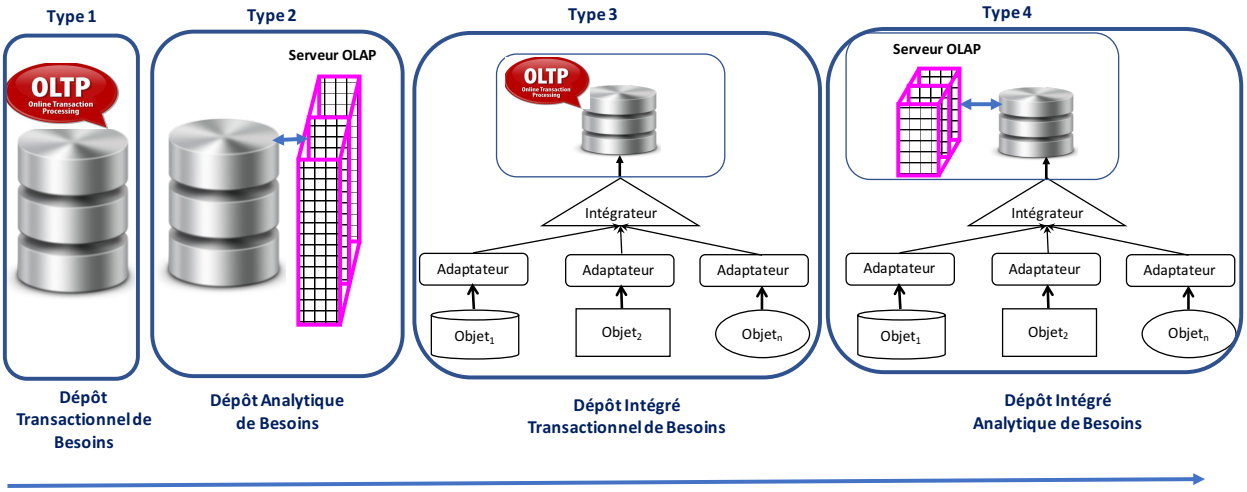

FIGURE 1.3 – Evolution des Dépôts de Besoins

Les besoins fonctionnels suivent le même chemin d'utilisation et d'analyse des autres entités comme nous avons citées. Ils peuvent être persistés dans un dépôt ou intégré et persisté dans un dépôt intégré. D'après BABOK [7], un dépôt de besoins est défini comme une méthode de stockage des besoins incluant les besoins en développement, les besoins en cours de validation et les besoins validés. Dans ce contexte, un dépôt peut être vu comme une base de données permettant de persister les besoins. Avec l'émergence des entreprises étendues, caractérisées

6. http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

7. A Guide to the Business Analysis Body of Knowledge® (BABOK® Guide) est le standard facilitant aux praticiens et aux utilisateurs de créer une valeur ajoutée et d'obtenir de meilleurs résultats à leurs activités commerciales.

par un nombre important de sous-entreprises et/ou des sous-traitants, un nombre d'initiatives a été lancé pour intégrer les besoins fonctionnels issus des partenaires hétérogènes [27]. Nous citons l'exemple de l'entreprise Airbus qui implique quatre sites dans différents pays : Hambourg (Allemagne), Filton (Angleterre), Toulouse (France) et Madrid (Espagne) [27]. Ces sous-entreprises travaillent souvent de façon autonome, leur nombre peut s'accroitre rapidement, et leurs besoins également. Des approches d'intégration des besoins conceptuelles et ontologiques ont été proposées [27, 157]. Le dépôt intégré final est souvent exploité par des outils transactionnels.

2 Bilan

En examinant la littérature, nous avons identifié l'évolution des dépôts de besoins (Figure 1.3). Le premier type de dépôt se fait à l'instar des données et permet des traitements transactionnels sur les besoins, un accès optimisé aux besoins et assure leur traçabilité. Le système Doors [8] d'IBM est un exemple de cette base.

Ce type a évolué pour un deuxième type intégrant la dimension décisionnelle. Un exemple de ce type de système est IBM doors next generation [9], permettant des analyses OLAP via un tableau de bord, des techniques de slice/dice sur les besoins et des rapports simplifiés [10]. L'aspect décisionnel concerne principalement l'exploitation du premier type de dépôt. Les aspects décisionnels ont été exploités dans les travaux de l'ingénierie des besoins de plusieurs façons. Certains travaux incorporent la prise de décision pendant le processus de l'IB, souvent pour faire un choix pertinent quant aux techniques utilisées, les parties prenantes à inclure, les priorités et choix des besoins en cas de besoins conflictuels, etc. [135, 160]. En cas de conflits, une négociation est envisagée pour amener les parties prenantes à un accord. Pour mener une négociation pertinente, plusieurs travaux font appel à des techniques d'analyse multicritère [135]. Ces travaux sont connus sous le nom "la prise de décision dans l'ingénierie des besoins". Ces techniques ont pour objectif de choisir la meilleure solution ou la solution optimale parmi tout un ensemble de solutions envisageables. D'après Wikipédia [11], un abus de langage courant consiste à la confondre avec l'informatique décisionnelle (en anglais "business intelligence") et l'aide à la décision multicritère. Dans cette thèse, nous nous sommes concentrés sur la dimension OLAP pour obtenir des analyses de synthèse telles que celles utilisées en analyse financière. Le type 2 que nous identifions dans la figure 1.3 fait référence au décisionnel OLAP.

Le 3ème type concerne les dépôts intégrés de besoins exploités avec des outils transactionnels [27, 157].

Pour reproduire les mérites des systèmes d'entreposage décisionnels de données, nous pen-

8. http://download.boulder.ibm.com/ibmdl/pub/software/rationalsdp/documentation
9. http://www-03.ibm.com/software/products/fr/ratidoorng
10. https://www.youtube.com/watch?v=qYK7_g4Fy44&feature=youtu.be
11. https://fr.wikipedia.org/wiki/Aide_à_la_décision_multicritère

sons qu'un *quatrième type de dépôt est nécessaire*, où les besoins fonctionnels de différents partenaires autonomes sont intégrés avec une approche d'entrepôt de données, avec l'ensemble de phases de conception (conceptuelle, logique, ETL, physique) (Figure 1.4). Ce nouveau type de dépôt permettrait d'effectuer des analyses de façon flexible, en se basant sur des structures optimisées pour les besoins d'analyse (schéma multidimensionnel), d'utiliser des outils d'analyses dédiés et de fournir des analyses compréhensibles par des décideurs et gestionnaires sans le recours systématique aux équipes IT.

3 Problématique

Construire un dépôt intégré de besoins fonctionnels est confronté à cinq exigences principales liées à la variété souvent générée par l'autonomie des sources. Garantir l'autonomie des partenaires signifie que chacun puisse avoir son propre vocabulaire, son propre univers de discours et son propre formalisme lors de la définition des besoins, sans se conformer à un schéma prédéfini. Par la suite, nous détaillons ces exigences.

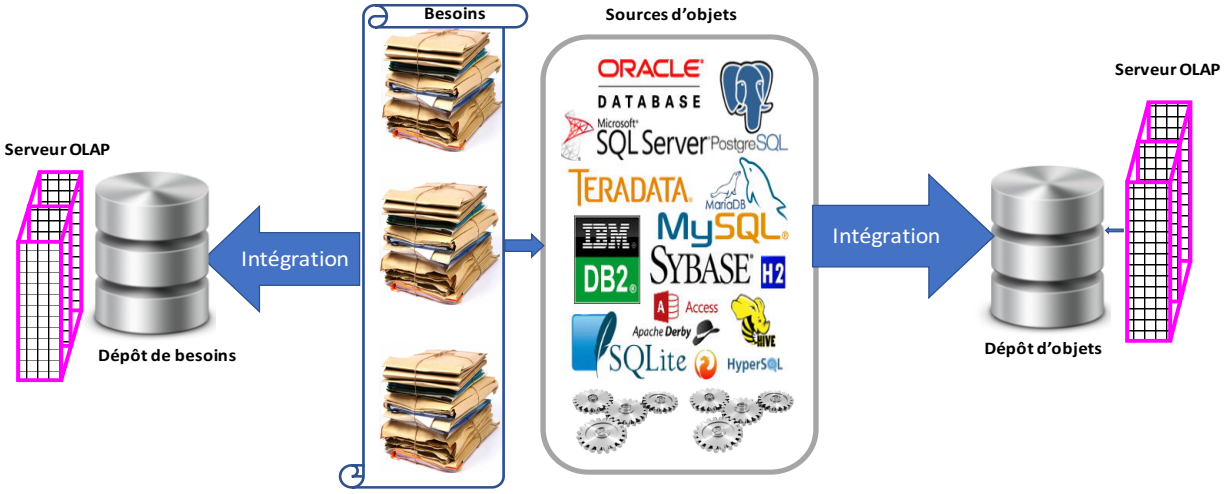

FIGURE 1.4 – Vers un dépôt décisionnel de besoins

– **Exigence N°1 : La diversité des vocabulaires d'expression de besoins.** L'expression des besoins peut se faire selon plusieurs vocabulaires ou terminologies propres utilisés pour l'expression des besoins. Cette situation est à l'origine de plusieurs types de conflits syntaxiques comme les conflits d'homonymie (le même nom d'entité utilisé avec des significations différentes), les conflits de synonymie (des termes différents utilisés avec des significations identiques) ou simplement les conflits dus à l'utilisation de langues différentes entre les partenaires (par exemple, certains besoins sont exprimés en langue française alors que d'autres sont exprimés en langue anglaise).
– **Exigence N°2 : La diversité des concepts utilisés pour exprimer les besoins.** Cette hétérogénéité est due aux différents points de vue des concepteurs sur les entités du monde

réel, ce qui implique d'avoir différentes interprétations des objets exprimés dans la définition des besoins. Les mêmes types de conflits sémantiques survenant entre les sources de données stockant des entités du monde réel, se manifestent également dans la définition des besoins qui utilisent ces entités. Ces conflits sont classés dans [64] comme suit :

- *Les conflits de représentation* : surviennent lorsqu'un même concept est représenté par des schémas différents selon le collecteur de besoins ayant fourni ce schéma. Par exemple, la distinction entre un client de sexe féminin et masculin peut se faire par un attribut ou par deux entités différentes.
- *Les conflits de contexte* : surviennent lorsqu'une entité est représentée par plusieurs représentations selon le contexte local d'un collecteur de besoins. Par exemple, le prix d'un article est considéré comme le prix hors taxe pour un collecteur/concepteur, alors qu'il est considéré comme le prix taxe comprise par un autre collecteur.
- *Les conflits de mesure de valeur* : surviennent lorsque la manière de coder la valeur d'une propriété du monde réel diffère d'un collecteur de besoins à un autre. Par exemple, le prix d'un article est calculé en euros par un collecteur alors qu'il est considéré en dollars par un autre collecteur de besoins.

L'hétérogénéité sémantique présentant le défi majeur dans le processus de définition d'un système d'intégration, de nombreuses approches ont proposé d'associer aux données des ontologies qui en définissent le sens, afin de réduire l'hétérogénéité sémantique. Une ontologie est définie comme une spécification explicite d'une conceptualisation [121]. Deux principales catégories d'ontologies ont été utilisées : les ontologies linguistiques et les ontologies conceptuelles. Les ontologies linguistiques définissent le sens des termes ainsi que les relations terminologiques entre eux (comme la synonymie et l'homonymie). Plusieurs ontologies linguistiques sont disponibles comme Wordnet et MeSH (Medical Subject Heading). Les ontologies conceptuelles définissent formellement la sémantique du domaine à travers des concepts, leurs propriétés ainsi que les relations entre les concepts. Nous considérerons dans nos propositions les deux couches ontologiques (linguistique et conceptuelle)

- **Exigence N°3 : La diversité des langages de modélisation des besoins.** L'expression des besoins peut se faire selon plusieurs langages de modélisation, qui s'étendent des langages complètement informels comme la langue naturelle, jusqu'aux langages formels (comme le langage Z ou B utilisés pour la formalisation des besoins), en passant par les langages semi-formels comme les diagrammes UML ou les modèles orientés buts. Selon leur expertise, des concepteurs différents peuvent choisir de modéliser leurs besoins utilisant différents langages appartenant à la même famille de formalismes ou même à différentes familles de formalismes (formel et semi formel par exemple). Une autre difficulté apportée par ces différents types de diversité est la présence de relations complexes entre les besoins. Après l'élimination des différents types de conflits existants entre les besoins, il est nécessaire d'identifier et d'analyser les relations complexes qui lient ces besoins entre eux. Nous pouvons citer les relations d'inclusion, de précédence ou d'équi-

valence. L'identification de ces relations permet d'identifier les besoins pertinents des besoins redondants et d'établir des relations d'ordonnancement et de priorité entre ces besoins.

– **Exigence N°4 : L'automatisation de l'intégration.** L'utilisation des ontologies est considérée par de nombreux travaux comme le facteur d'automatisation du processus d'intégration. Le niveau d'automaticité des approches d'intégration dépend du type d'ontologie utilisé. Les approches utilisant des ontologies linguistiques assurent la gestion des conflits syntaxiques, mais nécessitent la présence d'un expert humain pour le traitement des conflits sémantiques. Ces approches sont donc semi-automatiques. Contrairement aux ontologies linguistiques qui comparent automatiquement les « noms » des relations ou des attributs, les ontologies conceptuelles sont orientées "concept" et non "terme". Ces ontologies sont formelles et donc traitables par machine. L'utilisation d'ontologies conceptuelles permet d'automatiser le processus d'intégration. Plusieurs projets ont été menés utilisant des ontologies conceptuelles, comme le projet PICSEL [63] pour l'intégration des services web, le projet COIN [64] pour l'échange des données financières et le projet OntoDaWa [162], développé au laboratoire pour l'intégration des catalogues de composants de l'entreprise Renault. Ces projets ont aussi été encouragés par la prolifération des ontologies conceptuelles dans de nombreux domaines comme la médecine où nous trouvons l'ontologie UMLS (Unified Medical Language System) ou le domaine de l'ingénierie où nous trouvons des ontologies décrivant les catalogues de composants électroniques qui sont normalisées par l'ISO (PLIB : ISO 132584 "parts Library").
  Trois principales structures de systèmes d'intégration à base ontologique ont été proposées [155]: structure à base d'une ontologie unique, structure à base d'ontologies multiples, structure à base d'une ontologie partagée. Dans la structure à base d'une ontologie unique, chaque source d'informations est liée à une unique ontologie. Cette structure impose de trouver un vocabulaire minimal commun partagé entre les sources et limite l'autonomie des schémas des sources. De plus, des changements au niveau des sources peuvent affecter la conceptualisation du domaine représenté par l'ontologie. Dans la structure à base d'ontologies multiples, chaque source d'informations définit sa sémantique par une ontologie locale. Dans la structure à base d'une ontologie partagée, chaque source est décrite sémantiquement par sa propre ontologie locale. Afin de faciliter les mappings entre les ontologies locales, elles sont mises en correspondance avec une ontologie partagée modélisant un domaine particulier, appelée ontologie globale. Nous étudierons particulièrement les deux scénarios d'ontologie multiple et d'ontologie partagée, qui sont les scénarios les plus plausibles pour l'intégration des besoins.
– **Exigence N°5 : Exploiter autant que possible les avancés des entrepôts de données.** Historiquement, la technologie des entrepôts de données ont concerné les données traditionnelles. Ensuite ont émergé de nouveaux entrepôts pour prendre en compte d'autres types de données telles que les données XML, spatiales, sémantiques, tweets, données NoSQL, etc. Des outils d'analyse traditionnels ont également suivi cette évolution (par

ex. XML-OLAP pour les données semi-structurées ou S-OLAP pour les données spatiales). En analysant finement le paysage de cette technologie afin de la reproduire sur les besoins fonctionnels, nous constatons qu'elle est plus mature pour les données structurées. Des tentatives de construction des entrepôts de documents en exploitant la maturité des entrepôts de données ont été proposées [73]. Cette revisite a fait recours aux techniques et outils de traitement de la langue naturelle (TAL) [73] pour gérer la partie textuelle. Les besoins fonctionnels exprimés dans un formalisme semi-formel peuvent être vu comme un mixe entre les données et les documents. Nous devons nous rapprocher au maximum du processus de conception d'entrepôts de données, ce qui implique un bon choix des modèles de formalisation et de définition des besoins. Fournir un entrepôt de besoins doit également être accompagné des techniques d'exploitation OLAP adaptés aux besoins, que nous baptisons BF-OLAP (pour OLAP pour les besoins fonctionnels). A l'instar des requêtes OLAP orientées données, *BF-OLAP* permettrait de réponses à des requêtes OLAP sur les besoins comme l'évolution du nombre de besoins dans le temps, le coût des besoins par partenaire, etc. Les techniques OLAP conventionnelles comme le slice, dice, roll up ou drill down doivent également être supportées et adaptées aux besoins.

4 Nos Objectifs et Contributions

Reproduire l'expérience des entrepôts de données sur les besoins représente un enjeu important pour plusieurs projets dans diverses entreprises. Cela leur permet d'intégrer leurs besoins variés et autonomes et d'exploiter avec des outils décisionnels adéquats, une sorte d'OLAP sur les besoins. Pour offrir plus d'autonomie aux sources, nous supposons que chacune utilise son propre vocabulaire, sa propre conceptualisation et son propre formalisme pour décrire les besoins. Comme le montre la Figure 1.5, chaque source décrit la sémantique des besoins par une ontologie locale : *linguistique* pour décrire le vocabulaire utilisé et *conceptuelle* pour décrire les concepts utilisés.

Nos contributions dans cette thèse répondent à l'objectif principal de définir un entrepôt de besoins fonctionnels en permettent de définir le schéma de l'entrepôt et ses instances selon plusieurs scénarios d'intégration :

4.1 Les scénarios d'intégration

Nous avons étudié deux principaux scénarios d'intégration : (i) la structure ontologique à ontologie partagée, qui consiste à ce que chaque source d'un partenaire possède son ontologie locale qui référence à une ontologie partagée. Ce scénario existe dans certains domaines, mais limite l'autonomie des partenaires dans la définition de leurs besoins à un schéma global. (ii) La structure ontologique à ontologies multiples qui consiste à ce que chaque source d'un

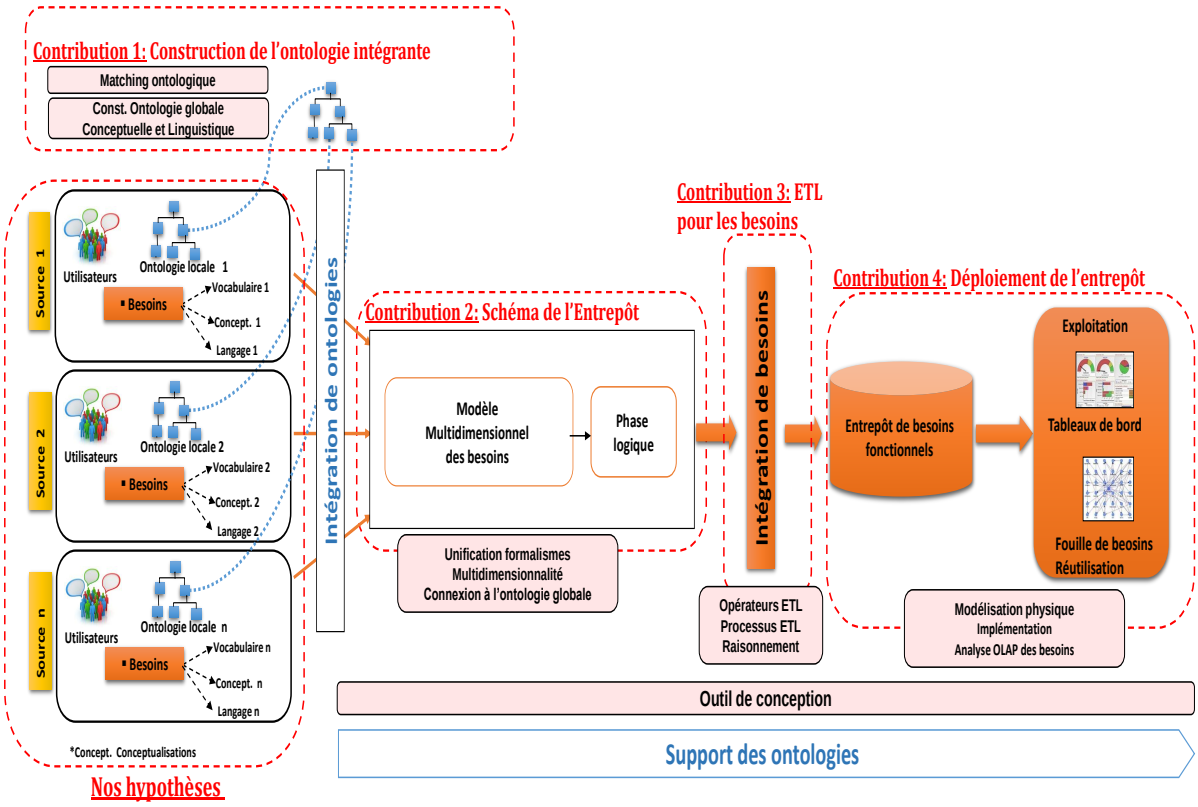

FIGURE 1.5 – Contributions

partenaire possède son ontologie définie localement indépendamment d'un schéma global, et indépendamment des autres partenaires. Ce scénario offre plus d'autonomie aux partenaires dans la définition de leurs besoins. Pour les deux scénarios d'intégration, nous avons fourni une démarche de construction d'une ontologie intégrante unifiant l'ensemble des termes et concepts utilisés dans la description des besoins des sources. Des techniques de matching ontologique sont utilisées et testées pour le scénario d'ontologies multiples.

4.2 Définition du schéma de l'entrepôt

Le schéma de l'entrepôt doit être un schéma intégrant les besoins et gérant les différents niveaux d'hétérogénéité (terminologique, conceptuelle, formalismes, et relations entre les besoins). Une solution d'intégration conjointe sera fournie.

Ce schéma sera défini par l'introduction d'un schéma pivot pour l'intégration des formats. Il sera connecté à l'ontologie intégrante construite, qui définit le sens des besoins. Ce schéma doit représenter les besoins de manière suffisamment détaillée afin de mieux les analyser et de mieux identifier les relations complexes qui lient ces besoins. Une de nos contributions a consisté à fournir un schéma pivot des besoins suivant une représentation dirigée par les tâches.

La représentation des besoins dirigée par les tâches a été adoptée dans plusieurs applications centrées utilisateur qui considèrent les utilisateurs et leurs besoins tout au long du processus de développement d'applications, ou les applications faisant interagir plusieurs utilisateurs comme les applications de design collaboratif [41], les systèmes multi-agents [40] ou les applications de crowdsourcing [74] dans lesquelles les travailleurs du Web complètent une ou plusieurs petites tâches souvent pour des micro-paiements.

Dans notre modèle pivot, nous optons pour la même représentation micro où chaque tâche est décomposée sous forme d'un triplet : le sujet effectuant la tâche, une action et un objet affecté par la tâche. Cette décomposition nous permet d'identifier précisément les relations entre les besoins comme les relations conflictuelles ou les relations d'équivalence. Cette représentation à base de tâche nous permet aussi de représenter un graphe de précédence, dans lequel les noeuds représentent les tâches et les liens les dépendances entre les tâches. Dans ce graphe de précédence, nous représentons deux principales relations entre les besoins : les relations d'ordonnancement et les relations de contenance. En utilisant le raisonnement ontologique, d'autres types de relations complexes entre les besoins sont identifiées automatiquement. Ce schéma sera ensuite organisé selon une modélisation multidimensionnelle, afin de permettre des analyses OLAP sur les besoins.

4.3 L'intégration des instances

Une fois le schéma de l'entrepôt de besoins défini, l'intégration des instances se fait par la définition d'un processus ETL. Ce processus permet d'extraire les instances de besoins sources, de les transformer selon le schéma cible, et de les charger dans l'entrepôt. Ce processus se base sur des opérateurs ETL pour la transformation des besoins selon des mappings établis. Nous avons redéfini les opérateurs ETL conventionnels afin de les adapter à l'intégration des besoins, et nous revisité la phase ETL classique destiné aux données, pour l'intégration des instances des besoins.

4.4 Déploiement et exploitation de l'entrepôt

Le déploiement de l'entrepôt de besoins consiste en son implémentation selon un ou plusieurs SGBD choisis, en effectuant les phases de conception logique et physique. L'exploitation de l'entrepôt de besoins consiste en son utilisation pour l'*analyse* des besoins entreposés. Cette analyse peut se faire via diverses techniques. Nous avons opté pour la définition de requêtes OLAP sur les besoins et la définition d'un cube multidimensionnel basé sur le modèle multidimensionnel que nous avons défini. Ce cube permet d'effectuer des analyses OLAP plus pointues, qui sont généralement requises par les décideurs d'une organisation.

5 Structure du mémoire

Ce manuscrit de thèse est structuré en deux parties : une partie état de l'art et une partie contributions.

5.1 Partie état de l'art

La première partie présente les concepts permettant d'élaborer nos propositions et comporte deux chapitres. Après le premier chapitre d'introduction, le chapitre 2 présente les concepts fondamentaux des trois disciplines que nous utilisons pour établir nos propositions : (i) *l'ingénierie des besoins* : où nous présentons principalement le processus de l'ingénierie des besoins ainsi que les principales techniques d'expression et de modélisation des besoins. (ii) *Les entrepôts de données* : nous définissons les systèmes d'entrepôts de données, leurs phases de construction et la place des besoins dans leur développement. (iii) *Les ontologies* : après la présentation de la notion d'ontologie de domaine, une classification des ontologies (linguistiques et conceptuelles), les langages de définitions des ontologies et la formalisation des ontologies, nous nous intéressons à leur rôle dans les systèmes de gestion des données, dans les systèmes d'intégration et dans le processus de l'ingénierie des besoins.

Dans le chapitre 3, nous sélectionnons, synthétisons et débattons d'un ensemble de travaux que nous considérons majeurs pour nos problématiques, qui sont liés à l'intégration des besoins, leur gestion (modélisation et analyse), ainsi qu'à leur exploitation. A l'issue de ce chapitre, nous positionnons nos contributions par rapport à l'existant que nous présentons par la suite.

5.2 Partie Contributions

La deuxième partie de ce mémoire est dédiée aux contributions pour la définition d'un entrepôt de besoins fonctionnels.

Dans le chapitre 4, nous décrivons notre première proposition d'un schéma global unifiant et intégrant les schémas des sources, dans notre cas, il s'agit de sources de besoins. Chaque partenaire (source de besoins) exprime ses besoins en utilisant ses propres termes (le jargon) et ses propres concepts (univers de discours). Pour décrire ces concepts et termes utilisés, chaque source possède son ontologie locale linguistique et conceptuelle. L'objectif de ce chapitre est de construire l'ontologie intégrante (des deux couches linguistique et conceptuelle) selon plusieurs scénarios d'intégration.

Dans le chapitre 5 nous proposons la construction du schéma d'un entrepôt de besoins en deux étapes : (i) la définition d'un modèle pivot unifiant les formalismes ou langages de modélisation représentant les besoins des différentes sources et (ii) la représentation multidimensionnelle du modèle. Ce modèle multidimensionnel de l'entrepôt de besoins sera ensuite connecté à

l'ontologie intégrante définie au chapitre précédent afin de permettre une définition sémantique de l'ensemble des besoins intégrés.

Dans le chapitre 6, nous proposons une démarche d'alimentation de l'entrepôt par les besoins des sources. Ceci passe par la définition d'un processus d'Extraction-Transformation et Chargement (ETL) des instances des sources au niveau du schéma cible de l'entrepôt. Un ensemble d'opérateurs ETL ont été définis et adaptés pour le cas des besoins. Afin d'assurer un processus ETL de qualité, nous avons enrichi les opérateurs ETL conventionnels par des opérateurs de raisonnement permettant d'identifier des relations entre les besoins ainsi que les besoins inconsistants afin de les éliminer. Des expérimentations sont réalisées afin d'évaluer les performances du processus ETL et du mécanisme de raisonnement. Ces expérimentations utilisent le banc d'essai LUBM [12], le document de besoins Courses Management System CMS [13],) et l'ontologie linguistique EuroWordNet [14], pour valider le scénario d'intégration d'ontologie partagée. Pour valider le second scénario d'ontologies multiple, nous utilisons trois systèmes de soumission à des conférence et des ontologies locales obtenues à partir du projet *MultiFarm* [15].

Dans le chapitre 7, nous procédons aux phases finales de déploiement et d'exploitation de l'entrepôt de besoins fonctionnels. Le déploiement de l'entrepôt comprend trois étapes : le choix du système de gestion de bases de données (SGBD) où nous avons opté pour deux SGBD sémantiques : Oracle et OntoDB, la conception logique et la conception physique de l'entrepôt. La phase d'exploitation est assurée par des requêtes OLAP et par l'implémentation d'un cube multidimensionnel des besoins unifiés.

Le dernier chapitre conclut ce mémoire de thèse, établit un bilan des contributions apportées et trace différentes perspectives de recherche.

6 Publications

- Zouhir Djilani, Selma Khouri: Understanding User Requirements Iceberg: Semantic Based Approach. Proceedings of the 5th International Conference on Model and Data Engineering (MEDI), pp. 297-310, Rhodes, Greece, September 26-28, 2015. LNCS, Springer [47].
- Zouhir Djilani, Nabila Berkani, Ladjel Bellatreche: Towards Functional Requirements Analytics. Proceedings of 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA), pp. 358-373, Imperial, Corfu, Greece, October 10-14, 2016. LNCS, Springer [45].
- Zouhir Djilani, Abderrahmane Khiat, Selma Khouri, Ladjel Bellatreche: MURGROOM: multi-site requirement reuse through GRaph and ontology matching. Proceedings of the

12. http://swat.cse.lehigh.edu/projects/lubm/
13. http://wwwhome.cs.utwente.nl/ goknila/sosym/
14. http://www.illc.uva.nl/EuroWordNet/
15. `http://oaei.ontologymatching.org/2015/multifarm/index.html`

18th International Conference on Information Integration and Web-based Applications and Services (iiWAS), pp. 160-169, Singapore, November 28-30, 2016. ACM [46].

– Zouhir Djilani, Selma Khouri, Ladjel Bellatreche, Abderrahmane Khiat : Les Besoins Fonctionnels Candidats à l'Entreposage et l'Analyse en Ligne. A apparaître dans les actes des journées Francophones sur les Entrepôts de Données et l'Analyse (EDA), Mai 2017, Lyon, France [48].

Première partie

Background & État de l'art

Chapitre 2

Notions de base

Sommaire

1 Introduction 24

2 L'ingénierie des besoins 25

2.1 Définitions 25

2.2 Le processus de l'ingénierie des besoins 26

2.3 Approches d'expression des besoins 27

2.3.1 Approche orienté buts 28

2.3.2 Approche à base de scénarii 28

2.3.3 Approches basées sur les automates et réseaux de Petri 28

2.4 Techniques de modélisation des besoins 29

2.4.1 Définitions 30

2.4.2 Les techniques informelles (Langage naturel) 31

2.4.3 Les techniques semi-formelles 32

2.4.4 Les techniques formelles 32

3 Les solutions d'entreposage 32

3.1 Les ED : des systèmes d'intégration matérialisés 33

3.2 La modélisation multidimensionnelle 35

3.3 L'étape d'intégration de données (ETL) 36

3.4 Déploiement de l'ED 36

3.5 La place des besoins utilisateurs dans les solutions d'entreposage 38

4 Les ontologies pour la gestion des données et des besoins 39

4.1 Définitions 39

4.2 Taxonomie des ontologies 40

4.2.1 Classes d'ontologies selon le type des concepts 40

4.2.2 Classes d'ontologies selon l'objet de conceptualisation 41

4.3 Les formalismes ontologiques 42

4.4 Représentation formelle des ontologies 42
4.5 Les ontologies pour la gestion des besoins 43
4.6 Les ontologies dans les systèmes de gestion des données 44
4.6.1 Les bases de données à base ontologique (BDBO) . . . 44
4.6.2 Les ontologies dans le processus d'intégration 46
5 Le matching comme solution d'intégration d'ontologies 48
5.1 Définitions 49
5.1.1 Terminologie 49
5.1.2 Méthodes de comparaison et mesures de similarité . . 50
5.2 Les étapes du processus du matching 51
5.3 Classification des techniques de matching d'ontologies 52
5.3.1 Techniques terminologiques 53
5.3.2 Techniques structurelles 53
5.3.3 Les techniques extensionnelles 54
5.3.4 Les techniques sémantiques 54
6 Conclusion 55

Résumé. Dans ce chapitre, nous présentons les notions nécessaires à la compréhension de nos contributions. Ces notions sont issues du croisement du trois principaux domaines : l'ingénierie des besoins, l'ingénierie ontologiques et les systèmes d'entrepôts de données. Notre principale problématique relève de l'ingénierie des besoins, où nous avons identifié le manque d'un système décisionnel de type entrepôt pour l'intégration et l'analyse des besoins fonctionnels. Dans le domaine de l'ingénierie des besoins, nous nous sommes intéressés au processus de l'IB, ainsi qu'aux techniques d'expression et de modélisation des besoins. Dans le domaine des systèmes d'entreposage, nous nous sommes intéressés à leur modélisation ainsi qu'aux différentes étapes du cycle de vie de cette solution, couvrant les principales phases d'intégration, de conception et d'analyse des entités entreposées. Notre solution exploitent également les ontologies conceptuelles et linguistiques à différentes étapes clés. Dans le domaine de l'ingénierie ontologique, nous nous sommes intéressés aux capacités d'expressivité et de raisonnement des ontologies, ainsi que leur rôle dans la gestion des besoins, dans les systèmes de gestion de données (bases et entrepôts de données) et dans les systèmes d'intégration. Nous montrons qu'elles constituent un outil puissant au service de ces différents domaines.

1 Introduction

Les besoins fonctionnels et non fonctionnels représentent la première ressource pour la conception de toute application, logiciel, système, etc. L'Ingénierie des Besoins (IB) est le domaine qui s'intéresse à l'étude des problématiques associées aux besoins. Dans les systèmes de gestion des données comme les entrepôts de données, les besoins constituent également la première brique de conception de ces systèmes.

Les systèmes d'entreposage sont nés d'un besoin d'*intégrer* et d'*analyser* des quantités importantes de données afin d'en extraire de nouvelles connaissances. L'augmentation continue du volume des données ainsi que la mondialisation ont rapidement conduit au stockage décentralisé des données dans des bases de données disparates présentant des *ilots d'informations*. La solution la plus utilisée actuellement au niveau industriel et académique, pour représenter ces données, les stocker et fournir des analyses décisionnelles de type OLAP sur ces données, consiste en la solution des entrepôts de données.

La construction des solutions d'entreposage repose sur deux principales composantes : les sources de données et les besoins des utilisateurs. Les besoins des utilisateurs constituent une entrée importante qui a été valorisée dans l'ensemble des phases de conception et d'exploitation de l'entrepôt : pour sa modélisation par la proposition de diverses approches de conception hybrides (alliant données et besoins), pour son optimisation, pour son évolution et plus généralement pour la gestion de sa qualité [90].

Ces contributions nous confortent dans l'idée de considérer les besoins au même titre que les données, en les entreposant au sein de la solution finale. Cette solution permet de conserver un référentiel dans lequel l'ensemble de besoins est stocké et pourra être analysé et fouillé à des fins décisionnelles ou pour reproduire des anciennes expériences dans de nouveaux projets.

La solution d'entreposage des *données* étant une solution suffisamment mature, nous avons choisi d'adapter cette solution pour l'entreposage des *besoins fonctionnels*. Nous nous appuyons également sur les ontologies, que nous considérons comme un outil puissant d'expression et de raisonnement sur les besoins.

Ce chapitre présente les notions essentielles à l'établissement de notre proposition. Nous commençons par présenter quelques notions relatives à l'ingénierie des besoins. Plus précisément nous analyserons les principales techniques d'expression et de modélisation des besoins. Nous détaillons également les notions relatives aux entrepôts de données. Nous analysons ensuite le rôle des ontologies pour la gestion des besoins et celle des données. La dernière section conclut ce chapitre.

2 L'ingénierie des besoins

L'*Ingénierie des Besoins* regroupe toutes les approches et techniques visant à améliorer la phase de spécification d'un système informatique où les besoins utilisateurs sont définis. Nous présentons dans ce qui suit les concepts de base que nous empruntons de ce vaste domaine en décrivant la terminologie utilisée, le processus de l'IB, les approches d'expression et les techniques de modélisation de besoins.

2.1 Définitions

Besoin : un besoin est une description d'une propriété ou des propriétés du système qui doivent être accomplies [7]. Un besoin peut souvent appartenir à l'une des catégories suivantes [38] :

1. *Besoin fonctionnel* : spécifie une fonction que le système ou un composant du système doit accomplir et dépend donc de ce dernier (*fonctionnalités du système*). Nous nous intéressons à l'entreposage de cette catégorie de besoins qui permet de fournir une vision du système à construire.
2. *Besoin non fonctionnel* : est toute caractéristique (attribut/contrainte) relative à la *qualité du système*. Un vocabulaire commun pour évaluer la qualité d'un système est défini par la norme internationale ISO/CEI 9126 [16] qui distingue six caractéristiques (fonctionnalité, fiabilité, utilisabilité, portabilité, maintenabilité, efficacité) se décomposant en une vingtaine de sous-caractéristiques.

Dans la littérature, nous avons rencontré les deux termes ***Besoin*** et ***Exigence***, parfois utilisée de façon interchangeable. Une *exigence* est définie par l'association française d'ingénierie système AFIS [17] comme un énoncé qui traduit ou exprime un besoin ou des contraintes (techniques, coûts, délais, etc.). Nous considérons dans ce manuscrit le terme *besoin* pour désigner les besoins fonctionnels.

Parties prenantes (*stakeholders*) sont représentées par toute partie intéressée par l'utilisation et l'exploitation du système. il y a de nombreuses parties prenantes comme les ingénieurs des besoins, les clients, les experts du domaine, les utilisateurs, les commanditaires, les ingénieurs de maintenance, les ingénieurs d'application, la direction de l'entreprise, les fournisseurs, les formateurs, etc. Ces différentes parties prenantes ont des points de vue divergents, voire conflictuels [130]. Nous utilisons dans ce manuscrit le terme "*besoins des utilisateurs*" (BU) pour décrire les besoins de l'ensemble des parties prenantes.

L'Ingénierie des Besoins (IB) : selon [146], la toute première définition de l'IB est celle de *Ross et Schoman* qui notent que "requirements definition is a careful assessment of the needs

16. https://www.iso.org/fr/standard/22749.html
17. `https://www.afis.fr/Processus%20dIngnierie%20Systme/Bonnes%20pratiques%20pour%20l'%C3%A9laboration%20d'un%20r%C3%A9f%C3%A9rentiel%20d'exigences.pdf`

that a system is to fulfil. It must say why a system is needed, based on current and foreseen conditions, which may be internal operations or an external forces. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed » [132].

Une autre définition plus récente, mais qui rejoint la première définition, est donnée dans [129] où l'IB est définie dans le cadre des systèmes d'information comme étant *"l'activité qui transforme une idée floue en une spécification précise des besoins, souhaits, et exigences exprimés par une communauté d'utilisateurs, et donc définit la relation existante entre un système et son environnement"*.

2.2 Le processus de l'ingénierie des besoins

Selon *Nuseibeh* et al. [117], le processus de l'IB est composé d'un ensemble d'étapes comme illustré dans la figure 2.1) et détaillé ci-dessous.

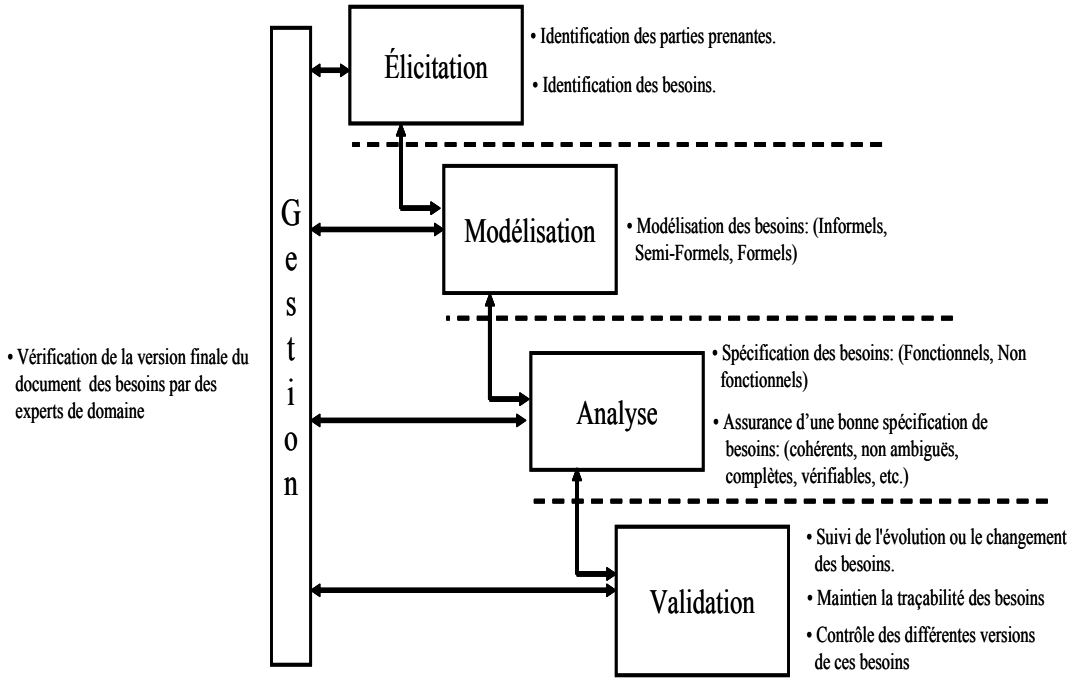

FIGURE 2.1 – Étapes du processus de l'ingénierie des besoins [27].

1. **Phase d'élicitation des besoins** : consiste à collecter et à développer les besoins des différentes sources/utilisateurs. Il existe diverses techniques d'élicitation des besoins allant des techniques traditionnelles aux techniques dirigées par les modèles en passant par les techniques collaboratives et le prototypage. Les informations collectées durant cette phase doivent souvent être interprétées, analysées, modélisées et validées afin d'avoir un ensemble suffisamment complet des besoins d'un système.

2. **Phase de modélisation des besoins** : consiste à spécifier des descriptions abstraites et interprétables du système. La modélisation des besoins se fait en utilisant un langage de modélisation. Différents types de langages peuvent être utilisés que nous détaillerons dans les points suivants (cf. section 2.4). Plusieurs techniques de modélisation des besoins peuvent être utilisées selon les informations que l'on cherche à modéliser comme les approches dirigées par l'organisation, dirigées par le domaine ou dirigées par les données.
3. **Phase d'analyse des besoins** : consiste à analyser les besoins élicités pour éliminer toutes les incomplétudes et incohérences, ambiguïtés, inadaptations aux besoins réels ou d'autres faiblesses de la spécification des besoins qui peuvent aboutir à des pertes de temps ou à la génération de nouvelles erreurs [130]. La norme *IEEE 830-1998* [18] recense les principales caractéristiques qu'une spécification des besoins doit vérifier : l'exactitude, la clarté, la complétude, la consistance, la stabilité, la vérifiabilité, la modifiabilité et la traçabilité.
4. **Phase de validation des besoins** : vise à vérifier la qualité des besoins et à s'assurer que les spécifications et modèles établis reflètent de façon exacte les attentes des parties prenantes. Dans cette phase, la gestion des conflits est une étape délicate qui consiste à trouver un accord entre les parties prenantes. Différentes techniques peuvent être utilisées comme le prototypage, l'animation de spécifications ou des techniques formelles. L'utilisation des ontologies est une approche ayant été proposée par de nombreux travaux [34].
5. **Phase de gestion des besoins** : consiste à gérer l'évolution des besoins tout en gardant la cohérence des autres besoins. Les évolutions courantes apportées aux spécifications des besoins incluent l'ajout ou la suppression des besoins et la correction des erreurs. Plusieurs techniques peuvent être utilisées pour la gestion des besoins comme le contrôle des versions et la gestion des liens de traçabilité entre les différentes parties du système.

Nous détaillons ci-dessous les approches d'expression de besoins ainsi que les langages de modélisation de besoins.

2.3 Approches d'expression des besoins

La littérature propose plusieurs approches pour éliciter et analyser les besoins. Chaque approche se fait selon l'angle d'étude du système. Dans ce qui suit, nous présentons brièvement les approches que nous considérons pour nos propositions :

1. Approche basée sur les scénarios
2. Approche dirigée par les buts
3. Approche basée sur les automates

18. http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

Notre choix pour ces trois approches s'explique par le fait d'avoir voulu des approches différentes montrant l'hétérogénéité des besoins au niveau de leur expression, et par conséquence, dans leur langage de modélisation comme nous allons le décrire dans la section qui suit.

2.3.1 Approche orienté buts

Un but est un objectif que le système en question doit atteindre [145]. L'approche orientée but a été proposée par Yu E. [165] affirmant que la modélisation explicite des buts dans le modèle des besoins peut satisfaire le critère de leur complétude. Quand les approches conventionnelles sont centrées sur le *Quoi*, les approches orientées but s'intéressent au *Pourquoi* du système [146]. Un but devient une exigence s'il est sous la responsabilité d'un seul acteur. Les buts sont formulés à différents niveaux d'abstraction, allant de buts stratégiques de haut niveau à des buts techniques de bas niveau. Les modèles de buts contiennent donc la décomposition d'un but en sous buts dans des graphes de réduction ET/OU [130].

Il existe différents modèles qui appliquent l'approche orientée buts, à l'instar de modèle GQM [154], les modèles du IStar framework [164, 165], KAOS [39], etc. Nous utilisons dans notre proposition le modèle orientée buts proposé dans [27] illustré en figure 2.2, qui se veut un modèle pivot des approches (KAOS, IStar, et Tropos) et qui repose sur le paradigme GQM (Goal/Question/Metric).

2.3.2 Approche à base de scénarii

Ces approches permettent de représenter les besoins par des scénarii où les événements sont des interactions entre l'utilisateur et le système. L'expression des scénarios peut se faire selon plusieurs modèles comme les diagrammes d'états, les diagrammes UML ou le langage naturel. Nous nous sommes intéressés aux diagrammes de cas d'utilisation d'UML. Ces derniers sont définis dans [79] comme *une description d'un ensemble de séquences d'actions, incluant des variantes, qu'un système effectue pour fournir un résultat observable et ayant une valeur pour un acteur*. Le diagramme des cas d'utilisation UML autorise l'établissement des relations entre les cas d'utilisation. Il offre deux principales relations : la généralisation/spécialisation, et les relations de dépendance (Inclusion, Extension).

La figure 2.3 présente le modèle de cas d'utilisation que nous utilisons dans notre proposition.

2.3.3 Approches basées sur les automates et réseaux de Petri

Plusieurs approches basées sur les automates et réseaux de Petri ont été proposées. Nous nous intéressons dans cette catégorie au modèle de traitements de la méthodologie Merise [128]. La méthode Merise permet de distinguer les traitements et les données. Les traitements peuvent

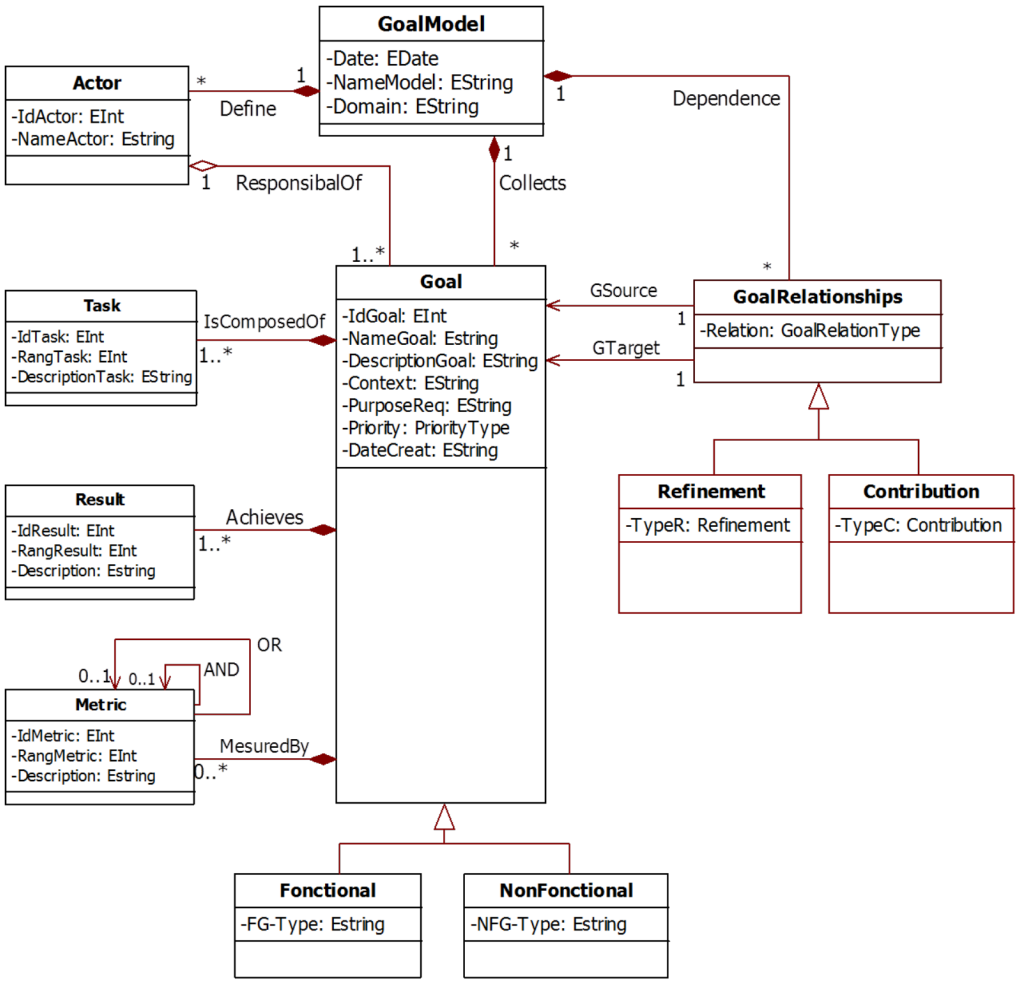

FIGURE 2.2 – Le modèle orienté but [27]

représenter de façon dynamique les besoins du système. Ces traitements sont représentés par le Modèle Conceptuel des Traitements (MCT) défini comme : *une succession de traitements déclenchés par des événements et qui donnent naissance à de nouveaux événements. Un traitement (opération) est une suite d'actions réalisées en vue d'obtenir un résultat* [122]. La figure 2.4 présente le modèle de traitement que nous utilisons dans nos propositions.

Dans ce qui suit, nous présentons les différentes techniques de modélisation des besoins et leurs langages associés.

2.4 Techniques de modélisation des besoins

Plusieurs techniques de modélisation des besoins ont été proposées dans la littérature et plusieurs d'entre eux sont mis en pratique dans l'industrie. Nous commençons par définir la no-

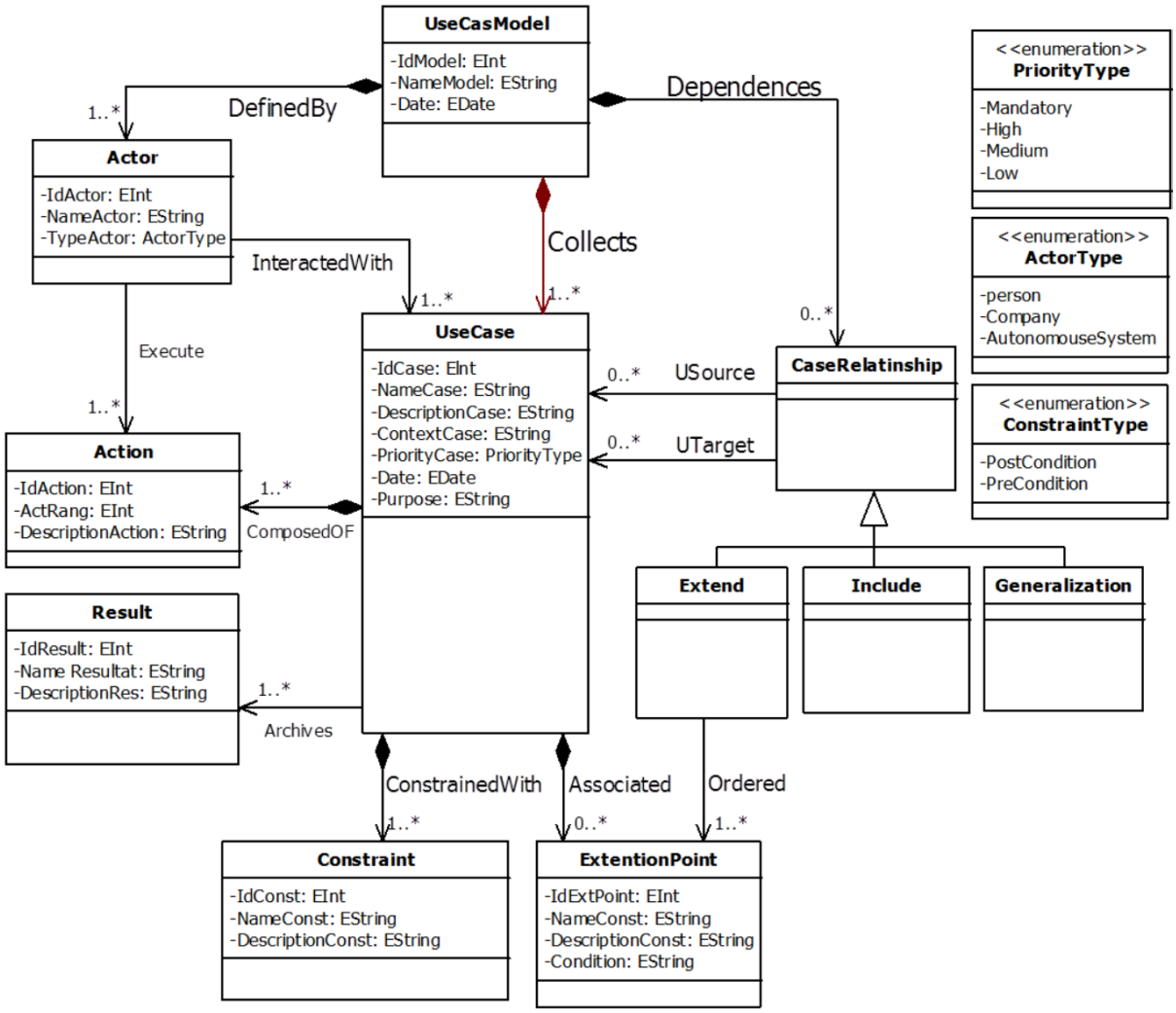

FIGURE 2.3 – Le modèle des cas d'utilisation [27].

tion de modèle, méta-modèle et langage de modélisation. Nous décrivons ensuite les principaux langages de modélisation utilisés pour la spécification des besoins.

2.4.1 Définitions

– Modèle : un modèle d'un système est une description ou spécification de ce système et de son environnement pour un certain usage. Un modèle est souvent présenté comme une combinaison de graphiques et de texte. Le texte peut être dans un langage de modélisation ou dans un langage naturel [19]. Dans l'IB, les modèles peuvent être utilisés pour capturer et formaliser les besoins des utilisateurs. Ils peuvent ainsi intégrer et vérifier les besoins [27].

19. http://www.omg.org/spec/

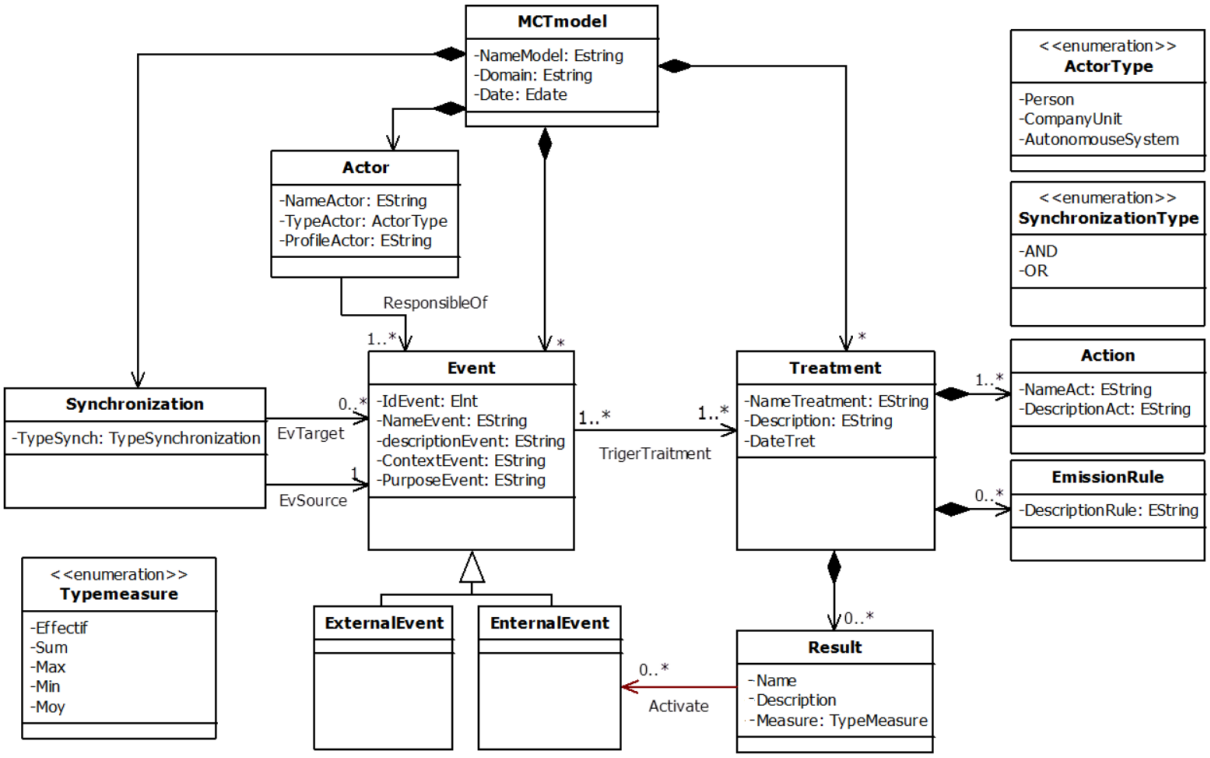

FIGURE 2.4 – Le modèle conceptuel de traitements [27].

– Méta-modèle : un méta-modèle définit la structure que doit avoir tout modèle conforme à ce méta-modèle. En d'autres termes, un méta-modèle est un modèle de modèles [20].
– Langage de modélisation : il sert à décrire un système général ou spécifique à un domaine et/ou un contexte par ses composants et leurs relations [27]. Il est défini par une syntaxe abstraite, une sémantique et une syntaxe concrète. Nous utilisons dans ce manuscrit les termes formalismes ou langages de modélisation de façon interchangeable. Chaque modèle de besoin peut être construit en utilisant un formalisme/langage de modélisation.

Nous décrivons dans ce qui suit les principaux langages de modélisations des besoins utilisés.

2.4.2 Les techniques informelles (Langage naturel)

Les techniques informelles se basent sur le langage naturel avec ou sans règles de structuration. Bien qu'elles ne requièrent aucune formation particulière et qu'elles garantissent la facilité de compréhension (due à la manière familière de communiquer) entre les parties prenantes du système, elles introduisent des risques d'ambiguïté, des malentendus et des incohérences car ni leur syntaxe, ni leur sémantique ne sont explicitement définies. Parmi ces techniques, nous

20. http://www.omg.org/spec/

pouvons citer : le questionnaire, l'interview et le cahier des charges.

2.4.3 Les techniques semi-formelles

Les techniques semi-formelles sont basées sur des notations graphiques avec une syntaxe précise. Ces techniques facilitent la communication entre les parties prenantes. Leur principal inconvénient est le manque de sémantique précise, laissant des risques d'ambiguïté et d'incohérence des besoins. Les trois modèles cités (orienté buts, cas d'utilisation UML et MCT) sont des exemples de modèles semi-formels. Ce sont les modèles que nous utilisons pour définir le modèle de l'entrepôt de besoins à définir (cf. chapitre 4). Le choix de ces modèles semi-formels est motivé par le fait d'avoir des langages fréquemment utilisés en industrie, qui constituent un intermédiaire entre les langages complètement formels et les langages orientés texte. Nous proposerons ainsi dans nos contributions, un modèle pivot qui factorisent ces trois formalismes utilisés par les sources de besoins.

2.4.4 Les techniques formelles

Les techniques formelles sont basées sur des notations mathématiques qui fournissent un cadre précis et non ambigu pour la modélisation des besoins à l'exemple de la méthode B [8]. L'avantage des ces techniques est de permettre la vérification des besoins. Leur inconvénient est qu'elles sont difficiles à manipuler par l'ensemble des parties prenantes qui ne maitrisent généralement pas ces formalismes.

Ce tour d'horizon des principales notions de l'IB nous a permis de poser les définitions nécessaires à nos propositions. Nous détaillons dans ce qui suit les solutions d'entreposage et nous analysons ensuite la place des besoins des utilisateurs dans ces solutions.

3 Les solutions d'entreposage

Les données sont au centre du traitement de plusieurs systèmes, appelés *systèmes de gestion des données*. Certains systèmes comme les bases de données (BD) s'intéressent aux transactions et aux opérations quotidiennes de l'entreprise en manipulant des données devant être immédiatement enregistrées avec précision dans le but d'être consultées (comme la gestion des ventes). Les BD permettent d'effectuer des traitements transactionnels de type OLTP (*OnLine Transaction Processing*). Les BD avancées comme les ED ou les BD analytiques se focalisent quant à elles sur les historiques des données et les métriques utilisées pour l'établissement des décisions tactiques ou stratégiques en utilisant des outils d'analyse. Les BD avancées requiert la définition d'une vue unifiée sur les données issues des différentes sources hétérogènes [53]. Ce processus peut se faire grâce aux *systèmes d'intégration* qui peuvent être virtuels ou matérialisés comme c'est le cas pour les ED.

Un ED est défini comme une "collection de données orientées sujet, intégrées, non volatiles et historisées, organisées pour supporter un processus d'aide à la décision" [78].

Le concept d'ED est utilisé dans divers domaines et applications notamment les projets industriels : services financiers, médecine, production, vente au détail, télécommunications, etc. Plusieurs projets industriels d'ED dans différents domaines ont été développés : *DWQ(Data Warehouse Quality)* [83], *Whips* [72], *OntoDawa* [18].

Edgar Codd [37] a proposé le concept de *traitement d'analyse en ligne* (OLAP) pour décrire les fonctionnalités de l'outil d'analyse *Essbase*, et depuis plusieurs nouveaux outils et règles OLAP ont suivi avec une nouvelle branche de l'informatique qualifiée de *décisionnelle* (business intelligence). Quelques notions relatives aux ED sont présentées ci-dessous.

3.1 Les ED : des systèmes d'intégration matérialisés

De nombreux travaux portant sur l'intégration ont été menés ces dernières décennies, où trois dimensions ont été mises en avant (voir figure 2.5) : **(i)** l'intégration des données (*Entreprise Information Integration (EII)*), **(ii)** les applications (*Entreprise Application Integration (EAI)*) et **(iii)** l'intégration des plateformes. Une véritable industrie autour des systèmes d'intégration a été créée [49, 71] profitant à plusieurs entreprises et industries. L'intégration des données a attiré le plus d'attention du fait de son importance [71]. La considération des besoins

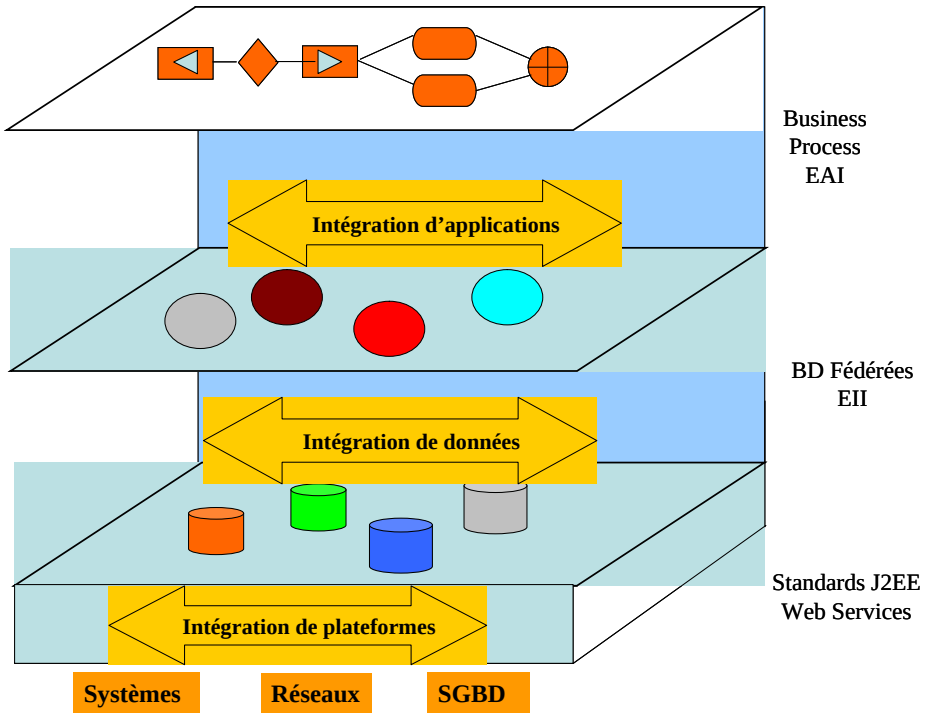

FIGURE 2.5 – Différents niveaux d'intégration [59].

utilisateurs dans l'intégration est quant à elle récente, complétant ainsi la figure précédente par une couche supérieure, celle de l'intégration des besoins. Nous la détaillerons dans le prochain chapitre (cf. chapitre 3, section 2.1).

Par définition, un système d'intégration consiste à fournir une interface *unique, uniforme* et *transparente* aux objets concernés par le processus d'intégration (données, des applications, des plateformes, etc) [18]. Un système d'intégration est formellement défini par le triplet $< G, S, M >$ où G représente le schéma global du système d'intégration, S représente l'ensemble des sources à intégrer et M décrit différentes assertions de mapping entre le schéma global et les sources [100].

Deux principales approches sont proposées pour la définition des mappings : Global as View (GaV) et Local as View (LaV) :

1. L'approche *GaV* : permet de construire le schéma global en fonction des schémas locaux par le biais d'un ensemble de relations. Plusieurs systèmes d'intégration de données utilisent l'approche *GaV*, tels que: *Whips* [72], *Tsimmis* [36] et *Momis* [22].
2. L'approche *LaV* : permet de construire le schéma des partenaires (sources à intégrer) sur la base du schéma global. Les relations des schémas des partenaires sources représentent des vues sur les relations du schéma global. Il existe plusieurs systèmes d'intégration des données qui utilisent l'approche *LaV* comme *Picsel* [63] et *InfoMaster* [60].

La difficulté lors de la mise en place d'un système d'intégration est de gérer les hétérogénéités syntaxiques et sémantiques entre les sources. Les conflits sémantiques sont généralement les plus difficiles à gérer. Cette gestion est effectuée lors de la définition des mappings. Goh et al. [64] ont distingué quatre types de conflits sémantiques définis dans le contexte de l'intégration des données, mais qui s'appliquent aussi bien dans le contexte de l'intégration des besoins.

1. *Conflit de représentation* survient lorsqu'un même objet conceptuel est modélisé différemment par les sources de données. Dans la pratique, la différence réside dans le nombre/type de classes et/ou de propriétés utilisées pour la modélisation.
2. *Conflits de noms* où deux types de conflits peuvent être distingués :
   (a) *Synonymie* lorsque des noms différents sont utilisés pour décrire la même notion (concept ou propriété), comme le concept *article* et *produit* qui désignent la même notion mais avec des termes différents.
   (b) *Homonymie* lorsque le même nom est utilisé pour des *concepts* différents, comme l'attribut *type* décrivant les types de *clients*, mais qui peut signifier client local ou étranger, ou alors masculin ou féminin.
3. *Conflits de contexte* où un même objet du monde réel peut être représenté dans les sources de données de plusieurs façons selon un contexte local correspondant à chaque source. Par exemple, un besoin désignant le prix d'un produit, TTC dans une source et hors taxe dans une autre source.

4. *Conflits de mesure* lorsque différentes unités de mesure sont utilisées pour mesurer le même concept, comme le *prix* qui peut être mesuré en *euros* ou bien en *dollars* selon les sources.

Outre la définition du schéma intégrant les différentes sources, le schéma cible de l'entrepôt doit être conçu, c'est le schéma G de <G,S,M>. Cette conception suit une modélisation multidimensionnelle que nous détaillons dans ce qui suit.

3.2 La modélisation multidimensionnelle

La *modélisation multidimensionnelle* popularisée par *Ralph Kimball* [93], est reconnue comme étant la plus appropriée aux besoins d'analyse et de prise de décision [9]. La modélisation multidimensionnelle est une technique de structuration et de visualisation des modèles de données dans le but de rendre leur analyse plus facile. Cette modélisation organise les données de façon à mettre en exergue le sujet analysé (le *fait*) et les différentes perspectives d'analyse (les *dimensions*), ce qui se rapproche de la pensée des analystes. Un modèle multidimensionnel est ainsi basé sur les deux concepts fondamentaux de *fait* et de *dimension* [93] :

1. Un *fait* représente le centre d'intérêt de l'entreprise ou de l'organisation autour duquel le processus de prise de décision doit tourner. Un fait est composé de certains attributs atomiques ou dérivés appelés *mesures*. Si l'on prend l'exemple du fait représentant les *Ventes* d'une entreprise, la quantité des produits vendus ou le chiffre d'affaire engendré sont des exemples de mesures.
2. Une *dimension* représente un point de vue (contexte) selon lequel le fait peut être analysé. Pour l'exemple du fait *Ventes*, celui-ci peut être analysé selon les dimensions *produit, temps* et *client*. En plus des attributs, les dimensions se composent d'un ensemble d'éléments organisés de façon *hiérarchique*. Par exemple, pour la dimension *produit*, la hiérarchie est composée de {type, catégorie}.

Le constructeur des modèles multidimensionnels est appelé un *cube* de données dont les cellules représentent les mesures du fait. La figure 2.6 illustre un exemple d'un cube *Vente* dont les dimensions sont *produit, temps* et *site*. Les cubes permettent plusieurs opérations OLAP appelées opérations de restructuration comme :

- L'opération *switch* : opération d'interchangement de la position des membres d'une dimension.
- L'opération *pivot* : opération de rotation du cube autour d'un des axes.
- L'opération *push* : opération de combinaison des membres d'une dimension aux mesures du cube.

D'autres opérations sont liées aux hiérarchies des dimensions permettant ainsi la hiérarchisation des données comme :

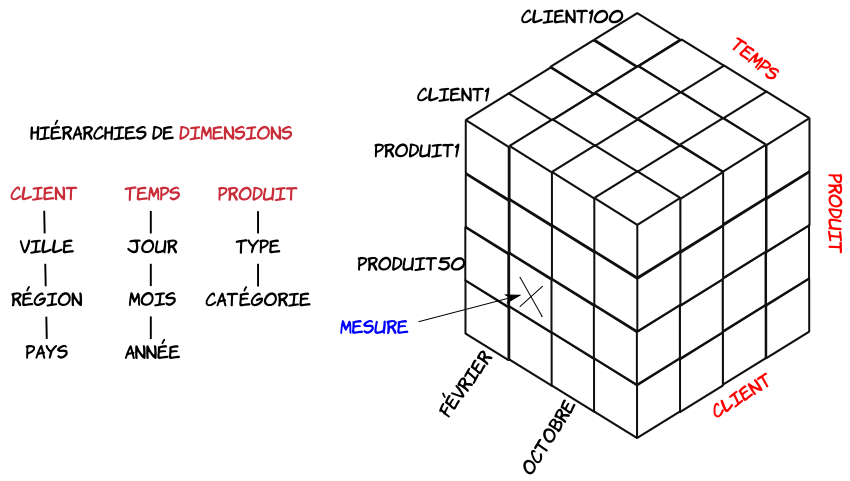

FIGURE 2.6 – Exemple d'un cube multidimensionnel des *Ventes*

– Opération *roll up* : opération de visualisation des données de manière résumée en allant d'un niveau particulier de la hiérarchie vers un niveau plus général.
– Opération *drill down* : opération de navigation d'un niveau particulier de la hiérarchie vers des données d'un niveau inférieur dans la hiérarchie.

3.3 L'étape d'intégration de données (ETL)

Nous avons présenté l'ED comme un système d'intégration défini par le triplet <G,S,M>. Une fois le schéma G de l'ED défini suivant une modélisation multidimensionnelle, les schémas des sources définis et les mapping entre G et S sont établis, un processus d'extraction-transformation-chargement (Extract-Trasform-Load) ETL est défini afin d'extraire les données des sources, de les transformer selon les mappings établis et de les charger pour peupler le schéma G. Il existe plusieurs outils ETL commerciaux et open source supportant ce processus comme *Talend open studio*, *IBM Data Warehouse Center* ou l'outil académique AJAX. La phase ETL est considérée comme la phase la plus consommatrice en temps car elle nécessite un nettoyage des données, afin d'assurer leur chargement dans l'entrepôt (principe du "garbage in, garbage out" pouvant même provoquer des erreurs lors de la prise de décision). La phase ETL est suivie par l'étape de la conception physique où sera implémenté et déployé le schéma de l'ED selon un SGBD choisi.

3.4 Déploiement de l'ED

Il existe trois principales représentations logiques et physiques pour les ED : *ROLAP*, *MOLAP* ou *HOLAP*.

L'approche *Multidimensional On-Line Analytical Processing* (MOLAP) implémente le cube

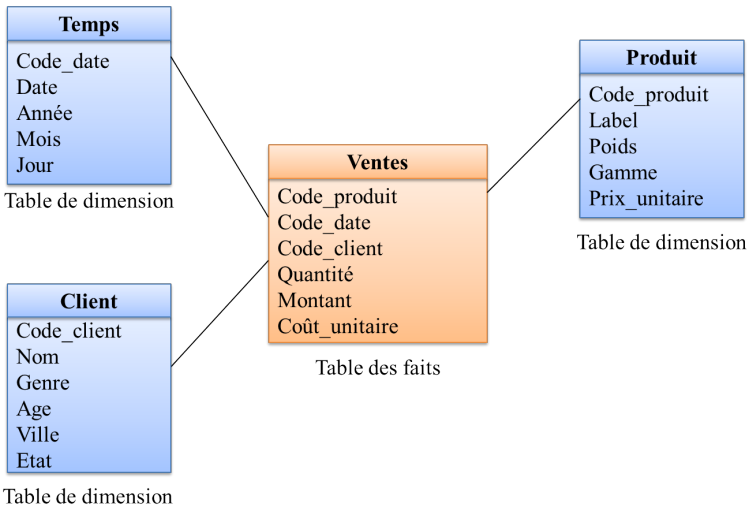

FIGURE 2.7 – Exemple d'une modélisation d'ED en étoile [11]

sous forme d'un tableau multidimensionnel. Cette implémentation a l'avantage d'accélérer le temps d'accès aux données mais elle s'avère être difficile à implémenter.

