Utilisation de métadonnées et d'ontologie pour l'aide à l'interprétation des résultats de classification

Abdourahamane Baldé

THÈSE

présentée à

L'UNIVERSITÉ PARIS DAUPHINE
U.F.R EDDIMO

pour obtenir le titre de

DOCTEUR EN INFORMATIQUE

soutenue par
Abdourahamane Baldé

Sujet de thèse

Utilisation de métadonnées et d'ontologie pour l'aide à l'interprétation des résultats de classification automatique

Soutenue le 25 Mai 2007 devant le jury composé de :

Directeur de thèse : **Edwin Diday**, Professeur, Université Paris Dauphine
Co-Directeur de thèse : **Yves Lechevallier**, Directeur de Recherche INRIA
Rapporteur : **Monique Noirhomme-Failure**, Professeur, FUNDP (Belgique)
Rapporteur : **Sylvie Després**, Maître de conférences-HDR, Université Paris V
Examinateur : **Marie-Aude Aufaure**, Maître de conférences-HDR, Université Lyon I
Examinateur : **Brigitte Trousse**, Chargée de recherches INRIA
Examinateur : **Suzanne Pinson**, Professeur, Université Paris Dauphine

« *L'université n'entend donner aucune approbation ni improbation aux opinions émises dans les thèses : ces opinions doivent être considérées comme propres à leurs auteurs.* »

Cette thèse est dédiée à la mémoire de ma mère, éducatrice et modèle.
Tu me manques !

Remerciements

C'est ici l'occasion pour moi de remercier les personnes qui ont participé,à travers leurs conseils et leurs encouragements conseils à la réalisation de ces travaux. Sans elles ce travail n'aurait probablement jamais vu le jour.

Ainsi, je tiens à remercier vivement le professeur Edwin Diday, qui m'a permis de m'orienter vers le domaine, ô combien passionnant de l'analyse de données. Merci de l'aide et des conseils que vous m'avez apportés dans le cadre de mes travaux.

Ma gratitude et mes remerciements s'adressent à mes rapporteurs, Mme Monique Noirhomme, Professeur à l'institut d'informatique de la FUNDP (Namur-Belgique) et Mme Sylvie Després, Maître de conférences-HDR à l'université Paris 5 qui m'ont fait l'honneur d'accepter d'évaluer ce travail. Leur lecture attentive ainsi que leurs remarques pertinentes m'ont permis d'améliorer ce document.

Il m'est difficile d'imaginer que cette thèse aurait pu être menée à bien sans leur direction bienveillante. C'est pourquoi je tiens à remercier du fond de cœur Yves Lechevallier. Ses conseils et Ses idées, tant sur le plan méthodologique que scientifique, ont profondément inspiré et nourri ma réflexion au cours de mes travaux. Malgré ses multiples responsabilités, il a toujours su faire preuve de disponibilité à chaque fois que j'avais besoin de ses conseils avisés. Je n'oublierai jamais l'aide précieuse qu'il m'a apportée.
Mes remerciements vont également à Marie-Aude Aufaure, grâce à qui j'ai pu découvrir le monde des ontologies. L'écoute et la sollicitude dont elle a fait preuve à mon égard resteront à jamais gravé dans ma mémoire. J'ai pleinement profité de sa grande expérience dans le domaine de l'ontologie.
Mes gratitudes vont aussi à Brigitte Trousse, dont la rigueur scientifique qu'elle a témoignée dans ses critiques tout le long de cette thèse, m'ont aidé à clarifier mes idées et mes approches. Merci pour le temps consacré à mon travail malgré ses charges de travail.

C'est aussi l'occasion pour moi de remercier très sincèrement Mme Suzanne Pinson qui m'a fait l'honneur de participer à mon jury et d'examiner ce travail.

Il m'est impossible d'exprimer toute ma gratitude à l'équipe exceptionnelle d'AxIS, dont l'accueil et la générosité m'ont été si précieux. Merci tout particulièrement à Stéphanie Aubin pour sa gentillesse et sa disponibilité à bien des égards. Mes remerciements vont également à Fabrice Rossi et à Marc Csernel pour vos conseils au cours de cette thèse. Je ne saurais terminer sans remercier très chaleureusement mon amie Alzennyr Da Silva, merci pour les bons moments partagés ensemble.

Enfin, je remercie du fond du cœur ma famille et ma femme sans le soutien desquelles je n'aurais jamais songé à me lancer dans cette aventure. Je vous aime et vous serai éternellement reconnaissant. Je n'oublie pas non plus mes amis Damien, Baba et Wagne avec qui j'ai passé des moments inoubliables.

Résumé

De nos jours, l'importance sans cesse croissante des masses de données conduit les décideurs de divers domaines à faire appel aux techniques d'Extraction de Connaissances à partir des Données (ECD). L'une des étapes principales de ce processus est la fouille de données. Or, interpréter les résultats issus de ces méthodes de fouille de données, en particulier pour les méthodes de classification automatique, reste complexe pour les utilisateurs. Ce qui rend indispensable l'utilisation d'une nouvelle approche permettant d'aider les utilisateurs à interpréter automatiquement leurs résultats. Ainsi, nous proposons une nouvelle approche, appelée AIMFD (Approche d'Incorporation de Métadonnées dans la Fouille de Données) qui se résume principalement à intégrer des informations sémantiques (*métadonnées*) permettant d'aider les utilisateurs à interpréter de manière automatique les résultats des méthodes de fouille de données. Nous appliquons cette approche au domaine de la classification automatique.
Cette nouvelle approche consiste à :

- Définir et à utiliser un ***Métamodèle pour l'Interprétation des Résultats de Classification (MIRC)***. Les techniques de métamodélisation constituent une des réponses possibles au problème d'hétérogénéité des formalismes d'interprétation en classification. Utiliser un métamodèle permet de spécifier l'abstraction d'un ou de plusieurs modèles d'interprétation. Il permet de définir la terminologie à utiliser pour définir nos modèles d'interprétation. Dans le cadre de cette thèse, l'utilisation d'un métamodèle d'interprétation permet de ne retenir que les concepts jugés pertinents dans le processus d'interprétation.
- Créer une ***Architecture d'utilisation de Métadonnées pour l'Interprétation des résultats de Classification (AMIC)***. Cette architecture définit le cadre d'utilisation de notre métamodèle d'interprétation (MIRC). Cette architecture traite ainsi la prise en compte des requêtes d'interprétation des utilisateurs. Nous présentons, à travers AMIC, les différentes phases de prise en compte et de traitements des requêtes d'utilisateurs. Cette architecture, basée sur le processeur *Saxon*, permet de gérer les scénarios d'interprétation écrits en langage *XQuery*. Elle sert ainsi d'interface entre les différentes composantes de notre métamodèle.
- Créer une ***Ontologie du Domaine de la Classification (ODC)*** en vue de faciliter l'automatisation de l'interprétation des résultats des méthodes de classification. Cette ontologie permet de connaître le scénario d'interprétation à utiliser suivant la méthode utilisée et/ou les données traitées. Elle facilite aussi la prise en compte de nouveaux critères d'interprétation.
- Implémenter tous ces aspects dans un ***outil d'Aide à l'Interprétation des Résultats de Classification (AIRC)*** grâce auquel nous pouvons exploiter notre approche et fournir aux utilisateurs une interface conviviale.

Cette thèse s'organise de la manière suivante :

- **Le Chapitre 1** présente la problématique de notre travail ainsi que les contributions apportées par cette thèse. Il passe en revue les différentes techniques d'interprétation et notre positionnement par rapport à celles-ci. A travers ces critiques, nous expliquons les objectifs que nous nous sommes fixés dans le cadre de ce travail.
- **Le Chapitre 2** explique le contexte et le positionnement de notre travail. Il s'agit d'expliquer notre positionnement par rapport aux domaines de l'*analyse des données symboliques*, de *l'extraction de connaissances à partir des données*, de *l'ingénierie des connaissances* et de *l'ingénierie ontologique*.
- **Le Chapitre 3** présente l'extraction des connaissances à partir de données. Dans ce chapitre, nous abordons les différentes étapes de ce processus : le pré-traitement, le traitement et le post-traitement ainsi que l'analyse de données symboliques. A travers ces descriptions, nous expliquons que notre travail se positionne dans l'anticipation de l'étape de post-traitement.
- **Le Chapitre 4** présente les concepts d'*ontologie* et de *métadonnées*. Ainsi, pour les ontologies, nous expliquons, entre autres, leur importance et leur processus de mise en œuvre. Quant aux métadonnées, nous expliquons en quoi elles sont un outil formidable de capitalisation de connaissances.
- **Le Chapitre 5** se réfère aux différents domaines dans lesquels les métadonnées sont utilisées. Il s'agit de montrer que ce concept est largement partagé et utilisé dans des domaines divers et variés.
- **Le Chapitre 6** explique dans un premier temps, les différentes techniques de métamodélisation utilisées. Puis nous présentons la stratégie que nous avons adoptée dans le cadre de cette thèse avant d'expliciter notre métamodèle. Ainsi, ce chapitre est consacré exclusivement à la présentation de notre métamodèle d'interprétation MIRC ainsi que son instanciation par un modèle d'interprétation utilisant les *Variables*.
- **Le Chapitre 7** présente la spécification de notre architecture AMIC dédiée aux composantes de notre métamodèle. Dans ce chapitre, nous présentons les différents modèles d'architecture logicielle et les raisons qui nous ont conduit à faire le choix de l'architecture *Modèle-Vue-Contrôleur (MVC)*. Ce chapitre est aussi consacré à l'explication de certaines technologies utilisées dans notre architecture. Ainsi, nous abordons les notions liées au langage XQUERY ou au langage de transformation XSLT.
- **Le Chapitre 8** présente notre ontologie pour le domaine de la classification ODC, dédiée à l'aide à l'interprétation des résultats de classification. Nous présentons les raisons qui ont conduit à mettre en œuvre une ontologie de ce domaine et le bénéfice que nous pouvons en tirer quant à l'automatisation du processus d'interprétation.
- **Le Chapitre 9** se consacre à l'explication de notre application. Il s'agit de montrer, à travers notre outil AIRC, l'apport de notre approche dans l'aide à l'interpréta-

tion en se basant sur deux jeux de données et sur des exemples de méthodes de classification.

Abstract

According to the huge volume of data produced by different information sources, managers of several domains increase use Knowledge discovery and data mining (KDD). However, interpreting results obtained by these methods are very complex. In this thesis, we propose to help end-users to interpret, automatically, the results of their clustering methods. Thus, this work addresses the last step of KDD process (postprocessing).
In our work, we propose a new approach, called AIMFD (Approche d'Incorporation de Métadonnées dans la Fouille de Données) and based on using semantic informations clustering process. This approach, by these characteristics, can be applied to others data mining methods.
This approach consists to :

- design and to use a new metamodel, based on our clustering ontology, for cluster interpreting process. A meta-model is an explicit model of the constructs and rules needed to build specific models within a domain of interest. This metamodel, mainly inspired from and based on Common Warehouse Metamodel (CWM), allows us to make automatic interpretation process ;
- to create an architecture based on Saxon processor. This architecture allows us to manage interpretation scenarios coding in *XQuery* ;
- to design clustering ontology. This minimal ontology of clustering domain helps us within the definition of these scenarios.

Then we defined some interpretation scenarios in our tool. To evaluate our work, we used this new approach on Weka and into Sclust Algorithm.

Table des matières

I Contexte et Problématique de la thèse **7**

1 Problématique et contributions **11**
1.1 Techniques d'interprétation de classes 12
1.2 Objectifs de la thèse 13
1.3 Contributions 14
1.4 Synthèse du Chapitre 1 15

2 Positionnement de notre thèse **17**
2.1 Positionnement par rapport à l'ADS 18
2.2 Positionnement par rapport à l'ECD 19
2.3 Positionnement par rapport à l'ingénierie des connaissances 21
2.4 Positionnement par rapport à l'ingénierie ontologique 22
2.5 Synthèse du Chapitre 2 23

II État de l'art **25**

3 Extraction des Connaissances à partir de Données **31**
3.1 Processus d'ECD 32
3.1.1 Enjeux autour du processus ECD 32
3.1.2 Prétraitement des données 33
3.1.3 Fouille de données 34
3.1.4 Interprétation des résultats en ECD 37
3.1.5 Fouille de données et PMML 42
3.2 Analyse des données symboliques 44
3.2.1 Objectifs de l'analyse des données symboliques 45
3.2.2 Objet Symbolique 45
3.2.3 Synthèse et discussion 48
3.3 Classification automatique 49
3.3.1 Problématique en classification automatique 51
3.3.2 Étapes de la classification automatique 52
3.3.3 Qualités et Propriétés d'une méthode de classification 53

3.3.4 Types de méthodes de classification 53
3.4 Approches d'interprétation en classification automatique 55
3.4.1 Interprétation du rôle des individus et des variables : 56
3.4.2 Interprétation des classes par les variables : 57
3.4.3 Autres approches d'interprétation : 58
3.5 Synthèse du Chapitre 3 59

4 Métadonnées et Ontologie **61**
4.1 Ontologie 62
4.1.1 Définitions 62
4.1.2 Importance d'avoir une ontologie 63
4.1.3 Étapes de mise en œuvre d'une ontologie 64
4.1.4 Processus de création d'ontologie 65
4.1.5 Langages et outils de représentation d'ontologie 67
4.1.6 Différences entre vocabulaire contrôlé, thésaurus, taxonomie, ontologie 71
4.1.7 Types d'ontologies 72
4.2 Métadonnées 75
4.2.1 Définitions de métadonnées 75
4.2.2 Métadonnées : outil de capitalisation de la connaissance 76
4.2.3 Typologie de métadonnées 76
4.2.4 Besoins des utilisateurs de métadonnées 78
4.3 Panorama des travaux utilisant les métadonnées et/ou l'ontologie dans le processus d'ECD 78
4.4 Synthèse du Chapitre 4 80

5 Domaines d'utilisation des métadonnées **81**
5.1 Métadonnées en statistique 82
5.1.1 Statistique, Web et Métadonnées 83
5.1.2 Fonctions et rôles des métadonnées en statistique 84
5.1.3 Utilisateurs de métadonnées en statistique 85
5.2 Métadonnées en analyse des données symboliques 86
5.3 Métadonnées en Systèmes décisionnels 88
5.4 Métadonnées en fouille de données : utile ou futile ? 89
5.5 Métadonnées en Documentation électronique 90
5.6 Autres domaines 91
5.7 Standards et normes de métadonnées 91
5.8 Synthèse du Chapitre 5 94

III Contributions pour l'aide à l'interprétation de classes 97

6 Métamodèle pour l'Interprétation des Résultats de Classification 103

6.1 Quelques définitions . . . 105
6.2 Objectifs de notre Métamodèle . . . 106
6.3 Techniques de métamodélisation . . . 107
6.3.1 L'exemple du MOF . . . 108
6.3.2 L'exemple du XMI . . . 111
6.3.3 L'exemple du CWM . . . 113
6.4 Apports de la métamodélisation . . . 115
6.5 Axes d'interprétation des résultats de classification automatique . . . 116
6.6 Notre stratégie de modélisation . . . 117
6.7 Spécification de notre Métamodèle d'Interprétation . . . 117
6.7.1 Architecture générale de notre métamodèle . . . 118
6.7.2 Spécification des concepts de notre métamodèle . . . 120
6.8 Exemple de Modèle d'interprétation . . . 122
6.8.1 Structure de notre fichier de métadonnées . . . 123
6.8.2 Cas d'utilisation . . . 123
6.8.3 Diagramme de séquences . . . 125
6.9 Synthèse du Chapitre 6 . . . 126

7 Architecture pour l'interprétation des résultats de classification 127

7.1 Modèles d'architecture logicielle . . . 128
7.1.1 Les modèles à couche . . . 129
7.1.2 Les modèles à objets . . . 130
7.1.3 Les modèles hybrides . . . 132
7.2 Spécification de notre modèle d'architecture . . . 134
7.3 Utilisation des standards dans notre architecture . . . 136
7.3.1 Utilisation de PMML . . . 137
7.4 Le langage de requêtes XQuery . . . 138
7.5 Le langage de transformation XSLT . . . 140
7.5.1 Notions de base . . . 140
7.5.2 Les avantages de XSLT . . . 141
7.5.3 Exemple . . . 141
7.6 Outils pour le traitement des flux XML . . . 143
7.6.1 L'API SAX (Simple API for XML) . . . 143
7.6.2 L'API DOM (Document Object Model) . . . 144
7.6.3 JDom : API de traitement de fichiers XML dédiée au langage Java 145
7.7 Synthèse du Chapitre 7 . . . 147

8 Ontologie de la classification **149**
8.1 Contextes d'utilisation d'ontologie 150
8.2 Objectifs de notre ontologie 151
8.3 Processus de création de l'ontologie 153
8.3.1 Stratégies de développement d'ontologie 155
8.3.2 Éléments de notre ontologie 156
8.3.3 Propriétés de notre ontologie 160
8.3.4 Axiomes de notre ontologie 161
8.4 Utilisation de l'API *Jena* 162
8.5 Evaluation d'ontologie 164
8.6 Synthèse du Chapitre 8 164

9 Applications de notre approche **165**
9.1 Présentation du processus de traitement de notre approche 166
9.2 Critères généraux permettant le calcul des critères d'interprétation 167
9.3 Présentation de notre outil : AIRC 168
9.4 Premier scénario d'interprétation 171
9.5 Second scénario d'interprétation 172
9.6 Ajout et/ou modification d'un scénario d'interprétation 173
9.7 Expérimentations 174
9.7.1 Les jeux de données 174
9.7.2 Expérimentation dans les Méthodes de Nuées Dynamiques 174
9.7.3 Expérimentation dans Weka 176
9.7.4 Expérimentation dans Sclust 183
9.7.5 Expérimentation dans AIRC 185
9.8 Synthèse du Chapitre 9 186
Bibliographie 195
Annexe 195

A Diagramme de classes de notre application **217**
A.1 Modélisation et acquisition du fichier de métadonnées 217
A.2 Utilisation de l'architecture Modèle Vue Contrôleur (MVC) 220

B Programme de transformation XSLT des fichiers de métadonnées **223**

C Description en langage OWL de notre ontologie **233**

Table des figures

1 Organisation générale de la thèse 1
2 Organisation de la partie I 9

2.1 Positionnement de notre thèse 18
2.2 Organisation de la partie II 27

3.1 Etapes d'ECD 32
3.2 Étapes dans un processus de prétraitement 35
3.3 Choix d'un fichier de données dans le module *Explorer* de WEKA 38
3.4 Valeurs de critères statistiques des variables (WEKA) 39
3.5 Interprétation des résultats de l'algorithme *J48* (WEKA) 39
3.6 Présentation du module *Experimenter* de WEKA 40
3.7 Analyse des résultats dans le module *Experimenter* dans WEKA 41
3.8 Structure générale du langage PMML 2.1 42
3.9 Modélisation par des objets symboliques 46
3.10 Étapes d'une classification automatique 52
3.11 Exemple de partition en deux classes 54
3.12 Exemple de classification par densité 55
3.13 Exemple de classification par grille 56

4.1 Étapes de mise en œuvre d'ontologie 64
4.2 Langages de réprésentation d'ontologie (- sémantique au + sémantique) . 68
4.3 Représentation du graphe RDF de la description de l'objet *TSS* 69
4.4 Différence entre vocabulaire contrôlé, thésaurus, taxonomie et ontologie . . 73

5.1 Métadonnées sur données symboliques 88
5.2 Contributions de notre approche dans l'interprétation des résultats de classification 99

6.1 Modélisation dirigée par un métamodèle général 104
6.2 Architecture du MOF 108
6.3 Exemple de métamodélisation du langage UML 110
6.4 Fonctionnement du standard XMI 112

6.5 Métamodèle CWM pour la fouille de données . . . 114
6.6 Relation entre outil de fouille de données, modèle d'interprétation et métamodèle . . . 118
6.7 Architecture de notre métamodèle . . . 119
6.8 Le concept de processus d'interprétation . . . 120
6.9 Le concept d'élément d'interprétation . . . 121
6.10 Le concept d'axe d'interprétation . . . 122
6.11 Modèle d'interprétation dirigé par les variables . . . 122
6.12 Scénario de cas d'utilisation . . . 124
6.13 Diagramme de séquence d'un scénario d'interprétation . . . 125

7.1 Système de base . . . 128
7.2 Architecture du modèle ARCH . . . 130
7.3 Architecture MVC (Model-View-Controller) . . . 131
7.4 Architecture du modèle PAC-AMODEUS . . . 133
7.5 Architecture de production de métadonnées . . . 134
7.6 Sortie du document *book.xml* après application de la transformation XSLT 143

8.1 Une vue du logiciel PROTÉGÉ2000 version 3.2 . . . 154
8.2 Représentation de notre ontologie . . . 159
8.3 Utilisation du pattern Stratégie . . . 164

9.1 Processus de traitement de notre application . . . 167
9.2 Présentation de notre outil . . . 168
9.3 Visualisation du fichier de métadonnées . . . 169
9.4 Édition des requêtes Xquery . . . 170
9.5 Fenêtre d'édition des requêtes Xquery . . . 170
9.6 Listing Nuées Dynamiques sur les Iris . . . 175
9.7 Contribution à l'inertie interclasse par variable (AIRC) . . . 176
9.8 Fenêtre de lancement de Weka . . . 177
9.9 Résultat de l'application de EM sur les Iris (WEKA) . . . 179
9.10 Résultat de l'application de KMeans sur les Iris (WEKA) . . . 180
9.11 Valeurs des critères de l'inertie interclasse pour chaque variable (AIRC) . 180
9.12 Valeurs des critères de l'inertie totale pour chaque variable (AIRC) . . . 181
9.13 Valeurs des critères de l'inertie intra-classe pour chaque variable (AIRC) . 182
9.14 Interprétation : variables discriminantes (AIRC) . . . 182
9.15 Listing de la méthode Sclust sur les données de température (tiré de SCLUST) 184
9.16 Variables discriminantes suivant leur contribution à l'inertie interclasse (AIRC) . . . 185

A.1 Schéma du Patron de méthode . . . 219
A.2 Schéma du patron de conception Stratégie . . . 220

A.3 Schéma du patron de conception Observateur 221

Liste des tableaux

3.1 Tableau de données 44
3.2 Tableau de critères de validation 50

5.1 Tableau de données des *Abalone* 86

9.1 Tableau des critères utilisés dans le module *Experimenter* de Weka 178

Organisation de la thèse

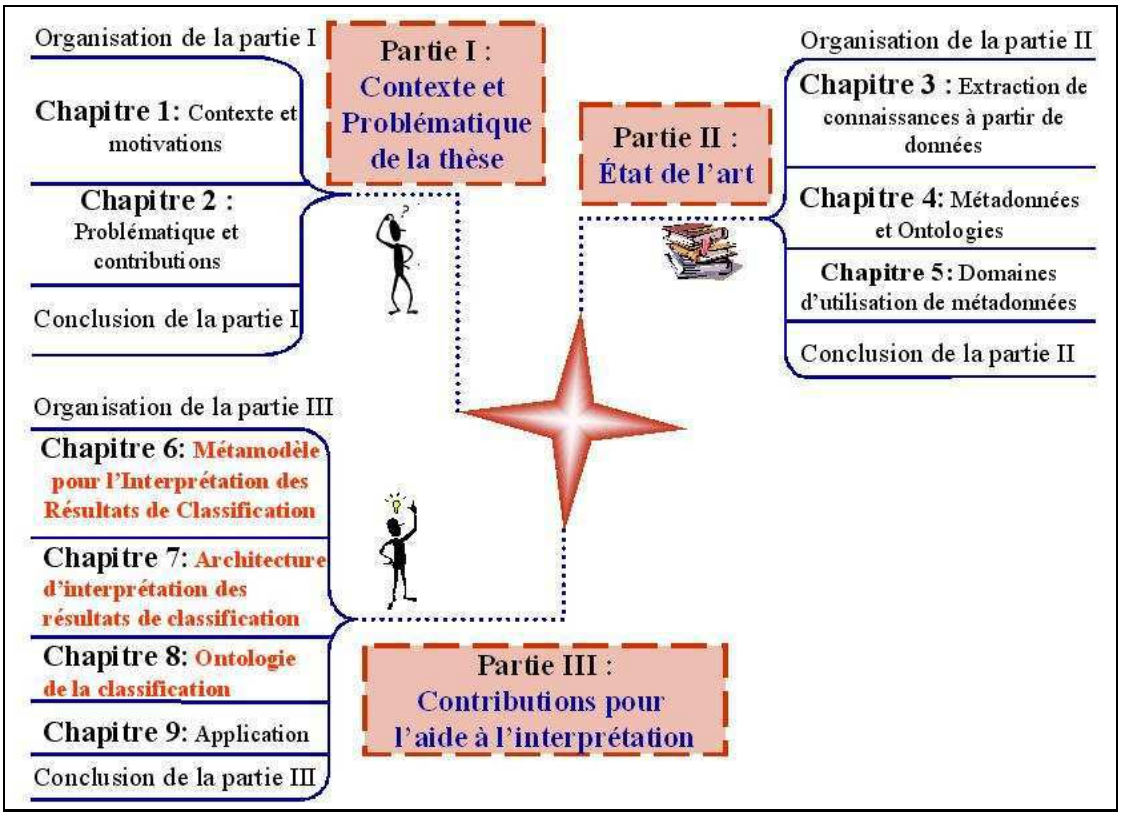

Fig. *1 - Organisation générale de la thèse*

Introduction Générale

De nos jours, tous les secteurs d'activité foisonnent de données de toutes sortes. Ces données sont souvent mal structurées et difficiles d'accès. Conserver la maîtrise sur ces grands ensembles d'information devient de plus en plus complexe. Les métadonnées constituent une voie pour aider l'utilisateur ou le gestionnaire d'information à comprendre, retrouver, comparer les données ou les informations. Pour s'en convaincre, prenons l'exemple simple de l'organisation d'une bibliothèque. En effet, pour faciliter la localisation d'un document, le système d'une bibliothèque définit des métadonnées sur chacun des éléments qu'on y trouve. Ces métadonnées sont constituées d'informations facilitant la recherche documentaire en répondant aux questions *qui* (l'auteur), *quoi* (le titre), et *quand* (la date de publication).

Dans le domaine de la ***fouille de données***, nous sommes confrontés à des problèmes relatifs à la structuration des résultats des différentes méthodes de fouille de données. Ainsi, suivant les outils et les méthodes utilisés, les structures des résultats sont très difficilement exploitables par un *utilisateur*. Il en est de même pour le domaine de la ***classification automatique***. Exploiter ces résultats en vue d'aider à une interprétation automatique de ceux-ci devient quasiment impossible.

L'objectif de nos travaux est d'anticiper la phase d'interprétation, en fournissant à l'utilisateur des informations *pertinentes* pouvant l'aider dans cette phase cruciale. Concrètement, il s'agit de créer un métamodèle d'interprétation modélisant le processus d'interprétation. Aider les utilisateurs à interpréter les classes obtenues constitue la particularité de cette thèse. Pour ce faire, nos travaux s'articulent autour de trois axes : *la définition d'un métamodèle d'interprétation*, *la définition d'une architecture* qui puisse exploiter ce métamodèle et, enfin, la mise en place d'une *ontologie de la classification* ayant pour rôle d'aider à l'automatisation du processus d'interprétation. Pour valider notre approche, nous avons construit un *prototype* aidant à l'interprétation des résultats des méthodes de classification. Ces méthodes sont issues du logiciel Weka et de l'algorithme Sclust.

Pour modéliser ce processus, nous avons utilisé les métadonnées. En effet, les métadonnées ont l'avantage de contenir des informations relatives au processus de collecte et de traitement des données. Ainsi, cette thèse s'organise de la manière suivante :

- **Le Chapitre 1** présente la problématique de notre travail ainsi que les contributions apportées par cette thèse. Il passe en revue les différentes techniques d'interprétation et notre positionnement par rapport à celles-ci. A travers ces critiques, nous expliquons les objectifs fixés dans cette thèse.
- **Le Chapitre 2** explique le contexte et le positionnement de notre travail. Il s'agit d'expliquer notre positionnement par rapport aux domaines de l'*analyse des données symboliques*, de *l'extraction de connaissances à partir des données*, de *l'ingénierie des connaissances* et de *l'ingénierie ontologique*.
- **Le Chapitre 3** présente l'extraction de connaissances à partir de données. Dans ce chapitre, nous abordons les différentes étapes de ce processus : le pré-traitement,

le traitement et le post-traitement ainsi que l'analyse de données symboliques. A travers ces descriptions, nous expliquons le positionnement de nos travaux dans ce processus.

- **Le Chapitre 4** présente les concepts d'*ontologie* et de *métadonnées*. Ainsi, pour les ontologies, nous expliquons, entre autres, leur importance et leur processus de mise en œuvre. Quant aux métadonnées, nous expliquons en quoi elles sont un outil formidable de capitalisation de connaissances.
- **Le Chapitre 5** se réfère aux différents domaines dans lesquels les métadonnées sont utilisées. Il s'agit de montrer que ce concept est largement partagé et utilisé dans des domaines divers et variés.
- **Le Chapitre 6** explique dans un premier temps, les différentes techniques de métamodélisation utilisées. Puis nous expliquons la stratégie que nous avons adoptée dans le cadre de cette thèse avant d'expliciter notre métamodèle. Ainsi, ce chapitre est consacré exclusivement à la présentation de notre métamodèle d'interprétation ainsi que son instanciation par un modèle d'interprétation basé sur les *Variables*.
- **Le Chapitre 7** présente la spécification de notre architecture dédiée à notre métamodèle. Dans ce chapitre, nous présentons les différents modèles d'architecture logicielle et les raisons qui nous ont conduit à faire notre choix. Ce chapitre est aussi consacré à l'explication de certaines technologies utilisées dans le cadre de notre architecture. Ainsi, nous abordons les notions liées au langage XQUERY ou au langage de transformation XSLT.
- **Le Chapitre 8** présente notre ontologie pour le domaine de la classification, dédiée à l'aide à l'interprétation des résultats de classification. Ainsi, nous expliquons les raisons qui ont conduit à mettre en œuvre une ontologie de ce domaine et le bénéfice que nous pouvons en tirer quant à l'automatisation du processus d'interprétation.
- **Le Chapitre 9** se consacre à l'explication de notre application. Il s'agit de montrer, à travers un outil, l'apport de notre approche dans l'aide à l'interprétation en se basant sur un jeu de données et sur des exemples de méthodes de classification.

Première partie

Contexte et Problématique de la thèse

Organisation de la partie I

Dans cette partie nous expliquons le contexte et la problématique qui sont liés à ce travail. Il s'agit, dans un premier temps, d'expliquer les raisons qui nous ont conduit à mener ce travail. Dans un second temps, nous expliquons notre positionnement et nos contributions dans les domaines de l'analyse des données symboliques, de la classification automatique et de l'ingénierie des connaissances. Cette partie est scindée en deux chapitres (figure 2) :

- **Le Chapitre 1** explique la problématique de ce travail et nos contributions dans le domaine de l'aide à l'interprétation des résultats des méthodes de fouille de données ;
- **Le Chapitre 2** explique le contexte de ce travail et le positionne par rapport aux domaines de l'analyse des données symboliques, de la classification automatique, de l'ingénierie des connaissances et de l'ingénierie ontologique.

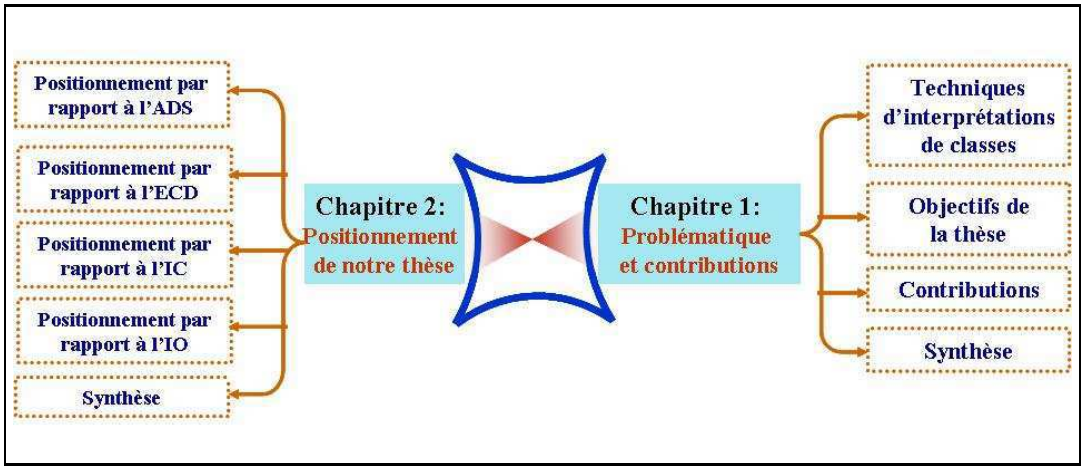

FIG. *2 - Organisation de la partie I*

Chapitre 1

Problématique et contributions

Introduction

DANS ce chapitre, nous expliquons la problématique autour du processus d'aide à l'interprétation des résultats de classification ainsi que nos contributions dans ce domaine. Il convient de rappeler qu'il existe peu de travaux qui se sont intéressés au domaine de l'aide à ***l'interprétation automatique*** de résultats de classification. L'une des rares études bibliographiques réalisées dans ce domaine est l'œuvre de [Lamiri et al., 2004]. Cette étude a montré, si besoin en était, qu'il n'existait pas de travaux génériques sur des méthodes d'aide à l'interprétation automatique des résultats issus de la fouille de données. Ceci est d'autant plus vrai pour le domaine de la classification automatique. En effet, toutes les méthodes d'interprétation connues à ce jour sont liées à la technique de fouille de données utilisée.

Dans ce travail, nous montrons qu'il est possible, grâce à notre approche, d'automatiser la phase d'aide à l'interprétation des résultats de classification. Par ailleurs, il est important de mettre en exergue la confusion faite entre les concepts de validation et d'interprétation.

Validation de classes Vs. Interprétation de classes

Rappelons que les deux tâches effectuées à la suite d'une classification automatique sont la validation et l'interprétation des résultats [Manco et al., 2004] :

- *Validation des classes :* une classification est qualifiée de « valide » si les classes découvertes correspondent à des groupements homogènes et séparés ;
- *Interprétation des classes :* une classification est « interprétable » si on arrive à associer la signification appropriée à chaque classe.
  Autrement dit, les classes qui ont une signification pour l'utilisateur sont utilisables pour l'objectif pour lequel elles ont été créées.
  Remarquons aussi que la majorité des travaux réalisés dans le cadre de l'interprétation utilisent des critères statistiques orientés validation plutôt que des critères orientés interprétation [Ben Ahmed, 2005].

Dans la section 3.3, nous expliquons plus en détails cette ambiguïté qui subsiste entre ces deux concepts. Pour mieux comprendre la problématique autour du concept d'interprétation, nous expliquons brièvement dans la section suivante les techniques utilisées dans le cadre d'une interprétation des résultats de méthodes de fouille de données (pour plus de détails cf. section 3.4).

1.1 Techniques d'interprétation de classes

Les techniques d'interprétation des classes les plus utilisées sont des techniques d'exploration visuelle combinées avec les indicateurs statistiques. La visualisation des résultats est généralement interactive, ce qui permet d'incorporer les connaissances expertes dans la tâche d'interprétation. Par exemple la représentation de données multidimensionnelles peut être réalisée par un diagramme plan [Zhang et al., 2003] ou par un graphique matriciel [Grinstein et al., 2001]. Par ailleurs, il convient de rappeler que l'interprétation est très liée à la méthode de fouille de données utilisée.