L'approche Relational On-Line Analytical Processing (ROLAP) implémente le modèle de l'ED selon une représentation relationnelle. Chaque fait est représenté par une *table de faits* et chaque dimension est représentée avec *une table de dimensions*. Les mesures sont stockées dans les tables de faits contenant les clés vers les tables de dimensions comme illustré par la figure 2.7. Les avantages du modèle relationnel en terme de gestion des données et de ses techniques d'optimisation fait que l'approche ROLAP, l'approche la plus utilisée [93]. Nous détaillons ci-dessous les trois types de schémas de modélisation utilisés pour l'approche ROLAP.

Le schéma en étoile : Ce schéma est caractérisé par le fait que les tables de dimension ne sont pas normalisées. La table de faits est normalisée et référence les tables de dimension par des clés étrangères (figure 2.7). Une table de dimension non normalisée rassemble des informations relatives aux hiérarchies, malgré la différence des propriétés de leurs niveaux. Cette représentation optimise le temps de traitement des plusieurs types de requêtes.

Le schéma en flocon de neige : Le schéma en flocon est le schéma relationnel de base, respectant les règles de normalisation, où les tables de dimension sont normalisées. Ces dernières sont décomposées en une ou plusieurs tables de dimensions de plusieurs niveaux formant la hiérarchie de la table de dimension originale sans toucher à la table de faits (figure 2.8). Ce type de schéma offre une meilleure visualisation et compréhension des données, mais peut créer de nouvelles jointures entre les tables de dimension et leurs sous-dimensions ce qui se traduit par un temps d'exécution plus lent.

Le schéma en constellation : Les schémas en constellation sont caractérisés par la présence de plusieurs tables de faits qui partagent des tables de dimensions communes.

L'approche *Hybrid On-Line Analytical Processing* (HOLAP) quant à elle, est la combinai-

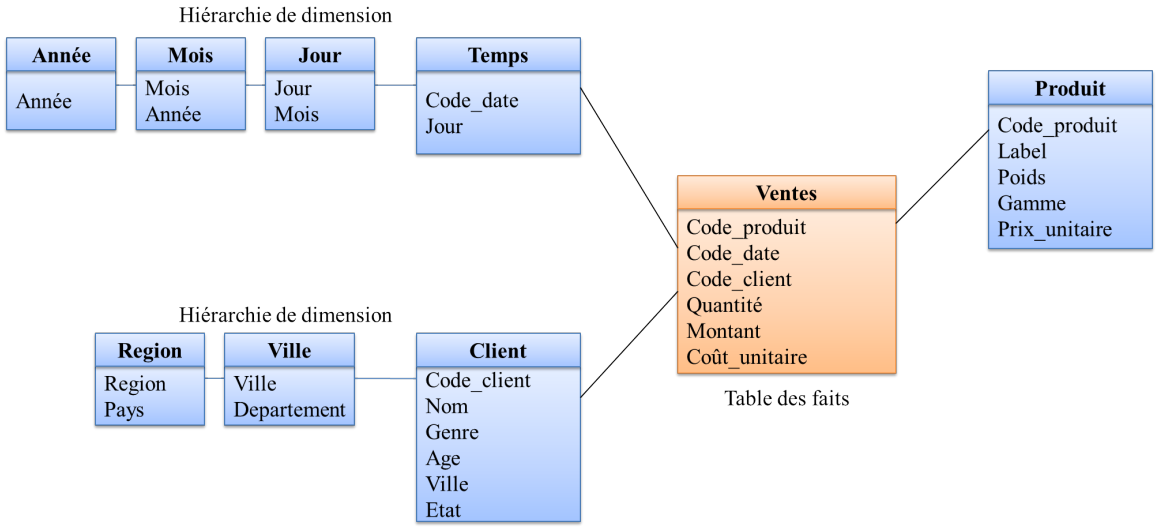

FIGURE 2.8 – Exemple d'une modélisation d'ED en flocon de neige [11]

son des approches ROLAP et MOLAP visant à bénéficier des avantages de chacune.

3.5 La place des besoins utilisateurs dans les solutions d'entreposage

Historiquement, la première génération des projets d'entreposage ne considérait que les phases de modélisation logique, ETL et la modélisation physique. Ce processus de conception a été complété par la modélisation conceptuelle [67] et la phase de définition des besoins où les besoins sont traités et analysés [90]. Plusieurs approches de conception *orientées besoins*, appelées aussi approches *descendantes*, ont ainsi émergé visant à fournir une vision de conception de l'ED à partir des besoins des utilisateurs [93, 62, 66]. D'autres approches dites *hybrides* proposent de considérer les deux sources d'informations (données et besoins) pour la conception de l'entrepôt [108]. Certaines études ont montré que la prise en compte des besoins détermine le succès ou l'échec du projet d'entreposage [127].

Les besoins des utilisateurs ont ainsi été valorisés dans le domaine des ED. Leur exploitation permet dans un premier temps de générer le schéma multidimensionnel de l'ED répondant aux exigences des décideurs et utilisateurs. L'exploitation des besoins a rapidement dépassé les premières étapes de conception pour se généraliser à l'ensemble des phases de construction de l'ED. Ainsi, l'utilisation des besoins se fait également sentir à la phase de modélisation physique pour différentes tâches d'optimisation. Plusieurs SGBD actuels (comme Oracle ou SQL Server) stockent leurs requêtes (issus des besoins) afin de les exploiter lors du processus d'optimisation. D'autres travaux reposent principalement sur les caractéristiques de la charge de requêtes exprimées par les utilisateurs (disponibles à partir des besoins) pour définir des structures d'optimisation de l'ED [94]. Dans nos travaux au laboratoire, nous avons développé des approches de prédiction de plusieurs structures d'optimisation dès les premières phases

de conception de l'entrepôt en utilisant l'ensemble des besoins des utilisateurs [90, 82]. Cette prédiction permet ensuite de faire des choix adaptés de SGBD et de plateformes de déploiement.

Les besoins sont également utilisés pour la personnalisation de l'ED. Ils peuvent exprimer des préférences des utilisateurs, notion permettant de filtrer le flux d'informations [19].

Les besoins sont enfin utilisés pour la gestion de la qualité de l'ED, qui commence par la compréhension des besoins des utilisateurs [117]. La qualité du système d'ED de manière générale se mesure en fonction de sa capacité à répondre aux buts et objectifs fixés [148].

Les deux premières parties de ce chapitre ont permis de synthétiser les notions relatives au processus de l'IB et aux systèmes d'entreposage. Pour faciliter la gestion des données dans les systèmes d'entrepôts et pour faciliter également la gestion des besoins dans le processus de l'IB, plusieurs travaux ont fait appel aux ontologies. Nous analysons dans ce qui suit les principales notions relatives aux ontologies et leurs diverses utilisations.

4 Les ontologies pour la gestion des données et des besoins

Nous avons assisté dernièrement au développement continu de plusieurs ontologies dans de nombreux domaines comme la médecine, les composants industriels ou l'avionique. Le succès des ontologies peut s'expliquer par leurs capacités descriptives et leurs capacités de raisonnement. Nous nous intéressons dans cette étude aux ontologies. Nous présentons dans ce qui suit les principales notions relatives aux ontologies.

4.1 Définitions

Les ontologies ont été définies par Gruber [69] comme "*une spécification explicite d'une conceptualisation*". Cette définition a été enrichie par [81] décrivant une ontologie comme une représentation formelle, explicite, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine, en termes de classes d'appartenance et de propriétés caractéristiques.

Cette définition met en avant trois caractéristiques qui distinguent une ontologie de domaine des autres modèles informatiques (tels que les modèles conceptuels), qui sont :

1. Formelle : exprimée dans un langage de syntaxe et de sémantique formalisées permettant ainsi des raisonnements automatiques.
2. Référençable : toute entité ou relation décrite dans l'ontologie peut être directement référencée par un identifiant unique, à partir de n'importe quel contexte, afin d'expliciter la sémantique de l'élément référençant.
3. Consensuelle : admise par l'ensemble des membres (et des systèmes) d'une communauté.

Chaque ontologie, quel que soit son type, est composée des éléments suivants inspirés par

les langages de modélisation : les *concepts* d'un domaine modélisés par les *classes* et leurs *attributs*, les *relations* entre ces concepts, et les *axiomes*. Ces derniers désignent les assertions acceptées comme vraies dans le domaine étudié. Les axiomes et les règles de raisonnement permettent de vérifier la cohérence d'une ontologie et d'inférer de nouvelles connaissances.

4.2 Taxonomie des ontologies

Les ontologies peuvent être classées selon le type des concepts ou selon leur objet de conceptualisation.

4.2.1 Classes d'ontologies selon le type des concepts

L'usage des ontologies diffère d'une communauté à une autre. Dans le domaine de la *linguistique*, les ontologies manipulent des mots et les relations entre ces mots telles que la synonymie et l'antonymie. Ces ontologies sont alors dites ***ontologies linguistiques (OL)***. *Wordnet* [21] est l'une des OL les plus connues.

Dans les autres communautés, on manipule plutôt des concepts et non des mots, ce qui a fait naître les ontologies dites *conceptuelles*. Gruber [69] distingue deux types de concepts dans une ontologie conceptuelle :

1. Concepts primitifs (canoniques) : qui représentent des concepts ne pouvant être définis par une définition axiomatique complète.
2. Concepts définis (non canoniques) : qui sont définis à base d'autres concepts canoniques ou non canoniques.

Dans le domaine des BD par exemple, on s'intéresse généralement aux concepts primitifs possédant une représentation unique afin d'exclure les redondances. Les ontologies ne contenant que des concepts primitifs sont appelées ***ontologies conceptuelles canoniques (OCC)***. Le modèle PLIB permet de définir des *OCC* [121].

Dans le domaine de *l'intelligence artificielle*, par exemple, et afin de pouvoir faire des déductions, on manipule plutôt des concepts définis. Les ontologies contenant des concepts primitifs et définis sont appelées ***ontologies conceptuelles non canoniques (OCNC)*** [121]. Les langages ontologiques issus du Web Sémantique comme le standard OWL [22] permettent de définir des *OCNC*. Ces langages permettent de raisonner sur l'ontologie pour inférer de nouveaux faits. Plusieurs mécanismes de raisonnements peuvent être utilisés comme la vérification des relations de subsomption, l'identification des classes équivalentes, la vérification de la satisfiabilité des concepts et la vérification de la consistance de l'ontologie et de ses instances.

21. http://wordnet.princeton.edu/
22. https://www.w3.org/OWL/

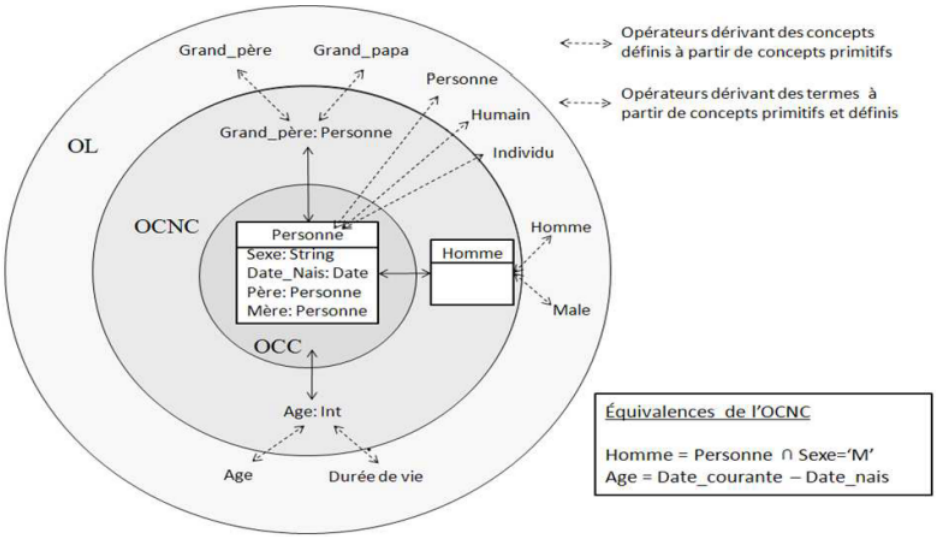

FIGURE 2.9 – Modèle en oignon [56]

Cet usage différent des ontologies a été matérialisé dans une taxonomie en couches proposée dans [121] illustrée par le *modèle en oignon* dans la figure 2.9.

Les relations entre les deux couches conceptuelle et linguistique sont définies dans ce modèle, où à chaque concept de la couche conceptuelle, on peut correspondre plusieurs termes de la couche linguistique (dans la même langue ou dans différentes langues).

4.2.2 Classes d'ontologies selon l'objet de conceptualisation

Cette classe concerne les ontologies conceptuelles. Les ontologies peuvent être classés selon l'objet de conceptualisation en plusieurs types [124]: ontologie de *représentation des connaissances*, ontologie *supérieure ou de haut niveau*, ontologie *générique*, ontologie de *domaine*, ontologie de *tâches* et ontologie d'*application*. Nous nous intéressons dans nos propositions aux *ontologies de domaine* et aux *ontologies de tâches*, qui sont définies comme suit :

1. Les ontologies de domaine sont définies comme des ontologies qui décrivent le vocabulaire correspondant à un domaine particulier, en spécialisant par exemple les concepts d'une ontologie de haut niveau. L'ontologie du domaine caractérise la connaissance du domaine où la tâche est réalisée [110].

2. Les ontologies de tâches sont utilisées pour décrire des tâches spécifiques à la résolution des problèmes des systèmes, telles que les tâches de diagnostic, de planification, de conception, etc. L'ontologie des objectifs d'apprentissage - *Learning Goal Ontology* [77] est un exemple d'ontologie de tâches dans le domaine de l'éducation.

4.3 Les formalismes ontologiques

La représentation d'une ontologie requiert un langage de définition (formalisme) qui offre les primitives nécessaires pour exprimer ses entités et relations. Plusieurs formalismes ont été proposés. Certains visent la description d'ontologies en vue de la gestion et l'échange des données comme RDF-Schema [141] (issu du Web sémantique) et PLIB (Parts LIBrary) [2, 3] (proposé pour la description des composants industriels). Ces formalismes cherchent à définir la sémantique des concepts de manière unique et utilisent donc des concepts canoniques.

D'autres formalismes ont été proposés pour la description d'ontologies permettant des déductions et des inférences, comme Daml, OIL, Daml+OIL et OWL. Ces formalismes sont issus du web sémantique et permettent la définition de concepts canoniques et non canoniques. OWL [23] est actuellement considéré par le consortium W3C [24] comme le standard de référence pour la définition des ontologies. Trois principales familles du langage OWL existent : OWL-Lite, OWL-DL et OWL-Full. OWL-DL est défini comme une extension d'OWL Lite, et OWL Full comme une extension d'OWL DL. Ces trois familles de langages établissent un compromis entre leur pouvoir expressif et leur pouvoir de raisonnement, dans le sens où plus le langage est expressif, moins il assure ses capacités de raisonnement.

4.4 Représentation formelle des ontologies

Une ontologie conceptuelle $\mathcal{OC}$ est définie formellement par un quintuplet suivant [90]. Cette formalisation est basée sur le formalisme des *logiques de description* (LD), qui est le formalisme à la base de la spécification du langage ontologique OWL. Ce formalisme (LD) est également considéré comme générique pour les modèles conceptuels conventionnels comme le modèle E/A ou le digramme de classes UML [33].

$$\mathcal{OC} :< C, R, Ref(C), Ref(R), F >, \text{ tel que :}$$

1. C : représente les *concepts* (classes) qui définissent le modèle ontologique
2. R : *roles* représente les propriétés des classes et les relations entre les concepts
3. $Ref(C) : C \rightarrow (Operator, Exp(C,R))$, est une fonction qui définit les classes qui composent l'ontologie.
   (a) $Operators$ peut être l'inclusion ($\subseteq$) ou l'équivalence ($\equiv$).
   (b) $Exp(C,R)$ est une expression composée de *concepts* et de *roles* de l'ontologie.
4. $Ref(R) : R \rightarrow (Operator, Exp(C,R))$, défini de la même façon que pour les concepts, les rôles de l'ontologie

23. https://www.w3.org/TR/owl-features/
24. https://www.w3.org/

5. F : c'est le formalisme ontologique utilisé comme *RDF*, *OWL*, *PLIB*, etc.

Analysant les travaux sur les ontologies lingusitiques [151], nous définissons une ontologie linguistique $\mathcal{OL}$ comme suit :

$$\mathcal{OL} :< T, Rel(T), C, RefC(C) >, \text{ tel que :}$$

1. T: représente l'ensemble des termes définis dans l'ontologie linguistique.
2. Rel(T): $T \rightarrow (Relation, 2^T)$ c'est la fonction qui représente les relations entre les termes (*synonyme, antonyme*, etc.).
3. C: représente les concepts ontologiques référencés par l'ontologie linguistique.
4. RefC(C): $C \rightarrow 2^T$ représente les relations entre les termes et les concepts ontologiques correspondants. Ce lien est explicité dans le modèle en oignon illustré en figure 2.9.

Après avoir parcouru l'essentiel des notions relatives aux ontologies, nous analysons dans ce qui suit l'utilisation des ontologies dans le domaine de l'IB ensuite dans les systèmes de gestion des données.

4.5 Les ontologies pour la gestion des besoins

Compte tenu de leurs caractéristiques, le domaine d'IB comme beaucoup d'autres domaines, a exploité les ontologies. Dans le cas d'une grande entreprise impliquant un nombre important de concepteurs hétérogènes, où les problématiques liées aux besoins sont plus difficiles à gérer, l'utilisation des ontologies s'avère très utile. Selon [34], les utilisations potentielles des ontologies dans le processus de l'IB comprennent : (i) la représentation du modèle des besoins en permettant notamment sa structuration, (ii) les structures d'acquisition des connaissances du domaine qui facilitent la compréhension des besoins et (iii) la connaissance du domaine d'application par les parties prenantes.

L'utilisation des ontologies a été projetée sur la plupart des phases du processus de l'IB détaillé précédemment. Durant la phase d'élicitation, l'ontologie permet d'assister le concepteur pour identifier les besoins des différentes parties prenantes. Dans la phase de modélisation, l'ontologie fournit une représentation formelle offrant des mécanismes suffisamment expressifs pour représenter le modèle des besoins. L'aspect formel des ontologies permet de modéliser explicitement les besoins des utilisateurs d'une manière automatique et interprétable par la machine. L'utilisation d'une ontologie dans la phase d'analyse des besoins contribue à assurer une spécification correcte et vérifiable des besoins, notamment en exploitant leurs capacités de raisonnement [27].

Une autre caractéristique des ontologies utilisée dans un processus de l'IB consiste en la *modularité ontologique*. La modularité ontologique est considérée comme une technique de

raisonnement. Les ontologies de domaine considèrent différents aspects d'un domaine à plusieurs niveaux de granularité [84]. Le modèle des besoins peut s'appuyer sur les différents modules extraits à partir d'une ou de plusieurs ontologies de domaine. De plus, la définition des modules rend le raisonnement plus facile car il sera appliqué sur des ontologies de plus petite taille [163]. La définition d'un module ontologique passe d'abord par la spécification d'un ensemble de termes à extraire appelé une signature. Cette signature correspond à l'ensemble des besoins dans notre contexte. Un module ontologique correspondra aux termes spécifiés ainsi qu'à leurs axiomes pertinents. Il existe plusieurs méthodes et formalismes de modularité d'ontologies [163] à l'exemple des logiques de description distribuées, E-connections, la segmentation d'ontologies, etc. Certaines méthodes sont supportées par des outils ou des plugins disponibles avec les éditeurs ontologiques comme Protégé, que nous utiliserons dans nos propositions.

4.6 Les ontologies dans les systèmes de gestion des données

Les systèmes de gestion des données (BD et ED) ont également trouvé intérêt dans l'utilisation des ontologies. Nous détaillons le rôle des ontologies pour les aspects suivants que nous utilisons dans nos propositions : les BD ontologiques, l'intégration des données et le matching ontologique.

4.6.1 Les bases de données à base ontologique (BDBO)

Le volume croissant des données référençant des ontologies et le besoin de systèmes susceptibles de gérer efficacement ces données, a donné naissance à la notion de base de données à base ontologique (BDBO). Une BDBO est une source de données qui, en plus des instances (données) contient l'ontologie qui en définit la sémantique, ce qui lui donne les caractéristiques suivantes [56] :

1. les données ainsi que l'ontologie définissant leur sémantique sont stockées au sein de la base de données, sur lesquelles les mêmes traitements peuvent être effectués (interrogation, insertion, suppression).
2. Chaque donnée est associée à son référent ontologique et inversement chaque concept ontologique est associé aux données qui lui correspondent.
3. L'ontologie locale à la base de données peut faire référence à une ontologie externe.

Différentes BDBO ont ainsi été proposées dans le monde académique comme OntoDB [25] et SESAME [26] et dans le monde industriel comme Oracle [27] et IBM SOR[107]. Les BDBO proposées diffèrent dans trois principales caractéristiques : les formalismes ontologiques utilisés,

25. https://forge.lias-lab.fr/projects/ontodb/wiki
26. https://www.w3.org/2001/sw/wiki/Sesame
27. http://docs.oracle.com/cd/B28359_01/appdev.111/b28397/sdo_rdf_concepts.htm

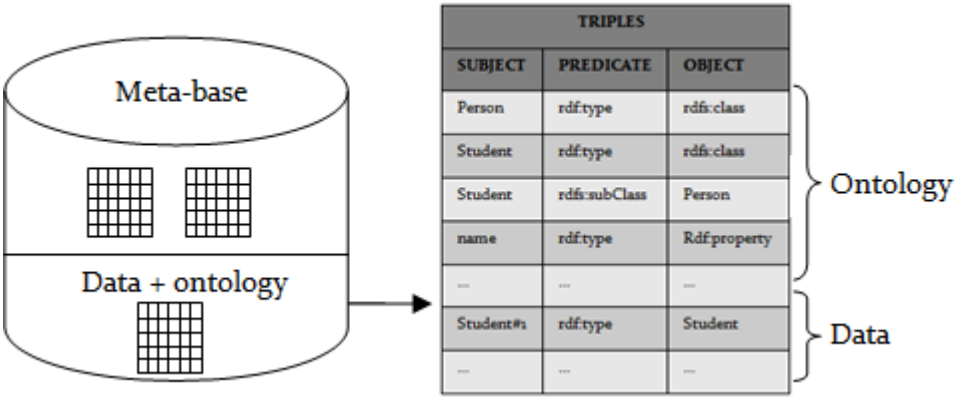

FIGURE 2.10 – Architecture BDBO type I

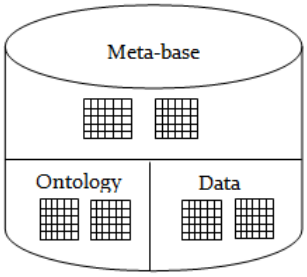

FIGURE 2.11 – Architecture BDBO type II

la représentation des ontologies et la représentation des instances ontologiques dans la base de données.

Les formalismes. Les formalismes des BDBO dépendent des formalismes des ontologies stockées. Par exemple, la BDBO OntoDB permet le stockage d'une ontologie décrite en PLIB, et la BDBO Oracle permet le stockage d'une ontologie décrite en OWL.

Représentation des ontologies. Contrairement aux bases de données traditionnelles où le modèle logique est stocké selon une approche relationnelle, dans une BDBO, deux niveaux de modélisation sont stockés : le niveau d'ontologie, et le niveau des instances ontologiques qui peuvent être stockés conjointement utilisant le même format de stockage ou séparément utilisant des formats différents. Ce stockage donne lieu à trois architectures de BDBO [56] :

1. Architecture de type I : qui consiste à utiliser un seul schéma pour représenter à la fois les concepts (classes, propriétés) de l'ontologie ainsi que les instances ontologiques (voir figure 2.10). La BDBO Jena2 [158] suit cette architecture.
2. Architecture de type II : l'ontologie et les instances sont stockées dans deux schémas différents (figure 2.11). La BDBO IBM SOR [107] suit cette architecture.
3. Architecture de type III : consiste à stocker, en plus de modèle ontologique et des instances ontologiques, le méta-modèle du formalisme d'ontologie utilisé (figure 2.12). L'on-

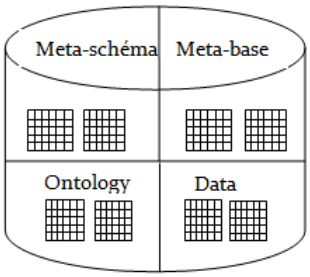

FIGURE 2.12 – Architecture BDBO type III

tologie constitue donc une instance de ce métaschéma. L'ontologie OntoDB [56] suit cette architecture.

Représentation des données à base ontologique. De même, trois principales approches *relationnelles* ont été définies pour la représentation des instances ontologiques [56] :

1. Approche verticale : souvent utilisée lorsque l'architecture est de type 1, consiste à représenter les instances de classes et des propriétés sous forme de triplets suivant le formalisme RDF (sujet, prédicat, objet).
2. Approche binaire : utilise des tables à une seule colonne pour stocker les instances de classes, et des tables à deux colonnes pour représenter les valeurs de propriétés des instances
3. Approche horizontale : est similaire à la représentation traditionnelle utilisée par les SGBD relationnels. Cette approche associe à chaque classe ontologique une table ayant une colonne pour chaque propriété qui est associée à une valeur.

4.6.2 Les ontologies dans le processus d'intégration

Les ontologies ont intervenu dans les systèmes d'intégration, matérialisés comme les ED et virtuels, principalement pour résoudre les problèmes d'intégration dûs aux différents conflits syntaxiques et sémantiques (cf. section 3). Leur usage dans les systèmes de gestion des données a par la suite été étendu aux autres tâches de conception comme la modélisation conceptuelle, la modélisation multidimensionnelle, le processus ETL, l'optimisation, etc [90].

L'introduction des ontologies dans les systèmes d'intégration a permis la gestion efficace des différents conflits de façon automatique. Les premières approches d'intégration proposaient de gérer manuellement les conflits sémantiques, et exigeaient donc l'intervention d'un expert humain. Les conflits syntaxiques sont traités de façon automatique. Plusieurs systèmes d'intégration des données ont suivi cette approche : le système *Tsimmis* [36], la fédération des bases de données et les systèmes multi-bases de données [18]. Deux limites majeures sont à mentionner pour ces approches : (1) la difficulté de l'évolution rapide des sources, et (2) la difficulté de considérer un nombre important des sources de données.

Les approches *semi-automatiques* permettent l'automatisation partielle des conflits sémantiques. Ces travaux se basent sur les ontologies linguistiques uniquement. Ces dernièrs traitent des termes, et non pas des concepts, ce qui peut générer des conflits de noms. Le système *Momis* [22] utilisant l'ontologie linguistique *WordNet* [28] suit cette approche.

Les approches *automatiques* se caractérisent par l'incorporation des *ontologies conceptuelles*, qui permettent le traitement automatique des conflits sémantiques des données sources [18]. Plusieurs projets utilisent cette approche comme *Buster* [150], *Picsel* [63] ou *SHOE* [75].

Dans [155] trois principales structures basées sur les ontologies conceptuelles ont été définies : (1) structure à base d'une ontologie unique, (2) structure à base d'ontologies multiples et, (3) structure à base d'une ontologie partagée. Ces structures sont classifiées selon la façon dont les ontologies conceptuelles sont connectées aux sources. Nous les détaillons dans ce qui suit.

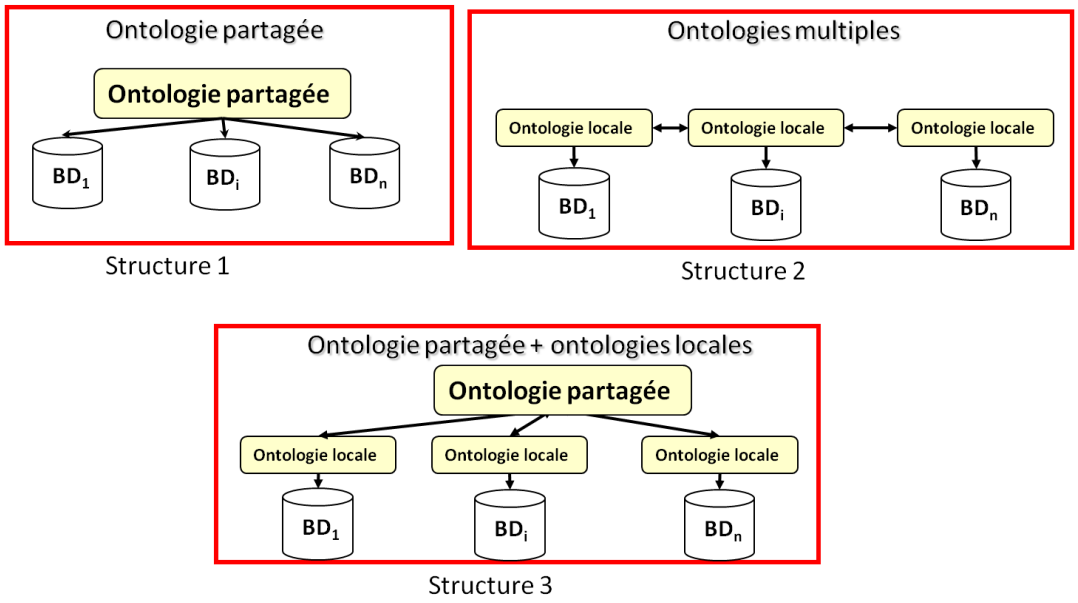

FIGURE 2.13 – Les structures d'ontologies conceptuelles dans les systèmes d'intégration [90]

4.6.2.1 - Structure basée sur une ontologie unique : correspond à une structure d'intégration à base d'une ontologie *unique*, comme les systèmes *Picsel* [63] et *COIN* [64]. Le développement de cette structure est conditionné par la définition de sources de données du même domaine et partageant un vocabulaire commun. Cette structure comporte de nombreux inconvénients : (i) l'ajout d'une nouvelle source peut nécessiter de redéfinir l'ontologie; (ii) cette structure ne permet pas une autonomie schématique des sources.

4.6.2.2 - Structure basée sur des ontologies multiples : dans cette structure, la sémantique de chaque source est définie par une ontologie locale. La mise en correspondance des différentes

28. https://wordnet.princeton.edu/

ontologies sources (locales) est effectuée deux par deux. Le principal inconvénient de cette approche est la complexité de définition des mappings. Les approches qui utilisent cette structure présentent une complexité de mappings : pour N sources, la complexité de mappings est de N(N-1)/2. Leur avantage est qu'elles offrent une grande autonomie à chaque source participant au système d'intégration.

4.6.2.3 - Structure basée sur une ontologie partagée : Dans cette structure, chaque source contient sa propre ontologie locale. Afin de réduire le nombre de mappings, les ontologies locales sont mises en correspondance avec une ontologie partagée. Ces approches supposent donc l'existence d'une ontologie partagée entre les sources. L'alignement entre les ontologies locales et l'ontologie partagée peut se faire a priori ou a posteriori [18]. Pour les approches *a posteriori*, chaque source définit de façon autonome sa propre ontologie. Des algorithmes de découverte de mappings (matching ontologique) sont utilisés afin d'aligner les entités des différentes ontologies locales et partagée. Dans le cas des approches *à priori*, les ontologies locales des sources sont définies à partir de l'ontologie partagée.

Nous étudierons dans nos propositions les deux dernières structures ontologiques, que nous considérons comme les plus réalistes pour notre contexte, et nous proposerons des solutions pour chaque scénario.

Dans ces structures d'intégration que nous venons de détailler, les sources d'informations sont définies par une ontologie locale. Certaines sources stockent leurs ontologies locales, ce qui donne lieu à une "base de données à base ontologique".

Les structures ontologiques présentées reposent sur la définition des mappings ontologiques entre les ontologies. Cette tâche fait appel au domaine du matching ontologique. Vu l'importance de cette tâche, nous lui réservons une nouvelle section pour sa présentation.

5 Le matching comme solution d'intégration d'ontologies

Les ontologies ont été introduites pour réduire les hétérogénéités syntaxiques et sémantiques, mais elle ont paradaxalement introduit une forme d'hétérogénéité à un plus haut niveau [116]. Cette hétérogénéité est dûe à la diversité des visions des concepteurs d'ontologies ainsi que la variation de la couverture du domaine [44].

La notion de *matching* (ou *appariement*) d'ontologies est apparue pour résoudre les hétérogénéités entre différentes ontologies en les alignant via un ensemble de correspondances sémantiques définies entre les entités de ces ontologies [54] et ceci dans le but d'assurer leur interopérabilité sémantique. Plusieurs domaines utilisent le matching ontologique comme le web sémantique, la communication entre agents, la composition des services web et les systèmes pair-à-pair (P2P) [54]. Pour notre étude, la notion de matching ontologique nous intéresse dans le cas des structures d'intégration à base ontologiques étudiées précédemment (cf. section

4.6.2), où les ontologies locales à chaque source sont conçues indépendamment et doivent être alignées.

Plusieurs méthodes d'alignement d'ontologies automatiques ou semi-automatiques ont été élaborées. Elles peuvent être de type terminologique, structurel, sémantique et extensionnel. Certaines méthodes combinent plusieurs types d'alignement [54].

Nous commençons par définir le processus de matching et de matching ontologique pour détailler par la suite les catégories de techniques de matching existantes.

5.1 Définitions

Nous définissons dans ce qui suit la terminologie essentielle relative au matching [54] :

5.1.1 Terminologie

Le processus de *matching* consiste à identifier des relations entre deux ou plusieurs présentations, il peut s'agir par exemple d'ontologies, de fichiers XML ou de schémas de bases de données. Le matching ontologique est le processus d'identification des correspondances entre les entités de différentes ontologies.

L'*alignement* est la sortie du processus de matching, c'est l'ensemble de correspondances ou mappings entre deux ou plusieurs ontologies. Les *mappings* sont des relations établies entre les entités des différentes ontologies.

Un *matcher* est une combinaison ou plusieurs fonctions utilisées pour calculer la similarité ou la dissimilarité entre deux entités.

Exemple. L'exemple ci-dessous illustre un processus de matching impliquant deux ontologies à aligner [54]. La première ontologie est représentée dans la partie gauche de la figure 2.14 avec comme concepts *Product*, *book*, *DVD*, etc. La deuxième ontologie est représentée de la même façon sur la partie droite de la figure 2.14 avec comme concepts *volume*, *Essay*, *Literature*, etc. Les correspondances découvertes par le matching sont présentées par des flèches. Ces correspondances représentent différentes relations sémantiques qui sont : l'équivalence($\equiv$), plus spécifique($\sqsupseteq$), plus général ($\sqsubseteq$), disjonction($\perp$). Une mesure de confiance de la correspondance peut également être fournie par le système de matching. Cette mesure est une valeur appartenant à l'intervalle [0,1]. Sur l'exemple de la figure 2.14, les flèches en gras désignent la relation d'*équivalence* tandis que le reste des flèches désignent les relations *subsume* ou *subsumé par* avec les valeurs de confiance correspondantes. L'alignement est quant à lui la sortie de ce processus de matching. Dans l'exemple, l'alignement représente l'ensemble des correspondances sémantiques extraites grâce au processus du matching dont une partie est représentée dans la figure 2.15.

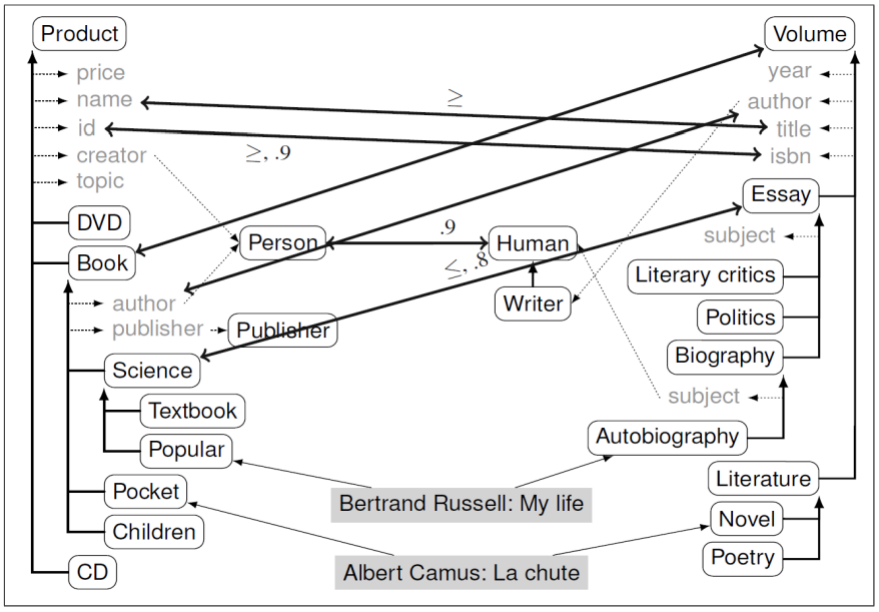

FIGURE 2.14 – Exemple du Matching de deux ontologies [54]

Book $=_{1.0}$ Volume	name $\geq_{1.0}$ title
id $\geq_{.9}$ isbn	author $=_{1.0}$ author
Person $=_{.9}$ Human	Science $\leq_{.8}$ Essay

FIGURE 2.15 – L'alignement généré par l'exemple [54]

5.1.2 Méthodes de comparaison et mesures de similarité

Les algorithmes d'alignement ontologique utilisent différentes mesures pour calculer la correspondance entre les entités ontologiques. Il existe plusieurs mesures comme la similarité/dissimilarité, la distance, la normalisation et l'ultramétrique [54]. Nous présentons ci-dessous trois mesures fréquemment utilisées :

Définition 1 (*Similarité*)

[54] La similarité $\gamma : O \times O \rightarrow R$ est une fonction qui associe à une paire d'entités x et y un nombre réel traduisant la similarité entre elles, tel que :

1. $\forall x, y \in O, \gamma(x, y) \geq 0$ *(positivité) ;*
2. $\forall x \in O, \forall y, z \in O, \gamma(x, x) \geq \gamma(y, z)$ *(maximalité) ;*
3. $\forall x, y \in O, \gamma(x, y) = \gamma(y, x)$ *(symétrie).*

Définition 2 (*Dissimilarité*)

[54] La dissimilarité $\theta : O \times O \rightarrow R$ est une fonction qui associe à une paire d'entités x et y un nombre réel traduisant la non-similarité entre elles, tel que :

1. $\forall x, y \in O, \theta(x, y) \geq 0$ *(positivité) ;*
2. $\forall x \in O, \forall y, z \in O, \theta(x, x) \leq \theta(y, z)$ *(minimalité) ;*
3. $\forall x, y \in O, \theta(x, y) = \theta(y, x)$ *(symétrie).*

Définition 3 (*Distance*)

[54] La distance est considérée comme une sorte de mesure de dissimilarité avec en plus la vérification des deux propriétés suivantes. Étant donné la fonction distance dist : $O \times O \rightarrow R$, *on aura :*

1. $\forall x, y \in O, dist(x, y) = 0$ *ssi* $x = y$ *(définitude) ;*
2. $\forall x, y, z \in O, dist(x, y) + dist(y, z) \geq dist(x, z)$ *ssi* $x = y$ *(intégralité triangulaire)*

Sur la base de ces mesures, plusieurs matchers (fonctions) peuvent être définies selon le contexte d'application. Euzenat [54] distingue trois classes selon que les objets comparés sont des chaines de caractères (comme les noms, les étiquettes et les commentaires/descriptions des entités), des valeurs numériques (de certaines propriétés de concepts) ou des ensembles (le cas des instances ontologiques par exemple).

Notons que le matching peut avoir d'autres entrées comme des seuils ou des poids et/ou des ressources externes telles que les dictionnaires ou les thésaurus comme illustré dans la figure 2.16.

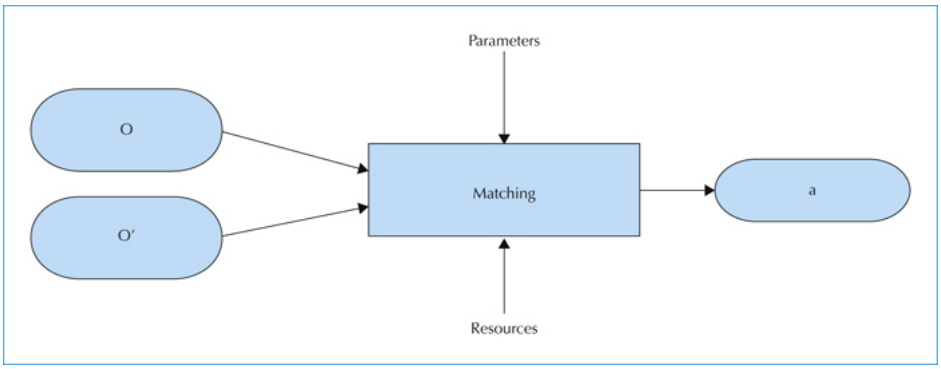

FIGURE 2.16 – Processus d'alignement d'ontologies [54]

5.2 Les étapes du processus du matching

Le processus du matching se compose d'un ensemble d'étapes afin d'aboutir à l'alignement de ses ontologies comme illustré dans la figure 2.17. La réalisation de chaque étape peut se faire selon plusieurs techniques visant à obtenir dans un temps raisonnable le maximum de correspondances sémantiques correctes et cohérentes. Nous expliquons ci-dessous chaque étape.

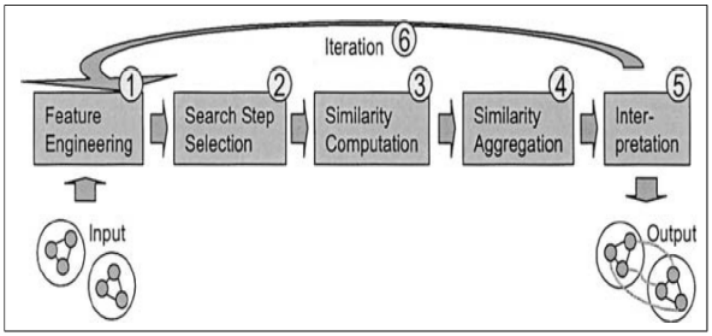

FIGURE 2.17 – Les étapes du processus du matching d'ontologies [52]

1. L'analyse consiste à extraire les différents concepts, relations et instances des deux ontologies O et O' à matcher ainsi que leurs caractéristiques comme les commentaires, les labels, les descriptions, etc. La normalisation et le nettoyage sont des exemples de techniques qui viennent faciliter le processus de comparaison.
2. Le calcul de la similarité consiste à appliquer un ou plusieurs matchers afin d'évaluer la *distance* entre les entités des ontologies à matcher. Plusieurs types de matchers existent selon la technique de matching utilisée, que nous détaillons dans la prochaine section.
3. L'agrégation des similarités consiste à combiner les valeurs de similarité calculées à l'étape précédente afin d'obtenir une seule valeur. Cette combinaison peut s'établir selon plusieurs stratégies. Par exemple, les méthodes de pondération sélectionnent soit la valeur de similarité maximale, minimale, la moyenne ou la somme pondérée des différentes valeurs de similarité calculées par les différents matchers. Cette dernière stratégie s'avère être la plus utilisée dans les systèmes du matching d'ontologies [103] et requiert l'assignation d'un poids pour chaque matcher selon son importance.
4. L'extraction de l'alignement consiste à appliquer une stratégie de filtrage des valeurs de similarités contenues dans une matrice de combinaison définie à l'étape précédente. Ces valeurs représentent les correspondances sémantiques entre les entités des ontologies à matcher. Il existe plusieurs stratégies de filtrage comme le seuillage où seules les valeurs se situant au-dessus d'un seuil, sont sélectionnées.
5. La réparation de l'alignement consiste à garantir la cohérence des correspondances obtenues en éliminant celles causant l'incohérence. Des techniques de raisonnement peuvent être appliquées afin de détecter les éventuelles incohérences.

5.3 Classification des techniques de matching d'ontologies

Selon Euzenat [54], les techniques du matching ontologique peuvent être classées selon leur type comme décrit ci-dessous. Une synthèse de ces techniques est également disponible dans [26].

5.3.1 Techniques terminologiques

Les matchers de type terminologique permettent de calculer la similarité entre les *termes* contenus dans les noms des concepts, des relations, des instances, des labels ou des commentaires, et ceci sans considérer la structure des entités ontologiques.

Le calcul de la similarité entre deux termes se fait selon deux approches : une approche comparant les chaînes de caractères et une approche en se basant sur les techniques issues du traitement linguistique.

Les techniques basées sur la comparaison des chaînes de caractères : considèrent les chaînes de caractères sans prendre en compte leur signification. Plusieurs méthodes ont été développées pour calculer la similarité entre des chaînes de caractères, par exemple, les tests de préfixe et de suffixe ou les fonctions de distance comme la distance d'édition et la mesure n-gramme [143].

Les techniques basées sur le traitement linguistique : englobent les techniques permettant de calculer la similarité entre les termes en les considérant comme des mots du langage naturel et non comme de simples chaînes de caractères. Les termes sont d'abord segmentés en *tokens* en utilisant des séparateurs comme les blancs, les points, les chiffres, etc. Ces tokens vont par la suite être analysés à l'aide d'algorithmes morphologiques issus du traitement automatique du langage naturel. Ces algorithmes opèrent sur la forme de base des tokens, i.e leur singulier au lieu du pluriel et les infinitifs des verbes. Ce mode d'opération est dit *intrinsèque* car il existe un autre mode *extrinsèque* qui calcule la similarité entre les tokens en utilisant des ressources externes telles que les dictionnaires ou les thésaurus (par exemple l'ontologie linguistique *WordNet*) pour détecter la synonymie ou la hyponymie entre les tokens.

5.3.2 Techniques structurelles

Ces techniques se basent sur la structure des entités et non sur leur terminologie. La structure peut être *interne* quand la comparaison porte sur les domaines des entités, le co-domaine, les types de données, la cardinalité, le nombre d'attributs, etc. Par exemple, deux entités sont considérées similaires, si leur nombre d'attributs ainsi que leurs types de données et leurs cardinalités sont proches.

À l'inverse, la structure est *externe* quand la comparaison porte sur les relations entre les entités. Ces techniques sont divisées en les sous-techniques suivantes [54] :

– *Les techniques à base de taxonomie :* dans ces techniques, l'ontologie est considérée comme un graphe. Ces techniques utilisent des algorithmes de graphe pour le calcul de similarité entre les noeuds (reprsentant les concepts ontologiques) en analysant les relations de type *is-a*. Par exemple, si deux concepts C_1 et C_2 sont considérés comme similaires alors les concepts voisins liés par une relation de type *is-a* a ces deux concepts sont également considérés similaires [55].

- *Les techniques à base de graphe :* dans ces techniques la structure de l'ontologie est considéré également comme un graphe. Des algorithmes de graphe sont appliqués afin de calculer la similarité entre les noeuds [54].
- *Les techniques à base de référentiel des structures :* ces techniques utilisent un référentiel pour le calcul de la similarité entre les entités. L'enregistrement des ontologies ainsi que les entités similaires permet d'obtenir ce référentiel. L'objectif de ces techniques est de permettre : (1) la réutilisation de l'alignement ontologique (l'alignement de référentiel) dans le cas d'existence d'ontologies similaires pour chaque ontologie en entrée, (2) initier une nouvelle tâche d'alignement entre l'ontologie similaire et l'ontologie en entrée [166].

5.3.3 Les techniques extensionnelles

Dans cette catégorie, la comparaison entre les deux ontologies porte sur les instances. Cette approche s'avère utile lorsque le nombre des instances est suffisamment important pour les deux ontologies à matcher. Lorsque les ontologies partagent des instances communes, des métriques ensemblistes peuvent être appliquées telles que la métrique de *JACCARD* qui évalue le chevauchement entre les concepts. Lorsque les ontologies ne partagent pas d'instances communes, des techniques qui procèdent à l'agrégation des informations des instances, peuvent être utilisées.

5.3.4 Les techniques sémantiques

Les matchers utilisés dans cette catégorie sont basés sur la logique de premier ordre ou les logiques de description. L'interprétation sémantique des ontologies d'entrée peut être exploitée pour déduire les relations entre les entités comme l'équivalence ou la subsomption, etc. Ces techniques appliquent des méthodes comme les techniques propositionnelles de satisfiabilité, par exemple le solveur SAT ou le raisonnement à base des logiques de description.

Un autre type de ces techniques utilise des ontologies externes de haut niveau comme *SUMO* (the Suggested Upper Merged Ontology) [29] ou FMA (the Foundational Model of Anatomy) [30]. Ces ontologies aident à déduire la sémantique des entités et permettent de structurer des ontologies d'entrée si nécessaire.

Il est à noter que le processus de matching d'ontologies peut faire appel à diverses techniques parmi celles citées. Ces techniques peuvent également être utilisées dans le but de raffiner l'alignement. La combinaison de ces techniques peut être séquentielle ou parallèle [26]. La combinaison séquentielle des matchers se fait en choisissant un ordre d'exécution. La combinaisons parallèle (appelée aussi composite) consiste à lancer parallèlement plusieurs matchers, et à combiner ensuite leurs résultats de matching en utilisant une stratégie d'agrégation.

29. http://www.ontologyportal.com/
30. http://sig.biostr.washington.edu/projects/fm/

6 Conclusion

Dans ce chapitre, nous avons présenté les définitions et notions relatives aux domaines d'intérêt de notre étude, qui sont :

1. *l'ingénierie des besoins* qui est le domaine au coeur de notre problématique. Notre principale contribution consiste à valoriser les besoins fonctionnels dans les systèmes de dépôt afin de mieux les exploiter au niveau décisionnel. Pour ce faire, les notions essentielles relatives aux besoins fonctionnels ont été définies dans ce chapitre.
2. *les entrepôts de données* qui constitue le moyen que nous proposons afin de valoriser les besoins et qui permet leur exploitation et leur analyse. Nous avons parcouru les définitions relatives aux entrepôts de données et leur rôle dans l'intégration, la modélisation et l'analyse des données. Ces mêmes notions seront appliquées et adaptées dans le contexte des sources de besoins.
3. *les ontologies conceptuelles et linguistiques* que nous utilisons comme support pour la résolution des problèmes d'intégration, d'analyse et de raisonnement sur les besoins.

Dans le chapitre suivant, nous présenterons un état de l'art sur les contributions existantes pour la résolution des différents problèmes liés à l'intégration, la gestion et l'exploitation des besoins des utilisateurs.

Chapitre 3

État de l'art sur l'unification, la gestion et l'exploitation des besoins utilisateurs

Sommaire

1 Introduction 60

2 Présentation des travaux : état de l'art 61

2.1 L'intégration des besoins utilisateurs 61

2.1.1 L'intégration des langages de modélisation 61

2.1.2 L'intégration des terminologies utilisées 65

2.1.3 L'intégration de l'univers du discours 66

2.2 La modélisation et l'analyse des BU 67

2.3 La matérialisation et l'exploitation des BU 70

3 Nos critères de classification 72

3.1 L'intégration des BU 72

3.2 Modélisation et analyse des BU 73

3.3 Matérialisation et exploitation des BU 73

4 Synthèse et positionnement de nos contributions 74

5 Conclusion 77

Résumé. Dans ce chapitre nous présentons les principaux travaux traitant de l'*unification*, la *gestion* et l'*exploitation* des besoins issus de sources hétérogènes. Ces dimensions représentent les trois principales phases d'un système d'entreposage. Même si notre contribution est la première à proposer une démarche complète et compréhensive menant à une solution d'entreposage des besoins, certains travaux ont déjà traité de problématiques inhérentes aux entrepôts de données. Afin d'analyser ces travaux, nous avons identifié les dimensions suivantes : (i) pour la phase d'*unification*, nous nous intéressons à l'intégration des besoins et la gestion de leurs hétérogénéités; (ii) pour la phase de *gestion* des besoins, nous nous intéressons à la modélisation et à l'analyse des besoins, ainsi qu'à l'analyse des relations entre eux; (iii) pour la phase d'*exploitation* des besoins, nous nous intéressons à la matérialisation des besoins et leurs diverses utilisations. Nous proposons ensuite une classification de ces travaux selon des critères identifiés pour chaque dimension définie. La synthèse des travaux retenus, leur analyse et leur comparaison nous permettra par la suite de positionner nos contributions par rapport à l'existant.

1 Introduction

Les systèmes d'intégration virtuels ou matérialisés prennent deux principales entrées : les sources de données et les besoins des utilisateurs. Les problématiques liées aux données ont été largement étudiées (intégration, unification, matérialisation, interrogation, analyse), alors que les mêmes problématiques se posant également aux besoins ont été peu ou pas étudiées.

L'importance d'une bonne gestion des BU est liée à leur impact sur l'ensemble des phases du cycle de vie des systèmes d'informations de manière générale (conception, exploitation et maintenance). La gestion de la qualité dans le développement d'un logiciel commence par une compréhension basique des besoins qui sont issus des parties prenantes [123].

Nous remarquons cependant que les premières études associées à la gestion des BU considèrent les besoins comme une boîte noire fournissant un ensemble uniforme de besoins (du point de vue de leur langages de modélisation et terminologie utilisés). Récemment, certains travaux ont remis en cause cette hypothèse et ont proposé de gérer l'intégration des besoins. Leur exploitation est cependant peu étudiée, et si c'est le cas, un système d'intégration virtuel ou une base de données sont proposés pour le stockage des besoins. La présence de besoins hétérogènes, issus de plusieurs partenaires, pose de nombreux défis à différents niveaux, et notamment pour :

- *l'intégration des besoins* qui permet d'assurer une vision unifiée des besoins éliminant différents types d'hétérogénéités.
- *la modélisation des besoins* qui assure une décomposition, une structuration et une représentation efficace et suffisamment détaillée des besoins. Cette modélisation peut se faire en utilisant plusieurs langages (formels, semi-formels et informels). L'absence d'un standard sur la structure d'un besoin rend ces représentations fortement hétérogènes.
- *l'analyse des besoins* qui permet la détection des conflits et des incohérences entre les besoins, ainsi que les relations complexes entre eux. L'identification de ces corrélations et incohérences devient nécessaire lorsque les besoins sont issus de diverses sources n'ayant pas forcément les mêmes langages.
- *la matérialisation des besoins* qui consiste à fournir les mécanismes de stockage et d'interrogation requis pour des besoins unifiés.
- *l'exploitation des besoins* à laquelle sont liées plusieurs problématiques importantes. L'exploitation des besoins consiste ainsi à fournir les moyens efficaces permettant la restitution, l'interrogation, la réutilisation et l'analyse des besoins unifiés. Cette dernière doit pouvoir se faire au niveau décisionnel afin d'identifier les indicateurs pertinents pour des besoins issues des parties prenantes d'un projet donné.

Nous considérons l'ensemble de ces problématiques comme des axes d'analyse des travaux existants. Afin de positionner nos contributions par rapport à l'existant, nous avons commencé par sélectionner les principaux travaux se rapportant aux axes cités (intégration, modélisation et analyse, matérialisation et exploitation des besoins). Ces travaux sont issus de différents do-

maines et principalement de l'ingénierie des besoins, du génie logiciel, ou de la gestion des données.

Après une synthèse des travaux sélectionnés, nous proposons une analyse détaillée de ces travaux se basant sur un ensemble de critères définis pour chaque axe. Ces critères nous permettent d'analyser les études proposées dans la littérature, de les comparer et de positionner notre proposition pour chaque axe et pour chaque critère identifié.

Ce chapitre s'organise comme suit: la section 2 présente un état de l'art des principaux travaux de gestion des besoins utilisateurs. La section 3 présente nos critères de classification pour les travaux retenus. La section 4 analyse les travaux selon les critères définis et positionne nos contributions par rapport à l'existant. La section 5 conclut ce chapitre.

2 Présentation des travaux : état de l'art

Dans cette section, nous présentons une synthèse des principaux travaux se rapportant à la gestion des BU. Nous nous intéressons plus particulièrement à leur intégration, leur analyse et modélisation et à leur exploitation. Pour chacun de ces axes, nous classifions les travaux selon leur principale contribution. Par exemple, pour l'axe d'intégration, nous classifions ces travaux selon leur contribution à l'intégration des modèles de représentation des besoins, l'intégration du vocabulaire utilisé et l'intégration de l'univers du discours. Dans chaque section, nous synthétisons les travaux retenus par ordre chronologique.

2.1 L'intégration des besoins utilisateurs

Dans cette section, nous présentons les travaux qui ont contribué principalement à l'intégration des besoins hétérogènes. Plusieurs types d'hétérogénéités, se rapportant aux langage de modélisation (formalismes) utilisés, à la terminologie utilisée ou aux concepts d'expression des BU, peuvent être gérées.

2.1.1 L'intégration des langages de modélisation

Les langages de modélisation doivent être unifiés car l'expression d'un même besoin peut se faire via une multitude de langages : informels (langage naturel), semi-formels (UML, Merise[128]) et formels (la méthode B). L'hétérogénéité entre ces langages peut se faire entre des langages de modélisation appartenant à la même classe (entre langages informels, semi-formels ou formels) ou entre des langages de modélisation n'appartenant pas à la même classe (par exemple, entre un langage semi-formel et un langage formel).

Dans cette partie, nous présentons les travaux qui ont contribué à l'intégration et l'unification des langages de modélisation des besoins.

Dans [157], *Wieringa et al.* proposent une approche afin d'intégrer les besoins exprimés dans trois différents types de langages de modélisation *formel, semi-formel* et *informel*. Cette approche est basée sur un framework nommé *TRADE*, qui fonctionne comme une boîte à outils pour les besoins et l'ingénierie de conception (*Toolkit for Requirements And Design Engineering*). *TRADE* permet la modélisation et la spécification des besoins exprimés par les utilisateurs. Il intègre plusieurs modèles semi-formels tels que le diagramme de classes et le diagramme de communication. Il est basé sur un langage formel appelé *Albert II* [50], et fournit au final une spécification cohérente d'un ensemble de besoins en les analysant.

Dans [105], *Lopez et al.* proposent un méta-modèle (illustré en figure 3.1) pour intégrer un ensemble de modèles *semi-formels* qui sont : les scénarii, le diagramme des cas d'utilisation, le diagramme d'activité, le diagramme de flux de données, le modèle de tâches et les workflows. Le méta-modèle est conçu avec l'intention de donner la possibilité de réutiliser les besoins. Pour évaluer le méta-modèle développé, un prototype de système appelé R^2 est implémenté.

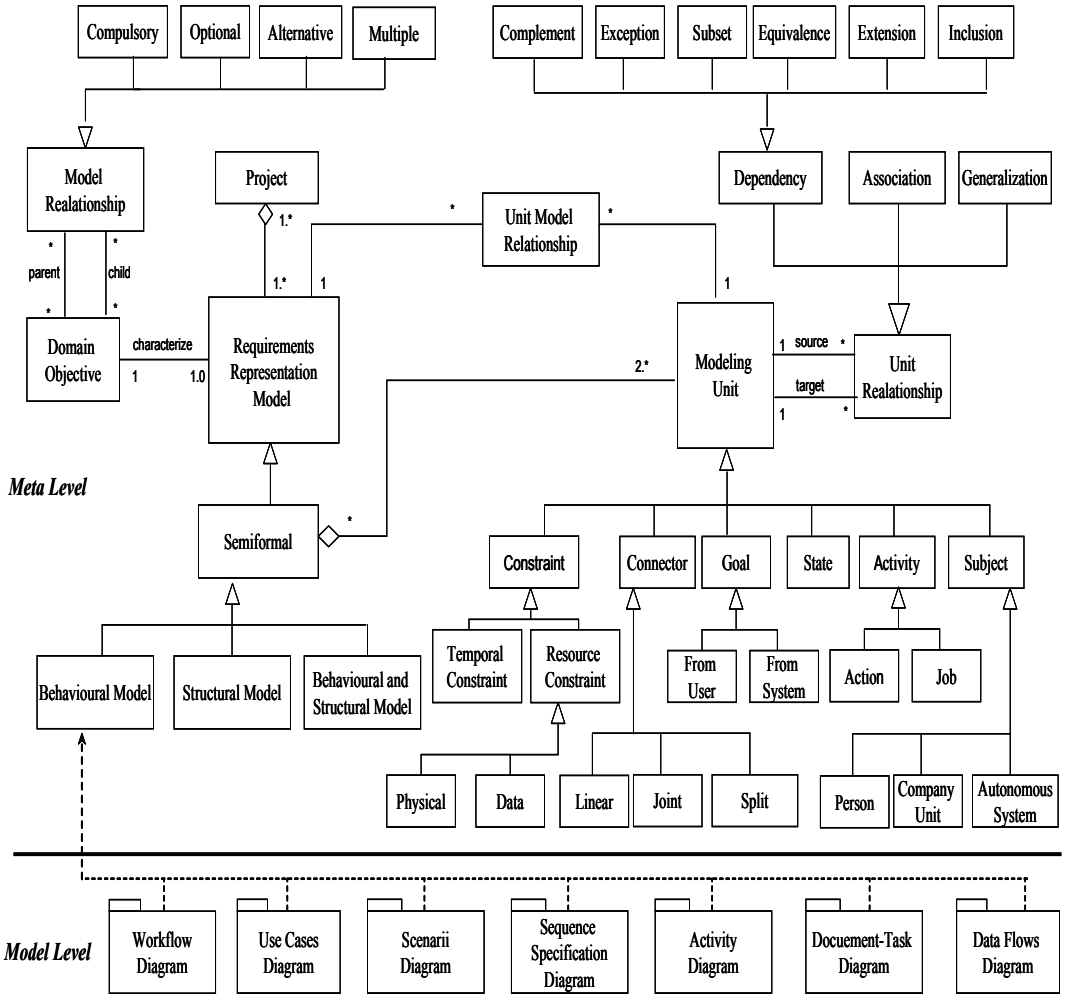

FIGURE 3.1 – Méta-modèle des besoins formalisé en UML [106]

Dans l'étude de *Lee et al*. [99], les auteurs présentent un *framework Onto-ActRE* de gestion des besoins permettant d'unifier différents langages de modélisation *semi-formels* définis par l'ingénierie de besoins. Ce framework utilise un processus ontologique afin de permettre une compréhension partagée des besoins. Pour permettre la participation de divers intervenants et experts, un outil est proposé supportant le framework.

Dans [113], *Navarro et al*. proposent un framework permettant d'intégrer et d'adapter cinq techniques de modélisation répandues dans le domaine d'ingénierie des besoins: l'approche traditionnelle (se basant sur la norme *IEEE 830-1998*), les cas d'utilisation, l'approche orientée aspects (en anglais, *Aspect Oriented*), l'approche orientée buts et la gestion de la variabilité (*variability management*). Le processus d'intégration utilise les techniques de méta-modélisation, et suit une approche dirigée par les modèles. Le méta-modèle est défini en intégrant un ensemble limité de concepts. Des recommandations sont proposées pour étendre l'ensemble des concepts tout en assurant la cohérence du méta-modèle résultant dans l'analyse des besoins.

Dans [149], *Vicente et al*. proposent un méta-modèle de besoins appelé '*REMM*' (*Requirements Engineering MetaModel*). Ce méta-modèle est décrit comme un modèle de référence qui inclut les concepts communs utilisés dans un processus d'ingénierie des besoins : besoins, parties prenantes, cas de tests, etc; ainsi que les relations entre les besoins (essentiellement DependenceTrace, InfluenceTrace, et ParentChildTrace). Ce méta-modèle est complété par un ensemble de contraintes *OCL* (*Object Constraint Language*) utilisées pour valider les besoins. L'approche proposée utilise l'ingénierie dirigée par les modèles (IDM) pour la définition des besoins. Le méta-modèle a pour but de définir des catalogues de besoins réutilisables, et non des besoins spécifiques pour un produit. Ce méta-modèle est implémenté dans un outil *REMM-Studio* pour faciliter l'intégration des modèles de besoins en entrée.

Dans [31], *Brottier et al.* proposent une approche d'intégration d'un ensemble de besoins définis par des spécifications textuelles avec des syntaxes différentes. Les besoins sont donc décrits de manière *informelle*. L'approche est basée sur l'ingénierie dirigée par les modèles. L'approche permet de fournir une plate-forme pour l'analyse des besoins appelée *Requirements to Analysis* (*R2A*). Le composant principal de cette plate-forme est un méta-modèle défini pour représenter les besoins du modèle global des besoins. Cette approche comporte trois étapes : (1) la définition d'un modèle de syntaxe abstrait par l'analyse de chaque spécification textuelle, (2) la définition d'un modèle intégrateur intermédiaire de besoins par l'interprétation de la sémantique abstraite de chaque modèle et (3) l'unification de ces modèles dans un seul modèle de besoins global (voir figure 3.3). Cette étude est complétée par une approche définissant la détection des incohérences des besoins dans le modèle global [119].

Dans [97], *Laleau et al.* proposent une approche d'intégration du langage semi-formel *SysML*, avec le langage formel de la méthode *B*[31]. L'étude utilise une approche de méta-modélisation. L'approche consiste en deux principales étapes : le langage *SysML* est d'abord étendu en lui associant un ensemble de concepts appartenant à différentes méthodes d'ingénie-

31. http://www.methode-b.com/

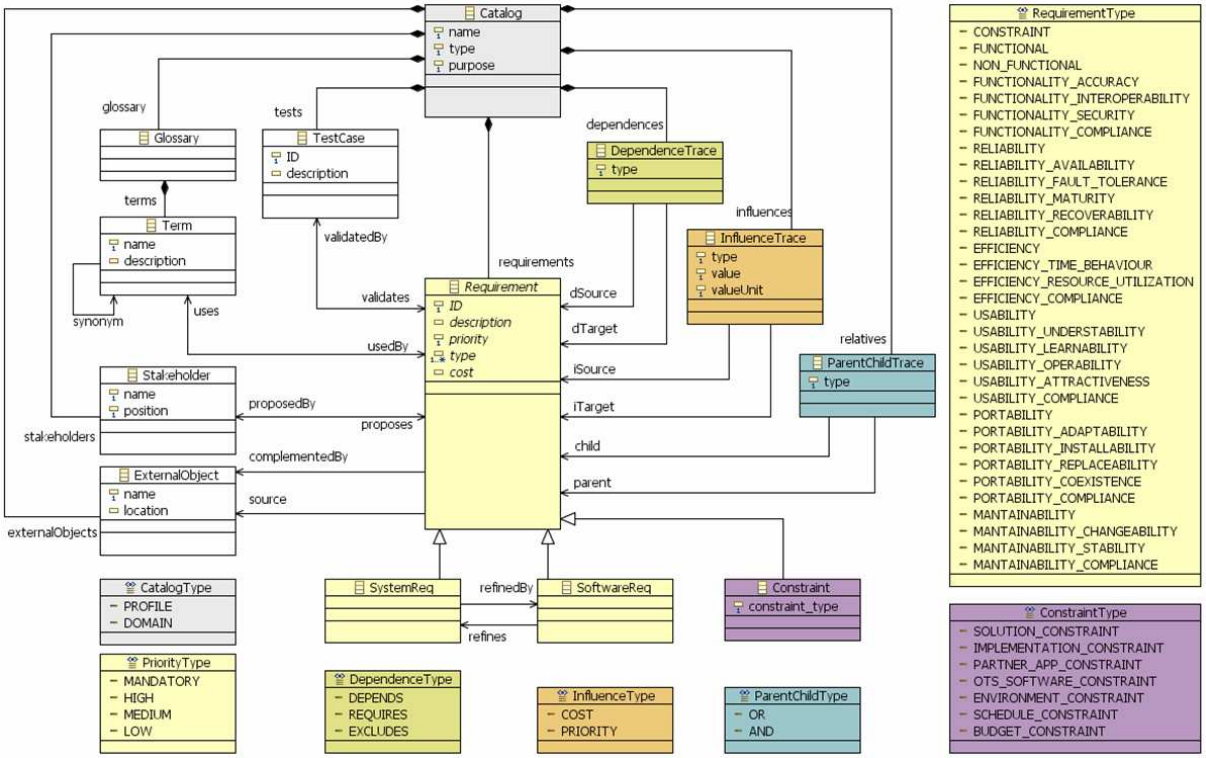

FIGURE 3.2 – Le MetaModel d'ingénierie des besoins dans '*REMM*' [149]

rie des besoins (notamment des concepts de la méthode orientée buts *KAOS*), ensuite l'analyse des besoins est effectuée en définissant un ensemble de règles pour obtenir une spécification formelle du langage *SysML*. L'évaluation de cette approche est réalisée par un outil développé en se basant sur l'environnement *Topcased*.

Dans la recommandation *ITU-T Z.*151 [4], le langage de modélisation de besoins *URN* (*User Requirements Notation*) est proposé pour unifier deux modèles de besoins : (i) le modèle orienté but '*GRL*' (*Goal-oriented Requirement Language*) utilisé pour la description des besoins non fonctionnels et des acteurs; (ii) le modèle utilisation '*UCM*' (*Use Case Maps*) utilisé pour la description des scénarios, des besoins non fonctionnels et des architectures. Le modèle *URN* est proposé pour permettre l'élicitation, l'analyse, la spécification et la validation des besoins utilisateurs. Il permet aussi de détecter les incohérences et les ambiguïtés entre les besoins. Ce modèle est supporté par un outil nommé *jUCMNav* pour la modélisation et la transformation des besoins selon la notation *URN*.

Dans [115], *Nguyen et al.* ont proposé un framework d'intégration de deux modèles semi-formels de description des besoins, à savoir le modèle des cas d'utilisation et le modèle orienté buts. Le framework proposé est conçu afin, d'une part, d'unifier ces deux modèles, et d'autre part, de fournir un moyen plus complet pour la modélisation et l'analyse des buts et des cas d'utilisation. Le framework permet d'automatiser l'extraction des modèles de buts et des cas d'utilisation à partir des documents textuels et l'analyse syntaxique et sémantique de ces mo-

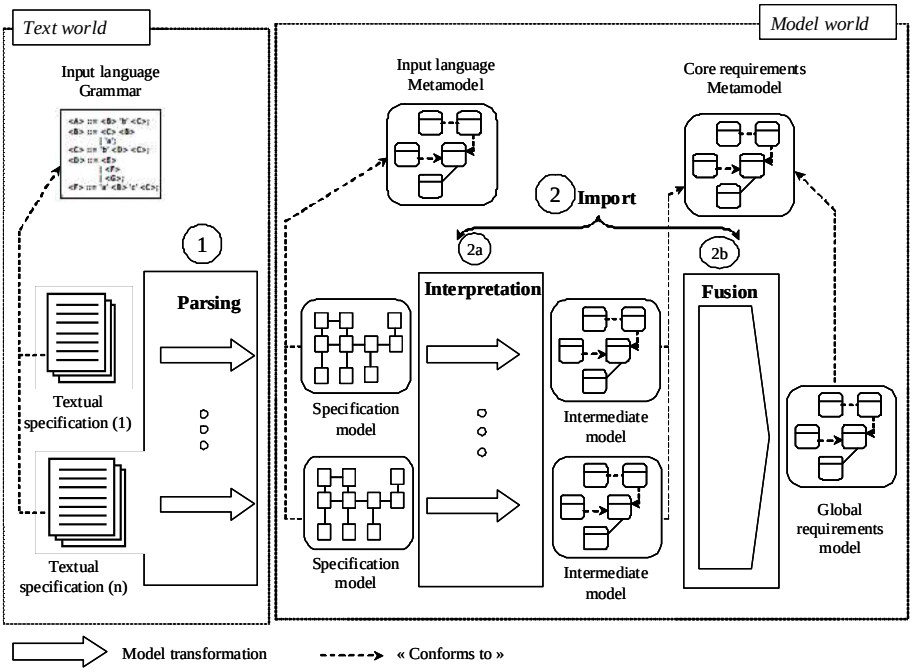

FIGURE 3.3 – Les étapes qui permettent de produire un modèle de besoins global[32]

dèles. Ce framework est composé de trois couches qui sont : (1) la *couche d'artefact* définit la classification des objets couramment utilisés dans un modèle d'intégration de buts et des cas d'utilisations et de leurs relations, (2) la *couche de spécification* fournit les règles de spécification de chaque type d'artefact sur la base d'une grammaire fonctionnelle [43], (3) la *couche ontologique* définit la structure d'ontologies qui peuvent être éventuellement intégrées aux spécifications d'artefacts pour faciliter la compréhension sémantique de l'ensemble des modèles.

Nous remarquons que ces travaux d'intégration des langages de modélisation utilisent deux principales techniques pour la définition de modèles unifiant les langages des besoins : l'ingénierie dirigée par les modèles et la méta-modélisation [149, 4, 31, 97, 113, 105] d'une part, et les techniques issues de la logique mathématique [97, 157] d'autre part. Ces travaux permettent d'unifier un type particulier de langages de modélisation (informel, semi-formel ou formel). D'autres travaux unifient plusieurs types de langages.

2.1.2 L'intégration des terminologies utilisées

L'intégration des besoins est également nécessaire afin de gérer l'hétérogénéité due aux conflits syntaxiques et sémantiques. Les conflits syntaxiques proviennent du fait que les collecteurs de besoins, souvent autonomes, utilisent des terminologies propres à eux. Les conflits sémantiques sont dus aux diverses conceptions des objets du monde réel.

Dans cette partie, nous présentons les travaux qui ont contribué à l'unification des vocabulaires et terminologies utilisés pour l'expression des besoins.

Dans [85], *Kaiya et al* proposent une approche sémantique d'analyse des besoins utilisés en génie logiciel. L'approche est basée sur un thésaurus et des règles d'inférence. La partie thésaurus comprend des concepts et des relations spécifiques nécessaires au traitement linguistique (*synonymes, antonymes, etc.*). Un mapping est établi entre le thésaurus et les spécifications de besoins. La technique proposée est considérée comme un traitement sémantique 'light', complémentaire aux techniques de traitement automatique de la langue naturelle. L'approche proposée permet d'analyser une spécification des besoins et de détecter les incohérences incluses dans une spécification de besoins. L'approche permet aussi de mesurer la qualité d'une spécification sur sa signification et de prédire les modifications futures des besoins en se basant sur l'analyse sémantique de l'historique des modifications effectuées.

Dans [161], *Wolter et al.* proposent une approche permettant l'analyse des besoins, rendant les besoins compréhensibles et sans ambiguïté. L'approche exploite le glossaire *WordNet* [32] [57]. Les mots des spécifications des besoins comportant des significations différentes participent à plusieurs synthèses et un score de fréquence indique quelles sont les significations les plus courantes et les moins fréquentes.

Dans [95], *korner et al.* fournissent une approche semi-automatique pour faciliter la gestion des spécifications textuelles de besoins. L'approche utilise des ontologies linguistiques qui permettent de fournir les '*sens communs*' entre les termes utilisés. L'approche permet de vérifier les défauts linguistiques dans les spécifications, en indiquant quelles parties de la spécification sont ambiguës ou inexactes. Elle offre ainsi un dialogue proposant des suggestions pour améliorer le texte. Un outil nommé '*Requirements engineering specification improver*' (*RESI*), supportant l'approche, est proposé.

Nous remarquons que, de façon similaire à l'intégration des sources de données, ces travaux exploitent une ontologie linguistique ou un thésaurus pour éliminer l'hétérogénie linguistique.

2.1.3 L'intégration de l'univers du discours

Les conflits sémantiques sont dus aux diverses conceptions des objets du monde réel et aux différents points des concepteurs sur le même objet. Comme cité précédemment, ces conflits peuvent être de différente nature, comme les conflits de noms, les conflits de contexte ou les conflits de mesure. Dans cette section nous présentons les travaux qui ont principalement contribué à l'élimination des conflits sémantiques des concepts utilisés lors de l'expression des besoins.

Dans [13], *Assawamekin et al.* proposent un framework nommé '*multiperspective requirements traceability*' (*MUPRET*), qui assure la traçabilité des besoins définis par différents utilisateurs ayant diverses perspectives. Le framework utilise une ontologie pour décrire la perspective de chaque partie prenante. Le vocabulaire de ces ontologies est unifié par l'utilisation d'une ontologie conceptuelle assurant une compréhension partagée et un échange de connaissances

32. lien: http://wordnet.princeton.edu/.

entre les diverses parties prenantes. Le framework *MUPRET* applique différentes techniques afin d'assurer l'intégration et la traçabilité des besoins, notamment les techniques de traitement du langage naturel, les règles de raisonnement afin de résoudre les hétérogénéités et le matching ontologique. Ce matching est utilisé pour faire correspondre sémantiquement les concepts des différentes ontologies des parties prenantes, et aussi pour générer automatiquement des relations de traçabilité.

Dans [27] (thèse développée au laboratoire), *Boukhari et al.* développent une approche qui assurent conjointement la gestion des hétérogénéités de vocabulaires et des formalismes de représentation des besoins utilisés. Cette approche est basée sur l'utilisation d'une ontologie conceptuelle partagée et la définition d'un langage de modélisation intégrateur. Des règles de raisonnement sont définies afin d'identifier les relations entre les besoins. L'approche est supportée par un outil *OntoReqTool*.

Dans [68], *Greenspan et al.* proposent une approche basée sur un langage de modélisation des besoins nommés RML (Requirements Modelling Language). Le langage RML proposé dans cette approche a donné lieu à plusieurs travaux dans le domaine de l'ingénierie de besoins. Le langage RML dispose de son propre langage ontologique qui permet l'expression des besoins.

Dans [134], *Saeki et al.* développent une approche et un outil pour l'analyse des besoins représentés selon une version étendue de l'analyse orientée buts. L'outil est nommé '*AGORA*' (*Attributed Goal-Oriented Requirements Analysis*). AGORA utilise l'ontologie de domaine afin d'assister l'utilisateur pour raffiner, ajouter, supprimer et décomposer des buts. La détection des relations de conflits entre les buts est assurée grâce a utilisation des ontologies conceptuelles.

L'ensemble de ces approches traitant de l'hétérogénéité conceptuelle proposent l'utilisation d'ontologies pour éliminer les conflits lexicaux, sémantiques et syntaxiques. Diverses techniques sont utilisées comme les techniques formelles (règles logiques, raisonnement ontologique, etc.) ou les techniques de traitement du langage naturel pour l'analyse des besoins. Plusieurs architectures d'ontologies sont étudiées comme l'architecture à ontologie unique [68, 134] ou l'architecture à plusieurs ontologies [28, 13].

2.2 La modélisation et l'analyse des BU

Dans cette section, nous présentons les principaux travaux portant sur la modélisation et l'analyse des besoins. Ces travaux ont pour but d'analyser la compréhension des besoins, de les exprimer et de les clarifier. Ces travaux ne gèrent généralement pas la dimension d'intégration car ils supposent que les besoins sont unifiés (du point de vue de leur langages de modélisation et terminologie utilisés). De nombreux modèles et mécanismes de représentation des besoins ayant déjà été proposés. Deux problématiques récurrentes qui reviennent lors de l'analyse des besoins consiste à : (i) identifier et à gérer les besoins conflictuels, i.e. des besoins exprimant des objectifs qui ne peuvent être réalisés simultanément, (ii) identifier les relations entre les besoins. Plusieurs travaux existants couvrant la phase d'analyse des besoins utilisent les ontologies pour

raisonner sur les objets du domaine concerné.

Nous avons retenus les travaux portant sur des contributions différentes comme la représentation d'un besoin, la représentation des relations entre les besoins ou le raisonnement sur les besoins.

Dans [76], *Heitmeyer et al.* proposent une approche d'analyse des besoins, pour la détection automatique des erreurs dans les spécifications des besoins exprimés par la notation tabulaire SCR (Software Cost Reduction). Plusieurs types d'erreurs sont identifiés comme le non-déterminisme, les cas manquants ou les définitions circulaires. L'approche se base sur un modèle formel de besoins reposant sur les automates finis. Le modèle est défini afin de représenter les besoins et la spécification d'un système à développer en fonction des automates. Cette approche est supportée par un éditeur de spécifications et un simulateur.

Dans [147] *Lamsweerde et al.* proposent une approche de gestion des conflits dans le processus d'ingénierie des besoins pour l'approche orientée buts *KAOS* [39]. L'approche repose sur des techniques formelles et des heuristiques pour la détection et la gestion des besoins conflictuels.

Dans [61], *Giorgini et al.* présentent un framework formel pour raisonner sur des modèles de besoins orientés buts. L'objectif du framework est d'analyser les buts à atteindre qui sont modélisés dans une structure hiérarchique pour détecter les buts satisfiables et non satisfiables. La structure proposée représente les relations ET/OU entre les besoins ainsi que les relations d'influence positive ou négative entre les besoins. L'approche proposée repose sur la définition de règles de propagation et d'un mécanisme de raisonnement permettant d'évaluer la satisfaction des buts.

Dans [51], *Dzung et al.* proposent également une approche de raisonnement lors de la phase d'élicitation des besoins. L'approche se base sur une ontologie de domaine afin de structurer les concepts utilisés dans la spécification des besoins et les relations sémantiques entre les besoins. Des mappings sont établis entre l'ontologie de domaine et les besoins. Un ensemble de règles d'inférence est défini afin de raisonner sur les besoins et détecter les incohérences possibles. Cette approche est supportée par un outil.

Dans [65], *Goknil et al.* proposent un méta modèle de besoins et définissent formellement les relations entre les besoins. Les relations sont formalisées avec une logique du premier ordre. Des règles de raisonnement sont ensuite définies et formalisées pour identifier de nouvelles relations et vérifier la consistance des besoins. L'approche proposée est supportée par un outil nommé '*Tool for Requirements Inferencing and Consistency Checking*' (*TRIC*[33]).

Dans [136], *Siegemund et al.* proposent une approche utilisant les ontologies pour structurer les concepts, les besoins et les relations entre les besoins lors d'un processus d'élicitation des besoins. L'approche repose sur un méta-modèle orienté buts, décrivant les besoins ainsi qu'un ensemble de règles de vérification de la cohérence et de la complétude des spécifications des be-

33. http://trese.cs.utwente.nl/tric/

soins. Pour valider la cohérence et la complétude des besoins, l'approche utilise des techniques de réponses aux requêtes pour détecter l'incomplétude des besoins. L'approche fournit ensuite des suggestions sur la façon de résoudre les causes d'incohérence et d'incomplétude.

Dans [109], *Mirbel et al.* proposent une approche pour vérifier la cohérence des besoins qui sont formalisés selon des modèles orientés buts. L'approche donne la possibilité de raisonner sur les besoins, de détecter les relations implicites entre les besoins et de vérifier les incohérences possibles entre eux. Cette approche utilise une démarche de méta-argumentation pour modéliser les besoins et leurs relations. L'approche est supportée par un outil permettant la visualisation des besoins traités.

D'autres travaux s'intéressent à l'analyse des besoins pour faciliter le processus de décision exécuté par l'ingénieur des besoins. Ces travaux se basent généralement sur des techniques d'aide à la décision multi-critères.

Dans [125], *Regnell et al.* discutent des questions et des moyens de recherche descriptive et prescriptive pour comprendre et soutenir le processus décisionnel dans l'ingénierie des besoins. Ces questions et des moyens permettent de comprendre des problèmes tels que: combien d'exigences sont rejetées trop tôt ou trop tard?; à quelle fréquence les exigences sont-elles spécifiées qui ne sont jamais diffusées?; quelle est la qualité adéquate d'une exigence avant de pouvoir entrer dans le processus?. Les auteurs présentent une approche de hiérarchisation des besoins et d'évaluation de l'ensemble des besoins.

Dans [14], *Aurum et al.* examinent les éléments des décisions macroéconomiques axées sur l'organisation ainsi que des micro-décisions axées sur les processus de l'ingénierie des besoins et illustre comment intégrer les modèles classiques de prise de décision avec les modèles de processus d'ingénierie des besoins. Cette intégration aide à formuler un vocabulaire et un modèle commun pour améliorer la gestion du processus d'ingénierie des besoins et contribue au processus d'apprentissage en validant et en vérifiant la cohérence de la prise de décision de l'ingénieur des besoins.

Les travaux présentés ci-dessus présentent plusieurs modèles de représentation des besoins. Certains modèles sont dédiés à une approche spécifique comme l'approche orientée buts tandis que d'autres travaux proposent plutôt des méta-modèles. D'autres travaux s'intéressent aux décisions prises par l'ingénieur des beosins et tentent de les analyser. L'analyse de ces travaux nous permet de constater que plusieurs approches donne une importance capitale à la représentation des relations entre les besoins. Ces relations sont à la base du raisonnement sur les besoins et la vérification de leur consistance. Les mécanismes de spécification et de raisonnement sont pour la plupart ontologiques [136, 51, 65] (pour les approches les plus récentes), ou formels et reposant sur des mécansimes mathématiques [61, 147, 76]. La gestion des besoins conflictuels entre les besoins est importante dans notre contexte , car il est nécessaire d'unifier des besoins provenant de sources différentes et qui ne partageant pas nécessairement les mêmes objectifs. Nous remarquons également que la plupart des travaux omettent certaines relations importantes dans la représentation des besoins, comme les relations d'ordonnancement entre les besoins.

2.3 La matérialisation et l'exploitation des BU

Dans cette section, nous présentons les principaux travaux portant sur le stockage et la matérialisation des BU. Nous montrons que cette matérialisation est fortement liée à des problématiques d'exploitation des besoins très diverses comme l'interrogation ou la réutilisation des besoins.

Le système IBM Rationale *DOORS* [34] (Dynamic Object-Oriented Requirements System) d'IBM permet le stockage, la visualisation et la traçabilité des besoins. *DOORS* permet de représenter et de stocker les besoins dans une base de données orientée objet. L'approche utilisée par DOORS consiste à considérer chaque « paragraphe » (*langage naturel*) des spécifications, décrites dans un document texte ou html ou une feuille de calcul (ligne) comme un besoin. A chaque besoin est attribué un UID lorsque le document est exporté vers la base de données DOORS. Tous les besoins du document sont affichés sous forme de lignes d'objets dans un module DOORS nommé *Formal Module*. Un objet de besoin peut être lié à un autre objet de besoin et permet la navigation entre des besoins. DOORS maintient également l'historique des besoins tout en fournissant les valeurs "De" et "À" de chaque changement de besoin.

Le système DOORS Next generation [35], enrichit les fonctionnalité de Doors en permettant des analyses OLAP via un tableau de bord, des techniques de slice/dice sur les besoins et des rapports simplifiés [36].

Dans [144], *Ambrosio et al.* proposent une approche de réutilisation des besoins appelée *SIREN* (*SImple REuse of software requiremeNts*). L'approche SIREN est basée sur un référentiel réutilisable de catalogues. Chacun de ces catalogues contient un ensemble de besoins liés entre eux, appartenant au même profil comme la Sécurité [144], la protection des données personnelles, ou portant sur le même domaine. L'objectif du référentiel est d'assurer la qualité des catalogues en ajoutant de nouveaux besoins pour améliorer ceux existants, et également de réutiliser les besoins existants dans de nouveaux projets. Dans [118], *Johan and al.* présentent une approche de gestion des besoins à grande échelle pour les entreprises, qui s'appuie sur la communication entre les partenaires du système. Les besoins sont exprimés en langage naturel. Des techniques linguistiques sont appliquées pour l'analyse des besoins et pour établir des liens entre les attentes des clients et les besoins exprimés. L'approche recommande la construction de référentiels de besoins pour les grandes entreprises afin de faciliter leur gestion et l'identification des conflits entre les besoins. L'approche est supportée par un outil *ReqSimile* [37].

Dans l'étude présentée précédemment [27], *Boukhari et al.* proposent une solution de matérialisation dans une base de données ontologique OntoDB. L'objectif du stockage des besoins est de pouvoir les restituer à tout moment à travers des requêtes émises sur la base.

34. http://www-03.ibm.com/software/products/fr/ratidoor
35. http://www-03.ibm.com/software/products/fr/ratidoorng
36. https://www.youtube.com/watch?v=qYK7_g4Fy44&feature=youtu.be
37. http://reqsimile.sourceforge.net

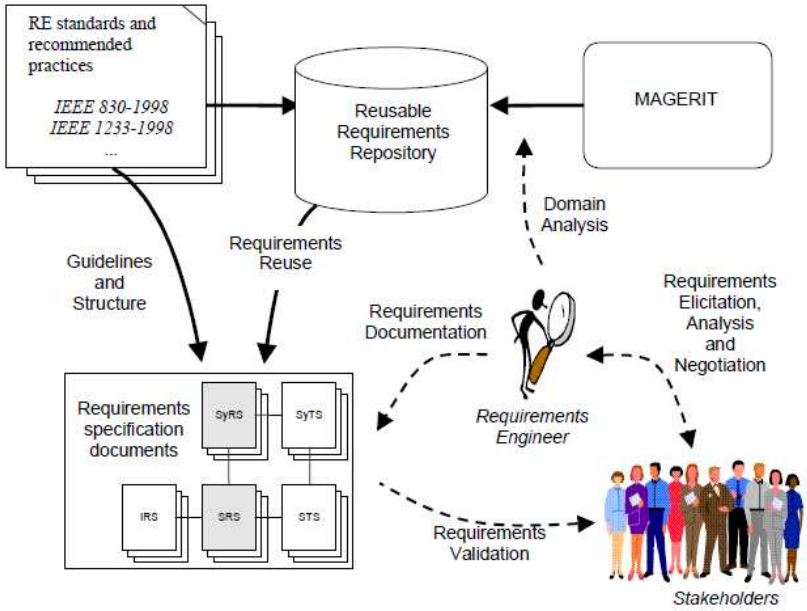

FIGURE 3.4 – Un bref résumé l'approche '*SIREN*' [144]

Dans [91], *Khouri et al.* proposent une approche d'intégration des données et des besoins utilisateurs dans un entrepôt sémantique de données, plus précisément dans une base de données à base ontologique. L'intégration des besoins se base sur l'étude menée dans [27]. L'entrepôt final reste orienté données, et les besoins sont stockés uniquement pour connecter les données aux besoins qui les représentent. Les besoins sont stockés au niveau du méta-schéma et le schéma de données est représenté de façon multidimensionnelle, en utilisant un schéma en étoile ou en constellation.

L'exploitation des beosins peut être orientée vers leur réutilisation. Dans [160], *Wohlin et al.* présentent une étude empirique des critères à considérer pour la réutilisation des besoins dans un projet. Cette étude a été menée à l'aide d'un questionnaire. Le questionnaire étudie l'importance de treize critères différents qui peuvent être utilisés dans le processus de décision. Le résultat montre que certains critères sont plus importants que d'autres : les critères axés sur les entreprises (critères axés sur le client et le marché) ainsi que les problèmes de gestion liés au coût-bénéfice et à la rapidité de livraison sont plus importants que les critères techniques liés à l'architecture et à l'évolution du logiciel.

Nous présentons également deux études proposant des référentiels de stockage des processus et des services. Ces travaux montrent l'intérêt de stocker de nouvelles "entités", autres que les données afin d'assurer leur gestion.

Dans [131], *Marcello et al.* proposent une architecture de référentiel de processus, nommé APROMORE (*Advanced PROcess MOdel REpository*). Cette architecture rassemble un ensemble riche de fonctionnalités permettant l'analyse, la gestion et l'utilisation de modèles de processus. L'objectif de cette architecture est de fournir des fonctionnalités avancées sur les

processus telles que l'analyse, le filtrage et la consolidation sur un ensemble de modèles de processus connectés. *APROMORE* est accessible comme service *SaaS* (*Software-as-a-Service*) open source.

Dans [142], *Yu-Jen et al.* proposent une plateforme pour l'intégration de services disparates dans le but de construire des processus personnalisés efficaces. La plateforme propose une unification des services collaboratifs, en permettant aux participants du processus d'interagir et de collaborer sur les cas pertinents. Un modèle structurant les services est proposé, représentant différentes entités telles que les API, les ressources utilisées, les événements et les tâches. L'objectif de cette plateforme est de permettre la représentation, la manipulation et la réutilisation des connaissances pour la personnalisation des processus.

De manière similaire, les besoins des utilisateurs, considérés comme les "entités" sources des autres entités (données, processus, API, etc.), peuvent être stockés pour assurer une gestion efficace. L'analyse de ces travaux nous permet de constater que plusieurs d'entre-eux soulignent l'importance de matérialiser les besoins, généralement dans des dépôts de type 'référentiels' ou 'bases de données', afin de faciliter leur gestion. Les utilisations de ces référentiels sont diverses mais portent principalement sur l'interrogation des besoins, leur visualisation et leur réutilisation. L'utilisation de ces dépôts reste à un niveau de gestion transactionnelle. Un niveau d'analyse OLAP plus élevé permettant une gestion décisionnelle, importante pour de nombreuses organisations, reste à explorer.

3 Nos critères de classification

Dans cette section, nous présentons les principaux critères de classification que nous avons identifiés afin d'analyser les travaux synthétisés. Nous avons défini un ensemble de critères pour chaque axe d'analyse (l'intégration des besoins, leur modélisation et analyse, ainsi que leur matérialisation et exploitation). Pour chaque critère, nous donnons l'acronyme utilisé pour représenter ce critère dans le tableau comparatif des travaux synthétisés qui est décrit dans cette section. Nous détaillons ensuite chaque critère.

3.1 L'intégration des BU

Le contexte de notre étude suppose l'existence d'un ensemble de sources de besoins issus de différents partenaires. L'intégration des besoins est l'une des principales tâches à accomplir. Cette intégration doit se faire à plusieurs niveaux afin d'éliminer les hétérogénéités :

- des formalismes (langages de modélisation) de représentation des BU (acronyme **If** dans le tableau comparatif 7.3). Ce critère permet ainsi d'identifier les travaux qui ont contribué à l'intégration des formalismes.
- de vocabulaire terminologique d'expression des besoins utilisé (**Lv**).

– de vocabulaire conceptuel d'expression des besoins utilisé, i.e. l'univers du discours comportant l'ensemble des concepts et leur définition (**Cv**).

Comme nous avons constaté que beaucoup d'études, s'inspirant des travaux d'intégration des sources de données, utilisent des approches ontologique pour la gestion des hétérogénéités, nous avons identifié un quatrième critère (**AIO**) identifiant l'architecture d'intégration ontologique utilisée (cf. chapitre 2, section 4.6.2). Trois architectures ontologiques peuvent être utilisées :

– ontologie unique utilisée par toutes les sources de besoins (**Ou**)
– ontologies multiples où chaque source participant au système a sa propre ontologie (**Om**)
– ontologie partagée où chaque source participant au système a sa propre ontologie, qui référence une ontologie partagée (**Op**)

3.2 Modélisation et analyse des BU

Pour ce troisième axe, nous identifions les critères suivants :

– le *Niveau de modélisation (**Nm**)* : ce critère permet d'identifier le niveau de formalisme de représentation des besoins utilisé (informel(**I**), semi-formel(**SF**) ou formel(**F**)).
– *type de modélisation (**Td**)* : ce critère permet d'identifier l'approche de représentation du modèle de besoins (orientée tâche, but, générique, etc.) utilisé. Certains modèles proposés par les auteurs se réfèrent à une approche particulière, dans ce cas on parle de modèles "ad hoc".
– *décomposition des besoins (**Db**)* : ce critère permet d'identifier le niveau de décomposition des besoins. Certains modèles optent pour des modèles représentant les besoins de manière générale, tandis que d'autres modèles proposent une représentation détaillée des éléments d'un besoin.
– *les relations entre les besoins (**Rc**)* : ce critère permet d'identifier les travaux représentant et analysant les relations entre les besoins.
– *raisonnement sur les besoins (**Rb**)* : ce critère permet d'identifier les travaux utilisant le raisonnement (ontologique, mathématique ou orienté pour la prise de décision de l'ingénieur de besoins) sur les besoins afin de détecter les besoins conflictuels ou les relations complexes entre les besoins.

3.3 Matérialisation et exploitation des BU

Pour ce deuxième axe, nous identifions les critères suivants :

– la *persistance des besoins (**Pb**)* : ce critère permet de déterminer les travaux qui ont proposé des solutions de matérialisation des besoins.
– *aspects multidimensionnels (**Ad**)* : ce critère permet d'identifier les travaux proposant une modélisation multidimensionnelle des objets matérialisés (données, processus, besoins, etc.).

- *interrogation des besoins (**Ib**)* : ce critère permet d'identifier les travaux proposant une interrogation et/ou une analyse conventionnelle (**Conv** dans le tableau) permettant une simple restitution des besoins ou une analyse OLAP (**OLAP** dans le tableau) des objets matérialisés.
- *historique des besoins (**Hb**)* : ce critère permet d'identifier les travaux qui préservent l'historique des objets matérialisés.
- *réutilisation des besoins (**Re**)* : ce critère permet de montrer les travaux ayant contribué à la réutilisation des objets matérialisés.

4 Synthèse et positionnement de nos contributions

Le tableau 7.3 compare les travaux synthétisés selon les critères présentés précédemment. Notons que les deux dernières lignes présentent les approches proposées au sein de notre laboratoire.

L'analyse de ce tableau nous permet de faire les constats suivants pour chacun des axes d'analyse :

- **l'intégration des besoins** :
  - l'intégration des besoins concerne généralement une seule dimension se rapportant à l'intégration des formalismes, du vocabulaire exprimé ou des conceptualisations utilisées. Au plus, deux dimensions d'intégration sont considérés. Cette situation peut laisser certaines ambiguïtés et conflits dans les spécifications des besoins unifiés. L'intégration conjointe des trois dimensions d'hétérogénéité, que nous traitons dans notre étude, n'a pas été considérée dans les travaux existants.
  - l'intégration des deux niveaux (schéma et instances) se fait dans les deux cas sans distinction entre les deux couches. Ainsi, l'intégration des instances des besoins dépend fortement du schéma du modèle de besoins proposé. Nous distinguons dans notre étude ces deux niveaux en définissant d'abord le schéma intégrant les sources et ensuite en fournissant les opérateurs nécessaires pour l'intégration des instances des besoins.
  - structure ontologique : nous remarquons d'abord que la plupart des travaux récents traitant de l'intégration des besoins ou même de leur analyse et modélisation exploitent des ontologies. Lorsqu'une approche ontologique est utilisée, la structure considérée repose sur l'existence d'une ontologie partagée entre l'ensemble des sources, qu'elle soit unique ou référencée par les ontologies des sources. Ce scénario n'est pas toujours réaliste. La structure d'ontologies multiples, qui survient souvent dans les projets et qui fournit une autonomie importante de chaque source, n'est pas étudié. Nous considérons dans notre étude les deux scénarios se référant aux structures d'ontologies multiples et d'ontologie partagée.
- **modélisation** : cet axe d'analyse nous permet de constater que :
  - la plupart des travaux s'intéressent aux modèles de besoins semi-formels. Certains tra-

Dimension	Hétérogénéité des sources				Modélisation, structure et analyse					Stockage et réutilisation				
	ICN			AIO										
Étude/Critères	Cv	Lv	If		Nm	Td	Db	Rc	Rb	Pb	Ad	Ib	Hb	Re
Wieringa et al.[157]	✗	✗	✓	✗	*I,SF,F*	*But*	✗	✗	✗	✗	✗	✗	✗	✗
Laleau et al. [97]	✗	✗	✓	✗	*SF, F*	*But*	✗	✗	✗	✗	✗	✗	✗	✗
Lopez et al.[105]	✗	✗	✓	✗	*SF*	✗	✗	✗	✗	✗	✗	✗	✗	✓
Saiki et al.[134]	✓	✗	✗	*Ou*	*SF*	*But*	✗	✓	✓	✗	✗	✗	✗	✗
Navaro et al.[113]	✗	✗	✓	✗	*SF*	✗	✗	✗	✗	✗	✗	✗	✗	✗
Greenspan et al.[68]	✓	✗	✗	*Ou*	*SF*	✗	✓	✗	✗	✗	✗	✗	✗	✗
ITU-T12.[4]	✗	✗	✓	✗	*SF*	*But*	✗	✗	✗	✗	✗	✗	✗	✗
Wolter et al.[161]	✗	✓	✗	*Ou*	*I*	✗	✗	✗	✗	✗	✗	✗	✗	✗
Vicente et al.[149]	✗	✗	✓	✗	*SF*	*But*	✗	✗	✗	✓	✗	✗	✗	✓
Korner and al.[95]	✗	✓	✗	*Ou*	*I*	✗	✓	✗	✗	✗	✗	✗	✗	✗
Kaiya and al.[85, 86]	✗	✓	✗	*Ou*	*I*	✗	✓	✗	✓	✗	✗	✗	✗	✗
Brottier et al.[32, 31]	✗	✗	✓	✗	*I*	*But*	✗	✗	✓	✗	✗	✗	✗	✗
Mirbel et al.[109]	✗	✗	✗	✗	*SF*	*But*	✗	✓	✓	✗	✗	✗	✗	✗
Heitmeyer et al.[76]	✗	✗	✗	✗	*F*	✗	✗	✗	✗	✗	✗	✗	✗	✗
Lamsweerde et al.[98, 147]	✗	✗	✗	✗	*SF*	*But*	✗	✗	✓	✗	✗	✗	✗	✗
Dzung et al.[51]	✓	✗	✗	*Ou*	*I*	✗	✗	✓	✓	✗	✗	✗	✗	✗
Siegemund et al.[136]	✓	✗	✗	*Ou*	*SF*	*But*	✗	✓	✓	✗	✗	✗	✗	✗
Goknil et al.[65]	✓	✗	✗	*Ou*	*F*	✗	✗	✓	✓	✗	✗	✗	✗	✗
Lee et al.[99]	✓	✗	✓	*Ou*	*SF*	*But*	✗	✗	✗	✗	✗	✗	✗	✗
N.Assawamekin et al.[13]	✓	✗	✗	*Om*	*I*	✗	✗	✗	✓	✓	✗	*Conv*	✗	✗
TH Nguyen et al.[115]	✗	✗	✓	✗	*SF*	*But*	✗	✓	✓	✗	✗	✗	✗	✗
Regnell et al.[125]	✗	✗	✗	✗	*I*	✗	✗	✗	✓	✗	✓	✗	✗	✗
Wohlin et al.[160]	✗	✗	✗	✗	*I*	✗	✗	✗	✗	✗	✓	✗	✗	✓
Aurum et al.[14, 15]	✗	✗	✓	✗	*I*	✗	✗	✗	✓	✗	✗	✗	✗	✗
DOORS [38]	✗	✗	✗	✗	*SF*	✗	✗	✓	✗	✓	✗	*Conv*	✓	✗
DOORS Next Gen [39]	✗	✗	✗	✗	*SF*	✗	✗	✓	✗	✓	✗	*OLAP*	✓	✗
Marcello et al.[131]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓
Yu-Jen et al.[142]	✗	✗	✗	✗	✗	*Tâche*	✓	✗	✗	✗	✗	✗	✓	✓
CL Liu et al.[104]	✓	✗	✗	*Ou*	*SF*	✗	✗	✓	✓	✗	✗	✗	✗	✗
Johan et al.[118]	✗	✗	✗	✗	*I*	✗	✗	✗	✗	✓	✗	✗	✗	✗
Giorgini and al.[61]	✗	✗	✗	✗	*SF*	*But*	✗	✗	✓	✗	✗	✗	✗	✗
Ambrosio et al.[144]	✗	✗	✗	✗	*I,SF*	*But*	✗	✗	✗	✓	✗	✗	✗	✓
Boukhari et al.[28]	✓	✗	✓	*Op*	*SF*	✗	✗	✓	✓	✓	✗	*Conv*	✗	✗
Khouri et al.[91]	✓	✗	✓	*Op*	*SF*	✗	✗	✓	✓	✓	✗	*Conv*	✗	✗
Notre approche	✓	✓	✓	*Op,Om*	*SF*	*Tâche*	✓	✓	✓	✓	✓	*OLAP*	✓	✓

TABLE 3.1 – Études des travaux connexes : comparaison et analyse

vaux proposent ou exploitent des langages informels, mais très peu de travaux s'intéressent aux modèles formels. Les résultats des travaux actuels portant sur les langages informels ou semi-formels (qui ont notamment contribué à l'émergence de nouveaux types d'entrepôt de données comme les entrepôts de tweets [126, 96, 111]), nous ont motivés à opter pour la représentation intermédiaire semi-formelle. Nous fournissons cependant une décomposition du besoin très proche du langage naturel (sujet, verbe, complément). Le modèle de besoins que nous proposons dans notre étude est un modèle générique regroupant trois modèles semi-formels répandus.

– plusieurs travaux s'intéressent aux approches orientées buts. D'autres modèles sont ad hoc ou génériques et unifient plusieurs modèles existants. Le manque de consensus concernant un modèle de besoins générique nous a poussé à proposer notre modèle

générique unifiant trois langages semi-formels.
  - le niveau de détail dans la représentation d'un besoin varie d'une étude à une autre. Mais la plupart des travaux considèrent une entité besoin (caractérisée par un ensemble d'attributs) sans détailler ses éléments. Nous représentons les besoins dans le modèle que nous proposons avec une fine décomposition de chaque besoin en tâches élémentaires permettant d'identifier précisément pour chaque tâche : ses sujets (acteurs), ses actions (les verbes) et les objets sur lesquels s'appliquent ce besoin. Cette décomposition est assez proche du langage naturel. Chaque élément du besoin sera connecté à une ontologie, et aura donc une sémantique précise et formelle.
  - la représentation des relations entre les besoins est à la base de la plupart des travaux proposant une analyse des besoins et la détection de leurs inconsistances. Nous représentons également les relations entre les besoins et nous les exploitons en définissant une base de règles de raisonnement qui utilise la décomposition détaillée des besoins afin de fournir une analyse plus fine pour inférer de nouveaux faits sur les besoins. Les relations que nous considérons englobent les relations citées dans la littérature, ainsi que des nouvelles relations d'ordonnancement dont nous avons remarqué l'utilité dans le cadre des projets industriels menés au laboratoire.
- **matérialisation** : est l'un des critères les plus pertinents pour notre étude. La matérialisation des besoins a été proposée dans certains travaux récents. Cette matérialisation se fait généralement dans des référentiels ou des bases de données afin de profiter des mécanismes de gestion efficaces de ces bases. Nous avons identifié trois principaux types de dépôts : un premier type de dépôt se fait à l'instar des données et permet des traitements transactionnels sur les besoins, un deuxième type intégre la dimension décisionnelle sur une base OLTP des besoins, le 3ème type concerne les dépôts *intégrés* de besoins exploités avec des outils transactionnels. Nous complétons cette liste par un quatrième type en proposant de tirer avantage des mécanismes avancés des entrepôts (intégration, gestion et analyse OLAP).
- **exploitation** : cette dimension de modélisation multidimensionnelle et d'analyse OLAP n'a pas été étudiée dans les travaux existants, à l'exception de l'étude [91] qui traite cette dimension sur les données (connectées aux besoins) et non pas sur les besoins entreposés, et du système IBM Doors Next Generation qui propose quelques techniques OLAP très simplifiées, car effectuées sur un dépôt de type 'base de données' stockant les besoins. L'exploitation des besoins a été étudiée dans d'autres perspectives et principalement pour la restitution des besoins, la gestion de leur historique, et leur réutilisation. Nous montrons que nos contributions englobent ces objectifs et permettent un niveau d'analyse décisionnel plus élevé que la simple restitution des besoins.

Dans les chapitres qui suivent, nous présentons nos contributions qui corrigent les lacunes identifiées lors de la synthèse des travaux existants. Pour relever ces nombreux défis, nous avons opté pour une solution d'entrepôt de BU qui assure :

- l'intégration des besoins selon une approche sémantique éliminant les conflits de forma-

lismes, les conflits syntaxiques et les conflits sémantiques entre les besoins. Cette intégration se fera aux deux niveaux : schéma et données. Pour ce faire, nous définirons un schéma intégrant ontologique de l'ensemble des source de besoins, et nous définirons un processus ETL adapté aux besoins pour l'intégration des instances des besoins.
– la modélisation et analyse des besoins par un modèle générique semi-formel permettant une décomposition atomique des besoins modélisés, et une modélisation multidimensionnelle des besoins unifiés. La décomposition détaillée des besoins facilite l'intégration et l'analyse des besoins et l'identification de la consistance et la cohérence de l'ensemble des besoins. Les relations entre les besoins seront également identifiées et analysées.
– l'exploitation des besoins par une approche de matérialisation permettant la persistance des besoins dans un système de gestion de base de données (SGBD), profitant ainsi des techniques d'interrogation et de réutilisation efficaces de ces systèmes. La solution de matérialisation proposée est un système d'entrepôt de données fournissant les mécanismes d'analyse OLAP nécessaires.

5 Conclusion

Nous avons présenté dans ce chapitre les principaux travaux qui ont contribué à l'intégration, la modélisation, l'analyse et l'exploitation des besoins. Nous avons commencé par présenter une synthèse des travaux retenus pour chacun des axes cités. Afin de comparer les contributions de ces travaux, nous avons défini un ensemble de critères pour chaque axe d'analyse. Un tableau comparatif des travaux selon les critères identifiés a été présenté. Une analyse des contributions existantes selon les critères définis nous a permis de positionner nos contributions par rapport aux travaux de la littérature. Notre vision dans ce travail est de revaloriser les besoins en proposant un nouveau type de dépôt de besoins permettant l'intégration, la gestion et l'analyse décisionnelle des besoins. Les quatre prochains chapitres présentent nos principales contributions.

Deuxième partie

Contributions: Construction d'un entrepôt sémantique de besoins fonctionnels

Chapitre 4

Construction de l'Ontologie Intégrante

Sommaire

1 Introduction 84

2 Architecture générale de l'approche d'entreposage 86

3 Scénarios d'intégration 87

4 Approche de construction de l'ontologie intégrante 88

4.1 Scénario d'une ontologie partagée 88

4.1.1 Hypothèse 1 88

4.1.2 Hypothèse 2 92

4.2 Scénario d'intégration "ontologies multiples" 93

4.2.1 Matching ontologique 94

4.2.2 Algorithme de génération de l'ontologie intégrante . . 99

5 Validation de l'approche : études de cas 99

5.1 Étude de cas 1 : ontologie partagée 101

5.2 Étude de cas 2 : ontologies multiples 103

5.2.1 Performances du mécanisme de matching 105

6 Conclusion 106

Résumé. Dans le domaine de l'intégration d'informations issues de diverses sources, les ontologies ont montré leur capacité à faciliter la gestion des conflits syntaxiques et sémantiques entre ces sources et à automatiser le processus d'intégration. Les systèmes d'intégration matérialisée de type entrepôt nécessitent de définir un schéma intégrant unifiant les sources. Dans notre contexte, ce schéma est représenté par une ontologie intégrante. Cette dernière est construite à partir des ontologies locales des sources, qui sont utilisées pour exprimer les termes et concepts employés pour définir les besoins locaux. Les besoins intégrés seront ainsi également définis sémantiquement par cette ontologie intégrante. Dans ce chapitre, nous procédons à la construction de l'ontologie intégrante selon deux scénarios différents d'intégration possibles : un premier scénario où les ontologies locales sont construites à partir d'une ontologie partagée, et un deuxième scénario où les ontologies locales sont définies indépendamment les unes des autres.

1 Introduction

Notre solution pour gérer les difficultés liées à l'intégration, à l'accès et à l'analyse des besoins issus de sources variées est une solution d'entrepôt sémantique de besoins fonctionnels (ESBF).

La première étape de tout projet d'entreposage consiste à assurer l'intégration des sources. Dans notre cas, il s'agit de sources de besoins. La phase d'intégration consiste à analyser et à gérer les conflits entre les Besoins des Utilisateurs (BU). Cette tâche est considérée comme l'une des activités fondamentales de l'analyse des besoins [140]. Elle représente une véritable difficulté lorsque plusieurs partenaires, qui peuvent avoir des compétences et des formations diverses, interagissent au sein d'un même système. Chaque partenaire exprime ses besoins en utilisant ses propres termes (le jargon) et ses propres concepts (l'univers de discours). Ces besoins sont ensuite structurés en un modèle de besoins utilisant un formalisme donné (le langage de modélisation). Cette problématique d'intégration des besoins est également présente chez les grandes entreprises avec lesquelles notre laboratoire [40] collabore, comme l'entreprise de construction aéronautique *Airbus* [41] [27] et l'entreprise multinationale française du secteur de l'énergie EDF [42] [133].

L'intégration des besoins peut facilement emprunter les solutions proposées et approuvées pour l'intégration des sources de données hétérogènes. L'intégration des sources de données a été largement étudiée, et plusieurs projets proposent d'utiliser les ontologies afin d'expliciter le sens de chaque objet manipulé et faciliter ainsi leur intégration, comme par exemple le projet OntoDaWa mené au laboratoire [114]. Plusieurs structures d'intégration à base ontologique ont ainsi été proposées (cf. chapitre 2, section 4.6.2) [155] :

1. Structure 1 : "*ontologie unique*", où chaque source d'informations est liée à une unique ontologie. Cette structure impose de trouver un vocabulaire minimal commun partagé entre les sources ce qui limite l'autonomie des schémas des sources;
2. Structure 2 : "*ontologies multiples*", où chaque source d'informations définit sa sémantique par une ontologie locale;
3. Structure 3 : "*ontologie partagée*", où chaque source est décrite sémantiquement par sa propre ontologie locale. Ces ontologies sont mises en correspondance avec une ontologie partagée modélisant un domaine particulier. Cette ontologie partagée est supposée être déjà existante.

Les types d'ontologie considérés dans ce contexte sont les ontologies de domaine et/ou de tâches. Dans le contexte des besoins fonctionnels, la structure (1) exige de trouver un vocabulaire minimal et limite fortement l'autonomie des sources. Les hypothèses des structures (2) et (3) sont plus plausibles.

40. http://www.lias-lab.fr/?lang=en
41. http://www.airbus.com/.
42. https://www.edf.fr/entreprises.

Pour gérer cette phase d'intégration, l'entrepôt est vu comme un système d'intégration. Les systèmes d'intégration sont définis par le triplet <G,S,M> où G est le schéma global unifiant les sources, S sont les schémas locaux et M est un ensemble de mappings entre G et S. Dans notre contexte, S sont représentées par les ontologies locales connectée aux besoins (décrit selon un semi-formalisme défini), G est l'ontologie intégrante connectée au modèle de besoins cible (à définir) et M est un ensemble de mappings à définir également. Nous nous intéressons dans ce chapitre à la *partie ontologique uniquement*, plus particulièrement à la construction de l'ontologie intégrante.

L'ontologie intégrante permet d'éliminer les conflits entre les sources. Deux principales catégories d'ontologies sont utilisées pour gérer les différents conflits syntaxiques et sémantiques entre les sources [121] (cf. chapitre 2, section 4.2) :

1. *les Ontologies Linguistiques ou Multilingues* (OL) qui sont utilisées pour définir le champ lexical des termes qui apparaissent dans un domaine donné, et les relations entre des termes tels que la synonymie ou l'antonymie. Ces ontologies gèrent principalement les conflits syntaxiques.

2. *Les Ontologies Conceptuelles* (OC) utilisées pour définir les concepts d'un domaine. Ces ontologies permettent de gérer les conflits sémantiques. Comme illustré dans le modèle ontologique en oignon (chapitre 2, section 4.2), un concept peut être associé à différents termes dans la couche linguistique de l'OL.

Notre approche se base sur les structures ontologiques citées précédemment (ontologie partagée et ontologies multiples). Les sources des besoins sont exprimées en utilisant des termes d'une ontologie linguistique locale, qui référencent des concepts d'une ontologie locale conceptuelle. Dans les travaux de conception des projets d'entreposage (des sources d'informations de manière générale) utilisant des ontologies, la structure 3 (ontologie partagée) est généralement suivie. L'ontologie partagée est supposée pré-existante. L'ontologie intégrante est définie à partir de cette ontologie partagée. Dans notre cas, nos contributions sont de :
(i) considérer les deux structures ontologiques et proposer un mécanisme de matching ontologique pour unifier les ontologies locales,
(ii) considérer l'unification des ontologies au niveau conceptuel et au niveau terminologique, contrairement aux travaux existants qui considèrent une seule couche ontologique (cf. chapitre 3, section 3.1). Dans ce contexte, l'utilisation de la couche conceptuelle contribuera à uniformiser les concepts utilisés, tandis que la couche linguistique et multilingue contribuera à l'expression des besoins de façon plus expressive, uniforme et multilingue. Une autre caractéristique de la couche conceptuelle est sa capacité de raisonnement que nous utilisons pour identifier les conflits et les incohérences dans les besoins (ce sera l'objet des prochains chapitres).

Ce chapitre est organisé en six sections. La section 2 présente l'architecture générale de notre démarche d'entreposage des sources de besoins. La section 3 présente les scénarios d'intégration de notre approche. La section 4 décrit l'approche de construction de l'ontologie intégrante, qui est la première étape requise dans notre démarche d'entreposage. La section 5

propose deux études de cas et une étude expérimentale pour valider l'approche proposée. La section 6 résume nos contributions pour ce chapitre.

2 Architecture générale de l'approche d'entreposage

La figure 4.1 présente l'architecture générale de notre approche.

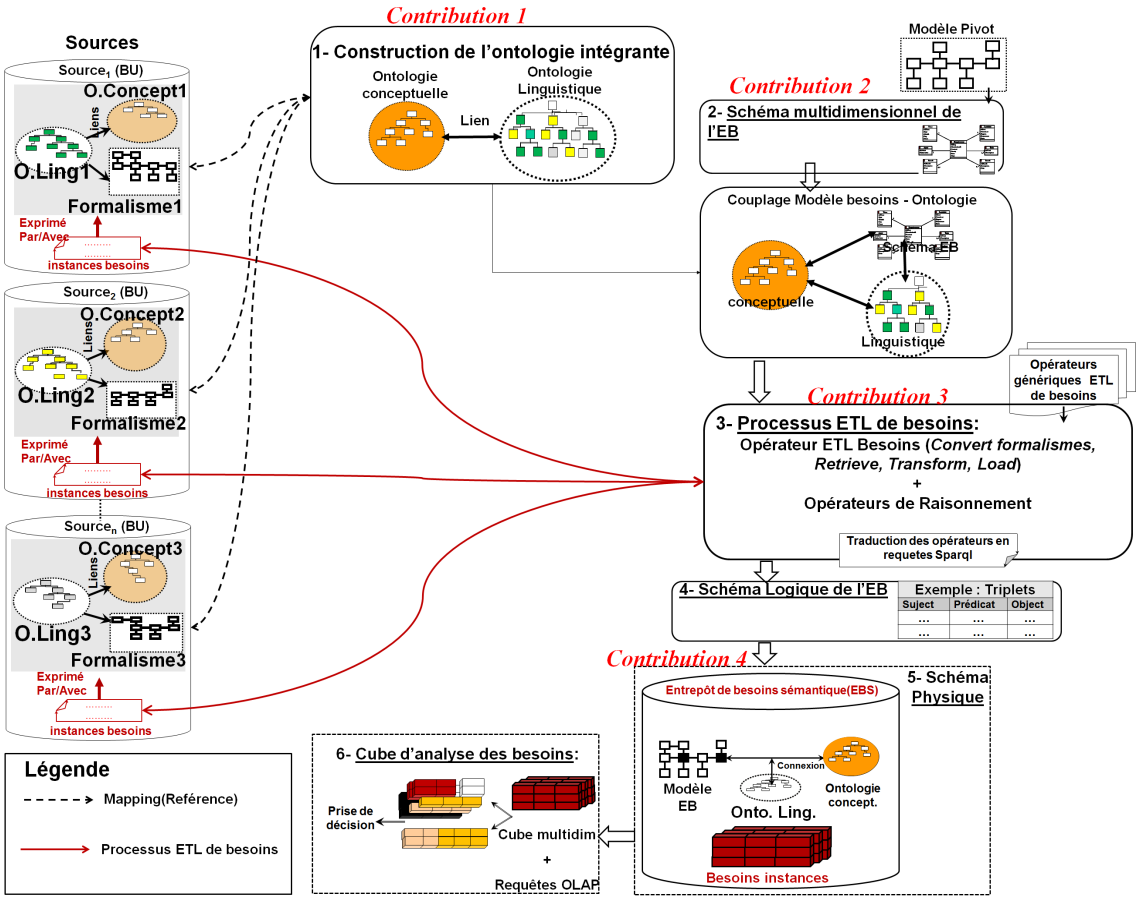

FIGURE 4.1 – Approche générale

Celle-ci se base sur les structures ontologiques citées en introduction (ontologie partagée et ontologies multiples). L'approche repose en entrée sur des sources de besoins qui décrivent l'ensemble des besoins selon un formalisme donné, utilisant les concepts d'une ontologie locale conceptuelle et les termes d'une ontologie locale linguistique. Ces ontologies référencent, en utilisant un ensemble de mappings, une ontologie intégrante (conceptuelle et linguistique) qui unifie les termes et concepts utilisés. Cette ontologie intégrante doit être construite.

Le modèle multidimensionnel de l'ESBF sera défini. Un processus ETL adapté aux besoins permettra d'extraire les besoins des sources, de les transformer et ensuite de les charger dans le

modèle de l'ESBF. En utilisant les mappings définis entre les ontologies locales et intégrante, le modèle de l'ESBF est ainsi connecté à l'ontologie intégrante, ce qui permettra ainsi de définir sémantiquement chaque composante du modèle. Par exemple, si le terme utilisé dans un besoin de la source S_1 est "Student" (avec son concept associé) et celui utilisé dans un besoin de la source S_2 est "Etudiant" (avec son concept associé), ces deux concepts correspondent au concept "Student" dans le modèle de l'ESBF, caractérisé par les propriétés *id*, *nom* et *age*. L'ontologie intégrante peut être sauvegardée lors de l'implémentation si un SGBD sémantique est choisi.

Nous nous intéressons dans ce chapitre à la construction de l'ontologie intégrante. La difficulté consiste définir les mapping M entre les ontologies. Ces mappings définis permettent la construction de l'ontologie intégrante selon les scénarios d'intégration détaillés ci-dessous.

3 Scénarios d'intégration

La construction de l'ontologie intégrante repose sur les deux structures ontologiques "ontologies multiples" et "ontologie partagée". Deux scénarios d'intégration sont possibles dans ce cas :

(i) dans le *premier scénario*, chaque source définit localement ses besoins en utilisant une ontologie locale indépendamment des autres sources selon la structure d'intégration ontologique 2 (ontologie multiple). Ce scénario ne requiert aucune hypothèse particulière, mais un processus de matching ontologique doit être défini entre les ontologies locales afin de les unifier et construire l'ontologie intégrante.

(ii) *d*ans le *deuxième scénario*, l'existence d'une ontologie partagée (conceptuelle et linguistique) est assumée à l'ensemble des sources (cas d'un donneur d'ordre qui fournit ce schéma partagé). Dans ce scénario, les mappings sont déjà disponibles puisque les sources locales sont considérées comme des vues sur l'ontologie partagée, mais l'autonomie des partenaires locaux se trouve limitée à ce schéma partagé.

De manière générale, les solutions d'entreposage ontologiques suivent le deuxième scénario et supposent l'existence d'un schéma partagé entre les sources, ce qui limite leur autonomie. Ce scénario n'est pas toujours réalisable. Nous proposons de relâcher cette contrainte en utilisant un processus de matching ontologique comme requis par le premier scénario. Cette contribution est ainsi valable pour tout type d'entrepôt (données, besoins ou autre) et nous l'exploitons pour l'entité (représentée par les besoins fonctionnels) qui nous intéresse.

La construction de l'ontologie intégrante repose sur les deux scénarios cités. Dans le premier scénario, elle est construite en utilisant un processus de matching. Nous revenons ainsi à la structure 3 d'une ontologie partagée référencée par les ontologies locales. Dans le deuxième scénario, elle peut être similaire à l'ontologie partagée existante ou elle peut aussi être extraite à partir de l'ontologie partagée.

Exemple 1

Supposons les deux besoins suivants issus respectivement des sources S_1 et S_2 :
B_1: "The system shall allow students to create teams".
B_2: "Le système doit permettre aux étudiants de s'inscrire".
Le terme utilisé dans le premier besoin de la source S_1 est Student (avec son concept associé utilisant les propriétés Id et Nom et Date de naissance) et celui utilisé dans le besoin de la source S_2 est "Etudiant" (avec son concept associé utilisant les propriétés Id et Nom et Age). Ces deux concepts correspondent au concept "Student" caractérisé par les propriétés id, nom et age dans l'ontologie intégrante.

Nous détaillons dans ce qui suit la construction de l'ontologie intégrante pour les deux scénario cités.

4 Approche de construction de l'ontologie intégrante

Notre approche se décompose en plusieurs étapes qui sont détaillées dans ce qui suit .

4.1 Scénario d'une ontologie partagée

Dans ce scénario, chaque source définit ses besoins en utilisant les termes et concepts d'une ontologie locale (conceptuelle et linguistique). Ces ontologies référencent une ontologie partagée supposée pré-existante. Ce scénario s'appuie sur plusieurs hypothèses que nous détaillons dans ce qui suit.

4.1.1 Hypothèse 1

Nous considérons l'existence d'une *ontologie conceptuelle de domaine* (OC) qui définit le domaine d'intérêt de l'entrepôt. L'hypothèse d'existence d'une ontologie de domaine partagée est affaiblie par le développement de plusieurs grandes ontologies dans divers domaines tels que l'industrie, la médecine, l'écologie, etc.

Nous commençons par rappeler la formalisation d'une ontologie conceptuelle (cf. chapitre 2, section 4.4) .

Formalisation 1

$OC :< C, R, Ref(C), Ref(R), \mathcal{F} >$, avec:
- *C: les classes de l'ontologie conceptuelle.*
- *R: les rôles de l'ontologie locale i de la source S_i correspondant exactement aux rôles de l'OC (ontologie conceptuelle).*

- $\mathcal{R}ef(C)$: *est une fonction qui associe à chaque classe un opérateur (inclusion ou équivalence) et une expression Exp sur d'autres classes et des propriétés.*
- $\mathcal{R}ef(R)$: *est une fonction qui associe à chaque rôle sa définition.*
- $\mathcal{F}$ *: le formalisme de l'ontologie comme OWL, RDF ou PLIB.*

Notons que, dans cette définition, nous nous intéressons au schéma de l'ontologie. Les instances ontologiques sont stockées au niveau des ontologie locales des sources.

Dans ce scénario, chaque concepteur d'une source de besoins construit son ontologie locale en utilisant les termes et concepts de l'ontologie partagée lors de l'expression des besoins.

Ce scénario implique l'extraction de l'ontologie locale correspondante à chaque source à partir de l'ontologie partagée. L'extraction d'ontologies locales s'effectue par projection des besoins de la source sur l'ontologie partagée : OL = $\prod_{Besoins}$ (OP) avec OL $\subseteq$ OP.

Dans le cas où les besoins des sources sont bien spécifiés, ceux-ci doivent contenir tous les concepts et propriétés nécessaires et manipulés par le système en question et appartenant à l'univers du discours (voir figure 4.2). Ces besoins sont formalisés en utilisant un langage de modélisation donné. Dans ce chapitre, nous nous intéressons uniquement à la partie ontologique. Nous considérerons l'intégration des langages de modélisation dans les prochains chapitres.

Afin d'assurer la complétude et la cohérence des ontologies locales extraites au niveau des sources à partir de l'ontologie partagée, nous avons utilisé une méthode de modularité. Une méthode de modularité se décompose en trois étapes: (1) projection des besoins d'une source sur l'ontologie partagée, (2) extraction du fragment ontologique annoté par les besoins de la source, et (3) construction de l'ontologie locale propre à une source. Rappelons que plusieurs méthodes de modularité ontologiques existent dans la littérature selon le langage ontologique utilisé (comme les opérateurs de modularité OWL ou l'opérateur *Case-Of* pour le langage PLIB [120]).

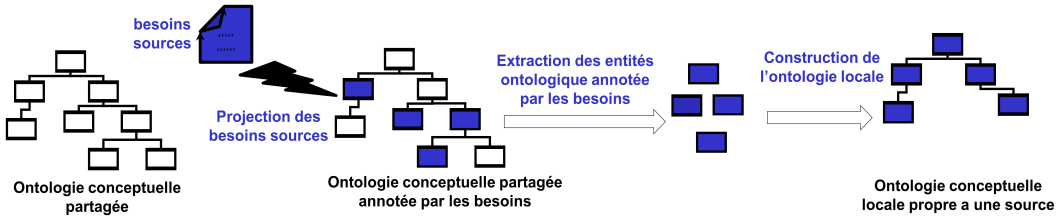

FIGURE 4.2 – Les étapes d'extraction de l'ontologie locale à partir de l'ontologie partagée [27]

Plusieurs scénarios d'extraction des ontologies locales à partir de l'ontologie partagée sont possibles :

1. OL ≡ OP : les concepts et propriétés de l'ontologie conceptuelle partagée (OP) correspondent exactement aux attentes des concepteurs (voir l'exemple dans la Figure 4.3). L'étape

d'expression des besoins au niveau des sources en termes de classes (C_{OLi}), de propriétés (R_{OLi}) et des fonctions ($Ref(C)_{OLi}$,$Ref(R)_{OLi}$) coïncide exactement avec les composantes de l'ontologie partagée. En d'autres termes, la description de l'ontologie locale correspond à la description de OP.

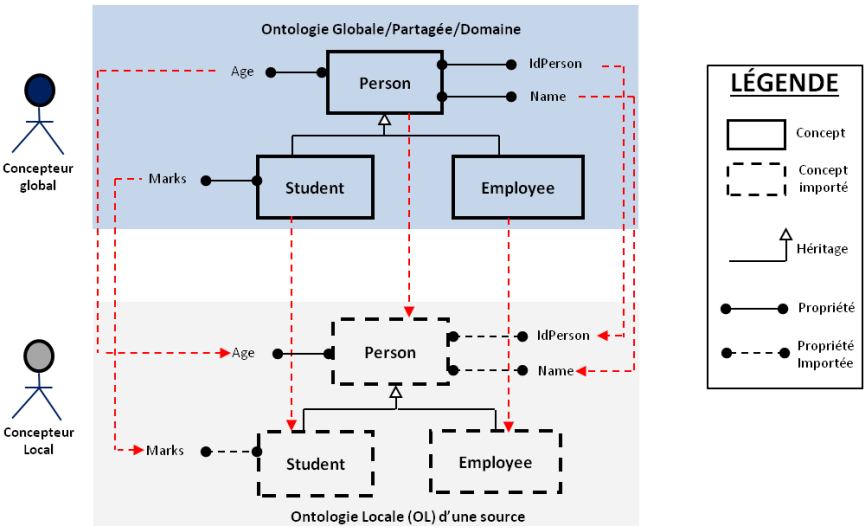

FIGURE 4.3 – Exemple d'extraction d'une ontologie locale équivalente à l'ontologie partagée

Formalisation 2

$OL_i :< C_{OLi}, R_{OLi}, Ref(C)_{OLi}, Ref(R)_{OLi}, \mathcal{F}_{OLi} >$, *avec:*

- $C_{OLi} \equiv C_{OP}$*: les classes de l'ontologie locale i de la source* S_i *correspondant exactement aux classes de l'OP.*
- $R_{OLi} \equiv R_{OP}$*: les rôles de l'ontologie locale i de la source* S_i *correspondant exactement aux rôles de l'OP.*
- $Ref(C)_{OLi} \equiv Ref(C)_{OP}$*: les fonctions sur les classes de l'ontologie locale i de la source* S_i *correspondant exactement aux fonctions sur les classes de l'OP.*
- $Ref(R)_{OLi} \equiv Ref(R)_{OP}$*: Les fonctions sur les rôles de l'ontologie locale i de la source* S_i *correspondant exactement aux fonctions sur les rôles de OP.*

2. OL $\subset$ OP : les concepts et propriétés de l'ontologie locale *i* sont extraits à partir de l'ontologie partagée et couvrent tous les attentes des utilisateurs (voir l'exemple dans la Figure 4.4). En d'autres termes, la spécification des besoins par les concepts ontologiques représente un fragment de l'ontologie partagée. Ainsi, OL $\subset$ OP signifie que:

Formalisation 3

$OL_i :< C_{OLi}, R_{OLi}, Ref(C)_{OLi}, Ref(R)_{OLi}, \mathcal{F}_{OLi} >$, *avec :*

- $C_{OLi} \subset C_{OP}$ *: les classes de l'ontologie locale i de la source* S_i *représentant un fragment*

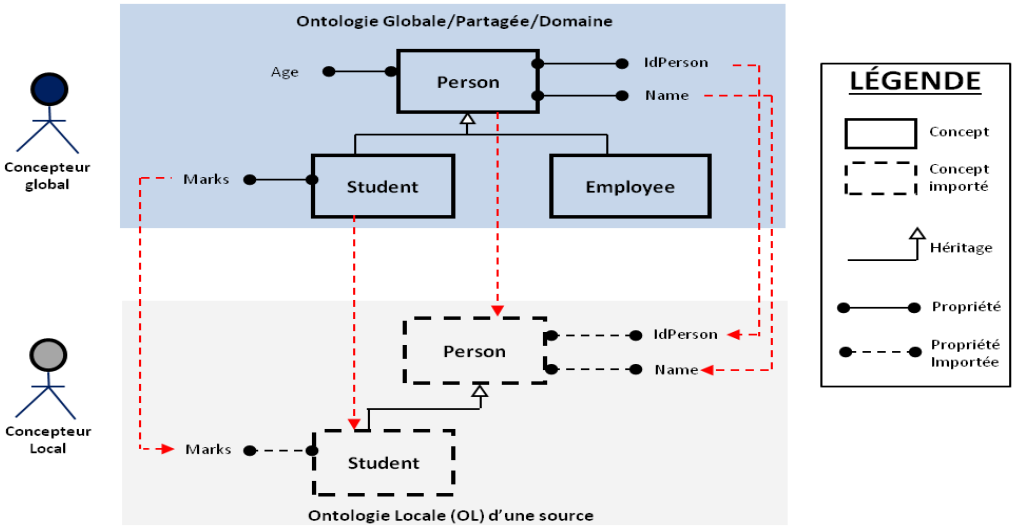

FIGURE 4.4 – Exemple d'extraction d'une ontologie locale incluse à l'ontologie partagée.

sur les classes de l'OP.

- $R_{OLi} \subset R_{OP}$ *: les rôles de l'ontologie locale i de la source* S_i *représentant un fragment sur les rôles de l'OP.*
- $Ref(C)_{OLi} \subset Ref(C)_{OP}$ *: les fonctions sur les classes de l'ontologie locale i de la source* S_i *représente un fragment sur les fonctions sur les classes de l'OP.*
- $Ref(R)_{OLi} \subset Ref(R)_{OP}$ *: les fonctions sur les rôles de l'ontologie locale i de la source* S_i *représentant un fragment sur les fonctions sur les fonctions sur les rôles de l'OP.*

3. OL ⊃ OP : le sous-ensemble de concepts (classes et propriétés) qui sont extraits à partir de l'ontologie partagée ne couvre pas toutes les attentes utilisateurs. Ce sous-ensemble nécessite d'être étendu et spécialisé pour répondre aux objectifs applicatifs spécifiques des utilisateurs (voir l'exemple dans la Figure 4.5). Afin de garder le référencement avec l'ontologie partagée, l'extension de l'ontologie locale se fait uniquement par spécialisation des classes et propriétés (relation de subsomption) ou par leur renommage. Le renommage n'influe pas sur la définition de l'ontologie locale *i* car chaque concept de l'ontologie locale *i* est associé à un identifiant unique et le référencement entre l'ontologie locale et partagée se fait via les identifiants. Ainsi, la relation OL ⊃ OP signifie que :

Formalisation 4

$\mathcal{OL}_i :< C_{OLi}, R_{OLi}, \mathcal{R}ef(C)_{OLi}, \mathcal{R}ef(R)_{OLi}, \mathcal{F}_{OLi} >$, *avec :*

- $C_{OLi} \supset C_{OP}$*: un sous-ensemble des classes de l'ontologie locale i de la source* S_i *est extrait de l'ontologie partagée et l'autre sous-ensemble est défini pour étendre l'ontologie locale par l'ajout de nouvelles classes (par relation de subsomption).*
- $R_{OLi} \supset R_{OP}$*: un sous-ensemble des propriétés de l'ontologie locale i de la source*

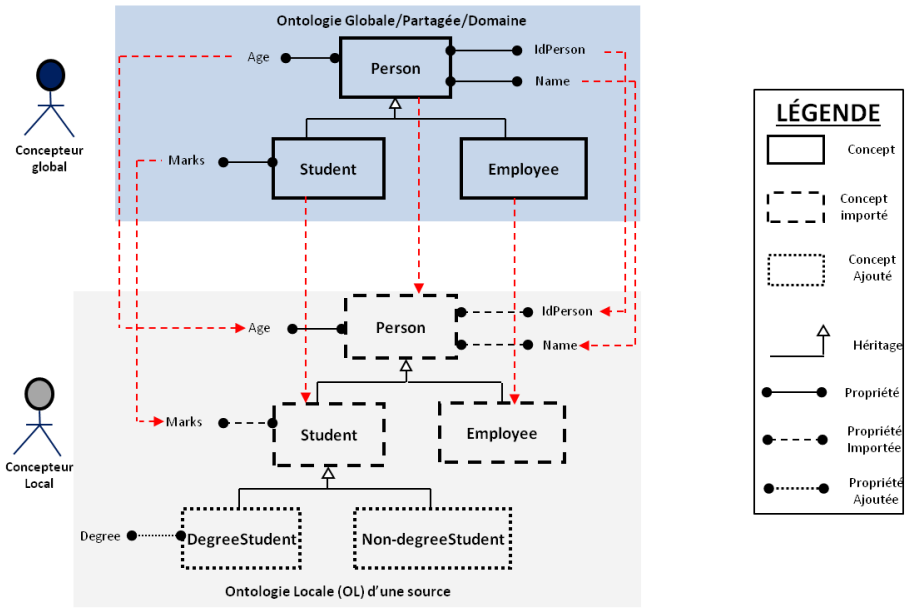

FIGURE 4.5 – Exemple d'extraction d'une ontologie locale qui englobe l'ontologie partagée

S_i est extrait de l'ontologie partagé et l'autre sous-ensemble est défini pour étendre l'ontologie locale par l'ajout de nouvelles propriétés (par relation de subsomption).

- *$Ref(C)_{OLi} \supseteq Ref(C)_{OP}$: un sous-ensemble des fonctions sur les classes de l'ontologie locale i de la source S_i est extrait de l'ontologie partagée et l'autre sous-ensemble est défini pour les concepts qui étendent l'ontologie locale.*
- *$Ref(R)_{OLi} \supseteq Ref(R)_{OP}$: un sous-ensemble des fonctions sur les propriétés de l'ontologie locale i de la source S_i est extrait de l'ontologie partagée et l'autre sous-ensemble est défini pour les rôles qui étendent l'ontologie locale.*

4.1.2 Hypothèse 2

La deuxième hypothèse porte sur l'existence d'une ontologie linguistique multilingue (OL) qui définit de manière consensuelle tous les termes utilisés par les différents partenaires sources participant au système. Ces partenaires peuvent être de différentes origines et peuvent donc utiliser des langues différentes (Anglais, Français, Espagnol, etc.) dans l'expression de leurs besoins locaux.

Dans ce scénario, le vocabulaire est uniformisé par le donneur d'ordre. Une seule ontologie linguistique/multilingue est utilisée par tous les partenaires (sources) afin de définir tous les termes utilisés dans l'expression des besoins au niveau des sources. Cette ontologie linguistique permet par la suite d'extraire les relations lexicales entre les termes telles que les synonymes, les antonymes, etc. Ainsi, chaque concept utilisé dans l'expression d'un besoin est extrait à partir de

l'ontologie conceptuelle partagée, qui elle-même utilise les termes de l'ontologie linguistique partagée. Nous rappelons que chaque concept de l'ontologie conceptuelle peut être associé à un ou plusieurs termes (voir la section cf.4.2 du chapitre 2) comme illustré dans l'exemple de la figure 4.6.

L'élaboration des termes/concepts de l'ontologie locale peut se faire en commençant par la sélection des concepts de l'ontologie conceptuelle qui référence un ou plusieurs termes de l'ontologie linguistique, ou en sélectionnant les termes de l'ontologie linguistique qui référence un concept de l'ontologie conceptuelle partagée. L'extraction de l'ontologie linguistique se fait simplement par extraction des termes et relations lexicales nécessaires à l'expression des besoins locaux.

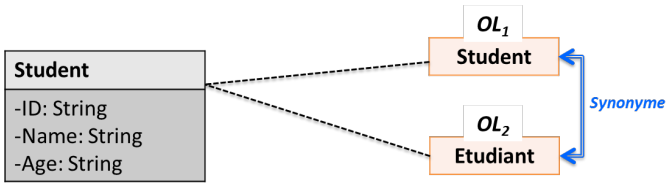

FIGURE 4.6 – Liaisons entre la couche linguistique et conceptuelle dans le modèle en oignon

Les deux couches ontologiques servent ainsi de vocabulaire précis et partagé pour définir la sémantique des besoins à l'aide de concepts et de termes consensuels et référençables.

Dans ce scénario d'ontologie partagée, les mappings sont définis dès la création des ontologies locales. L'ontologie intégrante est dans ce cas soit exactement similaire à l'ontologie partagée, soit définie comme un fragment de cette ontologie.

4.2 Scénario d'intégration "ontologies multiples"

Dans le deuxième scénario, chaque source définit son ontologie locale indépendamment des autres sources. Ce scénario ne nécessite aucune hypothèse particulière mais un processus de *matching ontologique* doit être défini afin d'unifier les ontologies locales.

Le processus de matching ontologique reçoit en entrée deux ou plusieurs ontologies et génère comme sortie un ensemble de correspondances sémantiques entre les entités des ontologies en cours de traitement [54, 88] (voir figure 4.7).

Notre travail est basé sur le processus de matching ontologique dénoté PBW (en anglais: Precision Based Weighting) proposé dans [21] (l'auteur avec lequel nous avons consigné un travail dans [46]). Contrairement à la plupart des systèmes de matching ontologiques existants qui sont basés sur le calcul de similarités et leur combinaison, cette proposition permet d'agréger différents matchers (conceptuels et linguistiques) en leur assignant des poids de façon automatique, afin de réaliser un meilleur alignement.

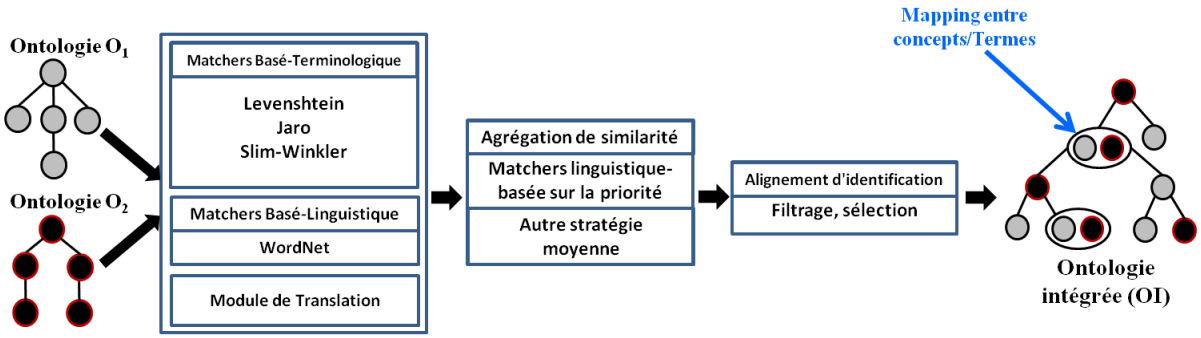

FIGURE 4.7 – Système de matching ontologique

Le processus de matching commence par le chargement des ontologies conceptuelles et linguistiques locales des sources en tant qu'entrées et calcule un ensemble de mappings M entre les ontologies conceptuelles locales et un autre M' entre les ontologies linguistiques. En sortie, le processus fusionne les ontologies conceptuelles locales des sources en une seule ontologie conceptuelle intégrée utilisant M et fusionne les ontologies linguistiques des sources en entrées en une seule ontologie linguistique intégrée utilisant M'.

Nous décrivons ci-dessous le mécanisme de matching.

4.2.1 Matching ontologique

Nous décrivons ci-dessous le mécanisme de matching qui est illustré dans la figure 4.7. Ce mécanisme permet d'effectuer une comparaison par paire entre les ontologies afin de matcher toutes les ontologies locales. Ce mécanisme inclut plusieurs étapes détaillées dans ce qui suit.

4.2.1.1 Étape 1 : traitement multilingue Les ontologies locales peuvent être décrites dans différentes langues. La gestion des conflits dûs au multilinguisme des termes utilisés est réalisée en utilisant un traducteur automatique basé sur le service Yandex ATL [43]. Yandex ATL est un service web fourni par *Yandex* [44] qui permet de traduire automatiquement un texte d'une langue vers une autre langue. Les résultats du traducteur *Yandex* ont été évalués par la campagne d'évaluation OAEI sur différents systèmes (CroLOM [87] et SimCat [89]). Ces campagnes d'évaluation ont montré les bons résultats de ce traducteur.

43. https://translate.yandex.com/?text=administrarlang=es-en
44. https://yandex.com/company/technologies/translation/

Exemple 2

Pour illustrer l'identification des mappings entre les concepts des ontologies conceptuelles et linguistiques locales, décrites ci-dessus, nous définissons les deux besoins (en Anglais et en Français) :

B_2: "The faculties enable students to manage courses"

B_3: "L'université permet aux étudiants de gérer les cours"

Les termes traduits (en anglais) sont les noms des concepts, attributs et relations, des différentes ontologies conceptuelles et linguistiques. La traduction des entités ontologiques en langue anglaise est nécessaire parce que le matcher linguistique (basé sur *WordNet*), le plus utilisé dans la pratique, est en langue anglaise. Ce matcher permet le calcul de similarité entre les mots.

La définition de ces besoins se fait par deux partenaires différents et donne lieu à deux ontologies (conceptuelle et linguistique) différentes qu'il faut intégrer dans une ontologie intégrante. La première étape consiste à traduire les termes utilisés, afin de faciliter leur comparaison, par exemple *université* est traduit en *Institute* et *permettre (permet)* est traduit en *allow*.

4.2.1.2 Étape 2 : matching terminologique linguistique

Le processus de matching calcule la similarité entre les entités normalisées en utilisant un matcher terminologique et linguistique.

Le matcher linguistique se base sur la similarité entre les termes des entités, par l'utilisation de techniques de traitement automatique du langage naturel ainsi que de moyens externes comme les dictionnaires. Le matcher terminologique se base sur le calcul de similarité entre des chaines de caractères (les noms des concepts, attributs, instances, commentaires, relations et labels).

Trois mesures de similarité sont combinées afin de gérer l'hétérogénéité terminologique, à savoir :

- **la distance de *levenshtein* (levenshtein-distance)** [101] : aussi nommée *distance d'édition*, elle reflète une distance mathématique entre deux chaînes de caractères. Elle prend la valeur (égale) au nombre minimal de caractères qu'il faut insérer, remplacer ou supprimer pour passer d'une chaîne vers l'autre. La valeur de cette distance dépend du nombre de caractères différents entre deux chaines de caractères Ch_1 et Ch_2 où la valeur augmente proportionnellement au nombre de différences entre les deux chaînes. Par exemple, pour les deux chaines Ch_1='*similarity*' et Ch_2='*similarité*', la valeur de la distance de levenshtein (LD) est LD(Ch_1, Ch_2) = 1, car les deux chaines de caractères contiennent un seul caractère *i* (de Ch_2) différent du *Y* de Ch_1.
- **la distance de *Jaro* (Jaro-distance)** [80] : cette mesure est fondée essentiellement sur l'ordre des caractères communs entre deux chaines de caractères Ch_1 et Ch_2. Cette mesure calcule également le coût de transposition des caractères communs entre deux chaines Ch_1 et Ch_2. Le coût de transposition entre deux chaines de caractères est généré quand

deux caractères communs du même rang dans les deux chaines Ch_1 et Ch_2 (i-ème caractère commun de Ch_1 et le i-ème caractère commun de Ch_2) ne sont pas égaux. La distance de Jaro est mesurée par la formule suivante : $d_j = \frac{1}{3}\ (\frac{m}{|Ch_1|} + \frac{m}{|Ch_2|} + \frac{m-t}{m})$, où :

- $|Ch_i|$: représente la longueur de la chaîne de caractères Ch_i.
- m : représente le nombre de caractères correspondants (voir l'exemple ci-dessous).
- t : représente le nombre de transpositions.
- pour considérer que deux caractères x et y appartenant respectivement à Ch_1 et Ch_2 sont identiques, il faut que la différence de leurs positions (éloignements) dans les deux chaines Ch_1 et Ch_2 ne dépasse pas la valeur suivante: $\lfloor \frac{max(|Ch_1|,|Ch_2|)}{2} \rfloor - 1$.
- le calcul du nombre de transpositions se fait en comparant le i-ème caractère de la chaine Ch_1 avec le i-ème caractère de la chaine Ch_2. Le nombre de transpositions est représenté par le nombre de fois où ces caractères sont différents, divisé par deux.

Exemple 3

Nous expliquons le fonctionnement de la distance de Jaro avec les deux chaines de caractères suivantes : Ch_1='course' et Ch_2='cousre'. La table de correspondance dressée ci-dessous, représente les cases de correspondance identique avec le 1 et 0 sinon.

	c	o	u	r	s	e
c	1	0	0	0	0	0
o	0	1	0	0	0	0
u	0	0	1	0	0	0
s	0	0	0	0	1	0
r	0	0	0	1	0	0
e	0	0	0	0	0	1

- L'éloignement maximal= $\lfloor \frac{max(|Ch_1|,|Ch_2|)}{2} \rfloor$ - 1 = $\lfloor \frac{max(6,6)}{2} \rfloor$ - 1 = $\frac{6}{2}$ - 1= 2.
- m=6 (représente le nombre de 1 dans la table), $|Ch_1|$= 6 (longueur de la chaine de caractère Ch_1), $|Ch_2|$=6 (longueur de la chaine de caractère Ch_2).
- Ces deux chaines de caractères Ch_1 et Ch_2 représentent respectivement les caractères correspondants {c,o,u,r,s,e} et {c,o,u,s,r,e}. Selon l'ordre des caractères dans les deux chaines on peut découvrir qu'il existe deux couples de caractères(r/s et s/r) différents (deux demi-transpositions), donc t=$\frac{2}{2}$+1.
- On peut calculer la distance de Jaro, $d_j = \frac{1}{3}\ (\frac{6}{6} + \frac{6}{6} + \frac{6-1}{6}) = 0,944$.

Notons qu'il existe une version étendue de la similitude de *Jaro*, nommée la similitude *JaAro-Winkler* [159], utilisée principalement dans la détection de doublons. Elle prend en considération le préfixe commun entre deux chaines de caractères Ch_1 et Ch_2.

- **la distance de *SLIM-Winkler* (SLIM-Winkler-distance)** [21] : c'est une extension de la mesure de distance proposée par *Winkler* pour la métrique de *Jaro*.

Afin de couvrir une hétérogénéité terminologique maximale et de détecter des concepts synonymes et antonymes, nous avons utilisé *WordNet* comme un dictionnaire externe.

Exemple 4

Considérons les deux exemples de besoins précédents décrits comme suit :

- *B_2: "The faculty enables students to manage course"*
- *B_3: L'université permet aux étudiants de gérer les cours"*

Après traduction en anglais, le système de matching calcule la similitude entre 'institute' (traduit de 'université') et 'faculty', 'enable' et 'allow' (traduit de 'permettre'). Cette étape a recours à l'ontologie Wordnet où une relation de synonymie est détectée entre 'institute' et 'faculty' et entre 'enable' et 'permettre'. Afin de faciliter le processus de matching, nous comparons les termes et concepts de même type (nom et verbe).