Pour s'en apercevoir, nous prenons l'exemple de [Celeux et al., 1989] relatif à l'interprétation des résultats de classification. Ces travaux présentent des indices de description d'une partition à partir d'un ensemble de variables. Ces indices sont fondés sur la *décomposition de l'inertie* de la population en *inertie interclasse* et en *inertie intra-classe* associées à la partition étudiée. Suivant leur rôle dans la description de la partition, on distingue trois types de variables :

- *variables actives :* ce sont des variables qui interviennent dans la détermination de la partition ;
- *variables passives ou illustratives :* comme leur nom l'indique, ce sont des variables qui n'interviennent pas dans la détermination de la partition ;
- *variables prédictives :* ce sont des variables qui sont à expliquer à partir des variables actives.

Les indices que nous proposons reposent sur la décomposition de l'inertie totale T en inertie interclasse B et en inertie intra-classe W. Cette décomposition étant vraie globalement, par variable, par classe et par couple (variable j, classe k).
Ainsi nous avons les relations suivantes :

$$T = B + W \tag{1.1}$$

$$T_j = B_j + W_j \tag{1.2}$$

$$T_k = B_k + W_k \tag{1.3}$$

$$T_j^k = B_j^k + W_j^k \tag{1.4}$$

Pour interpréter les résultats, l'utilisateur doit faire appel à un expert en classification qui propose une stratégie d'interprétation basée sur les indices précédents. Notre objectif est de mettre en œuvre une méthode générique d'interprétation flexible qui puisse fournir à l'utilisateur une aide efficace dans l'interprétation de ses résultats.

1.2 Objectifs de la thèse

La classification consiste à regrouper les individus ayant les mêmes caractéristiques dans les mêmes classes et ceux qui sont différents dans des classes disjointes. L'un des problèmes majeurs à résoudre dans ce processus est celui de l'interprétation des classes obtenues (cf. section 3.3.1).
En effet, les critères d'interprétation utilisés dépendent, en partie, de la méthode de fouille de données utilisée. Pour montrer cette corrélation, prenons l'exemple suivant. En arbre de décision, nous avons le critère de *corrélation entre variables* tandis qu'en classification nous avons les critères d'homogénéité. De plus, l'interprétation de ces résultats dépend très fortement de l'analyste, des critères utilisés, de l'objectif de départ ainsi que des données traitées. Ce qui pose un problème au niveau de la modélisation de ce processus d'interprétation car même si les algorithmes sont performants, nous pouvons obtenir des résultats difficilement exploitables par les utilisateurs finaux. Ainsi, un même résultat peut être interprété différemment suivant les points de vue du (ou des) utilisateur(s).

D'où la nécessité d'avoir une approche qui soit indépendante de la subjectivité de l'analyste et des divers critères d'interprétation existants. Ce qui peut s'effectuer en anticipant cette phase d'interprétation : il s'agit de fournir à l'utilisateur des informations pertinentes pouvant l'aider dans cette tâche. Dans cette thèse nos objectifs sont :

- de ***définir un métamodèle*** d'interprétation des résultats issus des méthodes de classification automatique. L'instanciation de ce métamodèle doit être possible pour tout modèle d'interprétation. De plus, les modèles d'interprétation obtenus doivent être exploitables de manière automatique.
  Par ailleurs, ce métamodèle doit être indépendant de la méthode de fouille de données utilisée et doit être basé sur l'ontologie du domaine de la classification ;

- d'utiliser ce métamodèle pour **extraire automatiquement** les éléments les plus pertinents qui ont été obtenus lors du processus de classification ;
- de définir, à partir de ce métamodèle, une **architecture qui puisse l'exploiter** dans le cadre de l'aide à l'interprétation ;
- de **définir une ontologie** du domaine de la classification pouvant être exploitée pour l'aide à l'interprétation automatique des résultats ;
- de définir des **scénarii d'interprétation** de classes.
  Ces scénarii doivent intégrer les métadonnées contextuelles, les critères statistiques (paramètres de la méthode, usage des variables, etc.), les concepts du domaine de la classification (à travers l'ontologie créée) ainsi que les informations éventuelles du domaine analysé.

L'une des particularités de cette thèse est l'utilisation de métadonnées comme éléments de capitalisation des connaissances extraites lors du processus de classification. Ces métadonnées ainsi obtenues sont associées aux résultats qu'elles décrivent. Ceci permet de garantir que la description des résultats est fidèle à la réalité et représente le statut réel des résultats.
Pour associer les métadonnées aux données/résultats, nous adoptons les principes de [Froeschl and Grossmann, 1999] qui sont :

- *le principe de la terminologie :* définir une sémantique pour les éléments de métadonnées du domaine, c'est-à-dire donner une description statique des données ;
- *le principe de l'intégrité :* transformer les métadonnées descriptives de données en des métadonnées décrivant le résultat du traitement des données. Ce principe assure l'exactitude sémantique et syntaxique du traitement des données ;
- *le principe du désir des utilisateurs :* les attentes des utilisateurs étant très différentes suivant les domaines, il convient d'accorder une importance particulière sur la modélisation de leurs besoins.

1.3 Contributions

Pour expliquer nos contributions dans le cadre de ce travail, nous prenons quatre domaines sur lesquels le sujet de cette thèse a un lien. Il s'agit des domaines de l'analyse des données symboliques, de la classification automatique, de l'ingénierie ontologique et de l'ingénierie des connaissances.

Dans le domaine de ***l'analyse des données symboliques***, notre approche a permis l'automatisation du processus d'ajout de métadonnées. En effet, les éléments de métadonnées utilisés jusqu'ici avec les objets symboliques étaient principalement des éléments descriptifs et dont on ne pouvait pas se servir dans l'interprétation des résultats obtenus sur ces données symboliques. Notre approche permet de prendre en compte tous types de métadonnées alors que jusqu'ici les métadonnées décrivant les objets symboliques étaient prédéfinies.

Dans le domaine de la ***classification automatique***, nous avons expliqué que l'objectif de ce travail est d'anticiper l'étape d'interprétation des résultats en fournissant aux utilisateurs les informations pertinentes en vue de l'interprétation. Ce qui s'est traduit par l'adoption d'une approche modélisant le processus d'interprétation par des métadonnées. Notre principal apport dans ce domaine se situe dans la mise en œuvre d'une approche indépendante de la méthode, de l'outil et des critères de classification utilisés.

Dans le domaine de l'***ingénierie des connaissances***, nous nous appuyons sur les techniques de métamodélisation existantes pour élaborer un métamodèle du processus d'interprétation. A ce niveau, il convient de rappeler qu'il n'existe aucun métamodèle rendant compte de ce processus d'interprétation. La complexité de ce processus étant grande, nous avons montré qu'il est important de définir un cadre de modélisation pour tous types d'interprétation.

Dans le domaine de l'***ingénierie ontologique***, nous avons utilisé des méthodes et outils de développement d'ontologie pour créer une ontologie du domaine de la classification. En effet, il semble évident qu'une interprétation automatique, dans un environnement aussi hétérogène, n'est possible que par la création d'une structure sémantique forte de ce domaine. Dans ce sens, nous avons spécifié l'ontologie dédiée au domaine de la fouille de données appelée DAMON [Cannataro and Comito, 2003]. Ceci nous a permis de créer une ontologie dédiée spécifiquement au domaine de la classification en vue d'aider à l'interprétation des résultats.

1.4 Synthèse du Chapitre 1

Dans ce chapitre, nous avons expliqué la problématique liée à l'automatisation du processus d'interprétation des résultats des méthodes de classification. A travers ce chapitre, nous avons montré que ce processus est extrêmement complexe et qu'à ce jour il n'existait pas de solution générique pouvant répondre à cette question.
Ensuite, à la lumière des difficultés évoquées, nous avons expliqué les objectifs que nous nous sommes fixés dans la réalisation de cette thèse. Ces objectifs tiennent en une expression : anticiper la phase d'interprétation des résultats. Puis nous expliquons brièvement l'approche de (méta)modélisation utilisée pour résoudre ce problème ainsi que les autres techniques auxquelles nous avons fait appel dans ce travail.
Enfin, nous avons expliqué les contributions de notre approche aux domaines qui nous ont été utiles dans la réalisation de ce travail.

Chapitre 2

Positionnement de notre thèse

Introduction

En classification, l'aide à l'interprétation consiste en un processus ou une méthode permettant de fournir aux utilisateurs des « informations pertinentes » permettant la compréhension des classes obtenues.
Toutefois, il convient de souligner l'aspect ambigu du terme « informations pertinentes ». En effet, suivant les données traitées et la méthode de classification utilisée, une même information peut être déterminante ou non dans la phase d'interprétation des résultats obtenus.

Pour pallier cette ambiguïté, nous proposons une approche basée sur l'utilisation des connaissances acquises au cours du processus de classification. Ces connaissances seront modélisées en utilisant des métadonnées.
Dans ce chapitre, nous discutons du contexte de cette thèse en expliquant son implication avec les domaines de l'analyse des données symboliques (ADS), l'extraction de connaissances à partir de données (ECD), de l'ingénierie des connaissances (IC) et de l'ingénierie ontologique (IO). Dans le même ordre d'idées, nous expliquons les motivations qui nous ont poussées à travailler sur ce thème. La figure 2.1 montre les liens entre notre travail et les domaines que nous avons évoqués précédemment.

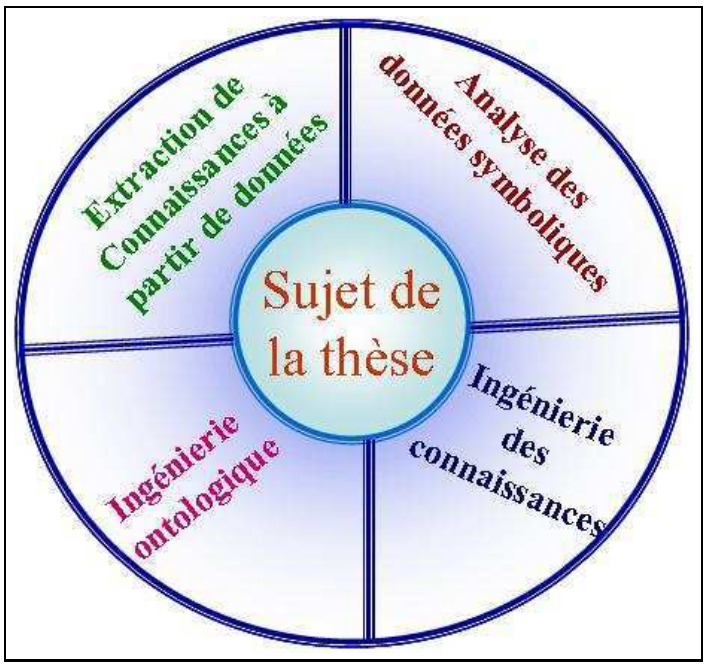

FIG. *2.1 - Positionnement de notre thèse*

2.1 Positionnement par rapport à l'ADS

Ce domaine initié et développé par E. DIDAY [Bock and Diday, 2000] a pour objectif de permettre une nouvelle modélisation en offrant un formalisme plus riche que dans le cadre classique. Les méthodes issues de ce domaine offrent ainsi aux utilisateurs la possibilité d'élargir les types d'objets à représenter, les types de mesures à utiliser ainsi que les opérateurs de généralisation. Dans ses travaux, E. DIDAY [Bock and Diday, 2000] a défini un panel d'objets symboliques parmi lesquels les assertions qui ont un lien étroit avec nos travaux. En effet, une assertion est un vecteur où chaque composant peut être un intervalle, un ensemble de valeurs ou une distribution de fréquences. A travers ce type de formalisme, on est en mesure de modéliser et de représenter un certain nombre de données complexes.

Dans cette perspective de généralisation, nous pouvons citer les travaux de V. STÉPHAN [Stéphan, 1998] qui a proposé une automatisation du processus de généralisation utilisé dans les méthodes d'ADS. Ainsi, dans les méthodes d'ADS, l'objectif est de permettre à l'utilisateur de passer d'une information détaillée et stockée dans une base de données à une information résumée prenant en compte la structure complexe des objets à analyser. Ces travaux ont abouti par la mise en place d'une technique permettant de résumer les informations d'une base de données par des objets symboliques (*Data Base To Symbolic Object*).

A travers cette description, nous constatons que l'objectif de l'ADS est d'aider les utilisateurs à mieux comprendre une information disparate en procédant à une généralisation de celle-ci. Or dans nos travaux, notre objectif est de permettre aux utilisateurs

de mieux interpréter les résultats issus de méthodes de fouille de données, y compris les méthodes utilisant le formalisme de représentation symbolique. Ainsi, grâce à notre approche nous sommes capables d'utiliser *l'intension* des objets symboliques qui ont servi à la classification pour aider à l'interprétation de nouveaux objets symboliques créés à partir des classes obtenues.

En effet, une classification sur des objets symboliques rend plus complexe l'interprétation des résultats issus de ces types de données. Ainsi, un utilisateur doit pouvoir avoir accès à l'historique des objets symboliques dont il dispose. Dans cette optique, il semble indispensable de stocker dans des métadonnées les informations sur les opérations de généralisation réalisées sur les données sources. Ainsi les travaux de [Papageorgiou et al., 2000a], [Papageorgiou et al., 2002], [Papageorgiou et al., 2000c], [Papageorgiou et al., 2000b] et [Vardaki, 2004] ont porté sur ces aspects et ont servi de base dans la modélisation des informations sémantiques à extraire à partir des objets symboliques.

2.2 Positionnement par rapport à l'ECD

L'ECD est un processus itératif et interactif se déroulant principalement en trois étapes : le *pré-traitement*, le *traitement* et le *post-traitement* des données. C'est cette dernière étape qui nous intéresse dans le cadre de ce travail.
Plusieurs définitions ont été avancées dans la littérature sur le concept de fouille de données. L'une d'entre elles est l'œuvre de [Grossman, 2004a] qui définit la fouille de données comme étant la découverte semi-automatique de patterns, de changements, d'associations, d'anomalies et d'autres structures statistiquement intéressantes à partir de données. Aussi, des définitions pour le moins inattendues concernant ce processus sont avancées, il s'agit par exemple de celle de D. Chorafas [Lefébure and Venturi, 1998] à savoir que la fouille de données consiste à « torturer l'information disponible jusqu'à ce qu'elle avoue ». Bien évidemment le but est de les faire « avouer » quelque chose de compréhensible par les utilisateurs, chose qui n'est pas aisée. Pour d'autres définitions sur ce concept, se reporter à [Marcos and Solange, 2005].

Cette thèse s'articule autour de la recherche d'une approche permettant d'anticiper la phase de *post-traitement* en fournissant aux utilisateurs des informations dans ce sens. En effet, la phase de *post-traitement* est la phase la plus délicate et la plus subjective et il est difficile d'automatiser cette étape.
Par ailleurs, plusieurs techniques d'extraction de connaissances existent (classification automatique, classement, règles d'association, etc. cf. § 3.1.3).

Dans le cadre de cette thèse, nous nous intéressons uniquement aux problématiques liées à la recherche d'une solution efficace d'automatisation du processus d'aide à l'interprétation des résultats issus de la classification automatique.
Il s'agit de répondre à la question : (Comment trouver une description des classes compréhensible par les utilisateurs et réalisable par le concepteur ?) C'est-à-dire trouver une approche générique permettant de modéliser le processus d'interprétation afin d'aider les

utilisateurs dans leurs tâches d'interprétation.

Comme le souligne FAYYAD dans [Fayyad et al., 1996], le concept d'ECD peut être vu comme un processus non trivial permettant d'identifier de nouveaux patterns potentiellement utiles et « interprétables » par les utilisateurs. Or en pratique, il est très difficile de trouver une interprétation objective à chacun des patterns identifiés. En effet, la formation de ces patterns est liée aux paramètres de la méthode d'analyse dont le nombre de classes souhaité, la mesure de similarité, la méthode, l'utilisateur et les critères d'interprétation utilisés.

Notre travail se justifie ainsi par le fait qu'à notre connaissance, il n'existe aucun travail dans la littérature sur la problématique d'aide à l'interprétation automatique des résultats dans le processus d'ECD. Pour pallier ce déficit, nous développons une approche basée sur la définition d'un Métamodèle pour l'Interprétation des Résultats de Classification (MIRC) que nous développons au Chapitre 6 et dont l'objectif est de formaliser le processus d'interprétation. Notre défi est de trouver un moyen de modéliser les multiples critères utilisés dans le cadre de l'interprétation des résultats de classification automatique. En effet, même si les algorithmes de classification utilisés sont performants, on peut obtenir des résultats difficilement exploitables (i.e. difficilement interprétables et utilisables) par les utilisateurs finaux.

Cette dans cette optique que nous proposons une approche basée sur l'utilisation de *métadonnées* comme outil de capitalisation des connaissances acquises au cours du processus de classification. Comme nous l'expliquons dans la partie III, notre approche se base sur la création d'un métamodèle modélisant le processus d'interprétation des résultats de classification. Ensuite nous avons mis en œuvre une architecture permettant d'exploiter ce métamodèle.

Il convient aussi de rappeler que les critères d'interprétation utilisés dépendent très fortement de l'utilisateur et de la méthode de classification et ce, même si l'objectif reste le même. De plus, le même résultat peut être interprété différemment selon le point de vue de l'utilisateur. C'est pourquoi notre approche consiste à identifier et à formaliser les connaissances, du domaine de la classification, en métadonnées. Ces métadonnées sont ainsi utilisées dans la phase d'interprétation des résultats.

Enjeux autour de la fouille de données

Le processus de fouille de données permet d'analyser, de comprendre et d'utiliser la connaissance extraite dans un système intelligent. Nous nous sommes intéressés à la phase de post-traitement, à savoir l'aide à apporter à l'utilisateur pour lui permettre d'interpréter facilement les résultats de son analyse. Compte tenu du fait que l'objectif de nos travaux (cf. § 1.2) est de mettre en œuvre une approche de modélisation du processus d'interprétation des résultats de méthodes de classification, il convient d'expliquer ce qu'est la fouille de données et les enjeux autour de ce processus. L'objectif de cette section est double : Montrer les difficultés auxquelles sont confrontés les utilisateurs désireux

d'interpréter les résultats issus d'une méthode de fouille de données. Ensuite expliquer l'apport de notre approche dans le processus d'interprétation des résultats obtenus dans le cadre des méthodes de classification.

La fouille de données est un processus analytique conçu pour explorer les données (de préférence de grands ensembles de données) à la recherche des patterns consistants et/ou des relations systématiques entre les variables. Ce processus peut se résumer en trois étapes [Han and Kamber, 2000] :

- l'exploration initiale ;
- le modèle de construction ou l'identification de patterns avec la validation/vérification ;
- l'application du modèle pour les nouvelles données dans le but de générer les prédictions.

Comme nous pouvons le constater, à travers les définitions avancées, l'interprétation des résultats des méthodes de fouille de données dépend de l'objectif de l'étude, du type de la méthode, de l'utilisateur, du type des données, etc. Et, à notre connaissance, il n'existe pas à ce jour de méthode permettant d'aider les utilisateurs à interpréter, de manière automatique, les résultats provenant de tous types de méthodes de classification et encore moins des méthodes de fouille de données.

2.3 Positionnement par rapport à l'ingénierie des connaissances

L'ingénierie des connaissances, développée dans le cadre de l'Intelligence Artificielle, est définie comme l'étude des concepts, méthodes et techniques permettant de modéliser et/ou d'acquérir les connaissances dans des domaines se formalisant a priori, peu ou pas [Charlet et al., 2000]. L'identification, la modélisation, l'explicitation, l'opérationnalisation et l'utilisation des connaissances sont parmi les activités centrales en ingénierie des connaissances. Dans ce travail nous nous intéressons à l'élaboration de métamodèles et de modèles pour le domaine de la fouille de données.

Plusieurs définitions sont données sur la notion de « modèle du domaine ». Nous retenons celle de CommonKADS[1] [Wielinga et al., 1994] qui le définit comme « une structuration des ontologies du domaine (concepts, relations entre les concepts, termes, etc.) selon un point de vue donné ». Dans ce travail la notion de point de vue est identique à celle d'axe d'interprétation. Mais pour l'identification de ces axes d'interprétation, la majorité des études dans la littérature se base essentiellement sur une étape d'élicitation (collecte de données, entretiens) suivie d'une étape de conceptualisation. Cette approche ascendante (« bottom-up ») présente des limites (coûteuse, risque d'incomplétude en terme d'axes d'interprétations, etc.).

[1] Standard Européen de modélisation des connaissances

Néanmoins c'est cette méthode que nous utilisons dans ce travail afin de trouver les axes d'interprétation que nous jugeons pertinents. Bien évidemment les limites de cette méthode, évoquées précédemment, ne se posent pas dans notre cas. En effet, nous avons fait le choix de ne retenir qu'un nombre connu et admis d'axes d'interprétation. A la lumière de nos études bibliographiques, nous supposons que tout processus d'interprétation se fait suivant trois axes au plus (*axe de la structure classificatoire, axe du tableau de données et axe des variables*). Ces axes sont développés dans la section 6.5.

Il existe aussi d'autres études utilisant des approches descendantes (« top-down ») qui commencent par la sélection, à partir d'une bibliothèque de modèles génériques comme celle de *CommonKADS*, du modèle le plus proche de l'application. Cette approche présente aussi des limites (difficulté de trouver un modèle générique, c'est ce qui présente dans notre cas et c'est la raison qui nous a poussé à ne pas adopter cette stratégie). Pour pallier les limites de ces approches, des approches mixtes ont été développées.

2.4 Positionnement par rapport à l'ingénierie ontologique

Le terme « ingénierie ontologique » a été proposé par R. MIZOGUCHI en 1997 pour désigner un nouveau domaine de recherche ayant pour but la construction et l'exploitation d'ontologies [Mizoguchi and Ikeda, 1997]. Il s'agit de construire des systèmes informatiques tournés vers le contenu, et non plus vers les mécanismes de manipulation de l'information. Ainsi, il insiste sur le fait que la *nature conceptuelle* d'une ontologie doit aider à faire partager des connaissances entre humains et ordinateurs d'une part et entre ordinateurs d'autre part. Pour lui, les fondements d'une ontologie se basent essentiellement sur les concepts de *conceptualisation*, de *réutilisation* et de *partage*.
Par ailleurs, il convient de noter la différence substantielle qui existe entre l'ingénierie des connaissances et l'ingénierie ontologique. En effet, selon R. MIZOGUCHI, l'ingénierie des connaissances est la recherche d'heuristiques spécifiques au domaine pour la résolution d'un problème alors que l'ingénierie ontologique est la recherche de concepts généraux, réutilisables, partageables et durables pour construire un modèle de connaissances capable d'aider des personnes à résoudre des problèmes. De plus, il faut savoir que l'ingénierie ontologique ne fait que réorganiser des connaissances existantes sur un domaine. Son principal objectif est de donner une base solide sur laquelle nous pouvons travailler pour produire de nouvelles connaissances dans le domaine. C'est ce qui explique que nous ayons consacré deux sections différentes à ces deux concepts.

En ce sens, l'apport de ce concept est d'une importance capitale dans les systèmes d'information orientés de plus en plus sur la sémantique. L'ingénierie ontologique, de par les méthodes de création et de conception existantes, nous a permis de mettre en œuvre notre ontologie du domaine de la classification. Cette ontologie spécifie l'ontologie dédiée au domaine de la fouille de données appelée DAMON [Cannataro and Comito, 2003]). Nous présentons dans la section 4.1, les méthodes, langages et outils existants pour la mise en place d'une ontologie. Dans ce domaine, notre apport principal reste la création

d'une ontologie pour le domaine de la classification automatique.

2.5 Synthèse du Chapitre 2

Dans ce chapitre, nous avons présenté les différents domaines connexes à notre travail. Ainsi, par rapport au ***domaine de l'analyse des données symboliques***, notre approche s'est basée sur les travaux réalisés par l'équipe de H. PAPAGEORGIOU. Ces travaux ont pour but de trouver un modèle de représentation pour les métadonnées descriptives du processus de collecte et d'agrégation des données.

Le ***domaine de l'ECD*** nous sert de base à ce travail. En effet, notre approche se positionne dans l'anticipation du processus d'interprétation des résultats de méthodes de fouille de données. Il s'agit de fournir aux utilisateurs une aide précieuse et efficace pour interpréter leurs résultats.

Par rapport au ***domaine de l'ingénierie des connaissances***, nous faisons appel à la notion de métamodélisation pour modéliser le processus d'interprétation. Ce qui se traduit par la création d'un métamodèle d'aide à l'interprétation des résultats de méthodes de classification. En effet, la complexité et la variété des critères d'interprétation rendent difficiles la mise en place d'une approche générique d'interprétation. Mais grâce à ce métamodèle, nous montrons qu'il est possible d'instancier n'importe quel modèle d'interprétation de résultats de classification.

Par rapport au ***domaine de l'ingénierie ontologique***, nous nous appuyons sur les méthodes et outils de développement d'ontologie pour mettre en place une ontologie du domaine de la classification. Cette ontologie spécifie l'ontologie DAMON dédiée au domaine de la fouille de données. Au chapitre suivant, nous abordons en détail la problématique de cette thèse ainsi que nos contributions dans le domaine de l'aide à l'interprétation des résultats de méthodes de classification.

Conclusion de la partie I

Dans cette partie, nous avons abordé les problématiques liées au processus d'interprétation des résultats des méthodes de classification.
Puis nous avons expliqué les objectifs de nos travaux qui se résument principalement en l'incorporation des métadonnées dans le processus d'aide à l'interprétation des résultats issus de méthodes de classification. Cette approche peut se résumer ainsi :

- la recherche d'un *métamodèle orienté à l'interprétation des résultats de classification*. Ce métamodèle devant nous permettre de nous abstraire des difficultés de modélisation du processus d'interprétation ;
- la *définition d'une architecture* permettant d'exploiter ce métamodèle et de fournir aux utilisateurs une aide dans leur processus d'interprétation ;
- la *définition d'une ontologie* du domaine de la classification permettant une automatisation de ce processus.

Ensuite, nous avons situé notre sujet de thèse à travers ces liens avec les domaines de l'extraction de connaissances à partir de données, de l'ingénierie des connaissances et de l'ingénierie ontologique. Le but étant de montrer notre positionnement scientifique par rapport aux domaines cités précédemment.
Enfin, nous avons expliqué brièvement nos contributions dans les domaines auxquels nous avons fait appel dans la réalisation de cette thèse. Dans la partie suivante, nous faisons un état de l'art sur quelques domaines et techniques que nous avons utilisés dans ce travail.

Deuxième partie

État de l'art

Organisation de la partie II

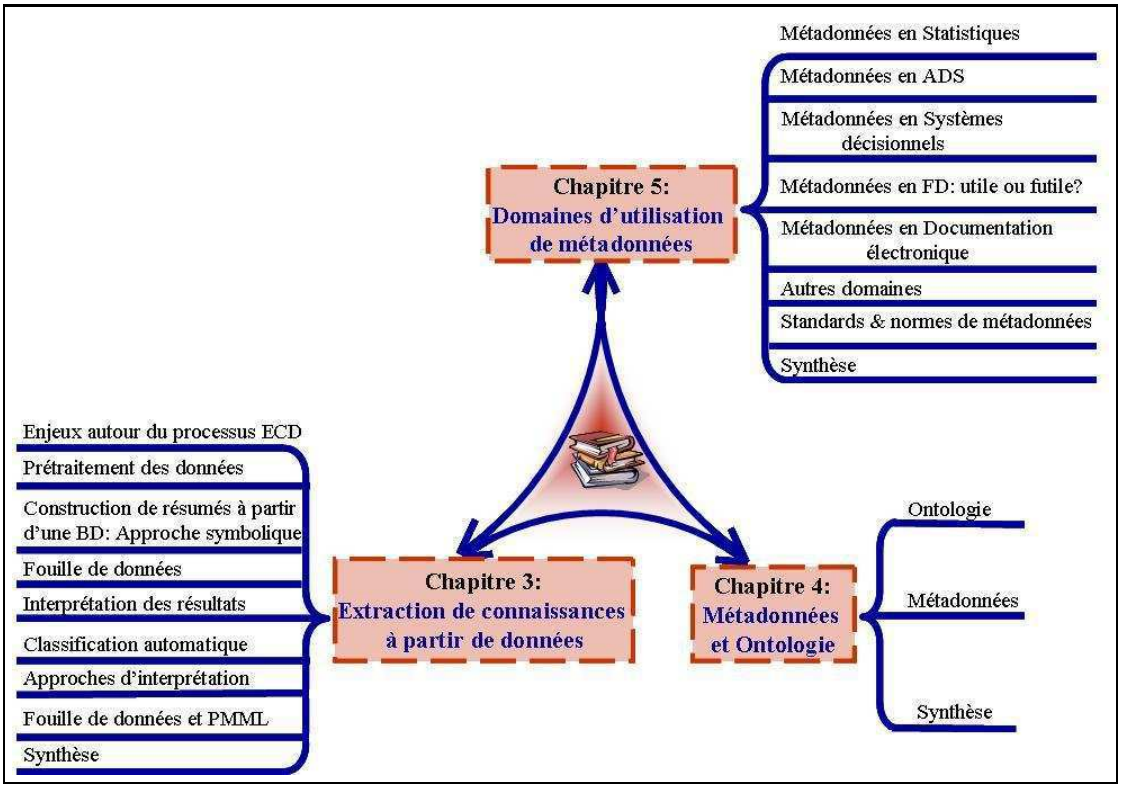

Fig. *2.2 - Organisation de la partie II*

Introduction de la partie II

L'OBJECTIF EST d'expliquer de manière approfondie les problématiques liées à notre travail afin de le positionner à travers les divers domaines abordés. Ainsi, dans cette partie, nous abordons ***l'extraction des connaissances à partie des données*** (Chapitre 3), les concepts de ***métadonnées et d'ontologie*** (chapitre 4) et les ***domaines d'utilisation des métadonnées*** (Chapitre 5) (tout ceci est explicité dans la figure 2.2). Pour ce qui est de la présentation d'un état de l'art sur les ***techniques de métamodélisation***, nous avons fait le choix de l'aborder dans le chapitre consacré à notre métamodèle (Chapitre 6).

Ainsi dans le ***Chapitre 3***, nous présentons un état de l'art sur l'*ECD*, la *fouille de données* et la *classification automatique*. L'objectif de ce chapitre étant de présenter le domaine sur lequel nous consacrons une partie de ce travail. Dans ce chapitre, nous expliquons les problématiques liées aux techniques d'interprétation dans le cadre des méthodes de fouille de données en général et de la classification en particulier. A travers ce chapitre, nous présentons les sources de complexité liées au processus d'interprétation des résultats issus des méthodes de classification. Par ailleurs, nous abordons dans ce chapitre le langage de représentation ***PMML*** dans nos travaux.

Dans le ***Chapitre 4***, nous abordons les concepts de *métadonnées* et d'*ontologie* en expliquant leur place dans nos travaux. Ainsi, nous montrons par exemple l'intérêt de formaliser le processus d'interprétation par des métadonnées. Par ailleurs, une large place est accordée à l'apport que pourrait avoir l'utilisation d'une ontologie dans ce processus de capitalisation de connaissance.

Dans le ***Chapitre 5***, nous faisons un état de l'art sur l'utilisation des métadonnées dans certains domaines. L'objectif est de montrer, à travers ce chapitre, que le concept de métadonnées est aujourd'hui largement utilisé dans des domaines divers et variés.

Chapitre 3

Extraction des Connaissances à partir de Données

Introduction

Avec l'avènement des grandes bases de données, les techniques d'extraction de connaissances sont les mieux à même de répondre aux besoins des utilisateurs de divers domaines, soucieux d'accéder à l'information pertinente dans le cadre des prises de décision. C'est dans ce contexte que l'ECD (Extraction de Connaissances à partir de Données) a fait son apparition.

Selon U. Fayyad, l'ECD peut être vue comme un processus non trivial permettant d'identifier de nouveaux patterns potentiellement utiles et interprétables par les utilisateurs [Fayyad et al., 1996]. Ce processus, itératif et interactif, se déroule en trois grandes étapes théoriques (cf.figure 3.1) : *le prétraitement des données, l'application des techniques de fouille de données et l'interprétation des résultats obtenus* (pour plus de détails voir [Fayyad et al., 1996], [Brachman and Anand, 1996] et [Han and Kamber, 2000]).

Dans ce chapitre, nous ne prétendons pas traiter toutes les facettes liées au processus d'ECD. En effet, ce domaine constitue un axe de recherche très actif et aborde plusieurs thématiques intéressantes qui dépassent le cadre de ce chapitre. Ce chapitre présente

dans la section 3.1 l'ECD, dans la section 3.2 l'analyse des données symboliques et dans la section 3.3 la classification automatique.

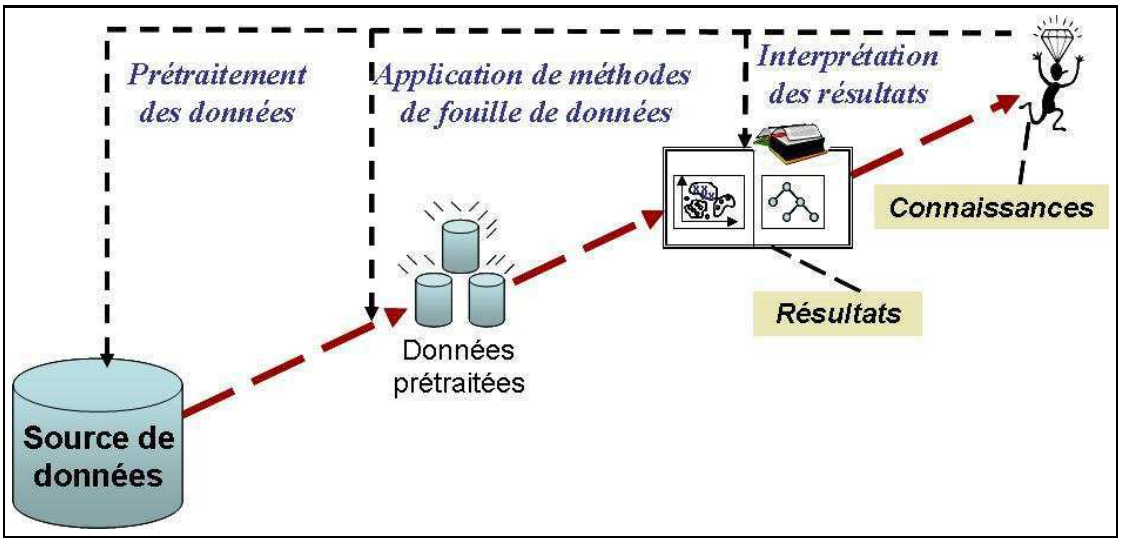

FIG. *3.1 - Etapes d'ECD*

3.1 Processus d'ECD

3.1.1 Enjeux autour du processus ECD

Il convient de rappeler que la plupart des algorithmes utilisés dans ce processus trouvent un nombre important de patterns dont l'analyse reste complexe. Par exemple les méthodes recherchant les règles d'association génèrent un grand nombre de patterns. Il s'agit de trouver les voies et moyens pouvant permettre une meilleure interprétation des résultats des règles d'association. Parmi ces travaux, nous pouvons citer ceux de [Marcos and Solange, 2005] qui consistent à réaliser une méthode pouvant aider à l'analyse des règles obtenues. Cette méthode consiste à utiliser des taxonomies dans l'étape de post-traitement de la connaissance. Ils ont ainsi proposé un algorithme (GART) qui utilise des taxonomies pour généraliser les règles d'association. Cette méthode permet, de diverses manières, d'analyser les règles ainsi généralisées.

D'autres travaux ont été réalisés dans ce domaine, ils consistent à générer, dans un premier temps, toutes les règles qui ont un support (respectivement une confiance) plus grand que le support (respectivement la confiance) minimum défini par l'utilisateur [Brisson et al., 2006]. En effet, de grandes valeurs de ces minimums peuvent générer des règles triviales. En revanche de petites valeurs vont générer une très grande quantité de patterns difficilement interprétables.

Pour pallier ce problème ils se proposent d'utiliser une nouvelle approche utilisant la notion d'ontologie. Il s'agit d'une méthode permettant d'intégrer la connaissance des experts dans les arbres de décision. Cette méthode est basée sur l'utilisation d'une ontologie

du domaine d'étude afin d'aider à la généralisation des règles extraites. Selon eux, l'utilisation d'une ontologie peut aider les utilisateurs à trouver les patterns intéressants parmi les règles générées. Ils ont mis en place un algorithme qui ne généralise qu'une partie de la règle d'association (la gauche ou la droite). En fait c'est à l'utilisateur de décider de la partie à généraliser et ce, après avoir observé un ensemble de règles sans taxonomies.
Ce regroupement se fait en des sous-ensembles représentant les antécédents ou des conséquences égaux. Suivant le choix de l'utilisateur, les sous-ensembles devraient être généralisés en utilisant les conséquences des règles. Les taxonomies sont ensuite utilisées dans le but de généraliser chaque sous ensemble (par exemple un chat et un chien sont des animaux domestiques).

Des algorithmes, tels que *CUMULATE AND STRATIFY*, utilisant des taxonomies pour générer des règles d'association existent déjà dans la littérature. Pour plus de détails, voir [Tseng and Lin, 2004], [Srikant and Agrawal, 1997], [Adamo, 2000], [Liu et al., 2000]. Dans ce domaine il existe aussi d'autres travaux qui ont traité de la prise en compte de points de vue[1] des experts du domaine afin d'enrichir sémantiquement les connaissances extraites durant un processus d'ECD [Behja et al., 2005b]. Il s'agit de garder la trace des points de vue donnés par les experts sur les analyses effectuées tout le long du processus d'ECD. Ces travaux ont abouti par la création d'une ontologie des éléments de modélisation du processus ECD basés sur les points de vue. Par ailleurs, il propose une plate-forme objet pour l'annotation multi-points de vue d'un processus d'ECD. Cette annotation est basée sur l'utilisation de certains patrons de conception. Il s'agit d'une plate-forme de développement, d'évaluation et d'évolution d'un système d'information basé sur le Web [Behja et al., 2005a].

3.1.2 Prétraitement des données

Rappelons que la qualité des résultats d'un processus d'ECD dépend en grande partie de la qualité des données utilisées [Fayyad et al., 1996]. Et d'après [Famili et al., 2000], le prétraitement sur les données consiste en toute action effectuée sur les données avant l'application d'une technique de fouille de données. C'est essentiellement une transformation des données initiales en des données plus utiles. Cette transformation devra éliminer au moins un problème des données initiales tout en préservant l'information.

En effet, les données initiales peuvent être incomplètes, bruitées, aberrantes et incohérentes [Famili et al., 2000], [Han and Kamber, 2000], [Pyle, 1999]. Ce processus de prétraitement peut être décrit à travers les quatre (4) étapes suivantes [Famili et al., 2000], [Han and Kamber, 2000] (cf. figure 3.2) :

[1]Le point de vue étant défini comme la perception, basée sur sa propre expérience, d'un expert sur un processus d'ECD

1. **Le nettoyage de données :** selon [Rahm and Do, 1997], cette tâche consiste à détecter et à supprimer des erreurs, du bruit et de l'incohérence des données afin d'améliorer leur qualité. Il s'agit de retravailler les données bruitées, soit en les supprimant, soit en les modifiant de manière à en tirer le meilleur profit.
2. **L'intégration de données :** cette tâche combine des données provenant de sources multiples (bases de données, sources externes, etc.), détecte et résout des conflits de valeurs [Calvanese et al., 2001] ;
3. **La transformation de données :** cette tâche consiste à extraire et/ou à construire de nouvelles variables pour fournir une nouvelle représentation des données adéquates à l'application, au domaine et à l'objectif de l'étude [Hellerstein, 1997].
   Plusieurs méthodes telles que l'agrégation, la généralisation, la normalisation sont utilisées dans cette étape pour permettre l'extraction de ces variables. Pour une vue détaillée sur ces méthodes, voir [Han and Kamber, 2000] ;
4. **La réduction de données :** le coût de mesure et l'exactitude des résultats d'une technique de fouille de données sont les deux raisons principales pour maintenir le nombre de données le plus faible possible [Hellerstein, 1997].
   Mais cette réduction peut causer une perte d'information. Il s'agit donc de trouver un compromis. Il existe plusieurs stratégies de réduction de données, pour plus de détails voir [Han and Kamber, 2000].

Il convient aussi de souligner que ces tâches ne sont pas toutes systématiquement réalisées dans tous les processus de fouille de données. De plus, cette séparation n'est que théorique car la plupart des outils de fouille de données réalisent ces tâches en même temps.

3.1.3 Fouille de données

Introduction

Domaine à la confluence de différents domaines (Base de données, Statistiques, Intelligence Artificielle, Visualisation, Parallélisme, etc.), la fouille de données a pour but d'extraire des connaissances nouvelles, valides et utiles à partir d'une masse de données. C'est un processus qui consiste à appliquer des algorithmes sur les données afin d'en extraire des modèles (ou motifs). Ainsi, la fouille de données fait appel à un lot de méthodes issues de la statistique, de l'analyse des données, de la reconnaissance des formes, de l'intelligence artificielle ou de l'apprentissage automatique. Parmi les méthodes les plus connues, nous pouvons citer les règles d'association [Agrawal et al., 1993], les méthodes de classement (ou d'apprentissage supervisé ou de classification supervisée) et les méthodes de classification automatique (ou d'apprentissage non supervisée ou de classification non-supervisée) (la description de ces méthodes est donnée dans la section 3.1.3).

On peut trouver un état de l'art sur ces différentes approches dans [Jouini et al., 2003]. D'autres travaux de synthèse sur les méthodes de classification peuvent être trouvés dans

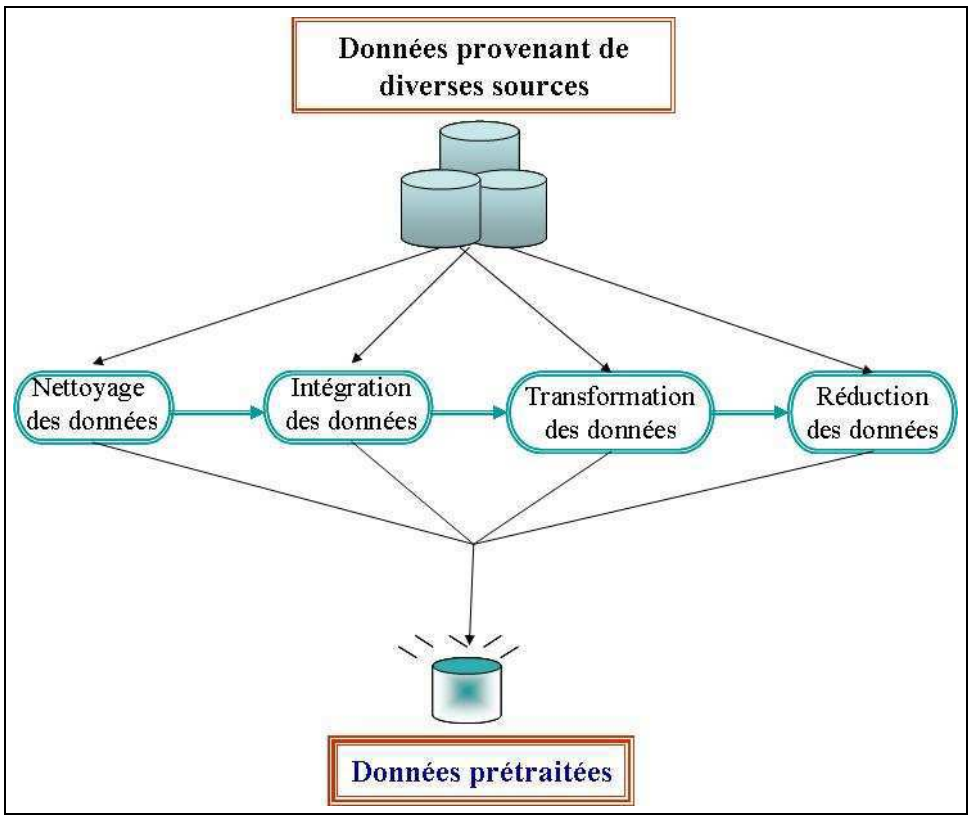

FIG. *3.2 - Étapes dans un processus de prétraitement*

[Han and Kamber, 2000], [Ben Ahmed et al., 2002], [Everitt, 1993], [Jain et al., 1999] et [Jain and Dubes, 1988]. Des comparaisons relatives à certains algorithmes de classification automatique sont abordées dans [Halkidi et al., 2001a] et [Zaït and Messatfa, 1997]. Dans cette section, nous abordons la problématique sur ce processus, les critères de qualité, les critères de sélection et la typologie des modèles de fouille ainsi que les tâches réalisées dans le cadre de ce processus. La dernière section est consacrée à la présentation de la méthode de fouille sur laquelle nous avons travaillé : la classification automatique. Comme nous l'expliquons dans la section 1.2, notre travail se focalise sur les méthodes de classification automatique.

Problématique en fouille de données

La fouille de données se situe à la croisée de plusieurs communautés scientifiques. En effet, elle fait appel à un lot de méthodes issues de la statistique, de l'analyse des données, de la reconnaissance des formes, de l'intelligence artificielle (connexionisme, réseaux de neurones) ou de l'apprentissage automatique.
Du fait de ces origines multiples dans des communautés universitaires différentes, une même méthode peut avoir des noms différents. On peut regrouper les méthodes en trois grandes catégories [Chauchat, 2002] (voir la section 3.1.3 pour plus de détails) :

- **méthodes de description uni, bi et multidimensionnelles :** issues de la statistique descriptive et de l'analyse des données, ainsi que de la visualisation graphique et de la réalité virtuelle ;
- **méthodes de structuration :** appelées classification automatique ou apprentissage non supervisé, ces méthodes proviennent de l'analyse des données, de la reconnaissance des formes, de l'apprentissage automatique et du connexionisme ;
- **méthodes explicatives ou prédictives :** dont le but est de relier un phénomène à expliquer à un phénomène explicatif.
  Elles sont utilisées pour prévoir un comportement qualitatif (guérison, achat, incident, panne, accident, etc.) ou numérique. Elles sont utilisées aussi pour classer de nouveaux cas dans des catégories prédéfinies (document, patient, client, etc.).
  Ces méthodes sont issues de la statistique, de l'économétrie, de la reconnaissance des formes, de l'apprentissage automatique et du connexionisme, voire du datamining pour les règles d'association.

Il convient aussi de rappeler que de nombreux algorithmes appartenant à ces trois familles ont été développés. Dans ce travail, nous nous intéressons à certaines d'entre elles afin de montrer l'utilité de notre approche.

Qualité d'une méthode de fouille de données

D'une manière générale, on mesure la qualité d'une méthode de fouille de données par les critères suivants :

- la rapidité de création ;
- la rapidité d'utilisation ;
- la compréhension par l'utilisateur ;
- la performance ;
- la fiabilité ;
- la performance dans le temps ;
- l'évolutivité.

Il va de soi qu'aucune méthode n'aura toutes ces qualités. Suivant le théorème de « no free lunch » de [Wolpert and Macready, 1997], il n'existe pas de meilleure méthode de fouille de données. Il faudra ainsi faire des compromis selon les besoins dégagés et les caractéristiques connues des outils. Pour une utilisation optimale, une combinaison de méthodes est judicieuse.

Catégories des méthodes de fouille de données

Il existe plusieurs travaux portant sur la catégorisation des méthodes de fouille de données, citons par exemple les travaux de [Han and Kamber, 2000], [Mitchell, 1997],

[Cormuéjols and Miclet, 2002] et [Agrawal and Srikant, 1994]. Dans notre travail nous nous limitons à la catégorisation suivante :

1. ***Classement, régression, prédiction :*** Il s'agit de trouver une classe ou une valeur selon un ensemble de descriptions. Ce sont des outils très utilisés. Les algorithmes reposent sur les arbres de décision, les réseaux de neurones, la classification Bayésienne, les k plus proches voisins, les algorithmes génétiques, les treillis de gallois, etc.

2. ***Association, sequencing :*** Il s'agit de trouver des similarités ou des associations. Le sequencing est le terme anglais utilisé pour préciser que l'association se fera dans le temps. Par exemple, si j'achète des layettes aujourd'hui, j'ai quatre fois plus de chance d'acheter, dans cinq mois, un lit de bébé. Ou encore si j'achète des pâtes et de la purée de tomates, j'ai deux fois plus de chance d'acheter aussi du parmesan.

3. ***Segmentation :*** La problématique est de trouver des groupes homogènes dans une population. On utilise souvent les algorithmes des k-means ou de Kohonen. La difficulté essentielle dans ce type de construction est la validation qui nécessite l'intervention d'experts humains. Dans ce travail, nous nous intéressons à la manière d'apporter une aide aux utilisateurs ne disposant pas d'une expertise leur permettant d'interpréter de façon convenable les résultats dont ils disposent. Et c'est en cela que notre approche est originale et intéressante dans ce domaine.

Par conséquent, à tout jeu de données et à tout problème correspond une ou plusieurs méthodes. Le choix se fera en fonction de la tâche à résoudre, de la nature et de la disponibilité des données, des connaissances et des compétences disponibles et de la finalité du modèle construit. Pour cela, les utilisateurs peuvent se baser sur les critères suivants : *la complexité de la construction du modèle*, *la complexité de son utilisation*, *ses performances*, *sa pérennité*, ou plus généralement *l'environnement de l'entreprise*.

3.1.4 Interprétation des résultats en ECD

Une étude bibliographique approfondie (cf. [Lamiri et al., 2004]) permet de constater, qu'il est quasiment impossible de trouver des travaux génériques sur des méthodes d'interprétation des résultats en fouille de données. Les techniques existantes, que nous abordons dans cette section, sont toutes dépendantes de la technique utilisée ou de l'outil réalisant la fouille de données. Dans cette section, nous présentons un exemple d'interprétation des résultats de méthodes supervisées dans le logiciel WEKA[2]. Puis, nous abordons les problématiques d'interprétation dans le domaine des méthodes non supervisées et plus précisément de la classification automatique.

[2]WEKA est un logiciel, développé par une équipe de chercheurs de l'université de Waikato (Nouvelle Zélande), qui permet le prétraitement et l'analyse des données. Il est disponible à l'adresse suivante : http ://www.cs.waikato.ac.nz/ml/weka/

Exemple d'interprétation des résultats de méthodes supervisées dans WEKA

L'objectif de ce paragraphe est de présenter un exemple d'interprétation des résultats issus de méthodes supervisées dans le logiciel WEKA. Pour cet exemple, nous prenons le jeu de données des ***Iris*** utilisés avec l'algorithme d'arbre de décisions *J48* implémenté dans WEKA. Sur les quatre modules contenus dans le logiciel WEKA (*SimpleCli, Explorer, Experimenter et KnowledgeFlow*), nous nous interessons uniquement aux modules *Explorer* et *Experimenter*. Nous commençons par présenter le module *Explorer* : en premier lieu il s'agit de choisir le fichier sur lequel nous effectuons le traitement (cf. figure 3.3).

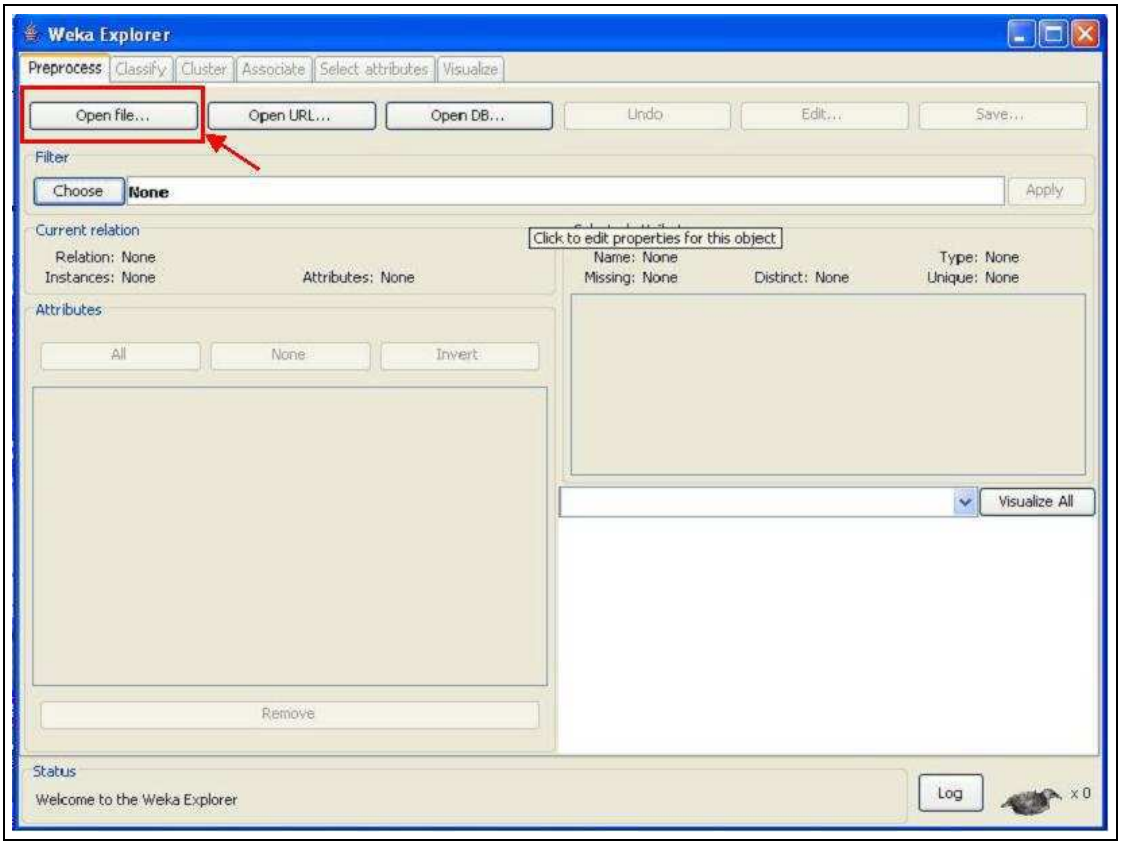

FIG. *3.3 - Choix d'un fichier de données dans le module* Explorer *de* WEKA

Les données ainsi chargées peuvent subir un prétraitement (par exemple *normaliser* les valeurs d'une variable). Une fois que ces données sont chargées, nous avons les représentations graphiques des différentes classes d'*Iris*. Nous pouvons aussi connaître les valeurs de certains critères statistiques des différentes variables utilisées lors du traitement des données (cf. partie **1** de la figure 3.3). Dans cet exemple nous sommes en présence des valeurs statistiques de la variable *longueur du sépale*. Tous ces traitements s'effectuent dans l'onglet *prétraitement*. Ensuite, nous passons à l'onglet *Classify* afin d'effectuer le choix de l'algorithme à utiliser pour le traitement des données.

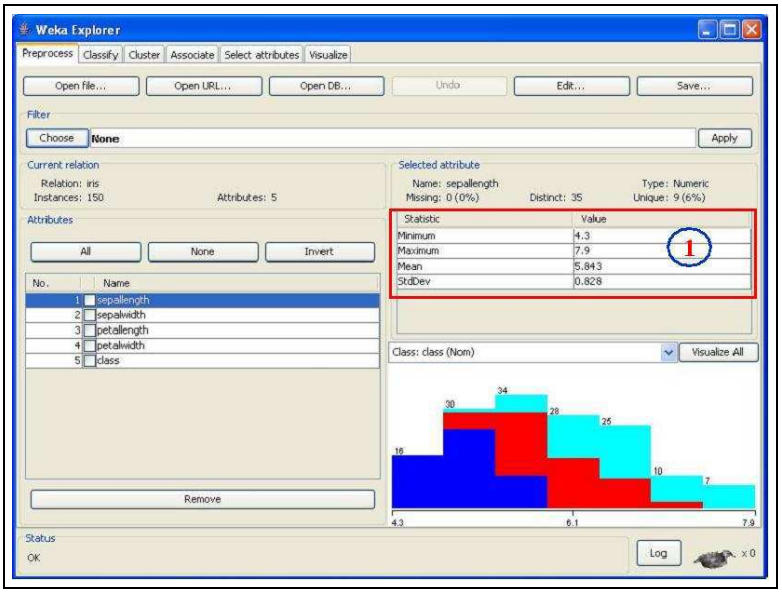

FIG. *3.4 - Valeurs de critères statistiques des variables* (WEKA)

A partir du moment où l'algorithme est choisi, il s'agit ensuite de choisir l'option de test (cf. partie **1** de la figure 3.5, dans ce cas nous optons pour la *validation croisée*). En exécutant, nous obtenons les valeurs des critères prédéfinis dans le logiciel WEKA (cf. partie **2** de la figure 3.5). La partie **3** de cette figure est celle relative à la *matrice de confusion* qui sert à calculer les valeurs des différents critères prédéfinis (*f-mesure, précision, mal-classés, pourcentage de mal-classés, etc.*).

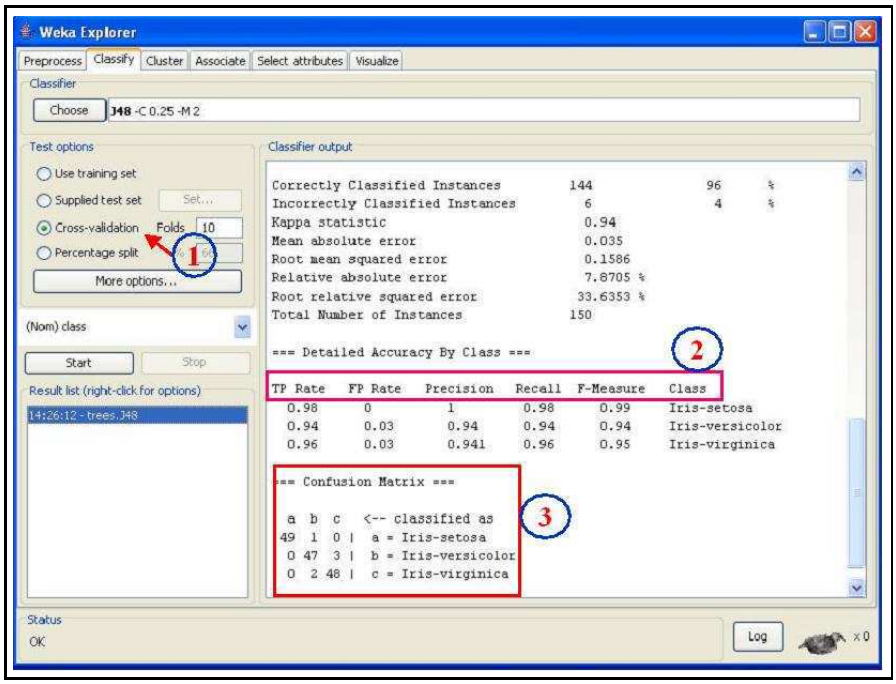

FIG. *3.5 - Interprétation des résultats de l'algorithme* J48 *(WEKA)*

Avec cette approche tous les critères sont déjà prédéfinis et l'interprétation des résultats est liée à la valeur de ces critères. Cependant pour certains algorithmes les valeurs de certains de ces critères sont nulles et dans ces conditions il est difficile de les comparer. De plus, l'ajout de nouveaux critères nécessite de modifier le code des algorithmes concernés. Notre objectif est de mettre en œuvre une approche qui offre les mêmes avantages que ceux fournis par WEKA dans le cadre supervisé. Dans le cadre des méthodes supervisées tous les critères d'interprétation sont dépendants de la matrice de confusion (). C'est à partir de cette matrice que s'effectue tous les calculs des différents critères d'interprétation utilisés dans le cadre *supervisé*.

Dans la dernière partie de cette section, nous présentons le module *Experimenter* du logiciel WEKA qui permet d'utiliser un ou plusieurs jeux de données avec un ou plusieurs algorithmes. Le but de ce module est de réaliser une comparaison entre différents modules sur un même jeu de données ou plusieurs jeux de données. L'intérêt de ce module est de pouvoir choisir, au vu des valeurs des critères prédéfinis, la méthode la mieux à même de répondre à nos objectifs. Dans la figure 3.7, les parties **1**, **2** , **3** et **4** représentent respectivement l'extension du fichier utilisé pour contenir les résultats de l'expérimentation, le nom de ce fichier, les données à traiter et les algorithmes utilisés dans la comparaison.

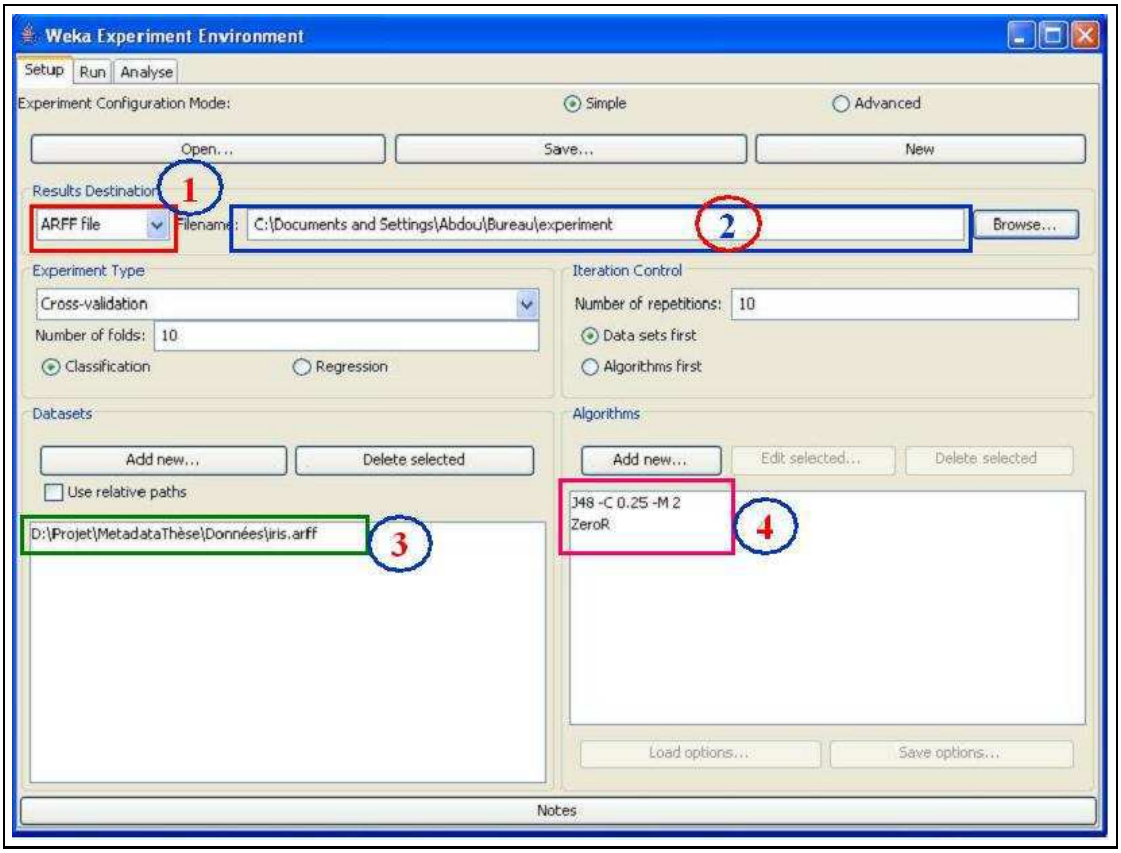

FIG. *3.6 - Présentation du module* Experimenter *de* WEKA

Ce module permet d'analyser les résultats issus de plusieurs algorithmes en comparant les valeurs de ces critères sur les différents algorithmes en jeu (cf. partie **2** de la figure 3.7 qui utilise le critère *pourcentage de bien classé*). On remarque que les valeurs de ce critère sont très différentes (**94.73%** pour l'algorithme d'arbre de décision *J48* contre **33.33 %** pour l'algorithme des règles de décision *ZeroR* (cf. partie **3** de la figure 3.7).

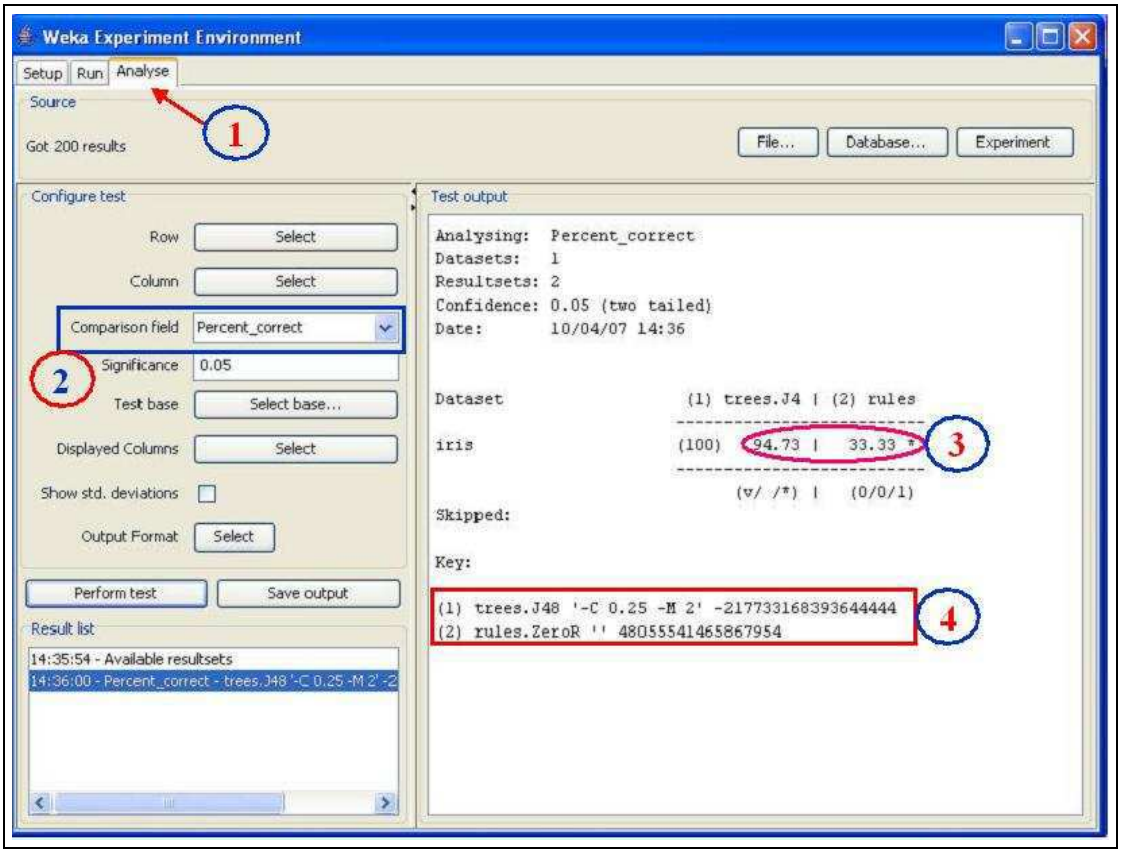

FIG. *3.7 - Analyse des résultats dans le module* Experimenter *dans* WEKA

Interprétation en classification automatique

L'interprétation en classification automatique est plus délicate car il n'y a pas de critère idéal mais différentes stratégies d'interprétation. C'est pourquoi il faut trouver une nouvelle approche qui rende flexibles les critères d'interprétation. Ainsi, il faut savoir que la stratégie d'interprétation la plus courante, en classification, est de décrire les classes à l'aide des individus et des variables. Une stratégie d'interprétation peut être décrite de la manière suivante : interpréter les classes en utilisant les individus qui la composent et/ou interpréter les classes en utilisant les variables. Ces procédés sont expliqués dans la section 3.4. Dans cette thèse, notre approche consiste à utiliser la même stratégie que celle mise en œuvre dans WEKA, dans le cadre supervisé, qui consiste à définir une *matrice de confusion* qui servira à calculer tous critères d'interprétation. Dans le cadre *non supervisé*, il est très improbable de définir une base qui va servir à calculer tous les scénarios.

3.1.5 Fouille de données et PMML

Basé sur le langage XML et issu du groupe de travail *Data Mining Group*[3], PMML (Predictive Model Markup Language) [DMG, 2006] est un langage qui décrit les modèles statistiques et de fouille de données. Concrètement, il décrit les modèles de fouille de données utilisés, les transformations réalisées au cours du prétraitement des données ainsi que les paramètres définissant les méthodes. La structure des modèles est décrite par XML Schema. Signalons que l'un des intérêts d'avoir des documents au format PMML est de permettre de réaliser sur un nouveau modèle de données conforme au modèle décrit dans le fichier PMML, le traitement de ces données. En effet, PMML permet de définir puis de partager des modèles statistiques et d'extraction de données entre différentes applications.

PMML est composé d'une partie dictionnaire de données, d'une partie dictionnaire de transformations, d'une partie entête et d'une partie contenant la description des modèles de fouille de données. La première contient la définition des champs utilisés dans les modèles de fouille de données. La seconde partie contient la description de ces champs et le dictionnaire de transformations qui peuvent être partagés par plusieurs modèles de fouille de données. De plus, chaque modèle peut définir ses propres éléments dérivés. Quant à la partie entête d'un document PMML, elle est constituée d'éléments contenant des informations nécessaires à l'utilisation d'un modèle. Ces éléments peuvent être un *copyright* ou une *description du logiciel* qui a généré le modèle. Cette entête peut aussi contenir des informations relatives aux *annotations* faites par les utilisateurs, aux informations sur la *version* ou sur le *nom de l'application*. Dans la figure 3.8, tirée de [DMG, 2006], nous avons une vue synthétique de la structure du langage PMML (version 2.1).

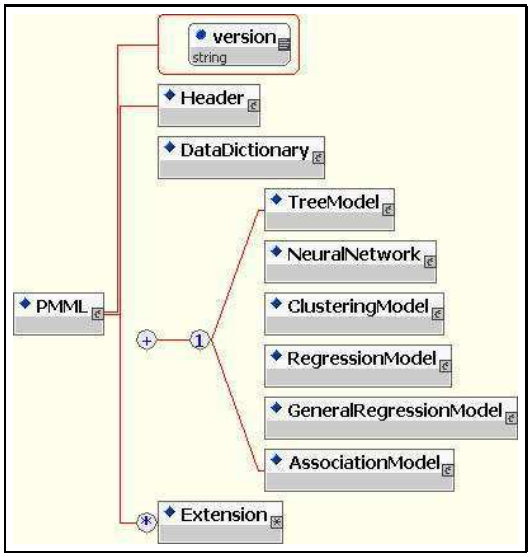

FIG. *3.8 - Structure générale du langage PMML 2.1*

[3]un consortium d'éditeurs de logiciels fondé en 1998 afin de développer des normes d'extraction de données

Si certaines descriptions des modèles dans PMML sont assez pertinentes, d'autres le sont beaucoup moins. C'est le cas du modèle de la classification automatique qui nous intéresse dans cette thèse. Ainsi seules les informations sur les distances, sur la distribution des données[4] et sur la mesure de similarité utilisée sont définies pour ce modèle. Il faut dire que PMML, comme son nom l'indique, est plutôt orienté pour les modules de classification supervisée. Rappelons tout de même que PMML possède un mécanisme d'extension du contenu du modèle de fouille de données mais ce mécanisme d'extension reste assez flou. En effet, il manque une explication claire sur la méthodologie à suivre pour procéder à cette extension. Par ailleurs beaucoup d'éléments sont constitués de texte libre. Ce qui pose quelques problèmes évidents quand il s'agit d'automatiser le traitement de ces modèles.