Le système renvoie un mapping M entre ces termes. Ces mappings seront également exploités lors de l'intégration des besoins comme cela sera expliqué dans les prochains chapitres.

4.2.1.3 Étape 3 : matching conceptuel Pour réduire l'hétérogénéité conceptuelle, nous avons utilisé les méthodes structurelles [54], qui sont divisées en deux catégories :

- Des techniques basées sur les contraintes internes (structure interne) : ces techniques sont basées sur le calcul de la similarité selon la structure interne des entités (leurs propriétés) [54]. Ce calcul se base sur la comparaison du domaine des entités, co-domaine, les cardinalités, le nombre d'attributs et les types de données qui composent les concepts. Par exemple, on peut considérer que deux entités (propriétés) sont similaires s'ils présentent une proximité dans les valeurs de données affectées à ces deux propriétés. En reprenant les deux besoins précédents B_2 et B_3, pour le calcul de similarité entre les autres entités '*student*', '*manage*' et '*course*' appartenant au besoin B_2, et les entités '*student*', '*manage*' et '*course*' appartenant au besoin B_3, le mécanisme de matching conceptuel compare les attributs (*id, name, etc.*) des concepts (deux par deux) en utilisant les matchers de chaînes et de structure interne. Ainsi, les deux concepts "institute" et "faculty" ont des attributs similaires (respectivement "Nom" et "Name") de même type et de cardinalité 1-1.
- Des techniques basées sur les structures externes : elles permettent de calculer la similarité entre les entités sur la base des relations entre les entités elle-mêmes. Ces techniques sont divisées en quatre sous-techniques [54] :
  - *Les techniques à base de taxonomie :* dans ces techniques, l'ontologie est considérée comme un graphe. Ces techniques utilisent des algorithmes de graphe pour le calcul de similarité entre les noeuds (représentant les concepts ontologiques) en analysant les relations de type *is-a*. Par exemple, si deux concepts C_1 et C_2 sont considérés comme similaires alors les concepts voisins liés par une relation de type *is-a* à ces deux concepts sont également considérés similaires [55].
  - *Les techniques à base de graphe :* dans ces techniques, la structure de l'ontologie est considéré également comme un graphe (les concepts ontologiques sont représentés par des noeuds et les propriétés ontologiques par des arcs). Des algorithmes de graphe sont

appliqués afin de calculer la similarité entre les noeuds [54].

- *Les techniques à base de référentiel des structures :* ces techniques utilisent un référentiel pour le calcul de la similarité entre les entités. L'enregistrement des ontologies ainsi que des entités similaires permet d'obtenir ce référentiel. L'objectif de ces techniques est de permettre : (1) la réutilisation de l'alignement ontologique (l'alignement de référentiel) dans le cas d'existence d'ontologies similaires pour chaque ontologie en entrée, (2) initier une nouvelle tâche d'alignement entre l'ontologie similaire et l'ontologie en entrée [166].
- *Les techniques par réutilisation de l'alignement :* basées sur la réutilisation d'un alignment existant, ces techniques permettent de gagner du temps, surtout pour les ontologies qui décrivent le même domaine [166].

Le système renvoie un mapping M entre ces concepts. Ces mappings seront exploités lors de l'intégration des besoins comme il sera expliqué au troisième chapitre de contribution.

4.2.1.4 Étape 4 : Fusion des ontologies Une fois les mesures de similarité calculées, une matrice de similarité est obtenue pour chaque matcher. Le processus de matching combine ensuite, dans une matrice unifiée, les matrices obtenues par chaque matcher. Cette matrice unifiée est obtenue en utilisant une stratégie de combinaison qui donne la priorité à la similarité calculée par *WordNet*. En d'autres termes, la similitude est conservée, si la valeur de similarité calculée à l'aide de *WordNet* est supérieure à la valeur de similarité calculée à l'aide des algorithmes de matching. Autrement, la similarité est retenue en utilisant la méthode d'agrégation moyenne.

Enfin, le processus de matching applique un filtre sur les valeurs de similitude combinées afin de sélectionner la plus pertinente. Ceci est réalisé en utilisant la stratégie maximale avec un seuil. En d'autres termes, le processus sélectionne la valeur de similarité maximale qui est supérieure à un seuil donné pour chaque ligne de la matrice de similarité unifiée. Cette similarité permet d'établir des mapping M et M' entre les ontologies.

Notons qu'un problème important pour les systèmes de matching basés sur les ontologies est la génération de mauvaises correspondances sémantiques, qui peuvent conduire à l'incohérence de l'ontologie globale intégrante. Pour éviter ce problème, nous avons utilisé le raisonneur *Pellet* [45] pour détecter ces mauvaises correspondances et les supprimer.

Concernant les concepts où notre système n'arrive pas à trouver des correspondances, le mécanisme de matching affiche les résultats des concepts appariés (matchés) et non matchés. Si le mécanisme de matching est considéré comme interactif, l'utilisateur peut sélectionner les concepts qui sont équivalents à partir de la liste des concepts non matchés. Cette validation est importante car l'intégration des besoins repose sur la qualité de l'ontologie intégrante.

Comme expliqué précédemment, le processus de matching (conceptuel et linguistique) compare les ontologies deux par deux, et génère à chaque étape deux ontologies (conceptuelle et linguistique) intégrantes sur la base des mappings calculés. L'ontologie intégrante est générée

45. https://www.w3.org/2001/sw/wiki/Pellet

par *fusion* des ontologies locales avec les mappings calculés. Pour les concepts et les termes qui ne sont pas appariés, ils sont définis dans l'ontologie intégrante tels qu'ils sont définis dans les ontologies locales. Le système de matching utilise le plugin *Prompt* [46], afin de fusionner les ontologies conceptuelles et linguistiques.

Ce mécanisme de matching (PBW)a également été évalué par la campagne d'évaluation OAEI [47] et a montré des résultats prometteurs.

4.2.2 Algorithme de génération de l'ontologie intégrante

Avant de présenter l'algorithme de matching global des ontologies locales, nous définissons quelques fonctions et symboles utilisés dans l'algorithme :

- OCI : représente l'ontologie conceptuelle intégrante.
- OLI : représente l'ontologie linguistique intégrante.
- C : c'est une entité qui peut être :
  - un concept ou propriété de l'ontologie conceptuelle locale oc_i.
  - un terme de l'ontologie linguistique locale ol_i.
- ***Translate (C, C')*** : c'est une fonction qui permet la traduction en ***Anglais*** des entités *C* en *C'*.
- **FunctionMergingConceptualOntologies ((oc_1, oc_2..., oc_n), M) :** c'est une fonction prédéfinie qui utilise le plugin *Prompt* pour fusionner les ontologies conceptuelles sources (oc_1, oc_2..., oc_n) avec l'utilisation des mappings M.
- **FunctionMergingLinguisticOntologies ((ol_1, ol_2,..., ol_n), M') :** c'est une fonction qui utilise le plugin *Prompt* pour fusionner les ontologies linguistiques sources (ol_1, ol_2,..., ol_n) avec l'utilisation des mappings M'.

Les étapes du processus de matching sont décrites dans l'algorithme 1.

5 Validation de l'approche : études de cas

Afin de valider l'approche proposée pour la construction de l'ontologie intégrante, nous avons considéré deux études de cas suivant les deux scénarios établis : ontologie partagée et ontologies multiples. Nous utiliserons ces deux études de cas dans les prochains chapitres pour l'évaluation des différentes étapes de notre approche.

46. http://protegewiki.stanford.edu/wiki/PROMPT
47. http://oaei.ontologymatching.org/

Input: les ontologies locales OC (oc_1, oc_2..., oc_n) et OL (ol_1, ol_2..., ol_n);
Output: *l'ontologie intégrante*;
pour *Chaque entité C (concepts/termes)* **faire**
| ***Translate (C, C')***;
fin
pour *Chaque couple d'entités C et C' de même type (concepts/termes)* **faire**
| ***Calcul similarité selon matcher terminologique***
| - Compute $Sim_{Levenshtein}$(C; C')
| - Compute Sim_{Jaro}(C; C')
| - Compute $Sim_{Slim-Winkler}$(C; C')
| ***Calcul similarité selon matcher linguistique***
| - Compute $Sim_{WordNet}$(C; C')
| ***Calcul similarité selon la structure interne***
| - Compute $Sim_{equality}$(C; C')
| ***Calcul similarité finale*** Sim_{Fin}
| **si** $Sim_{WordNet}(C; C') \geq Max(Sim_{Levenshtein}(C; C'), Sim_{Jaro}(C; C'), Sim_{Slim-Winkler}(C; C'))$ **alors**
| | $Sim_{Fin} = Sim_{WordNet}(C_{k<R_j<T_j^p,type>>}; C)$;
| | **sinon**
| | | Sim_{Fin}(C; C') = ($Sim_{Levenshtein}$(C; C') + Sim_{Jaro}(C; C') + $Sim_{Slim-Winkler}$(C; C') + $Sim_{WordNet}$(C; C')) / 4
| | **fin**
| **fin**
| **si** $Sim_{Fin}(C; C') \geq$ *S1: Threshold,* $Sim_{Fin}(C; C') \geq$ *S2: Threshold* **alors**
| | **si** *le couple C et C' sont des entités des ontologies conceptuelles sources* oc_i *et* oc_j **alors**
| | | **Return** M = M ∪ (C ≡ C') ;
| | **fin**
| | **si** *le couple C et C' sont des entités, des ontologies linguistiques sources* ol_i *et* ol_j **alors**
| | | **Return** M' = M' ∪ (C ≡ C');
| | **fin**
| **fin**
fin
FunctionMergingConceptualOntologies (OC,M)
FunctionMergingLinguisticOntologies (OL, M')

Algorithm 1: Algorithme de matching des ontologies locales

5.1 Étude de cas 1 : ontologie partagée

Nous considérons dans cette première étude de cas l'exemple d'un grand projet dans le domaine universitaire ayant plusieurs sous-projets (sous-systèmes : universités, instituts, écoles, etc.). Ces sous-projets sont répartis dans différentes régions. Les partenaires de chaque sous-projet exprime leurs besoins localement, en utilisant un formalisme (langage de modélisation des besoins) et un vocabulaire (concepts et termes utilisés lors de l'expression des besoins locaux).

Chaque source (SB_i) est peuplée localement par des besoins définis par le système de gestion des cours (en anglais : courses management system (CMS [48])). Nous considérons pour ce scénario trois sources de besoins (SB_1, SB_2 et SB_3), où chaque source a son propre concepteur de besoins (CB_1 : concepteur anglais, CB_2 : concepteur français et CB_3 : concepteur espagnol).

Ces besoins sont exprimés au niveau de chaque source à l'aide d'une ontologie conceptuelle locale référençant une ontologie globale conceptuelle (ontologie *Univ-Bench* [49] du benchmark ontologique LUBM [50]) et une ontologie globale linguistique (Ontologie *EuroWordNet* [51]). Un extrait de l'ontologie *Univ-Bench* est illustré en figure 4.8. Les ontologies *Univ-Bench* et *Wordnet* représentent les deux couches conceptuelle et linguistique de l'ontologie partagée dans cette étude de cas.

Le banc d'essai LUBM est utilisé généralement pour évaluer les systèmes avec différentes capacités de raisonnement et mécanismes de stockage. L'ontologie *EuroWordNet* [52] est basée sur le modèle de l'ontologie *Wordnet* [53]. *Wordnet* est définie en utilisant une seule langue ; le but du projet *EuroWordNet* est alors de créer un réseau sémantique entre les ontologies *WordNet* (figure 4.9) avec des liens sémantiques inter-langues [153]. L'avantage important de cette ressource ontologique est la possibilité d'attacher des concepts d'une langue à l'autre, par exemple L'indice interlinguistique *Inter Lingual Index* (ILI) représente un catalogue de concepts sur lesquels chaque langue peut pointer [152].

Les besoins présentés ci-dessous sont des exemples de besoins définis et adaptés à partir du système de gestion de cours (CMS):

```
/Source 1: Besoins définis selon un modèle de but par le concepteur anglais:
   B1:    The system shall allow student to create an account
   B3:    The system shall allow student to register in courses.
   B17 :  The system shall allow students to create teams.
   B19:   The system shall allow lecturers to create teams.
   B48 :  The system shall allow lecturers to create courses.

/Source 2: Besoins définis en modèle MCT de merise par le concepteur français:
   B48' :  Le système doit permettre aux conférenciers de créer les cours.
   B97' :  Le système doit permettre que l'administration de gérer les cours.
```

48. http://wwwhome.cs.utwente.nl/ goknila/sosym/.
49. http://swat.cse.lehigh.edu/onto/univ-bench.owl
50. http://swat.cse.lehigh.edu/projects/lubm/
51. http://www.illc.uva.nl/EuroWordNet/
52. http://www.illc.uva.nl/EuroWordNet/sample.html
53. https://wordnet.princeton.edu/

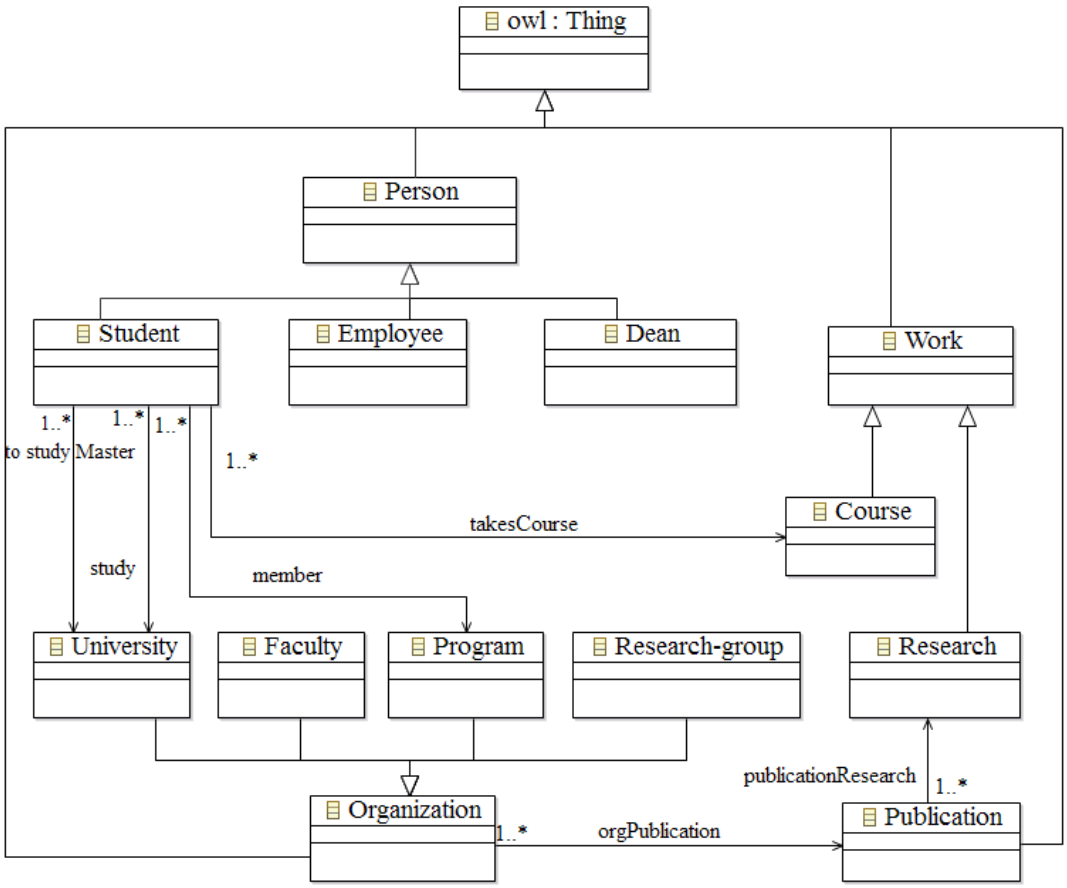

FIGURE 4.8 – Un extrait de l'ontologie Univ-Bench proposée par le benchmark LUBM.

```
   B99' :  Le système doit permettre uniquement à l'administration de supprimer des cours.
/Source 3: Besoins définis en modèle des cas d'utilisations par le concepteur espagnol:
   B68' :  El sistema permitirá que los profesores para gestionar la información del curso dinámico.
   B97'':  El sistema deberá permitir al administrador gestionar cursos.
   B98' :  El sistema deberá permitir que sólo el administración para crear nuevos cursos.
```

Les termes et concepts utilisés pour exprimer ces besoins sont définis dans une ontologie locale à chaque source. Les mappings entre les ontologies locales et l'ontologie partagée sont déjà définis. L'intégration des besoins des partenaires passe d'abord par l'intégration des ontologies locales qui peuvent présenter différentes hétérogénéités syntaxiques et sémantiques. L'intégration des ontologies permet d'unifier le vocabulaire utilisé dans l'entrepôt cible.

Par exemple, si nous considérons uniquement les exemples de besoins cités ci-dessus, l'ontologie partagée correspond au scénario où les besoins sont complètement définis par l'ontologie partagée. L'ontologie intégrante est donc définie comme un fragment de l'ontologie partagée (partie de *Univ-Bench* et de *Wordnet*). Elle est extraite à partir de l'ontologie partagée en utilisant un mécanisme de modularité.

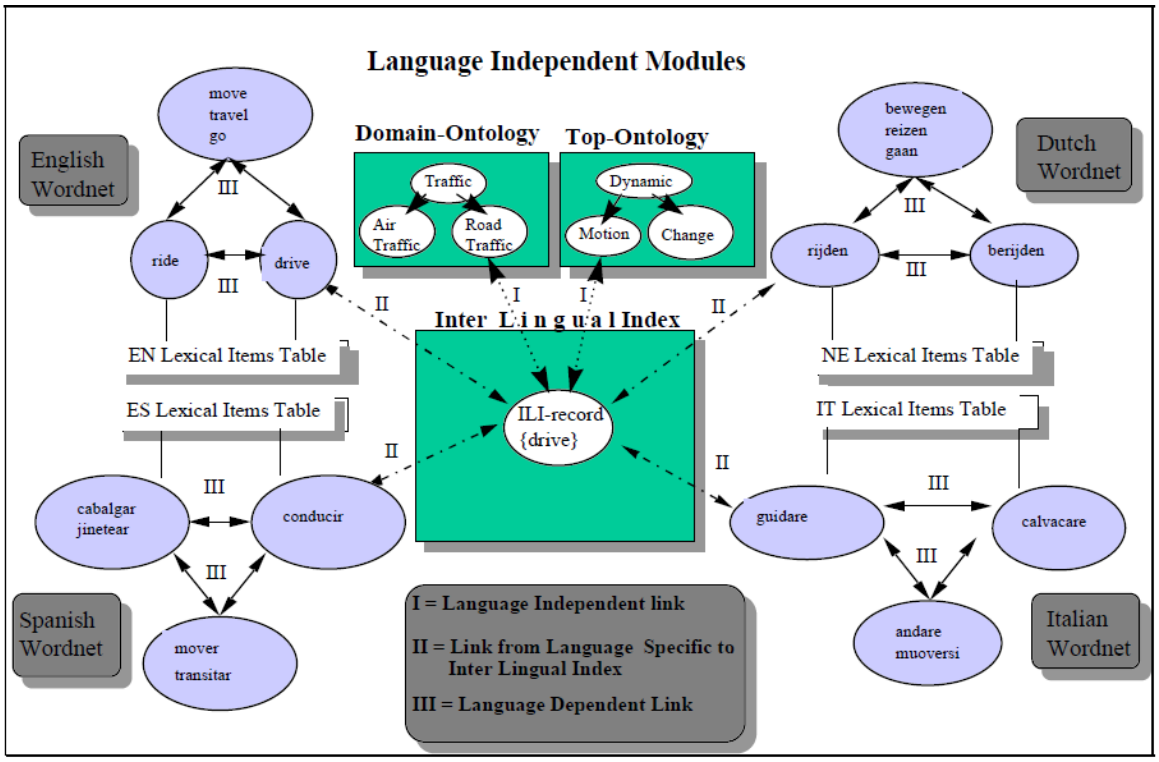

FIGURE 4.9 – L'architecture globale de l'ontologie *EuroWordNet* [153]

5.2 Étude de cas 2 : ontologies multiples

Le scénario de cette seconde étude de cas est illustré dans la figure 4.10). Il est basé sur les entrées suivantes :

- nous considérons comme sources trois systèmes de soumission de conférence : le système de gestion de conférences académique de Microsoft (CMT [54] : Microsoft's Academic Conference Management Service), le service de gestion des conférences (EDAS [55] : Managing conférence service) et le système de gestion des conférences (ConfOf [56] : Conference Management Tool).
- les trois sources utilisent des ontologies linguistiques locales issues respectivement de : *Wordnet* anglais, *Wordnet* espagnol et *Wordnet* français. Ces ontologies sont obtenues à partir du site *Open Multilingual Wordnet* [57].
- les trois sources utilisent des ontologies conceptuelles locales obtenues à partir du projet

54. https://cmt.research.microsoft.com/cmt/default.html
55. https://edas.info/doc/conference.html
56. https://www.conftool.net/en/index.html
57. http://compling.hss.ntu.edu.sg/omw/

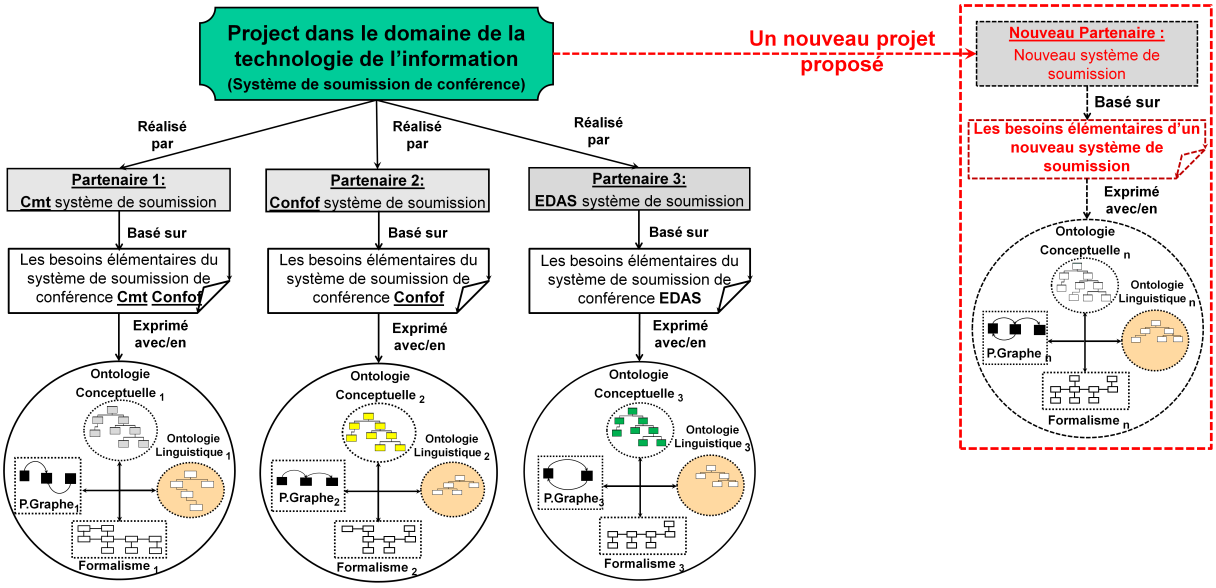

FIGURE 4.10 – Étude de cas 2 : intégration d'ontologies d'un système de soumission de conférences

MultiFarm [58] utilisé par la campagne d'évaluation OAEI2015 [59]. La description de chaque ontologie est donnée dans le site de la conférence OAEI2015 : `http://oaei.ontologymatching.org/2015/conference/index.html`.

– nous supposons que les trois sources définissent un ensemble de besoins (que nous avons formulé) décrits selon les termes et concepts définis dans leurs ontologies locales (conceptuelle et linguistique). Nous supposons que les trois sources utilisent des formalismes de modélisation de besoins différents pour exprimer leurs besoins locaux.

Les besoins présentés ci-dessous sont des exemples de besoins définis pour ce scénario :

```
/Source 1: Besoins définis selon un modèle de but par le concepteur anglais:

- B50 : The system shall allow the authors to login an account
- B51 : The system shall allow the authors to register the paper
- B52 : The system shall allow the  authors to add authors
- B53 : The system shall allow the authors to submit the paper

/Source 2: Besoins définis en modèle MCT de merise par le concepteur espagnol :

- B38 : El sistema permitirá al usuario configurar la contraseña.
- B67 : El sistema poder definir una presentación;
- B68 : El sistema poder administrar presentación;
- B96 : El sistema poder  crear sesiones (sessions for presentations);
- B70 : El sistema poder administrar sesiones (sessions for presentations);

/Source 3: Besoins définis en modèle des cas d'utilisations par le concepteur français :

- B11 : Le système doit permettre à l'utilisateur d'entrer son adresse;
- B12 : Le système doit permettre à l'utilisateur d'entrer son adresse e-mail;
- B14 : Le système doit permettre à l'utilisateur d'utiliser un compte pour plusieurs soumissions;
- B16 : Le système doit permettre au PC membre de d'utiliser le compte pour la soumission de papier;
- B17 : Le système doit permettre l'examinateur d'utiliser le compte pour la soumission de papier;
```

58. http://oaei.ontologymatching.org/2015/multifarm/index.html
59. http://oaei.ontologymatching.org/2015

L'ontologie intégrante doit être définie dans ce cas à partir des ontologies locales conceptuelles et linguistiques (indépendantes). Cette ontologie a été construite en suivant le mécanisme de matching défini précédemment. Nous évaluons les performances de ce mécanisme par une série d'expérimentations.

Nos évaluations ont été effectuées sur un ordinateur portable (HP Elite-Book 840 G2) avec un processeur Intel(R) CoreTM i5-5200U CPU 2.20 GHZ, 8 GB de RAM, un disque dur de 500 GB. Nous avons utilisé Windows10 64 bits.

5.2.1 Performances du mécanisme de matching

Nous avons évalué les performances du mécanisme de matching PBW en utilisant des métriques standards qui sont : la précision (*precision*), le rappel (*recall*) et la F-mesure (*F-measure*) adoptées dans la campagne d'évaluation OAEI. Ces mesures évaluent la performance du processus de matching par comparaison des mappings générés avec les mappings d'alignement de référence (fournis par OAEI). La Figure 4.11 résume les résultats obtenus.

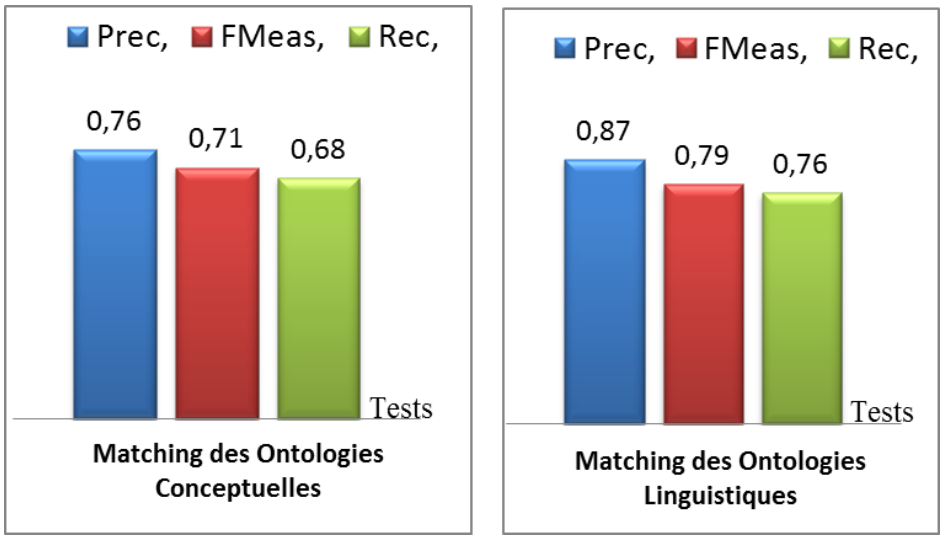

FIGURE 4.11 – Les résultats de Matching

Les résultats obtenus pour le matching des ontologies linguistiques sont très satisfaisants et atteignent une précision supérieur à 87% et un rappel égale à 76%. Ces résultat démontrent que selon les besoins exprimés et les ontologies locales qui les définissent, le mécanisme de matching réussit à découvrir un nombre important de mappings entre les termes des besoins exprimés. Ces résultats s'expliquent aussi par l'utilisation du dictionnaire *WordNet* après traduction, qui permet d'augmenter le nombre de mappings découverts. Cependant, dans certains

cas, le traducteur automatique renvoie un mot de remplacement qui n'est pas couvert par *WordNet* et qui n'a donc pas pu être découvert par le système de matching.

Les résultats obtenus pour le matching des ontologies conceptuelles donnent également des résultats satisfaisants et atteignent un rappel supérieur à 68% et une précision de 76%.

Il est intéressant de noter que nous pouvons augmenter la précision (*precision*) en utilisant les mécanismes de raisonnement proposés dans [139] afin d'éliminer les concepts qui causent l'inconstance de l'ontologie intégrante. Nous pouvons également augmenter le rappel (*recall*) en considérant des techniques de matching supplémentaires telles que l'apprentissage automatique ou les techniques fondées sur les connaissances. Le mécanisme de matching est donc appelé à être amélioré par diverses techniques, afin de permettre une meilleure intégration des besoins.

6 Conclusion

L'approche proposée dans ce chapitre vise à construire l'ontologie intégrante unifiant les ontologies locales définies pour exprimer les besoins locaux des partenaires participants au système d'intégration. Notre approche construit l'ontologie intégrante en considérant les deux couches ontologiques : linguistique et conceptuelle. De plus, la construction de cette ontologie est étudiées pour deux scénarios issus des deux structures ontologiques existantes : ontologie partagée et ontologies multiples. Ce dernier scénario est souvent ignoré dans les travaux existants. Dans le premier scénario, l'ontologie intégrante est définie à partie de l'ontologie partagée. Dans le second scénario, le mécanisme de matching PBW entre les ontologies locales a été exploité et adapté à notre contexte d'intégration des besoins. Le système de matching a été évalué selon trois mesures : la précision, le rappel et la F-mesure. Ces performances sont importantes car la qualité de l'ontologie intégrante en dépend.

Dans le chapitre suivant, nous présentons notre deuxième contribution qui consiste à fournir le modèle cible de l'entrepôt de besoins à définir.

Chapitre 5

Un Modèle Multidimensionnel pour l'Entrepôt des Besoins Fonctionnels

Sommaire

1 Introduction 109

2 Présentation des sources 110

2.1 Formalismes sources et ontologies locales 110

2.1.1 Connexion du formalisme MCT avec l'ontologie local (OntoMCT) 111

2.1.2 Connexion du formalisme des buts avec l'ontologie locale (OntoGoal) 111

2.1.3 Connexion du formalisme des cas d'utilisation avec l'ontologie locale (OntoUseCase) 111

3 Le modèle pivot proposé 114

4 Le modèle conceptuel de l'ESBF 114

4.1 Révision du modèle pivot 114

4.2 Modélisation multidimensionnelle du schéma de l'entrepôt . . . 119

4.2.1 Étendre le MPE avec les dimensions temps et localisation 119

4.2.2 Définition du schéma de l'entrepôt de besoins 121

5 Transformation des besoins sources vers le modèle de l'entrepôt . . . 123

6 Connexion des modèles de besoin et ontologiques 125

7 Conclusion 127

Résumé. Dans ce chapitre, nous proposons la construction du schéma d'un entrepôt de besoins en deux étapes permettent : (i) l'unification des formalismes ou langages de modélisation représentant les besoins des différentes sources participants au système, et (ii) la modélisation multidimensionnelle du schéma permettant par la suite l'analyse OLAP des besoins unifiés. Plusieurs formalismes peuvent être utilisés par les sources selon les compétences ou les préférences des partenaires du projet d'entreposage. La gestion de ces hétérogénéités se base sur un formalisme pivot de besoins proposé au sein de notre laboratoire dans le cadre de la thèse d'Ilyes Boukhari [27]. Nous prenons ce formalisme comme base de travail, et nous l'enrichissons pour avoir une modélisation détaillée de chaque besoin. Dans la deuxième étape, et ce afin de permettre une analyse OLAP des besoins, nous organisons le modèle pivot selon une modélisation multidimensionnelle, et nous l'enrichissons par deux nouvelles entités (localisation et temps). Enfin, les relations entre les besoins doivent être définies et identifiées. Cette contribution nous permet d'obtenir le modèle conceptuel multidimensionnel de l'entrepôt sémantique des besoins fonctionnels (ESBF) à définir. Ce modèle sera ensuite connecté à l'ontologie intégrante définie au chapitre précédent afin de permettre une définition sémantique de l'ensemble des besoins intégrés.

1 Introduction

La conception d'un projet d'entreposage passe par la définition du modèle conceptuel de l'entrepôt en question. Ce modèle contient les entités que l'entrepôt doit représenter et gérer. Suivant la même démarche, l'entreposage des besoins peut facilement emprunter les solutions proposées et approuvées pour l'intégration des sources de données. La définition de ce modèle passe ainsi par l'analyse des sources et/ou des besoins informationnels OLAP. Nous considérons ces deux sources dans notre approche.

Nous rappelons que notre approche globale (cf. figure 4.1, chapitre 4, section 4.1) prend comme entrées des sources de besoins formalisées selon un formalisme (langage de modélisation) donné, et utilisant les concepts d'une ontologie locale conceptuelle et les termes d'une ontologie locale linguistique. Ces ontologies référencent une ontologie intégrante (conceptuelle et linguistique) qui couvre et unifie l'ensemble des ontologies locales. Le modèle de l'entrepôt doit permettre d'unifier les formalismes sources et de représenter les besoins à entreposer.

Une première étude a été menée au sein de notre laboratoire, dans le cadre du travail de thèse de Ilyes Boukhari [27] où un modèle pivot intégrant des modèles semi-formels a été proposé. Nous prenons ce modèle pivot proposé comme base de travail. Nous analyserons et étendrons ce modèle pivot. Nous avons ainsi constaté que la signature d'un besoin nécessite une extension afin de permettre une analyse plus fine des besoins et de leurs relations. Une modélisation dirigée par les tâches sera adoptée. L'ordonnancement entre les tâches sera modélisé par un graphe nommé graphe de précédence. Le modèle sera ensuite étendu pour permettre l'analyse multidimensionnelle des besoins.

Dans ce travail, nous proposons une démarche qui fournit un modèle de l'entrepôt sémantique de besoins fonctionnels comportant les principales étapes suivantes :

(1) l'extension du modèle pivot unifiant les modèles de besoins sources,

(2) la modélisation multidimensionnelle du modèle,

(3) l'établissement de règles de mappings entre les besoins sources et le modèle de l'entrepôt,

(4) la connexion du modèle obtenu au modèle de l'ontologie intégrante construite au chapitre précédent. Ce couplage entre le modèle de l'entrepôt et le modèle d'ontologie permet d'expliciter la sémantique des entités du modèle.

Ce modèle conceptuel pourra ensuite être traduit en un schéma logique selon le modèle cible du SGBD choisi. Il sera ensuite alimenté par les instances stockées au niveau des sources. Ces points seront traités dans les prochains chapitres.

Avec l'approche présentée dans ce chapitre, nous aurons ainsi géré deux niveaux d'hétérogénéité : l'hétérogénéité des formalismes des besoins, et l'hétérogénéité du vocabulaire en unifiant les termes et les concepts utilisés, qui passe par la construction de l'ontologie intégrante. Comme nous l'avons montré dans le chapitre 3 état de l'art (cf. section 2.1), aucune

solution compréhensible n'a été proposée pour gérer les deux niveaux d'hétérogénéité cités.

Une autre contribution importante, qui n'est pas supportée par les approches existantes, consiste en la vision multidimensionnelle fournie aux besoins dès la phase de conception. Avec la modélisation multidimensionnelle proposée, nous aurons ainsi fourni une représentation des besoins permettant différentes analyses OLAP sur les besoins entreposés. Ces analyses seront définies et illustrées dans les prochains chapitres, une fois l'alimentation de l'entrepôt réalisée.

Ce chapitre est organisé en cinq parties. La section 2 présente les formalismes sources et leur connexion avec les ontologies locales. La section 3 présente le modèle pivot des besoins à partir duquel sera défini notre modèle de l'entrepôt. La section 4 présente le modèle conceptuel multidimensionnel de l'entrepôt à travers les étapes décrites ci-dessus : révision du modèle pivot, modélisation multidimensionnelle, les mappings entre les formalismes sources et le schéma cible et le couplage du modèle obtenu avec le modèle ontologique. La section 5 résume nos contributions et conclut ce chapitre.

2 Présentation des sources

Un modèle pivot reposant sur trois modèles fréquemment utilisés pour la description et la structuration des besoins a été proposé dans le cadre de la thèse d'Ilyes Boukhari [27] effectuée au laboratoire. Ce modèle pivot permet d'uniformiser les formalismes locaux des sources. Cette version du modèle pivot a été définie pour intégrer trois langages de modélisation semi-formels : le *diagramme des cas d'utilisation* d'UML, *le formalisme orienté buts* et le formalisme orienté traitement (le *modèle conceptuel de traitement* de la méthodologie française Merise). Le modèle orienté buts proposé repose sur le paradigme GQM (Goal/Question/Metric), il est lui même un modèle pivot des modèles de trois approches orientées buts (KAOS, IStar, et Tropos) [27].

Le choix de ces langages semi-formels est justifié par leur usage dans le domaine industriel et académique. Le modèle pivot a été défini après une analyse des composantes de chaque langage utilisé et de leurs modèles génériques. Nous considérons ce modèle comme modèle de base pour définir le modèle de l'entrepôt. Nous enrichissons ensuite le modèle pivot en détaillant la signature de chaque besoin et nous fournissons une modélisation multidimensionnelle au modèle de besoins.

Nous commençons par présenter le couplage de ces trois modèles avec les ontologies locales (conceptuelles et linguistiques) pour présenter le modèle pivot proposé.

2.1 Formalismes sources et ontologies locales

La description détaillée ainsi que la formalisation des trois formalismes sources cités précédemment est présentée dans [27]. Nous utilisons dans ce qui suit les mêmes modèles proposés dans [27]. Nous nous intéressons dans cette section uniquement à la connexion de ces modèles

de besoins avec les ontologies locales des sources. Notre approche permet aux concepteurs locaux d'exprimer leurs besoins sémantiquement en utilisant une ontologie locale source. Notre approche considère les deux couches ontologiques : conceptuelle et linguistique. Le modèle de besoin source doit être connecté à ces deux modèles. Cette connexion s'effectue au niveau de leur méta-modèle. Chaque formalisme source étend ainsi le modèle ontologique.

Les formalismes sources (OntoMCT, OntoGoal, OntoUseCase) obtenus sont formalisés comme suit : $OntoFsource_i$: $< \mathcal{OCS}ource_i, \mathcal{OLS}ource_i, \mathcal{FCS}ource_i, \mathcal{FLS}ource_i, \mathcal{F}ormalisme >$ où :

- $\mathcal{OCS}ource_i$ est l'ontologie locale conceptuelle de la source i. Nous rappelons qu'une ontologie conceptuelle est formalisée comme suit : $\mathcal{OCS}ource_i : < C_i, R_i, Ref(C)_i, Ref(R)_i, F_i >$.
- $\mathcal{OLS}ource_i$ est l'ontologie locale linguistique de la source i. Nous rappelons qu'une ontologie linguistique est formalisée comme suit : $\mathcal{OLS}ource_i : < T_i, Rel(T_i), RefT(C) >$.
- $\mathcal{FCS}ource_i$ est le formalisme des besoins définis dans une source i en utilisant les concepts et propriétés ontologiques conceptuels $\mathcal{FCS}ource_i \in 2^C U 2^R$.
- $\mathcal{FLS}ource_i$ est le formalisme des besoins définis dans une source i en utilisant les termes ontologiques (ontologie linguistique) $\mathcal{FLS}ource_i \in 2^T$.
- $\mathcal{F}ormalisme = UC_{model} \oplus Goal_{model} \oplus MCT_{model}$ ($\oplus$ signifie OU exclusif.)

2.1.1 Connexion du formalisme MCT avec l'ontologie local (OntoMCT)

Le concepteur qui utilise le formalisme conceptuel des traitements exprime ses besoins au niveau ontologique en connectant les éléments de chaque traitement *(Action, Résultat, Règles d'émission)* aux concepts et propriétés de l'ontologie locale, utilisant des termes de l'ontologie linguistique. Cette connexion se fait au niveau méta-modèle entre les modèle de besoins et les deux modèles ontologiques, comme illustré en figure 5.1.

2.1.2 Connexion du formalisme des buts avec l'ontologie locale (OntoGoal)

Le concepteur qui utilise le formalisme orienté buts exprime ses besoins au niveau ontologique en connectant les éléments de chaque but *(Action, Result, Metric)* aux concepts et propriétés de l'ontologie locale, utilisant des termes de l'ontologie linguistique. Cette connexion se fait au niveau méta-modèle entre les modèles de besoins et les deux modèles ontologiques, comme illustrée en figure 5.2.

2.1.3 Connexion du formalisme des cas d'utilisation avec l'ontologie locale (OntoUseCase)

Le concepteur qui utilise le formalisme des cas d'utilisation exprime ses besoins au niveau ontologique en connectant les éléments de chaque but *(Action, Résultat, Condition, Point d'extension)* aux concepts et propriétés de l'ontologie locale, utilisant des termes de l'ontologie

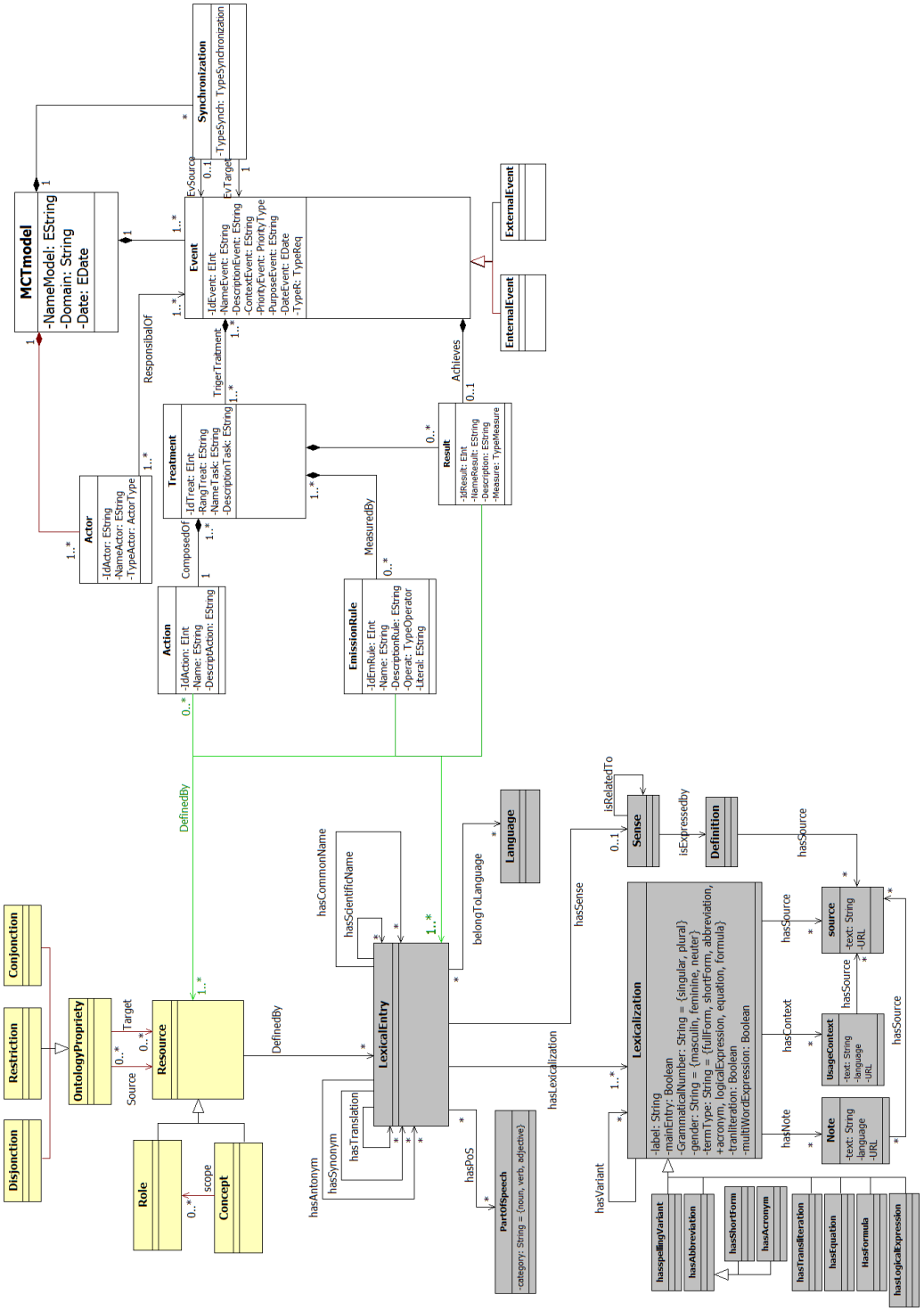

FIGURE 5.1 – Le framework OntoMct : connexion du modèle MCT avec les modèles ontologiques

linguistique. Cette connexion se fait au niveau méta-modèle entre les modèles de besoins et les

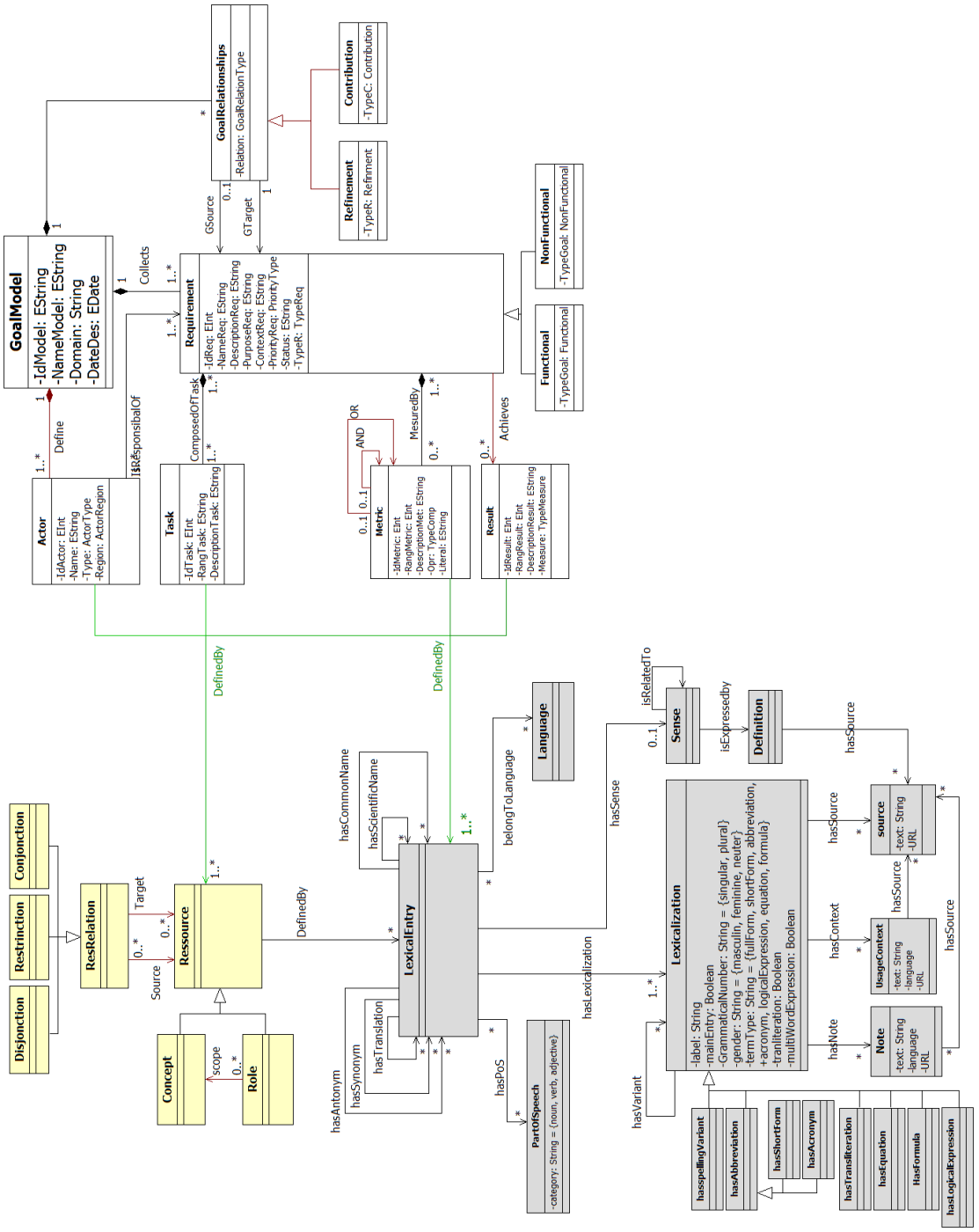

FIGURE 5.2 – Le framework OntoGoal : connexion entre le modèle orienté buts et les modèles ontologiques

deux modèles ontologiques, comme illustré en figure 5.3.

Nous proposons dans ce qui suit le modèle pivot unifiant les trois formalismes sources.

3 Le modèle pivot proposé

Le modèle pivot proposé dans [27] a été défini afin de permettre l'unification des trois formalismes sources présentés. Ce modèle est illustré dans la figure 5.4.

Ce modèle pivot a été formalisé comme suit [27] :

Formalisation 5

$Pivot_{model}$: $< \mathcal{A}ctor, \mathcal{R}equirement, \mathcal{R}elationships >$, où :

- $\mathcal{A}ctor = \{actor_1, actor_2..., actor_n\}$ est un ensemble d'acteurs (p. ex. concepteur).
- $\mathcal{R}equirement = \{req_1, req_2..., req_n\}$ est un ensemble de besoins exprimés par un acteur. Nous définissons un besoin (requirement) comme suit : $\mathcal{R}equirement_i : < \mathcal{A}, \mathcal{R}, C >$ un ensemble de besoins où :
  - $\mathcal{A} = \{a_1, a_2..., a_n\}$ est l'ensemble d'actions qu'un système effectue pour fournir un résultat. $\forall a_i \in \mathcal{A}$, $\exists p_j \subseteq \mathcal{P}$, telle que $f(a_i) = p_j$, où $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ est l'ensemble des propriétés satisfaites par un système.
  - $\mathcal{R} = \{r_1, r_2..., r_n\}$ est l'ensemble des résultats réalisés par le système.
  - $C = \{c_1, c_2..., c_n\}$ est l'ensemble des critères selon lesquels un résultat est quantifié.
- $\mathcal{R}elationships = \{relation_1, relation_2..., relation_n\}$ est l'ensemble des relations entre les besoins.

Dans ce modèle pivot, chaque besoin est défini par les entités : *Actions*, *Resultats Criteres*. Après l'analyse de ce modèle de besoins et son utilisation pour raisonner sur les besoins en utilisant une série de tests, nous avons conclu que cette formalisation ne permet pas la définition de tous les détails des besoins qui sont indispensables pour avoir une représentation unique de chaque besoin. Nous présentons dans ce qui suit notre modèle proposé.

4 Le modèle conceptuel de l'ESBF

Le modèle conceptuel de l'entrepôt de besoins fonctionnels (ESBF) passe par la révision du modèle pivot puis par la modélisation multidimensionnelle du modèle obtenu.

4.1 Révision du modèle pivot

Le modèle pivot permet de factoriser l'ensemble des formalismes utilisés par les sources de besoins. La révision du modèle pivot permet d'obtenir un modèle proche des vocabulaires

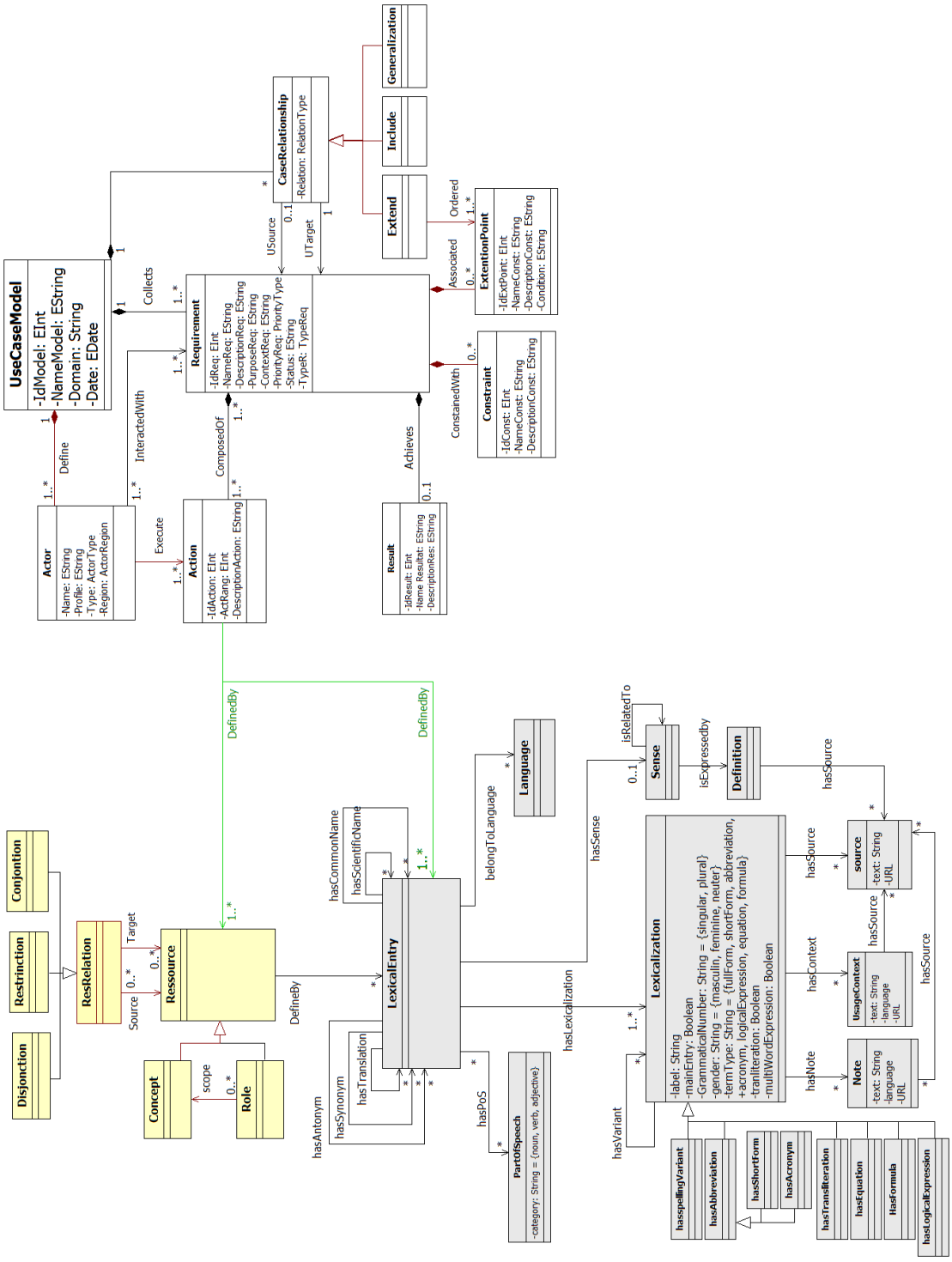

FIGURE 5.3 – Le framework OntoUseCase : connexion du modèle des cas d'utilisation aux modèles ontologiques

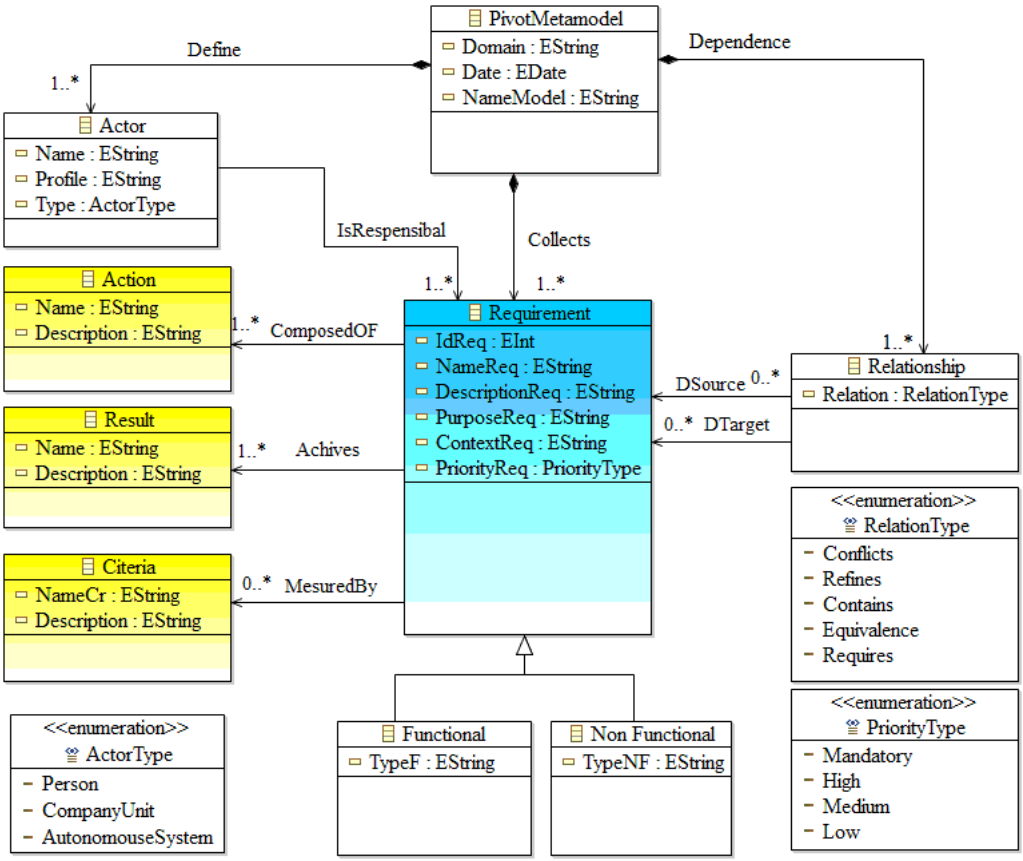

FIGURE 5.4 – Le méta-modèle pivot [28].

contrôlés utilisés par les entreprises lors de la définition des besoins, et assez proche du langage naturel. Nous commençons par détailler les limites du modèle pivot initial.

Comme expliqué précédemment, le modèle pivot présenté ci-dessus présentent certaines limites. Pour illustrer notre propos, prenons l'exemple de deux besoins, B_{17} et $B_{17'}$, définis à partir du document des besoins CMS:
B_{17} :"*The system shall allow students to create teams*".
$B_{17'}$:"*The system shall allow lecturers to create teams*".

L'analyse des deux besoins, définis dans l'exemple avec la première version du modèle pivot, montre que les deux besoins B_{17} et $B_{17'}$ ont les mêmes actions $\{Action_{17} = (create\ teams), Action_{17'} = (create\ teams)\}$ mais n'ont aucun résultat et aucun critère. L'analyse de ces deux besoins montre que les deux besoins sont égaux (B_{17} *Equal* $B_{17'}$).

Ceci est un résultat erroné car la réalisation des deux besoins a une influence différente sur le système conçu : les deux besoins B_{17} et $B_{17'}$ ont deux différents acteurs internes qui interagissent avec le système : *students*, *lecturers*. Ces deux besoins sont donc différents. Nous avons ainsi constaté que la signature d'un besoin nécessite une extension.

Afin de remédier à cette limite, nous avons proposé un nouveau modèle de besoins que nous appelons le *modèle pivot enrichi* (MPE) et qui est illustré en figure 5.5. Ce modèle permet de détailler toutes les composantes d'un besoin. Pour ce faire, nous nous sommes inspirés de la formalisation dirigée par les tâches. Celle-ci est utilisée dans les applications centrées utilisateur qui considèrent les utilisateurs et leurs besoins tout au long du processus de développement d'applications, comme par exemple les applications de crowdsourcing, le design collaboratif et les SMA. Chaque besoin (*Requirement*) précise une tâche détaillée en un triplet (sujet, verbe, objet). Un besoin est ainsi modélisé d'une façon plus proche de la représentation d'une phrase en langage naturel.

Ce modèle nous permet de représenter tous les éléments détaillés des besoins, ce qui facilite ainsi leur intégration et permet un raisonnement plus précis sur les besoins.

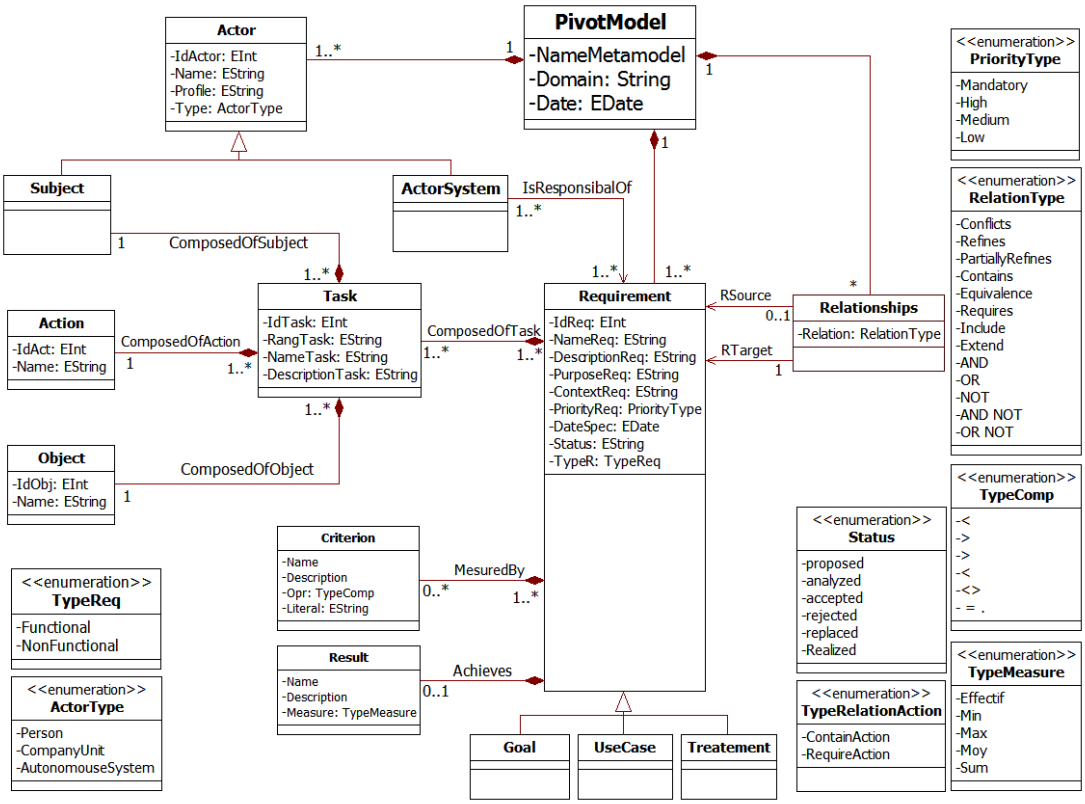

FIGURE 5.5 – Modèle pivot enrichi (MPE)

Nous formalisons le modèle pivot enrichi MPE comme suit :

Formalisation 6

$EPivot_{Modele}$: < Actor, Requirement, Relationships >, dans lequel :

- *$Actor = \{act_1, act_2, .., act_N\}$ est un ensemble d'acteurs qui interagissent avec le système. Nous distinguons deux types d'acteurs : l'acteur qui définit le besoin (acteur système) ou l'acteur précisé dans le besoin (sujet). Un acteur peut correspondre à une*

personne, une entité administrative ou de gestion dans l'organisation ou bien un système autonome.

- *Requirement* = $\{B_1, B_2, ..., B_n\}$ *correspond à l'ensemble des besoins exprimés par un acteur. Nous définissons un besoin* B_i *par le quadruplet :* $< TASK, R, C, T >$ *est tel que :*
  - $TASK = \{task_1, task_2, ..., task_m\}$, *un ensemble de tâches qu'exprime un besoin. Chaque* $task_i =< Subject, Action, Object >$, *tel que :*
    - $Subject$: *est un acteur qui utilise le besoin afin d'obtenir un résultat ;*
    - $Action$: *est une action représentée par un verbe que le système exécute pour avoir un résultat observable ;*
    - $object$: *est le concept (physique ou mental) concerné par l'action du besoin (verbe) ;*
  - $\mathcal{R} = \{r_1, r_2, ..., r_p\}$*: ce sont les résultats réalisés par le système ;*
  - $C = \{c_1, c_2, ..., c_k\}$*: un ensemble de critères qui quantifient le résultat ;*
  - T*: le type du formalisme de représentation du besoin, dans notre cas* $T \in \{ Goal, Usecase, Treatment \}$.
- $Relationships = \{relation_1, ..., relation_n\}$ *est l'ensemble de relations entre les besoins. Chaque* $relation_i = \{ Equals \mid Contain \mid Refine \mid Require \mid Conflicts \mid partiallyRefine \mid Include \mid Extend \mid Not \mid And \mid Or \}$. *Ces relations peuvent être explicitement exprimées ou obtenues à partir des relations entre les besoins.*

Pour illustrer l'intérêt du modèle enrichi, nous projetons l'exemple des deux besoins explicités précédemment sur ce modèle :
B_{17}: "*The system shall allow students to create teams*".
$B_{17'}$:"*The system shall allow lecturers to create teams*".

L'instanciation de ces deux besoins sur notre MPE montre que les deux besoins B_{17} et $B_{17'}$ ont les mêmes actions $\{Action_{17} = (create), (Action_{17'} = create\}$ et les mêmes objets $\{Object_{17} = (teams), (Object_{17'} = teams)\}$, mais ils ont des sujets différents : $\{Subject_{17} = (students), (Subject_{17'} = lecturers)\}$. Ces deux besoins ont donc des tâches différentes qui sont TASK= < *students, create, teams* > et TASK= < *lecturers, create, teams*>. Ils n'ont aucun résultat et aucun critère.

Le résultat de l'analyse de ces deux besoins avec le nouveau modèle montre que les deux besoins sont différents car ils ont une influence différente sur le système conçu, et plus précisément sur les acteurs internes : *students* et *lecturers*.

Les relations représentées dans le modèle sont de plusieurs types. Nous avons recensé les relations définies dans la littérature exprimant les liens qui peuvent exister entre deux ou plusieurs besoins. Six types de relations sont identifiés (*Equals, Conflicts, Contains, Refines, Partially-Refines, Requires*). Nous détaillerons ces relations durant la phase de raisonnement, où leur utilisation sera utile pour inférer de nouveaux faits sur les besoins et à vérifier leur consistance.

Nous représentons deux relations particulières dans un graphe que nous appelons graphe de précédence. Ces relations décrivent des relations spécifiques entre les tâches qui composent les besoins, à savoir : la contenance et l'ordonnancement entre les tâches. Ces relations sont définies dans un graphe de précédence, ou les noeuds représentent les verbes et les arcs représentent les relations entre ces noeuds (*Contain, Require*). Notons que le verbes utilisés sont disponibles dans l'ontologie linguistique/conceptuelle partagée (cf. chapitre 4, section cf.4). Pour établir son graphe de précédence, le concepteur se réfère à l'organisation interne de l'entreprise. Dans l'exemple suivant, nous illustrons la notion de graphe de précédence :

Exemple 5

> *B_1: The system shall allow students to create an account.*
> *B_2: The system shall allow students to connect to account.*
> *B_3: The system shall allow students to register in courses.*

La figure 5.6 présente le graphe de précédence des tâches que définissent les besoins B_1, B_2 et B_3. Ce graphe montre qu'un étudiant ne peut pas se connecter à son compte avant qu'il ne crée un compte et un étudiant ne peut pas s'enregistrer dans un cours avant qu'il ne crée un compte et qu'il ne se connecte à ce compte.

FIGURE 5.6 – Exemple de graphe de précedence d'une source.

Pour représenter le graphe de précédence, le modèle pivot initial est ainsi étendu par la classe *PrecedenceGraphAction* qui définit les relations (*Contains* et *Requires*) entre les actions des besoins.

4.2 Modélisation multidimensionnelle du schéma de l'entrepôt

Nous proposons un schéma d'entrepôt de besoins fonctionnels afin de permettre par la suite le stockage et l'analyse OLAP des besoins. Ce schéma est construit par extension du modèle pivot de besoins que nous avons proposé dans l'étape précédente, à l'aide de nouvelles entités. Nous détaillons cette étape dans la suite de cette section.

4.2.1 Étendre le MPE avec les dimensions temps et localisation

Nous avons commencé par étendre le modèle enrichi proposé, en utilisant deux nouvelles entités temps et localisation (*Time, Location*) comme illustré en figure 5.7 :

– la dimension *Time* représente le moment où le besoin a été exprimé au niveau du partenaire source. Nous avons représenté cette dimension par une classe contenant plusieurs propriétés :

- *IdTime* : représente l'identifiant du temps ;
- *Year*, *Month*, *Day* et *Hour* : représentent respectivement l'année, le mois, le jour et l'heure où le besoin est défini au niveau de la source.

– la dimension *Location* représente la localisation du partenaire source qui a exprimé ce besoin. Nous avons représenté cette dimension par une classe nommée *Location* et plusieurs propriétés :
  - *NameLoc* : représente le nom de la localisation qui peut être le nom d'une entreprise, université, etc. ;
  - *Region* : représente la situation géographique définie par la région où le besoin est défini, par exemple les régions des États-Unis (*Alaska, Californie, etc.*) ;
  - *Department* : représente la situation géographique définie par le département où le besoin est défini, par exemple les départements en France (*Charente-Maritime, Loire, etc.*) ;
  - *State* : représente la situation géographique définie par un état où le besoin est défini, par exemple (*France, Allemagne, etc.*).

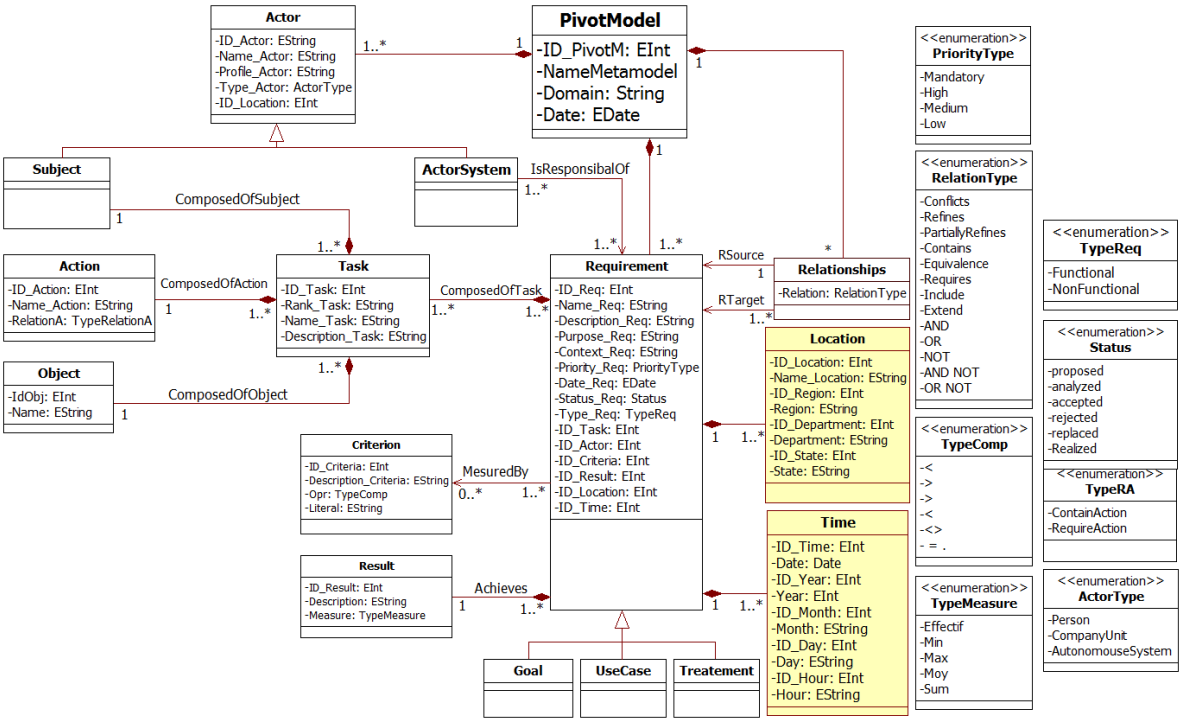

FIGURE 5.7 – Enrichissement du modèle pivot par les dimensions *Time* et *Location*

La formalisation du modèle obtenu étend la première formalisation (cf. section 4.1) comme suit :

Formalisation 7

$NPivot_{Modele}$: < Actor, Requirement, Relationships >, dans lequel Requirement est défini comme suit :

Requirement = { B_1,B_2,...,B_n} *est l'ensemble des besoins exprimés par un acteur. Nous définissons un besoin* B_i *par le quadruplet:* < *TASK, R, C, T , Location, Time* >*, tel que :*

- *TASK* = {$task_1$, $task_2$, ..., $task_m$} *: est l'ensemble de tâches qu'exprime un besoin. Chaque tâche* $task_i$ *est composée du triplet* < $Subject$, $Action$, $Object$ >.
- $R = \{r_1, r_2, ..., r_p\}$ *: ce sont les résultats réalisés par le système ;*
- $C = \{c_1, c_2, ..., c_k\}$ *: un ensemble de critères qui quantifient le résultat ;*
- *T: le type du formalisme de représentation du besoin, dans notre cas* $T \in$ { $Goal$, $Usecase$, $Treatment$ }.
- *Location: représente la localisation géographique du partenaire source qui a exprimé ce besoin. Elle est définie par un ensemble de propriétés (IdLoc, Department, etc.) qui permettent d'identifier la localisation.*
- *Time : représente la période où le partenaire source a exprimé un besoin. Elle est définie par un ensemble de propriétés (IdTime, Year, etc.) qui permettent de savoir le moment où le besoin est défini.*

Une fois étendu par les nouvelles dimensions, le nouveau modèle résultat nous permet par la suite de construire un schéma multidimensionnel des besoins fonctionnels.

4.2.2 Définition du schéma de l'entrepôt de besoins

La construction du schéma multidimensionnel représente une phase importante dans la phase conceptuelle de tout système d'entreposage. Cette phase doit identifier les concepts multidimensionnels de *faits, mesures, dimensions, attributs de dimension* et *hiérarchies de dimensions* (cf. section 2, chapitre 3.2).

Nous rappelons qu'un *fait* représente un sujet ou un thème d'analyse et une dimension représente un contexte d'analyse d'un fait. Une dimension peut être composée d'une hiérarchie de niveaux représentant différentes granularités pour analyser les données (*année* $\longrightarrow$ *semestre* $\longrightarrow$ *mois* $\longrightarrow$ *jour*). Chaque dimension est décrite par un ensemble de propriétés (attributs de dimensions). D'autre part, un *fait* est caractérisé par des *Mesures* ou attributs du *fait* décrivant des informations sur le thème analysé. Un schéma composé d'un *fait* et de ses *dimensions* donne un schéma en étoile ou flocon de neige (si les hiérarchies de dimensions sont décomposées).

Sur la base de ces définitions et afin d'identifier les dimensions et les faits de notre schéma multidimensionnel, nous nous sommes basés d'abord sur les composantes de notre modèle pivot enrichi (*Requirement, Actor, Task, Criteria, et Result*) proposé dans la section 4.1. Notre objectif est de permettre une analyse multidimensionnelle des besoins issus de divers partenaires selon plusieurs dimensions. Pour ce faire, nous avons conçu le schéma multidimensionnel de façon à mettre l'entité "*Requirement*" au centre de cette analyse, représentant ainsi une entité fait. Plusieurs mesures peuvent être définies pour ce fait comme le nombre maximal de besoins, l'évolution du nombre de besoins (dans le temps), le coût des besoins (par partenaire). Les attributs de ce *fait* sont les attributs identifiés pour l'entité "*Requirement*" tels que : la description

du besoins, son contexte, sa priorité, etc.

Pour identifier les dimensions, plusieurs travaux dans la littérature proposent d'analyser les besoins sur la base des cardinalités et associations liant le *fait* aux autres entités [92, 67]. Ces travaux proposent d'associer à chaque *fait* les *dimensions* qui lui sont liées par des relations binaires *un à plusieurs* (cardinalité (1..1) du côté de la *dimension* et (1..*) ou (0..*) du côté du *fait*). Dans certains cas, les relations plusieurs à plusieurs sont également tolérées entre un *fait* et une *dimension*. Ces travaux proposent aussi d'associer les niveaux de dimensions par une relation *un à plusieurs* entre le niveau de *dimension* supérieur (*ds*) et le niveau inférieur (*dl*) correspondant (relation (1..1) du côté de *ds* et une autre (1..*) du côté de *dl*).

Dans notre cas, notre principal intérêt concerne l'analyse des besoins "*Requirement*" selon plusieurs contextes, par exemple : le nombre de besoins par région ou le nombre de besoins par partenaire. En nous basant également sur les cardinalités entre l'entité "*Requirement*" et les autres entités du modèle pivot enrichi, nous avons pu identifier les dimensions suivantes : *Actor, Task, Criteria, Result, Localization et Time*.

Un schéma en étoile correspondant à notre modèle multidimensionnel est ainsi défini et illustré dans la figure 5.8. Il s'agit dans cette figure d'une réorganisation du modèle pivot enrichi MPE qui permet une meilleure visualisation du modèle multidimensionnel. Dans la suite de ce chapitre, nous utilisons le modèle complet MPE.

Ce schéma peut être étendu en un schéma en flocon de neige si nous normalisons les dimensions Location (location-Region-Department-State), Time (Time-Day-Month-Year) et Actor (Actor-Type).

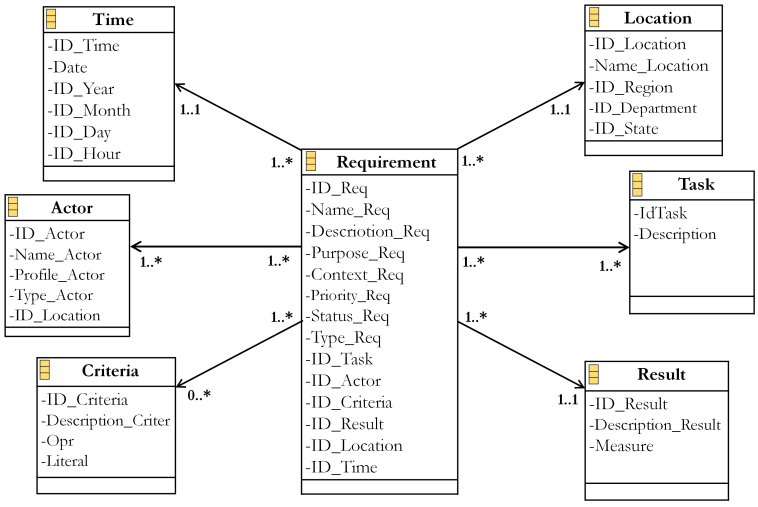

FIGURE 5.8 – Le schéma en étoile de l'entrepôt de besoins

Dans la section suivante, nous décrivons les mappings entre les modèles de besoins sources et le modèle de l'entrepôt.

5 Transformation des besoins sources vers le modèle de l'entrepôt

Dans cette partie, nous expliquons le processus de transformation des besoins définis au niveau des modèles sources vers notre modèle de l'entrepôt. Tout comme des mappings ont été établis entre les ontologie locales et l'ontologie intégrante, des mappings doivent être établis entre les modèles de besoins sources et le modèle cible de l'entrepôt. Par exemple, si le modèle source est un modèle de cas d'utilisation contenant une entité Action (référençant le concept *Gérer* dans l'ontologie locale), un mapping permet de lier cette action avec l'entité *Action* (référençant le concept *Manage* dans l'ontologie intégrante) du modèle de l'ESBF proposé.

Ces mappings reposent sur la transformation des modèles de besoins sources vers le modèle de l'entrepôt. Celle-ci se fait en établissant des règles de mises en correspondance. Puisqu'on manipule des modèles, nous avons opté pour une approche de l'*Ingénierie Dirigée par les Modèles* (IDM), afin d'établir ces correspondances et assurer la transformation automatique des instances des besoins sources vers le modèle pivot intégrateur (section 5).

Pour commencer, nous rappelons que la transformation entre modèles fait partie de l'IDM qui est une démarche utilisée dans le développement de logiciels. L'IDM est définie pour permettre l'utilisation des techniques et outils informatiques pour décrire des modèles, des concepts et des langages (UML, MOF, QVT, etc.). Dans l'IDM, la transformation des modèles a une place importante pour simplifier les systèmes complexes. Plusieurs outils commerciaux et open source sont disponibles pour assurer la transformation des modèles suivant des langages de transformation. Ces langages sont classés en plusieurs catégories : transformation de modèles vers d'autres modèles (M2M) (comme les langages ATL, Procedural QVT, Declarative QVT, XTend), transformation de modèles vers du texte (M2T) (MOFMTL *(implémentation : Acceleo), Jet, XPand*). Concernant les outils de création ou de manipulation de modèles, nous trouvons plusieurs exemples comme *Papyrus* ou *Sirius*. Ces outils disposent de systèmes de transformation qui permettent de transformer de $\{1, ..., n\}$ modèles sources vers $\{1, ..., m\}$ modèles cibles. Un système de transformation est basé sur des règles de transformation qui décrivent la manière de transformer un ou plusieurs concepts (ou propriétés) sources vers un ou plusieurs concepts (ou propriétés) cibles et un moteur de transformation.

Dans notre approche, nous avons utilisé l'IDM pour assurer la transformation des besoins de chaque modèle de besoins source vers le modèle de l'entrepôt (figure 5.9).

Cette transformation se fait par l'intermédiaire de règles de transformation définissant des *mapping* de chacune des composantes des besoins sources vers des composantes correspondantes dans le modèle cible. Le tableau suivant (figure 5.10) récapitule les mappings entre les entités des trois modèles de besoins sources et le modèle de l'entrepôt.

Nous avons utilisé le langage ATL pour formaliser ces règles de transformation (figure 5.11).

La Figure 5.12 montre un exemple d'une règle ATL qui définit la transformation des besoins

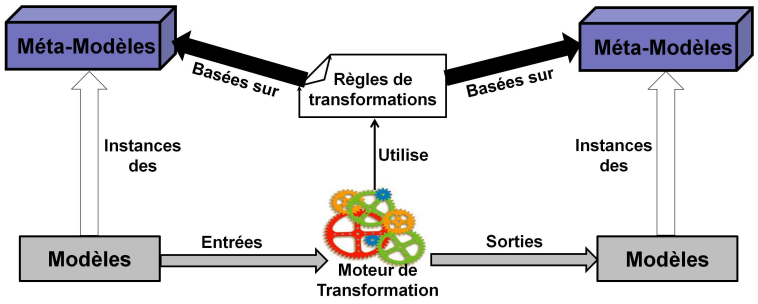

FIGURE 5.9 – Structure d'un système de transformation de modèles

REQUIREMENTS Sources Formats	CONCEPTS		PIVOT MODEL Format						
			ACTOR	REQUIREMENT					RELATIONSHIPS
				TASK			CRITERION	RESULT	
				SUBJET	ACTION	OBJECT			
GOAL –ORIENTED (SD1)	ACTOR		x	x					
	GOAL	TASK			x				
						x			
		METRIC					x		
		RESULT						x	
	AND /OR								x
	CONTRIBUTION RELATIONSHIPS								x
USE CASE (UML) (SD2)	ACTOR		x	x					
	USE CASE	ACTION			x				
						x			
		RESULT						x	
		EXTENSION POINT							x
		CONDITION					x		x
	CaseRelationships	GENERALIZATION							x
		INCLUDE							x
		EXTEND							x
MCT (MERISE) (SD3)	ACTOR		x	x					
	EVENT								
	TREATMENT	ACTION			x				
						x			
		RESULT						x	
		EMISSION RULES					x		
	SYNCHRONIZATION (AND/OR)								x

FIGURE 5.10 – Mapping des formalismes sources vers le modèle de l'entrepôt

du formalisme orienté buts (Goal) vers le modèle de l'entrepôt proposé. De la même façon nous avons défini toutes les règles afin d'assurer la transformation des instances de besoins définis dans les formalisme sources (MCT, Goal, UseCase) vers le format cible. Ces règles sont fournies en annexe (cf. Annexe 1.3.6).

Dans cette première étape, nous avons procédé à l'unification des formalismes sources vers le modèle cible de l'entrepôt. Nous procédons dans ce qui suit au couplage entre le modèle de l'entrepôt et le modèle ontogique.

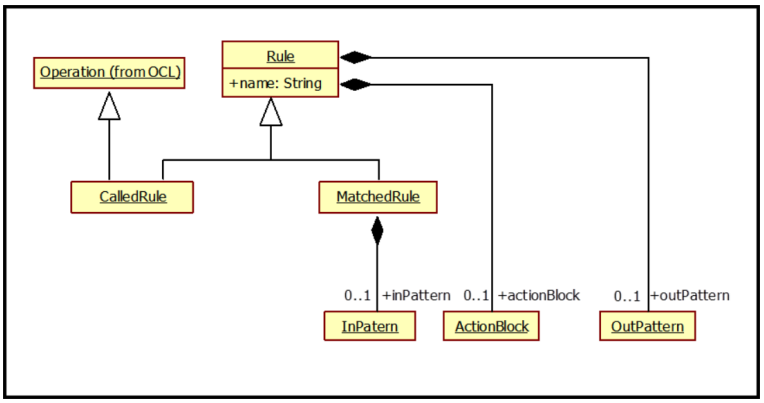

FIGURE 5.11 – Syntaxe d'une règle de transformation ATL (extraite du métamodèle de l'ATL)[42]

6 Connexion des modèles de besoin et ontologiques

Comme expliqué précédemment, l'ensemble des éléments du modèle de besoins sont définis sémantiquement en utilisant les deux couches de l'ontologie globale (conceptuelle et linguistique).

La connexion entre le modèle de l'entrepôt, l'ontologie conceptuelle et l'ontologie linguistique doit ainsi se faire afin de garder une trace sémantique de tout élément utilisé dans la définition des besoins.

La figure 5.13 illustre cette connexion entre les modèles utilisés. Nous connectons d'abord le modèle de l'entrepôt avec les éléments de l'ontologie conceptuelle. Les éléments du modèle de l'entrepôt qui sont définis par des éléments ontologiques sont donc : les éléments de *Task* (*Subject*, *Action* et *Object*), *Criteria* et *Result*. Le modèle ontologique repose sur la formalisation que nous avons fournie, dans laquelle un élément ontologique peut être un concept ou un rôle (attribut ou relation).

La couche linguistique de l'ontologie globale est également présentée. Le modèle se base sur le meta-modèle de Wordnet fourni dans [153]. La connexion se fait entre les couches conceptuelle et linguistique par le lien *hasLExicalEntry* qui relie l'entité *OntologyElement* de l'ontologie conceptuelle à l'entité *LexicalEntry* de l'ontologie linguistique. De cette façon, les éléments des besoins utilisent des ressources ontologie de l'ontologie conceptuelle qui sont exprimées par un ou plusieurs termes de l'ontologie linguistique.

Le modèle est étendu par la modélisation des règles de raisonnement qui nous permettent d'identifier de nouvelles relations entre les besoins. Cette partie du modèle est basée sur le métamodèle du langage SWRL [60], que nous utilisons pour formaliser nos règles de raisonnement. Les règles sont définies sur les ressources ontologiques (de l'ontologie conceptuelle) utilisées pour décrire les entités des besoins, le résultat de chaque règle étant une nouvelle relation entre les

60. https://www.w3.org/Submission/SWRL/

```
1.  module OntoGoalInstance2OntoEPivotInstance;
2.  create OUT : EPivotModel from IN : MMGoal;
3.  rule MMGoalGoal2EPivotModeRequirement
4.  {
5.      from
6.          s : MMGoal!Goal
7.      to
8.          t : EPivotModel!Requirement (
9.                  IdReq <— IdReq + 1,
10.                 IdReq <— s.IdGoal,
11.                 NameReq <— s.NameGoal,
12.                 DescriptionReq <— s.DescriptionGoal,
13.                 Context <— s.Context,
14.                 PurposeReq <— s.PurposeGoal,
15.                 PriorityReq <— s.Priority,
16.                 DateSpec <— s.DateCreat )
17. }
18. rule MMGoalActor2EPivotModelSubject
19. {
20.     from
21.         s : MMGoal!Actor
22.     to
23.         t : EPivotModel!Subject (
24.                 IdActor <— IdActor + 1,
25.                 IdActor <— s.IdActor,
26.                 Name <— s.NameActor )
27. }
28. rule MMGoalTask2EPivotModelAction
29. {
30.     from
31.         s : MMGoal!Task
32.     to
33.         t : EPivotModel!Action (
34.                 IdTask <— IdTask +1,
35.                 IdAction <— a.IdTask,
36.                 DescriptionAction <— TockenL(a.DescriptionTask )
37. }
38. rule MMGoalTask2EPivotModelObject
39. {
40.     from
41.         s : MMGoal!Task
42.     to
43.         t : EPivotModel!Object (
44.                 IdTask <— IdTask +1,
45.                 IdObject <— a.IdTask,
46.                 DescriptionObject <— TockenR(a.DescriptionTask )
47. }
48. rule MMGoalResult2EPivotModelResult
49. {
50.     from
51.         s : MMGoal!Result
52.     to
53.         t : EPivotModel! Result (
54.                 IdResult <— IdResult + 1,
55.                 IdResult <— a.IdResult,
56.                 Name <— a.RangResult,
57.                 Name <— a.Description )
58. }
59. rule MMGoalRelationships2EPivotModelRelationships
60. {
61.     from
62.         s : MMGoal! GoalRelationships
63.     to
64.         t : EPivotModel!Relationships (
65.                 IdRelationships <— IdRelationships + 1,
66.                 IdRelationships <— a.IdGoalRelationships,
67.                 Relationships <— a.GoalRelationships)
68. }
69. rule EPivotActor (Name: String)
70. {
71.     from
72.         s : MMGoal
73.     to
74.         t : EPivotModel!Actor (
75.                 IdActor <— IdActor +1,
76.                 Name <— 'EnglandPartner' )
77. }
```

FIGURE 5.12 – Exemple d'une règle de transformation entre les formalismes Goal et EPivot

besoins, i.e représenté par une instance de l'entité *Relationship* ou *PrecedenceGraphAction*.

7 Conclusion

Dans ce chapitre, nous avons proposé une approche fournissant le modèle conceptuel de l'entrepôt de besoins à définir. Cette approche comprend plusieurs étapes. Elle permet l'unification des formalismes de modélisation des besoins par raffinement d'un modèle pivot proposé au laboratoire, unifiant trois semi-formalismes : orienté buts, cas d'utilisation UML, et le modèle conceptuel de traitement de la méthode Merise. Un schéma multidimensionnel de l'entrepôt de besoins est ensuite construit en identifiant les différents concepts de faits, de dimensions et de leurs attributs.

Des règles de mises en correspondance permettent d'unifier les modèle de besoins sources selon le modèle cible de l'entrepôt. Ce dernier est ensuite connecté au modèle de l'ontologie intégrante afin de permettre une définition sémantique des besoins entreposés.

Nous proposons dans le prochain chapitre d'alimenter ce schéma cible par les instances des besoins sources. Ceci passe par la définition d'un processus ETL adapté aux besoins fonctionnels.

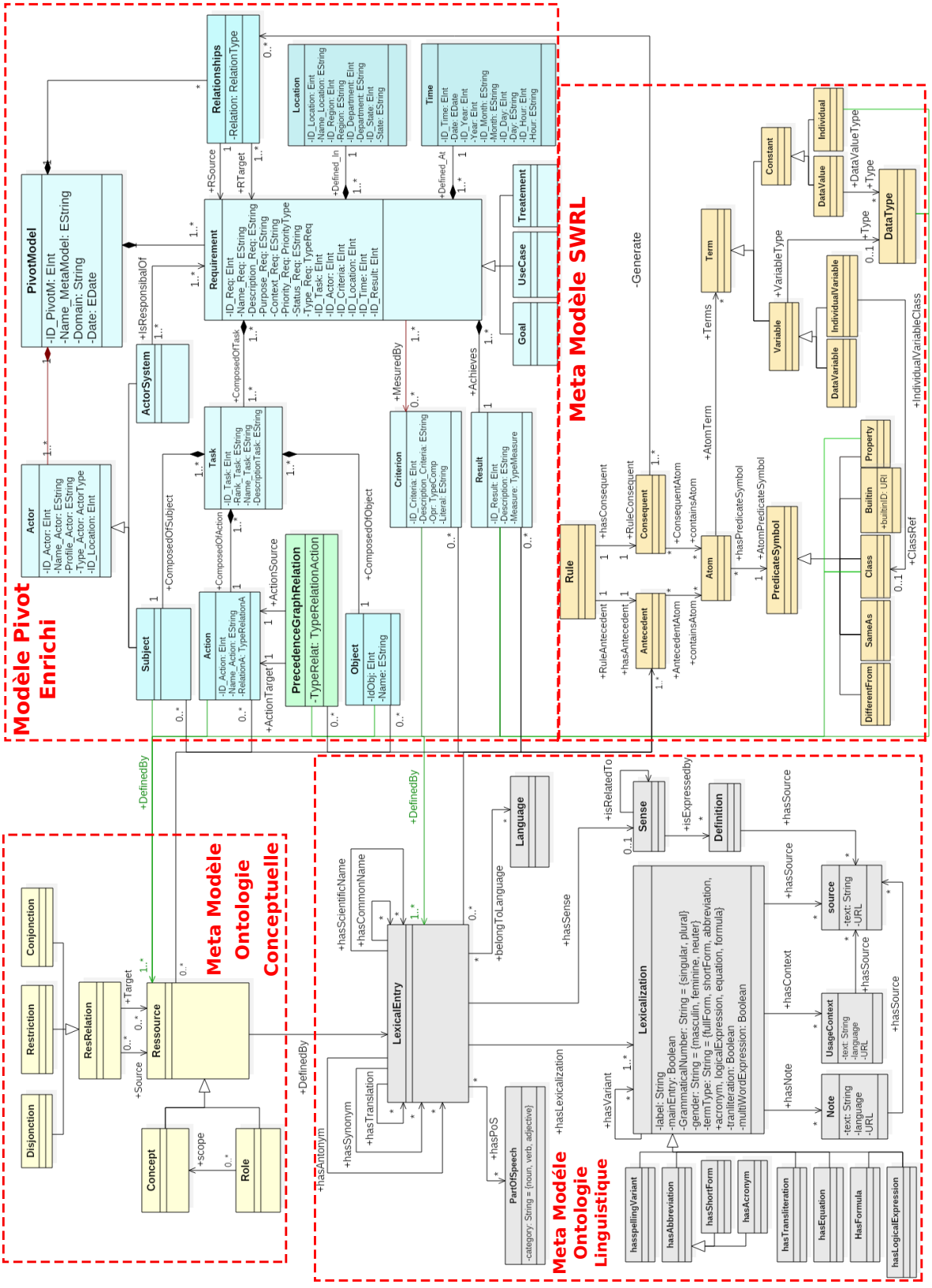

FIGURE 5.13 – Connexion entre les composantes de notre framework au niveau méta-modèle

Chapitre 6

Un Processus ETL pour les Besoins Fonctionnels

Sommaire

1 Introduction 132

2 Composantes de la phase ETL 133

2.1 Les mappings M 134

2.2 Les opérateurs ETL 134

2.2.1 Les opérateurs de base 135

2.2.2 Les opérateurs dédiés aux besoins 136

2.2.3 Les opérateurs d'inférence 141

2.2.4 Les opérateurs de gestion 143

2.3 Le processus ETL de besoins 145

2.4 L'algorithme ETL 146

3 Raisonnement sur les besoins 147

3.1 Relations sémantiques entre les besoins : définitions 149

3.2 Mécanisme de raisonnement 150

3.2.1 Les règles d'identification 150

3.2.2 Les règles d'inférence 153

3.2.3 Les règles de vérification de la cohérence 153

4 Implémentation et Expérimentation 154

4.1 Performances du raisonnement 155

4.2 Complexité de l'algorithme ETL de besoins 160

5 Conclusion 161

Résumé. Dans ce chapitre, nous proposons une démarche pour alimenter notre entrepôt de besoins fonctionnels. Le schéma de l'entrepôt de besoins (ESBF) défini dans le chapitre précédent permet l'unification des sources utilisant : (1) des formalismes différents et (2) des concepts et des termes différents. La troisième étape de la définition de l'entrepôt de besoins consiste à alimenter ce schéma obtenu de l'entrepôt par les besoins des sources. Ceci passe par la définition d'une phase d'Extraction-Transformation-Chargement (ETL) des instances des sources au niveau du schéma cible de l'entrepôt. Cette phase repose sur la définition des schémas des sources, du schéma cible et des mappings entre ces schémas. Notons que les différents mappings entre les formalismes et entre les ontologies ont déjà été définis. Un ensemble d'opérateurs ETL a été défini (adapté des opérateurs proposés dans la littérature) afin d'assurer le processus ETL. Nous avons ensuite enrichi ces opérateurs ETL conventionnels par des opérateurs de raisonnement permettant d'identifier des relations entre les besoins ainsi que les besoins inconsistants afin de les éliminer. Ce mécanisme de raisonnement sera évalué théoriquement. La complexité de l'algorithme ETL proposé sera également évaluée.

1 Introduction

La phase ETL est considérée comme l'une des phases les plus critiques et les plus consommatrices en temps dans un projet d'entreposage. Cette phase consiste à peupler le schéma de l'entrepôt par les instances des sources. Cette phase est basée sur la définition de trois principaux éléments : les schémas des sources, le schéma cible et les mappings établis entre les schémas sources et le schéma cible. Les mappings permettent de définir les correspondances entre les schémas sources et cible (comme l'agrégation des instances sources ou l'union de plusieurs concepts sources en un concept cible). Un processus ETL est ensuite défini afin de permettre l'extraction des instances à partir des schémas sources, leur transformation selon les mappings définis et leur chargement dans le schéma cible de l'entrepôt. Ce processus ETL est basé sur des opérateurs ETL assurant les transformations nécessaires.

Le principal objectif de ce chapitre est d'adapter le processus ETL conventionnel, proposé pour les sources de données, aux *besoins fonctionnels*. L'adaptation concerne d'abord les mappings.

Nous rappelons que les différents mappings entre les formalismes et entre les ontologies ont déjà été définis. Les mappings entre les formalismes sources et cible sont établis via des règles de correspondance ATL (chapitre 5, section 5). Les mappings ontologiques concernent les mappings entre les ontologies locales et l'ontologie intégrante sur les deux couches linguistique et conceptuelle (chapitre 4, section 4).

L'adaptation de la phase ETL concerne aussi chaque étape du processus ETL ainsi que les opérateurs ETL conventionnels. Nous avons d'abord adapté l'ensemble des opérateurs ETL conventionnels proposés dans la littérature. Nous avons ensuite enrichi ces opérateurs ETL par des opérateurs de raisonnement sur les besoins. Ce raisonnement permet d'identifier des relations entre les besoins ainsi que les besoins inconsistants afin de les éliminer. La définition de ces mécanismes de raisonnement permettent d'améliorer la qualité de l'ESBF. Cette dernière dépend en effet de la qualité de l'ETL (selon le principe du *Garbage in, Garbage out*). Un processus ETL utilisant ces opérateurs adaptés est ensuite défini pour alimenter le schéma de l'ESBF.

Ce chapitre est organisé en cinq sections. La section 2 présente la phase ETL adaptée aux besoins fonctionnels. La section 3 présente le mécanisme raisonnement qui permet d'inférer des faits sur les besoins et d'identifier les besoins inconsistants. La section 4 présente l'expérimentation de notre approche qui évalue les performances du mécanisme de raisonnement ainsi que la complexité de l'algorithme ETL. La section 5 conclut ce chapitre.

2 Composantes de la phase ETL

La phase ETL permet le peuplement du schéma de l'entrepôt par les instances des sources. Dans la définition du processus ETL, l'entrepôt est vu comme un système d'intégration. Les systèmes d'intégration sont formalisés par un triplet <G, S, M> [100] comme suit :

- le schéma G : représente le schéma global cible à peupler par les instances des sources. Dans notre cas, G est le schéma de l'entrepôt défini lors de la phase précédente, qui est formalisé par le modèle de l'ESBF et connecté au méta-modèle ontologique (conceptuel et linguistique). Ce modèle a été formalisé au chapitre précédent (cf. chapitre 5, section 6). Le schéma considéré est donc une instanciation de ce modèle selon le cas d'étude. Une instance d'un besoin devient donc une instance du modèle de l'ESBF. Ce besoin est décrit sémantiquement par les classes (ou leurs instances) et propriétés de l'ontologie conceptuelle intégrante.
- les schémas des sources S : représentent les sources qui alimentent le schéma cible G. Nos sources de besoins ont été présentées dans le chapitre précédent. Les schémas des sources sont constitués du formalisme local des besoins (par exemple, le formalisme MCT), connecté au méta-modèle ontologique. Les schémas considérés sont donc des instanciation de ce modèle, selon le cas d'étude. De la même façon que pour le schéma global *G*, une instance d'un besoin devient une instance du modèle de besoin local. Ce besoin est décrit sémantiquement par les classes (ou leurs instances) et propriétés de l'ontologie locale.
- les mappings M : représentent l'ensemble des mappings entre les schémas des sources (S) et le schéma global (G). Dans notre étude, les mappings sont définis entre les schémas des sources et le schéma de l'entrepôt, en utilisant des opérateurs ETL. Les mappings sont définis selon une approche GaV (cf. chapitre 2, section 3.1) où le schéma global est défini en fonction des schémas des sources. Par exemple, si le modèle de besoins source est un modèle de cas d'utilisation contenant une entité *Action* utilisant le concept ontologique *Gérer*, un mapping de cette action permet de sauvegarder cette action dans l'entité *Action* du modèle de l'ESBF proposé. Le mapping ontologique (entre les ontologies locales et l'ontologie intégrante) permettra de préciser que *Gérer* correspond au concept *Manage* dans l'ontologie intégrante. L'intégration de ce besoin se fait ainsi en exploitant les mappings entre les formalismes et les mappings ontologiques.
  Ces mappings se basent sur des opérateurs ETL qui permettent l'extraction des instances à partir des sources, leur transformation ainsi que leur chargement dans le schéma cible.

Nous précisions que nous exploitons principalement les schémas conceptuels ontologiques pour (G et S), la traduction vers le schéma logique se fait automatiquement par le SGBD choisi (et sera présenté au prochain chapitre).

Le schéma cible de l'entrepôt et les schémas des sources ayant déjà été présentés, nous nous intéressons dans ce qui suit à la définition des mappings et à leur formalisation.

2.1 Les mappings M

Les mappings permettent de mettre en correspondance les schémas des sources et le schéma global. Les mappings peuvent être formalisés comme suit [35] M:<*MapSchemaG, MapSchemaS, InputSet, OutputSet, Interpretation*>. Cette formalisation est basée sur le meta-modèle de mappings conceptuel proposé dans [29] :

- *MapSchemaG* et *MapSchemaS* : présentent respectivement le schéma global (schéma de l'entrepôt) et les schémas des sources concernées par le mapping.
- *InputSet* et *OutputSet* : *OutputSet* est un élément de *MapSchemaG*, qui est une classe du schéma de l'entrepôt à alimenter. *InputSet* est une expression sur les schémas des sources. Cette expression est construite en utilisant un ensemble d'opérateurs ETL. Skoutas et al. [138] ont défini dix opérateurs génériques d'un processus ETL : Extract, Retrieve, Merge, Union, Join, Store, Detecte-Duplicate, Filter, Convert, Aggregate. Nous adaptons ces opérateurs à notre contexte et nous les enrichissons dans le prochain point.
- *Interpretation* : présente l'interprétation *intentionnelle* (au niveau modèle) ou *extensional* (au niveau instances) du mapping. Une interprétation intentionnelle est généralement utilisée car elle réduit le nombre de mappings.
- *SemanticRelation* : représente la relation sémantique du mapping. Trois types de relations sont définies : *Équivalence*, *Contenance* or *Overlap*. Une relation d'équivalence indique que les éléments connectés par le mapping (*InputSet* et *OutputSet*) représentent le même aspect du monde réel. Aucune transformation n'est requise dans ce cas. La relation de contenance indique que l'élément dans un schéma (*InputSet* ou *OutputSet*) représente un aspect plus spécifique du monde que l'élément dans l'autre schéma. Le chevauchement indique que certains objets décrits par l'élément dans un schéma peuvent également être décrits par l'élément connecté dans l'autre schéma [29]. Ces deux derniers cas nécessitent donc des transformations.

Les mappings que nous exploitons sont définis selon cette formalisation. Nous détaillons dans ce qui suit les opérateurs ETL ainsi que l'algorithme ETL.

2.2 Les opérateurs ETL

Les mappings se basent sur les dix opérateurs génériques du processus ETL, définis dans [138], a savoir : *Extract, Store, Retrieve, Filter, Agregate, Join, Union, DD, Merge, Convert*. Nous adaptons ces opérateurs à notre contexte pour définir des opérateurs ETL définis sur les besoins. Nous allons ensuite enrichir ces opérateurs afin de prendre en compte toutes les transformations nécessaires pour l'intégration des besoins des sources.

Nous organisons ces opérateurs ETL en quatre groupes : (i) opérateurs de base, (ii) opérateurs pour les besoins fonctionnels, (iii) opérateurs d'inférence et (iv) opérateurs de gestion. Dans ce qui suit, nous présentons la signature de chaque opérateur et nous le définissons.

2.2.1 Les opérateurs de base

Les opérateurs de base sont les suivants : *Retrieve, Extract, DD* et *Store*. Nous les détaillons dans ce qui suit :

2.2.1.1 - Retrieve (S_i, F_i, B_j, C_j, TR_j) : cet opérateur est basé sur le principe de l'opération algébrique de sélection. Dans le contexte des données, l'opérateur *Retrieve* permet de rechercher les instances pertinentes au niveau d'une source S_i participant dans le système. Dans notre contexte d'un ETL défini sur les besoins, l'opérateur *Retrieve* est chargé de la récupération du besoin B_j appartenant à une source S_i formalisée avec un formalisme F_i, exprimé avec les classes C_j d'une ontologie conceptuelle locale et les termes TR_i d'une ontologie linguistique locale.

Par exemple, un besoin B_j est récupéré à partir de la source S_1 et formalisé avec le formalisme F_1 (MCT de merise), exprimé avec des concepts C_j de l'ontologie conceptuelle locale OC_1 et des termes TR_j de l'ontologie linguistique OL_1.

2.2.1.2 - Extract (S_i, F_i, B_j, C_j, TR_j, CS_j) : cet opérateur est basé sur le principe de l'opération algébrique de projection. Dans notre contexte d'un ETL sur les besoins, l'opérateur Extract permet la sélection et l'extraction des instances d'un besoin B_j d'une source S_i, formalisée avec un formalisme F_i, qui sont exprimées par des concepts C_j de l'ontologie conceptuelle locale et des termes TR_j de l'ontologie linguistique locale. Seules les instances respectant les contraintes CS_j définies sur les sources sont extraites.

Par exemple, l'application de l'opérateur *Extract* sur un besoin B_j formalisé en utilisant le formalisme F_2 (MCT de merise) en appliquant la condition CS_j (Projection sur *Actor*) donne comme Résultat (*Actor* = "conférenciers").

2.2.1.3 - Détection des doublons DD(B) : cet opérateur permet la détection et la suppression des doublons (les instances des classes en double). Dans notre contexte d'un ETL défini sur les besoins, l'opérateur DD(B) permet la détection des instances des besoins en double dans un ensemble d'instances de besoins B ayant le même format. L'opérateur permet aussi la suppression du doublon.

Par exemple, si on applique cet opérateur sur l'ensemble des besoins que nous avons présenté dans le chapitre 4, défini a partir du document de gestion des cours (CMS), deux besoins dupliqués (B48 et B48') seront détectés et le besoin B48' sera supprimé.

2.2.1.4 - Store(B_j, S_i, C, TR) : l'opérateur S*tore* permet le chargement des données dans un schéma cible. Dans notre contexte d'un ETL défini sur les besoins, l'opérateur *Store* permet le chargement du besoin B_j comme instance du schéma de la source S_i, B_j sera connecté aux

classes C (de l'ontologie conceptuelle intégrante de l'entrepôt) et aux termes TR (de l'ontologie linguistique intégrante). Dans notre cas, S_i est soit la data staging area ou le schéma cible de l'entrepôt.

Notons que dans tout processus ETL, le chargement des instances transite par une zone intermédiaire appelée 'zone de transit des données' (Data Staging Area, DSA) dans laquelle s'effectuent toutes les transformations nécessaires des instances avant leur chargement. Dans nos expérimentations, nous avons exploité une base sémantique Oracle comme DSA. Ce processus est illustré dans la figure 6.1.

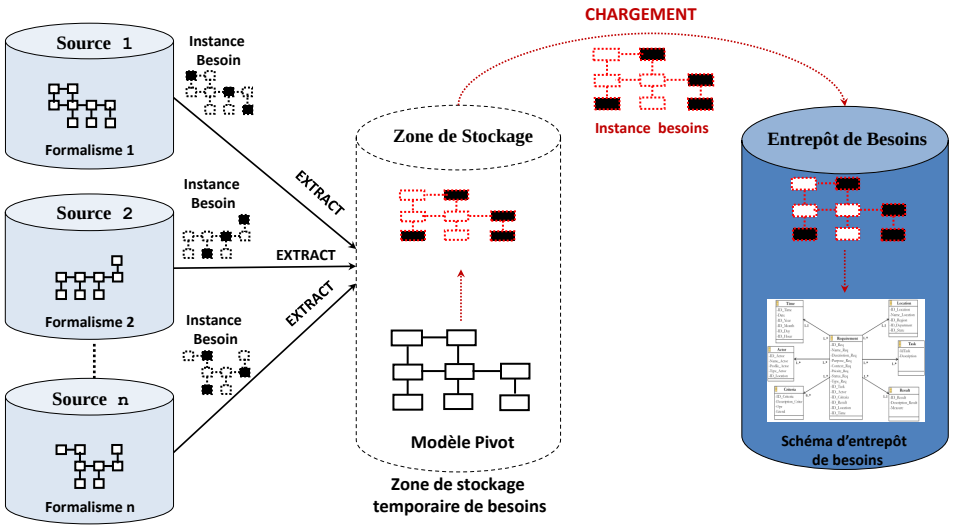

FIGURE 6.1 – Application de l'opérateur STORE pour chargement des besoins

2.2.2 Les opérateurs dédiés aux besoins

Ces opérateurs représentent une spécialisation des opérateurs génériques pour l'intégration des instances des besoins.

2.2.2.1 - Merge (S_i, B_i, B_j, CT) : l'opérateur Merge désigne la fusion des données pour les amener dans une zone de stockage. L'opérateur Merge admet la fusion des données qui contiennent des attributs compatibles appartenant à la même source S_i. Un autre opérateur *Union* sera défini pour l'union des instances provenant de sources différentes. Dans notre contexte d'un ETL défini sur les besoins, l'opérateur Merge permet la fusion de deux besoins B_i et B_j d'une source S_i selon certaines contraintes CT. Le besoin résultat B_T a un argument A_T obtenu à partir de la fusion des arguments $A_i \in B_i$ et $A_j \in B_j$. Notons que deux besoins peuvent être fusionnés quand ils partagent certains arguments d'une tâche (*sujet, action ou objet*) et les autres arguments sont complémentaires.

Cet opérateur peut être utilisé dans le cas le plus général pour unifier des besoins ayant les mêmes tâches mais des critères et des résultats différents. Une autre utilisation se base sur les contraintes CT. Ces dernières peuvent être définies sur la base des relations du graphe de précédence. Elles peuvent également être obtenues sur la base des relations entre les concepts de l'ontologie conceptuelle.

Exemple 6

Soit les trois besoins suivants :

B_1= "The system shall allow lecturers to create teams"

B_2="The system shall allow lecturers to update teams"

B_3= "The system shall allow lecturers to delete teams".

Ces trois besoins partagent un même sujet lecturers et le même objet teams. Une nouvelle action : manage est déduite qui contient les trois actions (create, update et delete), sur la base des relations de contenance. Le besoin résultat est : "The system shall allow lecturers to manage teams" que contiennent les trois besoins B_1, B_2 et B_3.

Exemple 7

Soit les deux besoins suivants :

B_1= "The system shall allow teachers to create teams"

B_2="The system shall allow students to create teams".

Ces deux besoins partagent la même action (create) et le même objet (teams). Un nouveau sujet (Person) est déduit sur la base des relations de subsomption entre les classes que définit l'ontologie conceptuelle : (Person est une superclasse exclusive de student et de teacher). Le besoin résultat est : "The system shall allow Person to create teams". Il contient les deux besoins B_1, B_2.

2.2.2.2 - Union (S_i, S_j, B_i, B_j, CT) : l'opérateur Union permet d'unifier des instances provenant de sources différentes. Dans notre contexte d'un ETL défini sur les besoins, cet opérateur permet l'unification des besoins appartenant à différentes sources $B_i \in S_i$ et $B_j \in S_j$, selon certaines contraintes CT.

Cet opérateur peut être utilisé dans le cas le plus général pour unifier des besoins ayant les mêmes tâches mais des critères et des résultats différents.

Une autre utilisation se base sur les contraintes CT. De la même façon que pour l'opérateur *Merge*, les contraintes CT sont définies sur la base des relations du graphe de précédence. Elles peuvent également être obtenues sur la base des relations entre les concepts de l'ontologie conceptuelle. Nous donnons l'exemple suivant pour illustrer notre propos.

Exemple 8

Soit les deux besoins suivants :

B_1= "The system shall allow teachers to create teams"

B_2="The system shall allow students to create teams".

Ces deux besoins, provenant de sources différentes, doivent être d'abord convertis vers le format intégrateur avant que l'opération d'union soit appliquée. Cette condition est prise en compte dans l'algorithme ETL qui sera présenté dans la prochaine section.

Ces deux besoins partagent la même action (create) et le même objet (teams). Un nouveau sujet est dérivé à partir de l'ontologie conceptuelle où les deux sujets (teacher, student), appartenant respectivement aux besoins B_1 et B_2, sont fusionnés dans un seul sujet person (en utilisant le constructeur subclass de l'ontologie conceptuelle). Le besoin résultat est : "The system shall allow persons to create teams" formalisé en formalisme cible.

2.2.2.3 - Convert (S_i, B_j, fct) : l'opérateur de conversion (*Convert*) est utilisé afin de modifier le format des données, par exemple pour modifier la mesure de valeur d'une donnée.

Dans notre contexte d'un ETL défini sur les besoins, cet opérateur permet la conversion des besoins $B_j \in S_i$ selon une fonction de conversion fct.

Cet opérateur de conversion peut s'appliquer sur les arguments d'un besoin utilisant une propriété de l'ontologie conceptuelle (*"data-type property"*). Par exemple, la conversion de l'attribut date de création d'un besoin à partir d'un format YYYY/MM/DD vers un format DD/MM/YYYY, est illustré dans l'exemple de la figure 6.2.

L'opérateur de conversion concerne également les formalismes utilisés. En effet, cet opérateur est utilisé pour convertir les besoins B_j d'une source donnée S_i définis selon un formalisme ML_i, vers le format intégrateur (modèle cible de l'entrepôt). Dans ce cas, la fonction de conversion fct correspond aux règles de traduction de formalismes décrites dans le chapitre précédent (cf. chapitre 5, section 5).

2.2.2.4 - Filter (S_i, B_j, CS) : l'opérateur *Filter* correspond à la sélection des instances respectant certaines contraintes (CS). Dans notre contexte d'un ETL défini sur les besoins, cet opérateur permet la sélection des besoins B_j appartenant à une source S_i, qui répondent à une ou plusieurs contraintes (CS). Les contraintes sont définies dans l'entrepôt cible sous forme d'axiomes.

Par exemple, l'application de l'opérateur sur l'ensemble des besoins de la source S_2, en appliquant la condition CS_j (*Actor* = "conférenciers") donne comme résultat deux instances de besoins $B_{48'}$ et $B_{48''}$. De même, si l'on souhaite avoir les besoins de la source S_2 filtrés en fonction de la contrainte que l'action du besoin soit *create*, ceci donne comme résultat les trois besoins B_{48}, B_{49} et $B_{98'}$.

2.2.2.5 - Join (S_i, S_j, B_i, B_i, *Relation*) : l'opérateur Join permet d'unir deux entités de la même source ou de sources différentes ayant des attributs communs. Dans notre contexte d'un ETL défini sur les besoins, cet opérateur permet la jointure des besoins $B_i \in S_i$ et $B_j \in S_j$

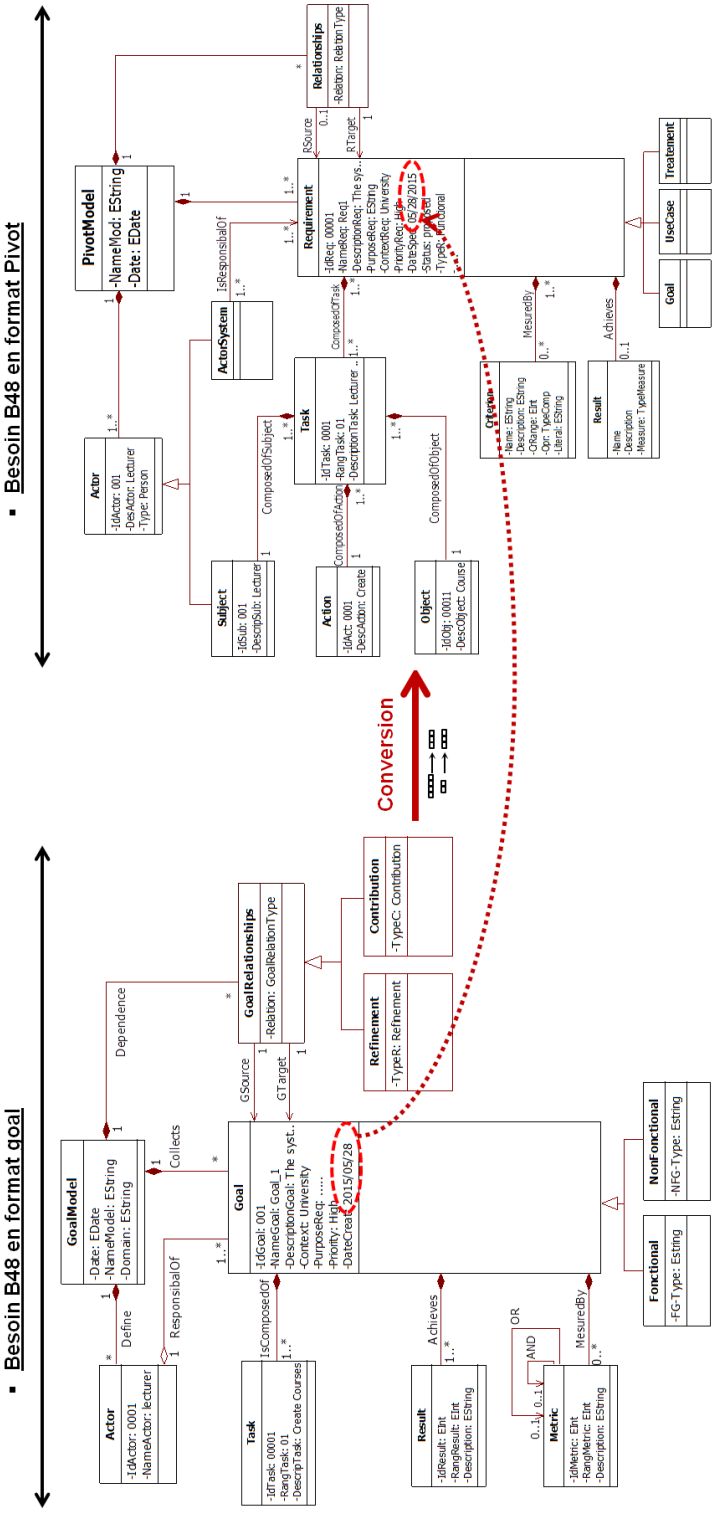

FIGURE 6.2 – Le résultat d'application de l'opérateur *CONVERT* sur les besoins sources

liés par une relation complexe *Relation* (*Contain, Refine, Require, ...*). Ces relations entre les

besoins peuvent être définies dès le chargement des besoins dans l'entrepôt. Comme illustré dans la figure 6.3, les relations (par exemple Require) sont définies entre les besoins au niveau d'une source, mais également entre les besoins une fois ces derniers unifiés dans l'entrepôt.

D'autres opérateurs (*identification* et *Inférence* présentés dans les points suivants) peuvent permettre d'identifier d'autres relations supplémentaires en utilisant les règles de raisonnement.

Nous prenons l'exemple suivant pour illustrer cet opérateur.

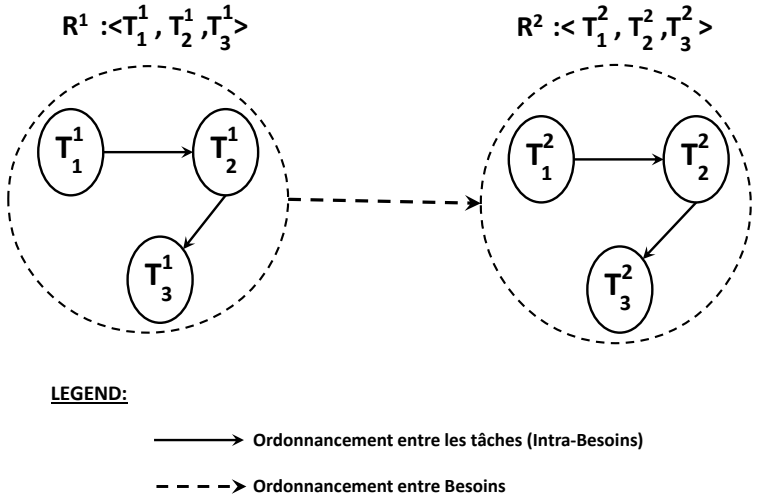

FIGURE 6.3 – L'ordonnancement des tâches entre les besoins

Exemple 9

Soit les deux besoins suivants :

B_1= "The system shall allow administrators to create mail-accounts" de la source S_1

B_2="The system shall allow students to log into mail-account" de la source S_2.

Ces deux besoins partagent le même objet (mail-accounts). Un lien d'ordonnancement (Require (B_2, B_2)) entre les deux besoins doit être défini une fois ces deux besoins unifiés dans l'entrepôt. Ce lien d'ordonnancement indiquera que les étudiants ne peuvent pas se connecter à leurs comptes qu'après la création de ces comptes par les administrateurs. L'unification de ces besoins ne se fait pas avec un simple opérateur Union, mais nécessite un Join.

2.2.2.6 - Aggregate (S_i, B_j, Agr) : l'opérateur *Aggregate* permet de regrouper des instances appartenant à une ou plusieurs sources S_i dans un but d'analyse, en utilisant des fonctions d'agrégation comme : AVG, SUM, MAX, MIN, $COUNT$.

Dans notre contexte d'un ETL défini sur les besoins, cet opérateur applique la fonction d'agrégation *Agr* sur un des arguments du besoin B_j de la source S_i. L'agrégation concerne

généralement les arguments : Critère qui quantifie la satisfaction des besoins et Result qui représente les résultats réalisés par le besoin B_j. Par exemple, prenons quatre besoins ayant les mêmes tâches, et ayant respectivement les critères suivants (B_1 : Age > 20, B_2 : Age < 55, B_3 : age < 40, B_4: age > 30). Nous pouvons regrouper ces besoins, sur la base des valeurs définies par les critères de satisfaction des besoins, en utilisant les fonctions d'agrégation MAX et MIN. Par exemple, le critère Age < 55 et Age > 30 correspond aux critères du besoin unifiant ces besoins dans l'entrepôt.

2.2.3 Les opérateurs d'inférence

Notre objectif étant de faciliter, gérer et optimiser les transformations nécessaires au cours des phases de conception, nous nous intéressons également aux opérateurs permettant la correction des incohérences, inconsistances et l'établissement d'un ordre entre les besoins. Ces opérations permettent d'assurer la qualité du processus ETL et donc la qualité de l'entrepôt final. Suivant cet objectif, nous enrichissons la liste des opérateurs avec certains opérateurs de raisonnement qui seront appliqués avant le chargement des besoins dans l'entrepôt. Nous présentons ci-dessous la signature de ces opérateurs et nous détaillons l'ensemble des règles de raisonnement utilisées par ces opérateurs dans la section suivante :

2.2.3.1 - Identification (B, S, IdentRule) : l'opérateur *Identification* permet le raisonnement sur un ensemble de besoins B d'une source S afin d'identifier les relations entre les besoins telles que *Equal, Contains, Refines, etc.*. Dans notre cas, la source S est la DSA, i.e. la zone de stockage intermédiaire. Cet opérateur utilise un moteur d'inférence et un ensemble de règles d'identification définies sur les besoins *IdentRule* (présentées dans la prochaine section). Cet opérateur est illustré dans la figure 6.4).

2.2.3.2 - Inference (B, S, InfRule) : l'opérateur *Inference* permet d'inférer des nouvelles relations complexes (*Equals, Contains, Refines, etc.*) entre les besoins B d'une source S en se basant sur les relations existantes. Cet opérateur applique des règles d'inférence *InfRule* (présentées dans la prochaine section). Dans notre cas, la source S est la staging area, i.e. la zone de stockage intermédiaire. Cet opérateur utilise un moteur d'inférence. L'opérateur est illustré dans la figure 6.5.

Par exemple, supposons qu'il existe une relation Equals (B_1, B_2) et Contains (B_2 B_3). Par la règle d'inférence : "Equals(B_i, B_j) et Contains(B_j, B_k) alors Contains(B_i, B_k)" nous pouvons en déduire : Contains (B_1, B_3).

2.2.3.3 - CheckConsistency (B, S, ConsistenceCheckRule) : l'opérateur *CheckConsistency* permet le raisonnement sur un ensemble de besoins B d'une source S, en utilisant des règles ConsistenceCheckRule (présentées dans la section suivante) dans le but de vérifier la cohérence

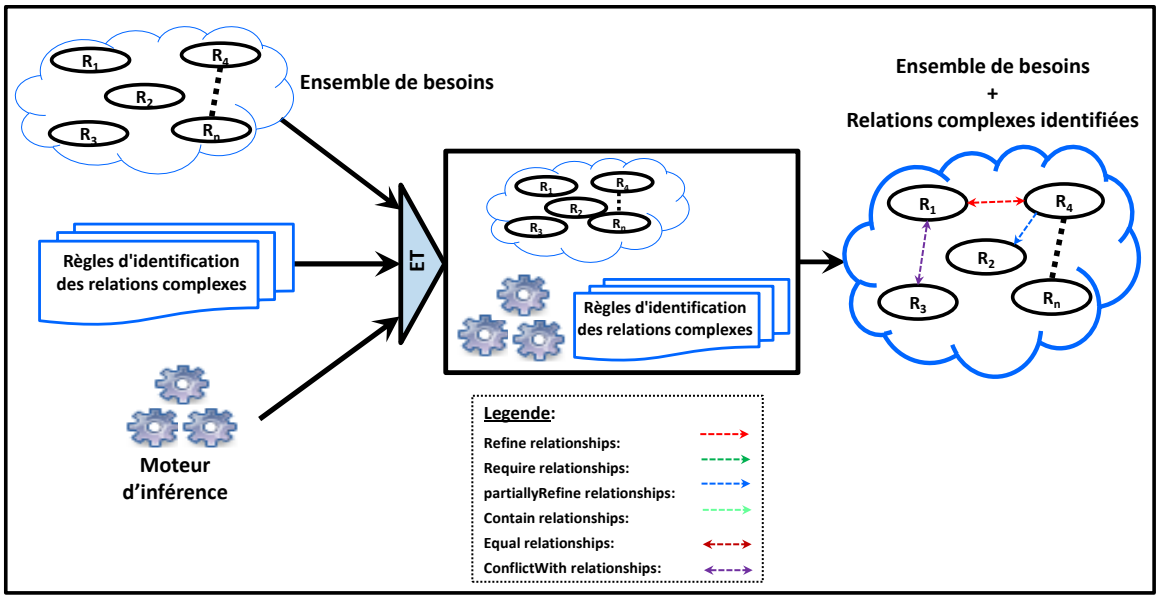

FIGURE 6.4 – Architecture de l'opérateur Identification

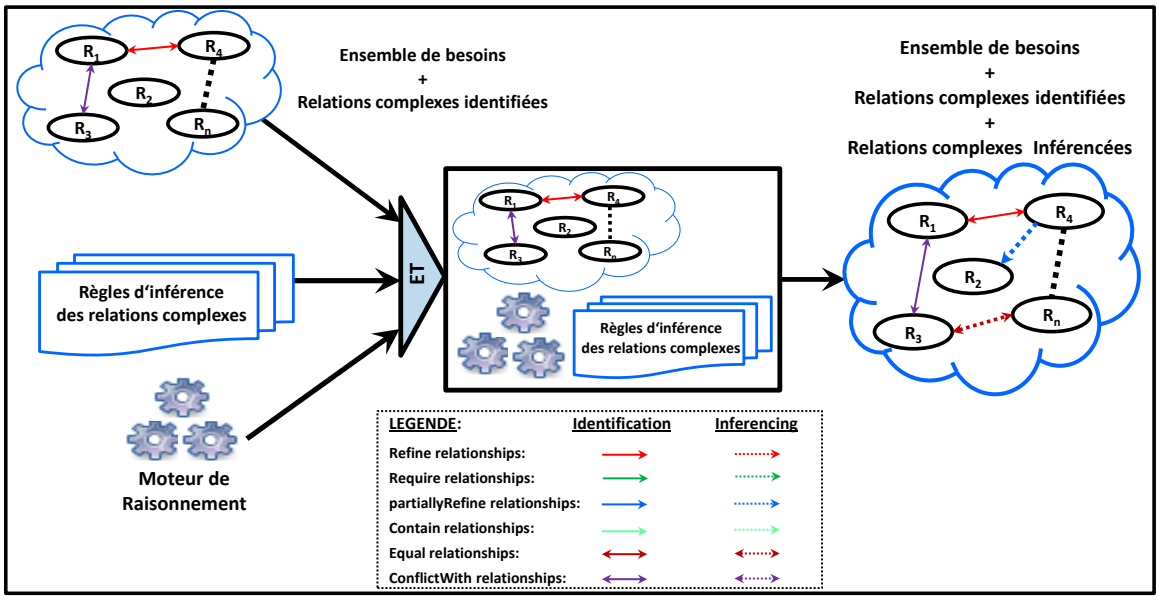

FIGURE 6.5 – Architecture de l'opérateur Inference

des besoins sur la base des cohérences des relations complexes entre eux. Dans notre cas, la source S est la staging area, i.e. la zone de stockage intermédiaire. Cet opérateur utilise un moteur d'inférence. L'opérateur est illustré dans la figure 6.6.

Par exemple, nous supposons que nous avons deux besoins B_1 et B_2 qui contiennent des relations complexes entre eux (B_1 Equals B_2, et B_1 ConflictWith B_2), et nous avons la *règle de la vérification de la cohérence* suivante : Si Equals (B_i, B_j) et Conflicts (B_i, B_j) alors inconsistency (B_j, B_k). Nous pouvons identifier après le raisonnement que les deux besoins B_1 et B_2 sont

inconsistants.

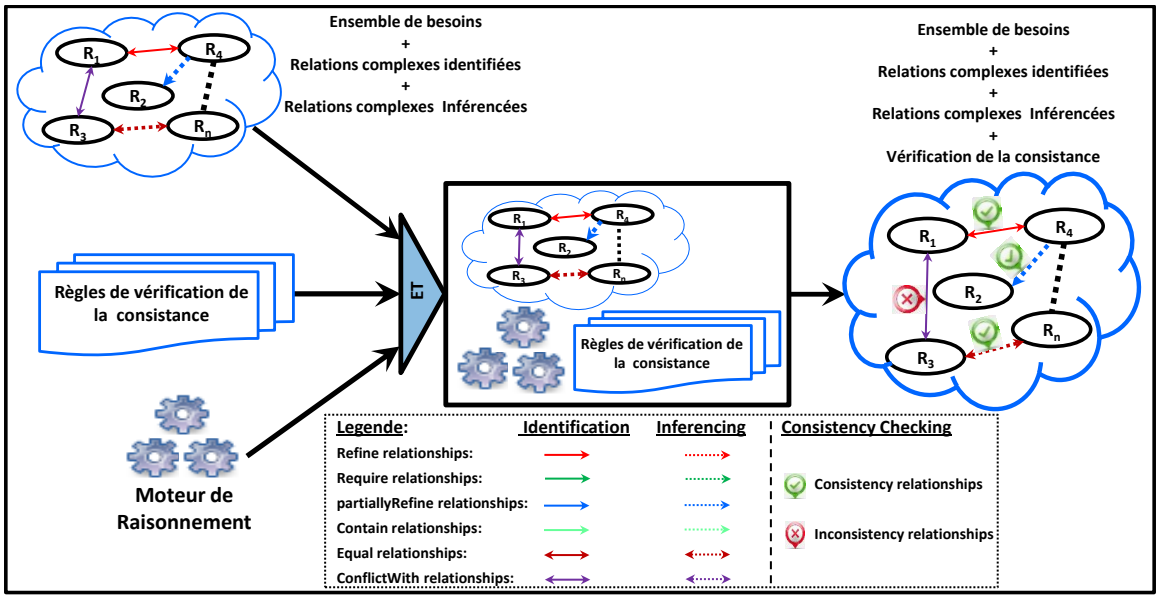

FIGURE 6.6 – Architecture de l'opérateur ConsistencyChecking

2.2.4 Les opérateurs de gestion

Certains opérateurs ont été ajoutés pour gérer les arguments des besoins tels que : *AddSurrogateKey, AddArgument, UpdateArgument, DeleteArgument*. Dans ce qui suit, nous explicitons chaque opérateur :

2.2.4.1 - AddSurrogatekey (S, B) : l'opérateur *AddSurrogatekey* permet une génération automatique de numéros de séquence (*clés de substitution uniques*) pour chaque besoin de l'ensemble B dans la source S. Dans notre cas, S correspond à la staging area.

La figure 6.7 montrer un exemple de génération de clé de séquences pour les besoins et les tâches.

2.2.4.2 - AddArgument(B_j, S, Ar_j) : l'opérateur AddArgument(B_j, S, Ar_j) permet d'ajouter un argument Ar_j à un besoin B_j d'une source S. L'argument peut être une tâche, un critère, un résulat, etc. Cet opérateur est utilisé au cours des opérations de transformation et de fusion des besoins.

2.2.4.3 - UpdateArgument(B_j, S, Ar_j) : l'opérateur UpdateArgument(B_j, S, Ar_j) permet de mettre à jour un argument Ar_j d'un besoin B_j d'une source S. L'argument peut être une tâche,

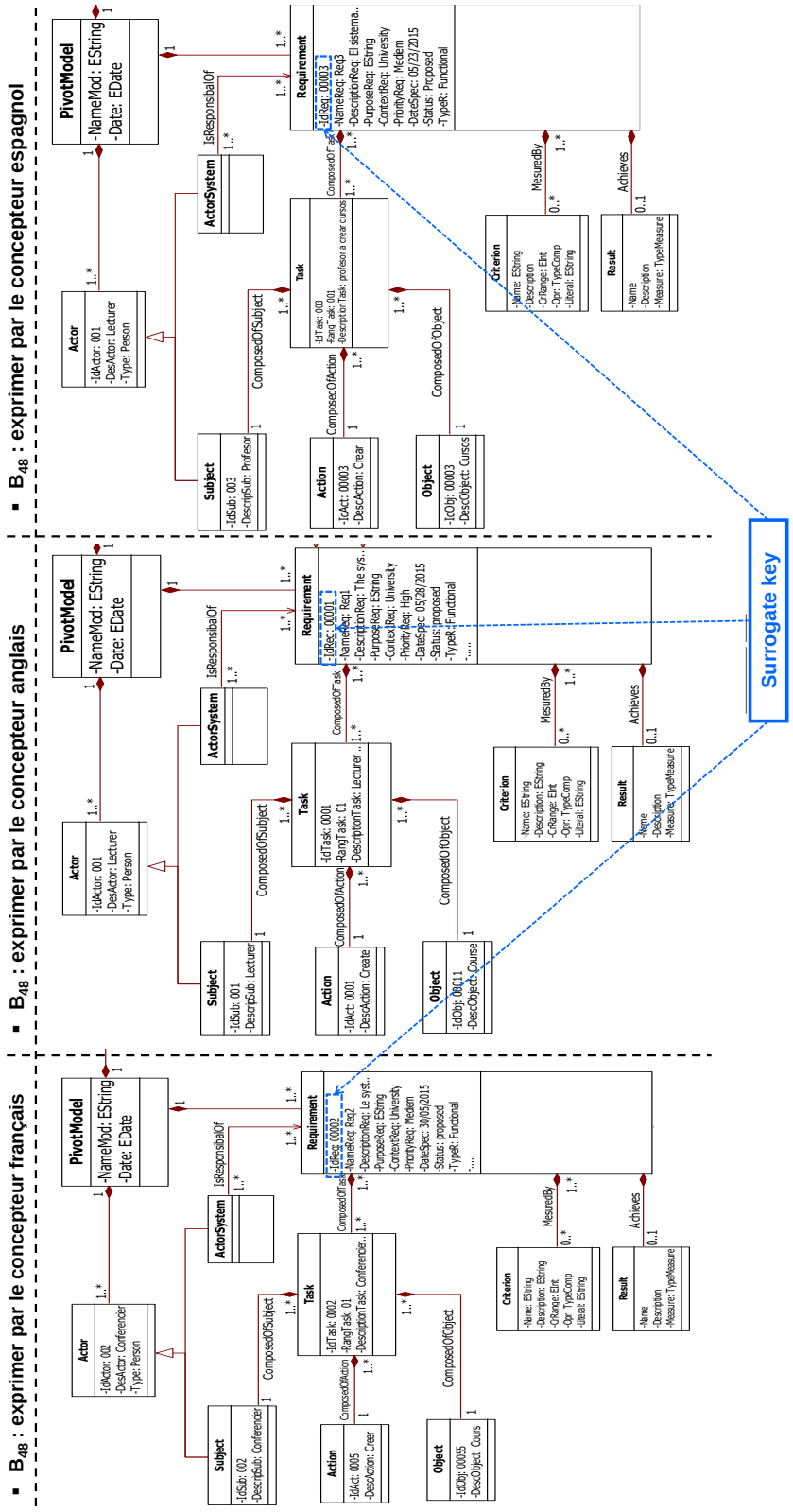

FIGURE 6.7 – Exemple de génération de clé par l'opérateur AddSurrogatekey

un critère, un résulat, etc. Cet opérateur est utilisé au cours des opérations de transformation et de fusion des besoins.

2.2.4.4 - DeleteArgument(B_j, S, Ar_j) : l'opérateur DeleteArgument(B_j, S, Ar_j) permet de supprimer un argument Ar_j d'un besoin B_j d'une source S. L'argument peut être une tâche, un critère, un résulat, etc. Cet opérateur est utilisé au cours des opérations de transformation et de fusion des besoins.

2.3 Le processus ETL de besoins

Les processus ETL se base sur les mappings entre les besoins sources et le schéma cible de l'entrepôt. Le processus ETL utilise les opérateurs ETL définis ci-dessus.

Nous rappelons que l'ensemble des besoins est défini en fonction des concepts ontologiques. Dans nos modèles, nous manipulons principalement les schémas ontologiques (concepts et rôles) et pas les instances. Comme expliqué précédemment (cf. section 2.1), un ensemble de mappings entre les besoins est défini suivant la formalisation M:<$Schema_{DW}$, $Schema_{Sources}$, $Besoin_S$, $Besoins_{DW}$, *interprétation intentionnelle, Relation de Contenance* >.

Un exemple de mapping serait qu'un besoin $Besoin_{DW}$ nécessite la conversion des besoins B_1 de la source S_1 pour qu'ils soient conformes au formalisme cible (Convert(S_1, $Besoin_{S1}$, FormatCible)) et la conversion du besoin B_2 de la source S_2 (Convert(S_2, $Besoin_{S2}$,FormatCible)), puis enfin l'union des ces deux besoins Union(S_1, S_2, B_1, B_2,∅) (en supposant par exemple, que ces besoins partagent la même tâche).

Les mappings sont identifiés semi-automatiquement en fonction des concepts ontologiques utilisés pour définir les besoins. Certains mappings peuvent être identifiés automatiquement car chaque argument d'un besoin est défini par son concept ontologique, et les concepts ontologiques du schéma cible sont définis en fonction des concepts des schémas sources. Par exemple, si un mapping est défini entre les formats des dates dans l'ontologie, les arguments des besoins utilisant ces concepts subiront la même conversion. D'autres mappings doivent être définis manuellement par le concepteur qui reste maître de sa conception en validant chacune des étapes de la conception. La découverte complètement automatique des mappings a fait l'objet de nombreuses études, et sera traitée comme perspective de ce travail.

Selon les mappings définis, le processus ETL permet l'extraction des besoins (en utilisant l'opérateur Extract, Retrieve ou Filter), leur transformation en utilisant les opérateurs nécessaires (agregate, convert, union, les opérateurs utilisant le raisonnement, etc) et leur chargement dans l'entrepôt (en utilisant l'opérateur store). Ce processus ETL est illustré en figure 6.8. Les flux ETL couvrent les étapes suivantes : (i) introduire les besoins sources (InputREQ) participant à la construction de l'entrepôt , (ii) réaliser des opérations unaires ou binaires sur les besoins (ReqOp) , (iii) identifier des relations complexes entre les besoins (REQRelation) , et

(iv) charger les résultats de besoins (REQResult) dans l'entrepôt. Nous définissons ainsi formel-

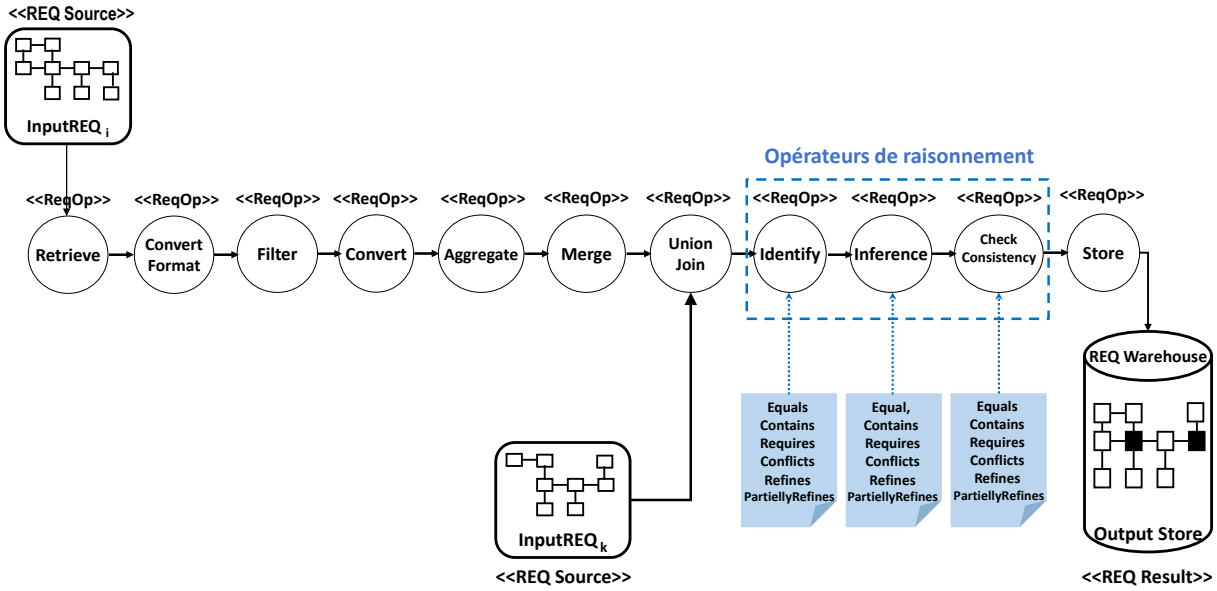

FIGURE 6.8 – ETL de besoins

lement un processus ETL comme suit : <InputREQ, ReqOp, REQRelation, REQResult> (voir figure 6.8), où :

- InputREQ : est un ensemble fini de besoins fonctionnels en entrée (les éléments et les instances du modèle pivot de besoins).
- ReqOp : est un ensemble d'opérateurs du processus ETL (extract, retrieve, merge, union, join, etc.) définis sur les besoins.
- REQRelation : est un ensemble fini de relations (d'identification, d'inférence et de consistance).
- REQResult : est un ensemble de besoins liés par les relations complexes entre eux (ajoutées ou modifiées au cours du processus).

2.4 L'algorithme ETL

L'algorithme 2 formalise le processus ETL présenté ci-dessus. Les principales étapes de l'algorithme sont les suivantes pour chaque source de besoins :

- Extraire les besoins B_j concernés par le mapping de chaque source S_i, en utilisant l'opérateur *Retrieve*.
- Convertir les besoins du format source vers le format intégrateur (*pivot*) en utilisant l'opérateur *Convert*. Les fonctions de conversion sont basées sur les règles de transformation de modèles que nous avons définies dans la section 5 du chapitre 5.

- Filtrer certains besoins d'une source S_i. L'opérateur *Filter* est utilisé, en appliquant une contrainte CS présente au niveau du schéma cible.
- Convertir les arguments des besoins si nécessaire en utilisant l'opérateur *Convert* et une fonction d'agrégation.
- Agréger les besoins en utilisant une fonction d'agrégation et l'opérateur *Aggregate*.
- Stockage temporaire des besoins dans la staging area en utilisant l'opérateur *Store*.
- Si deux besoins B_i et B_j qui doivent être unifiés sont dans la même source, l'algorithme utilise l'opérateur *merge*.
- Si deux besoins B_i et B_j qui doivent etre unifiés sont de sources différentes, l'algorithme utilise l'opérateur *union*.
- Si deux besoins B_i et B_j présentent une relation entre eux, l'algorithme utilise l'opérateur *join* pour les unifier.
- Chargement des besoins sources B_j au niveau du schéma cible de l'entrepôt en utilisant l'opérateur *Store*.
- Ajout d'une clé unique à un besoin B_j en utilisant de l'opérateur *AddSurrogatekey*. Cette opérateur génère automatiquement les clés des nouveaux besoins en fonction des clés existantes au niveau de l'entrepôt.
- Détection et suppression des besoins en double (équivalents) en utilisant l'opérateur *DD*. Cet opérateur agit suivant le résultat des opérateurs de raisonnement.
- Raisonnement sur la base de l'ensemble d'opérateurs *Identification, Inference et Check-Consistency*, afin de détecter l'ensemble des relations complexes entre les besoins et de vérifier leur cohérence.
- Chargement des besoins transformés dans l'entrepôt de besoins.

Notons que les opérateurs *AddArgument, UdpdateArgument, DeleteArgument* et *Exract* sont utilisés lors des transformations requises.

Nous présentons dans ce qui suit les règles de raisonnement utilisées par les opérateurs ETL décrits ci-dessus.

3 Raisonnement sur les besoins

Notre approche permet de raisonner sur les besoins afin d'en extraire de nouvelles relations complexes qui permettront au concepteur de gérer et de valider les besoins du système global. Ces relations sont définies dans notre approche sur la base des différents types de relations entre les concepts et les termes qui sont définis dans l'ontologie conceptuelle et linguistique. Nous commençons par présenter les relations sémantiques considérées dans notre approche. Nous présentons ensuite les règles de raisonnement que nous définies.

begin

1. **Inputs :** schéma $\mathcal{E}ntrepot$, S_i: sources de besoins ($InputREQ$).
2. **Output :** $\mathcal{E}ntrepot$ (schéma + instances) contenant $OutputREQ$.

DSA = ∅;
pour *chaque mapping M* **faire**
  pour *chaque Besoin ($B_j \in M$)* **faire**
    Retrieve (S_i, Formalism, B_j, C_j, Tr_j) //C_j, Tr_j sont les concepts et termes utilisés par ce besoin
    Convert(S_i, B_j, F_{dw}) //F_{dw} présentent les règles de conversion vers le modèle cible de l'entrepôt
    si *(il existe des conditions CF sur les concepts ontologiques cibles)* **alors**
      Filter(S_i, B_j, CF)
    fin
    si *(il existe des conditions CV sur les concepts ontologiques cibles)* **alors**
      Convert(S_i, B_j, CV) // CV présentent les règles de conversion des attributs utilisés
    fin
    si *(il existe des fonction d'agrégation Fct sur les concepts ontologiques cibles)* **alors**
      Aggregate(S_i, B_j, Fct);
      Store (DSA, B_j, C'$_j$, Tr'$_j$) // C'$_j$, Tr'$_j$ sont les concepts ontologiques cible qui correspondent à C_j, Tr_j des sources
    fin
    si *(il existe des conditions CMerge avec un des besoins intégrés de la même source)* **alors**
      Identifier (B_k) //B_k est un besoin à merger avec B_j;
      Merge (DSA, B_j, B_k, CMerge)
    fin
    si *(il existe des conditions CUnion avec un besoin d'une autre source)* **alors**
      Identifier (B_k); Union (S_i, DSA, B_j, B_k, CMerge)
    fin
    si *(il existe des conditions CJoin avec un besoin d'une autre source)* **alors**
      Identifier (B_k); Join (S_i, DSA, B_j, B_k, CJoin)
    fin
  fin
fin
DD(BDW); // BDW est l'ensemble des besoins du DSA ;
Identification (BDW, DSA, *IdentRule*);
Inference (BDW, DSA, *InfRule*);
CheckConsistency (BDW, DSA, *ConsistenceCheckRule*);
Store (DW, BDW)

end

Algorithm 2: $\mathcal{A}$lgorithme de l'$\mathcal{ETL}$ des besoins

3.1 Relations sémantiques entre les besoins : définitions

Plusieurs types de relations peuvent exister entre les besoins, leurs définitions (informelles) dans la littérature [5, 65] sont présentées comme suit : soit R_1, R_2, .. R_n un ensemble de besoins :

- **Relations complexes entre les besoins** : relations qui peuvent exister entre deux ou plusieurs besoins. Six types de relations complexes sont identifiées (*Equals, Conflicts, Contains, Refines, PartiallyRefines, Requires*) :

  - *Relation d'égalité* (*Equals*) :
    R_1 égal à R_2, si et seulement si la réalisation de R_1 a la même influence sur le système que la réalisation de R_2, et les propriétés de R_1 sont exactement les mêmes ou sont équivalentes à celles de R_2 et vice-versa. La relation d'égalité est *symétrique, réflexive* et *transitive* [65].