Quelle place pour PMML en fouille de données ?

La capacité à extraire et à analyser les modèles qui se cachent sous les données peut permettre aux organisations de mieux comprendre le comportement de leurs clients, les tendances de leurs marchés et donc de tirer profit de leurs informations. C'est ce qui explique la naissance et l'adoption de la norme PMML. La charte PMML consiste à permettre à une application de produire un modèle et à une autre application d'utiliser ce modèle simplement en lisant le fichier de données PMML.

Un modèle développé à l'aide d'un outil d'extraction de données réalisé sur un ensemble d'apprentissage peut ensuite être déployé ou exploité sur la totalité d'un entrepôt de données. Mais PMML a ses faiblesses, surtout quand il s'agit de préparer des données brutes en vue d'une analyse. Avec PMML, les modèles ne sont pas appliqués directement sur les données détaillées de l'entrepôt de données, mais sur un jeu spécialisé de données sélectionnées pour leur valeur prédictive. Elles sont nettoyées ou transformées, prêtes à être traitées par les modèles analytiques. PMML permet certaines transformations de données liées à des algorithmes spécifiques. Il est toutefois insuffisant quand il s'agit de gérer des processus complexes de nettoyage, de transformation et d'agrégation de toutes les données sélectionnées car PMML part du principe que la préparation préalable des données a déjà été effectuée.
Il est aussi inefficace quand il s'agit d'aider à l'interprétation des résultats car, comme nous le rappelons au début de cette section, PMML permet de décrire des modèles de fouille de données en vue de leur partage. De plus, il ne spécifie pas les paramètres utilisés pour créer le modèle mais seulement la représentation du modèle résultant comme par exemple les nœuds dans un arbre de décision ou les classes dans le modèle K-means.

Notre approche diffère, entre autres, de PMML par le fait qu'elle est orientée vers l'aide à l'interprétation de résultats des méthodes de classification (extensible à toutes méthodes de fouille de données) tandis que PMML ne permet qu'une description des paramètres

[4]par exemple la matrice de covariance

de méthodes en vue d'une réutilisation sur d'autres données. Néanmoins, l'utilisation de fichiers PMML dans notre travail permet d'extraire des métadonnées descriptives (pouvant être utiles dans la description des résultats de méthodes de fouille de données). En effet, PMML étant devenu un standard de représentation très populaire, utilisé dans l'API JDM par exemple, il nous semble utile de l'intégrer dans notre processus de traitement.

3.2 Analyse des données symboliques

L'objectif de cette section est d'expliquer, qu'avant la phase de fouille de données proprement dite, il existe d'autres approches permettant de généraliser les données afin d'aider à la compréhension des résultats. Cette approche est celle de l'analyse des données symboliques (ADS). Elle se propose d'aider les utilisateurs à mieux modéliser leurs résultats. Contrairement à l'analyse des données qui ne traite que des données dites « simples », l'ADS se propose de pallier ces limites en utilisant un formalisme adapté aux *données complexes*.

Comme l'explique V. Stéphan dans sa thèse (cf. [Stéphan, 1998]), en analyse des données classiques, l'unité statistique employée est celle de l'individu décrit par un $p-$*uplet* (où p est le nombre de variables). Les entrées se présentent ainsi sous la forme d'un tableau *individus* $\times$ *variables* (Tableau 3.1) où $Y_j(w_i)$ représente la description de l'individu w_i pour la variable Y_j, chaque observation $Y_j(w_i)$ étant atomique.

	Y_1	...	Y_j	...	Y_p
w_1			...		
...			...		
w_i	...	...	$Y_j(w_i)$		
...					
w_n					

Tab. *3.1 - Tableau de données*

Remarquons que si ce type d'observation est bien adapté pour des données simples, il se révèle insuffisant lorsqu'il s'agit de décrire l'*imprécision*[5] concernant la valeur observée sur un individu ou encore la *variation*[6] présente au sein des observations d'une classe d'individus. Dans cette section, nous abordons plus précisément l'aspect qui consiste à résumer les informations d'une base de données à l'aide du formalisme symbolique. Puis nous montrons la particularité de notre approche par rapport à ce type de formalisation.

[5] L'*imprécision* est définie comme l'impossibilité d'observer avec un niveau arbitraire d'exactitude une donnée.

[6] La *variation* exprime le fait que plusieurs réalisations d'expérience pour un même objet et sur une même variable sont différentes.

3.2.1 Objectifs de l'analyse des données symboliques

L'objectif de l'ADS est de proposer un *formalisme bien adapté à la représentation des données complexes*. Les notions d'*imprécision* et de *variation* sont modélisées sans passer par une étape de codification. Concrètement, il s'agit de proposer un nouveau formalisme qui consiste à analyser un ensemble d'individus tout en prenant en compte les données répétées et la variation interne de chacun d'entre eux. Ces objectifs sont résumés à travers les points suivants :

- permettre la description de grands volumes de données sans perte d'information. Ces descriptions doivent néanmoins être compréhensibles par l'utilisateur ;
- pour garder la confidentialité et permettre un traitement aisé sur ces nouvelles données, ces descriptions doivent être écrites dans un formalisme proche de la représentation habituelle d'un ensemble de données ;
- le résumé ainsi obtenu, à travers ces descriptions, doit être une étape intermédiaire pour d'autres analyses sur ces concepts.

Dans cette section, nous évoquons, entre autres, les notions d'objet symbolique et de fonction de généralisation.

3.2.2 Objet Symbolique

Selon E. DIDAY, un *objet symbolique* est une modélisation d'un concept pour représenter son intension qui doit inclure le *mode de calcul de son extension* (***et surtout pas son extension !***). Son *extension* étant l'ensemble des individus dont la description a une « bonne » adéquation à la description d'une classe d'individus, partie de l'extension de ce concept [Diday, 2003], [Diday, 1991]. Ainsi, dans un tableau de données symboliques, nous pouvons avoir dans chaque case des données de types différents suivant les colonnes.

Le modèle de base d'un objet symbolique se compose d'un espace des individus Ω et d'un espace des descriptions D où on suppose qu'il existe : y : une application de Ω dans D,
et T : une application dite de *fusion* ou de *généralisation* de $D \times D$ dans un ensemble dit de généralisation G contenant D et permettant d'associer une description à un couple de descriptions [Schweiger and Sklar, 1983]. La figure 3.9 citée par E.DIDAY [Diday, 2003] exprime cette modélisation par objet symbolique.

Définition 3.1. *Un objet symbolique est un triplet $s = (a, R, d)$ où a dépend de la relation R et de la description d [Diday et al., 2000], [Diday, 1991] avec $d \in D$.* Cette relation binaire R permet de comparer la description d à une autre description de D. La fonction a permet d'évaluer le résultat de la comparaison (à l'aide de R) de la description d'un individu de Ω par rapport à la description donnée d.

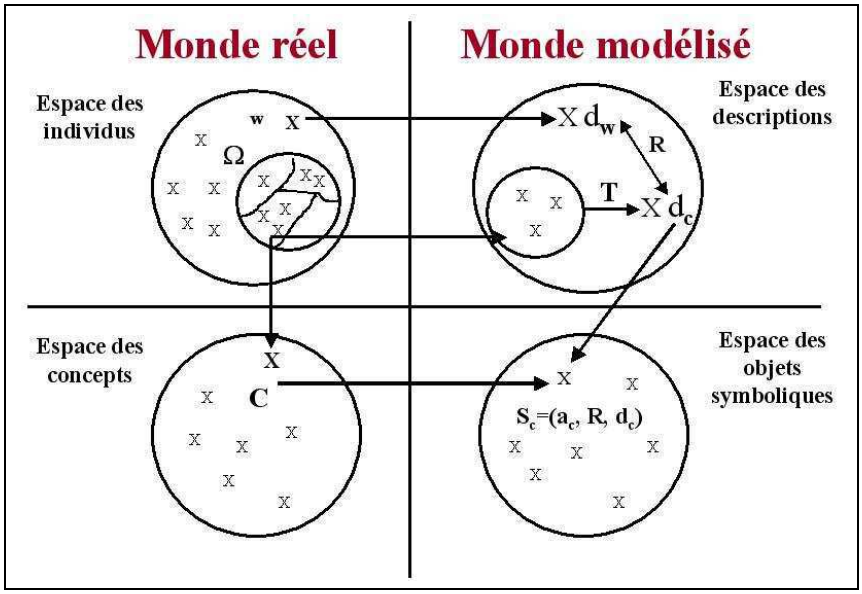

FIG. *3.9 - Modélisation par des objets symboliques de concepts issus d'une classification automatique où : Ω est un échantillon d'individus, $a_c(w) = [y(w)\ R\ d_c]$, et T est un opérateur de généralisation*

Exemple d'objet symbolique : Nous commençons par donner un exemple trivial d'objet symbolique de la forme $a\,(w) = [y\,(w)\ \ R\ d]$ où la seule information sur R et d permet de définir la fonction de reconnaissance a. Ainsi, on définit chaque objet symbolique $s = (a, R, d)$ par $a(w) = [profession(w) \subseteq \{enseignant, medecin\}] \wedge [age(w) \subseteq [25, 35]]$ où $d = (\{enseignant, medecin\}, [25, 35])$
$[d'\ R\ d] = [d'_1\ R_1\ d_1] \wedge \left[d'_2\ R_2\ d_2\right]$ avec $R_1 \equiv R_2 \equiv \subseteq$. Ainsi $a(w) = vrai$ si la profession de w est *enseignant* ou *médecin* et l'âge de w appartient à l'intervalle $[25, 35]$.

Types d'objets symboliques : Il convient d'indiquer qu'il existe deux types d'objets symboliques : les *objets symboliques booléens* définis sur Ω vérifient $a(w) \in \{vrai,\ faux\}$ et les *objets symboliques modaux* où $a(w) \in [0, 1]$

Cas d'une construction d'objets symboliques à partir d'une base de données relationnelle : La nécessité d'introduire des objets symboliques s'explique, entre autres, par le fait que la réalité multidimensionnelle est trop complexe pour entrer simplement dans le carcan d'un tableau de données (même symboliques). En effet, une ligne du tableau de données symboliques peut s'exprimer sous forme d'un objet symbolique mais à l'inverse, les objets symboliques ne peuvent pas être réduits simplement à une ligne d'un tableau de données.
Comme nous l'expliquons en introduction de cette section, pour construire un objet, nous avons besoin d'un opérateur de généralisation, d'un opérateur de comparaison et d'un opérateur d'agrégation.

Méthodes de généralisation : La forme la plus simple consiste à décrire une classe par l'union des modalités atteintes par ses éléments.
Prenons l'exemple de la généralisation de chaque groupe par un processus utilisant une approche non supervisée, la description est obtenue à partir des conjonctions des descriptions de chaque individu du groupe.

$$d_j(G_i) = y_j(\omega_1) \oplus \cdots \oplus y_j(\omega_{ni}) \quad \forall j = 1, \ldots, p \quad n_i = |C_i| \tag{3.1}$$

avec d_j : opérateur de généralisation appliqué sur la variable y_j. En fonction du caractère qualitatif ou quantitatif de la variable, **l'opérateur de généralisation** est égal à :

$$\oplus(\{Y_j(\omega)/\omega \in C_i\}) = \begin{cases} [min\{Y_j(\omega)/\omega \in C_i\}, max\{Y_j(\omega)/\omega \in C_i\}] & \text{si } Y_j \text{ quantitative} \\ \{Y_j(\omega)/\omega \in C_i\} & \text{si } Y_j \text{ qualitative} \end{cases} \tag{3.2}$$

Néanmoins, l'objectif n'est pas de lister toutes les méthodes de généralisation. Nous nous contentons ainsi d'aborder celles expliquées dans [Stéphan, 1998]. Cette méthode consiste à généraliser les données provenant d'une base de données afin de construire des *objets symboliques* qui seront ensuite traités par des méthodes d'analyse de données symboliques. Cette modélisation prendra en compte la variabilité du groupe de données. Et la fonction de généralisation utilisée est celle définie ci-après et qui considère la famille d'application suivante :

$$g : P(\omega) \to D$$
$$\{w_1, \ldots, w_k\} \mapsto d$$

où d est une description qui généralise les vecteurs de données $(Y(w_1), \ldots, Y(w_k))$. Ainsi, nous pouvons dire que d *généralise* les individus $\{w_1, \ldots, w_k\}$. Pour la fonction g, il existe plusieurs choix possibles. La caractéristique commune étant que la généralisation s'effectue sur les *marginales*, c'est-à-dire sur les vecteurs $(Y_j(w_1), \ldots, Y_j(w_k))_{j=1,\ldots,p}$.

Soit $C \in P(\Omega)$ et $d = g(C)$, où $d = (d_1, \ldots, d_j, \ldots, d_p)$:

- $D = P(O_1) \times \cdots \times P(O_p)$: on utilise la fonction $(d = \oplus(\{Y(w) | w \in C\}))$;
- $D = Q_1^{pr} \times \cdots \times Q_p^{pr}$: on prend en compte les fréquences d'observations des individus dans le cas des variables qualitatives. Soit $\Delta_j = O_j$, on a :

$$d_j = q_j \text{ où } q_j(v) = \begin{cases} \frac{card(\{w \in C \mid Y_j(w) = v\})}{card(C)} & \text{si } v \in \oplus(\{Y_j(w) \mid w \in C\}) \\ 0 & \text{sinon} \end{cases} \tag{3.3}$$

La formalisation par des objets symboliques consiste à associer à chaque description $d \in D$ une fonction d'adéquation permettant de trouver l'ensemble des $w \in \Omega$ en relation avec d.

L'opérateur de comparaison : Selon E. DIDAY [Bock and Diday, 2000], suivant la sémantique désirée pour la modélisation, on peut considérer trois grandes formes d'opérateurs de comparaison R entre la description d d'un individu et celle d'une classe d_c :

- la ressemblance calculée à l'aide d'une dissimilarité entre descriptions symboliques [Bock and Diday, 2000] ;
- l'appariement, par exemple l'opérateur d'inclusion dans le cas booléen ;
- la probabilité a posteriori qu'un concept soit satisfait par un individu connaissant la probabilité a priori qu'un individu satisfasse un concept.

L'opérateur d'agrégation : Il permet l'agrégation des résultats obtenus par l'opérateur de comparaison sur chaque variable. On peut citer les $t-normes$, ou d'autres types d'indices dits « flexibles » [Bock and Diday, 2000], comme opérateurs d'agrégation.

3.2.3 Synthèse et discussion

L'objectif de cette section était d'expliquer qu'il existe une approche qui permet de résumer les informations provenant d'une base de données afin de leur rendre moins complexe pour l'utilisateur, d'une part. D'autre part, cette approche va permettre l'application de nouvelles méthodes d'analyse sur ces nouvelles données. Résumer des données provenant de bases de données a des avantages, comme l'explique V. STÉPHAN [Stéphan, 1998] :

- les résumés permettent de prendre en compte la notion de variabilité d'un groupe. En effet, ces résumés ne se limitent pas uniquement aux indicateurs centraux calculés sur les observations d'un groupe de données, ils permettent ainsi de prendre en compte la notion de variabilité de ce groupe ;
- permettre de faire des analyses ultérieures sur les résumés ainsi obtenus.

Cette approche a permis de construire à partir d'une base de données, des données symboliques. Le tableau qui en résulte, muni parfois de règles et de taxonomies, permet de décrire des concepts résumant un vaste ensemble de données. Ensuite, il s'agit d'analyser ce tableau pour en extraire des connaissances par des méthodes d'analyse de données symboliques. En effet, cette approche permet l'élaboration d'une base de connaissances décrites par des assertions et conçue pour être analysée par des méthodes d'analyse de données symboliques.

Comme nous pouvons le constater, le principe de l'ADS consiste à analyser un ensemble d'individus tout en prenant en compte la statistique propre, les données répétées, la variation interne de chacun d'entre eux. Ainsi, à la différence des tableaux de données classiques, les *tableaux de données symboliques* autorisent plusieurs valeurs par case, ces valeurs étant parfois pondérées et liées entre elles par des règles et des taxonomies.
Or dans nos travaux, l'objectif est de permettre aux utilisateurs d'être en possession

d'informations supplémentaires leur permettant de mieux interpréter les résultats de leur classification.

3.3 Classification automatique

Selon [Diday, 1998], la classification automatique est un ensemble de méthodes et d'algorithmes consistant à découper une population d'objets en plusieurs classes, en tenant compte des variables qui les caractérisent et de la mesure de ressemblance choisie. Dans cette thèse, la définition qui nous semble la plus appropriée est celle donnée dans [Jambu, 1978]. Pour JAMBU, la classification est définie comme étant l'ensemble des processus aptes à être exécutés sur ordinateur pour constituer des hiérarchies de classes ou de simples partitions, établies à partir de tableaux de données. Le résultat de cette classification sera des hiérarchies, c'est-à-dire des structures classificatoires. La figure 3.10 (tirée de [Halkidi et al., 2001b]) montre les différentes étapes d'une classification automatique.

Notre objectif est de tenter d'apporter une aide pouvant permettre à l'utilisateur de comparer différentes classes qu'il aura à sa disposition et ce en utilisant des métadonnées extraites au cours du processus de classification.
Nous rappelons que deux tâches sont généralement effectuées à la suite d'une classification automatique [Manco et al., 2004] :

- ***Validation des classes :*** une classification est qualifiée de « valide » si les classes découvertes correspondent à des groupements statistiquement homogènes et séparés [Bertrand and Ben Mufti, 2006], [Ben Mufti, 2006] ;
- ***Interprétation des classes :*** une classification est « interprétable » si on arrive à associer la signification appropriée à chaque classe. Autrement dit, les classes ont une signification pour l'utilisateur et elles sont donc utilisables pour l'objectif pour lequel elles ont été créées [Jambu, 1978].

A travers notre recherche bibliographique, nous avons constaté, là encore, que la majorité des travaux développe des critères statistiques qui sont orientés *validation* plutôt qu'*interprétation* des classes. Dans la plupart des algorithmes de classification, on utilise une évaluation 2D des ensembles de données pour permettre à l'utilisateur de vérifier visuellement la validité des résultats (c'est-à-dire se faire une idée sur la manière dont les algorithmes de classification ont découvert les classes à partir des ensembles de données) [Halkidi et al., 2002].

Il est clair que la visualisation des ensembles de données est une vérification cruciale des résultats de classification [Rouard et al., 1998]. Mais dans le cas de données multidimensionnelles, la visualisation effective de ces données devient alors très difficile. De plus la perception des classes en utilisant les outils de visualisation disponibles est une tâche difficile pour des utilisateurs n'ayant pas l'habitude.

Le processus d'évaluation des résultats de classification est connu sous le nom de *« cluster validity »*. Selon [Theodoridis and Koutroumbas, 1999], [Manco et al., 2004] et

[Gordon, 1999], il existe trois approches pour vérifier la validité d'une classe (critères résumés dans le tableau 3.2).

- ***la première est basée sur les critères externes :*** ceci implique que nous évaluons les résultats d'un algorithme de classification par une structure pré spécifiée, qui est imposée sur un ensemble de données et reflète notre intuition sur la structure de classification des ensembles de données. *Exemple :* le taux d'erreur de l'opération de classification (i.e. le nombre des cas mal classés), indice de Rand, coefficient de JACCARD, indice d'HUBERT, etc. (voir [Manco et al., 2004], [Halkidi et al., 2002] et [Lamiri et al., 2004] pour une description détaillée) ;
- ***la seconde approche est basée sur les critères internes :*** dans ce cas les résultats de classification sont évalués en terme de quantités qui impliquent les vecteurs des ensembles de données eux-mêmes (par exemple la matrice de proximité). Il s'agit d'évaluer les résultats en fonction des quantités calculables à partir des données disponibles. *Exemple :* similarité intra-classe (i.e. compacité) et dissimilarité interclasses (séparation), coefficient de corrélation (pour plus de détail cf. à [Halkidi et al., 2002], [Bertrand and Ben Mufti, 2006]) ;
- ***la dernière approche est basée sur les critères relatifs :*** ici, l'idée de base est l'évaluation d'une structure de classification en la comparant à d'autres schémas de classification, résultant du même algorithme mais avec des paramètres différents. *Exemple :* calcul des critères de compacité et de séparation en modifiant certains paramètres de la classification comme le nombre de classes, les poids des variables, etc. D'amples détails sont donnés dans [Halkidi et al., 2001a].

	Critère Interne	***Critère Externe***	***Critère Relatif***
Description	Les résultats de classification sont évalués uniquement en utilisant les quantités et les caractérisques provenant des ensembles de données.	Les résultats sont évalués suivant une structure pré-spécifiée.	Les résultats sont évalués en les comparant à d'autres résultats utilisant des critères différents.
Exemples	coefficient de corrélation, similarité intra-classe	coefficient de jaccard, indice de rand, ...	modifier le nombre de classes

TAB. *3.2 - Tableau de critères de validation*

Les deux premières approches sont basées sur les tests statistiques et leur principal inconvénient réside dans leur coût de calcul.
Ces différentes approches proposées dans la littérature marchent bien lorsque les classes sont compactes. Or, il existe différentes applications où nous avons des classes avec des

formes arbitraires (données spatiales, médecine, biologie, etc.). Dans ces cas les critères traditionnels de validité (variance, densité et continuité, séparation, etc.) ne sont pas suffisants. C'est pourquoi il est utile d'avoir des critères tenant compte de :

- la qualité intra classe
- la séparation inter classe
- la forme des classes

Une comparaison de la plupart de ces critères a fait l'objet d'une thèse [Toledo, 2005]. Cette question est aussi abordée dans [Sousa and Tendeiro, 2005], [Roth et al., 2002] et [Yatskiv and Gusarova, 2004] pour d'autres algorithmes de classification tels que les algorithmes génétiques [Bolshakova and Azuaje, 2003]. Aussi, YATSKIV avance que dans le processus d'interprétation des classes, les propriétés suivantes sont examinées : la *densité*, la *taille* et la *forme* des classes, la *séparabilité* des classes, la *robustesse* de la classification [Yatskiv and Gusarova, 2004].

3.3.1 Problématique en classification automatique

Le principal objectif de la classification est de définir des partitions ou des hiérarchies de partitions, sur un ensemble d'éléments comparables deux par deux, qui respectent les ressemblances entre eux. Les éléments à classer peuvent être des objets ou des variables d'un ensemble de données. Dans chaque étape les classes les plus ressemblantes sont regroupées selon un critère prédéfini.

Selon [Fayyad et al., 1996], la classification est l'une des tâches les plus utiles dans le processus de fouille de données pour la découverte de groupes et l'identification de distributions et de patterns intéressants dans des données sous-jacentes. Ce processus de classification est perçu comme un processus non supervisé puisqu'il n' y a pas de classes prédéfinies ni d'exemples montrant le type de relations souhaitées valides sur les données [Berry and Linoff, 1996]. La classification est donc une technique qui permet de trouver un partitionnement en k classes ayant un sens ou du moins similaires. Ce nombre k de classes peut être donné ou découvert. Cette technique de fouille de données est composée de plusieurs méthodes pouvant être regroupées en trois grandes catégories : les méthodes de partitionnement, les méthodes hiérarchiques, les méthodes par voisinage dense (cf. section 3.3.4 pour plus de détails).

Les techniques de classification sont utilisées dans plusieurs domaines afin d'aider à l'identification de caractéristiques méconnues jusqu'alors par les utilisateurs. Ainsi, ces techniques sont utilisées pour identifier les caractéristiques démographiques ou psychographiques des consommateurs avec des achats historiques similaires, ou isoler les différences entre les groupes de produits. Les gestionnaires peuvent ainsi étudier les classes individuelles des consommateurs ou des produits en détail et ce en vue de maximiser les résultats pour des futures stratégies marketing. Néanmoins l'une des problématiques les

moins explorées à ce jour est celle relative à l'automatisation du processus d'interprétation des résultats de ces techniques. Or, il est évident que sans cette dernière étape, il est impossible de tirer profit de l'apport de ces méthodes.

3.3.2 Étapes de la classification automatique

La figure 3.10 résume les différentes étapes de la classification automatique. Voici une explication plus détaillée de ces différentes étapes :

- **la sélection des données :** le but est de sélectionner proprement les données sur lesquelles la classification doit être réalisée. Ainsi le prétraitement des données peut être nécessaire avant leur utilisation dans la tâche de classification.
- **l'algorithme de classification :** cette étape est relative au choix d'un algorithme pouvant engendrer la définition d'un bon schéma de classification pour un ensemble de données.
  Dans cette étape on peut être amené à choisir la mesure de proximité (mesure qui quantifie la manière dont deux données sont « similaires ») ou le critère de classification.
- **la validation des résultats :** l'exactitude des résultats d'algorithme de classiification est vérifiée en utilisant des techniques et critères appropriés.
- **l'interprétation des résultats :** dans beaucoup de cas, les experts du domaine doivent intégrer les résultats de la classification avec d'autres évidences expérimentales afin de rédiger la bonne conclusion. Ce qui explique la difficulté de formaliser et d'automatiser cette étape.

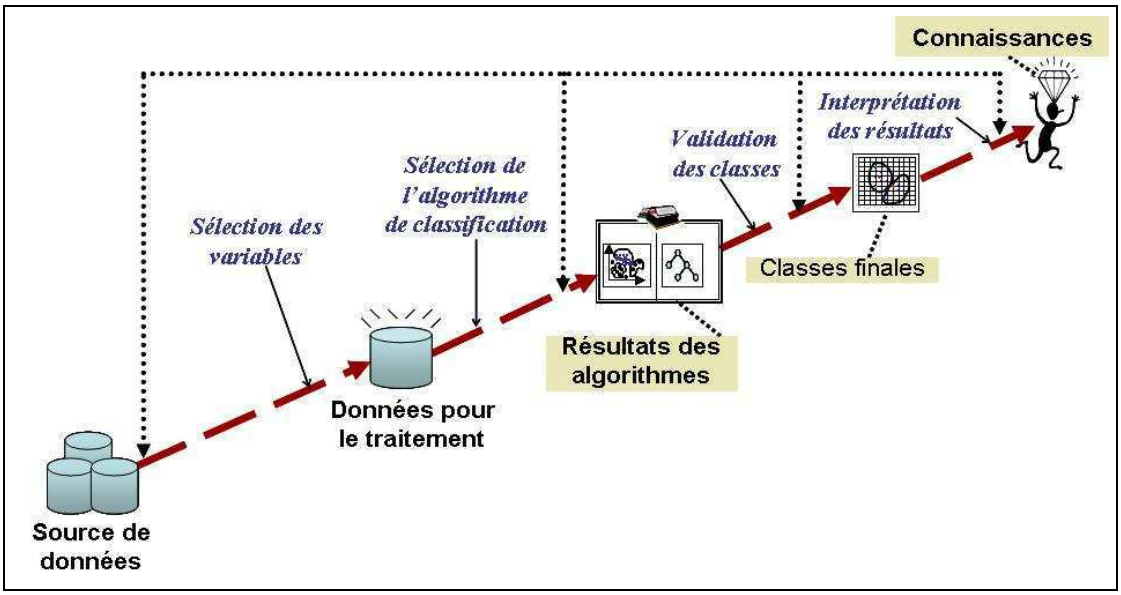

FIG. *3.10 - Étapes d'une classification automatique*

3.3.3 Qualités et Propriétés d'une bonne méthode de classification

D'une manière générale, la qualité d'une classification dépendra de la mesure de similarité utilisée et de l'implémentation de cette mesure.
Il convient de rappeler que la définition de la similarité entre objets dépendra du type des données et du type de la similarité recherchée.
En ce qui concerne la tâche d'interprétation des classes, les techniques les plus utilisées sont des techniques d'exploration visuelle combinées avec les statistiques descriptives. La visualisation des résultats est généralement interactive, ce qui permet d'incorporer les connaissances expertes dans la tâche d'interprétation.

Dans ses travaux, P. ADRIAANS dresse une liste non exhaustive de critères que devrait vérifier une bonne méthode de classification. Il s'agit des critères avancés dans [Adriaans and Zantinge, 1996] :

1. *la scalabilité :* beaucoup de méthodes donnent des résultats satisfaisant sur des petits ensembles de données mais dès que l'ensemble devient important, leurs résultats se détériorent ;
2. *la capacité à traiter différents types de données* ;
3. *la capacité à découvrir des formes de classes de manière arbitraire :* en effet beaucoup de méthodes déterminent des classes en se basant sur une distance prédéfinie, ce qui entraîne des classes avec des tailles et des densités similaires. Or, une classe peut avoir une forme très différente des autres ;
4. *la méthode doit poser un minimum de contraintes* concernant la connaissance du domaine pour déterminer les paramètres. En effet, certaines méthodes requièrent que l'utilisateur détermine certains paramètres dans l'analyse (par exemple le nombre de classes souhaité). Or, les résultats de classification peuvent être très sensibles à ces paramètres. De plus, ces paramètres sont souvent très difficiles à déterminer par un utilisateur peu avant ;
5. *la capacité à traiter des données bruitées* ;
6. *la non sensibilité à l'ordre* auquel les données sont extraites. En effet, le même ensemble de données présenté de diverses manières, pour certaines catégories de méthodes, peut produire des classes complètement différentes ;
7. *la dimension des attributs* ;
8. *la classification sous contraintes.*

3.3.4 Types de méthodes de classification

Les méthodes dites de classification non supervisée, par exemple la méthode des k moyennes et ses variantes, résolvent une tâche dite non supervisée, c'est-à-dire qu'elles ne nécessitent aucune information sur les données. En effet, la classification peut être utile

pour découvrir une structure cachée qui permettra d'améliorer les résultats de méthodes d'apprentissage supervisé.

En choisissant une bonne notion de distance, ces méthodes peuvent s'appliquer à tout type de données. De plus, ces méthodes sont faciles à implémenter : en effet elles ne nécessitent que peu de transformations sur les données (excepté les normalisations de valeurs numériques). Par contre les performances de ces méthodes (la qualité des groupes constitués) sont dépendantes, en partie, du choix de la mesure de similarité. Ce qui est une tâche délicate surtout lorsque les données sont de types différents. De plus ces méthodes sont sensibles au choix des paramètres, en particulier, le choix du nombre k de groupes à constituer. Un mauvais choix de k produit de mauvais résultats. Ce choix peut être fait en combinant différentes méthodes, mais la complexité de l'algorithme augmente.

Selon HALKIDI dans [Halkidi et al., 2001b], les algorithmes de classification peuvent être classés suivant :

- le type de données en entrée de l'algorithme ;
- le critère de classification définissant la similarité entre les données ;
- la théorie et les concepts fondamentaux sur lesquels les techniques d'analyse de classification sont basées.

Ainsi, ces méthodes peuvent être des méthodes de *partitionnement*, des méthodes *hiérarchiques*, des *méthodes basées sur la densité* ou des *méthodes basées sur les grilles ou réseaux* :

1. ***Classification par partitionnement :*** Les techniques de partitionnement produisent une partition sur l'ensemble des données. La figure 3.11 montre un exemple de partition à deux classes. Bien évidemment les classes ne sont pas aussi bien séparées comme c'est le cas ici. Comme nous l'expliquons dans les sections précédentes, le plus difficile est de donner à chaque classe sa signification.

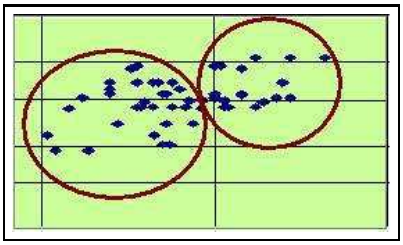

FIG. *3.11 - Exemple de partition en deux classes*

2. ***Classification hiérarchique :*** Ces techniques produisent une série de partitions imbriquées.
   Elles peuvent être de deux types :
   - *agglomératives :* on commence par les objets et on forme des groupes jusqu'à ce que tous les groupes appartiennent à une seule classe ;

– *divisives :* il s'agit de partir d'une classe de départ formée par tous les objets et à la scinder en de nouvelles classes, plus petites, et ce jusqu'à ce que chaque objet appartienne soit à une classe ou jusqu'à une condition d'arrêt [Chavent et al., 2002].

Pour ces méthodes, on n'exige pas de connaître le nombre k de classes. Elles utilisent une matrice de distances comme critère de classification.

Une condition de terminaison peut aussi être utilisée (par exemple le nombre de classes). La hiérarchie de classes est un arbre de classes appelé ***dendogramme***. Les feuilles de l'arbre représentent les objets et les nœuds intermédiaires représentent les classes.

3. ***Classification basée sur la densité de voisinage :***

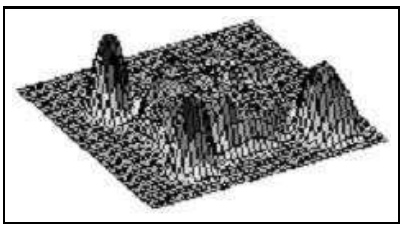

FIG. *3.12 - Exemple de classification par densité*

Contrairement aux autres méthodes, elle se base sur la densité afin de découvrir des classes ayant des formes complexes (un exemple est donné dans la figure 3.12 tirée de [Han and Kamber, 2000]).

Pour ces méthodes, l'utilisation des mesures de similarité (distance) est moins efficace que l'utilisation de la densité de voisinage. En effet, minimiser la distance interclasses n'est pas toujours un bon critère pour reconnaître des formes (applications géographiques, reconnaissance de formes, tumeur, etc.)

4. ***Classification basée sur les réseaux (ou grilles) :*** Ces méthodes permettent de traiter de grands ensembles de données. En effet, ces méthodes ne dépendent pas des données mais uniquement du nombre de cellules dans chaque dimension et dans chaque espace quantifié (un exemple tiré de [Han and Kamber, 2000] est donné dans la figure 3.13).

Le choix de la méthode de classification étant fortement dépendant du type des données et des objectifs fixés.

3.4 Approches d'interprétation en classification automatique

L'interprétation peut être vue comme une évaluation de la qualité de la classification et une description des classes obtenues.

La classification obtenue est liée aux variables choisies dans la description des individus. C'est ce qui explique que certaines stratégies d'interprétation sont très liées aux types de variables. Différents types de variables, suivant leur rôle dans le processus de classification,

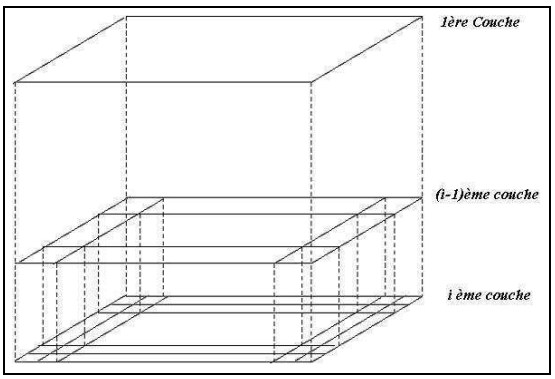

FIG. *3.13 - Exemple de classification par grille*

peuvent être distingués. Nous avons les *variables actives* (celles sur lesquelles sera basée la classification des individus), les *variables illustratives ou supplémentaires* qui serviront à décrire les classes constituées. Ce sont les variables qui décrivent les caractéristiques de l'individu (variables sociodémographiques par exemple).