  - *Relation de conflit*s (*conflicts*) :
    Le besoin R_1 est en conflit avec le besoin R_2, si et seulement si la réalisation de R_1 exclut la réalisation de R_2 et vice versa. Donc la réalisation des deux besoins R_1 et R_2 présente une contradiction dans le système. La relation de conflits est *symétrique, non-réflexive* et elle n'est pas *transitive*. Dans la littérature, il existe plusieurs classifications de conflits entre les besoins. Par exemple, *Van Lamsweerde et al.* [147] distinguent plusieurs relations de conflits entre les besoins : *conflits* (excluant la réalisation simultanée des besoins), *divergence* (cas où les besoins se contredisent dans une forme plus faible que le conflit), *concurrence* (un cas particulier de divergence), et *obstruction* (un cas limite de divergence).

  - *Relation de contenance* (*Contains*) :
    Le besoin R_1 contient (R_2,..., R_n), si et seulement si (R_2,..., R_n) font parti de l'ensemble R_1 (hiérarchie part-whole). La relation de contenance *Contain* est *non réflexive, non symétrique* et *transitive*. Cette relation permet à un besoin complexe d'être décomposé en plusieurs parties [1]. Un besoin composite peut affirmer que le système doit faire $\mathcal{A}$ et $\mathcal{B}$ et C, qui peut être décomposé selon les besoins suivants : le système doit faire $\mathcal{A}$, le système doit faire $\mathcal{B}$, et le système doit faire C. Pour ce rapport, toutes les parties sont nécessaires pour satisfaire le besoin composé.

  - *Relation de raffinement* (*Refines*) :
    Le besoin R_1 raffine le besoin R_2, si et seulement si R_1 est dérivé de R_2 en ajoutant plus de détails à ses propriétés. La relation *Refines* est *non-réflexive*, *non symétrique* et *transitive*.

  - *Relation de raffinement partiel* (*partially Refines*) :

Le besoin R_1 *partiallyRefines* le besoin R_2, si et seulement si R_1 est dérivé de R_2 en ajoutant plus de détails aux propriétés de R_2 et en excluant les propriétés non raffinées de R_2. La relation *Refines* est *non symétrique*, *non-réflexive*, et *transitive*.

– *Relation d'ordonnancement* (*Requires*) :
Le besoin R_1 *requires* le besoin R_2, si et seulement si R_1 est réalisé uniquement lorsque R_2 est réalisé. La relation d'ordonnancement *Requires* peut être considérée comme une condition préalable pour la réalisation du premier besoin. La relation *Requires* est *non symétrique*, *non-réflexive* et *transitive*.

3.2 Mécanisme de raisonnement

Nous avons défini un mécanisme global de raisonnement sous forme d'une *base de règles* (B*aseR*) qui contient trois types de règles de raisonnement : (1) règles d'identification (IdentR) des relations complexes entre les besoins définis ci-dessus ; (2) règles d'inférence (InferR) pour inférer de nouvelles relations complexes entre les besoins, se basant principalement sur les caractéristiques de transitivité des relations identifiées par les règles IdentR ; (3) règles de vérification de la consistance (CheckCR) des besoins.

Cette *base de règles* est formalisée en utilisant le *langage des règles pour le web sémantique* (6.9), en anglais c'est le *Semantic Web Rules Language* (SWRL [61]). Nous rappelons que le méta-modèle SWRL [30], étend notre *framework* proposé (voir la figure 5.13, chapitre 5, section 6), afin de donner la possibilité de définir notre *base de règles*.

La base de règles (BaseR) est composée de trois types de règles de raisonnement : BaseR= < IdentR, InferR, CheckCR > tel que :

– IdentR : un ensemble de règles d'identifications IdentR ∈ IdentR, $identR_i$ ∈ { $identR_{Equal}$, $identR_{Contain}$, $identR_{Refine}$, $identR_{Require}$, $identR_{Conflicts}$, $identR_{partiallyRefine}$ }.
– InferR : un ensemble de règles d'inférence InferR ∈ InferR, $inferR_i$ ∈ { $inferR_{Equal}$, $inferR_{Contain}$, $inferR_{Refine}$, $inferR_{Require}$, $inferR_{Conflicts}$, $inferR_{partiallyRefine}$ }.
– CheckCR : un ensemble de règles de vérification de cohérence pour vérifier la cohérence des relations complexes identifiées (directes et indirectes).

3.2.1 Les règles d'identification

Nous donnons ci-dessous la définition de chaque règle d'identification :

- Requires Relation : intuitivement, cette règle signifie qu'un besoin B_i est en relation d'ordonnancement (*Requires*) avec le besoin B_j si :

(1) tous les sujets (*Subject*), les objets (*Object*) et les contraintes (*Criteria*) des deux besoin

61. *http* : *//www.eclipse.org/atl/usecases/SharingRulesBetweenOCLUML$_S$WRLOWL/*

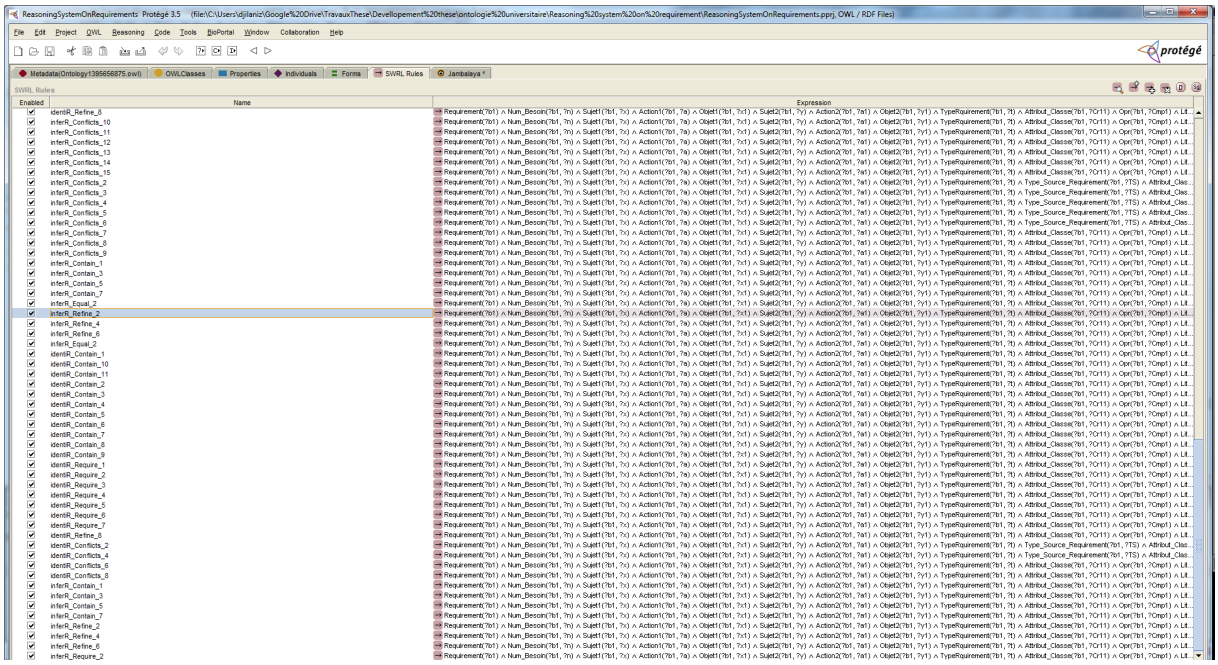

FIGURE 6.9 – Définition des règles en langage SWRL (éditeur Protégé)

sont équivalents ou synonymes. La relation de synonymie entre les entités qui composent un besoins est extraite a partir de l'ontologie linguistique.

(2) il existe au moins une action (*Action*) du besoin B_i qui est en relation d'ordonnancement (*Require*) avec une action (*Action*) du besoin B_j et les autres actions des deux besoins sont en relation d'équivalence ou de synonymie. Cette relation d'ordonnancement (*Require*) entre les actions (verbes) est extraite à partir du graphe de précédence.

– la *relation Requires* est formalisée en langage SWRL comme suit :
Soient B_1 et B_2 deux besoins:
(1) $\wedge$ (2) $\wedge$ (3) $\Rightarrow$ *require* (B_1, B_2), tel que :
Pour $j \geq 0$ et $k \geq 1$:
(1) - $\forall$ $(s_{1.j}, o_{1.j})$,$(s_{2.j}, o_{2.j})$.(($(s_{1.j}, o_{1.j}) \in$ (Domaine(S_1) $\times$ Domaine(O_1)) $\wedge$ $(s_{2.j}, o_{2.j}) \in$ (Domaine(S_2) $\times$ Domaine(O_2))) : (($sameAs(s_{1.j},s_{2.j})$ $\vee$ $similarTo(s_{1.j},s_{2.j})$) $\wedge$ ($sameAs(o_{1.j},o_{2.j})$ $\vee$ $similarTo(o_{1.j},o_{2.j})$)))

(2) - $\forall$ $(a_{1.j}, a_{1.k})$, $(a_{2.j}, a_{2.k})$.(($(a_{1.j}, a_{1.k}) \in$ Domaine$(A_1)^2$ $\wedge$ $(a_{2.j}, a_{2.k}) \in$ Domaine$(A_2)^2$ $\wedge$ $(a_{1.j} \neq a_{1.k}) \wedge (a_{2.j} \neq a_{2.k})$) : (($sameAs$ $(a_{1.j}, a_{2.j})$ $\vee$ $synonymTo$ $(a_{1.j}, a_{2.j})$ $\vee$ $RequireAction(a_{1.j}$, $a_{2.j})$) $\wedge$ ($RequireAction(a_{1.k}, a_{2.k})$)))

(3) - $\forall$ $(c_{1.j}, c_{2.j})$.($(c_{1.j}, c_{2.j}) \in$ (Domaine(C_1) $\times$ Domaine(C_2)) $\Rightarrow$ $(c_{1.j} \supseteq c_{2.j})$)

Notons que la relation *Requires* peut être modélisée sous forme d'un graphe, que nous appelons graphe de précédence et qui permet de gérer l'ordonnancement entre les besoins. L'algorithme ETL proposé manipule cette structure de graphe pour les relations *Requires*.

Nous précisions également que nous avons effectué cette formalisation en langage SWRL pour la totalité des règles citées ci-dessous.

- Equals Relation : intuitivement, cette règle signifie qu'un besoin B_i est en relation d'équivalence (*Equal*) avec le besoin B_j si :

- tous les sujets (*Subject*), les actions (*Action*), les objets (*Object*) et les contraintes (*Criteria*) des deux besoins B_i et B_j sont équivalents ou synonymes.

- Refines Relation : intuitivement, cette règle signifie qu'un besoin B_i est en relation de raffinement (*Refines*) avec le besoin B_j si :

(1) tous les sujets (*Subject*), les actions(*Action*) et les objets (*Object*) des deux besoins B_i et B_j sont équivalents ou synonymes.

(2) il existe au moins une contrainte (*Criteria*) du besoin B_i qui est incluse dans un autre contrainte (*Criteria*) du besoin B_j et les autres critères des deux besoins sont en relation d'équivalence ou de synonymie.

- PartiallyRefines Relation : intuitivement, cette règle signifie qu'un besoin B_i est en relation de raffinement partielle (*PartiallyRefines*) avec le besoin B_j si :

(1) tous les sujets (*Subject*), les actions(*Action*) et les objets (*Object*) des deux besoins B_i et B_j sont équivalents ou synonymes.

(2) il existe au moins une contrainte (*Criteria*) du besoin B_i qui est incluse dans une autre contrainte (*Criteria*) du besoin B_j.

- Conflicts Relation : intuitivement, cette règle signifie qu'un besoin B_i est en relation de conflit (*Conflicts*) avec le besoin B_j si :

(1) tous les sujets (*Subject*), les objets (*Object*) et les critères (*Criteria*) des deux besoins B_i et B_j sont équivalents ou synonymes.

(2) il existe au moins une action (*Action*) du besoin B_i qui est en relation d'antonymie avec une autre action (*Action*) du besoin B_j et les autres actions (verbes) des deux besoins sont en relation d'équivalence ou de synonymie. La relation d'antonymie (opposite) entre les actions est définie dans le l'ontologie linguistique.

- Contains Relation : intuitivement, cette règle signifie qu'un besoin B_i est en relation de Contenance (*Contains*) avec le besoin B_j si :

(1) tous les actions (*Action*) et les contraintes (*Criteria*) des deux besoins B_i et B_j sont équivalents ou synonymes.

(2) il existe au moins une sujet (*Subject*) ou un objet (*Object*) du besoin B_i qui représente respectivement une super-classe d'un autre sujet (*Subject*) ou d'un objet (*Object*) du besoin

B_j et les autres sujets (*Subject*) ou objets (*Object*) des deux besoins sont en relation d'équivalence ou de synonymie. La relations de hiérarchies (subclass, et superclass) entre les entités que définissent les deux besoins B_i et B_j sont extraites à partir de l'ontologie conceptuelle.

3.2.2 Les règles d'inférence

Le processus d'inférence automatique consiste à dériver de nouvelles relations en se basant sur les relations définies. Ces règles utilisent principalement les caractéristiques de transitivité des relations identifiées.

Par exemple, considérons soient les besoins B_1, B_2, B_3. Si on a la relation suivante $Refine(B_1, B_2) \wedge Refines\ (B_2, B_3)$, alors la relation $Refine\ (B_1, B_3)$ est aussi vraie. De la même façon, nous avons formalisé les règles d'inférences présentées ci-dessous :

- $\text{Refines}(B_i, B_j) \wedge \text{Refines}(B_j, B_k) \Rightarrow \text{Refines}(B_i, B_k)$
- $\text{Refines}(B_i, B_j) \wedge \text{Contains}(B_k, B_j) \Rightarrow \text{Refines}(B_i, B_k)$
- $\text{Contains}(B_i, B_j) \wedge \text{Contains}(B_j, B_k) \Rightarrow \text{Contains}(B_i, B_k)$
- $\text{Equals}(B_i, B_j) \wedge \text{Equals}(B_j, B_k) \Rightarrow \text{Equals}(B_i, B_k)$
- $\text{Contains}(B_i, B_j) \wedge \text{Conflicts}(B_j, B_k) \Rightarrow \text{Conflicts}(B_i, B_k)$
- $\text{Refines}(B_i, B_j) \wedge \text{Conflicts}(B_i, B_k) \Rightarrow \text{Conflicts}(B_k, B_j)$
- $\text{Conflicts}(B_i, B_j) \wedge \text{Conflicts}(B_j, B_k) \Rightarrow \text{Conflicts}(B_k, B_i)$
- $\text{PartiallyRefines}(B_i, B_j) \wedge \text{PartiallyRefines}(B_j, B_k) \Rightarrow \text{PartiallyRefines}(B_i, B_k)$
- $\text{Contains}(B_i, B_j) \wedge \text{PartiallyRefines}(B_k, B_j) \Rightarrow \text{Contains}(B_i, B_k)$
- $\text{PartiallyRefines}(B_i, B_j) \wedge \text{Conflicts}(B_i, B_k) \Rightarrow \text{Conflicts}(B_k, B_j)$
- $\text{Refines}(B_i, B_j) \wedge \text{PartiallyRefines}(B_j, B_k) \Rightarrow \text{PartiallyRefines}(B_k, B_i)$
- $\text{Requires}(B_i, B_j) \wedge \text{Requires}(B_j, B_k) \Rightarrow \text{Requires}(B_i, B_k)$
- $\text{Requires}(B_i, B_j) \wedge \text{Equals}(B_j, B_k) \Rightarrow \text{Requires}(B_i, B_k)$
- $\text{Requires}(B_i, B_j) \wedge \text{Contains}(B_j, B_k) \Rightarrow \text{Requires}(B_i, B_k)$
- $\text{Equals}(B_i, B_j) \wedge (B_j \text{ Conflicts } B_k) \Rightarrow \text{Conflicts}(B_i, B_k)$

3.2.3 Les règles de vérification de la cohérence

Ces règles permettent d'identifier automatiquement, parmi les relations existantes, celles qui provoquent une contradiction. Par exemple, soient les besoins B_1, B_2, si les règles suivantes $Equals\ (B_1, B_2) \wedge Conflicts\ (B_1, B_2)$ sont vérifiées, nous pouvons déduire que les deux besoins B_1, B_2 ont un problème de consistance. Dans ce qui suit, nous définissons les relations les plus pertinentes qui nous permettent de détecter les inconsistances :

- $\text{Refines}\ (B_i, B_j) \wedge \text{Refines}\ (B_j, B_i) \Rightarrow \text{Inconsistency}(B_i, B_j)$.
- $\text{Refines}\ (B_i, B_j) \wedge \text{Contains}(B_i, B_j) \Rightarrow \text{Inconsistency}(B_i, B_j)$.
- $\text{Contains}(B_i, B_j) \wedge \text{Contains}(B_j, B_i) \Rightarrow \text{Inconsistency}(B_i, B_j)$.

- Contains(B_i, B_j) ∧ Conflicts(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Refines(B_i, B_j) ∧ Conflicts(B_i, B_j) ⇒ Inconsistency(B_k, B_i).
- PartiallyRefines(B_i, B_j) ∧ Conflicts(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Equals(B_i, B_j) ∧ Conflicts(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Conflicts(B_i, B_j) ∧ Requires(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Equals(B_i, B_j) ∧ Refines(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Equals(B_i, B_j) ∧ Requires(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Equals(B_i, B_j) ∧ PartiallyRefines(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Equals(B_i, B_j) ∧Contains(B_j, B_i) ⇒ Inconsistency(B_i, B_j).
- Refines(B_i, B_j) ∧ Requires(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Refines(B_i, B_j) ∧ PartiallyRefines(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- PartiallyRefines(B_i, B_j) ∧ Requires(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- Contains(B_i, B_j) ∧ Requires(B_i, B_j) ⇒ Inconsistency(B_i, B_j).
- PartiallyRefines(B_i, B_j) ∧ Contains(B_i, B_j) ⇒ Inconsistency(B_i, B_j).

De cette façon, nous avons formalisé les trois types de règles de raisonnement de la base de règles BaseR (règles d'identification, d'inférence et de consistance). Les opérateurs de raisonnement (présentés dans la section précédente) qui sont utilisés dans le processus ETL sont basés sur ces règles de raisonnement proposées. Le résultat de l'exécution de ces règles permet soit d'enrichir le schéma de l'ESBF par les nouvelles relations identifiées ou inférées (classe *Relationships* dans le modèle) , soit de filtrer les besoins à charger. Si deux besoins sont identifiés comme inconsistants, le choix est laissé au concepteur de les charger ou pas. Nous proposons les actions suivantes que peut prendre le concepteur du système : soit les deux besoins R_i et R_j :

- (1) si R_i $refine/PartiallyRefine$ R_j, le concepteur peut garder R_j et éliminer R_i ;
- (2) si R_i *equal* R_j, selon une politique de choix (date d'expression, source prioritaire, attribut priorité attribué au besoin, etc.) le concepteur peut garder un des deux besoins et éliminer l'autre ;
- (3) si R_i *Contains* R_j, le concepteur peut garder R_i et éliminer R_j ;
- (4) si R_i est inconsistant avec R_j, le concepteur doit choisir un des deux besoins en définissant une stratégie d'élimination ;
- (5) si R_i *require* R_j, le concepteur doit tenir compte de l'ordonnancement des besoins dans le processus de conception du système (d'abord R_j, ensuite R_i).

Nous présentons dans ce qui suit les expérimentations proposées afin de valider les contribution de ce chapitre (raisonnement et ETL).

4 Implémentation et Expérimentation

Nous présentons d'abord une série d'expérimentations permettant de valider le mécanisme de raisonnement proposé. Nous proposons ensuite une deuxième série d'expérimentation permettant d'évaluer la complexité de l'algorithme ETL.

4.1 Performances du raisonnement

Nous commençons par présenter l'environnement d'implémentation puis nous présentons les expérimentations que nous avons réalisées.

Notre environnement d'implémentation utilise l'éditeur ontologique *Protégé* [62],qui est un éditeur libre et *open-source*. *Protégé* est considéré comme l'éditeur ontologique le plus abouti et le plus utilisé ; il est fortement soutenu par les développeurs et aussi par la communauté académique (utilisé par une communauté de plus de 76000 personnes). Dans ce chapitre, nous pouvons évalué le raisonnement sur l'éditeur ontologique *protégé* puisque nous n'avons pas encore procédé au déploiement physique de l'entrepôt sur un SGBD. Dans le prochain chapitre, cette même expérimentation sera reconduite après le déploiement de l'entrepôt sur le SGBD sémantique d'Oracle.

Dans toute l'étude d'évaluation, nous avons exploité le langage ontologique OWL. Nous avons utilisé la même étude de cas du scénario "Ontologie partagée" présentée au chapitre 4, section 3. Nous rappelons et détaillons cette étude de cas :

1. l'ontologie multilingue (*EuroWordNet*) [63].
2. l'ontologie *Univ-Bench* du banc d'essai universitaire (LUBM) que nous avons considéré comme l'ontologie intégrante de notre approche. Cette ontologie est formalisée en utilisant le langage OWL-Lite afin de permettre un raisonnement décidable [70].
3. le plug-in *ProSé* [64] pour la modularité d'ontologies afin d'extraire l'ontologie locale de chaque source ($LocalO_1$, $LocalO_2$, $LocalO_3$) à partir de l'ontologie partagée (Ontologie de domaine LUBM).
4. cent besoins extraits à partir du document de gestion de besoins CMS. Ces besoins sont définis dans différents formalismes :
   - trente-cinq besoins représentés selon le formalisme orienté but (Goal), exprimés en langue anglaise (par un partenaire *Anglais*).
   - vingt-cinq besoins représentés selon le modèle conceptuel des traitements (MCT), exprimés en langue française (par un partenaire *Français*).
   - quarante besoins représentés selon le formalisme des cas d'utilisation (UML), exprimés en langue espagnole (par un partenaire *Espagnol*).
5. les règles de mises en correspondance des formalismes sources avec le formalisme cible (pivot) sont définies et exécutées en utilisant le plug-in ATLAS *Transformation Language* (ATL) [65] de l'IDE Eclipse utilisant le langage Java.

62. http://protege.stanford.edu/
63. http://www.illc.uva.nl/EuroWordNet/
64. http://krono.act.uji.es/people/Ernesto/safety-ontology-reuse
65. http://www.eclipse.org/atl/

6. le moteur de raisonnement $Jess$ [66] .

Comme illustré en figure 6.10, les règles d'identification s'exécutent un seule fois et les règles d'inférence et de vérification de la consistance s'exécutent plusieurs fois jusqu'à ce qu'aucun nouveau fait ne soit inféré.

Afin d'assurer l'évaluation de chaque partie de notre approche, nous avons suivi les étapes illustrées dans la figure 6.11.

Cette évaluation est donc réalisée en quatre principales étapes, chaque étape évaluant un Framework, i.e. une partie du mécanisme de raisonnement global. Les Frameworks sont présentés ci-dessous. Nous effectuons ensuite une étude comparative entre les quatre Frameworks :

- (1) le premier framework (EPivot) est constitué du modèle de besoins de l'entrepôt, présenté dans l'étape 1 de la figure 6.11. Les instances des besoins sources sont donc définis avec les termes utilisés dans le document CMS. Ces instances sont transformées selon le modèle cible de l'entrepôt en utilisant les règles de mappings définies entre les formalismes. A cette étape, nous n'utilisons pas les ontologies.
- (2) le deuxième framework (OntoEPivot) est constitué du modèle de besoins connecté à l'ontologie conceptuelle. Il est présenté dans les deux étapes 1 et 2 de la figure 6.11. A cette étape, nous exploitons l'ontologie conceptuelle uniquement. Les ontologies locales conceptuelles des trois sources sont extraites à partir de l'ontologie *Univ-Bench*. Une fois l'ontologie locale extraite, une opération de couplage de cette dernière avec le formalisme source correspondant est établie (dans notre cas d'étude : $OntoGoal$, $OntoUseCase$, $OntoMCT$). Les concepteurs sources peuvent décrire leurs besoins en fonction des concepts de cette ontologie locale. Par la suite, une connexion est établie entre l'ontologie globale conceptuelle et le modèle de besoins. Cette connexion se fait avec l'éditeur Protégé, en étendant le méta-modèle ontologique (OWL) par de nouvelles classes définissant le modèle de besoins. L'algorithme ETL est ensuite exécuté, exécutant ainsi le système de raisonnement basé sur les règles de raisonnement définies.
- (3) le troisième framework (OntoDLEPivot) est constitué du modèle de besoins connecté à l'ontologie conceptuelle et à l'ontologie linguistique. Il est présenté dans les trois étapes 1, 2 et 3 de la figure 6.11. A cette étape, nous exploitons l'ontologie conceptuelle et aussi l'ontologie linguistique. Les ontologies locales conceptuelles des trois sources sont extraites à partir de l'ontologie *Univ-Bench*. Les ontologies locales linguistiques des trois sources sont extraites à partir de l'ontologie *EuroWordnet*. Une fois les ontologies locales extraites, une opération de couplage de ces dernières avec le formalisme source correspondant est établie. Les concepteurs sources peuvent décrire leurs besoins en fonction des concepts et des termes de ces ontologies locales. Par la suite, une connexion est établie entre l'ontologie globale conceptuelle, l'ontologie linguistiques globale et le modèle de besoins. L'algorithme ETL est ensuite exécuté, exécutant ainsi le système de raisonnement basé sur les règles de raisonnement définies.

66. http://www.jessrules.com

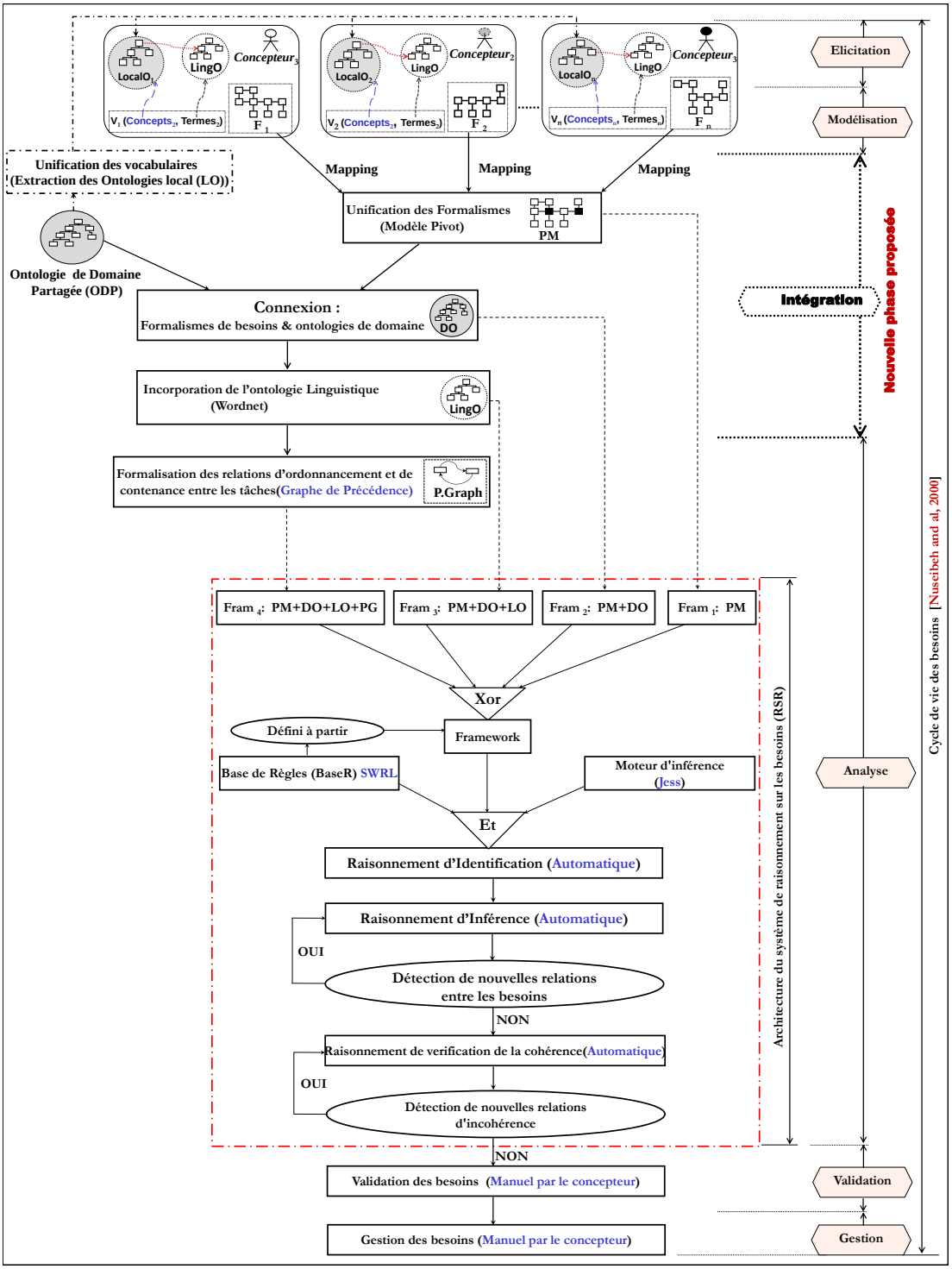

FIGURE 6.10 – Architecture et étapes d'exécution du mécanisme de raisonnement sur les besoins

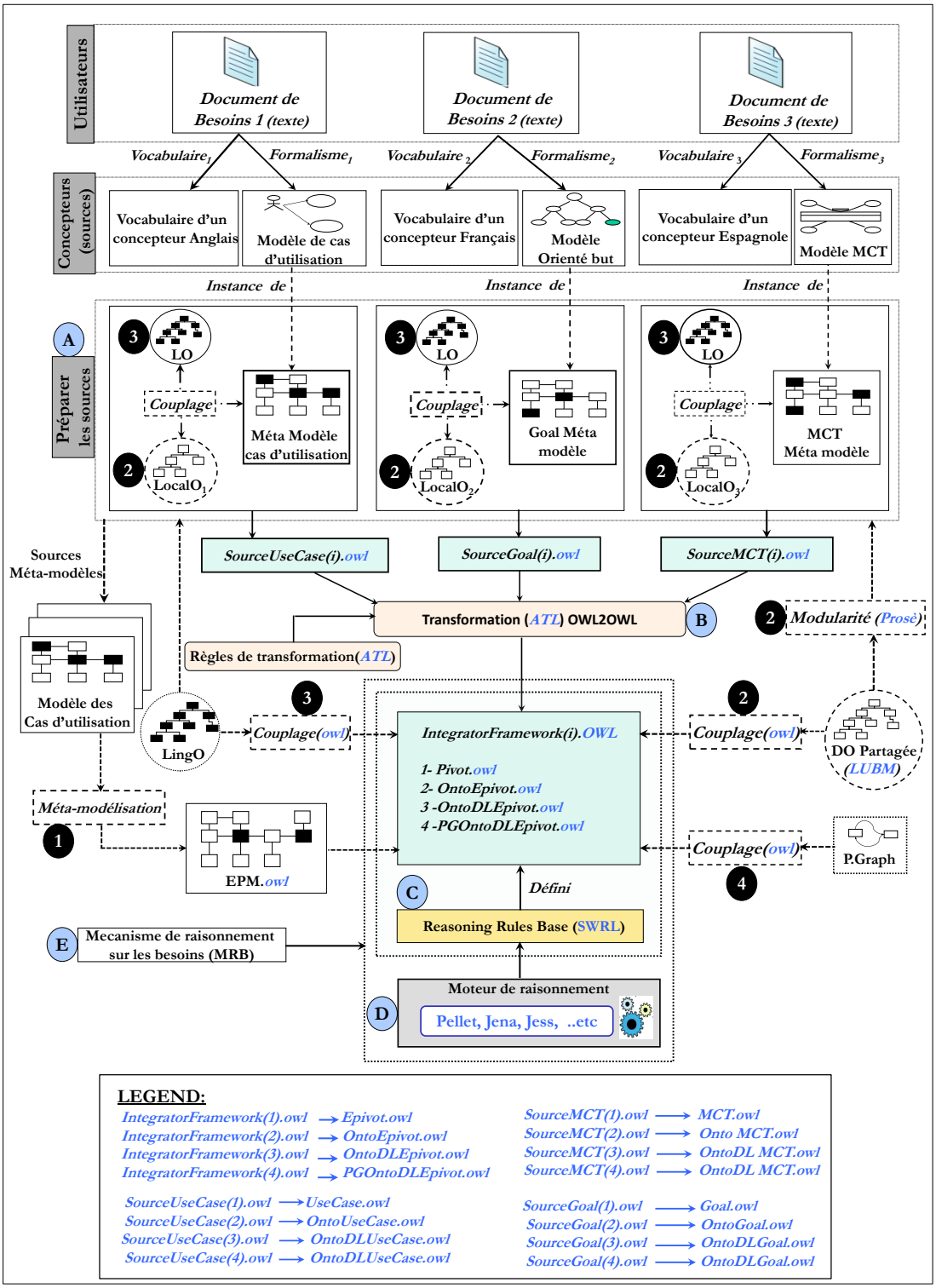

FIGURE 6.11 – Les étapes (1,2,3,4) de validation de notre approche

– (4) le quatrième framework (PGOntoDLEPivot) est constitué du modèle de besoins connecté à l'ontologie de domaine (LUBM) à l'ontologie linguistique (*EuroWordNet*) et au graphe de précédence définissant l'ordonnancement des relations *Require*. Le framework est présenté dans les quatre étapes 1, 2, 3 et 4 de la figure 6.11. A cette étape, nous exploitons tout le mécanisme de raisonnement. Les ontologies locales conceptuelles et linguistiques sont extraites et connectées aux formalismes des besoins. Le graphe de précédence est défini au niveau de l'ontologie intégrante. L'algorithme ETL est ensuite exécuté, exécutant ainsi le système de raisonnement basé sur les règles de raisonnement définies.

Cette décomposition de l'approche en étapes permet de montrer la faisabilité et l'intérêt de chaque partie du framework pour le raisonnement (l'ontologie conceptuelle, l'ontologie linguistique et le graphe de précédence). Notre approche est testée sur les aspects suivants : (i) le taux

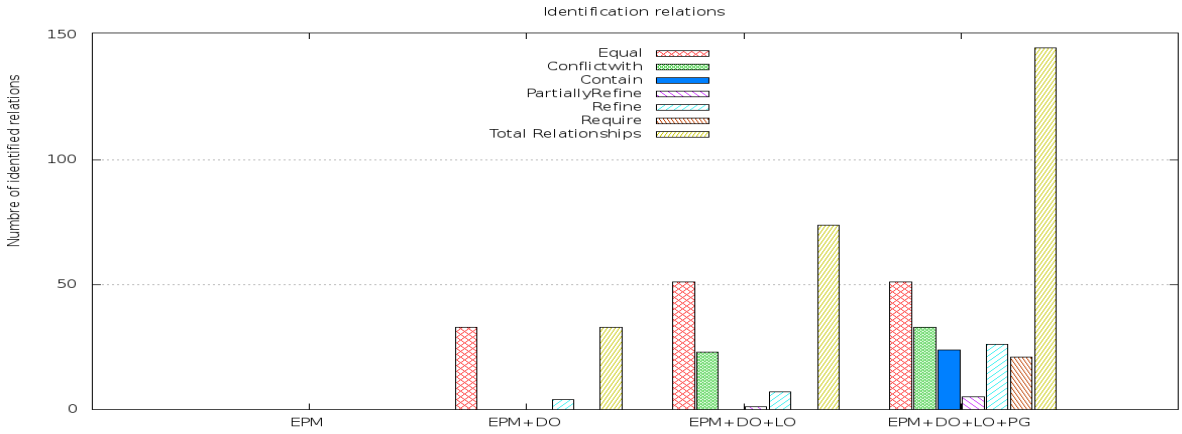

FIGURE 6.12 – Relations détectées dans chaque framework proposé

de relations détectées entre les besoins pour chaque *framework* défini ci-dessus , (ii) le temps d'exécution de chaque *framework* et (iii) l'évolutivité de l'approche.

Les expérimentations sont exécutées sur un poste de travail avec un processeur *Intel* (*R*), *Core* (*TM*) i7-4770 *CPU* 3,40 *GHz*, 8 *GO* de *RAM*, sous le système d'exploitation *Windows* 7 *Professionnel* 64 *bits*. Les figures 6.12 et 6.13 montrent que notre *framework* proposé identifie un nombre de relations entre les besoins plus important qu'une approche d'analyse conventionnelle de besoins. Les inconsistances entre les besoins sont également mieux identifiées avec notre approche. Les aspects d'évolutivité et de temps d'exécution sont vérifiés par raisonnement sur 20 sous-ensembles de besoins $s_1, ..., s_{20}$ de besoins, où la taille de chaque sous-ensemble s_i est de $5 * i$ besoins, $1 =< i <= 20$. La figure 6.14 montre que notre approche est exécutée dans un délai de temps raisonnable et peut être évolutive.

Dans le point suivant, nous évaluons la complexité de l'algorithme ETL de besoins proposé.

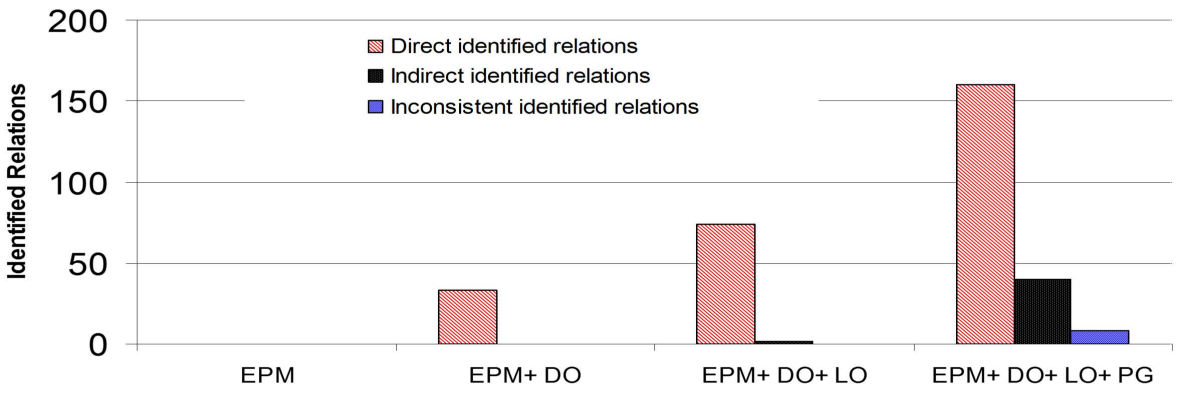

FIGURE 6.13 – Identification des relations inconsistantes.

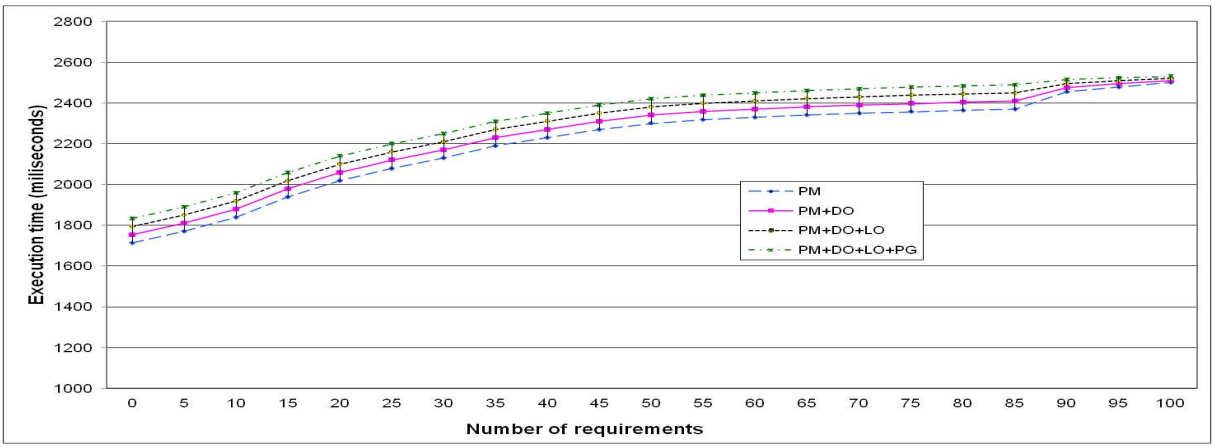

FIGURE 6.14 – évolutivité et temps d'exécution.

4.2 Complexité de l'algorithme ETL de besoins

L'algorithme ETL que nous avons proposé permet d'extraire les besoins des sources, d'effectuer plusieurs transformations sur ces besoins et de les charger dans l'entrepôt. Nous avons testé notre algorithme ETL sur le même scénario (ontologie partagée) décrit précédemment. Nous le testerons dans le prochain chapitre sur l'entrepôt déployé. Afin de tester l'algorithme ETL, nous examinons le nombre d'itérations de l'algorithme afin d'alimenter (peupler) l'entrepôt par les besoins. La complexité du temps d'exécution de l'algorithme est $O(n)$, où n est le nombre de concepts impliqués dans le processus ETL (les concepts du schéma cible). La figure 6.15 montre le nombre d'itérations pour les concepts impliqués dans le schéma de l'entrepôt. Il indique un temps d'exécution polynomial.

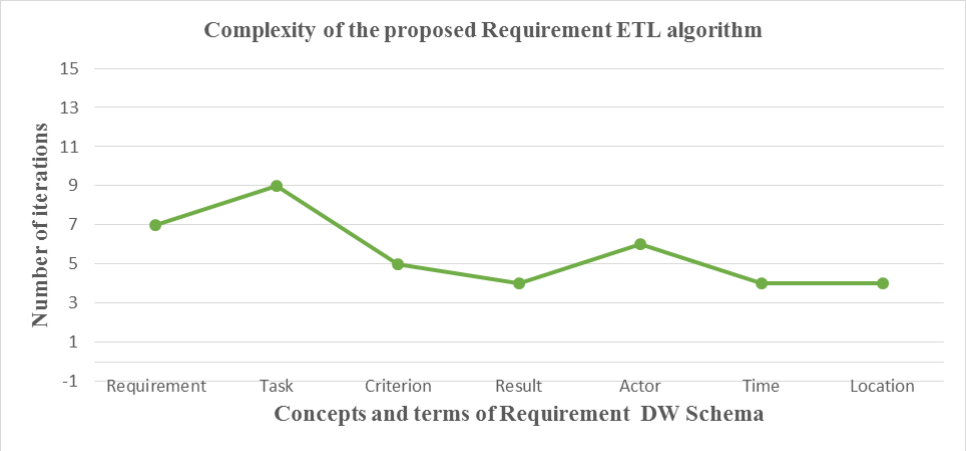

FIGURE 6.15 – Complexité de l'algorithme d'ETL de besoins proposé

5 Conclusion

Dans ce chapitre, nous avons présenté la phase ETL adaptée aux sources de besoins. Cette adaptation concerne toutes les étapes de la phase ETL, plus précisément : les schémas sources et cibles, l'établissement des mappings entre ces schémas, les opérateurs ETL et le processus ETL pour l'intégration des instances des besoins. De nouveaux opérateurs spécifiques aux besoins ont également été définis. Pour gérer le mécanisme de raisonnement sur les besoins que nous avons mis en place, nous avons également défini trois opérateurs (*identification, inference, ConsistencyCheck*).

La validation de notre proposition a été effectuée par l'évaluation du mécanisme de raisonnement ainsi que l'évaluation de la complexité de l'algorithme ETL proposé. Dans le prochain chapitre, nous présentons une approche de déploiement physique de l'entrepôt dans un SGBD choisi, et nous présentons des techniques d'analyse des besoins de l'entrepôt cible.

Chapitre 7

Déploiement de l'Entrepôt Sémantique de Besoins Fonctionnels

Sommaire

1 Introduction 165
2 La conception logique de l'entrepôt 166
3 La conception physique de l'entrepôt 167
3.1 Déploiement de l'entrepôt de besoins sur une BDBO Oracle . . . 168
3.1.1 Construction du schéma de l'entrepôt 169
3.1.2 Traduction des opérateurs ETL 170
3.2 Déploiement de l'entrepôt sur une BDBO OntoDB 175
4 Exploration et analyse des besoins entreposés 176
5 Evaluation : ETL et raisonnement 178
5.1 Scénario 1 : ontologie partagée 179
5.1.1 Les besoins sources 179
5.1.2 Moteur d'inférence 179
5.1.3 Machine physique: 180
5.1.4 Performances de l'ETL et du raisonnement 180
5.2 Scénario 2 : ontologies multiples 181
5.2.1 Performance de l'ETL 181
5.2.2 Performance du mécanisme de raisonnement 183
6 Présentation de l'outil d'unification des besoins 184
6.1 Architecture fonctionnelle de l'outil 184
6.2 Fonctionnement de l'outil 186
6.2.1 Chargement et visualisation des entrées de notre système 186
6.2.2 Mécanisme de matching 188
6.2.3 Processus ETL 189
6.2.4 Mécanisme de raisonnement 190

7 Conclusion 191

Résumé. Dans ce chapitre nous procédons aux phases finales de déploiement et d'analyse de l'entrepôt de besoins fonctionnels. Suivant le même effort d'adaptation des processus conventionnels d'un projet d'entreposage, le déploiement de l'entrepôt doit comprendre les trois étapes suivantes : le choix du système de gestion de bases de données (SGBD) cible, la phase de conception logique et la phase de conception physique. L'objectif ultime d'un projet d'entrepôt est de pouvoir analyser les objets entreposés par diverses techniques. Nous proposons d'évaluer ces phases par l'implémentation de l'entrepôt de besoins sur un SGBD sémantique industriel Oracle ainsi qu'un SGBD académique développé au laboratoire OntoDB. La phase d'analyse est assurée par des requêtes OLAP et l'implémentation d'un cube multidimensionnel des besoins unifiés. Des expérimentations sont réalisées afin d'évaluer les performances du processus ETL et du mécanisme de raisonnement pour les deux scénarios : ontologie partagée et ontologies multiples.

1 Introduction

Un système d'entreposage doit permettre l'*unification*, le *stockage* et l'*analyse* des objets entreposés. Dans notre contexte d'entreposage des besoins, l'*unification* des besoins est réalisée aux trois niveaux (terminologique, conceptuel et formalisme) et un processus ETL a été défini pour l'unification des instances dans le schéma de l'entrepôt. Cette intégration s'est faite par une approche à base d'ontologies, qui sont efficaces grâce à leurs caractéristiques de spécification et de raisonnement.

Le *stockage* des besoins se fait au sein d'un SGBD cible choisi. Ce stockage comporte une phase de conception logique puis une phase de conception physique du schéma de l'entrepôt. Dans notre approche, le SGBD en question peut être conventionnel ou sémantique. L'ontologie intégrante est couplée au modèle cible de l'entrepôt, et n'est pas partie intégrante du modèle cible. Le stockage de l'ontologie reste donc un choix du concepteur. Nous avons opté pour les solutions de stockage sémantiques car elles permettent d'exploiter les caractéristiques ontologiques. L'implémentation de l'entrepôt de besoins sera ainsi réalisée sur deux SGBD sémantiques : le SGBD commercial Oracle et le SGBD académique OntoDB. Cette implémentation ne concerne pas uniquement le schéma de l'entrepôt de besoins mais également les opérateurs ETL définis dans le chapitre précédent. Afin d'exécuter le processus ETL, chaque opérateur est traduit en une requête Sparql s'exécutant sur les sources. Les performances de raisonnement ontologiques validées dans le chapitre précédent au niveau ontologique uniquement (sur l'éditeur d'ontologie Protégé) seront reconduites sur le SGBD Oracle.

L'entrepôt implémenté fourni ainsi une vision unifiée et multidimensionnelle des besoins issus de sources fortement hétérogènes. L'*analyse* des besoins entreposés peut se faire via diverses techniques. Nous avons opté pour la définition de requêtes OLAP sur l'entrepôt et par la définition d'un cube multidimensionnel basé sur le modèle multidimensionnel que nous avons défini. Ce cube permet d'effectuer des analyses OLAP plus pointues, qui sont généralement requises par les décideurs d'une organisation.

Des expérimentations sont réalisées afin d'évaluer les performances du processus ETL et du mécanisme de raisonnement pour les deux scénarios : ontologie partagée et ontologies multiples.

A l'issue de ce chapitre, nous aurons fourni une solution d'entreposage couvrant les principales phases de conception et d'exploitation du cycle de vie d'un projet d'entreposage. Ce cycle comprend les phases de : définition des besoins, modélisation conceptuelle (chapitre 4), phase ETL (le chapitre 5 pour la définition des mappings ontologiques et le chapitre 6 pour le processus ETL), la phase de modélisation logique, la phase de modélisation physique et la phase d'analyse (chapitre 7). Pour montrer la faisabilité de l'approche dans sa globalité, nous avons développé un outil Case implémentant les principales étapes de l'approche.

Ce chapitre est structuré selon les sections suivantes. La deuxième section présente la conception logique du schéma de l'entrepôt. La troisième section présente la conception physique de

l'entrepôt aux deux niveaux : schéma et instances. La quatrième section présente les aspects analytiques des besoins entreposés. La cinquième section présente l'évaluation de notre approche après le déploiement physique. La sixième section présente l'outil implémentant l'approche globale. La dernière section conclut ce chapitre.

2 La conception logique de l'entrepôt

Une fois le schéma multidimensionnel de l'entrepôt conçu, la phase de conception logique requiert la traduction de ce schéma de l'ESBF vers un schéma logique. Cette traduction peut être représentée en utilisant l'un des modèles relationnels sémantiques connus dans la littérature à savoir : le modèle horizontal, le modèle vertical ou le modèle binaire (cf. chapitre 2, section 4.6.1). Nous rappelons que la représentation *verticale* permet de structurer les instances par une table unique de trois colonnes (*sujet, prédicat* et *objet*). La traduction du schéma conceptuel de l'entrepôt de besoins en un schéma logique selon la représentation verticale est illustrée dans la figure 7.1). Chaque ressource (concept ou rôle) du schéma de l'entrepôt (*Actor, Task, Criterion, Result, Period et Location*) est décrite par un ensemble de triplets. Le nom de la ressource est représenté par la colonne *Subjet*. Le type de ressource décrit par le triplet est représenté par la colonne *Predicate*. L'objet décrit par la ressource est représenté par la colonne *Objet*. La

Triples		
Subject	**Predicate**	**Object**
Req#1	Type	Requirement
Req#1	Requirement_ID	Ident1
Req#1	Requirement_Name	Name1
Req#1	Purpose_Req	Purpose1
Req#1	Context_Req	Context1

FIGURE 7.1 – Schéma de l'entrepôt de besoins selon l'approche verticale

traduction du schéma conceptuel de l'entrepôt de besoins en un schéma logique selon la représentation *horizontale* est illustrée dans la figure 7.2). Chaque classe du schéma de l'entrepôt est représentée par une table où chaque propriété de cette classe est représentée par une colonne. La traduction du schéma conceptuel de l'entrepôt de besoins en un schéma logique selon la représentation *binaire* est illustrée dans la figure 7.3). Dans cette représentation, les classes et propriétés du schéma de l'entrepôt correspondent à des tables ayant des structures différentes. Nous présentons dans ce qui suit le déploiement physique de l'entrepôt.

Requirement

ID_Req	Name_Req	Description_Req	Purpose_Req	Context_Req	Priority_Req	Status_Req	Type_Req	ID_Task	ID_Actor	ID_Criteria	ID_Result	ID_Location	ID_Time
ID_Req#1	Name#1	Description#1	Purpose#1	Context#1	Priority#1	Status#1	Type#1	ID_Task#1	ID_Actor#1	ID_Criteria#1	ID_Result#1	ID_Location#1	ID_Time#1
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Task

Task_ID	Rank_Task	Description_Task	ID_Subject	ID_Action	ID_Object
ID_Task#1	Rank#1	Description#1	ID_Subject#1	ID_Action#1	ID_Object#1
...	...	...	...	...	...

Actor

ID_Actor	Name_Actor	Profile_Actor	Type_Actor
ID_Actor#1	Name_Actor#1	Profile_Actor#1	Type_Actor#1
...	...	...	...

Result

ID_Result	Description_Result	Measure
ID_Result#1	Name#1	Profile#1
...	...	...

Location

ID_Location	Name_Location	ID_Region	ID_Department	ID_State
ID#1	Name#1	Region#1	Department#1	State#1
...	...	...	...	...

Criteria

ID_Criteria	Description_Criteria	Profile_Criteria	Type_Criteria
ID#1	Description#1	Profile#1	Criteria#1
...	...	...	...

Time

ID_Time	Date	ID_Year	ID_Month	ID_Day	ID_Hour
ID#1	Date#1	Year#1	Month#1	Day#1	Hour#1
...	...	...	...	...	...

FIGURE 7.2 – Schéma de l'entrepôt de besoins selon l'approche horizontale

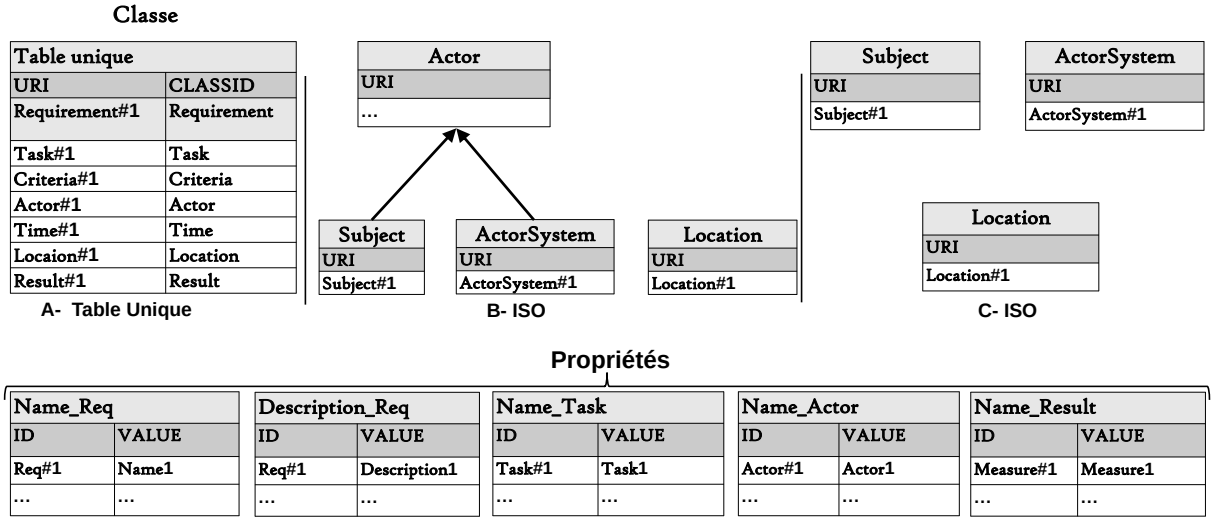

FIGURE 7.3 – Schéma de l'entrepôt de besoins selon l'approche binaire

3 La conception physique de l'entrepôt

La phase de conception finale de l'entrepôt de besoins consiste à implémenter le schéma logique de l'entrepôt dans un SGBD cible. La figure 7.4 montre le processus de déploiement de la structure de l'entrepôt de besoins. Afin de stocker les besoins dans l'entrepôt, nous proposons d'étudier la faisabilité de cette phase sur deux architectures de BDBO : une architecture de type I en utilisant la base sémantique d'Oracle et une architecture de type III en utilisant la base sémantique d'OntoDB. Les détails sur les types d'architectures de stockage existantes sont fournis dans la section 4.6.1 du chapitre 2.

Nous présentons dans ce qui suit les détails de la conception physique sur ces deux BDBO :

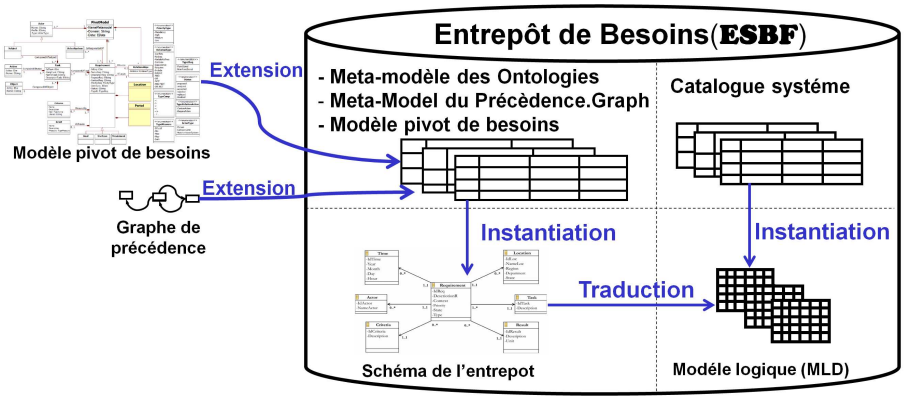

FIGURE 7.4 – Déploiement de la structure d'entrepôt sémantique de besoins fonctionnels

Oracle et OntoDB.

3.1 Déploiement de l'entrepôt de besoins sur une BDBO Oracle

La BDBO d'Oracle suit une architecture de type I dans laquelle les instances et le schéma ontologiques sont stockés dans le même schéma sans distinction. Ce schéma de stockage suit une approche verticale sous forme d'une table à trois colonnes. Oracle est basé sur la sémantique de RDF [67], ce qui permet d'interroger l'entrepôt en utilisant le langage d'interrogation du web sémantique SPARQL [68]. Pour les aspects de raisonnement, Oracle repose sur OWLPrime qui est une famille de la logique de description (LD) utilisant un certain nombre d'opérateurs LD.

Dans Oracle, les données sont des triplets RDF stockés dans un schéma central [10]. Tous les triplets des graphes RDF sont stockés dans la base de données sous ce schéma central. Seules les références (IDs) à ces triplets sont stockées dans les tables d'application définies par l'utilisateur, qui peuvent contenir d'autres attributs liés aux triplets. Des fonctions sont fournies pour récupérer les triplets réels si nécessaire. Le méta-schéma ontologique d'Oracle contient plusieurs tables nécessaires pour le stockage des données : *rdf_link*$, *rdf_node*$, *rdf_value*$, *rdf_blank_node*$ et *rdf_model*$. Dans ce qui suit, nous présentons ces différents tables du RDF :

- la table *rdf_model*$ permet de stocker les noms et les IDs de tous les graphes RDF dans la base de données. *rdf_model*$ contient plusieurs attributs (*MODEL_ID*, *MODEL_NAME*).
- la table *rdf_value*$ permet de stocker les valeurs de texte (*i.e. URIs, blank nodes, et literals*) pour un triplet. Chaque entrée de texte est stockée de manière unique. *rdf_value*$ contient plusieurs attributs(*VALUE_ID*, *VALUE_NAME*, *VALUE_TYPE*, *LITERAL_TYPE*, *LONG_VALUE*).
- la table *rdf_node*$ permet de stocker les noms et les IDs des valeurs de texte qui parti-

67. https://www.w3.org/RDF/
68. https://www.w3.org/TR/rdf-sparql-query/

cipent en tant que sujets ou objets de triplets. *rdf_node*$ contient plusieurs attributs(*NODE_ID*, *ACTIVE*).

– la table *rdf_link*$ permet de stocker les triplets pour tous les graphes RDF dans la base de données. La table est partitionnée pour améliorer les performances des requêtes. *rdf_node*$ comprend les colonnes suivantes (*LINK_ID*, *START_NODE_ID*, *P_VALUE_ID*, *END_NODE_ID*, *LINK_TYPE*, *ACTIVE*, *CONTEXT*, *REIF_LINK*, *MODEL_ID*).

– la table *rdf_blank_node*$ permet de stocker éventuellement les noms et les IDs des noeuds vierges qui doivent être réutilisés lorsqu'ils sont rencontrés dans les triplets entrants. Les valeurs *Blank-nodes* ne sont généralement pas réutilisées comme les autres valeurs de texte. *rdf_blank_node*$ utilise les colonnes suivantes (*NODE_ID*, *NODE_NAME*, *ORIG_NAME*, *MODEL_ID*).

Dans ce qui suit, nous présentons les étapes de construction du schéma de l'entrepôt sémantique de besoins fonctionnels portant sur une BDBO d'oracle ainsi que la traduction des opérateurs ETL que nous avons définis sur les besoins fonctionnels :

3.1.1 Construction du schéma de l'entrepôt

La figure 7.5 présente le résultat de notre première étape où nous avons étendu le méta-schéma sémantique d'Oracle par le modèle de besoins que nous avons défini. Comme le montre

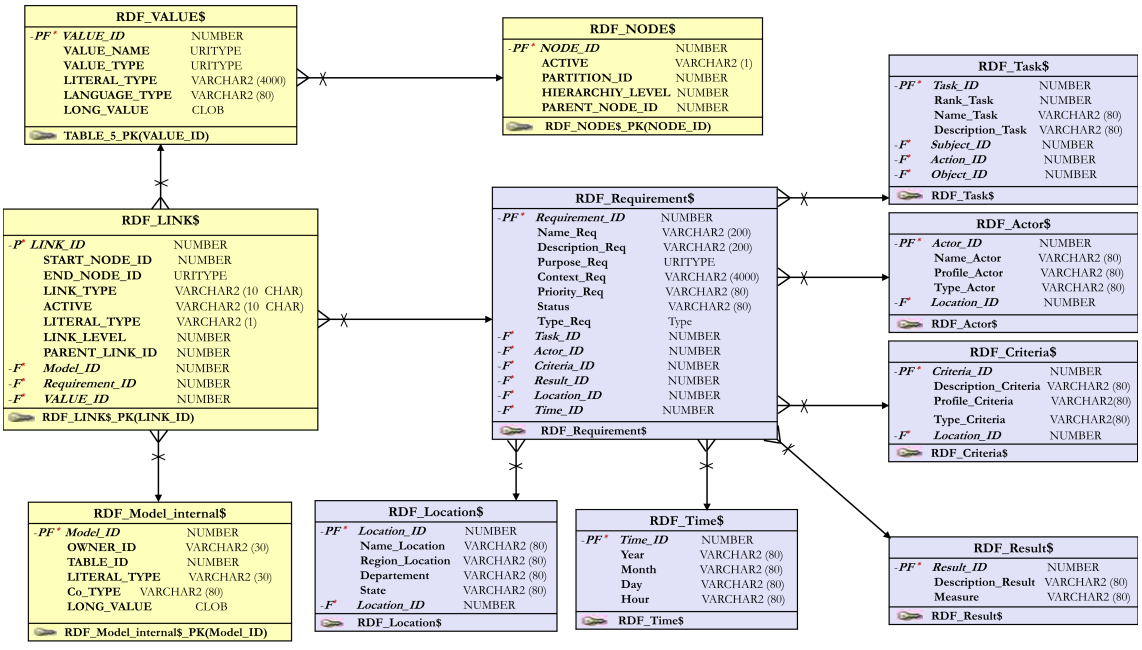

FIGURE 7.5 – Extension du méta-schéma sémantique d'Oracle avec le modèle de besoins

la figure, nous avons défini les composantes (entités) du modèle pivot de besoins par des classes et leurs attributs. Chaque classe du modèle pivot (*Requirement, Actor, Task, Criteria, Result, Location et Time*) représente une nouvelle méta-classe dans le méta-schéma d'Oracle. Chaque

attribut, par exemple (Requirement : *Requirement_ID*, *Requirement_Name*, *Description_Req*, *etc*.) est représenté par un attribut de la méta-classe correspondante sous forme d'une *propriété data-value*.

3.1.2 Traduction des opérateurs ETL

Afin de peupler le schéma de l'entrepôt par les instances des sources, nous avons exploité l'algorithme ETL que nous défini. Chaque opérateur est représenté par une requête Sparql. Dans ce qui suit, nous donnons la requête correspondante à chaque opérateur ETL. Notons que nous considérons *req* comme le nom de l'espace de stockage du schéma de l'entrepôt . Nous illustrons nos propos avec des exemples extraits de l'étude de cas présentée précédemment, où l'url de l'ontologie est la suivante :
PREFIX req:< *http* : *//www.owl − ontologies.com/OntoReqUnivWordnet.owl#* >.

– l'opérateur *Retrieve* : l'opérateur *Retrieve* permet de récupérer les instances "instances-Besoin" appartenant à l'une des classes "ClassREQ" de l'entrepôt. La traduction de l'opérateur en SPARQL donne :

```
Select ?instancesBesoin# Where
                          {?instancesBesoin# rdf:type ESBFSpace:ClassREQ}
```

Exemple 10

Pour récupérer l'ensemble des besoins intégrés, nous utilisons la requête suivante :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\RETREIVE: permet de récupérer l'ensemble des besoins.
Select ?instancesREQ where
                           {?InstancesREQ rdf:type req:Requirement}
```

– l'opérateur *Extract* : l'opérateur *Extract* permet l'extraction (projection verticale) des instances "instanceBesoin" de la classe "ClassREQ" en utilisant certaines composantes du besoin avec une condition (Condition) sur des propriétés (DataProperty). Cet opérateur se traduit en SPARQL comme suit :

```
Select ?instancesBesoin ?Property where
        {?InstancesBesoin# prefix:property ?Property
         ?Property Condition}
```

Exemple 11

Extraire les besoins (InstanceREQ) et leurs priorité, tel que ces besoins ont la priorité (Priority_Req) la plus élevée (High), se traduit en SPARQL par :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\EXTRACT: permet d'effectuer une projection sur les des besoins fonctionnels
           suivant une condition Condition.

Select ?instanceREQ# ?Priority\_Req where
```

```
{?instanceREQ# req:hasPriority ?Priority\_Req
?InstanceREQ# rdf:type req:Requirement.?InstanceREQ req:Priority_Req='high'}
```

– l'opérateur *Union* : l'opérateur *Union* permet l'unification des besoins "instanceBesoin" appartenant à des classes de sources différentes "ClassREQ1" et "ClassREQ2", avec ou sans conditions. Cet opérateur se traduit en SPARQL comme suit :

```
Select ?instancesBesoin# where
          {?InstancesBesoin# rdf:type prefix1:ClassREQ1}.
          Union
          {?InstancesBesoin# rdf:type prefix2:ClassREQ2}.
```

Exemple 12

l'union des besoins (InstanceREQ) que contiennent des relations de contenance entre eux dans l'ESBF se traduit en SPARQL par :

```
PREFIX req1:<http://www.owl-ontologies.com/ontoS1.owl\#>
PREFIX req2:<http://www.owl-ontologies.com/ontoS2.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\UNION: permet l'union des besoins fonctionnels

  Select ?instanceREQ# where
            {?instanceREQ# rdf:type req1:ClassREQ}.
            Union
            {?InstancesREQ# rdf:type req2:ClassREQ}.
```

– l'opérateur *Merge* : l'opérateur *Merge* permet la fusion des besoins "instanceBesoin" appartenant à deux classes "ClassREQ1" et "ClassREQ1" de la même source, avec ou sans conditions. Cet opérateur est utile lorsque l'on souhaite fusionner des besoins dans la staging area. Cet opérateur se traduit en SPARQL comme suit :

```
Select ?instancesBesoin# where
          {?InstancesBesoin# rdf:type prefix:ClassREQ1}.
          Union
          {?InstancesBesoin# rdf:type prefix:ClassREQ2}.
```

Exemple 13

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\MERGE: permet de fusionner les besoins fonctionnels dans la même source

  Select ?instanceREQ# where
            {?instanceREQ# rdf:type req:ClassREQ1}.
            Union
            {?InstancesREQ# rdf:type req:ClassREQ2}.
```

– l'opérateur *Join* : l'opérateur *Join* permet la jointure des besoins ayant des relations entre eux. Cet opérateur se traduit en SPARQL comme suit :

```
Select ?instancesBesoin1 ?instancesBesoin2 where
          {?InstancesBesoin1 rdf:type prefix:ClassREQ.
          ?InstancesBesoin2  rdf:type prefix:ClassREQ.}.
          {?InstancesBesoin1 rdfs:Relation ?InstancesBesoin2}
```

Exemple 14

La jointure des besoins (InstanceREQ) sur la base des relations d'ordonnancement (Requires) entre eux se traduit en SPARQL par :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>

     Select ?instancesREQ1 ?instancesREQ2 where
               {?instancesREQ1 rdf:type req:Requirement.
               ?instancesREQ2  rdf:type req:Requirement.}.
               {?instancesREQ1 req:Require ?instancesREQ2}
```

– l'opérateur *DD* : l'opérateur *DD*, que nous avons défini pour les besoins fonctionnels, permet la détection et la suppression des besoins "instanceBesoin" en doubles appartenant à la classe "ClassBesoin" de l'ESBF. Cet opérateur se traduit en SPARQL par :

```
Select Distinct ?instancesBesoin# where
                {?InstancesBesoin# rdf:type ESBFSpace:ClassREQ.?InstanceBesoin#}
```

Exemple 15

Supprimer les besoins (InstanceREQ) en double appartenant à la classe "Requirement" de l'ESBF se traduit en SPARQL par :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\DD: permet de détecter et de supprimer les besoin
           fonctionnels en double.