Dans cette section, nous abordons différentes stratégies d'interprétation existantes et ensuite nous expliquons la stratégie que nous adoptons dans cette thèse et les raisons qui nous ont poussé à l'adopter.

3.4.1 Interprétation du rôle des individus et des variables :

Cette stratégie consiste à examiner, pour chaque classe, son effectif, son diamètre (distance entre les deux points les plus éloignés), sa séparation (distance minimum entre la classe considérée et la classe la plus proche) et le numéro de la classe la plus proche. On peut examiner aussi les identités des individus les plus proches du centre de gravité de la classe ou « parangons » et/ou les identités des individus les plus éloignés du centre de gravité de la classe ou « extrêmes ». Les critères d'interprétation utilisés dans cette section sont ceux définis dans [Jambu, 1978] [Celeux et al., 1989] et qui sont relatifs à la décomposition de *l'inertie totale* T en *inertie intra-classe* W et en *inertie interclasse* W. Nous reprenons les formules 1.1, 1.2, 1.3 et 1.4 abordées dans la section 1.1 :

$$T = B + W$$

$$T_j = B_j + W_j$$

$$T_k = B_k + W_k$$

$$T_j^k = B_j^k + W_j^k$$

Indices relatifs à l'interprétation des variables Dans ce paragraphe, nous abordons l'aspect de l'interprétation des classes basée sur les variables.

- *Indice général :* la part d'inertie conservée ou expliquée en assimilant les individus aux k centres de gravité s'écrit : $R = \frac{B}{T}$.

Plus R est fort, plus la partition constitue une bonne représentation des individus. Si $R = 1$ cela signifie que le nuage des individus se réduit aux nuages des k centres de gravité des classes.

- *Contribution des variables :* pour chaque variable j, on définit un indice analogue à R.

$$COR(j) = \frac{B_j}{T_j} \tag{3.4}$$

Cet indice représente la part d'inertie de la variable j pris en compte par la partition. Il mesure ainsi le pouvoir discriminant de la variable par rapport à la partition. Il est intéressant de comparer ses valeurs avec R.
En effet si $COR(j) >$ (respectivement $<$)R, la variable j est plus (respectivement moins) discriminante que la moyenne puisque R représente le pouvoir discriminant « moyen » des variables vis-à-vis de la partition P.

- *Contribution relative à l'inertie interclasse et/ou intra-classe :* il s'agit de la contribution relative de la variable j à l'inertie interclasse et/ou intra-classe de la partition.

$$CTR(j) = \frac{B_j}{B} \text{ avec } \sum CTR(j) = 1. \tag{3.5}$$

Les deux derniers indices ont tendance à varier dans le même sens [Celeux et al., 1989]. Ceci est particulièrement vrai dans le cas où la partition a des classes de volume analogue. Le cas, rare, où on peut avoir $COR(j)$ faible (respectivement fort) et $CTR(j)$ fort (respectivement faible) est intéressant : cela signifie que la variable j malgré une forte (respectivement faible) contribution à l'inertie de P est peu (respectivement très) discriminante.

3.4.2 Interprétation des classes par les variables :

Cette stratégie consiste à calculer, pour chacune des variables, un critère mesurant sa pertinence dans l'interprétation de la classe. Par exemple, la variable *prix* peut être un critère fort pour une classe tandis que la variable *âge* peut être un critère faible pour cette même classe. On devra alors se poser la question de savoir si tous les éléments de la classe ont des valeurs particulières de cette variable (condition nécessaire d'appartenance à la classe) ? Ou encore est-ce que certaine(s) valeur(s) de cette variable ne se rencontrent que dans cette classe (condition suffisante d'appartenance à la classe) ?

Indices relatifs à l'interprétation des classes par des variables Dans ce paragraphe, il s'agit d'interpréter les classes du point de vue des variables. En effet, pour chaque variable j et chaque classe k, on définit les critères suivants :

- *le pouvoir discriminant de la variable :* $COR(j,k) = \frac{B_j^k}{T_j}$ représente le pourcentage

du pouvoir discriminant de la variable j pris en compte par la classe k avec

$$\sum_{k=1}^{k} COR(j,k) = COR(j) \tag{3.6}$$

Plus cet indice est fort, plus la variable j aura un comportement homogène sur les éléments de la classe k. Examiner cet indice est particulièrement intéressant pour les variables à fort pouvoir discriminant : il indique la répartition entre les classes de ce pouvoir discriminant.

- $CTR(j,k) = \frac{B_j^k}{B_k}$ qui représente la *contribution relative de la variable* j et de la classe k à l'inertie interclasse. Cet indice est complémentaire à l'indice précédent : il permet de déterminer les variables qui caractérisent le plus chaque classe.

Ces deux derniers indices s'interprètent comme les *contributions relatives et absolues* de l'analyse factorielle. D'autres indices complémentaires peuvent être utilisés pour affiner l'interprétation des classes. Dans ces travaux l'interprétation d'une classe est similaire à la recherche d'informations sur son *hétérogénéité*, sur son *manque de cohésion*, sur son *isolation* ou sur sa *séparation* du reste des classes [Bertrand and Ben Mufti, 2006].

3.4.3 Autres approches d'interprétation :

D'autres travaux ont été réalisés dans le cadre du processus d'aide à l'interprétation. Il s'agit des travaux de JAMBU [Jambu, 1978], qui définit l'aide à l'interprétation comme étant toute technique ou calcul qui permet d'éprouver le bien fondé des classes obtenues par une quelconque méthode de classification automatique. Pour permettre une meilleure interprétation des résultats, il a proposé d'appliquer des représentations différentes d'une même métrique. Chaque type de représentation (arborescente, en partition, en classes empiétantes, en nuage de points, etc.) possède des particularités dues à la méthode elle-même et non forcément aux données qu'on prétend analyser.
Il part du principe selon lequel qu'un utilisateur expert se trouvant en présence d'une représentation arborescente, voire d'une simple partition, s'interroge sur la confiance qu'il doit accorder à la séparation entre les classes. Cet utilisateur s'interroge aussi sur la signification de la hauteur à laquelle on place ces classes d'une part. D'autre part, un nombre important de questions peuvent se poser à lui : Peut-on caractériser ces classes par les éléments non pris en considération dans la classification ? A quel niveau faut-il s'arrêter dans l'interprétation des classes ?

Indices relatifs à l'interprétation des classes Contrairement au paragraphe précédent, celui-ci s'intéresse aux aspects liés aux critères statistiques des classes. Ainsi, il convient de rappeler que chaque classe est décrite par un ensemble de critères définis ci-après :

- $T(k) = \frac{T_k}{T}$ représente le pourcentage d'inertie extrait par la classe k,
- $B(k) = \frac{B_k}{B}$ représente la contribution relative de la classe k à l'inertie interclasse,
- $W(k) = \frac{W_k}{W}$ représente la contribution relative de la classe k à l'inertie intra-classe.

On a

$$\sum_{k=1}^{k} T(k) = \sum_{k=1}^{k} B(k) = \sum_{k=1}^{k} W(k) = 1 \tag{3.7}$$

$B(k)$ indique la position de la classe k par rapport au centre de gravité global. Plus cet indice est fort, plus la classe occupe une place excentrée.
$W(k)$ indique la concentration de la classe k indépendamment de sa position. Il doit être examiné en regard de la taille de la classe. Une valeur de $W(k)$ faible sera d'autant plus significative de la concentration de la classe que son cardinal sera élevé.

Nous voyons clairement les difficultés auxquelles sont confrontés les utilisateurs dès qu'il s'agit de trouver une méthode générique d'interprétation de résultats d'une méthode de fouille de données et plus particulièrement de la classification. D'autres types de méthodes d'interprétation ont été mis en œuvre, il s'agit par exemple de la méthode d'interprétation d'une classification hiérarchique pour expliquer une variable [Tallur, 1982]. Aussi, il convient de rappeler que l'interprétation des résultats des méthodes de classification dépend fortement du type des données et des objectifs fixés. C'est ce qui explique que dans le cadre de cette thèse, notre choix s'est porté sur la prise en compte de ces diverses approches.

3.5 Synthèse du Chapitre 3

Dans ce chapitre, nous avons abordé le processus d'ECD (cf. section 3.1.1) et nous nous sommes intéressés à l'étape de fouille de données (cf. section 3.1.3) et plus précisément aux méthodes de classification automatique (cf. section 3.3). En effet, l'objectif de ce chapitre était de présenter les problématiques liées à la complexité de trouver une approche générique d'interprétation des résultats de méthodes de fouille de données en général.

Ainsi, nous avons évoqué différentes approches d'interprétation (cf. section 3.4) et montré leurs limites dans le cadre de ce travail. En effet, le but de ce travail n'est pas de chercher à créer une nouvelle approche d'interprétation mais de trouver une approche générique pouvant être utilisée par toute méthode de classification et bien au-delà.

A travers cet état de l'art, nous avons montré que, malgré la multitude et la diversité des méthodes d'interprétation, toutes ces méthodes pouvaient être classées suivant trois (3) axes : *l'axe des individus*, *l'axe de la structure classificatoire* et *l'axe des variables*. Ces différents points seront abordés dans le Chapitre 6 consacré à notre approche de modélisation du processus d'interprétation.

Dans la section 3.1.5 de ce chapitre, nous avons abordé le langage PMML et sa place dans le processus de fouille de données. Par ailleurs, nous avons montré les différences fondamentales existantes entre ce langage et notre approche.

Chapitre 4

Métadonnées et Ontologie

Introduction

DANS ce chapitre, nous abordons les concepts de *métadonnées* et d '*ontologie* ainsi que leur utilisation dans le processus de l'extraction de connaissances à partir de données. Dans la section 4.1, nous expliquons la problématique liée au concept d'ontologie. Nous présentons quelques définitions sur ce terme (§ 4.1.1), son importance (§ 4.1.2), les étapes de sa mise en œuvre (§ 4.1.3). Puis, nous abordons le processus de création d'une ontologie dans le § 4.1.4 avant d'évoquer les langages et outils de représentation des ontologies dans le § 4.1.5. Enfin, nous abordons dans les § 4.1.6 et § 4.1.7 les différences existantes entre les concepts d'ontologie, de vocabulaire contrôlé, de thésaurus et de taxonomie d'une part et la typologie d'ontologie d'autre part.

Dans la section 4.2, nous présentons quelques définitions sur le concept de *métadonnées* (cf. § 4.2.1) ainsi que les typologies et les classifications des utilisateurs de métadonnées dans le domaine des statistiques (§ 4.2.3 et § 4.2.4). Mais avant, nous présentons au § 4.2.2, la place des métadonnées dans le processus de capitalisation de la connaissance.

Dans la section 4.3, nous présentons les rares travaux utilisant l'incorporation de métadonnées et/ou d'ontologies dans les étapes du processus d'ECD.

4.1 Ontologie

Introduction

D'une manière générale, les ontologies sont construites dans le but de permettre le partage et la réutilisation d'information tout en sachant que la sémantique est indépendante de l'utilisateur et du contexte [Cannataro and Comito, 2003].
La création d'une ontologie permettra de :

- nommer, identifier un domaine ;
- définir un vocabulaire applicable sur ce domaine ;
- définir les règles d'utilisation de ce vocabulaire (relations entre les mots) : hiérarchisation (classe/sous-classes/instances), relations structurelles (container) et/ou fonctionnelles, etc.

4.1.1 Définitions

Dans cette section, nous essayons de donner un éventail de définitions utilisées pour qualifier le concept d'ontologie.

Définition 4.1. En Intelligence Artificielle, la définition communément admise est celle énoncée par T. GRUBER qui définit *l'ontologie comme une spécification explicite d'une conceptualisation [Gruber, 1993]*. De ce point de vue, la construction d'une ontologie interviendrait après le travail de conceptualisation. Cette démarche de conceptualisation consiste à identifier, au sein d'un corpus, les connaissances spécifiques au domaine de connaissances à représenter.

Cette définition a fait l'objet d'une précision de la part de B. STUDER qui voit ainsi une ontologie comme la spécification formelle et explicite d'une conceptualisation partagée [Studer, 1998]. Dans cette définition, il convient de mesurer la portée de chaque terme utilisé. Ainsi, le terme *« spécification explicite »* indique qu'une ontologie est un ensemble de concepts, de propriétés, d'axiomes, de fonctions et de contraintes explicitement définis. Le terme *« formel »* précise que cette conceptualisation doit pouvoir être comprise et interprétée par une machine. Le terme *« partagé »* précise l'aspect consensuel du vocabulaire employé. Ce qui suppose que l'on doit assurer une réutilisation de la formalisation choisie. Enfin, le terme *« conceptualisation »* implique également l'aspect intentionnel, lié à un objectif de réalisation.
N. GUARINO affine cette définition dans [Guarino and Giaretta, 1995], en considérant qu'une ontologie est une spécification partielle et formelle d'une conceptualisation (introduction du niveau ontologique). Cette remarque fait apparaître les différents rôles attribués aux ontologies suivant les domaines (pour plus de détails, voir [Fürst, 2002], [Gruber, 1993], [Gruber et al., 1995] et [Guarino and Giaretta, 1995]).

Définition 4.2. L'ontologie peut aussi être définie comme étant *un outil permettant de fournir une conceptualisation sommaire d'une information et un ensemble de termes à utiliser dans cette représentation [Parekh et al., 2004].*

Définition 4.3. *Une ontologie est un ensemble de concepts clairement explicités et un ensemble de relations entre ces concepts [Chandrasekaran et al., 1999].* Une ontologie peut donc être représentée avec une taxonomie (arbre d'héritage) des concepts.

Comme nous l'avons dit précédemment, une ontologie doit présenter les fonctions (ou caractéristiques) du domaine qu'il représente. Elle doit, en outre, classifier les principaux concepts du domaine et rendre évidentes les relations et les contraintes entre ces concepts. Les ontologies sont des représentations de connaissances, contenant des termes et des énoncés qui spécifient la sémantique d'un domaine de connaissance donné dans un cadre opérationnel fixé.
Dans cette section, nous abordons les thèmes liés à la modélisation, à la création, à la représentation et aux rôles des ontologies ainsi que les outils permettant leur création.

4.1.2 Importance d'avoir une ontologie

Avant d'évoquer l'importance que peut avoir l'utilisation des ontologies, nous estimons qu'il est utile de donner un aperçu sur les rôles et les fonctions que peut jouer une ontologie. Il s'agit de permettre la communication entre les systèmes, entre les utilisateurs et les machines ou entre simples utilisateurs. Ces rôles sont décrits dans les travaux de [Gruninger and Lee, 2002]. Leur utilisation permettrait aussi l'inférence, la réutilisation et l'organisation de la connaissance.
En prenant le domaine de la logique de programmation, les ontologies y jouent, selon [Miller, 2000], deux fonctions principales :

- fournir un moyen de visualisation d'un domaine et d'organisation de l'information ;
- définir un vocabulaire commun en expliquant la signification des termes et des relations entre ces termes.

Comme nous pouvons le constater, une ontologie permet de clarifier la structure de la connaissance [Chandrasekaran et al., 1999]. Ainsi, l'ontologie forme le coeur de tout système de représentation de connaissances pour un domaine. En effet, sans les conceptualisations de la connaissance d'un domaine, il est difficile d'avoir un vocabulaire pour la représenter.

Pour créer une ontologie il semble donc primordial d'effectuer une bonne analyse du domaine en clarifiant les terminologies employées pour cette création. Et l'utilisation de ces ontologies s'explique selon [Parekh et al., 2004], par les facteurs suivants :

- l'ontologie peut être construite pour fournir un vocabulaire (partagé et commun) utilisé dans la description d'un ensemble de données et permettre ainsi de définir un vocabulaire compris et partageable par tous ;

- l'ontologie peut fournir un schéma conceptuel pour tout type de données sans se soucier de son format, de sa structure ou de sa taille ;
- l'ontologie peut être conçue pour comprendre sémantiquement le contenu et la structure des données contenues dans une base de données ;
- l'ontologie peut être utilisée pour aider les fournisseurs de données à entrer les métadonnées dans un format sémantiquement valide ;
- l'interopérabilité entre les ensembles de données hétérogènes peut être atteinte en utilisant une ontologie partagée ;
- l'ontologie peut être vue comme étant le modèle de représentation de connaissances le plus avancé.

Pour conclure cette section, il convient de rappeler que la construction d'une ontologie doit se faire sur un domaine de connaissances bien délimité par un objectif opérationnel clair, et portant sur des connaissances objectives dont la sémantique puisse être exprimée rigoureusement et formellement.

4.1.3 Étapes de mise en œuvre d'une ontologie

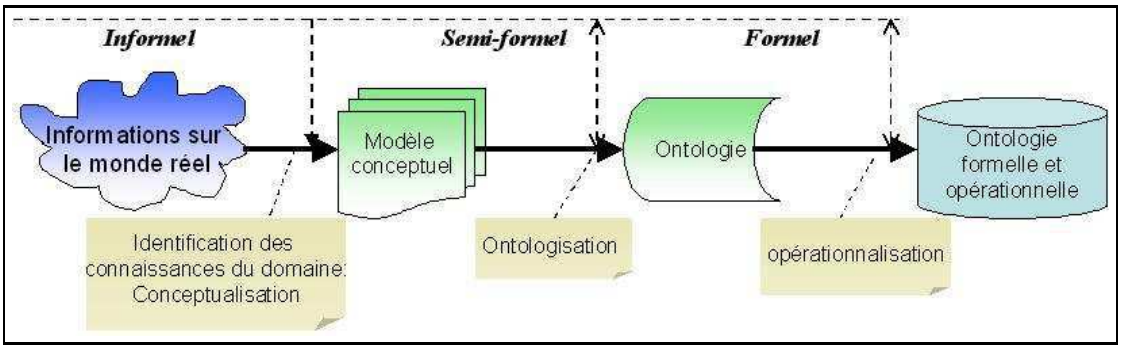

FIG. *4.1 - Étapes de mise en œuvre d'ontologie*

Pour mettre en place une ontologie, diverses démarches peuvent être entreprises par les concepteurs. Cependant, quelle que soit la démarche, il convient de respecter quelques principes. Ainsi, on commencera par identifier les connaissances contenues dans un corpus représentatif du domaine. Ce processus porte le nom de *conceptualisation*. Ensuite, une modélisation semi-formelle du domaine sur lequel portera l'ontologie s'impose. Ce processus porte le nom d'*ontologisation* [Kassel et al., 2000]. De plus, cette ontologie doit être traduite dans un langage formel et compréhensible par la machine.
Le langage de représentation de l'ontologie doit donc permettre de représenter et de manipuler les connaissances modélisées par l'ontologie. Ce processus de traduction est appelé *opérationnalisation*. Ce processus est détaillé dans la figure 4.1 et pour plus de détails, se référer à [Fürst, 2002], [Gandon, 2002].

La séparation de ces deux processus permettra d'avoir des ontologies pouvant être utilisées comme des composants logiciels dans des systèmes différents. Des travaux relatifs au cycle de vie des ontologies, inspirés du génie logiciel, ont été proposés par [Dieng et al., 2001]. Il convient aussi de noter que le processus de mise en œuvre d'une ontologie n'est pas linéaire. Ainsi, il est possible de revenir sur des étapes déjà validées en amont pour bâtir une ontologie opérationnelle adaptée à des besoins spécifiques.

4.1.4 Processus de création d'ontologie

Rappelons que le processus de création d'une ontologie est souvent très complexe et exige un certain consensus entre les différents acteurs concernés par sa création et son utilisation [Farquhar et al., 1997].
En effet, la construction d'une ontologie consiste d'une part, à identifier les concepts d'un domaine de connaissance et la sémantique qui leur est associée. Il s'agit aussi de représenter cette sémantique de façon indépendante des diverses applications dans lesquelles pourra être utilisée l'ontologie, d'autre part.
Pour sa création, plusieurs méthodes ont été définies [Corcho et al., 2001], [López, 1999]. Dans cette section, nous abordons la phase de conception qui est celle relative à la définition des objectifs opérationnels d'une ontologie. Ensuite, nous expliquons de manière plus détaillée la phase proprement dite de la création d'ontologie. Enfin, il convient de signaler que, lorsque les ontologies sont construites automatiquement a partir de texte, des problèmes de conception subsistent (cf. [Roche, 2007] pour des détails).

Conception d'ontologie

Pour que la conceptualisation d'un domaine se fasse de manière non ambiguë, il est nécessaire de préciser les objectifs de celle-ci. En effet, un même terme peut désigner deux concepts différents dans deux contextes différents [Bachimont, 2000]. Il est donc impossible de dissocier la représentation des connaissances d'un domaine et la modélisation des traitements que l'on souhaite leur appliquer. Ce qui signifie que toute tâche de modélisation de connaissances doit se faire dans un domaine précis avec un but précis, condition nécessaire à l'unicité de la sémantique associée aux termes du domaine.
Selon [Holsapple and Joshi, 2002], la conception d'une ontologie est dictée par un certain nombre de principes qu'il détaille dans dans ses travaux. De plus, il avance cinq approches utilisées dans la conception d'une ontologie pour un domaine : *l'inspiration* (un point de vue individuel sur le domaine), *l'induction* (cas spécifique dans le domaine), *la déduction* (principes généraux sur le domaine), *la synthèse* (ensemble d'ontologies existantes, chacune fournissant une caractérisation partielle du domaine) et *la collaboration* (des points de vue multiples avec possibilité de couplage entre ontologies). Bien entendu ces approches peuvent être mélangées.

Méthodes de création d'ontologie

Bien qu'aucune méthodologie générale n'ait pour l'instant réussi à s'imposer, de nombreux principes et critères de construction d'ontologies ont été proposés. Ces méthodologies peuvent porter sur l'ensemble du processus de construction. C'est le cas de METHONTOLOGY, élaborée en 1998 par FERNANDEZ dans [Fernández-Lòpez et al., 1997] et qui couvre tout le cycle de vie d'une ontologie. Nous pouvons citer les travaux de USCHOLD et de GRUNINGER [Uschold, 1995], [Grüninger and Fox, 1995] proposant une méthodologie générale inspirée du domaine de la gestion des entreprises. Ou encore la modélisation proposée dans le cadre des Web Bases [Atzeni et al., 1998].
D'autres méthodologies se focalisent sur une des étapes du processus explicitées en section 4.1.3. Celle présentée dans [Aussenac-Gilles et al., 2000] insiste sur l'étape de conceptualisation par l'analyse d'un corpus textuel. La méthodologie ONTOSPEC propose une aide à la structuration des hiérarchies de concepts et de relations durant la phase d'ontologisation [Kassel, 2002]. C'est également le cas des critères de classification des propriétés proposés par GUARINO dans [Guarino and Welty, 2000a]. Nous pouvons aussi trouver des détails sur d'autres méthodes de construction des ontologies dans [López, 1999] ainsi qu'un état de l'art sur ces méthodes dans [Ben Mustapha et al., 2006].
Il convient de rappeler que la construction d'ontologies peut être réalisée par des approches utilisant des méthodes manuelles. C'est par exemple le cas de la méthode utilisée dans CYC [Douglas and Guha, 1990], dont la principale étape de construction est l'extraction manuelle de la connaissance commune de différentes sources en est une illustration. Une vue d'ensemble des différentes méthodologies de construction d'ontologie est fournie dans [López, 1999]. D'autres travaux relatifs à l'aspect de la construction semi-automatique d'une ontologie sont réalisés dans [Fortuna et al., 2005]. Citons aussi les travaux de [Litvak et al., 2005] qui présente une nouvelle méthode pour la classification de documents Web basée sur une ontologie, dont le principal apport est le traitement de l'ontologie basée sur quatre langues.

Dans cette thèse, nous adoptons la méthode dite de la « méthodologie d'entreprise » dont les étapes sont les suivantes :

1. ***Définition du domaine et de la portée de l'ontologie :***
   Dans cette thèse, il s'agit du domaine de la fouille de données. Pour construire une ontologie consistante, il est nécessaire d'identifier à quoi l'ontologie va servir et à quels types de questions l'ontologie devra répondre. Ainsi, notre ontologie suit une caractérisation des méthodes de classification automatique.

2. ***Identification des concepts clés et des propriétés :***
   Il s'agit d'établir ce que nous voulons dire sur ces termes et quelles propriétés ces termes doivent avoir. Il existe deux stratégies : la stratégie *top-down* et la stratégie *bottom-up*. La première, adoptée dans ce travail, consiste à définir tous les concepts du domaine et à les spécialiser au fur et à mesure. En effet, nous avons commencé

par définir les classes des concepts génériques de la classification pour les spécialiser par la suite.

La catégorisation des méthodes de fouille de données se fait suivant les critères suivants [Cannataro and Comito, 2003] :

- les tâches de fouille de données réalisées par le logiciel. Par exemple les résultats produits ou la connaissance découverte ;
- le type de méthodologie que le logiciel utilise dans son processus de fouille ;
- le type de sources de données utilisées par le logiciel ;
- le niveau d'interaction requis avec l'utilisateur. Par exemple vérifier si le processus de fouille est complètement autonome ou s'il requiert une intervention de l'utilisateur.

3. ***Définition d'une hiérarchie de concepts à travers les relations taxonomiques :***

   Les taxonomies sont utilisées pour organiser la connaissance ontologique dans le domaine et ce en utilisant deux types de relations :

   - spécialisation *« est-un »* : par exemple, nous pouvons dire qu'un *carré* **est une** *figure géométrique* ;
   - généralisation *« partie-de »* : par exemple, au niveau du corps humain, on peut dire que la *main* est une **partie du** *corps humain.*

4. ***Construction des axiomes :***

   Les axiomes fournissent un moyen de représentation des informations sur les concepts. Par *exemple* les contraintes sur leur structure interne et leurs relations mutuelles, vérifient leur exactitude ou déduisent de nouvelles informations.

Pour conclure cette section, il convient de remarquer que le modèle de construction adopté dans ce travail est un modèle ascendant (nous partons des connaissances à représenter pour aboutir à une représentation formelle). Mais une construction descendante est possible, ce qui consiste à choisir un modèle opérationnel de représentation, en fonction de l'objectif d'utilisation de l'ontologie, puis à instancier ce modèle avec les connaissances du domaine.

4.1.5 Langages et outils de représentation d'ontologie

Ici, il convient de rappeler que dans le domaine des ontologies, les travaux relatifs à la création de langages utilisés et acceptés par tous ont atteint un certain niveau de maturité. De ce fait, nous avons aujourd'hui un certain nombre de langages tels que *RDF* ou *OWL* qui ont fait l'objet de propositions de la part de l'organisme en charge de la coordination des actions liées au *Web sémantique*. Par ailleurs, il existe une multitude d'outils de représentation d'ontologies variant selon les techniques utilisées et les objectifs affichés par leurs développeurs. Dans cette section, nous abordons certains de ces langages et outils.

Langages de représentation d'ontologie

Il existe divers langages de représentation d'ontologies. Celui que nous avons adopté dans ce travail est le OWL (Web Ontology Language) [OWL, 2004]. Ce langage, recommandation du W3C [W3C, 2006] et conçu pour aider à la réalisation du Web Sémantique (WS)[1] se décline en trois versions à la puissance d'expressivité croissante. Il s'agit de *OWL Lite*, *OWL DL* et *OWL Full*. La version adoptée dans le cadre de cette thèse est *OWL DL*. En effet, cette version qui repose sur les logiques de description, permet à l'ontologie d'être exploitée assez directement par les moteurs d'inférence implémentant les logiques de description. Ce qui a pour avantage de ne pas implémenter une ontologie écrite dans cette version de *OWL* pour pouvoir l'utiliser. Quant à *OWL Full*, il est conçu par l'union de la syntaxe de *RDF* et de *OWL*. Cette version présente l'avantage d'être syntaxiquement et sémantiquement compatible avec le langage RDF. Mais son inconvénient réside dans le fait que sa forte puissance d'expressivité le rend indécidable. Et quant à la version *OWL Lite*, elle a une puissance d'expressivité plus limitée, cependant elle est plus simple à implémenter.

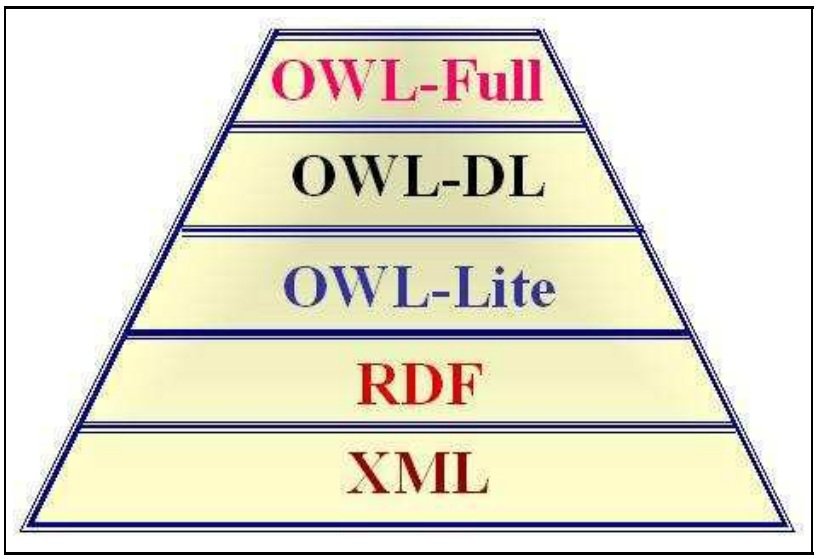

FIG. *4.2 - Langages de réprésentation d'ontologie (- sémantique au + sémantique)*

Revenons à présent sur la notion de Web Sémantique dont l'objectif avoué est d'ajouter une couche sémantique au-dessus du Web. Plus précisément, il s'agit de permettre aux machines de comprendre et de traiter l'information pertinente [Cregan et al., 2005]. En effet, il est inutile de rappeler que l'ensemble des thèmes traités dans les différentes pages Web est tel qu'une recherche syntaxique par mot-clés retourne très souvent des pages qui ne portent pas toutes sur le même domaine de connaissances. L'exploitation efficace des ressources du Web suppose donc que les moteurs de recherche puissent accéder à

[1]Le Web Sémantique est une vision future dans laquelle les informations seront données dans un format compréhensible utilisant les ontologies.

la sémantique de chaque page. C'est ce que recherche le Web Sémantique qui, utilisé avec OWL, permettra de passer d'une interopérabilité syntaxique à une interopérabilité sémantique. En effet, comme nous l'expliquions précédemment, le pouvoir d'expressivité de OWL permet de définir des ontologies avec une forte sémantique.
Pour résumer, nous pouvons dire que le Web sémantique permet d'ajouter une couche sémantique au-dessus de la couche HTML. Ceci permettra d'intégrer dans chaque page la représentation des connaissances qu'elle contient. Ainsi, les différentes applications Internet (moteurs de recherche, services, etc.) pourront alors accéder à la sémantique intégrée dans les pages Web. S'agissant des limites liées à l'utilisation du langage OWL, d'un point de vue pratique, quelques critiques sont avancées dans [Cregan et al., 2005].

Un autre langage de représentation de connaissances adopté par le W3C est RDF (Ressource Description Framework) [Brickley and Guha, 1999]. Utilisant la syntaxe XML (Extended Markup Language [XML, 2002]), RDF permet de décrire des ressources Web en termes de ressources, propriétés et valeurs. Une ressource peut être une page Web (identifiée par son URI[2]) ou une partie de page (identifiée par une balise). Les propriétés couvrent les notions d'attributs, de relations ou d'aspects et servent à décrire une caractéristique d'une ressource en précisant sa valeur. Les valeurs peuvent être des ressources ou des chaînes de caractères. Voici un exemple de description basé sur ce langage : où l'on essaye d'expliquer que l*objet TSS* a pour nom *Inertie totale* et a pour valeur 125.323 (sa représentation graphique est donnée dans la figure 4.3).

```
<?xml version="1.0"?>
      <rdf:RDF
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:dc="http://purl.org/dc/elements/1.1/">
         <rdf:Description rdf:about="TSS">
             <nom>Inertie totale</nom>
             <valeur>125.323</valeur>
         </rdf:Description>
</rdf:RDF>
```

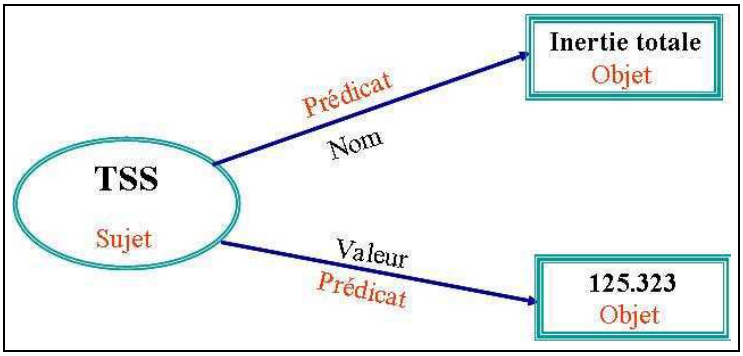

FIG. *4.3 - Représentation du graphe RDF de la description de l'objet* TSS

[2]United Ressource Identifier

Outils de représentation d'ontologie

Nous disposons déjà de nombreux outils et environnements de construction d'ontologies [Mizoguchi, 2004]. Dans cette section, nous abordons les outils utilisés dans la création d'ontologies. Cette liste, non exhaustive, tient compte des outils les plus significatifs et qui constituent une implémentation d'une méthodologie précise. Il convient de rappeler que tous ces outils offrent des supports pour le processus d'ontologisation, mais peu d'entre eux offrent une aide à la conceptualisation.

Citons TERMINAE, logiciel de construction d'ontologies à partir de textes, qui permet d'assister l'utilisateur à repérer les candidats termes d'un domaine et à les formaliser en concepts [Biébow and Szulman, 1999], [Terminae, 2002]. Grâce à l'utilisation des outils de TAL[3], cet outil permet une traçabilité des textes vers l'ontologie et inversément. Ce qui aboutit à une formalisation logique des connaissances. Rappelons aussi que les outils d'aide à la construction d'ontologie sont plus ou moins indépendants des formalismes de représentation. Ainsi, DOE (Differential Ontology Editor) [Troncy and Issac, 2002], [DOE, 2002] offre la possibilité de construire les hiérarchies de concepts et de relations en utilisant les principes différentiels énoncés par B. BACHIMONT [Bachimont, 2001], puis en ajoutant les concepts référentiels. La sémantique des relations est ensuite précisée par des contraintes.

De même, l'outil ODE (Ontology Design Environment) décrit dans [ODE, 2002] et [Blàzquez et al., 1998], permet de construire des ontologies basées sur la méthodologie appelée METHONTOLOGY [Fernández-Lòpez et al., 1997]. L'utilisateur construit son ontologie dans un modèle de type frame, en spécifiant les concepts du domaine, les termes associés, les attributs et leurs valeurs, les relations de subsomption. L'ontologie opérationnelle est alors générée en utilisant des formalismes ONTOLINGUA [Farquhar et al., 1997], [Ontolingua, 2002] ou FLOGIC [Kiefer et al., 1995]. Une adaptation de cet outil spécifiée pour le Web existe. Il s'agit de WEBODE [WebOde, 2002].

ONTOEDIT (Ontology Editor) [Mädche et al., 2000], [OntoEdit, 2002] est également un environnement de construction d'ontologies indépendant de tout formalisme. Il permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Aussi, cet environnement dispose des outils graphiques dédiés à la visualisation d'ontologies. Il intègre aussi un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Cet outil offre, à travers son plug-in ONTOKICK, la possibilité de générer les spécifications de l'ontologie par l'intermédiaire des questions de compétences [Sure et al., 2002].

Dans cette thèse nous utilisons un outil qui n'est pas lié à des formalismes de représentation. Il s'agit de PROTÉGÉ2000 [Noy et al., 2000], [Protége2000, 2002] qui est une interface modulaire permettant l'édition, la visualisation, la vérification des contraintes d'ontologies. Il permet aussi l'extraction d'ontologies à partir de sources textuelles, et la fusion

[3] Traitement Automatique des Langues

semi-automatique d'ontologies. Le modèle de connaissances sous-jacent à PROTÉGÉ2000 est issu du modèle des frames et contient des classes (concepts), des slots (propriétés) et des facets (valeurs des propriétés et contraintes), ainsi que des instances des classes et des propriétés.
PROTÉGÉ2000 autorise la définition de méta classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie.

Un autre outil se démarque de ceux cités précédemment, il s'agit de ONTOLINGUA [Farquhar et al., 1997], [Ontolingua, 2002]. ONTOLINGUA est un serveur d'édition d'ontologies au niveau symbolique : une ontologie est directement exprimée dans le formalisme ONTOLINGUA, qui est une extension du langage KIF (KNOWLEDGE INTERCHANGE FORMAT) [KIF, 2002]. Cet outil, qui supporte l'inclusion d'axiomes, permet aussi d'inclure une ontologie dans celle qui est en cours de construction [Farquhar et al., 1995].
Le langage ONTOLINGUA utilise des classes, des relations, des fonctions, des objets (instances) et des axiomes pour décrire une ontologie. Une relation (ou une classe) peut contenir des propriétés nécessaires et/ou suffisantes (contraintes) qui définissent la relation (ou la classe).

Moins ambitieux qu'ONTOLINGUA, OILED (OIL EDITOR) est un éditeur d'ontologies utilisant le formalisme OIL [OILED, 2002], [Bechhofer et al., 2001]. Il est essentiellement dédié à la construction de petites ontologies dont on peut ensuite tester la cohérence à l'aide de FACT (un moteur d'inférences bâti sur OIL).
Enfin DUINEVELD et ses collègues [Duineveld et al., 2000] décrivent et comparent d'autres environnements de développement d'ontologies.

4.1.6 Différences entre vocabulaire contrôlé, thésaurus, taxonomie, ontologie

Pour modéliser un domaine, nous n'avons pas forcément besoin de construire une ontologie de celui-ci. Dans certains cas une simple *taxonomie* reste suffisante. Dans cette section, notre objectif est d'expliquer les différences existant entre les concepts de *vocabulaire contrôlé*, de *taxonomie*, de *thésaurus* et d'*ontologie* (cf. figure 4.4).

Un *vocabulaire contrôlé* est une liste de termes définis par un groupe. Cette liste n'a pas l'ambiguïté du langage naturel. Concrètement, un *vocabulaire contrôlé* assure qu'un sujet sera décrit en utilisant le même terme préférentiel[4] chaque fois qu'il est indexé, facilitant ainsi la recherche d'information sur un sujet spécifique. Par exemple dans une base de données, le *vocabulaire contrôlé* garantit qu'un sujet sera décrit avec les mêmes termes préférentiels. Ainsi, durant une recherche, on peut trouver plus facilement tous les renseignements relatifs à un sujet précis. Par contre, cette liste de termes ne contient pas de signification et n'est pas organisée.

Une *taxonomie* est un *vocabulaire contrôlé* ayant une structure hiérarchique. Elle est sémantiquement pauvre. Selon [Daconta et al., 2003], les taxonomies sont généralement

[4]Un terme préférentiel est un mot qu'un grand nombre de personnes connaissent et utilisent. C'est aussi un terme autorisé dans un domaine donné.

utilisées dans le cadre d'une recherche sans but précis. Néanmoins, elle fournit un cadre conceptuel pour la discussion, l'analyse ou la récupération de renseignements. Concrètement, une taxonomie est souvent créée pour décrire des catégories et des sous-catégories de sujets qu'on trouve dans un domaine donné.

Un *thésaurus* est un vocabulaire contrôlé arrangé dans un ordre connu. Les termes qui le constituent sont organisés et régis par des relations sémantiques ou hiérarchiques. Ainsi, en utilisant un thésaurus, on peut faire des recherches plus fructueuses. En effet, un thésaurus est un sous-ensemble du langage qu'on utilise dans la vie quotidienne. Il comprend des renseignements sur les relations entre les mots et les phrases.

Comme nous l'expliquons dans la section 4.1.1, une *ontologie* est une structure hiérarchique des connaissances établie à l'aide d'un ensemble de concepts précis pour créer un vocabulaire convenu permettant l'échange d'information. Les différents concepts qui entrent dans la formation d'ontologies ont ete définis dans [Roche, 2006] d'un point de vue terminologique et computationnel. Ainsi, la différence fondamentale avec les concepts déjà définis est le fait que dans une ontologie, il existe des relations sémantiques entre les concepts. Ces relations peuvent être des :

- connaissances de composition ;
- définitions complètes de ce qu'est un concept par exemple ;
- contraintes d'intégrité ;
- fonctions de calcul ;
- propriétés algébriques comme par exemple la symétrie ou la transitivité ;
- connaissances par défaut ;
- relations inverses ;
- etc.

Ces propriétés permettent d'utiliser des moteurs de raisonnement pour inférer de nouvelles connaissances.

4.1.7 Types d'ontologies

M. USCHOLD rappelle que le domaine de connaissance sur lequel porte l'ontologie n'est pas le seul critère permettant de la typer. Dans [Uschold, 1996], il considère que les ontologies varient suivant trois dimensions : le *degré de formalisme* de la représentation (variant continuellement depuis le rigoureusement formel jusqu'au hautement informel), *l'objectif opérationnel* (communication entre utilisateurs, interopérabilité entre systèmes, application à un problème d'ingénierie comme la réutilisabilité de composants, la résolution de problèmes) et *le sujet* (domaine de connaissance, connaissances de raisonnement, connaissances liées au modèle de représentation).

Le *degré de formalisme* d'une ontologie doit être choisi en fonction des contextes d'usage de l'ontologie. En effet, il peut y avoir contradiction entre, d'une part, un besoin

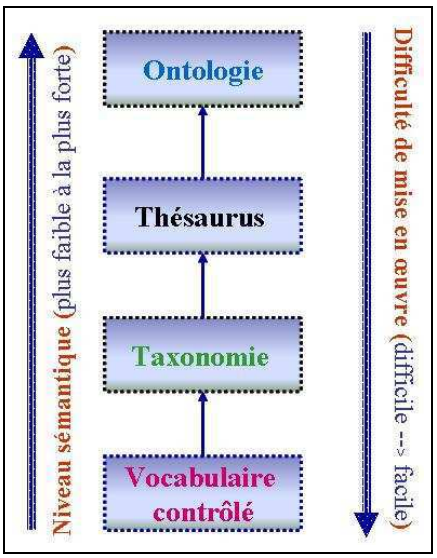

FIG. *4.4 - Différence entre vocabulaire contrôlé, thésaurus, taxonomie et ontologie*

de formalisme lié au partage des connaissances et à la réutilisabilité, et, d'autre part, la nécessité de produire une ontologie compréhensible par les utilisateurs. Un critère également utilisé pour caractériser une ontologie est sa granularité, c'est-à-dire le niveau de détails de sa conceptualisation. En fonction de *l'objectif opérationnel*, une connaissance plus ou moins fine du domaine est nécessaire, et des propriétés considérées comme accessoires dans certains contextes peuvent se révéler indispensables pour d'autres applications.

Selon le domaine à modéliser, nous pouvons distinguer les ontologies suivantes : [Chandrasekaran et al., 1999], [Guarino, 1997c], [Mizoguchi and Ikeda, 1997] et [Burgun and Bodenreider, 2001] :

- *les ontologies d'application :* elles ont un domaine de validité restreint et correspondent à l'exécution d'une tâche. Concrètement elles contiennent des concepts dépendants d'un domaine et d'une tâche particuliers, qui sont généralement subsumés par des concepts de ces deux ontologies. Plus précisément, il s'agit de mettre en relation les concepts d'un domaine et les concepts liés à une tâche particulière, de manière à en décrire l'exécution (par exemple : ***interpréter les résultats d'une application***).
- *les ontologies de domaine :* elles permettent de spécifier les connaissances d'un domaine (par exemple la *classification, l'éducation, la fouille de données, etc.*). Ce type d'ontologie régit un ensemble de vocabulaires et de concepts qui décrivent un domaine d'application [Mizoguchi et al., 2000]. Ce type d'ontologie peut aussi être obtenu en spécialisant une ontologie de haut niveau. De plus, elles ne sont pas propres à une tâche particulière. ***Dans ce travail nous adoptons ce type d'ontologie***. Ce type d'ontologie sera développé au paragraphe suivant.

- *les ontologies générales :* appelées aussi *ontologies génériques* ou *core ontology*, elles ne sont pas propres à un domaine. En effet, elles véhiculent des connaissances génériques (mais moins abstraites qu'une ontologie de haut niveau) réutilisables à travers différents domaines [Gómez-Pérez, 1999], [Heujst et al., 1997]. Ce qui rend leur précision très moyenne.
- *les ontologies de haut niveau :* ces ontologies ont une granularité large car elles portent sur des notions assez génériques telles que l'espace, le temps, les évènements, les états, les évènements, etc. Cette catégorie d'ontologie est en effet une ontologie générale intégrant les fondements philosophiques comme les principes à respecter pour sa conception [Guarino, 1997a], [Sowa, 1995a], [Sowa, 1995b]. Les concepts des trois autres types d'ontologie peuvent y faire référence. Voici des exemples de ce type d'ontologie : SUMO(Suggested Upper Merged Ontology) et BFO (Basic Formal Ontology).
- *les ontologies de tâches :* elles décrivent le vocabulaire concernant une tâche générique (par exemple : interpréter, classifier, etc), notamment en spécialisant les concepts d'une ontologie de haut niveau. Plus précisément, elles régissent un ensemble de vocabulaires et de concepts qui décrivent une structure de résolution de problèmes inhérente aux tâches et indépendante du domaine [Mizoguchi et al., 2000].
- *les ontologies de représentation des connaissances :* ces ontologies, comme expliqué dans [Gómez-Pérez, 1999], [Heujst et al., 1997], regroupent les concepts impliqués dans la formalisation des connaissances.

D'autres types de classification d'ontologie basés sur d'autres critères (tels que le niveau de détail, le niveau de complétude ou le niveau de formalisme) peuvent être trouvés dans [Fürst, 2002], [Guarino, 1997b], [Bachimont, 2000], [Uschold and Grüninger, 1996] et [Studer, 1998]

Ontologies de domaine :

Comme nous l'avions dit précédemment, ces ontologies vont permettre de spécifier les connaissances d'un domaine, indépendamment du type de manipulation effectuée sur ces connaissances. Elles sont appelées ontologies de domaine car elles sont construites sur un domaine particulier de la connaissance. De nombreuses ontologies de domaine existent déjà, telles que MENELAS dans le domaine médical [Zweigenbaum, 1999], ENGMATH pour les mathématiques [Gruber and Olsen, 1994], TOVE dans le domaine de la gestion des entreprises [Gruber et al., 1995], etc. Ce type d'ontologie peut être aussi décomposé en des sous types suivant leur niveau de granularité [Sowa, 2000], [Guarino et al., 1994]).

4.2 Métadonnées

Introduction

Notre objectif est d'expliquer très clairement ce que sont les métadonnées, en quoi peuvent t-elles aider dans le processus de capitalisation et surtout quels sont les types et les utilisateurs de ce concept. Ainsi, nous mettons en exergue certaines définitions les plus communément acceptées et utilisées par une large communauté. Mais pour nous, il s'agit d'expliquer en quoi l'utilisation de métadonnées peut être une solution efficace dans notre approche. De ce point de vue quelques remarques s'imposent. Dans nos travaux, les métadonnées constituent un outil permettant de structurer de manière sémantique les informations pertinentes lors d'un processus de classification. Ces informations sont des données participant au processus d'aide à l'interprétation.

4.2.1 Définitions de métadonnées

Dans la littérature, plusieurs définitions ont été avancées et parmi elles nous avons celles de [Sundgren, 1995] qui définit les métadonnées suivant divers aspects. Ainsi les métadonnées sont définies suivant trois axes (réprésentation, contenu, objectif). Suivant le premier axe, *les métadonnées sont définies comme la représentation physique de la méta information* (de même que les données sont les représentations de l'information). Suivant le deuxième axe, *les métadonnées fournissent l'information sur les données et sur les processus de production et d'utilisation des données.* Ou encore *les métadonnées sont des données dont on a besoin pour une production et une utilisation propre des données sur lesquelles elles nous fournissent de l'information* (dernier axe).

La définition la plus répandue et la plus imprécise est celle qui consiste à définir *les métadonnées comme étant des données sur les données.* Comme telles, les métadonnées peuvent être stockées et gérées dans des bases de données, souvent appelées registre ou répertoire de métadonnées. Ici, il est difficile de savoir à partir de quand les données sont des métadonnées ou de simples données.

Les métadonnées peuvent aussi être définies comme étant des données qui nous renseignent sur les différents aspects d'autres données : les aspects orientés sur le contenu, les aspects techniques [Lindblom and Sundgren, 2004]. Cette définition rejoint celle qui a été avancée en début de paragraphe.

W. Grossman [Grossman, 2004b] définit *les métadonnées comme étant un outil qui fournit les informations additionnelles nécessaires.* Les conditions préalables à cette information additionnelle sont :

- la connaissance de l'utilisateur sur les données ;
- l'usage qui sera fait des données.

Pour plus de détails sur les diverses définitions autour de ce concept, on pourra se reporter à [Lamb, 2001] qui fait un état de l'art des différentes interprétations afférentes

au terme de « métadonnées ». Il montre aussi l'influence de ces interprétations dans la construction des différents systèmes statistiques qui facilitent la collection, le traitement et la publication des informations statistiques.

4.2.2 Métadonnées : outil de capitalisation de la connaissance

Il existe plusieurs approches de modélisation des métadonnées, citons celles élaborées en statistique dans le cadre des 4ème et 5ème programmes Européens pour la recherche technologique et le développement.
En effet, le principe fondamental qui régit la modélisation des métadonnées se base sur le fait que celles-ci doivent être considérées comme des données ordinaires et modélisées comme telles.
Dans un monde très interconnecté, nous ne pouvons pas savoir à l'avance la destination que prendront nos informations avant d'arriver à l'utilisateur final. Ainsi, les métadonnées peuvent être potentiellement extraites par plusieurs applications différentes sans avoir accès à leur schéma. Ces même métadonnées peuvent être traitées, transférées, formatées, transcrites dans d'autres langues, avant d'arriver à l'utilisateur final (humain ou autre système).
Une des approches de modélisation proposée par [Vale and Pellegrino, 2000] est la suivante :

- analyser et décomposer les métadonnées de façon à ce que chaque élément de métadonnée soit le plus élémentaire possible ;
- faciliter ce processus de décomposition et de recomposition, en rendant les métadonnées abstraites. Par ailleurs, il faudra s'assurer que les métadonnées sont codifiées dans un format compréhensible par la machine (les textes libres devant être extrêmement limités).

Cette décomposition en éléments élémentaires permet de rendre flexible la migration des métadonnées dans d'autres formats, structures ou langues. C'est ce point de vue que nous adoptons dans la réalisation de ce travail.

4.2.3 Typologie de métadonnées

Suivant les domaines et leur utilisation, les métadonnées se déclinent sous différents aspects. D'une manière générale, les métadonnées sont très hétérogènes dans leurs fonctions, leurs représentations et leurs usages. Toutefois, dans cette section, nous tentons de dégager les différentes classifications réalisées dans un certain nombre de travaux.
Suivant ***les domaines***, nous avons la classification suivante : En ***statistique***, nous avons la classification faite par [NU and ECE, 1998] et qui distingue trois groupes de métadonnées statistiques :

- *les métadonnées aidant les utilisateurs à rechercher des données :* il s'agit de chercher la disponibilité des données à différents niveaux, faire le lien entre différentes variables et thèmes ou sujets. Elles peuvent aussi faire référence au contenu ; par exemple les périodes couvertes par les données (dans le cas des séries temporelles) ;
- *les métadonnées sur le contenu (métadonnées pour l'interprétation des données) :* il s'agit des informations qui décrivent la ***population***, l'***objet cible***, la ***définition des variables***, la ***définition des unités de mesure***, la ***période référence***, le ***temps entre*** la période de référence et celle de publication, la ***fréquence/périodicité des statistiques***, les ***descriptions des paramètres de temps***, les ***méthodes d'agrégation***, les ***ajustements*** (par exemple les ajustements saisonniers), etc ;
- *les métadonnées pour estimer la qualité des données :* il s'agit des informations relatives aux instruments de mesure (sources administratives, les enquêtes d'entreprise, etc.), à la comparabilité dans le temps (pour les séries temporelles). Cette comparabilité inclut une description des coupures dans la série temporelle et les valeurs oubliées avec leur explication. Il s'agit aussi des informations relatives à la comparabilité avec les statistiques alternatives (collectées via un instrument de mesure différent) pour les même variables, etc.

Suivant leur ***emplacement par rapport aux données***, nous avons deux types de métadonnées décrits dans les travaux de [Kent and Schuerhoff, 1997], [Farmakis et al., 2001], [Karge and Rauch, 2001], [Pellegrino, 2000] et de [Bi and Lamb, 2001]

- *les métadonnées actives :* qui sont celles qui accompagnent les données. Elles sont saisies et mises à jour par des interventions humaines ;
- *les métadonnées passives :* qui sont celles qui font référence aux données mais n'y sont pas connectées. Elles sont saisies séparément en utilisant des formulaires.

Dans le ***domaine des bases de données distribuées***, les métadonnées sont utilisées pour faciliter la comparaison et la compréhension des différentes bases de données. Dans ce sens, [Bi et al., 1999] distingue deux types de métadonnées :

- *les métadonnées sémantiques :* elles fournissent les informations sur la définition de concept, la classification, la référence des données, les données qui fournissent une information additionnelle sur la création des bases (par exemple les formules de dérivation utilisées), etc.
- *les métadonnées structurelles :* elles représentent l'information qui décrit l'organisation, la structure des bases de données distribuées. On peut citer l'exemple de *l'information sur l'institution*, le *nom de la base de données*, le *schéma des composants de la base de données*, les *attributs*, les *types de données* des attributs et les *relations syntaxiques* entre eux.

Bien évidemment il existe d'autres types de métadonnées suivant les domaines, leur utilisation ou leur structure. Pour plus de détails, voir les travaux effectués dans [Pukli, 2004], [Ma and Lamb, 2001], [LaPlant et al., 1996], [Aberer and Read, 1998], [Booleman, 2004], [NU and ECE, 1998].

4.2.4 Besoins des utilisateurs de métadonnées

D'une manière générale, les utilisateurs ont des besoins d'utilisation des métadonnées dans les cas suivants :

- *localiser* et trouver les données qu'ils recherchent ;
- *comprendre* les données trouvées ;
- *fournir* une documentation unifiée des différentes sources de métainformation pour la recherche de l'information ;
- *décider* de l'exactitude et de leur appropriation en comparaison avec les besoins originaux ;
- *combiner* différentes données comme les micro et les macro données de diverses sources si nécessaire ;
- *estimer* ces données (en termes de qualité et/ou de contenu).

4.3 Panorama des travaux utilisant les métadonnées et/ou l'ontologie dans le processus d'ECD

Nous présentons les différents travaux qui utilisent les métadonnées et/ou les ontologies dans le processus d'ECD afin d'améliorer les résultats obtenus ou d'aider à l'interprétation de ces résultats. Ainsi, nous abordons *l'utilisation d'ontologie du domaine de l'étude* dans le processus de fouille de données. Ensuite, nous présentons *l'utilisation de métadonnées* (représentées par les connaissances du domaine) dans la sélection des attributs et dans l'interprétation des résultats et ce tout au long du processus d'ECD. Puis, nous présentons l'utilisation de points de vue des experts dans le processus d'ECD. Enfin, nous présentons les travaux d'incorporation des connaissances du domaine dans la fouille de données.

Utilisation d'ontologie du domaine de l'étude : ces travaux consistent à construire une ontologie du domaine de l'étude, une base de données orientée "fouille" et une base de connaissances. L'objectif de cette approche est d'essayer d'améliorer les étapes de prétraitement et de post-traitement en fouille de données [Brisson et al., 2006]. Ces travaux présentent la construction du système d'information qui consiste en la création de trois éléments cités précédemment. Cette approche présente quelques inconvénients dont le fait qu'il faut construire pour chaque nouveau domaine de l'étude son ontologie. Et comme la construction de l'ontologie n'est pas un processus automatique, alors il est difficile d'identifier à chaque fois les concepts qui doivent constituer l'ontologie. De plus, la prise en compte de cette nouvelle ontologie dans l'application n'est pas garantie car il n'existe

aucune indication sur l'évolutivité et la flexibilité de leur application.

Utilisation des connaissances du domaine : cette approche, appelée AICEF (Approche d'Incorporation des Connaissances Expertes dans la Fouille de données) et développée par [Ben Ahmed, 2005], propose l'élaboration et l'utilisation de métadonnées multi-vues comme un moyen pour l'incorporation des connaissances formalisées dans le processus d'ECD. Il s'agit d'utiliser des métadonnées dans la phase de sélection des attributs ainsi que dans la phase d'interprétation des résultats. Pour soutenir son approche, il se base sur l'utilisation et la formalisation des connaissances du domaine de l'accidentologie en métadonnées. Ces métadonnées peuvent aider l'expert à sélectionner les attributs qui sont pertinents vis-à-vis de l'objectif de l'étude. L'intérêt d'une telle est de réduire le risque, par l'expert, d'oublier un point de vue pertinent lors de la sélection des attributs. Cette approche présente les mêmes inconvénients cités précédemment mais restent néanmoins beaucoup plus flexible car il existe un réel travail de (méta)modélisation du domaine de l'accidentologie.

Prise en compte des points de vue relatifs à un processus d'ECD : il s'agit de prendre en compte les points de vues des experts en ECD relativement à leur domaine mais aussi au domaine analysé afin d'annoter les principles décisions tout au long du processus en relation avec les objectifs poursuivis : sélection de variales, choix de méthodes etc. [Behja et al., 2005b]. Dans ce travail, l'approche visant à ajouter des métatdonnées est globale avec un travail plus poussé au niveau de l'étape de pré-traitement des données (proposition d'une ontologie) alors que dans notre cas notre travail se situe au niveau de l'étape de l'interpétation suite à une classification. De plus, H. Behja propose une plate-forme objet de développement, d'évaluation et d'évolution d'un système d'information basé sur le Web [Behja et al., 2005a]. Comme nous pouvons le constater, ces travaux ne s'intéressent pas à aider les utilisateurs dans leur processus d'interprétation.

Utilisation des métadonnées dans le processus de fouille de données : cette approche consiste à utiliser des métadonnées lors de la phase de fouille de données afin d'enrichir et d'améliorer les résultats [Boussaid and Loudcher, 2006]. Ces métadonnées sont obtenues ***manuellement*** en partant par exemple des règles de gestion du domaine de l'étude. Dans ces travaux, les métadonnées sont représentées à l'aide du formalisme RDF puis des graphes conceptuels. Elles sont ensuite intégrées aux données sur lesquelles la fouille a lieu. L'idée principale est de modifier une information originale par une information plus générale. L'approche est testée sur des données et des métadonnées du domaine de l'accidentologie avec l'utilisation d'un algorithme en arbre de décision. Les résultats montrent une nette amélioration du taux de mauvais classement. Cette approche, comme nous pouvons le constater, est très adaptée au domaine *supervisé* car il est plus facile d'utiliser des connaissances sur des méthodes dont nous pouvons gérer le déroulement, ce qui n'est pas le cas dans le cadre *non supervisé* qui nous intéresse dans cette thèse. De plus, *l'acquisition des métadonnées* se fait *manuellement*.

4.4 Synthèse du Chapitre 4

L'objectif de ce chapitre était de présenter les concepts de *métadonnées* et d'*ontologie* en commençant par présenter quelques définitions qui nous ont paru pertinentes. Ensuite, ce chapitre a été consacré à la présentation de quelques particularités de ces concepts. Ainsi, la section 4.1.5 présente quelques langages et outils de représentation des ontologies. A travers cette section, nous expliquons le choix que nous avons fait quant à l'utilisation du langage et de l'outil de représentation d'ontologie, respectivement *OWL* et PROTÉGÉ2000. Dans la dernière section de ce chapitre, nous avons présenté les différents travaux qui s'intéressent à l'incorporation des connaissances dans les étapes du processus d'ECD. L'objectif était de montrer notre positionnement et l'originalité de nos travaux dans ce domaine.

Le chapitre suivant fait le point sur l'utilisation de *métadonnées* dans certains domaines tels que la *documentation électronique*, *le Web*, etc. A la lumière des remarques apportées dans ce chapitre, nous pouvons avancer que l'utilisation de métadonnées dans le processus d'interprétation semble être une solution viable.

Chapitre 5

Domaines d'utilisation des métadonnées

Introduction

L'UTILISATION des métadonnées est présente dans des domaines divers et variés. Dans ce chapitre, nous effectuons une synthèse de l'utilisation des métadonnées dans certains domaines connexes à notre travail. L'objectif est de voir s'il existe des domaines dans lesquels les métadonnées sont utilisées. Dans l'affirmative, il s'agit de trouver de quelle manière elles le sont et pour quels buts. Pour effectuer ce travail de synthèse, nous nous proposons d'étudier les techniques d'utilisation des métadonnées dans des domaines tels que la **statistique**, la **documentation électronique** et les **systèmes décisionnels**. Le choix de ces domaines s'explique par le fait que nos travaux se focalisent sur la manière d'aider les utilisateurs à interpréter automatiquement les résultats de leurs méthodes de classification. Or, le processus de classification est un processus qui fait appel en amont à des techniques et/ou outils provenant du domaine de la statistique, de la documentation et de la fouille de données. En ce sens, il semblait important de voir de quelles manières les métadonnées sont utilisées dans ces domaines.

5.1 Métadonnées en statistique

Depuis l'avènement du Web, l'utilisation des données statistiques a changé. Ce domaine a vu le rôle et l'utilisation des métadonnées prendre une grande envergure. En effet, les données statistiques sont désormais accessibles à des utilisateurs néophytes et qui recherchent les moyens pour interpréter les informations auxquelles ils ont accès [Kokolakis et al., 2001]. Dans ce contexte les métadonnées peuvent garantir la qualité des données et permettre leur interprétation.

Il existe aujourd'hui un certain nombre de guides et de normes édictés par les Instituts Nationaux de Statistique (INS) tant au niveau International qu'Européen. Des propositions de modélisation de la méta information statistique, en utilisant des métadonnées structurées pouvant être exploitées algorithmiquement plutôt que d'avoir du texte, ont été avancées. Ceci permettrait de comparer des tables statistiques de différents INS ainsi que d'évaluer leur niveau d'harmonisation. D'autres avantages peuvent être avancés tels que l'estimation du temps mis pour réaliser différentes étapes du processus statistique ; ou encore identifier les goulots d'étranglement et faciliter ainsi les prises de décisions.

Une étude portant sur les standards de métadonnées dans ce domaine est effectuée dans [Bargmeyer and Gillman, 2000]. Ces travaux portent aussi sur la description et l'importance de ces standards dans les agences statistiques. Aussi, il est prouvé aujourd'hui que les métadonnées sont essentielles dans toutes les étapes du processus statistique pour identifier la signification des données [Booleman, 2004]. L'un des avantages des métadonnées consiste à automatiser les traitements statistiques de l'information [Karge and Rauch, 2001], [Papageorgiou et al., 2000a]. L'utilité d'avoir des métadonnées dans le processus de production statistique est expliquée dans [Grossmann, 1998].

D'autres travaux traitent des nouveaux défis et des rôles des métadonnées dans le domaine de la fouille de textes et de la fouille de données dans les systèmes statistiques [Soltés, 2004]. Ces travaux évoquent les deux fonctions clés des métadonnées statistiques, il s'agit :

- de *servir d'outil pour la fouille de données* c'est-à-dire l'extraction et la découverte des données à partir d'un entrepôt de données particulier ;
- d'*assister dans l'interprétation des données* à extraire et/ou des données déjà extraites.

Pour utiliser les métadonnées dans ce domaine, certains travaux se focalisent sur la mise en place d'un métamodèle de métadonnées statistiques [Barboni et al., 2001]. Certains éléments de métadonnées statistiques ont été développés dans [NU and ECE, 2000b] ainsi qu'un ensemble de directives de modélisation de ces éléments [Sundgren, 1995]. Dans le même cadre, nous assistons à l'utilisation des métadonnées dans le processus de réalisation des questionnaires [Lamb, 1993]. Ce qui confirme l'intérêt porté à l'utilisation des métadonnées dans ce domaine.

Dans cette section, nous nous intéressons aux liens entre métadonnées, statistique et Web ainsi que les fonctions que doivent remplir les métadonnées dans le domaine de la statistique. Enfin, nous verrons l'utilisation que est faite des métadonnées dans ce domaine.

5.1.1 Statistique, Web et Métadonnées

Comme nous l'avons dit en introduction de cette section, l'avènement du Web a radicalement changé l'utilisation des données statistiques. Dans ce contexte, et compte tenu de l'hétérogénéité du Web, les métadonnées devant assister les utilisateurs dans la recherche et l'interprétation des données doivent être standardisées. Ces métadonnées peuvent être regroupées de la manière suivante :

- *les métadonnées assistant la recherche et la navigation :* elles donnent des informations générales sur les sites Web. Elles permettent ainsi aux utilisateurs de comprendre le cadre général du site ;
- *les métadonnées assistant à l'interprétation :* elles sont orientées sur le contenu. Elles dépendent beaucoup du sujet et sont orientées vers le groupe d'utilisateurs. De ce fait il est souhaitable d'avoir les données et les métadonnées ensemble de telle sorte que les utilisateurs puissent choisir le mode d'interprétation de l'information statistique ;
- *les métadonnées assistant les post-traitements :* ce sont des métadonnées qui peuvent être incluses lorsque les utilisateurs ont l'intention de réutiliser les informations provenant du Web pour d'autres applications.

Pour que les métadonnées puissent assister les utilisateurs dans la phase d'interprétation, nous avons besoin des métadonnées relatives à la description. *Par exemple* les informations relatives à la *population statistique*, la *couverture géographique*, *l'unité d'observation*, les *classifications* et les *standards appliqués*, etc. Nous pouvons aussi avoir d'autres éléments de métadonnées tels que les *désignations des lignes/colonnes dans les tables*, les *définitions des labels*, les *unités de mesure* , la *comparabilité sur le temps* (coupures en séries, données oubliées), la *source des données*, l'*explication des standards de symbole dans les tables*, toute information sur le *copyright*, les *restrictions d'usage*, etc.

A ce niveau, il convient de rappeler la typologie de métadonnées effectuée par FARMAKIS [Farmakis et al., 2001] qui distingue les ***métadonnées structurelles*** des ***métadonnées méthodologiques.*** Les premières sont utilisées dans l'interprétation de l'information statistique par les utilisateurs finaux. Elles doivent être compréhensibles par la machine (dans la littérature, elles sont appelées ***métadonnées actives***). Tandis que les secondes sont des métadonnées de description des informations statistiques et qui ne permettent pas d'interpréter l'information statistique. D'autres travaux relatifs à des propositions de modélisation des métadonnées statistiques dans le but d'aider les utilisateurs dans leur processus d'interprétations sont proposés dans [NU and ECE, 2000a]. Pour plus de

détails sur les typologies de métadonnées suivant les domaines, cf. à la section 4.2.3.

Nous voyons, à travers ces exemples, que les métadonnées peuvent assister les utilisateurs dans la recherche de l'information statistique ainsi que dans l'interprétation de leur contenu. Après leur téléchargement, elles peuvent aider dans les post-traitements des applications statistiques. Nous verrons en détail les fonctions que peuvent jouer les métadonnées dans le domaine de la statistique dans la section 5.1.2.

5.1.2 Fonctions et rôles des métadonnées en statistique

Ici, il convient de rappeler les raisons pour lesquelles on pourrait être amené à utiliser les métadonnées. Ces raisons sont évoquées dans [SCI, 2006] :

- éviter qu'une documentation soit lacunaire car ceci entraîne souvent la perte de données essentielles ;
- faciliter la promotion des données ;
- éviter d'oublier les méthodes de collecte et de traitement des données ;
- éviter la perte d'information à la suite du départ d'un employé ;
- permettre l'utilisation ultérieure des données ;
- déterminer si des données d'autres sources peuvent être utiles et savoir comment les utiliser.

Les travaux relatifs aux rôles des métadonnées sont nombreux, nous nous limitons dans cette section à détailler ceux réalisés par [Lamb, 1993]. Dans ses travaux, il définit les fonctions de base des métadonnées, en statistique, qui sont :

- contrôler et gérer le processus de production statistique ;
- informer les utilisateurs sur le contenu et la qualité des données ;
- documenter le système statistique ;
- promouvoir la modélisation, la coordination et l'intégration du système statistique.

Dans le domaine statistique, un bon modèle de métadonnées peut augmenter la transparence des informations statistiques en facilitant la vérification de la cohérence et de la consistance des données traitées.
Parmi les avantages que peut constituer l'utilisation des métadonnées dans le cas d'une enquête statistique, nous pouvons citer :

- qu'elles fournissent l'information qui rende les données compréhensibles et partageables ;
- qu'elles soient un répertoire de connaissances et d'expertise ;
- qu'elles structurent l'information et stockent les connaissances expertes d'un domaine précis de spécialistes ;
- qu'elles soient utiles pour estimer la qualité et la fiabilité des données ;

- qu'elles soulignent les différences entre les pays et les écarts des standards internationaux ;
- qu'elles soient très importantes pour les utilisateurs dans la sélection et l'interprétation des données.

Il faut rappeler que les métadonnées jouent un rôle important dans la documentation au niveau de l'aspect qualité [Froeschl and Grossmann, 1999]. Ainsi, il n'est pas surprenant de remarquer que ces dernières années l'intérêt porté aux métadonnées augmente.

D'une manière générale, dans les systèmes d'information, les métadonnées peuvent jouer un rôle très important durant tout le cycle de vie d'un système d'information [Vassiliadis and Stavrakas, 1998] :

- durant la phase de conception, les métadonnées peuvent fournir au concepteur des informations pertinentes sur la structure et la signification des divers concepts et structures qu'il a déjà créés ;
- durant la phase de développement et de maintenance des logiciels, les métadonnées vont jouer le rôle de guide dans la localisation de l'information ;
- durant la phase d'évaluation, les métadonnées sont très utiles dans l'amélioration de l'interprétabilité des composants et les informations du système de l'évaluateur ;
- durant la phase du reverse engineering, il est évident que la présence des schémas riches de métadonnées est une procédure moins pénible.

5.1.3 Utilisateurs de métadonnées en statistique

L'objectif de cette section est d'expliquer qu'il existe deux grands groupes d'utilisateurs répertoriés dans ce domaine :

- les statisticiens et/ou les experts du domaine statistique ;
- les utilisateurs de l'information statistique ayant peu ou une expérience limitée des statistiques.

A travers cette première classification, M. PETRAKOS classe les utilisateurs de métadonnées de la manière suivante [Petrakos et al., 2001] :

- *les fournisseurs des données :* les personnes impliquées dans la modélisation et la réalisation des enquêtes, la collecte des données, leur compilation et leur dissémination.
  Ils utilisent une information correcte sur les *unités*, les *concepts*, les *populations*, etc. Pour modéliser une enquête, ils suivent le processus de collecte et de préparation des données pour retrouver les métadonnées pertinentes, avant de disséminer finalement les données accompagnées de métadonnées.
- *les utilisateurs des données :* il s'agit des politiques, des médias ou encore des académies.