Select Distinct ?instanceREQ where
           {?InstanceREQ rdf:type req:Requirement.?InstanceREQ}
```

– l'opérateur *Aggregate* : l'opérateur *Aggregate* permet l'agrégation des besoins sur la base d'une fonction F (F={COUNT, AVG, MIN, MAX, SUM, etc.}), appartenant à la classe "ClassBesoin". Cet opérateur se traduit en SPARQL comme suit :

```
Select (Count(?instancesBesoin) AS?count)
where
     {?InstancesBesoin rdf:type prefix:ClassREQ}
Group By ?InstancesBesoin
```

Exemple 16

Calculer le nombre de besoins (InstanceREQ) de l'entrepôt se traduit en SPARQL par :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\AGREGATE: permet l'agrégation des besoins suivant une fonction F.


Select (Count(?InstancesREQ) AS?count)
        where
             {?InstancesREQ rdf:type req:Requirement}
        Group By ?InstancesREQ
```

– l'opérateur *Filer* : l'opérateur *Filter* permet de filtrer les besoins (instancesBesoin) autorisant uniquement ceux qui vérifient certaines conditions "Cond". Cet opérateur se traduit en SPARQL par la requête :

```
Select ?instancesBesoin# ?P where
                  {?instancesBesoin# rdf:type prefix:ClassREQ
                  ?instancesBesoin# prefix:property ?P
                      FILTER (?P, "Cond")}
```

Exemple 17

Filtrer les besoins (instanceREQ) dans la classe "Requirement" de l'entrepot autorisant seulement les besoins que contiennent l'action Action=create se traduit en SPARQL par :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
\\FILTER: permet de filtrer l'ensemble de besoins entrants selon la propriété P.
Select ?instanceREQ# ?Action where
                   {?InstanceREQ rdf:type req:Requirement
                       ?InstanceREQ req:hasAction ?Action
                   FILTER (?Action, "create")}
```

– l'opérateur *Store* : l'opérateur *Store* permet de stocker (charger) les besoins (instancesBesoin) dans la data staging ou dans l'entrepôt final. Cet opérateur se traduit en SPARQL par la requête :

```
INSERT {prefix:ClassREQ  ?instancesBesoin}
```

Exemple 18

La requête Sparql pour stocker les besoins dans la classe "Requirement" de l'entrepôt se traduit en SPARQL par :

```
PREFIX req:<http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>

INSERT {req:Requirement ?instancesREQ}
```

Par ailleurs, pour implémenter les opérateurs de raisonnement (identification, inférence et vérification de la consistance), nous avons mis en place des déclencheurs (*triggers*) qui implémentent la base de règles de raisonnement que nous avons définie. Nous présentons dans ce qui suit des exemples des triggers appliqués :

– l'opérateur *Identification* : l'exemple suivant illustre le code qui permet d'identifier les relations de conflits entre les besoins :

```
Create or replace TRIGGER REQUIREMENT\_RELATION\_IDENTIFICATION
BEFORE INSERT OR DELETE OR UPDATE ON university\_rdf\_data
BEGIN
   //Création de la base des règles d'identification
   EXECUTE SEM\_APIS.CREATE\_RULEBASE('identif\_rb');

   //Index qui associent la base de règles au modèle de l'ontologie
```

```
   EXECUTE SEM\_APIS.CREATE\_RULES\_INDEX('rdfs\_rix\_university',
   SEM\_Models('university'),SEM\_Rulebases('OWLPRIME','identify\_rb'));

   //Insertion de la règle définie
   INSERT INTO mdsys.semr\_identif\_rb VALUES('rule','(?R1 :ComposedOfTask ?T1) (?R2 :ComposedOfTask ?T2)
   (?T1 :TaskComposedOfAction ?A1) (?T2 :TaskComposedOfAction ?A2) (?A1 :Antonym ?A2)', NULL,
   '(?R1 :conflictwith ?R2)', SEM\_ALIASES(SEM\_ALIAS('',
   'http://www.owl-ontologies.com/OntoReqUnivWordnet1.owl\#')));
   Commit;
   //Génération des relations en exploitant la règle définie
   EXECUTE SEM\_APIS.CREATE\_ENTAILMENT('rdfs\_rix\_university',
   SEM\_MODELS('university'),SEM\_RULEBASES('OWLPRIME','identify\_rb'));
END;
```

– l'opérateur *Inference* : l'exemple suivant illustre un code qui permet d'inférer les relations *Refines* entre les besoins :

```
Create or replace TRIGGER REQUIREMENT\_RELATION\_INFERENCE
BEFORE INSERT OR DELETE OR UPDATE ON university\_rdf\_data
BEGIN
   //Création de la base des règles d'inférence
   EXECUTE SEM\_APIS.CREATE\_RULEBASE('infer\_rb');

   //Index qui associent la base de règles au modèle de l'ontologie
   EXECUTE SEM\_APIS.CREATE\_RULES\_INDEX('rdfs\_rix\_university',
   SEM\_Models('university'),SEM\_Rulebases('OWLPRIME','infer\_rb'));

//Insertion de la règle définie
   INSERT INTO mdsys.semr_infer_rb VALUES('1_ruleS','(?R1 :refines ?R2)
            (?R2 :refines ?R3)', NULL, '(?R1 :refines ?R3)', SEM_ALIASES(
     SEM_ALIAS('','http://www.owl-ontologies.com/OntoReqUnivWordnet1.owl\#')));
   Commit;
   //Génération des relations en exploitant la règle définie
   EXECUTE SEM\_APIS.CREATE\_ENTAILMENT('rdfs\_rix\_university',
   SEM\_MODELS('university'),SEM\_RULEBASES('OWLPRIME','infer\_rb'));
END;
```

– l'opérateur *CheckConsistency*: l'exemple suivant illustre un code qui permet de vérifier la consistance des relations entre les besoins :

```
Create or replace TRIGGER REQUIREMENT\_RELATION\_CHCKCONSISTENCE
BEFORE INSERT OR DELETE OR UPDATE ON university\_rdf\_data
BEGIN
   //Création de la base des règles de vérification de la consistance

   EXECUTE SEM\_APIS.CREATE\_RULEBASE('consist\_rb');

   //Index qui associent la base de règles au modèle de l'ontologie
   EXECUTE SEM\_APIS.CREATE\_RULES\_INDEX('rdfs\_rix\_university',
   SEM\_Models('university'),SEM\_Rulebases('OWLPRIME','consist\_rb'));

    //Insertion de la règle définie
   INSERT INTO mdsys.semr_consist_rb VALUES('1_ruleSC','(?R1 :equal ?R2)
               (?R1 :contain ?R2)', NULL, '(?R1 :inconsistencywith ?R2)', SEM_ALIASES(
               SEM_ALIAS('','http://www.owl-ontologies.com/OntoReqUnivWordnet1.owl\#')));
   Commit;
   //Génération des relations en exploitant la règle définie
   EXECUTE SEM\_APIS.CREATE\_ENTAILMENT('rdfs\_rix\_university',
   SEM\_MODELS('university'),SEM\_RULEBASES('OWLPRIME','consist\_rb'));
END;
```

En suivant les mêmes étapes, nous avons réalisé la phase de conception physique en utilisant la BDBO OntoDB comme expliqué ci-dessous.

3.2 Déploiement de l'entrepôt sur une BDBO OntoDB

Afin d'évaluer la faisabilité de la conception physique, nous avons déployé notre entrepôt sur la BDBO OntoDB qui suit une architecture de *type III*. Ce type d'architecture comporte quatre principales parties distinctes : le méta-schéma ontologique, le schéma ontologique, le méta-schéma des données et le schéma des données. La nouvelle BDBO OntoDB que nous avons créée implémente le schéma de l'ontologie conceptuelle. L'articulation entre l'ontologie conceptuelle et l'ontologie linguistique (qui est externe à OntoDB) reste valide et se fait via des labels. Nous avons commencé par étendre le méta-schéma ontologique par le modèle pivot des besoins que nous avons défini. Cette extension est illustrée en figure 7.6. Le schéma de l'entrepôt est ensuite traduit en un schéma relationnel horizontal (schéma logique supporté par OntoDB). L'extension du méta-schéma d'OntoDB avec le schéma de l'entrepôt (Modèle pivot) se réalise par la création d'une table Entity pour chaque classe du schéma pivot. OntoDB fournit son

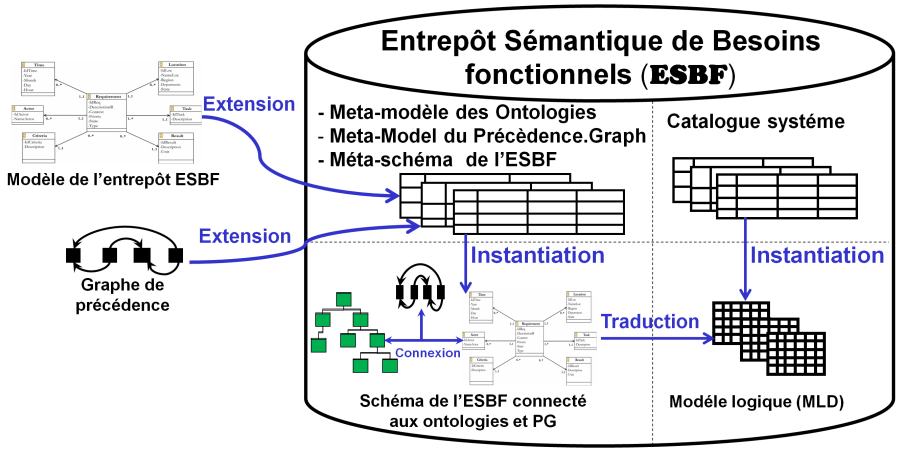

FIGURE 7.6 – La BDBO OntoDB étendue par le schéma pivot de l'entrepôt

propre langage de définition et de manipulation des Ontologies [81] qui est le langage OntoQL. Nous avons ainsi commencé par définir des requêtes OntoQL pour l'extension du méta-schéma d'OntoDB. Voici la syntaxe de la requête qui permet la création des entités :

```
CREATE Entity #[nom de l'entité] ([nom d'attribut, type d'attribut (Property)])
```

Dans ce qui suit, nous présentons des exemples de quelques requêtes OntoQL définies afin d'étendre le méta-schéma d'OntoDB avec les entités du schéma pivot de l'entrepôt :

```
Create Entity  #Actor(#its_properties REF (#Property))
```

```
Create Entity  #Task(#its_properties REF (#Property) Array)
Create Entity  #Criteria (#its_properties REF (#Property) Array)
Create Entity  #Result(#its_properties REF (#Property) Array)
Create Entity  #Location(#its_properties REF (#Property))
Create Entity  #Time(#its_properties REF (#Property))
Create Entity #Requirement(#Property (#Requirement_ID Int, #Requirement_Name String, #Description String,
#Purpose_Req String, #Context_Req String, #Priority_Req String, #Status String, #Type_Req String),
#ResponsibleOf REF(#Actor), #HasTask REF (#Task),  #HasCriteria REF (#Criteria),
#HasLocation REF (#Location),
#ExpressedIn REF (#Time), #Achieves REF (#Result))
```

Le chargement des instances ontologiques dans ce schéma étendu s'est fait via le processus ETL. Cette implémentation a nécessité la traduction des opérateurs ETL en requêtes OntoQL de la même façon que pour la BDBO Oracle.

L'extraction des besoins à partir de l'entrepôt s'effectue avec des requêtes d'interrogation OntoQL dont la syntaxe générale est la suivante [81] :
⟨*query specification*⟩ ::= ⟨*select clause*⟩⟨*from clause*⟩[⟨*where clause*⟩]

[⟨*group by clause*⟩][⟨*having clause*⟩][⟨*order by clause*⟩]

[⟨*namespace clause*⟩][⟨*language clause*⟩]

Nous remarquons que la structure d'une règle OntoQL ressemble à la structure d'une règle SQL. La règle est composée d'un ensemble d'attributs appartenant à l'un des types suivants : primitif, collection, référence ou une classe ontologique.

Par exemple, la requête suivante permet l'extraction des besoins provenant d'une source située géographiquement en *France*:

```
Select g.#Description_Req  from #Requirement g, Lacation L where L.#State="French";
```

Nous décrivons dans la section suivante notre approche d'analyse multidimensionnelle de l'entrepôt déployé, en définissant et en utilisant un cube sur les besoins entreposés. Cette étape d'analyse constitue la dernière phase du cycle de vie d'un projet d'entreposage.

4 Exploration et analyse des besoins entreposés

Notre approche couvre également la phase d'exploitation des besoins qui permet aux décideurs de faire diverses analyses sur les besoins entreposés.

La première solution est d'effectuer des analyses via des requêtes Sparql. Plus précisément, nous avons utilisé des requêtes SPARQL *endpoints* [69]. SPARQL *endpoints* est un service du protocole SPARQL qui permet aux utilisateurs d'interroger une base de connaissances via le langage SPARQL. SPARQL *endpoints* est conçu avec une interface conviviale vers une base de connaissances. Nous avons défini des requêtes de type agrégation.

69. https://www.w3.org/wiki/SparqlEndpoints

Exemple 19

L'exemple suivant montre une requête d'agrégation qui permet de calculer le nombre de besoins que contient l'entrepôt :

```
CONSTRUCT {
    ?req dw:ReqCount ?ReqCount. ?req dw:Requirement ?IDReq.
    ?req dw:Task ?task. ?req dw:Actor ?IDActor. ?req dw:Location ?LocId.}
WHERE {
    ?req rdf:type s:Requirement . ?req s:Requirement ?IDReq .
    ?req s:Task ?task . ?req s:Actor ?IDActor . ?req s:Location ?LocId .}
```

Exemple 20

Le second exemple ci-dessous permet au décideur de calculer le nombre de besoins exprimés par un concepteur situé géographiquement a PARIS (France) et de les regrouper de façon ordonnée par période. La requête SPARQL correspondant à ce besoin est la suivante :

```
PREFIX qb: <http://purl.org/linked-data/cube\#>
PREFIX req: <http://req.org/schemas/>
SELECT  ?req ?loc ?time (COUNT(?REQ) AS ?requirementcount)
WHERE {
        { ?li a qb:Observation;
                    qb:dataSet <http://req.org/data/dataset-aggview2>;
                        req:Location ?loc ; req:Time ?time ; req:revenue ?REQ.}
        FILTER(?Loc \= 'PARIS')}
GROUP BY ?loc ?time
ORDER BY ?time DESC(?requirementcount)
```

Pour l'exploitation de notre entrepôt de besoins, nous avons aussi recours à une solution alternative ne nécessitant pas une maitrise de la technologie d'entreposage. Cette solution consiste à utiliser QB4OLAP [70] qui est un vocabulaire supportant les analyses OLAP. Ce langage est basé sur le Data cube vocabulary (QB) qui est le vocabulaire fourni par le W3C pour la publication des données multidimensionnelles en utilisant le standard RDF.

Le vocabulaire utilisé dans QB4OLAP étend le vocabulaire de QB et offre la possibilité de définir les cubes OLAP dans RDF. Il offre également la possibilité de mettre en oeuvre les opérateurs OLAP (par exemple : *Roll-up, Slice, and Dice*) comme des requêtes SPARQL directement applicables sur des représentations RDF.

Dans ce qui suit, nous présentons quelques requêtes QB4OLAP qui permettent la définition du cube multidimensionnel de besoins.

```
prefix req: <http://www.owl-ontologies.com/OntoReqUnivWordnet.owl\#>
PREFIX  qb:  <http://purl.org/linked-data/cube\#>
PREFIX  qb4o:  <http://purl.org/olap\#>

//Definir le cube des besoins et ses dimensions
req:RequirementCube  a  qb:DataStructureDefinition;
rdfs:label  "Requirement" ;
```

70. https://www.w3.org/TR/vocab-data-cube-use-cases/

```
rdf:type  req:Requirement.

   //Dimensions
    qb:component  qb:dimension  req:Location.
    qb:component  qb:dimension  req:Time.
    qb:component  qb:dimension  req:Actor.
    qb:component  qb:dimension  req:Task.
    qb:component  qb:dimension  req:Criterion.
    qb:component  qb:dimension  req:Result.
    qb:component  qb:measure  req:context.

   //Définition  des  mesures
    qb:component qb:measure req:RquirementCount; qb4o:AggregateFuction
    qb4o:Count; rdf:predicate req:IDreq;

    qb:component qb:measure req:ResultReqAvg; qb4o:AggregateFuction
    qb4o:avg ; rdf:predicate req:IdResult;

  //Définition des attributs
    qb:component  qb:attribute  req:DescriptionReq.
    qb:component  qb:attribute  req:PriorityReq.
    qb:component  qb:attribute  req:Status.
    qb:component  qb:attribute  req:TypeR.
    qb:component  qb:attribute  req:context.
```

Les données obtenues dans une structure QB4OLAP reçoivent des faits qui peuvent être stockés sous la forme des instances *qb:Observation* (dans la terminologie OLAP cela correspond à des faits indexés par des dimensions). Toutes les instances de dimension sont stockées sous forme de triplets. Les valeurs agrégées pour les mesures sont calculées sur la base des types de fonctions *qb4o:AggregateFunction* (une fonction d'agrégation peut être AVG, COUNT, MAX, MIN, SUM). Chaque instance du *fait* représente un point dans l'espace multidimensionnel formé par les dimensions. Pour chaque point, un ensemble de *valeurs de mesure* est enregistré.

Par exemple, la requête QB4OLAP correspondant au besoin cité précédemment (qui permet au décideur de calculer le nombre de besoins exprimés par un concepteur situé géographiquement a Paris (France) et de les regrouper de façon ordonnée par période) :

```
<http://www.owl-ontologies.com/OntoReqUnivWordnet1.owl\#R1>  a  qb:Observation;
qb:dataSet  req:RequirementDataWarehouse;
req:RequirementCount  36;
req:Country  <http://www.owl-ontologies.com/OntoReqUnivWordnet1.owl\#21> .
req:CountryName  "Paris".
```

5 Evaluation : ETL et raisonnement

Notre évaluation concerne les performances du processus ETL ainsi que le mécanisme de raisonnement (partie importante du processus ETL). Nous commençons par rappeler l'environnement d'implémentation utilisé. Nous effectuons cette évaluation pour les deux scénarios étudiés (ontologies multiples et ontologie partagée)

5.1 Scénario 1 : ontologie partagée

Nous considérons à nouveau l'étude de cas décrite dans le chapitre précédent (section 4). Nous utilisons cette fois le SGBD Oracle pour le stockage des besoins sources et des besoins de l'entrepôt.

5.1.1 Les besoins sources

Comme pour la première série de tests, nous avons généré trois ensembles de besoins définis dans trois formalismes (le modèle de but, le modèle de cas d'utilisation et le modèle de traitement de la méthode Merise), contenant respectivement 40, 30 et 40 besoins. Ces besoins sont générés en utilisant le benchmark ontologique universitaire LUBM [71] et le système de gestion des cours (CMS). Nous avons *EuroWordNet* comme une ontologie linguistique.

Chaque ensemble de besoins contient différents tâches, critères et résultats. Le nombre d'instances (en format *N-Triple*) utilisées est indiqué dans le Tableau 5.1.1. Le SGBD Oracle est utilisé pour déployer les besoins sources et cibles (au niveau entrepôt).

Pour les sources, nous avons créé trois BDBO Oracle basées sur les trois formalismes (orienté but, use case d'UML et de traitement de la méthode MERISE) pour stocker les trois ensembles de besoins générés. Le besoins sont stockés en étendant le méta-modèle ontologique d'Oracle par le modèle de besoin.

Oracle propose différents formats pour le chargement de données tels que : RDF/XML, N-TRIPLES, N-QUADS, TriG et Turtle. Nous avons choisi le format N-Triple(.nt) qui correspond à une représentation verticale (à trois colonnes) pour charger les instances en utilisant l'outil *Oracle Bulk*.

Source	Formalisme utilisé	nombre d'instances
Ensemble de besoins$_1$	Goal Formalisme	14 432
Ensemble de besoins$_2$	Merise Formalisme	14 315
Ensemble de besoins$_3$	UML Formalisme	14 371

TABLE 7.1 – Caractéristiques des besoins utilisés

5.1.2 Moteur d'inférence

Oracle intègre un moteur d'inférence défini sur la base des raisonneurs *TrOWL* [72] et *Pellet* [73]. Oracle fournit un support complet pour l'inférence native dans les bases de données en ce

71. http://swat.cse.lehigh.edu/projects/lubm/
72. http://trowl.org/
73. https://www.w3.org/2001/sw/wiki/Pellet

qui concerne : RDF, RDFS, RDFS++, OWLPRIME, OWL2RL, etc. Oracle compile les règles d'inférence directement vers SQL et utilise *Oracle's native cost-based SQL optimizer* pour choisir un plan d'exécution efficace pour chaque règle.

L'ensemble des règles définies ont été implémentées et des exemples du code de chaque type de règle ont été fournis dans la section précédente.

5.1.3 Machine physique:

Nos évaluations ont été effectuées sur un ordinateur portable (HP Elite-Book 840 G2) avec un Intel(R) CoreTM i5-5200U CPU 2.20 GHZ et 8 GO de RAM et un disque dur de 500 GB. Nous avons utilisé le système d'exploitation Windows 10 de 64 bits. Nous avons aussi utilisé le SGBD *Oracle 12c release 1*.

Nous présentons dans ce qui suit les tests effectués.

5.1.4 Performances de l'ETL et du raisonnement

Oracle offre des mécanismes de d'inférence sur plusieurs types de modèles. Nous évaluons le mécanisme d'inférence en utilisant les fragments OWL-Prime, RDF, RDFS et les règles de raisonnement que nous avons définies pour : (i) identifier les relations complexes entre les besoins, (ii) inférer des nouvelles relations sur la base des relations existantes et (iii) vérifier la consistance et la cohérence des besoins.

Nous avons testé les règles de raisonnement sur les trois fragments d'Oracle cités précédemment. Le tableau 5.1.4 montre les résultats obtenus. Le tableau représente :

- le nombre d'instances inférées en utilisant les différents fragments et les règles de raisonnement.
- la colonne *Entrepôt de besoins* représente le nombre d'instances (besoins) cumulées dans l'entrepôt, i.e. les instances chargées et les instances inférées.
- le temps de chargement des instances des besoins.

Ce tableau montre le nombre important de relations complexes inférées par l'utilisation des règles définies. Le temps de chargement (basé sur le processus ETL) reste raisonnable même avec l'utilisation du mécanisme de raisonnement.

La figure 7.7 montre le nombre d'instances inférées et la performance en temps du raisonnement pour chaque fragment d'Oracle, d'abord sans l'utilisation des règles de raisonnement, puis avec l'utilisation des règles de raisonnement définies. Nous remarquons que le fragment RDFS est le plus performant en termes d'inférence.

La figure 7.8 indique le nombre de relations identifiées et inférées pour chaque type de relations entre les besoins (refine, contain, require, equal, conflictWith). Les résultats de l'expérimentation montrent que les deux étapes du mécanisme de raisonnement (identification des relations directes et l'inférence des relations indirectes entre les besoins) sont complémentaires

Critères	instances inférées	Entrepôt de besoins	Temps (Millisecondes)
RDF fragment	47	43 118	3750
RDFS fragment	18 583	61 701	8620
OWLprime fragment	400	62 101	5110
Règles d'identification	1019	63120	4400
Règles de Consistance	312	63432	2280
Règles d'inférence	3062	66494	8900

TABLE 7.2 – Performance de l'inférence : temps et nombre d'instances

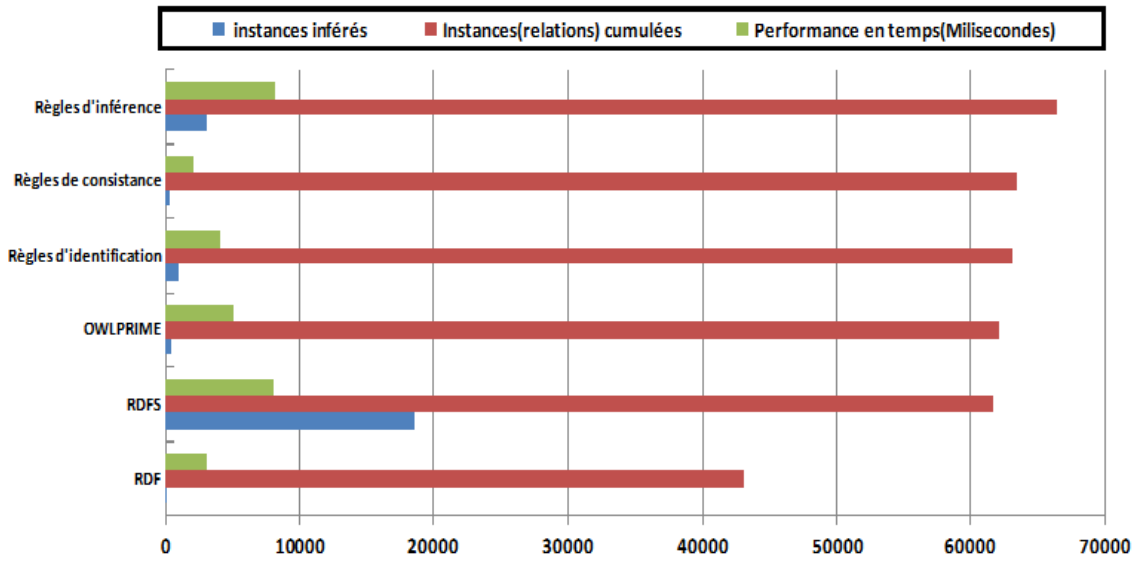

FIGURE 7.7 – Le temps d'exécution et le nombre de relations inférées

pour tous les types de relations entre les besoins.

5.2 Scénario 2 : ontologies multiples

Nous utilisons le même scénario que celui décrit au chapitre 4, section 5.2, qui concerne un système de gestion de conférence. L'ontologie intégrante a déjà été présentée. Nous procédons directement à l'évaluation du processus ETL et du mécanisme de raisonnement.

5.2.1 Performance de l'ETL

Comme le montre le tableau 7.3, nous avons évalué notre système de raisonnement en fonction de plusieurs étapes. A chaque étape, nous avons changé : (i) le nombre de besoins ($i * 10$), (ii) la taille de l'ontologie conceptuelle (nombre de concepts), et (iii) la taille de l'ontologie

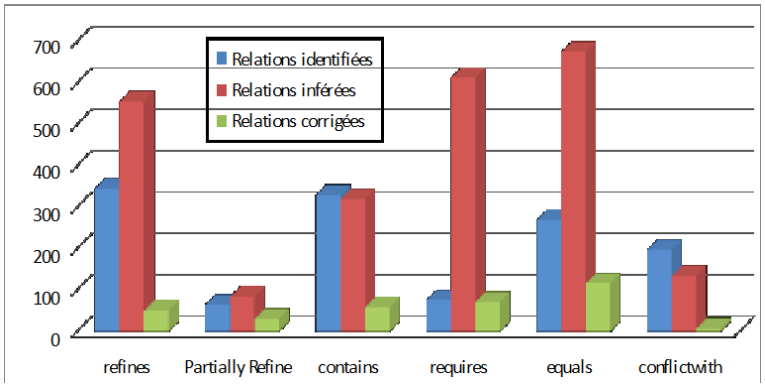

FIGURE 7.8 – Les relations identifiées et inférées dans l'entrepôt

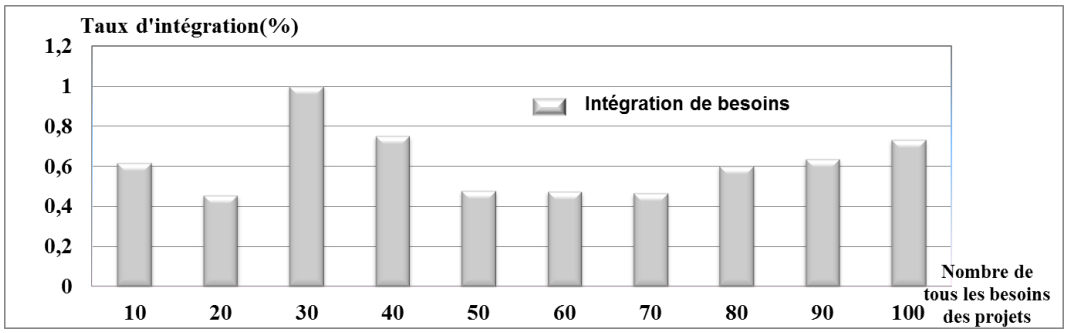

FIGURE 7.9 – Taux d'intégration des besoins

linguistique (nombre de termes).

Étape	1	2	3	4	5	6	7	8	9	10
Nbr de Termes	*7*	*14*	*21*	*30*	*37*	*45*	*50*	*55*	*59*	*63*
Nbr de Concepts	*12*	*24*	*32*	*38*	*45*	*53*	*59*	*68*	*74*	*79*
Nbr de besoins	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*

TABLE 7.3 – Composants de chaque étape de l'évaluation

La figure 7.9 montre le taux d'intégration des besoins. Pour ce scénario, nous nous sommes concentrés sur le taux de besoins intégrés car il dépend des mappings ontologiques identifiés lors du matching. Ce taux diffère dans les étapes de l'expérimentation car le processus ETL dépend des termes et concepts utilisés par les besoins et donc du résultat du matching ontologique. Certains mappings ontologiques n'ayant pas été identifiés, nous avons choisi de ne pas intégrer les besoins utilisant ces concepts rejetés par le matching.

5.2.2 Performance du mécanisme de raisonnement

L'évaluation de l'efficacité du mécanisme de raisonnement est réalisée sur les aspects suivants :

– (i) le taux de détection des relations complexes entre les besoins (*Requires,Refines, etc.*) ;
– (ii) le temps d'exécution et l'évolutivité (scalabilité) du processus de raisonnement ;

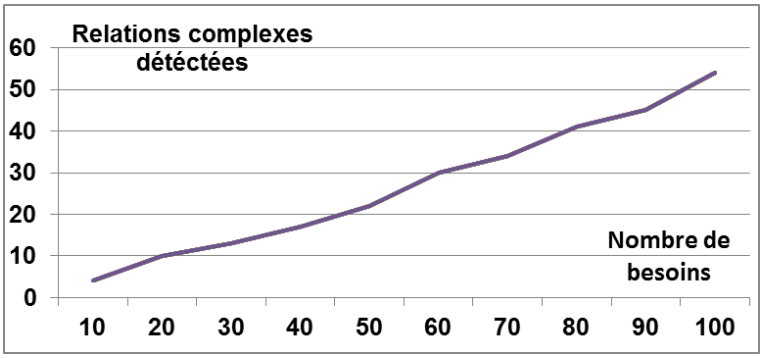

FIGURE 7.10 – Relations détectées

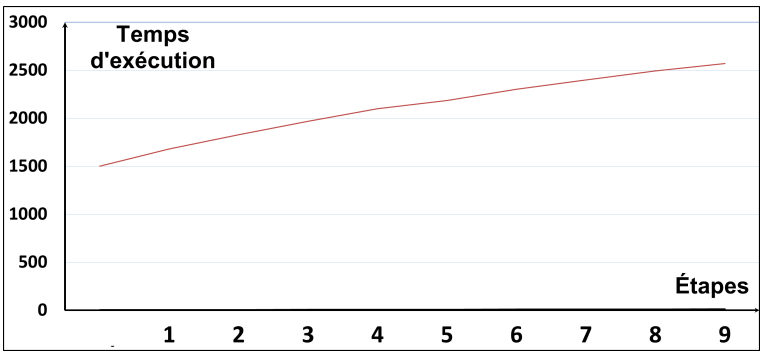

FIGURE 7.11 – Scalabilité et temps d'exécution

La figure 7.10 montre que le mécanisme de raisonnement détecte une quantité importante des relations complexes entre les besoins par rapport à une approche conventionnelle d'analyse des besoins. Ces résultats, qui diffère du premier scénario (ontologie partagée) s'expliquent par le fait que la détection des relations complexes entre les besoins par le mécanisme de raisonnement dépend du nombre de concepts découverts par le système de matching.

Les aspects d'évolutivité (*scalability*) et le temps d'exécution sont illustrés dans la figure 7.11. Il montre que le processus est réalisé dans un délai raisonnable pour l'ensemble des besoins et peu s'échelonner suivant une échelle logarithmique.

6 Présentation de l'outil d'unification des besoins

Afin de démontrer la faisabilité de notre approche, nous avons développé un outil Case implémentant toutes les étapes proposées dans notre approche. L'outil est développé en langage JAVA. Cet outil englobe les principales contributions présentées dans les chapitres précédents. L'outil est destiné au concepteurs souhaitant intégrer plusieurs sources de besoins hétérogènes dans un entrepôt.

Suivant notre approche, l'outil offre une grande autonomie aux concepteurs, et donne à chaque source la possibilité d'exprimer ses besoins localement sur la base de son propre formalisme en utilisant des ontologies conceptuelle et linguistique locales. L'outil supporte les deux scénarios d'ontologie partagée et d'ontologies multiples. Pour le premier scénario, les mappings sont définis directement par le concepteur. Pour le second scénario, l'outil implémente le processus de matching ontologique que nous avons utilisé. Afin de ne pas restreindre les concepteurs aux trois langages de modélisation des besoins cités précédemment, l'outil permet l'introduction de règles de transformation entre les formalismes sources et le formalisme cible.

Dans ce qui suit, nous présentons l'architecture fonctionnelle de l'outil ainsi que ses principaux modules.

6.1 Architecture fonctionnelle de l'outil

L'architecture fonctionnelle de l'outil proposé est présentée dans la figure 7.12. Cette figure illustre l'étude de cas proposée pour le scénario d'ontologies multiples, car il nécessite un processus de matching et donc englobe le premier scénario. L'outil est composé de cinq modules, que nous présentons dans ce qui suit :

- **Chargement et visualisation des sources** : ce module permet le chargement des sources. Il contient plusieurs composants :
  - *chargement des besoins sources en langage naturel* : ce composant est responsable du chargement et de la visualisation des sources de besoins représentées en langage naturel.
  - *chargement des besoins sources en formalisme local* : ce composant est responsable du chargement et de la visualisation des besoins formalisés selon les trois langages de modélisation (MCT Model, UML Model et Goal Model). Pour ce composant de visualisation, seul ces trois formalismes sont supportés. Les règles de correspondance peuvent cependant être définies sans passer par ce composant s'il s'agit de nouveaux langages de modélisation des besoins.
  - *chargement des ontologies conceptuelles sources (locales) utilisées pour l'expression des besoins* : ce composant est responsable du chargement et de la visualisation des ontologies conceptuelles locales (par exemple : les ontologies EDAS, CMT et ConfOf).
  - *chargement des ontologies linguistiques sources (locales) utilisées pour l'expression*

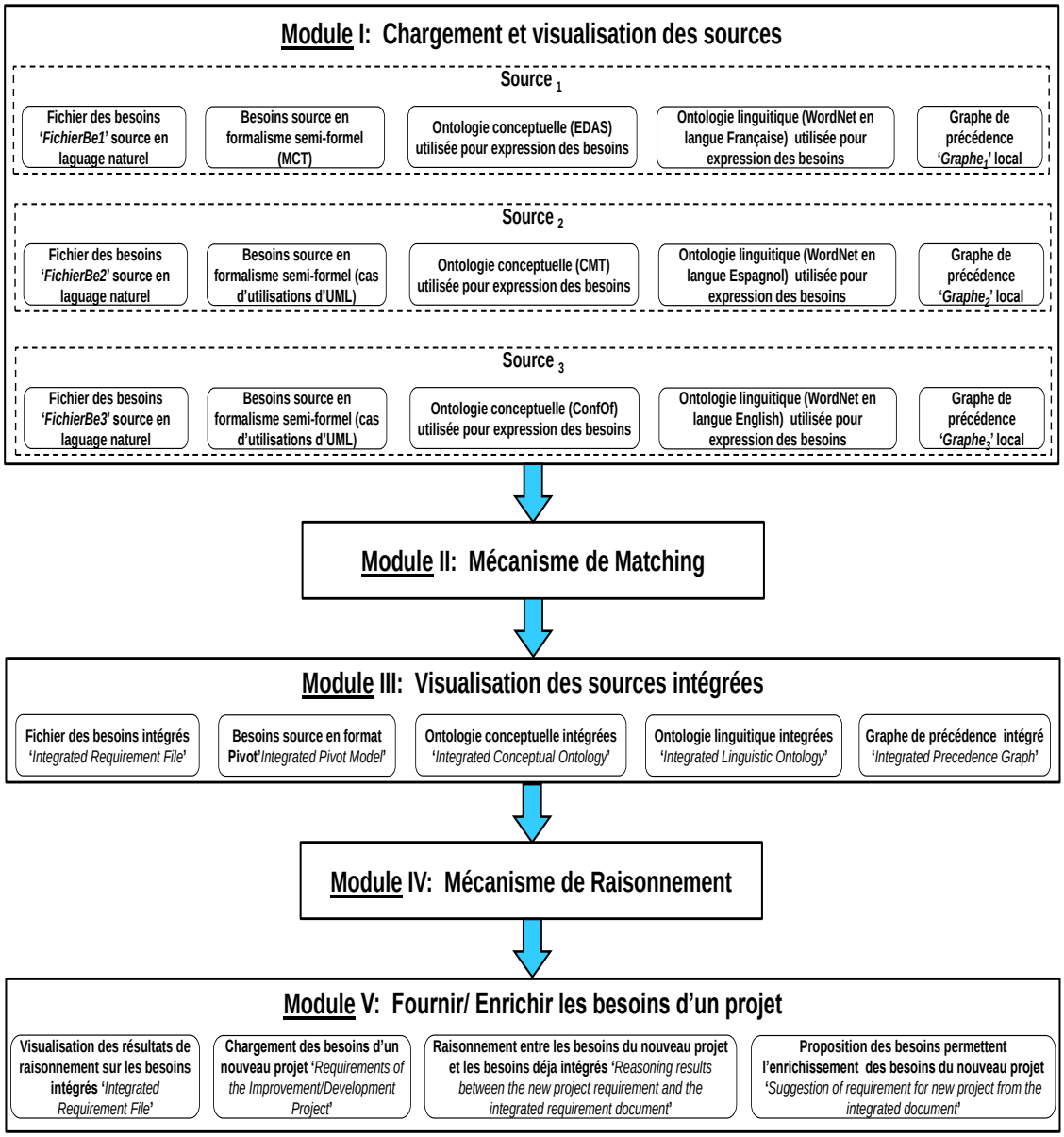

FIGURE 7.12 – Architecture fonctionnelle de l'outil

des besoins : ce composant est responsable du chargement et de la visualisation des ontologies linguistiques locales (par exemple, les ontologies *WordNet* en langue française, *WordNet* en langue espagnole et *WordNet* en langue anglaise).

- **mécanisme de matching** : ce module permet l'exécution de l'algorithme de matching sur l'ensemble des ontologies locales en entrées.
- **visualisation des sources intégrées** : ce module permet la visualisation des résultats du mécanisme de matching qui est l'ontologie intégrante, ainsi que les règles de correspondance entre les formalismes sources et cible de l'entrepôt.
- **ETL et mécanisme de raisonnement** : ce module permet de réaliser le processus ETL ainsi que le raisonnement sur les besoins, afin de détecter des relations complexes entre eux et vérifier la cohérence des relations détectées. Le mécanisme est basé sur l'ensemble des règles de raisonnement que nous avons définies.
- **exploiter des besoins d'un projet** : ce module permet de visualiser les besoins intégrés.

6.2 Fonctionnement de l'outil

Dans ce qui suit, nous présentons les détails des différents modules présentés :

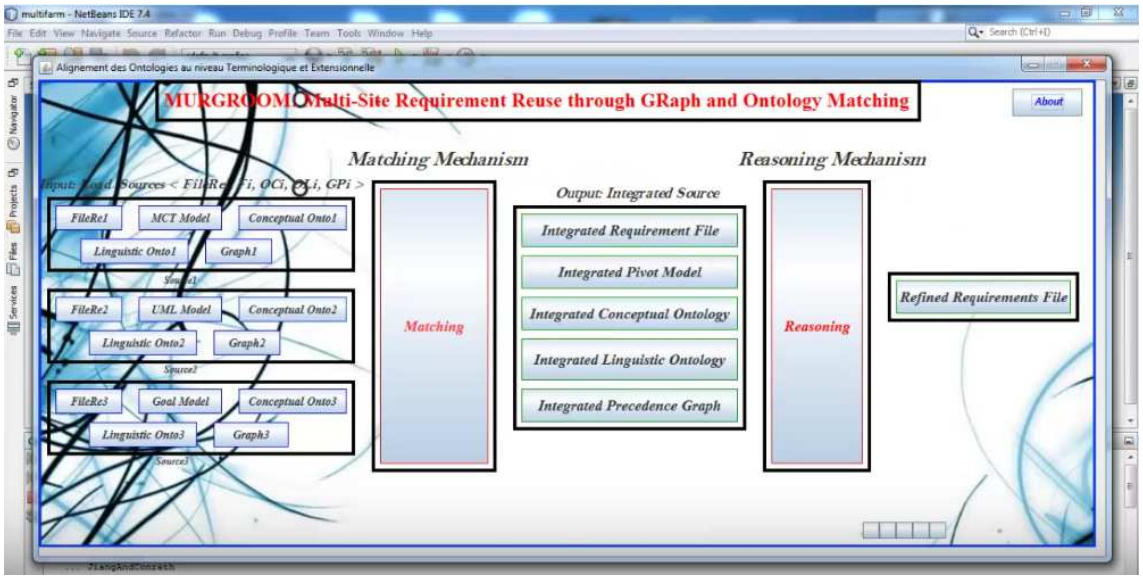

FIGURE 7.13 – Menu principal de notre outil

6.2.1 Chargement et visualisation des entrées de notre système

Avec ce composant, le concepteur peut charger et visualiser les partenaires sources (sites). Le choix de chaque partenaire source (site) est représenté par les cinq entrées (voir sur la figure 7.12 la partie 1), qui sont détaillées dans ce qui suit :

Chargement et visualisation des besoins sources en langage naturel : la figure 7.14 illustre le résultat de ce composant qui permet d'afficher les besoins sources en langage naturel. Ce composant est utile pour le concepteur global qui peut visualiser le document de besoins de chaque source locale. Ce composant peut également être utile dans le cas où l'outil serait étendu pour analyser le document de besoins par des techniques de traitement automatique du langage naturel afin de les faire correspondre avec l'ontologie linguistique/conceptuelle.

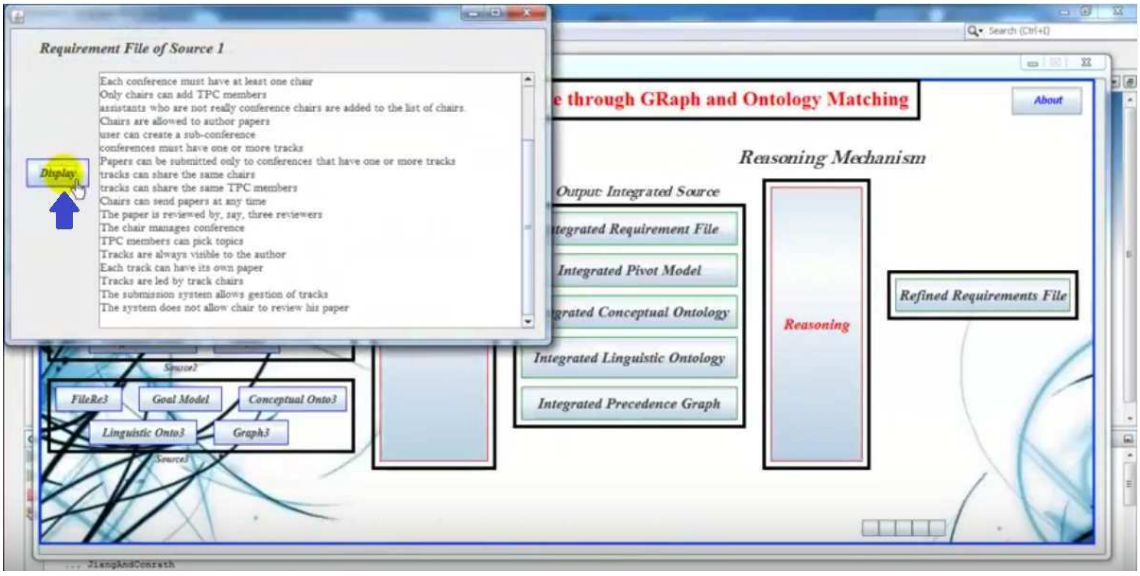

FIGURE 7.14 – Visualisation des besoins d'une source en langage naturel

Chargement et visualisation des besoins exprimés selon le formalisme source : ce composant permet le chargement et la visualisation des besoins des sources à intégrer selon les éléments des formalismes sources (*modèle MCT, cas d'utilisation UML* et *modèle d buts*). La figure 7.15 illustre l'interface qui permet cette visualisation.

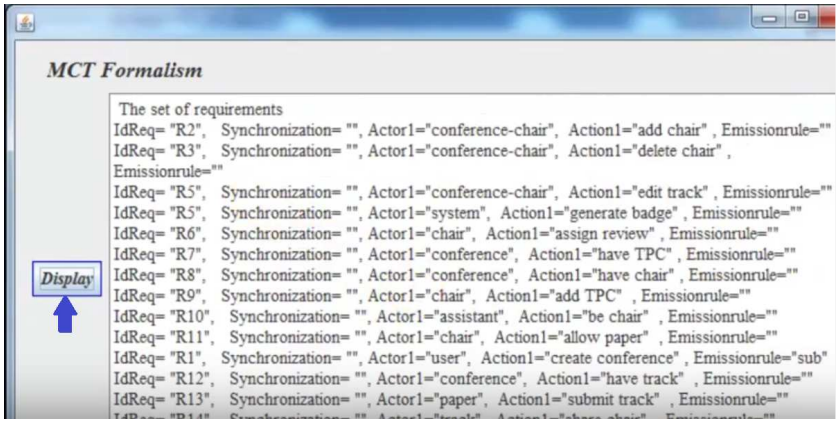

FIGURE 7.15 – Visualisation des besoins d'une source exprimés en formalisme local (MCT)

Chargement et visualisation des ontologies conceptuelles sources : la figure 7.16 illustre

le chargement d'une ontologie conceptuelle locale. Ceci se fait par le chargement du fichier OWL de l'ontologie. La visualisation de cette ontologie se fait sous une forme arborescente dans l'outil.

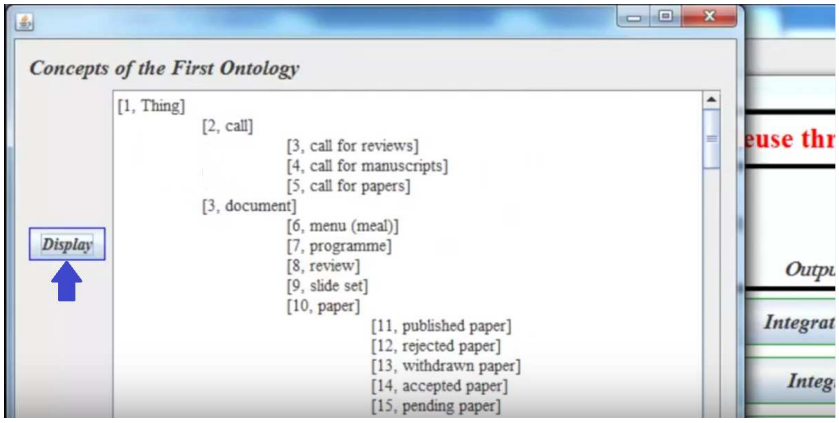

FIGURE 7.16 – Visualisation d'une ontologie conceptuelle d'une source

Chargement des ontologies linguistiques sources : la figure 7.17 illustre le chargement d'une ontologie linguistique locale. La visualisation de l'ontologie permet de vérifier les relations linguistique entre les termes de l'ontologie.

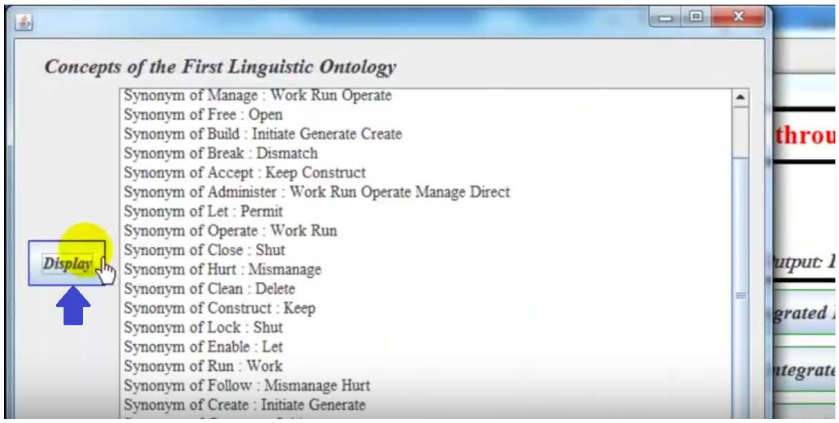

FIGURE 7.17 – Visualisation d'ontologie linguistique d'une source

6.2.2 Mécanisme de matching

Ce module contient deux principaux composants : (1) exécution du processus de matching des ontologies sources et (2) visualisation des résultats de matching. Dans ce qui suit, nous présentons chaque composant.

6.2.2.1 Exécution du processus de matching : l'outil permet au concepteur d'exécuter l'algorithme de matching après le chargement des ontologies sources. En sortie, l'outil établit les

mappings entre les ontologies locales et construit l'ontologie intégrante au format OWL.

6.2.2.2 Visualisation des résultats du processus de matching : notre outil permet aussi au concepteur de visualiser les résultats du processus de matching. La figure 7.18 illustre l'ontologie intégrante conceptuelle obtenue. La figure 7.19 illustre l'ontologie intégrante linguistique obtenue.

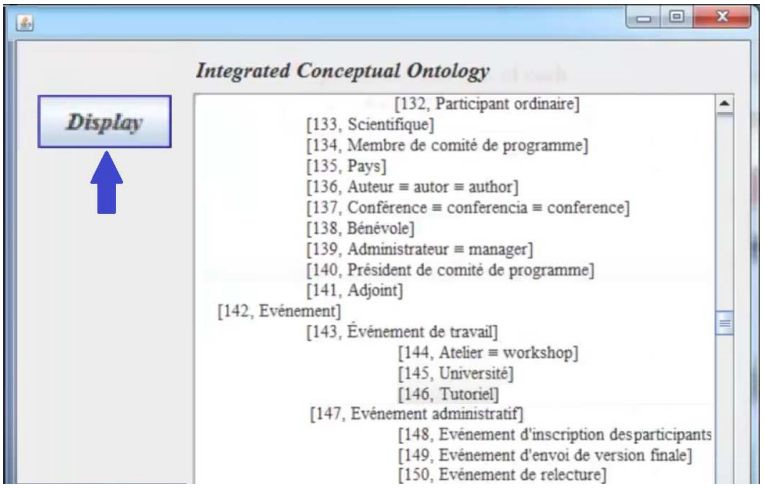

FIGURE 7.18 – Visualisation de l'ontologie conceptuelle globale intégrée

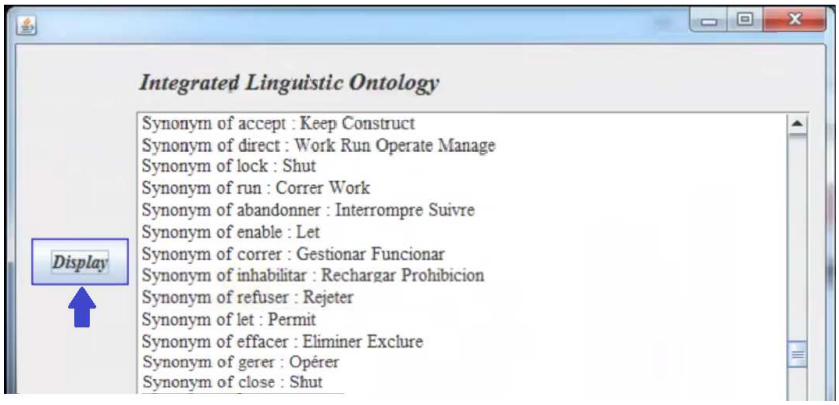

FIGURE 7.19 – Visualisation de l'ontologie linguistique globale intégrée

6.2.3 Processus ETL

Notre outil fournit la possibilité de définir les mappings entre les schémas sources et cible, d'exécuter le processus ETL et de visualiser le résultat du processus ETL qui sont les besoins intégrés, modélisés selon le format cible proposé pour l'entrepôt, et utilisant les termes et concepts de l'ontologie intégrante. La figure 7.20 illustre ce résultat.

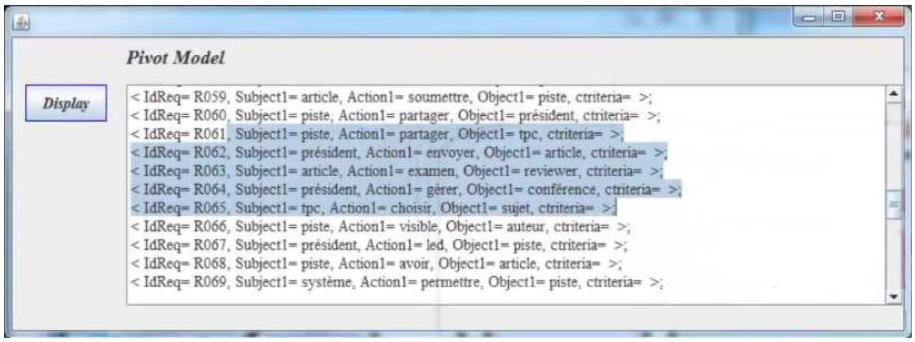

FIGURE 7.20 – Visualisation des besoins intégrés en format cible

Ce composant fournit en sortie un fichier OWL contenant l'ensemble des besoins unifiés et intégrés. L'intégration se fait donc au niveau ontologique. Cette ontologie peut être chargée dans un SGBD sémantique choisi par le concepteur. Cette dernière étape ne se fait pas via l'outil, car chaque SGBD possède son propre mécanisme de chargement des ontologies.

6.2.4 Mécanisme de raisonnement

FIGURE 7.21 – Visualisation du résultat du processus de raisonnement

Ce composant permet le raisonnement sur l'ensemble des besoins afin de détecter les relations complexes entre eux, et de vérifier leurs cohérences. Ce mécanisme est basé sur l'ensemble de règles de raisonnement (identification, inférence et vérification de la cohérence) formalisées en utilisant le langage SWRL. La figure 7.21 montre les relations identifiées/inférées suite à l'exécution du processus de raisonnement sur les besoins.

7 Conclusion

Nous avons proposé dans ce chapitre de déployer l'entrepôt sur un SGBD donné afin de procéder à l'analyse multidimensionnelle des besoins unifiés. Le déploiement comporte le choix du SGBD, la conception logique du schéma de l'entrepôt et sa conception physique. Nous avons pour deux SGBD sémantiques utilisant des modèles logiques/physiques différents : le SGBD Oracle et le SGBD académique OntoDB. L'analyse des besoins intégrés se fait par la définition de requêtes d'agrégation puis par la définition d'un cube multidimensionnel. Divers exemples d'implémentation sur les SGBD choisis sont fournis dans ce chapitre. Afin d'illustrer la faisabilité de notre approche globale, depuis la construction de l'ontologie intégrante jusqu'à l'intégration des instances des besoins, nous avons proposé un outil implémentant les principales contributions de l'approche. Nous concluons notre étude dans le prochain chapitre et nous fournissons quelques perspectives qui restent à explorer.

Chapitre 8

Conclusion et perspectives

Sommaire

1 Conclusion & Perspectives 195
1 Les besoins du CMS document 217
2 Les besoins du systèmes de gestion des conférences: 225

1 Conclusion & Perspectives

Dans ce chapitre, nous présentons une synthèse générale qui nous permet de résumer nos contributions, et de discuter quelques perspectives ayant émergé de notre étude.

1.1 Synthèse du travail

Nous avons abordé ce travail de thèse en identifiant l'intérêt des dépôts de besoins pour la gestion, la persistance et l'exploitation des besoins fonctionnels. Nous avons classifié ces dépôts en trois principaux types. Le premier type de dépôt se fait à l'instar des données et permet des traitements transactionnels sur les besoins. Le deuxième type intègre la dimension décisionnelle OLAP sur les besoins.

Dans un environnement de développement intense de systèmes destinés pour les grandes entreprises, il devient nécessaire de faire collaborer un nombre important de partenaires, qui peuvent être situés dans les différents coins du monde. L'intégration des sources de besoins de ces différents partenaires a été identifiée comme un enjeu important dans la littérature. La gestion de l'intégration de ces sources doit faire face à de nombreux types d'hétérogénéité et de conflits syntaxiques, sémantiques, de langages, etc. Le troisième type de dépôt répond à cette préoccupation et concerne les dépôts *intégrés* de besoins exploités avec des outils transactionnels. Nous avons remarqué le manque d'un quatrième type de dépôt, qui constitue une évolution naturelle des trois types existants, et qui fournit un dépôt d'intégration des besoins et incluant la dimension décisionnelle. Nous avons proposé dans ce travail d'élever les solutions d'entreposage conventionnels, orientés données, vers une solution dédié à l'entreposage des besoins. Nous avons suivi une approche de conception ontologique, qui a fait ses preuves dans les entrepôts conventionnels.

Ce travail a nécessité un grand effort de réadaptation des solutions d'entreposage, suivant une approche holistique, concernant les aspects suivants : la conception du schéma de l'entrepôt, la modélisation multidimensionnelle du schéma, le mappings entre les schémas sources et le schéma cible, le processus ETL, l'implémentation de l'entrepôt et enfin l'analyse OLAP des besoins entreposés. Notre approche est également associée à un outil.

1.2 Contributions

Ce travail de thèse est multidisciplinaire faisant interagir trois principales disciplines pour l'établissent de nos propositions : l'ingénierie des besoins, les solutions d'entrepôts de données et les ontologies. Ce travail a d'abord nécessité une revue de littérature des notions clés issues de ces trois disciplines. Nous avons proposé un deuxième état de l'art synthétisant et comparant les principaux travaux problèmes d'intégration, de gestion (modélisation et analyse) et d'exploitation (transactionnelle ou décisionnelle) des besoins des utilisateurs. Cet état de l'art une

sélection des travaux les plus pertinents dans un domaine foisonnant de propositions, il a également nécessité une identification des critères de comparaison permettant de positionner nos contributions, que nous résumons dans ce qui suit.

1.2.1 Construction de l'ontologie intégrante

Afin de répondre à l'objectif principal de notre travail, qui consiste en la proposition d'un entrepôt de besoins fonctionnels, nous avons émis quelques hypothèses sur les sources de besoins. Afin de nous rapprocher le plus possible d'un scénario réel de projets des grandes entreprises, nous avons considéré un système faisant intervenir plusieurs partenaire ayant chacun une source de besoins. Chaque partenaire décrit ses besoins par sa propose terminologie, sa vision de modélisation des concepts du monde réel et son langage de modélisation. Afin d'expliciter la sémantique de chaque source, cette dernière est connectée à une ontologie locale linguistique pour décrire les termes utilisés et conceptuelle pour décrire les concepts utilisés. Nous retrouvons ce scénario dans de nombreux systèmes d'intégration orientés données, où l'utilisation des ontologies a été largement adoptée. Notre première contribution a consisté à construire l'ontologie intégrant l'ensemble des ontologies locales et éliminant de ce fait les conflits syntaxiques, linguistiques et sémantiques entre les partenaires. Deux structures ontologiques sont envisageables à ce niveau : un premier scénario où les ontologies locales sont construites à partir d'une ontologie partagée (un schéma commun imposé), et un deuxième scénario où les ontologies locales sont définies indépendamment les unes des autres sans se référer à aucun schéma global. Ce dernier scénario a nécessité l'utilisation de techniques de matching ontologique. Deux expérimentations sont proposées pour tester ces deux scénarios.

1.2.2 Définition du schéma de l'entrepôt de besoins

La deuxième contribution a consisté à définir le schéma de l'entrepôt de besoins. Ce schéma est défini en deux étapes : (i) l'unification des formalismes ou langages de modélisation représentant les besoins des différentes sources participants au système (ii) la modélisation multidimensionnelle du schéma. Plusieurs formalismes peuvent être utilisés par les sources selon les compétences ou les préférences des partenaires du projet d'entreposage. La gestion des ces hétérogénéités de formalismes se base sur un modèle pivot de besoins proposé dans notre laboratoire dans le cadre de la thèse d'Ilyes Boukhari [27]. Nous avons adapté et enrichi ce modèle pour fournir une modélisation détaillée de chaque besoin permettant d'identifier les sujets du besoins, ses actions et les objets sur lesquels agit le besoin. Ce modèle enrichi permet également d'identifier plusieurs types de relations entre les besoins. Dans la deuxième étape, et afin de permettre des analyses OLAP sur les besoins, nous organisons le modèle pivot selon une modélisation multidimensionnelle mettant l'entité besoin au centre de l'analyse et identifiant plusieurs entités considérées comme les dimensions d'analyse possibles. Ce modèle obtenu sera ensuite connecté à l'ontologie intégrante définie précédemment afin de permettre une définition

sémantique de l'ensemble des besoins.

1.2.3 Définition d'un ETL pour l'alimentation de l'entrepôt

Le schéma de l'entrepôt de besoins étant défini, la troisième contribution a consisté à définir une démarche d'alimentation de ce schéma par les instances stockées au niveau des sources. Cette démarche a nécessite de revisité la phase ETL conventionnelle et son adaptation pour les besoins. Cette phase repose sur la définition des mappings entre les schémas sources et le schéma cible de l'entrepôt. Ces mappings sont définis en utilisant un ensemble d'opérateurs ETL. Nous avons adapté ces opérateurs ETL et nous les avons enrichi par des opérateurs de raisonnement afin d'assurer un processus ETL de qualité. Un processus (algorithme) ETL a également été proposé et testé. Le mécanisme de raisonnement est également testé sur deux bancs d'essai.

1.2.4 Déploiement et exploitation de l'entrepôt de besoins

La quatrième contribution procède aux phases finales de déploiement et d'analyse de l'entrepôt de besoins fonctionnels obtenu. Le déploiement de l'entrepôt comprend trois étapes : le choix du système de gestion de bases de données (SGBD) cible, la phase de conception logique et la phase de conception physique. L'implémentation de l'entrepôt de besoins est proposé sur deux SGBD sémantiques Oracle et OntoDB. L'objectif ultime d'un projet d'entrepôt est de pouvoir analyser les objets entreposés par diverses techniques. La phase d'analyse dans notre approche est assurée par des requêtes OLAP et l'implémentation d'un cube multidimensionnel des besoins unifiés.

Des expérimentations sont réalisées afin d'évaluer nos propositions utilisant le bancs d'essai LUBM, le document de besoins Courses Management System CMS, l'ontologie linguistique EuroWordNet et les ontologies du projet MultiFarm. Un prototype d'outil implémentant nos principales contributions est également proposé, destiné au concepteur lui permettant de valider l'ensemble des étapes de conception et de générer l'entrepôt de besoins.

Dans la section suivante, nous présentons quelques perspectives que nous jugeons importantes dans les travaux futurs.

1.3 Perspectives

Les contributions proposées dans cette thèse laissent envisager de nombreuses perspectives.

Les perspectives envisageables à *court terme* sont les suivantes :

1.3.1 Intégration des bases de connaissances

Notre approche envisage le scénario de définition des sources de besoins par des ontologies locale. Nous envisageons comme première perspective d'étendre notre approche par l'utilisation d'une base de connaissances (Knowledge base) telle que *Yago* ou *Bpedia*. D'une part, de nombreuses études ont identifié le rôle crucial des bases de connaissances pour les tâches analytiques, car elles fournissent une vision très globale d'un domaine et permettent l'analyse de plusieurs entités (acteurs, lieux, produits, etc.). D'autre part, des travaux récents proposent d'exploiter les bases de connaissances pour construire une solution d'entreposage. L'utilisation de bases de connaissances peut donner plus de valeur à l'entrepôt cible.

L'utilisation des bases de connaissances peut également servir à l'évaluation de la qualité des ontologies considérées. Notre approche repose sur l'utilisation et l'intégration des ontologies sources définissant les besoins. La gestion de la qualité de ces ontologies est primordiale.

1.3.2 Validation de l'approche sur un projet réel

Notre approche doit encore être testée sur de grandes sources de besoins issues de projets réels considérant différents langages de modélisation (informels, semi formels et formels). De nombreuses dimensions d'évaluation de l'approche peuvent être considérées comme la faisabilité et robustesse de l'approche, le passage à l'échelle lorsque le nombre de besoins évolue considérablement, les optimisations à envisager, les aspects économiques de la mise en œuvre de l'approche, etc.

Les perspectives envisageables à *moyen terme* sont les suivantes :

1.3.3 D'autres mécanismes de raisonnement

Notre approche propose un mécanisme de raisonnement testé et validé afin d'identifier les relations entre les besoins, d'inférer de nouvelles relations et d'identifier les incohérences entre les besoins. Ce mécanisme de raisonnement peut être étendu afin de permettre la découverte automatique de mappings entre les sources de besoins et le schéma de l'entrepôt. L'alimentation de l'entrepôt de besoins repose sur le processus ETL qui exploite un ensemble de mappings entre les besoins sources et le schéma cible de l'entrepôt. L'évolution des sources peut avoir un impact sur la définition de ces mappings. Un mécanisme de raisonnement permettrait de mettre à jour les mappings et de découvrir et identifier de nouveaux mappings.

Un autre mécanisme de raisonnement permettrait d'établir une stratégie de validation/rejet des besoins en cas d'incohérences et de conflits entre un ensemble de besoins. Cette étape est gérée manuellement par le concepteur dans notre approche.

1.3.4 Stratégie d'évolution des sources

Notre approche prend comme entrées les sources de besoins constituées de l'ensemble des besoins des partenaires et des ontologies définissant ces besoins. Dans un environnement complexe où les besoins des partenaires peuvent évoluer rapidement pour diverses raisons (technologiques, économiques, etc), l'établissement d'une stratégie globale de gestion de l'évolution des sources de besoins est nécessaire. La traçabilité des besoins serait une étape clé dans cette gestion de l'évolution. Cette évolution peut concerner : les besoins eux mêmes, les langages de modélisation utilisées, les ontologies (concepts, termes, langages, etc), etc.

Les perspectives envisageables à *long terme* sont les suivantes :

1.3.5 Davantage de décisionnel sur les besoins

Nous proposons également d'enrichir la phase d'exploitation de l'entrepôt de besoins par d'autres techniques inspirées des techniques utilisées pour l'OLAP sur les données, en fournissant : un langage de requêtes OLAP dédié aux besoins qui peut être complété par des outils d'interrogation destinés aux décideurs et gestionnaires (une sorte d'outil QBE pour les besoins), la fouille des besoins entreposés, des techniques de visualisation des besoins unifiés, des rapports prédéfinis ou adhoc ou des techniques de réutilisation des besoins existants.

Ce travail peut être complété par d'autres techniques décisionnelles (autres que l'OLAP) comme l'aide à la décision multi-critères. Le processus de prise de décision décrit par Herbert Simon [137] connu sous le nom de modèle IDC, se compose de trois principales phases : Intelligence, Design et Choice . En se référant à ce processus, l'analyse OLAP se place à la phase à la phase d'*intelligence* permettant d'analyser la situation décisionnelle, alors que d'autres approches comme l'aide à la décision multicritère se place à la phase de *choix* [112]. De nombreuses autres techniques décisionnelles peuvent ainsi compléter notre approche, et même influencer l'approche de conception pour : la sélection des besoins à entreposer ou la définition du processus ETL prenant en compte des besoins pondérés.

1.3.6 Vers un entrepôt générique

Une dernière perspective serait de définir une approche de conception simultanée des données et des besoins des partenaires interagissant dans un projet. Les besoins des utilisateurs permettent de définir les données et les traitement d'un système donné. La connexion entre les données et les besoins peut ainsi se faire aisément. L'intégration des besoins non fonctionnels vus comme des contraintes peut également être considérée. L'exploitation des outils et techniques OLAP sur les données et besoins supportant plusieurs croisement de leurs dimensions pourrait apporter une valeur ajoutée aux décideurs du projet. Ce travail pourrait être généralisé pour fournir un entrepôt d'entités (objets) génériques prenant en compte toutes les entités clés dans une entreprise ou organisation (données, besoins, processus, etc).

Bibliographie

[1] Omg: Sysml specification. omg ptc/06-05-04, http://www.sysml.org/specs.htm.

[2] Industrial automation systems and integration - parts library - part 42: Description methodology: Methodology for structuring parts families. Technical report, 1998.

[3] Industrial automation systems and integration - parts library - part 25: Logical resource: Logical model of supplier library with aggregate values and explicit content. Technical report, 2004.

[4] User requirements notation (urn) - language definition. Technical report, 2012.

[5] Guide to software engineering body of knowledge. IEEE Computer Society, Colorado, Accessed 19 October 2009.

[6] Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A. Bernstein, Michael J. Carey, Surajit Chaudhuri, Jeffrey Dean, AnHai Doan, Michael J. Franklin, Johannes Gehrke, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Donald Kossmann, Samuel Madden, Sharad Mehrotra, Tova Milo, Jeffrey F. Naughton, Raghu Ramakrishnan, Volker Markl, Christopher Olston, Beng Chin Ooi, Christopher Ré, Dan Suciu, Michael Stonebraker, Todd Walter, and Jennifer Widom. The beckman report on database research. *Commun. ACM*, 59(2):92–99, 2016.

[7] A. Abran, J.W. Moore, P. Bourque, and R.E. Dupuis. *Guide to the Software Engineering Body of Knowledge.* http://www.swebok.org/.IEEE Computer Society, 2004.

[8] J.R. Abrial. *The B-book: assigning programs to meanings*. Cambridge University Press, New York, USA, 1996.

[9] Christopher Adamson. *Mastering data warehouse aggregates: solutions for star schema performance*. Wiley, 2012.

[10] Nicole Alexander and Siva Ravada. Rdf object type and reification in oracle. *Technical White Paper [cited 11/29/06*, 2005.

[11] Kerkad Amira. *L'interaction au service de l'optimisation à grande échelle des entrepôts de données relationnels*. PhD thesis, dec 2013.

[12] Y. Arens and C. A. Knoblock. Sims: Retrieving and integrating information from multiple sources. *Proceedings of the International Conference on Mana-*

gement of Data (SIGMOD'1993), pages 562–563, May 1993.

[13] N Assawamekin, T Sunetnanta, C Pluempitiwiriyawej, et al. Ontology-based multiperspective requirements traceability framework. 2009.

[14] Aybüke Aurum and Claes Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003.

[15] Aybüke Aurum and Claes Wohlin. Aligning requirements with business objectives: A framework for requirements engineering decisions. In *Proceedings of Requirements Engineering Decision Support Workshop*. Paris, France, 2005.

[16] Seyed-Mehdi-Reza Beheshti, Boualem Benatallah, Sherif Sakr, Daniela Grigori, Hamid Reza Motahari-Nezhad, Moshe Chai Barukh, Ahmed Gater, and Seung Hwan Ryu. *Process Analytics - Concepts and Techniques for Querying and Analyzing Process Data*. Springer, 2016.

[17] Ladjel Bellatreche. *Contributions à la conception et l'exploitation de systèmes d'intègration de donnèes*. Habilitation à diriger des recherches en informatique, Universitè de Poitiers, Dècembre 2009.

[18] Ladjel Bellatreche, Nguyen Xuan Dung, Guy Pierra, and Dehainsala Hondjack. Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry*, 57(8):711–724, 2006.

[19] Ladjel Bellatreche, Arnaud Giacometti, Patrick Marcel, Hassina Mouloudi, and Dominique Laurent. A personalization framework for olap queries. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 9–18. ACM, 2005.

[20] Ladjel Bellatreche and Robert Wrembel. Special issue on: Evolution and versioning in semantic data integration systems. *Journal on Data Semantics*, 2(2-3):57–59, 2013.

[21] Moussa Benaissa and Abderrahmane Khiat. A new approach for combining the similarity values in ontology alignment. In *International Conference on Computer Science and its Applications CIIA 2015*, pages 343–354. Springer, 2015.

[22] Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio Vincini. The momis approach to information integration. 2001.

[23] Nabila Berkani, Ladjel Bellatreche, and Boualem Benatallah. A value-added approach to design BI applications. In *18th International Conference on Big Data Analytics and Knowledge Discovery (DaWaK)*, pages 361–375, 2016.

[24] Philip A. Bernstein and Umeshwar Dayal. An overview of repository technology. In *Proceedings of the International Conference on Very Large Databases*, pages 705–713, 1994.

[25] Tassadit BOUADI. *Analyse Multidimensionnelle Interactive de Rèsultats de Simulation. Aide Ã la Dècision dans le Domaine de l'agroècologie*. PhD thesis, Universitè de Rennes 1, dec 2014.

[26] Tassadit BOUADI. *Dèveloppement d'un Environnement pour l'Alignement*

des Ontologies Une Approche à Base d'Instances. PhD thesis, Universitè d'Oran 1, Algèrie, dec 2016.

[27] Ilyes Boukhari. *Intégration et exploitation de besoins en entreprise étendue fondées sur la sémantique*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2014.

[28] Ilyès Boukhari, Ladjel Bellatreche, and Stéphane Jean. An ontological pivot model to interoperate heterogeneous user requirements. In *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, pages 344–358. Springer, 2012.

[29] Saartje Brockmans, Peter Haase, Luciano Serafini, and Heiner Stuckenschmidt. Formal and conceptual comparison of ontology mapping languages. In *Modular Ontologies*, pages 267–291. Springer, 2009.

[30] Saartje Brockmans, Peter Haase, and Rudi Studer. A mof-based metamodel and uml syntax for networked ontologies. In *Intl. Semantic Web Conf. Georgia, US*, 2006.

[31] E. Brottier, B. Baudry, Y.Traon, D. Touzet, and B. Nicolas. Producing a global requirement model from multiple requirement specifications. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, (EDOC'07)*, page 390. IEEE Computer Society, 2007.

[32] Erwan Brottier, Benoit Baudry, Yves Le Traon, David Touzet, and Bertrand Nicolas. Producing a global requirement model from multiple requirement specifications. In *Enterprise Distributed Object Computing Conference, 2007. EDOC 2007. 11th IEEE International*, pages 390–390. IEEE, 2007.

[33] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In *Logics for databases and information systems*, pages 229–263. Springer, 1998.

[34] V. Castaneda, L. Ballejos, M.Laura Caliusco, and M. Rosa Galli. The use of ontologies in requirements engineering. *Global Journal of Researches in Engineering (GJRE)*, 10 Issue 6:2–8, 2010.

[35] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International Journal of Intelligent and Cooperative Information Systems*, 2(04):375–398, 1993.

[36] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The tsimmis project: Integration of heterogenous information sources. 1994.

[37] Edgar Frank Codd. Data models in database management. In *ACM SIGMOD Record*, volume 11, pages 112–114. ACM, 1980.

[38] Y. Constantinidis. *Expression des besoins pour le système d'information: Guide d'élaboration du cahier des charges - Préface de Michel Volle*. Solutions d'entreprise. Eyrolles, 2011.

[39] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1):3–50, 1993.

[40] Scott A DeLoach, Mark F Wood, and Clint H Sparkman. Multiagent systems engineering. *International Journal of Software Engineering and Knowledge Engineering*, 11(03):231–258, 2001.

[41] Françoise Détienne. Collaborative design: Managing task interdependencies and multiple perspectives. *Interacting with computers*, 18(1):1–20, 2006.

[42] Samba Diaw, Rédouane Lbath, and Bernard Coulette. Etat de l'art sur le développement logiciel dirigé par les modèles. *Technique et Science Informatique TSI*, 29:505–536, 2008.

[43] Simon C Dik. The theory of functional grammar, part 1: The structure of the cause. 1989.

[44] Warith-Eddine DJEDDI. *Alignement sèmantique des ontologies de grande Taille*. PhD thesis, Universitè Badji Mokhtar Annaba, 2013.

[45] Zouhir Djilani, Nabila Berkani, and Ladjel Bellatreche. Towards functional requirements analytics. In *International Symposium on Leveraging Applications of Formal Methods*, pages 358–373. Springer, 2016.

[46] Zouhir Djilani, Abderrahmane Khiat, Selma Khouri, and Ladjel Bellatreche. Murgroom: multi-site requirement reuse through graph and ontology matching. In *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services*, pages 160–169. ACM, 2016.

[47] Zouhir Djilani and Selma Khouri. Understanding user requirements iceberg: Semantic based approach. In *Model and Data Engineering*, pages 297–310. Springer, 2015.

[48] Zouhir Djilani, Selma Khouri, Abderrahmane Khiat, and Ladjel Bellatreche. Les besoins fonctionnels candidats á l'entreposage et l'analyse en ligne. In *Journées "Analyse de données textuelles", en conjonction avec EDA 2017*, pages 160–169, 2017.

[49] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[50] Philippe Du Bois. The albert ii language: on the design and the use of a formal specification language for requirements analysis. *Unpublished doctoral dissertation). University of Namur, Belgique*, 1995.

[51] Dang Viet Dzung and Atsushi Ohnishi. Ontology-based reasoning in requirements elicitation. In *Software Engineering and Formal Methods, 2009 Seventh IEEE International Conference on*, pages 263–272. IEEE, 2009.

[52] Marc Ehrig. *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[53] Ramez Elmasri. *Fundamentals of database systems*. Pearson Education India, 2008.

[54] J. Euzenat and P. Shvaiko. Ontology matching, second edition. *Springer-Verlag, Heidelberg*, 2013.

[55] Jerome Euzenat, Thanh Le Bach, J Barrasa, P Bouquet, J De Bo, R Dieng-Kuntz, M Ehrig, M Hauswirth, Mustafa Jarrar, R Lara, et al. State of the art

on ontology alignment. *Knowledge Web Deliverable D*, 2:2–3, 2004.

[56] Chimène Fankam. *OntoDB2: un système flexible et efficient de Base de Données à Base Ontologique pour le Web sémantique et les données techniques*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2009.

[57] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[58] F. FranÇois Goasdoué, V. Lattès, and M. C. Rousset. The use of carin language and algorithms for information integration: The picsel system. *International Journal of Cooperative Information Systems (IJCIS)*, 9(4):383–401, December 2000.

[59] G. Gardarin. Intégration de données et applications via xml. `http://georges.gardarin.free.fr/Cours_Total/X6-BusinessIntegration.ppt`.

[60] Michael R Genesereth, Arthur M Keller, and Oliver M Duschka. Infomaster: An information integration system. In *ACM SIGMOD Record*, volume 26, pages 539–542. ACM, 1997.

[61] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Formal reasoning techniques for goal models. In *Journal on Data Semantics I*, pages 1–20. Springer, 2003.

[62] Paolo Giorgini, Stefano Rizzi, and Maddalena Garzetti. Goal-oriented requirement analysis for data warehouse design. In *Proceedings of the 8th ACM international workshop on Data warehousing and OLAP*, pages 47–56. ACM, 2005.

[63] Gloria-Lucia Giraldo Gómez. *Construction automatisée de l'ontologie des systèmes médiateurs: application à des systèmes intégrant des services standards accessibles via le Web*. PhD thesis, Paris 11, 2005.

[64] Cheng Hian Goh, Stéphane Bressan, Stuart Madnick, and Michael Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems (TOIS)*, 17(3):270–293, 1999.

[65] Arda Goknil, Ivan Kurtev, Klaas van den Berg, and Jan-Willem Veldhuis. Semantics of trace relations in requirements models for consistency checking and inferencing. *Software & Systems Modeling*, 10(1):31–54, 2011.

[66] M. Golfarelli. From user requirements to conceptual design in data warehouse design - a survey. In *Data Warehousing Design and Advanced Engineering Applications : Methods for Complex Construction*, pages 1–16. IGI Global, 2009.

[67] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. The dimensional fact model: a conceptual model for data warehouses. *International Journal of Cooperative Information Systems*, 7(02n03):215–247, 1998.

[68] Sol Jaffe Greenspan. Requirements modeling: a knowledge representation approach to software requirements definition. 1984.

[69] T. R. Gruber. A translation approach to portable ontology specifications. In *Knowledge Acquisition*, volume 5, pages 199–220, 1993.

[70] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005.

[71] A. Halevy, A. Rajaraman, and J. Ordille. Data integration: The teenage years. In *Proceedings of the 32Nd International Conference on Very Large Data Bases (VLDB)*, pages 9–16, 2006.

[72] Joachim Hammer, Hector Garcia-Molina, Jennifer Widom, Wilburt Labio, and Yue Zhuge. The stanford data warehousing project. 1995.

[73] Lilia Hannachi, Nadjia Benblidia, Omar Boussaïd, and Fadila Bentayeb. Community cube: a semantic framework for analysing social network data. *IJMSO*, 10(3):155–169, 2015.

[74] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 203–212. ACM, 2010.

[75] Jeff Heflin, James Hendler, and Sean Luke. Shoe: A knowledge representation language for internet applications. 1999.

[76] Constance L Heitmeyer, Ralph D Jeffords, and Bruce G Labaw. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 5(3):231–261, 1996.

[77] Akiko Inaba, Thepchai Supnithi, Mitsuru Ikeda, and Jun'ichi Toyoda. An overview of learning goal ontology. In *ECAI 2000 Workshop on Machine Learning in Computer Vision*, pages 23–30, 2000.

[78] William H Inmon. *Building the data warehouse*. John wiley & sons, 2005.

[79] I. Jacobson and l. Spence K. Bittner. *Use Case Modeling*. Addison Wesley Professional, 2002.

[80] Matthew A Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[81] Stéphane Jean. *OntoQL, un langage d'exploitation des bases de données à base ontologique*. PhD thesis, Université de Poitiers, 2007.

[82] Stéphane Jean, Idir Ait-Sadoune, Ladjel Bellatreche, and Ilyès Boukhari. On using requirements throughout the life cycle of data repository. In *International Conference on Database and Expert Systems Applications*, pages 409–416. Springer, 2014.

[83] Manfred A Jeusfeld, Christoph Quix, and Matthias Jarke. Design and analysis of quality information for data warehouses. In *International Conference on Conceptual Modeling*, pages 349–362. Springer, 1998.

[84] E. Jiménez-Ruiz, B.C. Grau, U. Sattler, T. Schneider, and R. Berlanga Llavori. Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *Description Logics*, 2008.

[85] Haruhiko Kaiya and Motoshi Saeki. Ontology based requirements analy-

sis: lightweight semantic processing approach. pages 223–230, 2005.

[86] Haruhiko Kaiya and Motoshi Saeki. Using domain ontology as domain knowledge for requirements elicitation. In *Requirements Engineering, 14th IEEE International Conference*, pages 189–198. IEEE, 2006.

[87] Abderrahmane Khiat. Crolom: Cross-lingual ontology matching system. *Ontology Matching*, page 146, 2016.

[88] Abderrahmane Khiat and Moussa Benaissa. A new instance-based approach for ontology alignment. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 11(3):25–43, 2015.

[89] Abderrahmane Khiat, Elhabib Abdelillah Ouhiba, Mohammed Amine Belfedhal, and Chihab Eddine Zoua. Simcat results for oaei 2016. *Ontology Matching*, page 217, 2016.

[90] Selma Khouri. *Cycle de vie sémantique de conception de systèmes de stockage et manipulation de données*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2013.

[91] Selma Khouri, Ladjel Bellatreche, Stéphane Jean, and Yamine Ait-Ameur. Requirements driven data warehouse design: We can go further. In *ISOLA*, pages 588–603. Springer, 2014.

[92] Ralph Kimball and Margy Ross. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.

[93] Ralph Kimball, Margy Ross, et al. The data warehouse toolkit: the complete guide to dimensional modelling. *New York ua*, 2002.

[94] Hideaki Kimura, George Huo, Alexander Rasin, Samuel Madden, and Stanley B Zdonik. Coradd: Correlation aware database designer for materialized views and indexes. *Proceedings of the VLDB Endowment*, 3(1-2):1103–1113, 2010.

[95] Sven J Körner and Torben Brumm. Natural language specification improvement with ontologies. *International Journal of Semantic Computing*, 3(04):445–470, 2009.

[96] Maha Ben Kraiem, Jamel Feki, Kaïs Khrouf, Franck Ravat, and Olivier Teste. Olap of the tweets: from modeling toward exploitation. In *Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on*, pages 1–10. IEEE, 2014.

[97] R. Laleau, F. Semmak, A Matoussi, D. Petit, A. Hammad, and B.Tatibouet. A first attempt to combine sysml requirements diagrams and b. *Innovations in Systems and Software Engineering (ISSE)*, 6(1-2):47–54, 2010.

[98] A. Lamsweerde. Conceptual modeling: Foundations and applications. chapter Reasoning About Alternative Requirements Options, pages 380–397. Springer-Verlag, 2009.

[99] Seok Won Lee and Robin A Gandhi. Ontology-based active requirements engineering framework. In *Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific*, pages 8–pp. IEEE, 2005.

[100] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.

[101] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[102] A. Y. Levy, A. Rajaraman, and J. J. Ordille. The world wide web as a collection of views: Query processing in the information manifold. *Proceedings of the International Workshop on Materialized Views: Techniques and Applications (VIEW'1996)*, pages 43–55, June 1996.

[103] Juanzi Li, Jie Tang, Yi Li, and Qiong Luo. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Trans. on Knowl. and Data Eng.*, 21(8):1218–1232, 2009.

[104] Chi-Lun Liu. Cdnfre: Conflict detector in non-functional requirements evolution based on ontologies. *Computer Standards & Interfaces*, 2016.

[105] O. López, M. A. Laguna, and F. J. G.Peñalvo. A metamodel for requirements reuse. In *VII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pages 427–428, 2002.

[106] Oscar López, Miguel A Laguna, and Francisco José García Peñalvo. Metamodeling for requirements reuse. In *WER*, pages 76–90, 2002.

[107] Jing Lu, Li Ma, Lei Zhang, Jean-Sébastien Brunner, Chen Wang, Yue Pan, and Yong Yu. Sor: A practical system for ontology storage, reasoning and search. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 1402–1405. VLDB Endowment, 2007.

[108] Jose-Norberto Mazón, Juan Trujillo, and Jens Lechtenbörger. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. *Data & Knowledge Engineering*, 63(3):725–751, 2007.

[109] I. Mirbel and S. Villata. Enhancing goal-based requirements consistency: An argumentation-based approach. In *13th International Workshop in Computational Logic in Multi-Agent Systems (CLIMA)* ., pages 110–127, 2012.

[110] Riichiro Mizoguchi, Kouji Kozaki, Toshinobu Sano, and Yoshinobu Kitamura. *Construction and Deployment of a Plant Ontology*, pages 113–128. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[111] Imen Moalla, Ahlem Nabli, Lotfi Bouzguenda, and Mohamed Hammami. Data warehouse design approaches from social media: review and comparison. *Social Network Analysis and Mining*, 7(1):5, 2017.

[112] Manuel Mora, Guisseppi Forgionne, and J Gupta. Decision making support systems. *Idea Group Publishing, Hershey*, 2003.

[113] E. Navarro, J.A Mocholi, P. Letelier, and I. Ramos. A metamodeling approach for requirements specification.pdf. *Journal of Computer Information Systems (JCIS)*, 46:67–77, 2006.

[114] Dung Xuan Nguyen. *Intégration de*

bases de données hétérogènes par articulation à priori d'ontologies: application aux catalogues de composants industriels. PhD thesis, Université de Poitiers, 2006.

[115] Tuong Huan Nguyen, John Grundy, and Mohamed Almorsy. Integrating goal-oriented and use case-based requirements engineering: The missing link. In *Model Driven Engineering Languages and Systems (MODELS), 2015 ACM/IEEE 18th International Conference on*, pages 328–337. IEEE, 2015.

[116] N. Noy. Ontology mapping and alignment. In *The Third Summer School on Ontological Engineering and the Semantic Web (SSSW'05)*, 2005.

[117] B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering, (ICSE'00)*, pages 35–46. ACM, 2000.

[118] Johan Natt och Dag, Vincenzo Gervasi, Sjaak Brinkkemper, and Björn Regnell. A linguistic-engineering approach to large-scale requirements management. *IEEE Software*, 22(1):32–39, 2005.

[119] Gilles Perrouin, Erwan Brottier, Benoit Baudry, and Yves Le Traon. Composing models for detecting inconsistencies: A requirements engineering perspective. In *Requirements Engineering: Foundation for Software Quality*, pages 89–103. Springer, 2009.

[120] Guy Pierra. Un modéle formel d'ontologie pour l'ingénierie, le commerce électronique et le web sémantique: Le modéle de dictionnaire sémantique plib. *Journées Scientifiques WEB SEMANTIQUE, Paris*, 2002.

[121] Guy Pierra. Context-explication in conceptual ontologies: the plib approach. In *Proceedings of 10th ISPE International Conference on Concurrent Engineering: Research and Applications (ce'03) : Special Track on Data Integration in Engineering*, pages 243–253, 2003.

[122] G. Pierre. Merise: Modélisation de systèmes d'information. 2005.

[123] Klaus Pohl. *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[124] Valéry Psyché, Olavo Mendes, Jacqueline Bourdeau, et al. Apport de l'ingénierie ontologique aux environnements de formation à distance. *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, 10:89–126, 2003.

[125] Björn Regnell, Barbara Paech, Aybüke Aurum, Claes Wohlin, Allen Dutoit, and Johan Natt och Dag. Requirements mean decisions!–research issues for understanding and supporting decision-making in requirements engineering. In *First Swedish Conference on Software Engineering Research and Practise: Proceedings*, 2001.

[126] Nafees Ur Rehman, Svetlana Mansmann, Andreas Weiler, and Marc H Scholl. Building a data warehouse for twitter stream exploration. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pages 1341–1348. IEEE Computer Society, 2012.

[127] Stefano Rizzi, Alberto Abelló, Jens Lechtenbörger, and Juan Trujillo. Research in data warehouse modeling and design: dead or alive? In *Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pages 3–10. ACM, 2006.

[128] A. Rochfeld and H. Tardieu. Merise: An information system design and development methodology. *Information & Management*, 6:143–159, 1983.

[129] C. Rolland. Requirements engineering for cots based systems. *Information and Software Technology (IST)*, 41(14):985–990, 1999.

[130] Colette Rolland. Ingénierie des besoins: L'approche l'ecritoire. *Journal Techniques de l'Ingénieur*, page 1, 2003.

[131] Marcello La Rosa, Hajo A. Reijers, Wil M. P. van der Aalst, Remco M. Dijkman, Jan Mendling, Marlon Dumas, and Luciano García-Bañuelos. APROMORE: an advanced process model repository. *Expert Systems with Applications*, 38(6):7029–7040, 2011.

[132] D.T. Ross and K.E. Schoman. Structured analysis for requirements definition. *IEEE Transactions on Software Engineering*, 3(1):6–15, 1977.

[133] Kevin Royer. *Vers un entrepôt de données et des processus: le cas de la mobilité électrique chez EDF*. PhD thesis, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers, 2015.

[134] M Hayashi Saeki and H S Kaiya. A tool for attributed goal-oriented requirements analysis. *Automated Software Engineering*, 2009.

[135] Vikas Shukla. *Comprehensive méthodology for the complex systems' requirements engineering & decision making*. PhD thesis, Toulouse, INSA, 2014.

[136] Katja Siegemund, Edward J Thomas, Yuting Zhao, Jeff Pan, and Uwe Assmann. Towards ontology-driven requirements engineering. In *Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference (ISWC), Bonn*, 2011.

[137] Herbert A Simon. The new science of management decision. 1960.

[138] Dimitrios Skoutas and Alkis Simitsis. Ontology-based conceptual design of etl processes for both structured and semi-structured data. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(4):1–24, 2007.

[139] Alessandro Solimando, Ernesto Jiménez-Ruiz, and Giovanna Guerrini. Detecting and correcting conservativity principle violations in ontology-to-ontology mappings. In *International Semantic Web Conference*, pages 1–16. Springer, 2014. [DOI:10.1007/978-3-319-11915-1_1].

[140] Ian Sommerville. Integrated requirements engineering: A tutorial. *Software, IEEE*, 22(1):16–23, 2005.

[141] S. Staab, M. Erdmann, A. Maedche, and s. Decker. Knowledge media in healthcare. chapter An extensible approach for modeling ontologies in RDF(S), pages 234–253. IGI Publishing, 2002.

[142] Yu-Jen John Sun, Moshe Chai Barukh, Boualem Benatallah, and Seyed-Mehdi-Reza Beheshti. Scalable saas-based process customization with casewalls. In *ICSOC*, pages 218–233, 2015.

[143] Yufei Sun, Liangli Ma a, and Shuang Wang. A comparative evaluation of string similarity metrics for ontology alignment. *Information and Computational Science*, 12:957–964, 2015.

[144] Ambrosio Toval, Joaquín Nicolás, Begoña Moros, and Fernando García. Requirements reuse for improving information systems security: a practitioner's approach. *Requirements Engineering*, 6(4):205–219, 2002.

[145] Axel V. Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, (RE'01)*, pages 249–263. IEEE Computer Society, 2001.

[146] Axel van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *Proceedings of the 22nd international conference on Software engineering ,(ICSE'00)*, pages 5–19. ACM, 2000.

[147] Axel Van Lamsweerde, Robert Darimont, and Emmanuel Letier. Managing conflicts in goal-driven requirements engineering. *Software Engineering, IEEE Transactions on*, 24(11):908–926, 1998.

[148] Panos Vassiliadis, Mokrane Bouzeghoub, and Christoph Quix. Towards quality-oriented data warehouse usage and evolution. *Information Systems*, 25(2):89–115, 2000.

[149] C. Vicente-Chicote, B. Moros, and A. Toval. Remm-studio: an integrated modeldriven environment for requirements specification, validation and formatting. *Journal of Object Technology*, 6(9):437–454, 2007.

[150] Ubbo Visser, Heiner Stuckenschmidt, Holger Wache, and Thomas Vögele. Enabling technologies for interoperability. In *Workshop on the 14th International Symposium of Computer Science for Environmental Protection*, pages 35–46. Citeseer, 2000.

[151] Piek Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[152] Piek Vossen, Laura Bloksma, Wim Peters, Claudia Kunze, Andreas Wagner, Karel Pala, Kadri Vider, and Francesca Bertagna. Extending the inter-lingual-index with new concepts. *Deliverable 2D010, EuroWordNet, LE2-4003*, 1999.

[153] Piek Vossen, Laura Bloksma, Horacio Rodriguez, Salvador Climent, Nicoletta Calzolari, Adriana Roventini, Francesca Bertagna, and Antonietta Alonge. The eurowordnet base concepts and top ontology. 1998.

[154] V.R.Basili, G. Gianluigi, and H.D. Rombach. The goal question metric approach. computer science technical report series cs-tr-2956. Technical report, 1992.

[155] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. Ontology-based integration of information-a survey of existing approaches. In *IJCAI-01 workshop: ontologies and information sharing*, volume 2001, pages 108–117. Citeseer, 2001.

[156] Paul Westerman. *Data Warehousing: Using the Wal-Mart Model*. Morgan

Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.

[157] R. Wieringa, E. Dubois, and H. S. Huyts. Integrating semi-formal and formal requirements. In *Proceedings of the 9th International ConferenceAdvanced Information Systems Engineering, (CAiSE'97)*, pages 19–32, 1997.

[158] Kevin Wilkinson, Craig Sayers, Harumi Kuno, and Dave Reynolds. Efficient rdf storage and retrieval in jena2. In *Proceedings of the First International Conference on Semantic Web and Databases*, pages 120–139. CEUR-WS. org, 2003.

[159] William E Winkler. The state of record linkage and current research problems. In *Statistical Research Division, US Census Bureau*. Citeseer, 1999.

[160] Claes Wohlin and Aybüke Aurum. What is important when deciding to include a software requirement in a project or release? In *Empirical Software Engineering, 2005. 2005 International Symposium on*, pages 10–pp. IEEE, 2005.

[161] Katharina Wolter, Michal Smialek, Daniel Bildhauer, and Hermann Kaindl. *Reusing terminology for requirements specifications from WordNet*. IEEE, 2008.

[162] Dung Nguyen Xuan, Ladjel Bellatreche, and Guy Pierra. Ontodawa, un système d'intégration à base ontologique de sources de données autonomes et évolutives. *Ingénierie des Systèmes d'Information*, 13(2):97–125, 2008.

[163] Wang Y., Haase P., and Bao J. A survey of formalisms for modular ontologies. In *In: IJCAI 2007. Workshop SWeCKa.*, 2007.

[164] E. Siu-Kwong. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, 1996.

[165] E. Siu-Kwong. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering, (RE'97)*, page 226. IEEE Computer Society, 1997.

[166] Katrin Simone Zaiss. *Instance-based ontology matching and the evaluation of matching systems*. PhD thesis, Düsseldorf, Univ., Diss., 2010, 2010.

Annexe A :

L'annexe suivante présente les règles de transformation des besoins locales au modèle pivot des besoins.

1. **Règle 1 : OntoGoal-to-OntoPivot**. Cette règle décrit la transformation des besoins (instances ontologiques) exprimés par le modèle de buts vers le modèle pivot des besoins comme le montre la figure 1.
2. **Règle 2 : OntoUcase-to-OntoPivot**. Cette règle décrit la transformation des besoins (instances ontologiques) exprimés par le modèle des cas d'utilisation vers le modèle pivot des besoins comme le montre la figure 2.
3. **Règle 3 : OntoMCT-to-OntoReq**. Cette règle décrit la transformation des besoins (instances ontologiques) exprimés par le modèle des traitements vers le modèle pivot des besoins comme le montre la figure 3.