Pour ces utilisateurs, les métadonnées permettront de retrouver l'ensemble des données qu'ils recherchent [Pellegrino, 2000].

- *les seconds fournisseurs de données :* les organisations qui reçoivent les données statistiques et qui les recompilent avant de les disséminer à leur tour.
  Ces utilisateurs multi-sources incluent des organisations internationales telles que l'Union Européenne (U.E), le Fonds Monétaire International (F.M.I), l'Organisation de Coopération et de Développement Économiques (O.C.D.E), les Nations Unies (N.U) ou l'Organisation Mondiale du Commerce (O.M.C)
  Ils ont besoin d'être capables de déterminer si les données provenant de différents pays sont comparables ou non. C'est ce qui explique que ces différentes organisations aient mis en place un certain nombre de standards qui permettent de savoir les informations à ne pas omettre (pour plus de détails, voir [OCDE, 2004], [OCDE, 1997], [Sundgren, 1995], [Sundgren, 1999], [Vale and Pellegrino, 2000]).

5.2 Métadonnées en analyse des données symboliques

Pour expliquer l'utilisation des métadonnées dans le domaine de l'analyse des données symboliques, il convient de rappeler que dans le cadre du projet Asso[1], des travaux ont proposé et défini des métadonnées sur les objets symboliques [Papageorgiou et al., 2000c], [Vardaki, 2004], [Papageorgiou, 2003]. L'objectif était de permettre aux utilisateurs de mieux comprendre le processus d'agrégation des données en donnant des informations supplémentaires sur les objets symboliques. Pour expliquer l'utilisation des métadonnées, nous prenons l'exemple suivant portant sur l'agrégation des *abalone* dont les données proviennent du site de l'UCI [Warwick et al., 1994][2].

Nom	Type Données	Mesure	Description
Sex	Nominal		M, F, and I (infant)
Length	Continuous	mm	Longest shell measurement
Diameter	Continuous	mm	perpendicular to length
Height	Continuous	mm	with meat in shell
Whole Weight	Continuous	grams	whole abalone
Sucked Weight	Continuous	grams	weight of meat
Viscera Weight	Continuous	grams	gut weight (after bleeding)
Shell Weight	Continuous	grams	after being dried

Tab. *5.1 - Tableau de données des* Abalone

[1]Il s'agit d'un logiciel public issu du projet Européen ASSO et accessible à l'adresse suivante : http ://www.info.fundp.ac.be/asso/. Il contient l'implémentation de la plupart des méthodes d'analyse symbolique.

[2]Ces données sont disponibles sur le site de l'UCI Machine Learning à l'adresse suivante : http ://www.ics.uci.edu/ mlearn/MLSummary.html. Il s'agit de 4177 instances décrites par 8 variables (dont une nominale).

A travers ce tableau de données, nous pouvons constater qu'il existe des informations que **nous risquons de perdre** si elles ne sont pas stockées dans des métadonnées. Il s'agit par exemple des *unités de mesure* ou de la description des variables. Les *métadonnées extraites* de ce tableau sont : le *type de variable* (nous avons huit variables dont 7 sont **continues** et 1 qui est **nominale**). Une autre métadonnée concerne *l'unité de mesure* des variables continues et les *valeurs prises* par la variable nominale. Ainsi, nous constatons que les *unités de mesure* utilisées sont le *mm* et le *gram*. Par ailleurs, il convient de remarquer qu'il n'existe pas de métadonnées sur les données elles-mêmes.

En agrégeant ces données en données symboliques, un certain nombre de métadonnées peuvent être extraites. Les travaux de [Vardaki, 2004], [Papageorgiou et al., 2002], [Papageorgiou et al., 2001] et [Papageorgiou et al., 2000c] vont dans ce sens. Pour les objets symboliques créées, nous avons les métadonnées obtenues par le logiciel SODAS[3] sur les *abalones* (cf. figure 5.1) avec la méthode SOE [Noirhomme-Fraiture and Rouard, 2000]. A travers cette figure, nous voyons que la métadonnée *type de variable* n'a plus la même valeur pour les variables symboliques. Ainsi, les variables symboliques sont de type *intervalle*. De plus, nous avons à présent de nouvelles métadonnées sur les *données symboliques*, il s'agit du **nombre d'individus** formant chaque nouvel objet symbolique, de **l'opérateur d'agrégation** utilisé pour construire les données symboliques. Dans cet exemple, l'opérateur d'agrégation utilisé est celui donné dans la formule 3.2 et que nous avons repris ici :

$$\oplus(\{Y_j(\omega)/\omega \in C_i\}) = \begin{cases} [min\{Y_j(\omega)/\omega \in C_i\}, max\{Y_j(\omega)/\omega \in C_i\}] & \text{si } Y_j \text{ quantitative} \\ \{Y_j(\omega)/\omega \in C_i\} & \text{si } Y_j \text{ qualitative} \end{cases}$$

Cet opérateur d'agrégation permet de sélectionner les valeurs *minimales* et *maximales* prises par une catégorie d'*abalone*. Ainsi, dans cet exemple, le premier enregistrement du tableau de la figure 5.1 indique que pour les *abalones* de sexe femelle ayant quatre à six anneaux, nous avons une valeur de leur *diamètre* comprise dans l'intervalle [0.19 : 0.47]. Ces **informations** qui peuvent paraître triviales **s'avèrent indispensables** dans un environnement de partage ou pour des utilisateurs n'ayant pas de connaissances sur les données de départ. Ainsi, nous remarquons que dorénavant, les valeurs prises par les variables symboliques ne sont pas des valeurs atomiques. En effet, un objet d'objet symbolique étant une agrégation de plusieurs individus, il est important d'avoir une description sémantique de celui-ci. C'est ce qu'apporte, entre autres, l'utilisation des métadonnées dans ce contexte.

[3] réalisé dans le cadre du projetASSO

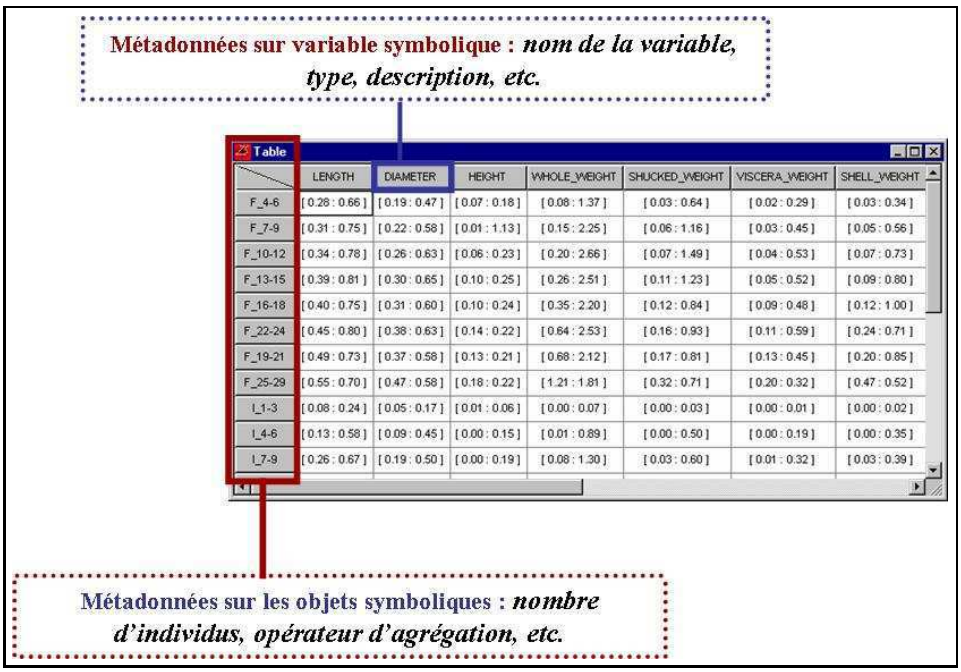

FIG. *5.1 - Métadonnées sur données symboliques*

Nous insistons ici sur le fait que l'utilisation des métadonnées dans un domaine tel que l'analyse des données symboliques est indispensable dans la mesure où les objets symboliques représentent une description des données de départ. Pour pouvoir interpréter, ou du moins utiliser les bons critères d'interprétation, les résultats des méthodes d'ADS appliquées sur ces données, nous avons besoin d'avoir des connaissances sur les données d'origine.

5.3 Métadonnées en Systèmes décisionnels

Dans cette section, nous abordons le rôle que peuvent jouer les métadonnées dans les systèmes décisionnels. Il s'agit plus particulièrement des entrepôts de données et de la fouille de données. En effet dans ces domaines, les métadonnées sont stockées dans des répertoires et sont utilisées pour permettre de décrire les données utilisées dans les analyses et les prises de décisions. Ces métadonnées peuvent fournir des informations sur :

- la définition exacte des données (sémantique) ;
- la source des données (date, origine) ;
- la façon dont elles sont calculées, agrégées ;
- les règles de gestion qui s'y rapportent ;
- le processus d'extraction, de transformation et de chargement qui a été mis en œuvre (c'est-à-dire la technologie utilisée pour permettre d'effectuer des synchronisations massives d'information d'une banque de données vers une autre).

Les métadonnées obtenues dans ce contexte sont essentielles pour conforter les analyses effectuées à partir des données. Certains outils d'extraction et de gestion des métadonnées existent à travers ces outils décisionnels.
Des travaux sur les problèmes de l'utilisation et de la gestion des métadonnées dans le domaine des bases de données (plus particulièrement des entrepôts de données).

5.4 Métadonnées en fouille de données : utile ou futile ?

Il nous a semblé utile d'apporter quelques informations sur l'intérêt que représente l'utilisation des métadonnées en fouille de données et par la même occasion de faire un lien étroit entre ces deux domaines.
En effet, il y a encore quelques années, certains spécialistes en fouille de données ne trouvaient pas de liens étroits entre métadonnées et fouille de données. Il s'agit par exemple de W. Inmon qui n'arrivait pas à établir une relation claire entre ces deux concepts [Inmon, 1996]. Or, les métadonnées sont des données servant à décrire d'autres données quel que soit leur support. Cette description peut couvrir plusieurs aspects des données. Ces aspects peuvent être :

- une information structurelle : comment les données sont stockées et organisées ;
- une information sur les mesures : informations sur la quantité des données et la manière dont elles sont distribuées ;
- etc.

Ceci montre que les métadonnées sont implicitement ou indirectement des données qui décrivent un ou plusieurs aspects d'une base de données. Pour montrer l'évidente relation entre les métadonnées et la fouille de données, il convient de déterminer le rôle des données historisées dans le processus de fouille de données (cf. aux travaux de H. Behja [Behja et al., 2005b], [Behja et al., 2005a] pour plus de précisions).

En effet, il semble clair que les données « historisées » sont la colonne vertébrale d'un entrepôt de données car elles constituent 99% d'un entrepôt de données (ceci étant même la définition d'un entrepôt de données). Ainsi, ces données intéressent très fortement les experts en fouille de données car elles fournissent les bases pour trouver les patterns cachés, les tendances, les relations, les associations, etc. Les changements dont peuvent faire l'objet des données au cours du temps peuvent intervenir au niveau de :

- *la structure :* cette structure change de manière continue ;
- *la signification :* un jour il y a une définition sur ce qu'est un produit, le jour d'après la définition de ce qu'est ce produit a déjà changé ;
- *les classements :* classer par exemple les ventes Parisiennes dans celles des ventes de la région Parisienne ;
- *le calcul :* la variation de la taxe sur un produit donné par exemple ;
- *les conditions économiques :* la variation du taux de l'emploi par exemple.

Les changements cités ci-dessus montrent qu'il existe plusieurs aspects des données et des traitements qui changent perpétuellement. Ainsi, si nous nous plaçons à un instant t, les données paraissent statiques or on s'aperçoit très vite que ces données, vues de manière historique, sont en perpétuel changement. Ce qui explique l'utilisation des métadonnées dans ce processus. En effet, les métadonnées vont décrire les données ainsi que les divers changements intervenus sur celles-ci dans le temps.

Pour s'en convaincre, prenons l'exemple d'un utilisateur effectuant une analyse basée sur des données créées en 2001. Avec les informations dont il dispose sur l'historique des données, il peut comprendre la raison pour laquelle les revenus sont passés d'un montant au cours d'une année à un montant plus (respectivement moins) significatif que l'année précédente. Les raisons de cette augmentation (respectivement diminution) peuvent être expliquées par un contexte économique, social, politique, etc.
Cet exemple montre la place que peut avoir l'utilisation des métadonnées dans un processus de fouille de données. Il semble évident que pour interpréter des données évoluant dans le temps, il est indispensable de connaître le contexte dans lequel ces données ont été collectées et/ou calculées. Or, ce contexte est stocké et géré dans l'infrastructure de métadonnées. Les métadonnées, par définition, doivent permettre de décrire les divers changements opérés sur les données dans le temps. En l'absence de ces métadonnées il sera difficile, voire impossible, d'interpréter les données contenues dans un entrepôt de données et par conséquent les traitements effectués sur celles-ci. Nous voyons, à travers cet exemple, que les métadonnées peuvent être utilisées en amont et en aval du processus de fouille de données.

5.5 Métadonnées en Documentation électronique

En général, les systèmes d'information du Web ont des problèmes communs : le déficit de connaissances sur le contenu, la structure et la localisation de l'information d'un site Web. Ainsi, les utilisateurs souhaitent, dans leur majorité, qu'il y ait des métadonnées riches permettant d'aider à éliminer ce type de problèmes.
Ce domaine fait partie des pionniers en matière d'utilisation des métadonnées. En effet, les métadonnées étaient principalement utilisées pour aider à la recherche d'informations. Ce qui est très réducteur du rôle que peuvent jouer les métadonnées dans ce domaine. C'est ce qu'a tenté d'expliquer [Froeschl and Grossmann, 1999] dans son article portant sur le rôle des métadonnées dans l'amélioration de la qualité des documents numériques.

Pour expliquer le rôle des métadonnées dans ce domaine, prenons l'exemple d'un organisme documentaire (une médiathèque par exemple) qui utilise des notices (qui sont en réalité des métadonnées) pour décrire le contenu des documents. A n'en pas douter, ces métadonnées facilitent la gestion interne des ressources documentaires et permettent d'optimiser la recherche et la localisation des documents. Car ces notices contiennent des informations sur la *source du document* (titre, auteur, année, sujet, éditeur, etc.), la *nature du document* (monographie, périodique, cédérom, etc.), son *contenu informationnel*

(descripteurs, mots-clés, résumé, etc.) et sa *localisation physique* (la cote).

D'autres travaux relatifs a l'utilisation des métadonnées dans ce domaine ont été effectués par [Keith, 2001] et [Baldonado et al., 1996]. Un panorama des travaux en métadonnées dans le domaine de la documentation a été réalisé dans [Role, 1999]. Nous pouvons aussi trouver des travaux sur l'extraction automatique de métadonnées, à partir de documents électroniques, basée sur la méthode de Support Vector Machine [Han et al., 2003].

5.6 Autres domaines

Dans cette section, nous abordons d'autres domaines utilisant les métadonnées. Il s'agit par exemple du domaine biomédical où nous avons des travaux relatifs à l'utilisation des métadonnées [Mougin et al., 2003]. Ces travaux consistent à montrer l'intérêt des métadonnées pertinentes dans la résolution des problèmes d'hétérogénéité de diverses sources.

D'autres domaines ont eu recours aux métadonnées. C'est le cas du système CUBER [Pöyry et al., 2002] relatif à l'apprentissage en éducation. C'est le cas aussi du système pour images mobiles (pour des détails, voir [Sarvas et al., 2004]) ou de la Géoscience [Parekh et al., 2004] ainsi que des travaux de WEAVER [Weaver et al., 2001].
Nous retrouvons aussi les métadonnées dans le domaine des données spatiales où elles sont utilisées pour décrire les contenus, les qualités, les localisations et les autres caractéristiques sur les données spatiales [80]. Dans ce domaine, les métadonnées permettent d'apporter des connaissances sur :

- *la disponibilité :* les informations nécessaires pour déterminer l'ensemble des données existant pour une zone géographique ;
- *l'aptitude pour l'utilisation :* les informations utiles pour déterminer si un ensemble de données rencontre un besoin spécifique ;
- *l'accès :* les informations nécessaires pour acquérir un ensemble de données identifié ;
- *le transfert :* les informations nécessaires pour traiter et utiliser un ensemble de données.

5.7 Standards et normes de métadonnées

Sur le Web, on assiste à présent à la naissance de nouveaux moteurs de recherche (on pense notamment à Exalead) qui savent utiliser « intelligemment » des métadonnées. Au départ, les travaux relatifs à ce domaine étaient ceux du Dublin Core [Core, 2006] qui est une initiative dédiée à ces questions depuis 1995. Cette initiative traite des standards de métadonnées en documentation électronique. Dans cette thèse, nous utilisons certains éléments de ce standard dans notre métamodèle d'interprétation.
Rappelons que les éléments de métadonnées proposés par cette initiative sont au nombre de quinze (15), répartis autour de trois domaines, et permettant d'identifier et de décrire

les ressources documentaires. La sémantique de ces éléments a été établie par un consensus international de professionnels provenant de diverses disciplines. L'objectif était de réaliser une norme indépendante des plates-formes, des matériels et des logiciels informatiques, répondant aux besoins suivants :

- présenter, dans sa totalité, l'information descriptive trouvée dans les instruments traditionnels de recherche d'archives ;
- préserver les relations hiérarchiques existant entre les niveaux de description ;
- transmettre l'information descriptive héritée d'un niveau hiérarchique à l'autre ;
- permettre au chercheur de se déplacer à l'intérieur de la structure d'information hiérarchique.

Ces éléments sont les suivants :

- *Contenu :* Titre, Sujet, Description, Source, Langage, Relation, Couverture
- *Propriété intellectuelle :* Créateur, Éditeur, Contributeur, Droits
- *Matérialisation :* Date, Type, Format, Identifiant

Pour le moment ce n'est qu'un standard (pas d'obligation légale et non soumis à une norme de qualité). Néanmoins, ces métadonnées servent à vulgariser le contenu informationnel d'un document électronique en donnant la possibilité d'intégrer ces éléments de métadonnées dans le document lui même (par exemple dans l'en-tête des documents HTML ou en tant que fichier XML autonome).
Ces éléments de métadonnées ont une valeur ajoutée certaine et seront sûrement normalisées (type ISO) à terme. En effet, des moteurs de recherche dédiés à la lecture et au décryptage sémantique de ces métadonnées permettraient une optimisation et une efficacité accrue des recherches d'information opérées par un utilisateur ou un robot sur le Web. Comme nous l'expliquons précédemment, les requêtes fournies par les utilisateurs donnent des résultats pour le moins surprenants.

Par ailleurs, l'utilisation des métadonnées constitue à coup sûr un moyen permettant la mise en place d'un Web sémantique. En effet, leur utilisation et leur exploitation permettraient de :

- ne rendre visibles et lisibles que les informations pertinentes pour l'utilisateur (avec indice de pertinence) ;
- diminuer les risques de désorientation liés à une masse d'information somme toute non pertinente comme c'est le cas aujourd'hui.

On pourrait aussi penser que l'utilisation des métadonnées dans ce domaine permettra d'opérer des recherches équivalentes avec celles réalisées dans les bibliothèques.

En statistique, les propositions de standardisation et de normalisation des métadonnées sont de plus en plus fréquentes. Ainsi, des organisations internationales comme les Nations Unies [Sundgren, 1995], l'OCDE (1996) ou le FMI (1998) ont depuis des années entamé un processus de standardisation des métadonnées dans bon nombre de domaines. Citons aussi

le travail de BARGMEYER [Bargmeyer and Gillman, 2000] qui a eu un large impact sur les efforts de construction des systèmes de métadonnées dans la communauté statistique. Les descriptions de plusieurs standards de métadonnées et leur importance sont fournies dans ses travaux, ainsi que des applications de ces standards dans des instituts statistiques.

En statistique justement, l'un des standards les plus utilisés est le SDMX [SDMX, 2006]. Son objectif est l'échange des informations statistiques (données et métadonnées) dans des activités collectives. Ce qui peut se faire dans le cadre d'un échange direct (modèle batch) ou par la mise en ligne des données et des métadonnées sur un site Web. Ce site Web est alors accessible à tous les partenaires (modèle de dissémination). Dans les deux cas, il y a un besoin incontestable de comprendre la nature de ce que nous voulons échanger (ou de ce qui a été échangé). En effet, tout échange d'information peut être bénéfique s'il est basé sur une terminologie commune.
L'autre standard, dans ce domaine, est venu de la part de Metadata Common Vocabulary [MCV, 2006]. Son objectif est de développer une compréhension commune des standards d'éléments de métadonnées décrivant les concepts statistiques et les méthodologies utilisées par les statisticiens dans la collecte, le traitement et la diffusion des données statistiques.

D'autres propositions de standards de métadonnées ont été réalisées notamment par W. LAPLANT [LaPlant et al., 1996]. Il propose des standards de métadonnées sur le contenu portant le nom de SDSM (Standard for Survey Design and Statistical Methodology Metadata). Ce standard fournit une description et des informations sur les données statistiques. Beaucoup de travaux ont été réalisés dans les communautés de standards nationaux et internationaux telles que l'ANSI (American National Standard Institute) et l'ISO (International Organization for Standardization) pour réaliser des consensus sur la standardisation des métadonnées. L'objectif de cette section n'est pas de donner une liste exhaustive de tous les travaux relatifs à ce domaine. Aussi pour plus de détails sur les standards dans le domaine de la statistique, se référer à [Olenski, 2001].

5.8 Synthèse du Chapitre 5

Ce chapitre a été consacré à faire le point sur l'utilisation de métadonnées dans certains domaines tels que les *Statistiques* (cf. section 5.1), l'*analyse des données symboliques* (cf. section 5.2), les *bases de données* (cf. section 5.3), la *fouille de données* (cf. section 5.4), la *documentation électronique* (cf. section 5.5) ainsi que dans d'autres domaines.

A travers ces sections, nous montrons que la notion de métadonnées est très largement partagée par une grande majorité de domaines. Nous évoquons dans la section 5.1 le rôle que peuvent jouer les métadonnées en apportant des informations indispensables à la compréhension des données traitées.

Dans les systèmes décisionnels, il est tout aussi important d'avoir des métadonnées afin de comprendre le processus de traitement des données (c'est ce que nous expliquons dans la section 5.3). De plus, nous montrons à travers la section 5.4 que l'utilisation de métadonnées dans le processus de fouille de données semble évidente.

Conclusion de la partie II

Cette partie a été consacrée à présenter une vue d'ensemble sur les différents travaux réalisés dans des domaines connexes à nos travaux. Ainsi, dans le *Chapitre 3*, nous expliquons le processus d'ECD et le niveau d'implication de nos travaux dans ce domaine. En effet, le processus d'ECD se trouve à la base de nos travaux car notre objectif est de trouver une approche permettant d'aider l'utilisateur à anticiper la phase d'interprétation des résultats provenant des méthodes de fouille de données. Plus précisément nous prenons l'exemple des méthodes de classification. Ainsi, ce chapitre a servi à montrer que le domaine de la classification constitue un champ d'application idéal compte tenu de la complexité d'interprétation et du caractère *non supervisé* de ce processus. Dans ce même chapitre, nous évoquons les travaux réalisés dans le cadre de l'utilisation des métadonnées dans le domaine de l'analyse des données symboliques. Une autre partie de ce chapitre est consacrée à l'explication des différentes approches d'interprétation utilisées dans le cadre de la fouille de données. Notre positionnement par rapport à ce domaine a été explicité de manière claire. Dans le *Chapitre 4*, nous évoquons les notions de métadonnées et d'ontologie. Il s'agit d'expliquer les caractéristiques qui nous ont intéressées dans ces différentes approches et en quoi notre approche était différente. Enfin dans le *Chapitre 5*, nous montrons l'intérêt porté par les domaines divers et variés à l'utilisation de métadonnées. L'objectif est de montrer en quoi l'utilisation des métadonnées pourrait nous être utile dans nos travaux.

Troisième partie

Contributions pour l'aide à l'interprétation de classes

Organisation de la partie III

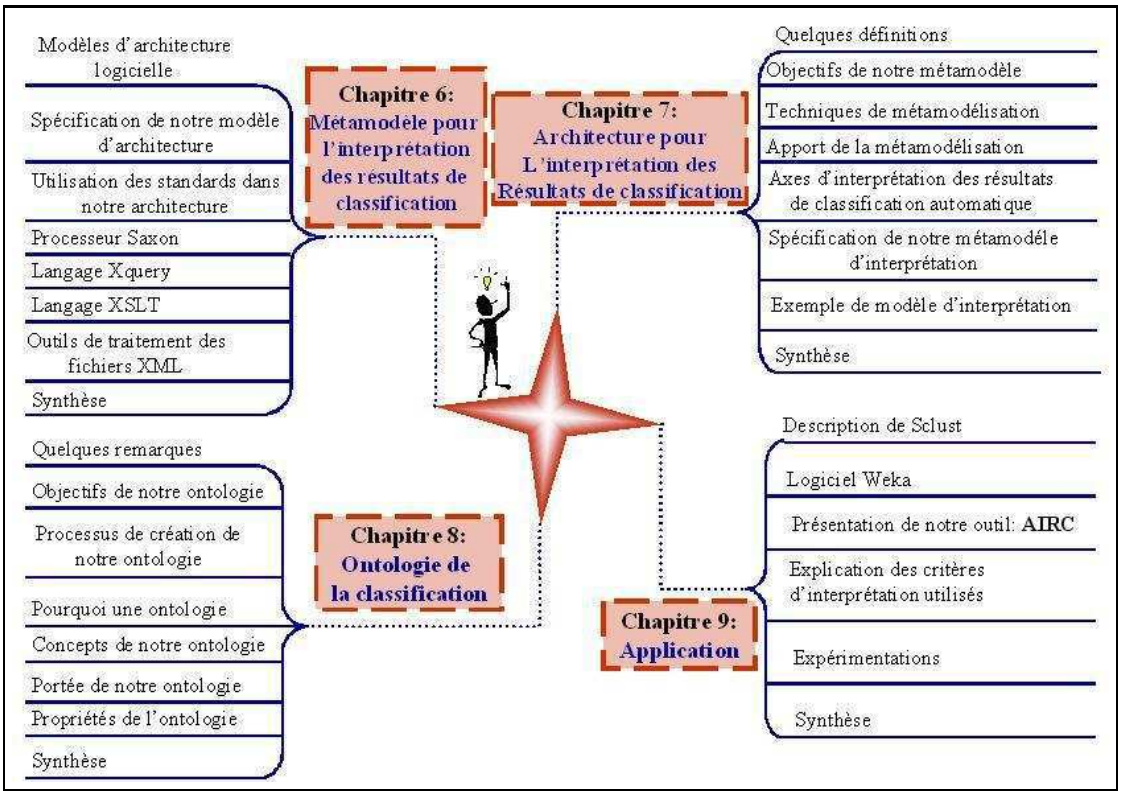

FIG. *5.2 - Contributions de notre approche dans l'interprétation des résultats de classification*

Introduction de la partie III

Cette partie est consacrée à la présentation de notre approche. En effet, comme nous l'expliquons en introduction de cette thèse, il n'existe quasiment pas de travaux relatifs à l'interprétation automatique des résultats des méthodes de fouille de données. A ce niveau, il convient de préciser que notre objectif *n'est pas de créer* de nouveaux critères d'interprétation mais plutôt d'aider les utilisateurs à utiliser les critères existants dans leurs tâches d'interprétation. Or, suivant les outils et les méthodes de classification, il existe une multitude de critères d'interprétation. Ainsi, cette partie est consacrée à expliciter les différentes composantes de notre approche :

- Le **Chapitre 6** est consacré à notre métamodèle d'interprétation. Il s'agit d'expliquer en quoi une métamodélisation semble être la solution que nous avons privilégiée dans nos travaux. Pour ce faire, nous consacrons une partie de ce chapitre à faire un succinct état de l'art des différentes techniques de métamodélisation existantes.
- Le **Chapitre 7** est dédié à l'explicitation de l'architecture de notre approche. De la même manière que dans le Chapitre 6, nous avons fait un succinct état de l'art sur les différents modèles d'architecture utilisés à ce jour. Par ailleurs, ce chapitre est aussi consacré à l'explication de certaines technologies utilisées lors de notre implémentation.
- Le **Chapitre 8** se consacre à l'explication de l'ontologie du domaine de la classification. Il s'agit d'expliquer l'ontologie minimale construite dans le but d'automatiser le processus d'interprétation.
- Le **Chapitre 9** a pour but d'expliquer notre prototype implémenté pour valider notre approche.

Chapitre 6

MIRC : Métamodèle pour l'Interprétation des Résultats de Classification

Introduction

Les métadonnées sont un outil fournissant les informations additionnelles nécessaires à la compréhension des données [Grossman, 2004b]. Les conditions préalables à ces informations additionnelles dépendent de la connaissance de l'utilisateur sur les données et de l'usage qui sera fait de celles-ci. Dans cette thèse, les métadonnées sont utilisées afin de servir lors du processus de capitalisation des connaissances disponibles tout le long du processus de classification. Dans ce contexte, il est important de définir le cadre d'extraction et d'utilisation de ces métadonnées. Ce processus de modélisation a conduit à la création d'un métamodèle définissant le cadre de modélisation du processus de classification.

En effet, il convient de différencier le processus de modélisation et celui de métamodélisation. Le premier consiste à extraire les caractéristiques abstraites du domaine à modéliserTandis que le second consiste à appliquer une nouvelle fois ce processus d'abs-

traction sur le modèle lui même (on obtient ainsi le métamodèle du domaine). Ce méta-modèle contient ainsi les éléments invariants du domaine ainsi que le savoir-faire associé au domaine. Ce phénomène de (méta)modélisation est expliqué dans la figure 6.1 tirée de [Le Moigne, 1999]. Les techniques de métamodélisation permettront de définir une approche qui se situe au dessus des langages et outils de conception existants. Et grâce à cette abstraction, il sera possible de réunir et de faire interagir différents outils de conception au sein d'un même environnement.

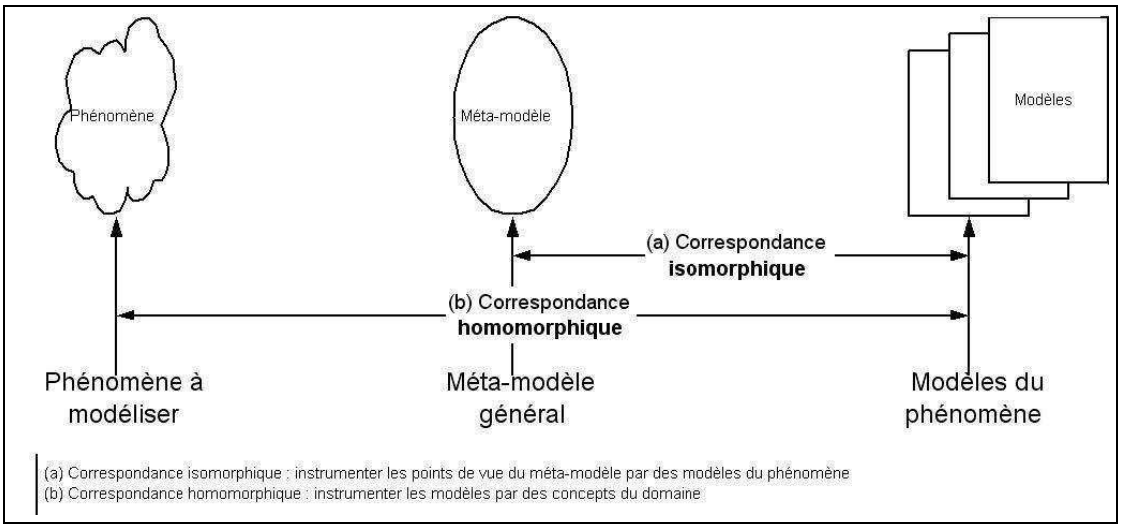

FIG. *6.1 - Modélisation dirigée par un métamodèle général*

Notre solution au problème de modélisation du processus est d'utiliser un métamodèle de ce domaine. Ce chapitre est organisé de la manière suivante :

- la section 6.1 donne *quelques définitions* des concepts de modèle et de métamodèle ;
- la section 6.2 est consacrée à la description des *objectifs* de notre métamodèle ;
- la section 6.3 décrit quelques unes des *techniques de métamodélisation* existantes ;
- la section 6.4 est consacrée à l'élucidation des raisons d'utiliser une métamodélisation dans nos travaux ;
- la section 6.5 est consacrée à la description des *axes d'interprétation* sur lesquels nous nous appuyons pour définir notre métamodèle ;
- la section 6.6 fait état de la *stratégie de modélisation* que nous avons adoptée dans le cadre de ce travail et les raisons qui ont guidé ce choix ;
- la section 6.7 est consacrée principalement à la *spécification de notre métamodèle*. Il s'agit de définir l'ensemble des classes, des associations, etc.
- la section 6.8 montre un exemple d'instanciation de notre métamodèle.

6.1 Quelques définitions

Voici les définitions des termes centraux de ce chapitre : il s'agit des concepts de *modèle* et de *métamodèle*.

Définition 6.1. *Un* ***modèle*** *désigne un* objet, *un* type *déterminé selon lequel des objets semblables peuvent être reproduits à de multiples exemplaires. C'est donc une représentation d'informations concrètes.*

Définition 6.2. *Un* ***modèle*** *est une abstraction d'un système qui se traduit, en général, par la disparition de détails d'ordre technique. Ainsi, un modèle représente le système qu'il décrit et doit pouvoir être utilisé à sa place pour répondre à un certain nombre de questions sur celui-ci [Bézivin and Gerbé, 2001].*

Définition 6.3. Dans le contexte du MOF, *un* ***modèle*** *est une collection de métadonnées mises en relation de manière à ce qu'elles décrivent des éléments d'information, eux même en relation [MOF, 2006].*

Définition 6.4. *Un* ***métamodèle*** *est la spécification d'une abstraction, c'est-à-dire d'un ou de plusieurs modèles. Cette spécification définit un ensemble de concepts importants pour exprimer des modèles, ainsi que les relations entre ces concepts. Le métamodèle définit la terminologie à utiliser pour définir des modèles [Bézivin and Gerbé, 2001].*

Définition 6.5. *Un* ***métamodèle*** *contient un ensemble de concepts et d'assertions qui vont définir la façon dont un modèle sera extrait d'un système. Autrement dit un métamodèle est le modèle de l'outil qui « sert à fabriquer » d'autres modèles.*

Le métamodèle peut être considéré comme un filtre qui ne retiendrait du système considéré qu'un certain nombre d'aspects jugés pertinents. Et d'une manière générale, les métamodèles définissent des modèles, servant eux mêmes à décrire des instances. Il est communément admis que le métamodèle est du niveau du domaine, le modèle de celui de l'application, et l'instance de celui du phénomène particulier et observable. Le métamodèle sert à décrire de façon formelle les concepts communs à l'ensemble des modèles d'un domaine ainsi que la syntaxe et la sémantique de la notation qui permet leur manipulation. Leur utilisation permet de rendre plus puissant, plus souple et plus versatile un logiciel car ils isolent l'application des changements intervenant dans le modèle d'application.