```
1.  module OntoGoalInstance2OntoEPivotInstance;
2.  create OUT : EPivotModel from IN : MMGoal;
3.  rule MMGoalGoal2EPivotModeRequirement
4.  {
5.      from
6.          s : MMGoal!Goal
7.      to
8.          t : EPivotModel!Requirement (
9.                  IdReq <— IdReq + 1,
10.                 IdReq <— s.IdGoal,
11.                 NameReq <— s.NameGoal,
12.                 DescriptionReq <— s.DescriptionGoal,
13.                 Context <— s.Context,
14.                 PurposeReq <— s.PurposeGoal,
15.                 PriorityReq <— s.Priority,
16.                 DateSpec <— s.DateCreat )
17. }
18. rule MMGoalActor2EPivotModelSubject
19. {
20.     from
21.         s : MMGoal!Actor
22.     to
23.         t : EPivotModel!Subject (
24.                 IdActor <— IdActor + 1,
25.                 IdActor <— s.IdActor,
26.                 Name <— s.NameActor )
27. }
28. rule MMGoalTask2EPivotModelAction
29. {
30.     from
31.         s : MMGoal!Task
32.     to
33.         t : EPivotModel!Action (
34.                 IdTask <— IdTask +1,
35.                 IdAction <— a.IdTask,
36.                 DescriptionAction <— 'TockenL'(a.DescriptionTask )
37. }
38. rule MMGoalTask2EPivotModelObject
39. {
40.     from
41.         s : MMGoal!Task
42.     to
43.         t : EPivotModel!Object (
44.                 IdTask <— IdTask +1,
45.                 IdObject <— a.IdTask,
46.                 DescriptionObject <— 'TockenR'(a.DescriptionTask )
47. }
48. rule MMGoalResult2EPivotModelResult
49. {
50.     from
51.         s : MMGoal!Result
52.     to
53.         t : EPivotModel! Result (
54.                 IdResult <— IdResult + 1,
55.                 IdResult <— a.IdResult,
56.                 Name <— a.RangResult,
57.                 Name <— a.Description )
58. }
59. rule MMGoalRelationships2EPivotModelRelationships
60. {
61.     from
62.         s : MMGoal! GoalRelationships
63.     to
64.         t : EPivotModel!Relationships (
65.                 IdRelationships <— IdRelationships + 1,
66.                 IdRelationships <— a.IdGoalRelationships,
67.                 Relationships <— a.GoalRelationships)
68. }
69. rule EPivotActor (Name: String)
70. {
71.     from
72.         s : MMGoal
73.     to
74.         t : EPivotModel!Actor (
75.                 IdActor <— IdActor +1,
76.                 Name <— 'EnglandPartner' )
77. }
```

FIGURE 1 – Règle 1 : transformation OntoGoal-to-OntoPivot

```
1.  module  OntoUseCaseInstance2OntoEPivotInstance;
2.  create  OUT : EPivotModel  from  IN : MMUseCase;
3.  rule MMUseCaseGoal2EPivotModeRequirement
4.  {
5.      from
6.          s : MMUseCase! UseCase
7.      to
8.          t : EPivotModel!Requirement (
9.              IdReq<- IdReq + 1,
10.             IdReq<- s.IdUseCase,
11.             NameReq<- s.NameUseCase,
12.             DescriptionReq <- s.DescriptionUseCase,
13.             Context<- s.ContextUseCase,
14.             PurposeReq<- s.PurposeUseCase,
15.             PriorityReq <- s.PriorityUseCase,
16.             StatusReq <- s.StatusUseCase,
17.             TypeReq  <- s.TypeUseCase,
18.             DateSpec <- s.DateCreat UseCase )
19. }
20.  rule MMUseCaseActor2EPivotModelSubject
21.  {
22.      from
23.          s : MMUseCase!Actor
24.      to
25.          t : EPivotModel!Subject (
26.              IdActor<- IdActor + 1,
27.              IdActor<- s.IdActor.UseCase,
28.              Name<- s.NameActor.UseCase )
29. }
30.  rule MMUseCaseAction2EPivotModelAction
31.  {
32.      from
33.          s : MMUseCase! Action
34.      to
35.          t : EPivotModel!Action (
36.              IdTask<- IdTask +1,
37.              IdAction<- a.IdTask,
38.              DescriptionAction<- TockenL(a.DescriptionTask ))
39. }
40.  rule MMUseCaseAction2EPivotModelObject
41.  {
42.      from
43.          s : MMUseCase! Action
44.      to
45.          t : EPivotModel!Object (
46.              IdTask<- IdTask +1,
47.              IdObject<- a.IdTask,
48.              DescriptionObject<- TockenR(a.DescriptionTask ))
49. }
50.  rule MMUseCaseResult2EPivotModelResult
51.  {
52.      from
53.          s : MMUseCase!Result
54.      to
55.          t : EPivotModel! Result (
56.              IdResult <- IdResult + 1,
57.              IdResult<- a.IdResult,
58.              Name<- a.RangResult
59.              Name<- a.Description )
60. }
61.  rule MMUseCaseRelationships2EPivotModelRelationships
62.  {
63.      from
64.          s : MMUseCase! UseCaseRelationships
65.      to
66.          t : EPivotModel!Relationships (
67.              IdRelationships<- IdRelationships + 1,
68.              IdRelationships<- a.IdUseCaseRelationships,
69.              Relationships<- a. UseCaseRelationships)
70. }
71.  rule EPivotActor (Name: String)
72.  {
73.      from
74.          s : MMUseCase! UseCaseRelationships
75.      to
76.          t : EPivotModel!Actor (
77.              IdActor <- IdActor +1,
78.              Name<- 'SpanishPartner' )
79. }
```

FIGURE 2 – Règle 2 : transformation OntoUcase-to-OntoPivot

```
1.  module  OntoMCTInstance2OntoEPivotInstance;
2.  create   OUT : EPivotModel   from   IN : MMMCT;
3.  rule MMMCTGoal2EPivotModeRequirement
4.  {
5.      from
6.          s : MMMCT! MCT
7.      to
8.          t : EPivotModel!Requirement (
9.                  IdReq<- IdReq + 1,
10.                 IdReq<- s.IdMCT,
11.                 NameReq<- s.NameMCT,
12.                 DescriptionReq <- s.DescriptionMCT,
13.                 Context<- s.ContextMCT,
14.                 PurposeReq<- s.PurposeMCT,
15.                 PriorityReq <- s.PriorityMCT,
16.                 StatusReq <- s.StatusMCT,
17.                 TypeReq  <- s.TypeMCT,
18.                 DateSpec <- s.DateCreat MCT )
19. }
20.  rule MMMCTActor2EPivotModelSubject
21.  {
22.     from
23.         s : MMMCT!Actor
24.     to
25.         t : EPivotModel!Subject (
26.             IdActor<- IdActor + 1,
27.             IdActor<- s.IdActor.MCT,
28.             Name<- s.NameActor.MCT )
29. }
30.  rule MMMCTAction2EPivotModelAction
31.  {
32.     from
33.         s : MMMCT! Action
34.     to
35.         t : EPivotModel!Action (
36.             IdTask<- IdTask +1,
37.             IdAction<- a.IdTask,
38.             DescriptionAction<- TockenL(a.DescriptionTask ))
39. }
40.  rule MMMCTAction2EPivotModelObject
41.  {
42.     from
43.         s : MMMCT! Action
44.     to
45.         t : EPivotModel!Object (
46.             IdTask<- IdTask +1,
47.             IdObject<- a.IdTask,
48.             DescriptionObject<- TockenR(a.DescriptionTask ))
49. }
50.  rule MMMCTResult2EPivotModelResult
51.  {
52.     from
53.         s : MMMCT!Result
54.     to
55.         t : EPivotModel! Result (
56.             IdResult <- IdResult + 1,
57.             IdResult<- a.IdResult,
58.             Name<- a.RangResult
59.             Name<- a.Description )
60. }
61.  rule MMMCTRelationships2EPivotModelRelationships
62.  {
63.     from
64.         s : MMMCT! MCTRelationships
65.     to
66.         t : EPivotModel!Relationships (
67.             IdRelationships<- IdRelationships + 1,
68.             IdRelationships<- a.IdMCTRelationships,
69.             Relationships<- a. MCTRelationships)
70. }
71.  rule EPivotActor (Name: String)
72.  {
73.     from
74.         s : MMMCT! MCTRelationships
75.     to
76.         t : EPivotModel!Actor (
77.            IdActor <- IdActor +1,
78.            Name<- 'FrenshPartner' )
79. }
```

FIGURE 3 – Règle 3 : transformation OntoMCT-to-OntoPivot

Annexe B

L'annexe suivante présente l'ensemble des besoins utilisés pour la validation de notre travail. Nous avons utilisé des besoins appartenant a deux systèmes différents: (1) document de système de gestion de cours (CMS) et (2) système de gestion des conférences (ConfOf, CMT et EDAS). Pour chaque un des deux systèmes, nous avons utilisé trois sources de besoins (besoins en langue anglais, besoins en langue française et besoins en langue espagnole) que nous présentons dans ce qui suit:

1 Les besoins du CMS document

Nous avons considéré un ensemble de besoins pertinents appartenant au CMS document, pour les trois sources utilisées(source de besoins en langue française, source de besoins en langue anglaise et source de besoins en langue espagnole) dans notre étude de cas. Dans ce qui suit, nous présentons les trois sources de besoins:

1. **Sources de besoins en langue anglais** : pour cette source nous avons utilisé les besoins extraits du système de gestion des cours en langue anglais, que nous prèsentons dans ce qui suit:
   - **B1** : The system shall allow students to create an account
   - **B2** : The system shall allow students to connect to account
   - **B3** : The system shall allow students to register in courses
   - **B4** : The system shall be able to stock static course information
   - **B5** : The system shall provide dynamic course information
   - **B6** : The system shall be able to store dynamic course information
   - **B7** : The system shall be able to provide course information
   - **B8** : The system shall provide a messaging system
   - **B9** : The system shall be able to store static course information
   - **B10** : The system shall be able to provide dynamic course information
   - **B11** : The system shall provide a messaging system
   - **B12** : The system shall enable students to subscribe/unsubscribe to exams
   - **B13** : The system shall desable students to subscribe/unsubscribe to exams
   - **B14** : The system shall let students to subscribe/unsubscribe to exams
   - **B15** : The system shall be able to let students to upload files
   - **B16** : The system shall be able to let students to Manage files
   - **B17** : The system shall allow students to create teams
   - **B18** : The system shall be able to enable students to upload files
   - **B19** : The system shall allow lecturers to create teams
   - **B20** : The system shall allow students to change their password
   - **B21** : The system shall allow students to manage their password

- **B22** : The system shall allow lecturers to create courses
- **B23** : The system shall allow lecturers to add courses
- **B24** : The system shall allow lecturers to create entirely new courses
- **B25** : The system shall allow lecturers to create entirely database courses
- **B26** : The system shall allow lecturers to recreate course (copied from a previous period)
- **B27** : The system shall allow readers to manage course
- **B28** : The system shall enable lecturers to administer grades (insert, update, ...etc)
- **B29** : The system shall enable lecturers to insert grades
- **B30** : The system shall enable readers to manage grades
- **B31** : The system shall allow lecturers to manage static course information
- **B32** : The system shall authorize lecturers to control static course information
- **B33** : The system shall authorize lecturers to control static course information
- **B34** : The system shall allow readers to manage dynamic course information
- **B35** : The system shall allow readers to manage dynamic course information
- **B36** : The system shall allow lecturers to post news messages
- **B37** : The system shall allow lecturers to send news messages
- **B38** : The system shall permit lecturers to send news messages
- **B39** : The system shall allow lecturers to manage the archive
- **B40** : The system shall let readers to manage the archive
- **B41** : The system shall allow only lecturers to administer student teams
- **B42** : The system shall allow only readers to manage student teams
- **B43** : The system shall provide grade statistics
- **B44** : The system shall give grade statistics
- **B45** : The system shall give grade
- **B46** : The system shall be easily extensible
- **B47** : The system shall be easily inextensible
- **B48** : The system shall be easily testable
- **B49** : The system shall be easily untestable
- **B50** : The system shall be scalable
- **B51** : The system shall be unscalable
- **B52** : The system shall be easily maintainable
- **B53** : The system shall be easily maintainable
- **B54** : The system shall allow only the administration to administer courses
- **B55** : The system shall allow only the manager to manage courses
- **B56** : The system shall allow only the administration to create new courses
- **B57** : The system shall allow only the administration to manage courses
- **B58** : The system shall allow only the administration to delete courses
- **B59** : The system shall let only the administration to delete courses
- **B60** : The system shall allow only the administration to update static course information
- **B61** : The system shall allow only the administration to modernise static course infor-

mation
- **B62** : The system shall allow the administration to enter lecturer information
- **B63** : The system shall permit the administration to enter lecturer information
- **B64** : The system shall allow the administration to calculate grade statistics
- **B65** : The system shall let the administration to calculate grade statistics
- **B66** : The system shall allow the administration to calculate grade statistics
- **B67** : The system shall disallow the administration to calculate grade statistics
- **B68** : The system shall allow the administration to calculate grade statistics
- **B69** : The system shall authorize the administration to calculate grade statistics
- **B70** : The system shall permit the administration to calculate grade statistics
- **B71** : The system shall allow the administration to calculate grade statistics
- **B72** : The system shall disauthorize the administration to calculate grade statistics
- **B73** : The system shall allow the department to create course
- **B74** : The system shall produce a messaging system
- **B75** : The system shall let department to create statistic course
- **B76** : The system shall authorise the department to create statistic course
- **B77** : The system shall allow readers to create courses
- **B78** : The system shall disallow readers to administer courses
- **B79** : The system shall allow the department to manage course
- **B80** : The system shall allow the department to create course
- **B81** : The system shall disallow the department to create course
- **B82** : The system shall be able to represente statistic course information
- **B83** : The system shall refuse the department to create statistic course
- **B84** : The system shall allow readers to create courses
- **B85** : The system shall disallow readers to manage courses
- **B86** : The system shall permit the department to manage courses
- **B87** : The system shall allow the department to create courses
- **B88** : The system shall disallow the department to create courses
- **B89** : The system shall be able to represente statistic courses information
- **B90** : The system shall authorise the department to create statistic courses
- **B91** : The system shall allow readers to create courses
- **B92** : The system shall disallow readers to administer courses
- **B93** : The system shall allow the department to manage courses
- **B94** : The system shall permit the department to create courses
- **B95** : The system shall disallow the department to create courses
- **B96** : The system shall be able to represente statistic courses information
- **B97** : The system shall authorise the department to create statistic courses
- **B98** : The system shall allow readers to create courses
- **B99** : The system shall disallow readers to administer courses
- **B100** : The system shall refuse the department to manage courses

2. **Sources de besoins en langue française** : pour cette source nous avons utilisé les besoins extraits du système de gestion des cours en langue française, que nous prèsentons dans ce qui suit:

- **B1** : Le système doit permettre aux étudiants de créer un compte
- **B2** : Le système doit permettre aux étudiants de se connecter au compte
- **B3** : Le système doit permettre aux étudiants de s'inscrire dans des cours
- **B4** : Le système doit pouvoir stocker des informations sur les cours statiques
- **B5** : Le système doit fournir des informations de cours dynamiques
- **B6** : Le système doit pouvoir stocker des informations sur les cours dynamiques
- **B7** : Le système doit pouvoir fournir des informations sur les cours
- **B8** : Le système doit fournir un système de messagerie
- **B9** : Le système doit pouvoir stocker des informations sur les cours statiques
- **B10** : Le système doit pouvoir fournir des informations sur les cours dynamiques
- **B11** : Le système doit fournir un système de messagerie
- **B12** : Le système doit permettre aux étudiants de s'abonner / se désabonner aux examens
- **B13** : Le système doit désactiver les étudiants pour s'abonner / se désabonner aux examens
- **B14** : Le système doit permettre aux étudiants de s'abonner/se désabonner aux examens
- **B15** : Le système doit permettre aux élèves de télécharger des fichiers
- **B16** : Le système doit permettre aux élèves de gérer les fichiers
- **B17** : Le système doit permettre aux étudiants de créer des équipes
- **B18** : Le système doit pouvoir permettre aux étudiants de télécharger des fichiers
- **B19** : Le système doit permettre aux conférenciers de créer des équipes
- **B20** : Le système doit permettre aux étudiants de modifier leur mot de passe
- **B21** : Le système doit permettre aux étudiants de gérer leur mot de passe
- **B22** : Le système doit permettre aux conférenciers de créer des cours
- **B23** : Le système doit permettre aux conférenciers d'ajouter des cours
- **B24** : Le système doit permettre aux conférenciers de créer des cours entièrement nouveaux
- **B25** : Le système doit permettre aux conférenciers de créer des cours entièrement sur base de données
- **B26** : Le système doit permettre aux enseignants de recréer un cours (copié d'une période antérieure)
- **B27** : Le système doit permettre aux lecteurs de gérer le cours
- **B28** : Le système doit permettre aux conférenciers d'administrer les notes (insérer, mettre à jour, ... etc)
- **B29** : Le système doit permettre aux conférenciers d'insérer des notes
- **B30** : Le système doit permettre aux lecteurs de gérer les notes

- **B31** : Le système doit permettre aux conférenciers de gérer l'information statique sur les cours
- **B32** : Le système autorise les conférenciers à contrôler les informations statiques
- **B33** : Le système autorise les conférenciers à contrôler les informations statiques
- **B34** : Le système doit permettre aux lecteurs de gérer les informations de cours dynamiques
- **B35** : Le système doit permettre aux lecteurs de gérer les informations de cours dynamiques
- **B36** : Le système doit permettre aux conférenciers de publier des messages d'actualité
- **B37** : Le système doit permettre aux conférenciers d'envoyer des messages d'actualité
- **B38** : Le système permet aux conférenciers d'envoyer des messages d'actualité
- **B39** : Le système doit permettre aux conférenciers de gérer les archives
- **B40** : Le système doit permettre aux lecteurs de gérer les archives
- **B41** : Le système ne doit autoriser que les enseignants à administrer des équipes d'étudiants
- **B42** : Le système doit permettre aux lecteurs seulement de gérer les équipes étudiantes
- **B43** : Le système doit fournir des statistiques de qualité
- **B44** : Le système doit donner des statistiques de qualité
- **B45** : Le système doit donner un grade
- **B46** : Le système doit être facilement extensible
- **B47** : Le système doit être facilement inextensible
- **B48** : Le système doit être facilement testable
- **B49** : Le système ne peut être facilement non testable
- **B50** : Le système doit être évolutif
- **B51** : Le système doit être non évolutif
- **B52** : Le système doit être facilement maintenable
- **B53** : Le système doit être facilement maintenable
- **B54** : Le système ne permet qu'à l'administration d'administrer des cours
- **B55** : Le système ne doit permettre que le gestionnaire de gérer les cours
- **B56** : Le système ne permet qu'à l'administration de créer de nouveaux cours
- **B57** : Le système ne permet qu'à l'administration de gérer les cours
- **B58** : Le système ne doit permettre à l'administration de supprimer des cours
- **B59** : Le système ne doit autoriser l'administration qu'à supprimer les cours
- **B60** : Le système doit autoriser uniquement l'administration à mettre à jour les informations statiques
- **B61** : Le système ne permet qu'à l'administration de moderniser l'information sur les cours statiques
- **B62** : Le système doit permettre à l'administration d'entrer dans l'information du conférencier
- **B63** : Le système doit permettre à l'administration d'entrer dans l'information du confé-

rencier
- **B64** : Le système doit permettre à l'administration de calculer les statistiques de grade
- **B65** : Le système doit permettre à l'administration de calculer les statistiques de grade
- **B66** : Le système doit permettre à l'administration de calculer les statistiques de grade
- **B67** : Le système interdit à l'administration de calculer les statistiques de grade
- **B68** : Le système doit permettre à l'administration de calculer les statistiques de grade
- **B69** : Le système autorise l'administration à calculer les statistiques de grade
- **B70** : Le système doit permettre à l'administration de calculer les statistiques de grade
- **B71** : Le système doit permettre à l'administration de calculer les statistiques de grade
- **B72** : Le système désiste l'administration pour calculer les statistiques de grade
- **B73** : Le système doit permettre au département de créer un cours
- **B74** : Le système doit produire un système de messagerie
- **B75** : Le système doit laisser le ministère créer un cours statistique
- **B76** : Le système autorise le département à créer un cours statistique
- **B77** : Le système doit permettre aux lecteurs de créer des cours
- **B78** : Le système interdit aux lecteurs d'administrer des cours
- **B79** : Le système doit permettre au département de gérer son cours
- **B80** : Le système doit permettre au département de créer un cours
- **B81** : Le système interdit au ministère de créer un cours
- **B82** : Le système doit pouvoir représenter les informations du cours statistique
- **B83** : Le système doit refuser au ministère de créer un cours statistique
- **B84** : Le système doit permettre aux lecteurs de créer des cours
- **B85** : Le système interdit aux lecteurs de gérer les cours
- **B86** : Le système doit permettre au département de gérer les cours
- **B87** : Le système doit permettre au ministère de créer des cours
- **B88** : Le système interdit au ministère de créer des cours
- **B89** : Le système doit pouvoir représenter l'information sur les cours statistiques
- **B90** : Le système autorise le ministère à créer des cours statistiques
- **B91** : Le système doit permettre aux lecteurs de créer des cours
- **B92** : Le système interdit aux lecteurs d'administrer des cours
- **B93** : Le système doit permettre au département de gérer les cours
- **B94** : Le système doit permettre au ministère de créer des cours
- **B95** : Le système interdit au ministère de créer des cours
- **B96** : Le système doit pouvoir représenter l'information sur les cours statistiques
- **B97** : Le système autorise le ministère à créer des cours statistiques
- **B98** : Le système doit permettre aux lecteurs de créer des cours
- **B99** : Le système interdit aux lecteurs d'administrer des cours
- **B100** : Le système doit refuser au ministère de gérer les cours

3. **Sources de besoins en langue espagnol** : pour cette source nous avons utilisé les besoins

extraits du système de gestion des cours en langue espagnol, que nous prèsentons dans ce qui suit:

- **B1** : El sistema permitirá a los estudiantes crear una cuenta
- **B2** : El sistema permitirá a los estudiantes conectarse a la cuenta
- **B3** : El sistema permitirá que los estudiantes se inscriban en cursos
- **B4** : El sistema deberá poder almacenar la información estática del curso
- **B5** : El sistema proporcionará información dinámica del curso
- **B6** : El sistema deberá ser capaz de almacenar información dinámica del curso
- **B7** : El sistema deberá poder proporcionar información del curso
- **B8** : El sistema proporcionará un sistema de mensajería
- **B9** : El sistema deberá poder almacenar información estática del curso
- **B10** : El sistema deberá ser capaz de proporcionar información dinámica del curso
- **B11** : El sistema proporcionará un sistema de mensajería
- **B12** : El sistema permitirá a los estudiantes suscribirse/cancelar los exámenes
- **B13** : El sistema deshabilitará a los estudiantes a suscribirse o cancelarse a los exámenes
- **B14** : El sistema permitirá que los estudiantes suscriban / se suscriban a los exámenes
- **B15** : El sistema debe permitir que los estudiantes carguen archivos
- **B16** : El sistema debe permitir que los estudiantes manejen archivos
- **B17** : El sistema permitirá a los estudiantes crear equipos
- **B18** : El sistema debe permitir a los estudiantes cargar archivos
- **B19** : El sistema permitirá a los profesores crear equipos
- **B20** : El sistema permitirá que los estudiantes cambien su contraseña
- **B21** : El sistema permitirá a los estudiantes administrar su contraseña
- **B22** : El sistema permitirá a los profesores crear cursos
- **B23** : El sistema permitirá a los profesores añadir cursos
- **B24** : El sistema permitirá a los profesores crear cursos completamente nuevos
- **B25** : El sistema permitirá a los profesores crear cursos de base de datos enteramente
- **B26** : El sistema permitirá a los profesores recrear el curso (copiado de un período anterior)
- **B27** : El sistema permitirá a los lectores
- **B28** : El sistema permitirá a los profesores administrar las calificaciones (insertar, actualizar, ... etc)
- **B29** : El sistema permitirá a los profesores insertar calificaciones
- **B30** : El sistema permitirá a los lectores administrar los grados
- **B31** : El sistema permitirá a los profesores administrar la información estática del curso
- **B32** : El sistema autorizará a los profesores a controlar la información estática del curso
- **B33** : El sistema autorizará a los profesores a controlar la información estática del curso
- **B34** : El sistema permitirá a los lectores gestionar la información dinámica del curso
- **B35** : El sistema permitirá a los lectores gestionar la información dinámica del curso
- **B36** : El sistema permitirá a los profesores publicar mensajes de noticias

- **B37** : El sistema permitirá a los conferenciantes enviar mensajes de noticias
- **B38** : El sistema permitirá a los profesores enviar mensajes de noticias
- **B39** : El sistema permitirá a los profesores administrar el archivo
- **B40** : El sistema permitirá a los lectores administrar el archivo
- **B41** : El sistema permitirá que sólo los profesores administren equipos de estudiantes
- **B42** : El sistema permitirá que sólo los lectores gestionen equipos de estudiantes
- **B43** : El sistema proporcionará estadísticas de grado
- **B44** : El sistema proporcionará estadísticas de grado
- **B45** : El sistema dará el grado
- **B46** : El sistema deberá ser fácilmente extensible
- **B47** : El sistema será fácilmente inextensible
- **B48** : El sistema deberá ser fácilmente comprobable
- **B49** : El sistema será fácilmente no comprobable
- **B50** : El sistema será escalable
- **B51** : El sistema deberá ser inescalable
- **B52** : El sistema deberá ser fácilmente
- **B53** : El sistema deberá ser fácilmente
- **B54** : El sistema sólo permitirá a la administración administrar cursos
- **B55** : El sistema sólo permitirá que el administrador administre cursos
- **B56** : El sistema sólo permitirá a la administración crear nuevos cursos
- **B57** : El sistema sólo permitirá a la administración administrar cursos
- **B58** : El sistema sólo permitirá a la administración suprimir cursos
- **B59** : El sistema permitirá que sólo la administración elimine cursos
- **B60** : El sistema sólo permitirá que la administración actualice la información estática del curso
- **B61** : El sistema sólo permitirá a la administración modernizar la información estática del curso
- **B62** : El sistema permitirá a la administración introducir la información del conferenciante
- **B63** : El sistema permitirá a la administración introducir la información del conferenciante
- **B64** : El sistema permitirá a la administración calcular las estadísticas de grado
- **B65** : El sistema deberá permitir a la administración calcular las estadísticas de grado
- **B66** : El sistema permitirá a la administración calcular las estadísticas de grado
- **B67** : El sistema no permitirá a la administración calcular las estadísticas de grado
- **B68** : El sistema permitirá a la administración calcular las estadísticas de grado
- **B69** : El sistema autorizará a la administración a calcular las estadísticas de grado
- **B70** : El sistema permitirá a la administración calcular las estadísticas de grado
- **B71** : El sistema permitirá a la administración calcular las estadísticas de grado
- **B72** : El sistema desautorizará a la administración para calcular las estadísticas de grado

- **B73** : El sistema permitirá al departamento crear cursos
- **B74** : El sistema producirá un sistema de mensajería
- **B75** : El sistema permitirá que el departamento cree cursos de estadística
- **B76** : El sistema autorizará al departamento a crear un curso de estadística
- **B77** : El sistema permitirá a los lectores crear cursos
- **B78** : El sistema no permitirá a los lectores administrar cursos
- **B79** : El sistema permitirá al departamento administrar el curso
- **B80** : El sistema permitirá al departamento crear cursos
- **B81** : El sistema no permitirá al departamento crear un curso
- **B82** : El sistema deberá poder representar la información estadística del curso
- **B83** : El sistema rechazará que el departamento cree cursos de estadística
- **B84** : El sistema permitirá a los lectores crear cursos
- **B85** : El sistema no permitirá a los lectores gestionar cursos
- **B86** : El sistema permitirá que el departamento administre cursos
- **B87** : El sistema permitirá al departamento crear cursos
- **B88** : El sistema no permitirá al departamento crear cursos
- **B89** : El sistema deberá poder representar información de cursos estadísticos
- **B90** : El sistema autorizará al departamento a crear cursos de estadística
- **B91** : El sistema permitirá a los lectores crear cursos
- **B92** : El sistema no permitirá a los lectores administrar cursos
- **B93** : El sistema permitirá al departamento administrar cursos
- **B94** : El sistema permitirá al departamento crear cursos
- **B95** : El sistema no permitirá al departamento crear cursos
- **B96** : El sistema deberá poder representar información de cursos estadísticos
- **B97** : El sistema autorizará al departamento a crear cursos de estadística
- **B98** : El sistema permitirá a los lectores crear cursos
- **B99** : El sistema no permitirá a los lectores administrar cursos
- **B100** : El sistema rechazará que el departamento administre cursos

2 Les besoins du systèmes de gestion des conférences:

Nous avons considéré un ensemble de besoins pertinents appartenant au systèmes de gestion de conférences (EDAS, ConfOf, CMT), pour les trois sources utilisées (source de besoins en langue française, source de besoins en langue anglaise et source de besoins en langue espagnole) dans notre étude de cas. Dans ce qui suit, nous présentons les trois sources de besoins:

1. **Sources de besoins en langue anglais** : pour cette source nous avons utilisé les besoins extraits du système de gestion des conférences EDAT en langue anglaise, que nous présentons dans ce qui suit:

- **B01** : The system shall allow the author to receive the password;
- **B02** : The author returns to the system;
- **B03** : The system shall allow the author to login account;
- **B04** : The system shall allow the author to enter email address;
- **B05** : The system shall allow the author to enter password;
- **B06** : The system must allow the user to register at the conference as a participant;
- **B07** : The system must allow the user to create an account;
- **B08** : The system must allow the user to select the "New Registration" link;
- **B09** : The system must allow the user to fill in the web form;
- **B10** : The system must allow the user to enter its full name;
- **B11** : The system must allow the user to enter his address;
- **B12** : The system must allow the user to enter his e-mail address;
- **B13** : The system will send password to author;
- **B14** : The system must allow the user to use an account for multiple submissions;
- **B15** : The system shall allow the author to clicks on the 'New User' button;
- **B16** : The system must allow the member PC to use the account for paper submission;
- **B17** : The system must allow the examiner to use the account for paper submission;
- **B18** : The system must allow the user to maintain an account
- **B19** : The system must allow the user to log in an account;
- **B20** : The system must allow the user to enter the user name;
- **B21** : The system must allow the user to enter the password;
- **B22** : The system must allow the user to remember the password;
- **B23** : The user forgot his password;
- **B24** : The system must allow the user to obtain a new password;
- **B25** : The system shall allow the author to clicks on the 'New User' link;
- **B26** : The system shall allow the author to fill in the form;
- **B27** : The system shall allow the chair to submit papers; (at any time, even after track deadlines have passed);
- **B28** : The system shall allow the chair to manage papers (but if they are looking at their own papers, their privileges are reduced to that of normal authors).
- **B29** : The system must allow the user to connect an account;
- **B30** : The system will allow the user to enter the email address;
- **B31** : The system will allow the user to enter the password;
- **B32** : The user to remember the password;
- **B33** : User forgot password;
- **B34** : The system should allow the user to access the main page of the CMT conference;
- **B35** : The system should allow the user to click on the link "Change password";
- **B36** : The system will send an email to the user's email address;
- **B37** : The system will send you an email a link to the user's email address;
- **B38** : The system will allow the user to set the password;

- **B39** : The system shall allow the Track chairs to assign reviewers;
- **B40** : The system shall allow the Track chairs to notify reviewers;
- **B41** : The system shall allow the Track chairs to delete reviewers;
- **B42** : The system shall allow the Track chairs to view reviewers;
- **B43** : The author haven't an account
- **B44** : The system shall allow the author to create an account
- **B45** : The system shall allow the author to login an account
- **B46** : The system shall allow the author to register the paper
- **B47** : The system shall allow the author to add authors
- **B48** : The system shall allow the author to submit the paper
- **B49** : The authors have an account
- **B50** : The system shall allow the authors to login an account
- **B51** : The system shall allow the authors to register the paper
- **B52** : The system shall allow the authors to add authors
- **B53** : The system shall allow the authors to submit the paper
- **B54** : The system shall allow the authors to check account;
- **B55** : The system shall allow the authors to enter email address;
- **B56** The system respond to authors;
- **B57** : The author has an account;
- **B58** : The system shall allow the author to login account;
- **B59** : The system shall allow the author to enter email address;
- **B60** : The system shall allow the author to enter password;
- **B61** : The author remember password;
- **B62** : The author forget password;
- **B63** : The system shall allow the author to leave password field blank;
- **B64** : The system will send password to author;
- **B65** : The system shall allow the author receives the password;
- **B66** : The author hasn't an account;
- **B67** : The system can define a presentation;
- **B68** : The system can manage presentation;
- **B69** : The system shall allow the users to add names;
- **B70** : The system shall allow the users to edit names;
- **B71** : The system shall allow the users to delete names;
- **B72** : The system can create sessions (presentations);
- **B73** : The system can manage sessions (sessions for presentations);

2. **Sources de besoins en langue espagnol** : pour cette source nous avons utilisé les besoins extraits du système de gestion des conférences CMT en langue espagnole, que nous présentons dans ce qui suit:

- **B01** : El sistema permitirá que el autor reciba la contraseña;

- **B02** : El autor vuelve al sistema;
- **B03** : El sistema permitirá al autor acceder a la cuenta;
- **B04** : El sistema permitirá que el autor introduzca una dirección de correo electrónico;
- **B05** : El sistema permitirá al autor ingresar contraseña;
- **B06** : El sistema debe permitir al usuario registrarse en la conferencia como participante;
- **B07** : El sistema debe permitir al usuario crear una cuenta;
- **B08** : El sistema debe permitir al usuario seleccionar el enlace "Nuevo registro";
- **B09** : El sistema debe permitir al usuario rellenar el formulario web;
- **B10** : El sistema debe permitir que el usuario introduzca su nombre completo;
- **B11** : El sistema debe permitir al usuario ingresar su dirección;
- **B12** : El sistema debe permitir que el usuario introduzca su dirección de correo electrónico;
- **B13** : El sistema enviará la contraseña al autor;
- **B14** : El sistema debe permitir al usuario usar una cuenta para múltiples envíos;
- **B15** : El sistema permitirá que el autor haga clic en el botón "Nuevo usuario";
- **B16** : El sistema debe permitir que el PC miembro utilice la cuenta para la presentación de papel;
- **B17** : El sistema debe permitir al examinador utilizar la cuenta para la presentación de documentos;
- **B18** : El sistema debe permitir al usuario mantener una cuenta;
- **B19** : El sistema debe permitir al usuario iniciar sesión en una cuenta;
- **B20** : El sistema debe permitir al usuario ingresar el nombre de usuario;
- **B21** : El sistema debe permitir que el usuario ingrese la contraseña;
- **B22** : El sistema debe permitir al usuario recordar la contraseña;
- **B23** : El usuario olvidó su contraseña;
- **B24** : El sistema debe permitir al usuario obtener una nueva contraseña;
- **B25** : El sistema permitirá que el autor haga clic en el enlace "Nuevo usuario";
- **B26** : El sistema permitirá que el autor complete el formulario;
- **B27** : El sistema permitirà que el presidente presente los documentos; (En cualquier momento, incluso después de que los plazos de seguimiento hayan pasado;
- **B28** : El sistema permitirà que el presidente administre documentos (pero si estàn buscando sus propios documentos, sus privilegios se reducen a los de los autores normales);
- **B29** : El sistema deberá permitir al usuario conectarse una cuenta;
- **B30** : El sistema permitirá al usuario introducir la direcciõn de correo electrõnico;
- **B31** : El sistema permitirá que el usuario introduzca la contraseña;
- **B32** : El usuario a recordar la contraseña;
- **B33** : El usuario olvidó la contraseña;
- **B34** : El sistema deberá permitir al usuario acceder a la página principal de la conferencia CMT.
- **B35** : El sistema deberá permitir que el usuario haga clic en el enlace "Cambiar contra-

seña".
- **B36** : El sistema enviará un correo electrónico a la dirección de correo electrònico del usuario.
- **B37** : El sistema le enviará un correo electrónico un enlace a dirección de correo electrónico del usuario.
- **B38** : El sistema permitirá al usuario configurar la contraseña.
- **B39** : El sistema permitirá que las sillas de pista asignen revisores;
- **B40** : El sistema permitirá a las sillas de pista notificar a los revisores;
- **B41** : El sistema permitirá que las sillas de pista eliminen a los revisores;
- **B42** : El sistema permitirá que las sillas de pista vean a los revisores;
- **B43** : El autor no tiene una cuenta;
- **B44** : El sistema permitirá al autor crear una cuenta;
- **B45** : El sistema permitirá al autor acceder a una cuenta;
- **B46** : El sistema permitirá que el autor registre el documento;
- **B47** : El sistema permitirá que el autor añada autores;
- **B48** : El sistema permitirá al autor presentar el documento;
- **B49** : Los autores tienen una cuenta;
- **B50** : El sistema permitirá a los autores acceder a una cuenta;
- **B51** : El sistema permitirá a los autores registrar el documento;
- **B52** : El sistema permitirá a los autores añadir autores;
- **B53** : El sistema permitirá a los autores presentar el documento;
- **B54** : El sistema permitirá a los autores revisar la cuenta;
- **B55** : El sistema permitirá a los autores introducir la dirección de correo electrónico;
- **B56** : El sistema responde a los autores;
- **B57** : El autor tiene una cuenta;
- **B58** : El sistema permitirá al autor acceder a la cuenta;
- **B59** : El sistema permitirá que el autor introduzca una dirección de correo electrónico;
- **B60** : El sistema permitirá al autor ingresar contraseña;
- **B61** : El autor recuerda la contraseña;
- **B62** : El autor olvidó la contraseña;
- **B63** : El sistema permitirá que el autor deje el campo de contraseña en blanco;
- **B64** : El sistema enviará la contraseña al autor;
- **B65** : El sistema permitirá que el autor reciba la contraseña;
- **B66** : El autor no tiene una cuenta;
- **B67** : El sistema poder definir una presentación;
- **B68** : El sistema poder administrar presentación;
- **B69** : El sistema poder crear sesiones (sessions for presentations);
- **B70** : El sistema poder administrar sesiones (sessions for presentations);

3. **Sources de besoins en langue française** : pour cette source nous avons utilisé les besoins

extraits du système de gestion des conférences confOf en langue française, que nous prèsentons dans ce qui suit:

- **B01** : Le système doit permettre à l'auteur de recevoir le mot de passe;
- **B02** : L'auteur retourne au système;
- **B03** : Le système doit permettre à l'auteur de se connecter au compte;
- **B04** : Le système doit autoriser l'auteur à entrer une adresse èlectronique;
- **B05** : Le système doit autoriser l'auteur à entrer son mot de passe;
- **B06** : Le système doit permettre à l'utilisateur d'enregistrer à la confèrence en tant que participant;
- **B07** : Le système doit permettre à l'utilisateur de crèer un compte;
- **B08** : Le système doit permettre à l'utilisateur de sélectionner le lien "Inscription Nouveau";
- **B09** : Le système doit permettre à l'utilisateur de remplir le formulaire web;
- **B10** : Le système doit permettre à l'utilisateur d'entrer son nom complet;
- **B11** : Le système doit permettre à l'utilisateur d'entrer son adresse;
- **B12** : Le système doit permettre à l'utilisateur d'entrer son adresse e-mail;
- **B13** : Le système enverra un mot de passe à l'auteur;
- **B14** : Le système doit permettre à l'utilisateur d'utiliser un compte pour plusieurs soumissions;
- **B15** : Le système doit permettre à l'auteur de cliquer sur le bouton 'Nouvel utilisateur';
- **B16** : Le système doit permettre au PC membre d'utiliser le compte pour la soumission de papier;
- **B17** : Le système doit permettre l'examinateur d'utiliser le compte pour la soumission de papier;
- **B18** : Le système doit permettre à l'utilisateur de tenir un compte
- **B19** : Le système doit permettre à l'utilisateur de se connecter un compte;
- **B20** : Le système doit permettre à l'utilisateur d'entrer le nom d'utilisateur;
- **B21** : Le système doit permettre à l'utilisateur d'entrer le mot de passe;
- **B22** : Le système doit permettre à l'utilisateur de se rappeler le mot de passe;
- **B23** : L'utilisateur oublié son mot de passe;
- **B24** : Le système doit permettre à l'utilisateur d'obtenir un nouveau mot de passe;
- **B25** : Le système doit permettre à l'auteur de cliquer sur le lien 'Nouvel utilisateur';
- **B26** : Le système doit permettre à l'auteur de remplir le formulaire;
- **B27** : Le système doit permettre au prèsident de soumettre des documents; (à tout moment, même après la fin des délais de la piste);
- **B28** : Le systmè doit permettre au président de gérer les documents (mais s'ils regardent leurs propres documents, leurs privilèges sont réduits à ceux des auteurs normaux);
- **B29** : Le système doit permettre à l'utilisateur de connecter un compte;
- **B30** : Le système permettra à l'utilisateur d'entrer l'adresse e-mail;
- **B31** : Le système permettra à l'utilisateur d'entrer le mot de passe;

- **B32** : L'utilisateur se souvient du mot de passe;
- **B33** : Mot de passe oublié par l'utilisateur;
- **B34** : Le système devrait permettre à l'utilisateur d'accéder à la page principale de la conférence CMT;
- **B35** : Le système devrait permettre à l'utilisateur de cliquer sur le lien "Modifier le mot de passe";
- **B36** : Le système enverra un courrier électronique à l'adresse électronique de l'utilisateur;
- **B37** : Le système vous enverra un mail un lien vers l'adresse électronique de l'utilisateur;
- **B38** : Le système permettra à l'utilisateur de définir le mot de passe;
- **B39** : Le système doit permettre aux chaises de piste d'assigner des réviseurs;
- **B40** : Le système doit permettre aux chaises de piste d'aviser les examinateurs;
- **B41** : Le système doit permettre aux chaises de piste de supprimer les examinateurs;
- **B42** : Le système doit permettre aux chaises de piste de voir les critiques;
- **B43** : L'auteur n'a pas de compte;
- **B44** : Le système doit permettre à l'auteur de créer un compte;
- **B45** : Le système doit permettre à l'auteur de se connecter à un compte;
- **B46** : Le système doit permettre à l'auteur d'enregistrer le document;
- **B47** : Le système doit permettre à l'auteur d'ajouter des auteurs;
- **B48** : Le système doit permettre à l'auteur de soumettre le document;
- **B49** : Les auteurs ont un compte;
- **B50** : Le système doit permettre aux auteurs de se connecter à un compte;
- **B51** : Le système doit permettre aux auteurs de se connecter à un compte;
- **B52** : Le système doit permettre aux auteurs d'ajouter des auteurs;
- **B53** : Le système doit permettre aux auteurs de soumettre le document;
- **B54** : Le système doit permettre aux auteurs de vérifier le compte;
- **B55** : Le système doit permettre aux auteurs de saisir une adresse électronique;
- **B56** : Le système rèpond aux auteurs;
- **B57** : L'auteur a un compte;
- **B58** : Le système doit permettre à l'auteur de se connecter au compte;
- **B59** : Le système doit autoriser l'auteur à entrer une adresse électronique;
- **B60** : Le système doit autoriser l'auteur à entrer son mot de passe;
- **B61** : L'auteur se souvient du mot de passe;
- **B62** : L'auteur oublie le mot de passe;
- **B63** : Le système doit permettre à l'auteur de laisser le champ du mot de passe vide;
- **B64** : Le système enverra un mot de passe à l'auteur;
- **B65** : Le système doit permettre à l'auteur de recevoir le mot de passe;
- **B66** : L'auteur n'a pas de compte;
- **B67** : Le système peut définir une présentation;

- **B68** : Le système peut gérer la prśentation;
- **B69** : Le système doit permettre aux utilisateurs d'ajouter des noms;
- **B70** : Le système doit permettre aux utilisateurs d'éditer des noms;
- **B71** : Le système doit permettre aux utilisateurs de supprimer des noms;
- **B72** : Le système peut créer des sessions (présentations);
- **B73** : Le système peut gérer les sessions (sessions pour les présentations);

Table des figures

1.1 Architecture de médiation 5
1.2 Architecture d'un dépôt traditionnel/décisionnel 6
1.3 Evolution des Dépôts de Besoins 7
1.4 Vers un dépôt décisionnel de besoins 9
1.5 Contributions 13

2.1 Étapes du processus de l'ingénierie des besoins [27]. 26
2.2 Le modèle orienté but [27] 29
2.3 Le modèle des cas d'utilisation [27]. 30
2.4 Le modèle conceptuel de traitements [27]. 31
2.5 Différents niveaux d'intégration [59]. 33
2.6 Exemple d'un cube multidimensionnel des *Ventes* 36
2.7 Exemple d'une modélisation d'ED en étoile [11] 37
2.8 Exemple d'une modélisation d'ED en flocon de neige [11] 38
2.9 Modèle en oignon [56] 41
2.10 Architecture BDBO type I 45
2.11 Architecture BDBO type II 45
2.12 Architecture BDBO type III 46
2.13 Les structures d'ontologies conceptuelles dans les systèmes d'intégration [90] . 47
2.14 Exemple du Matching de deux ontologies [54] 50
2.15 L'alignement généré par l'exemple [54] 50
2.16 Processus d'alignement d'ontologies [54] 51

2.17 Les étapes du processus du matching d'ontologies [52] 52

3.1 Méta-modèle des besoins formalisé en UML [106] 62
3.2 Le MetaModel d'ingénierie des besoins dans '*REMM*' [149] 64
3.3 Les étapes qui permettent de produire un modèle de besoins global[32] 65
3.4 Un bref résumé l'approche '*SIREN*' [144] 71

4.1 Approche générale 86
4.2 Les étapes d'extraction de l'ontologie locale à partir de l'ontologie partagée [27] 89
4.3 Exemple d'extraction d'une ontologie locale équivalente à l'ontologie partagée 90
4.4 Exemple d'extraction d'une ontologie locale incluse à l'ontologie partagée. . . 91
4.5 Exemple d'extraction d'une ontologie locale qui englobe l'ontologie partagée . 92
4.6 Liaisons entre la couche linguistique et conceptuelle dans le modèle en oignon . 93
4.7 Système de matching ontologique 94
4.8 Un extrait de l'ontologie Univ-Bench proposée par le benchmark LUBM. . . . 102
4.9 L'architecture globale de l'ontologie *EuroWordNet* [153] 103
4.10 Étude de cas 2 : intégration d'ontologies d'un système de soumission de conférences 104
4.11 Les résultats de Matching 105

5.1 Le framework OntoMct : connexion du modèle MCT avec les modèles ontologiques 112
5.2 Le framework OntoGoal : connexion entre le modèle orienté buts et les modèles ontologiques 113
5.3 Le framework OntoUseCase : connexion du modèle des cas d'utilisation aux modèles ontologiques 115
5.4 Le méta-modèle pivot [28]. 116
5.5 Modèle pivot enrichi (MPE) 117
5.6 Exemple de graphe de précedence d'une source. 119
5.7 Enrichissement du modèle pivot par les dimensions *Time* et *Location* 120
5.8 Le schéma en étoile de l'entrepôt de besoins 122
5.9 Structure d'un système de transformation de modèles 124
5.10 Mapping des formalismes sources vers le modèle de l'entrepôt 124

5.11 Syntaxe d'une règle de transformation ATL (extraite du métamodèle de l'ATL)[42] 125
5.12 Exemple d'une règle de transformation entre les formalismes Goal et EPivot . . 126
5.13 Connexion entre les composantes de notre framework au niveau méta-modèle . 128

6.1 Application de l'opérateur STORE pour chargement des besoins 136
6.2 Le résultat d'application de l'opérateur $CONVERT$ sur les besoins sources . . 139
6.3 L'ordonnancement des tâches entre les besoins 140
6.4 Architecture de l'opérateur Identification 142
6.5 Architecture de l'opérateur Inference 142
6.6 Architecture de l'opérateur ConsistencyChecking 143
6.7 Exemple de génération de clé par l'opérateur AddSurrogatekey 144
6.8 ETL de besoins 146
6.9 Définition des règles en langage SWRL (éditeur Protégé) 151
6.10 Architecture et étapes d'exécution du mécanisme de raisonnement sur les besoins 157
6.11 Les étapes (1,2,3,4) de validation de notre approche 158
6.12 Relations détectées dans chaque framework proposé 159
6.13 Identification des relations inconsistantes. 160
6.14 évolutivité et temps d'exécution. 160
6.15 Complexité de l'algorithme d'ETL de besoins proposé 161

7.1 Schéma de l'entrepôt de besoins selon l'approche verticale 166
7.2 Schéma de l'entrepôt de besoins selon l'approche horizontale 167
7.3 Schéma de l'entrepôt de besoins selon l'approche binaire 167
7.4 Déploiement de la structure d'entrepôt sémantique de besoins fonctionnels . . . 168
7.5 Extension du méta-schéma sémantique d'Oracle avec le modèle de besoins . . 169
7.6 La BDBO OntoDB étendue par le schéma pivot de l'entrepôt 175
7.7 Le temps d'exécution et le nombre de relations inférées 181
7.8 Les relations identifiées et inférées dans l'entrepôt 182
7.9 Taux d'intégration des besoins 182
7.10 Relations détectées 183
7.11 Scalabilité et temps d'exécution 183
7.12 Architecture fonctionnelle de l'outil 185

7.13 Menu principal de notre outil 186
7.14 Visualisation des besoins d'une source en langage naturel 187
7.15 Visualisation des besoins d'une source exprimés en formalisme local (MCT) . . 187
7.16 Visualisation d'une ontologie conceptuelle d'une source 188
7.17 Visualisation d'ontologie linguistique d'une source 188
7.18 Visualisation de l'ontologie conceptuelle globale intégrée 189
7.19 Visualisation de l'ontologie linguistique globale intégrée 189
7.20 Visualisation des besoins intégrés en format cible 190
7.21 Visualisation du résultat du processus de raisonnement 190

1 Règle 1 : transformation OntoGoal-to-OntoPivot 214
2 Règle 2 : transformation OntoUcase-to-OntoPivot 215
3 Règle 3 : transformation OntoMCT-to-OntoPivot 216

Liste des tableaux

3.1 Études des travaux connexes : comparaison et analyse 75

7.1 Caractéristiques des besoins utilisés 179
7.2 Performance de l'inférence : temps et nombre d'instances 181
7.3 Composants de chaque étape de l'évaluation 182

Glossaire

ABox : Assertion Box
AFIS : Association Française pour l'Information Scientifique
AGORA : Attributed Goal-Oriented Requirements Analysis
AJAX : Asynchronous JavaScript and XML
API : Application Programming Interface
APROMORE : An advanced PROcess MOdel REpository
ATL : ATLAS Transformation Language
BD : Base de Données
BDBO : Base de Données à base ontologique
BF-OLAP : Besoins fonctionnels On-Line Analytical Processing
BU : Besoins utilisateurs
BPEL : Business Process Execution Language
BPMN : Business Process Model Notation
CC : Classe canonique
CMS : Courses Management System
CNC : Classe non canonique
CPU: Central Processing Unit
CroLOM : Cross-Lingual Ontology Matching System
DAML : DARPA Agent Markup Language
DBpedia : Data Base pedia
DL : Description Logic
DOORS : Dynamic Object Oriented Requirements System
DWQ : Data Warehouse Quality
EAI : Entreprise Application Integration
ED : Entrepôt de données
EB : Entrepôt de besoins
EBF : Entrepôt de Besoins fonctionnels
EDF : Électricité de France
EII : Entreprise Information Integration

EMF : EclipseModeling Framework
ESBF : Entrepôt Sémantique de Besoins Fonctionnels
ETL : Extract-Transform-Load
FMA : the Foundational Model of Anatomy
GaV : Global as View
GlaV : Generalized Local as View
GPU : Graphics Processing Unit
GQM : Goal-Question-Metric
GRL : Goal-oriented Requirement Language
HOLAP : Hybrid On-Line Analytical Processing
KAOS : Knowledge Acquisition in autOmated Specification
IB : Ingénierie des Besoins
IDM : Ingénierie Dirigée par les Modèles
ILI : Inter Lingual Index
iStar : Intentional STratégic Actor Relationships
IT : Information Technology
LaV : Local as View
LIAS : Laboratoire d'Informatique et d'Automatique pour les Systèmes
LISI : Laboratoire d?Informatique Scientifique et Industrielle
LUBM : Lehigh University BenchMark
LD : Logiques de description
LO : Ontologie Locale
MeSH : Medical Subject Heading
MCD : Modèle Conceptuel des Données
MCT : Modèle Conceptuel des Traitements
MDA : Model-Driven Architecture
MLD : Modèle Logique des Données
MOMIS : Mediator EnvirOnment for Multiple Information Sources
MOLAP : Multidimensional On-Line Analytical Processing
MPD : Modèle Physique des Données
MUPRET : multiperspective requirements traceability
NELL : Never ending Language Learning
NoSQL : Not only Structured Query Language
OAEI : Ontology Alignment Evaluation Initiative
OC : Ontologie Conceptuelle
OCC : Ontologies Conceptuelles Canoniques
OCL : Object Constraint Language
OCNC : Ontologies Conceptuelles Non Canoniques
OIL : Ontology Inference Layer
OL : Ontologies Linguistiques

OL++AP : On-Line Analytical Processing
OLTP : On-Line Transaction Processing
OMG : Object Management Group
Onto-ActRE : Ontology-based ACTive Requirements Engineering
OWL : Ontology Web Language
OWL-DL : Ontology Web Language Description Logics
PBW : Precision Based Weighting
PICSEL: Production d'Interfaces à base de Connaissance pour des Services en Ligne
PLIB : Parts Library - Norme ISO 13584
RDF : Ressource Description Framework
RDFS : RDF Schéma
REMM : Requirements Engineering MetaModel
RESI : Requirements engineering specification improver
RML : Requirements Modelling Language
ROLAP : Rolational On-Line Analytical Processing
SaaS : Software-as-a-Service
SCR : Software Cost Reduction
SGBD : Système de Gestion de Bases de Données
SGBDR : SGBD Relationnel
SID : Système d'Intégration de Données
SimCat : Semantic Similarity Measurement for Role-Governed Geospatial Categories
SIREN : SImple REuse of software requiremeNts
SPARQL : Sparql Protocol and RDF Query Language
SQL : Structured Query Language
SOLAP : Spatial On-Line Analytical Processing
SSB : Star Schema Benchmark
SUMO : the Suggested Upper Merged Ontology
SysML : Systems Modeling Language
SWRL : Semantic Web Rule Language
TBOX : Terminological Box
TRADE : Toolkit for Requirements And Design Engineering
TRIC : Tool for Requirements Inferencing and Consistency Checking
TSIMMIS : The Stanford-IBM Manager of Multiple Information Sources
TTC : Toute Taxe Comprise
UCM : Use Case Maps
UID : User identifier
UML : Unified Modeling Language
UMLS : Unified Medical Language System
URI : Uniform Resource Identifier
URL : Uniform Resource Locator

URN : User Requirements Notation
XML : eXtensible Markup Language
YAGO : Yet Another Great Ontology
WHIPS : WareHouse Information Project at Stanford

Résumé

Les besoins fonctionnels et non fonctionnels représentent la première brique pour la conception de toute application, logiciel, système, etc. L'ensemble des traitements associés aux besoins est établi dans le cadre de l'Ingénierie des Besoins (IB). Le processus de l'IB comporte plusieurs étapes consistant à découvrir, analyser, valider et faire évoluer l'ensemble des besoins relatifs aux fonctionnalités du système. La maturité de la communauté de l'IB lui a permis d'établir un cycle de vie bien déterminé pour le processus de besoins qui comprend les phases suivantes : l'élicitation, la modélisation, la spécification, la validation et la gestion des besoins. Une fois ces besoins validés, ils sont archivés ou stockés dans des référentiels ou des dépôts au sein des entreprises. Avec l'archivage continu des besoins, ces entreprises disposent d'une mine d'informations qu'il faudra analyser afin de reproduire les expériences cumulées et le savoir-faire acquis en réutilisant et en exploitant ces besoins pour des nouveaux projets. Proposer à ces entreprises un entrepôt dans lequel l'ensemble de besoins est stocké représente une excellente opportunité pour les analyser à des fins décisionnelles et les fouiller pour reproduire des anciennes expériences. Récemment, la communauté des processus (BPM) a émis le même besoin pour les processus. Dans cette thèse, nous souhaitons exploiter le succès des entrepôts de données pour le reproduire sur les besoins fonctionnels. Les problèmes rencontrés lors de la conception des entrepôts de données se retrouvent presque à l'identique dans le cas des besoins fonctionnels. Ces derniers sont souvent hétérogènes, surtout dans le cas d'entreprises de grande taille comme Airbus, où chaque partenaire a la liberté d'utiliser ses propres vocabulaire et formalisme pour décrire ses besoins. Pour réduire cette hétérogénéité, l'appel aux ontologies est nécessaire. Afin d'assurer l'autonomie de chaque source, nous supposons que chaque source a sa propre ontologie. Cela nécessite des efforts de matching entre les ontologies afin d'assurer l'intégration des besoins fonctionnels. Une particularité importante liée à l'entreposage de besoins réside dans le fait que ces derniers sont souvent exprimés à l'aide des formalismes semi-formels comme les use cases d'UML avec une partie textuelle importante. Afin de nous rapprocher le plus possible de ce que nous avons fait dans le cadre de l'entreposage de données, nous proposons un modèle pivot permettant de factoriser trois semi-formalismes répandus utilisés par les sources de besoins avec une description précise de ces derniers. Ce modèle pivot permettra de définir le modèle multidimensionnel de l'entrepôt de besoins, qui sera ensuite alimenté par les besoins des sources en utilisant un algorithme ETL (Extract, Transform, Load). À l'aide des mécanismes de raisonnement offerts par les ontologies et des métriques de matching, nous avons nettoyé notre entrepôt de besoins. Une fois l'entrepôt déployé, il est exploité par des outils d'analyse OLAP. Notre méthodologie est supportée par un outil couvrant l'ensemble des phases de conception et d'exploitation d'un entrepôt de besoins.

Abstract

Functional and non-functional requirements represent the first step for the design of any application, software, system, etc. All the issues associated to requirements are analyzed in the Requirements Engineering (RE) field. The RE process consists of several steps consisting of discovering, analyzing, validating and evolving the requirements related to the functionalities of the system. The RE community proposed a well-defined life-cycle for the requirements process that includes the following phases: elicitation, modeling, specification, validation and management. Once the requirements are validated, they are archived or stored in repositories in companies. With the continuous storage of requirements, companies accumulate an important amount of requirements information that needs to be analyzed in order to reproduce the previous experiences and the know-how acquired by reusing and exploiting these requirements for new projects. Proposing to these companies a warehouse in which all requirements are stored represents an excellent opportunity to analyze them for decision-making purposes. Recently, the Business Process Management Community (BPM) emitted the same needs for processes.

In this thesis, we want to exploit the success of data warehouses and to replicate it for functional requirements. The issues encountered in the design of data warehouses are almost identical in the case of functional requirements. Requirements are often heterogeneous, especially in the case of large companies such Airbus, where each partner has the freedom to use its own vocabulary and formalism to describe the requirements. To reduce this heterogeneity, using ontologies is necessary. In order to ensure the autonomy of each partner, we assume that each source has its own ontology. This requires matching efforts between ontologies to ensure the integration of functional requirements. An important feature related to the storage of requirements is that they are often expressed using semi-formal formalisms such as use cases of UML with an important textual part. In order to get as close as possible to our contributions in data warehousing, we proposed a pivot model factorizing three well-known semi-formalisms. This pivot model is used to define the multidimensional model of the requirements warehouse, which is then alimented by the sources requirements using an ETL algorithm (Extract,Transform, Load). Using reasoning mechanisms offered by ontologies and matching metrics, we cleaned up our requirements warehouse. Once the warehouse is deployed, it is exploited using OLAP analysis tools. Our methodology is supported by a tool covering all design phases of the requirements warehouse.