Dans cette thèse, nous proposons d'appliquer cette démarche au processus d'interprétation des résultats de classification. Car l'emploi d'un métamodèle est une extension logique du processus d'abstraction utilisé pour créer des objets de modèle. Un modèle objet est une abstraction des données, et peut être décrit à l'aide de métadonnées. Un métamodèle est une abstraction des métadonnées.

Le but de la métamodélisation est de définir un type de modèle avec tous ces types d'éléments et leurs contraintes. Pour ce faire il existe trois approches :

– définir un métamodèle en partant de rien ;

- modifier un métamodèle existant : ajout, suppression, modification d'éléments et des contraintes sur leurs relations ;
- spécialiser un métamodèle existant en ajoutant des éléments et des contraintes ; ce qui correspond aux profils UML. Un profil UML est une spécialisation du métamodèle UML.

L'une des exigences dans la définition des métamodèles est de garantir les règles de présentation, de consistance et de transformation des modèles.

Les métamodèles permettront la gestion et l'échange des modèles. Il existe déjà, dans la littérature, plusieurs métamodèles et un effort de standardisation de ces métamodèles a donné naissance à des spécifications telles que le Meta Object Facility (MOF) de l'Object Management Group (OMG), le standard Xml Metadata Interchange (XMI). Dans ce chapitre, nous présentons les travaux qui nous ont inspirés dans cette thèse.

6.2 Objectifs de notre Métamodèle

Il convient de rappeler que les deux attributions récurrentes du concept de métamodèle sont d'une part l'expression des règles pérennes pour le domaine d'application, et d'autre part la définition de la forme et du langage d'écriture des modèles. De ce point de vue le modèle est une instance du métamodèle visant à la description d'un cas particulier (une application du domaine concerné).
En ce qui nous concerne, notre métamodèle aspire à répondre aux problèmes posés par la diversité des approches d'interprétation des résultats de classification. Comme nous l'expliquons dans la section 3.4, suivant la méthode et l'outil utilisés, les utilisateurs sont confrontés à diverses approches d'interprétation. Il est ainsi très difficile de trouver un modèle unique pouvant représenter de manière claire ces différents points de vue.

L'objectif du métamodèle que nous proposons est de trouver une description générique de tous les modèles d'interprétation possibles dans le domaine de la classification automatique. Il s'agit d'obtenir un outil d'aide à la modélisation. C'est pourquoi la section 6.7 est consacrée à expliquer la démarche à suivre pour exploiter de manière efficace notre métamodèle. En effet, l'intérêt de fonder une démarche de modélisation sur un métamodèle est de s'extraire des contraintes liées à la création d'un modèle de domaine. Dans notre contexte, le métamodèle permettra de fournir aux différents acteurs concernés par les divers aspects de l'interprétation un modèle épuré contenant des critères compréhensibles par tous.
Dans la section suivante (section 6.3), nous abordons les différentes techniques de métamodélisation utilisées dans des domaines divers et variés.

6.3 Techniques de métamodélisation

Introduction

Les techniques de métamodélisation sont mises en œuvre dans différents contextes et/ou domaines. C'est le cas de ***l'ingénierie des connaissances*** où des travaux ont porté sur l'utilisation de métamodèles comme base de représentation et de la gestion des connaissances. Pour plus de détails, voir les travaux de [Gerbé and Kerhervé, 1998], [Guarino and Welty, 2000b] et [Tannebaum, 2002]. Dans ce contexte, un métamodèle est un moyen permettant d'exprimer l'ontologie d'un domaine. Un autre domaine utilisant ce concept est celui de la ***communauté du génie logiciel*** où les métamodèles servent de support à la génération de code ou encore à la supervision des systèmes [Sibilla and Jocteur-Monrozier, 1999]. Dans les ***Workflow***, les processus peuvent être modélisés en utilisant les techniques de métamodélisation dans les activités liées au workflow [Le Pallec, 2000].

L'adoption par l'OMG (Object Management Group)[1] du ***MOF*** (Meta-Object Facility) [Crawley et al., 1997], [OMG, 2006] traduit l'évolution de manière radicale de la notion de métamodélisation. La spécification MOF définit un langage générique et universel représentant le cœur des aspects de métamodélisation traités par l'OMG. Le paragraphe 6.3.1 donne des détails sur ce concept qui nous a servi de base dans la réalisation de notre métamodèle.
Dans le paragraphe 6.3.2, nous abordons le format d'échange de modèles basé sur la sémantique du MOF et sur la syntaxe de XML : ***XMI*** (XML Metadata Interchange) (pour plus de détails, cf. [XMI, 2006]). Cette spécificité permet au XMI de fournir une mise en correspondance entre MOF et XML. Ce qui permet d'intégrer, de partager et de comparer différents métamodèles générés par différents outils et par différentes organisations. Ce standard fournit une DTD ou un schéma XML pour les modèles d'information de manière non ambiguë.
Le dernier concept que nous abordons dans le paragraphe 6.3.3 est celui du Métamodèle d'entrepôt commun connu sous le nom de ***Common Warehouse Metamodel (CWM)*** [CWM, 2006]. Ce standard publié en 2001 et adopté par l'OMG, est une architecture basée sur un modèle qui offre des fonctionnalités performantes pour une large gamme d'entrepôts de données hétérogènes. Ce modèle a été conçu pour faciliter l'interopérabilité des métadonnées entre les outils d'entrepôt de données et les répertoires de métadonnées dans un environnement distribué et hétérogène.

Tous les concepts qui seront développés dans les sections suivantes ont eu un impact à différents niveaux sur la réalisation de notre métamodèle. Ces relations seront explicitées dans les sections correspondantes aux concepts cités précédemment.

[1] Rassemble tous les acteurs industriels concernés par les technologies objet

6.3.1 L'exemple du MOF

Pour faire face à la variété et à l'incompatibilité de métamodèles, l'OMG a mis en œuvre une structure générale d'intégration pour tous les métamodèles : MOF. MOF est une spécification définie par des industriels pour répondre à un besoin en terme de modélisation d'information de plus en plus diverse et hétérogène. MOF est ainsi utilisé pour définir des métadonnées en tant qu'objets CORBA. Rappelons que CORBA permet à deux applications de communiquer entre elles sans se soucier de leur localisation ou de leur mode de conception. Dans le contexte du MOF, une métadonnée est un terme générique attribué à une donnée représentant de l'information. Ainsi, une métadonnée peut représenter l'information contenue dans un système ou le système lui même. Cependant, toutes les métadonnées respectent les mêmes règles de structuration et de cohérence, en l'occurence la syntaxe abstraite commune. ***Dans ce travail, nous nous sommes inspirés de l'architecture du MOF*** pour élaborer notre architecture de métamodélisation.

MOF se retrouve au sommet d'une architecture à quatre couches décrite dans la figure 6.2. Dans cette architecture, chaque niveau entretient une relation d'instanciation avec le niveau supérieur. La clé de voûte de cette architecture est la présence d'un niveau de méta-métamodélisation fournissant un langage commun, permettant de définir et de lier ensemble métamodèles et modèles. En effet, le modèle MOF est utilisé pour définir la structure et la sémantique des métamodèles.

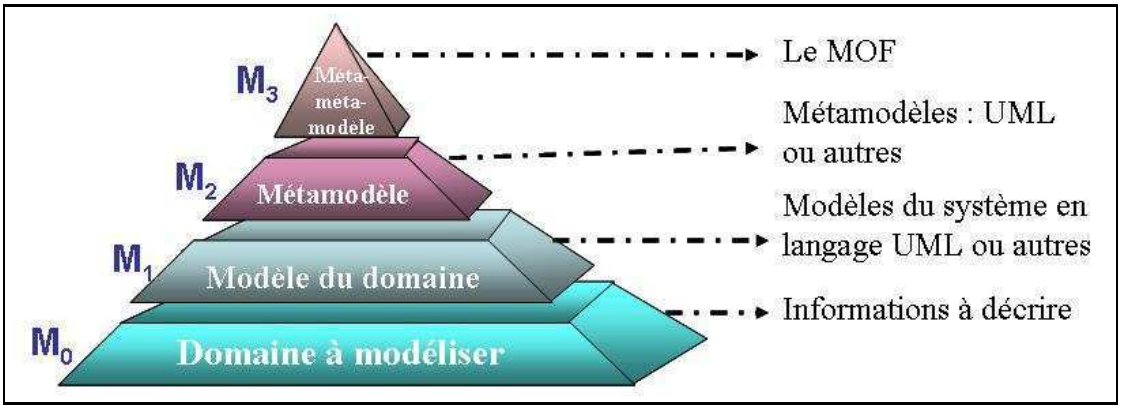

FIG. *6.2 - Architecture du MOF*

Cette architecture, autour de laquelle s'est aujourd'hui formé un consensus, est composée des niveaux suivants :

- **M0** : ce niveau correspond au système modélisé. C'est aussi le niveau contenant les informations à décrire. Pour faire le parallèle avec les langages objets, ce niveau contient les instances des classes.
- **M1** : ce niveau est constitué par les modèles du système réel défini dans un certain langage. C'est le niveau contenant les métadonnées décrivant l'information.

Ce niveau correspondrait dans les langages objets à la définition des classes. Par exemple une classe sera définie à l'aide d'attributs et de méthodes.

- **M2** : ce niveau est constitué par les métamodèles. C'est le niveau contenant les méta-métadonnées, c'est-à-dire la description de la structure et de la sémantique des métadonnées. Ces métamodèles peuvent être perçus comme des langages de description de différents types de données. Par exemple, ce niveau contient la définition de ce qu'est une classe, une opération, un attribut ou une association. Bref c'est le niveau qui contient l'ensemble des règles de construction et de notation des modèles. Ainsi, une classe est définie comme ayant un nom et deux collections : ses attributs et ses méthodes. Puis un attribut est défini comme ayant un nom et un type, ainsi de suite.
- **M3** : ce niveau est constitué du seul et unique MOF. C'est le niveau qui décrit la structure et la sémantique des méta-métadonnées. Ces descriptions sont regroupées au sein de méta-métamodèles. Elles représentent un langage pour définir les différents types de méta-métadonnées.

Voici un exemple concret de métamodélisation qui reprend les quatre 4 niveaux que nous venons de décrire (cf. figure 6.3). Cet exemple est relatif à l'utilisation de la métamodélisation dans le langage UML. Ainsi, nous constatons que les concepts *Class*, *Attribut* et *Association* (appartenant au niveau **M2**) sont tous des ***instances*** du concept *Class* (qui lui représente le niveau **M3**). Au niveau **M1**, nous voyons que les concepts *Nom*, *Personne et Voiture* et *possède* sont, respectivement, des ***instances*** des concepts *Attribut*, *Class* et *Association*. Enfin le niveau **M0** les instances du domaine sur lequel porte l'étude, ici nous avons le concept *UnePersonne* qui est ***instance*** du concept *Personne*. Certains concepts de ce langage sont définis au paragraphe suivant.

Concepts du MOF

MOF est un langage d'expression de métamodèles. Comme nous l'expliquons en introduction de ce chapitre, un métamodèle est un langage permettant de décrire des modèles d'un domaine d'intérêt particulier. Par exemple, UML est un métamodèle dédié à la description des artefacts logiciels à objets. Un métamodèle contient un ensemble de concepts, de relations et de contraintes. Dans ce paragraphe, nous abordons certains concepts du standard MOF, pour plus de détails voir [MOF, 2006]. Dans ce travail, nous utilisons les concepts suivants :

- *classe* : qui permet de définir les types des méta-objets du MOF ;
- *association* : qui permet de modéliser des relations binaires entre les méta-objets ;
- *type de donnée* : qui permet de modéliser les autres données (types primitifs ou autres) ;
- *paquetage* : qui permet de rendre ces modèles modulaires.

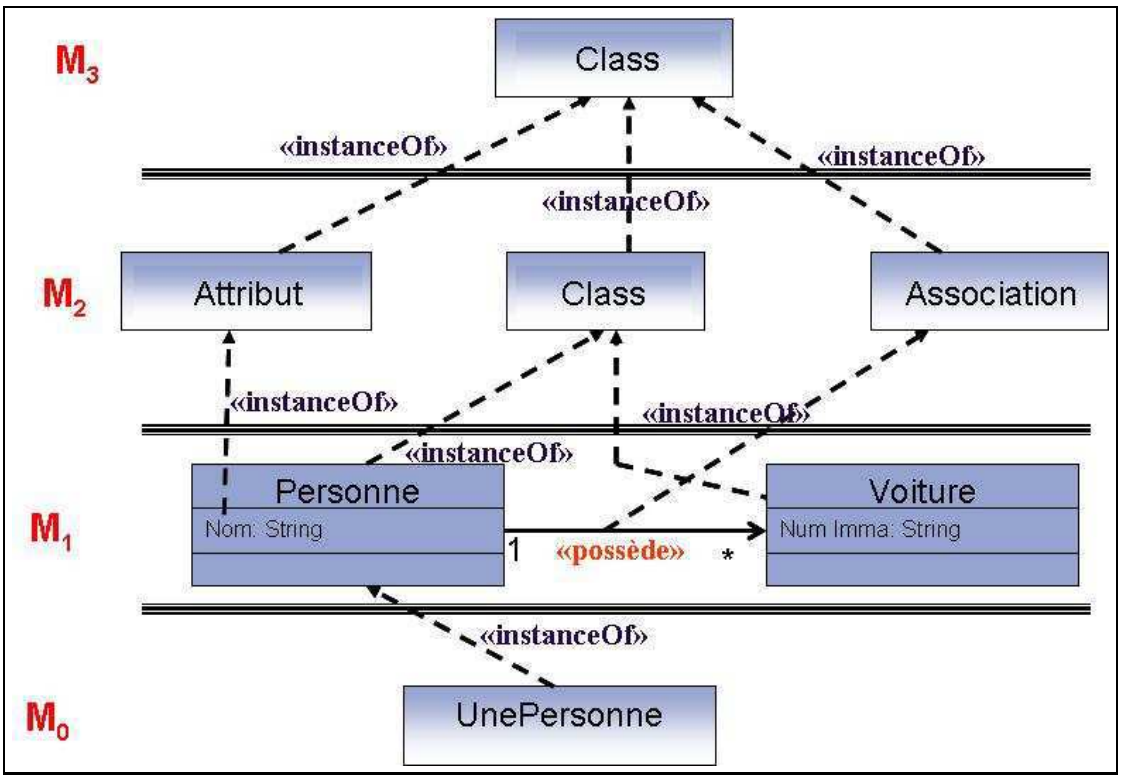

FIG. *6.3 - Exemple de métamodélisation du langage UML*

Ainsi, toute entité du niveau *M1* aura pour type une classe ou un type de données défini au niveau *M2*. De même, toute entité du niveau *M2* aura pour type une classe ou un type de données défini au niveau *M3*. Une classe MOF définie au niveau *M3* trouvera ses instances au niveau *M2* et ainsi de suite. Parmi ces concepts, seuls les concepts de *classes* et de *types de données* définissent des types susceptibles d'avoir des instances.

Objectifs

MOF est un langage de modélisation utilisé pour représenter des métamodèles et les modèles qui en découlent. Il a pour but de fournir un cadre de travail supportant tout type de métamodèles et permettant, au fur et à mesure de l'évolution des besoins, d'enrichir ces métamodèles avec de nouveaux concepts.
Grâce à son ouverture, MOF peut supporter un grand nombre d'utilisations, de même qu'il peut être supporté par un grand nombre d'applications. Ainsi, il peut être utilisé pour définir le métamodèle d'un domaine d'intérêt particulier. MOF est alors considéré comme un méta-métamodèle car il définit des métamodèles. Actuellement, on le retrouve dans la définition du métamodèle du langage UML.

Pour conclure, nous pouvons dire que MOF est un langage de modélisation à « objets ». Ainsi, il est utilisé pour définir la structure et la sémantique de métamodèles, qu'ils soient généralistes ou liés à un domaine plus spécifique (comme dans notre cas). Par

exemple, il est bien adapté pour définir à la fois des métamodèles des formalismes à « objets », des métamodèles plus traditionnels (relationnels, entité-association) ou tout autre métamodèle élémentaire.
L'un de ses avantages est qu'il supporte la spécification. En effet, il est utilisé pour spécifier la technologie et le domaine spécifique aux métamodèles et la définition des objets. Le processus de spécification utilise les notations UML (Classes, Associations, associations n-aires, etc.) Par ailleurs, les systèmes compatibles MOF peuvent s'échanger des metadonnées en utilisant CORBA à partir d'une variété de langage (Java, C++, etc.).

Notation du MOF

Comme nous l'expliquons dans le paragraphe 6.3.1, MOF est basé sur un cadre de modélisation objet, sous-ensemble du cœur de UML. Ainsi, la notation graphique utilisée pour représenter les métamodèles MOF est la notation UML « adaptée » au MOF. Par conséquent, les associations et les classes représentées via cette notation ne sont pas des associations et des classes UML, mais des associations et des classes MOF. MOF, par sa réflexivité, peut se représenter via cette notation. Les concepts de classes, d'attributs, d'associations et de méthodes ont été introduits dans MOF grâce à la notation UML.

Par ailleurs, il convient de rappeler que MOF fournit un environnement non fermé pour la modélisation car il peut être étendu par héritage ou par composition de manière à représenter des modèles plus évolués. Il fournit ainsi un ensemble d'éléments de modélisation servant à construire des métamodèles, tout en incluant des règles pour leur utilisation.
Pour montrer l'impact de ce concept, rappelons que l'approche de modélisation utilisée dans MOF se retrouve aussi dans le langage XML où les niveaux de l'architecture de métamodélisation, proposée par MOF, correspondent respectivement à des niveaux du langage XML. Ainsi, les niveaux *M0*, *M1*, *M2* et *M3* correspondent respectivement aux données du système, aux données modélisées en XML, à la DTD de XML et au langage XML lui-même.

6.3.2 L'exemple du XMI

Comme nous l'expliquons en introduction de la section 6.3, XMI (XML Metadata Interchange) est un format d'échange basé sur la sémantique du MOF et sur la syntaxe du langage XML. Il permet ainsi la définition, l'interaction, la manipulation et l'intégration des données et des objets XML. Ainsi, il fournit des règles par lesquelles un schéma peut être généré pour tout métamodèle basé sur MOF. Ce format est issu d'une mise en correspondance des descriptions MOF de métamodèles et de la façon de définir celles-ci dans une DTD ou un schéma XML (cf. figure 6.4). L'application de cette mise en correspondance sur un métamodèle permet d'obtenir une DTD ou un schéma standard pour l'échange des modèles décrits à l'aide de ce métamodèle. ***L'utilisation de ce standard***

dans notre travail se justifie par le fait qu'il permettra à divers outils de classification de s'échanger les modèles d'interprétation basés sur notre métamodèle dédié à l'aide à l'interprétation des résultats de classification automatique.
Pour aller dans le même sens, rappelons qu'il existe une DTD standard permettant l'échange de modèles UML[2] ainsi qu'une DTD standard permettant l'échange de métamodèles MOF. Certains outils du génie logiciel (Rational Rose, ArgoUML, Objecteering, etc.) permettent d'exporter et d'importer les modèles sous forme de documents XMI.

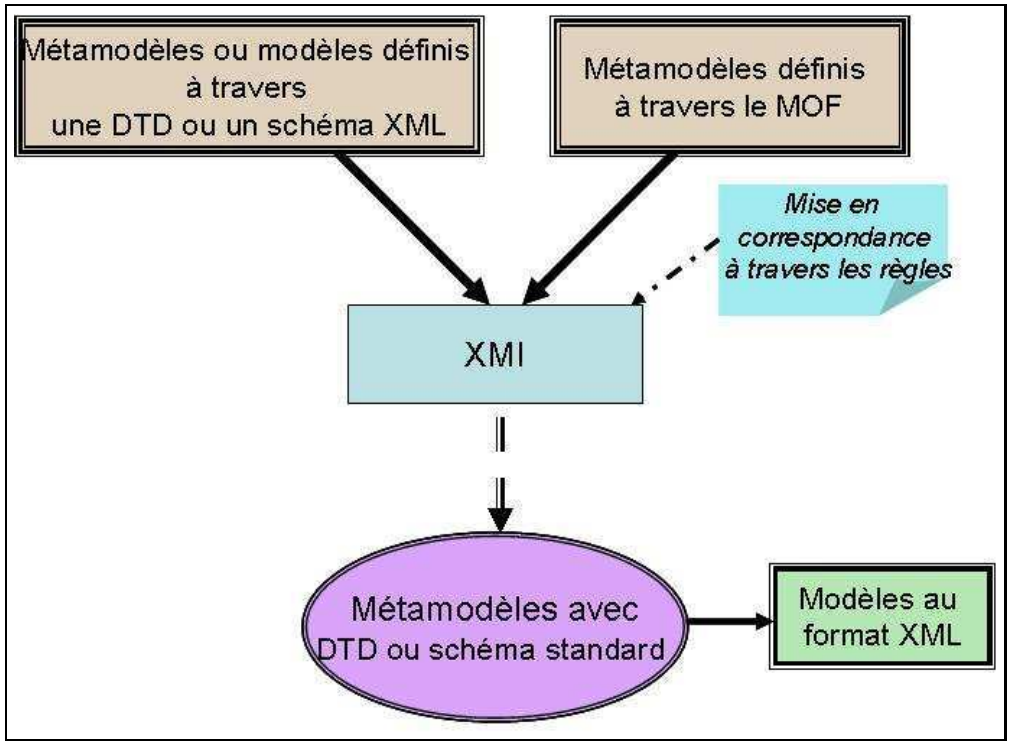

FIG. *6.4 - Fonctionnement du standard XMI*

Pour récapituler, rappelons que le XMI permet :

- le traitement et la manipulation des métadonnées (création de DTD exprimant une structure, génération et manipulation de documents XML) ;
- la mise au point des modèles en réalisant l'échange des notations UML ;
- la fourniture d'un ensemble de règles pour traduire les modèles UML.

L'effet économique et technique est important parce que jusqu'ici il était pratiquement impossible à un groupe de travailler sur des modèles communs en utilisant des outils de conception différents. XMI est l'outil permettant de travailler de façon concourante.

[2]Unified Modelling Language : langage graphique conçu pour décrire, spécifier, développer et documenter un système informatique

Objectifs

L'objectif principal de XMI est de permettre l'échange de métadonnées entre outils de modélisation basés sur UML et la communication des répertoires de métadonnées basés sur MOF. Il faut comprendre que les *métadonnées* décrivent les métamodèles à échanger. Ce type de modélisation s'avère donc très utile dans un environnement où il est impossible pour des utilisateurs de travailler sur des modèles communs. C'est le cas des systèmes utilisant des outils de conception provenant de différents fournisseurs. C'est le cas aussi des outils de fouille de données utilisant divers critères pour interpréter les résultats issus de la classification.

En définitive, XMI et ses règles d'utilisation correspondantes peuvent jouer les rôles suivants :

- permettre le traitement et la manipulation des métadonnées (création de DTD exprimant une structure, génération et manipulation de documents XML correspondants) ;
- permettre la mise au point des modèles pour des ensembles d'acteurs hétérogènes en réalisant l'échange des notations UML ;
- fournir un ensemble de règles pour traduire les modèles UML et produire les messages correspondant à des scénarios d'échange.

Prenons un exemple concret d'utilisation du standard XMI. Il s'agit de deux composantes A et B (appartenant ou non au même domaine) qui souhaitent développer des échanges d'information. Les vues du monde de ces composantes sont exprimées par des modèles différents. Cependant, leurs relations se traduisent obligatoirement par une vue commune représentant une partie de leurs activités. La première étape consistera à exprimer leur scénario d'échange (ou d'interaction) dans le langage UML. Ensuite l'expression en XMI des éléments relatifs à leur relation permettra de mettre en place de manière progressive un scénario commun de communication entre ces deux composantes.

6.3.3 L'exemple du CWM

Il s'agit de rappeler dans cette section qu'il existe des travaux génériques relatifs à la métamodélisation du domaine de la fouille de données.
Ces travaux, menés par le groupe OMG, consistent à proposer un métamodèle décrivant trois domaines conceptuels de la fouille de données (*le modèle de description, les programmes et les attributs*). Il s'agit du standard CWM (*Common Warehouse Metamodel*) [CWM, 2006].

Comme nous le rappelons en introduction de la section 6.3, CWM a été conçu pour faciliter l'interopérabilité des métadonnées entre les outils d'entrepôt de données et les répertoires de métadonnées dans un environnement distribué et hétérogène.
Ainsi, compte tenu de l'hétérogénéité des entrepôts de données, ce standard permettra

aux utilisateurs d'enrichir de manière sémantique les données contenues dans ces entrepôts.
En effet, CWM fournit un cadre de représentation des métadonnées sur les données originales, les données cibles, les transformations et les analyses sur les données, les traitements et les opérations qui ont permis la création et la gestion de l'entrepôt de données. Il va de soi que l'utilisation de ces métadonnées permettra une meilleure utilisation des données contenues dans les entrepôts en fournissant les informations nécessaires à leur traitement. Les métadonnées importées ou exportées dans ce format peuvent être échangées à l'aide d'outils d'entrepôt qui interprètent les métadonnées conformes au standard CWM.

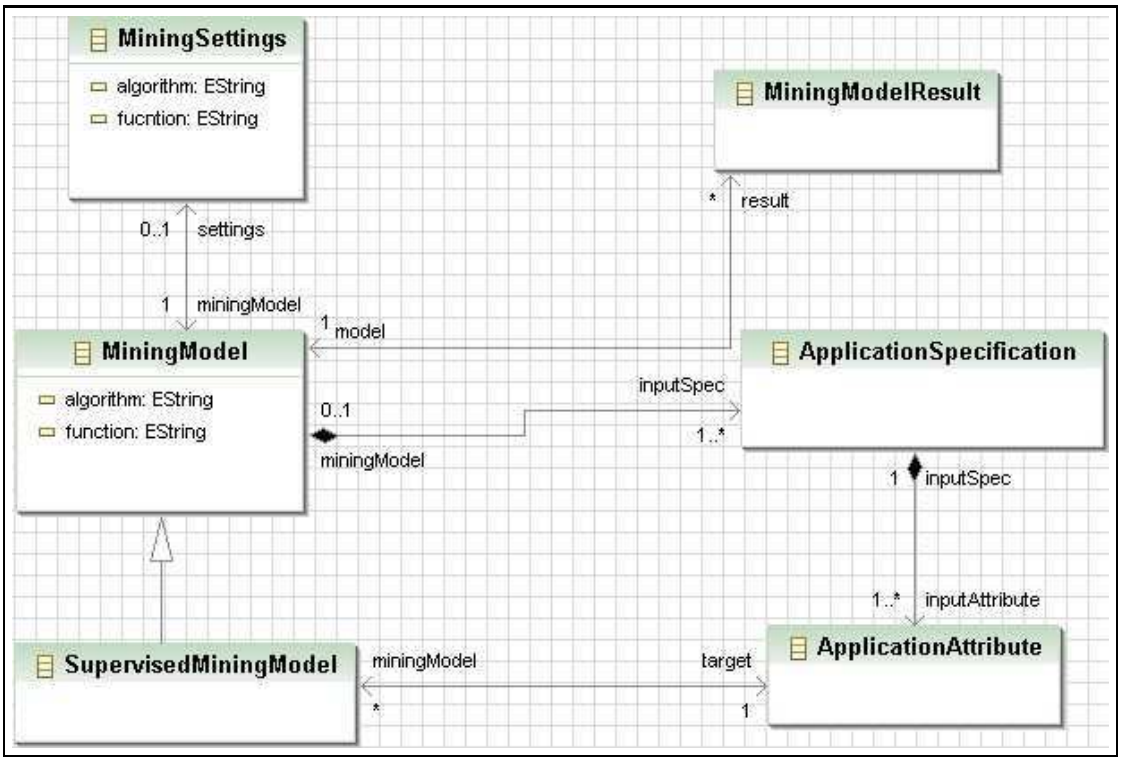

FIG. *6.5 - Métamodèle CWM pour la fouille de données (figure tirée de [CWM, 2006])*

Dans notre travail, nous utilisons ce standard comme point de départ de la représentation générique d'un modèle de fouille de données. Concrètement ce métamodèle décrit un modèle mathématique produit ou généré par l'exécution d'un algorithme de fouille de données. Ainsi, nous pouvons voir que le concept *MiningModel (Modèle de fouille de données)* est **composé** de plusieurs *ApplicationSpecification* (il s'agit de spécifications d'applications d'un modèle de fouille de données). Ce même concept de *MiningModel* fournit un résultat défini par le concept *MiningModelResult* (le résultat d'un modèle de fouille de données) et dispose d'un ensemble de programmes ou application (concept *MiningSettings*, qui contient les différents algorithmes et/ou fonctions d'un modèle de fouille de données). Enfin, nous avons le concept *SupervisedMiningModel* qui, comme son nom l'indique, est une spécification du modèle de fouille de données. Ce même concept dispose

d'attributs (concept *ApplicationAttribut* qui compose le compose le concept *ApplicationSpecification*). La figure 6.5 montre la partie de ce métamodèle relative au domaine de la fouille de données.

Synthèse de la section 6.3 et positionnement

Dans cette section, nous avons présenté quelques techniques de métamodélisation utilisées dans le domaine de l'ingénierie des connaissances. Nous avons commencé par expliquer le format d'échange *XMI*, format d'échange d'UML fondé sur le langage XML, qui permet de sérialiser et d'échanger des modèles sur des plate-formes et des systèmes. Nous avons expliqué que ce type de modélisation est très utile dans le cadre de nos travaux car il permet de modéliser diverses approches d'interprétation suivant un modèle générique compréhensible par tous.

Ensuite, nous avons expliqué ce que représente le standard *CWM* dans le cadre de cette thèse. En effet, nous nous appuyons sur la modélisation de ce standard pour définir notre métamodèle. A travers cette spécialisation, l'objectif est de montrer que notre approche s'intègre avec l'existant.

Enfin, après avoir expliqué ce qu'est le *MOF*, nous avons montré les liens entre notre métamodèle et *MOF*. Ainsi, nous nous sommes inspirés de l'architecture de *MOF* pour modéliser notre architecture d'exploitation de métadonnées définie au Chapitre 7.

6.4 Apports de la métamodélisation

Chaque outil de classification utilise un formalisme particulier dans l'interprétation des résultats de classification qu'il produit. Par exemple, le logiciel Weka utilise, entre autres, le format *csv* pour représenter ses résultats. De plus, il utilise les mêmes critères prédéfinis pour aider à l'interprétation et à la validation des résultats. Or, d'une méthode de fouille à une autre, il n'est pas possible d'utiliser les mêmes critères. Toutefois, de nombreux recoupements existent entre ces divers formalismes. Il convient donc de pouvoir les identifier afin d'optimiser l'automatisation du processus d'interprétation des résultats de classification. Pour les identifier et les utiliser à bon escient, les techniques de métamodélisation constituent une des réponses possibles à ce problème.
En effet, la métamodélisation est une technique de définition des concepts à utiliser pour modéliser des systèmes. La métamodélisation apporte ainsi la flexibilité nécessaire à la fourniture de moyens adaptés aux besoins d'un processus logiciel, pour concevoir des applications. La métamodélisation est donc une tentative pour *décrire un domaine* et ce pour un *objectif particulier*.

Pour montrer l'apport de la métamodélisation, rappelons que leur utilisation est présente à différents niveaux dans les systèmes d'information. Ainsi, les intergiciels comme CORBA [OMG, 2006], EJBs [DeMichiel et al., 2000], .Net [Thai and Lam, 2001]CCM [OMG, 2006], J2EE [J2EE, 2006] utilisent la notion de métamodélisation. En effet, ils pro-

posent des abstractions pour définir des applications indépendamment des plates-formes ou des langages à utiliser lors de leur mise en œuvre. Toutefois, il n'existe pas d'intergiciel universel et leur multiplication entraîne le besoin croissant d'un niveau d'abstraction supérieur. Cette abstraction doit permettre de capitaliser la définition des applications et les rendre plus pérennes et portables au dessus de ces différentes technologies. L'utilisation de modèles de définition des applications permet d'atteindre ce niveau d'abstraction. Enfin, rappelons que la définition de modèles requiert la mise en œuvre de métamodèles ; c'est dire toute l'importance que revêt ce concept dans le domaine de l'ingénierie des connaissances.

Par ailleurs, comme nous l'expliquions tantôt, il ne peut pas y avoir un métamodèle universel utilisable pour décrire tous les systèmes d'information. Il est donc important de comprendre qu'un métamodèle doit être défini pour un objectif précis. Un aspect très intéressant d'un métamodèle est celui relatif à la sémantique.
En effet, il est important de donner un sens aux éléments définis dans un métamodèle. Il est encore plus important de partager ce sens entre ses différents utilisateurs. L'utilisation d'assertions logiques ou de tout autre moyen de description formelle [Bézivin et al., 1995], [Saeki, 2000] sont des possibilités permettant d'introduire de la sémantique dans un métamodèle.
Dans notre travail, ***l'introduction de la sémantique*** se fera par l'utilisation d'une ontologie du domaine de la classification (cf. Chapitre 8).

6.5 Axes d'interprétation des résultats de classification automatique

Les études bibliographiques réalisées dans le cadre de l'interprétation des résultats de classification ont montré que si les critères utilisés dans ce processus sont très homogènes, tous les processus d'interprétation obéissent à un certain nombre de règles. En effet, le processus d'interprétation peut être très différent suivant les méthodes de classification et les outils utilisés pour les implémenter. Néanmoins, tous ces processus d'interprétation partagent un certain nombre de concepts communs que nous avons décidés de modéliser. Ainsi, dans tout processus d'interprétation, les résultats sont interprétés suivant trois axes :

- *l'axe « structure classificatoire » :* l'interprétation des résultats issus d'une méthode hiérarchique ne sera certainement pas la même que celle relative aux résultats produits par une méthode de partitionnement. En effet, dans le premier cas on tâchera de chercher la bonne coupure de l'arbre alors que dans le second cas on s'attachera à trouver les classes ayant une certaine caractéristique.

- *l'axe « individus » :* une stratégie d'interprétation peut consister par exemple à rechercher les individus (ou objets) présentant une certaine caractéristique. Ainsi, on est amené à trouver une explication sémantique à chaque classe construite (mais en pratique il arrive que beaucoup de classes ne fournissent aucune sémantique claire

sur les individus qui la composent). A travers cet axe nous aurons par exemple les concepts d'*individu central*, d'*individu outlier* ou d' *individu aberrant (parangon)*.

- *l'axe « variables » :* à travers cet axe, on peut mesurer la contribution d'une variable dans la construction d'une classe : on parlera ainsi de *variable discriminante*. On peut aussi chercher à trouver les variables dites *prédictives* ou *actives* et ce, à travers leur rôle dans le processus de classification.

Dans la section 6.7, nous spécifions dans quelle mesure ces axes sont utilisés dans la modélisation de notre approche. Mais avant, nous évoquons dans la section 6.6, la stratégie de modélisation que nous adoptons dans cette thèse.

6.6 Notre stratégie de modélisation

Comme le rappelle NOY dans [Noy et al., 2000], il n'y a pas une méthode correcte et unique pour modéliser un domaine. Une bonne organisation des informations du domaine dépend fondamentalement des finalités de l'application et des profils des utilisateurs auxquels elle vise à fournir des services. Ainsi, il peut y avoir plusieurs stratégies de modélisation des connaissances d'un domaine.

Dans notre approche, nous optons pour une stratégie basée sur l'utilisation d'une approche générique basée sur un métamodèle. Ce métamodèle représente les domaines conceptuels basés sur les axes d'interprétation que nous évoquons dans la section 6.5. Dans ce travail, le métamodèle dédié au processus d'interprétation est un outil permettant la création des différents modèles d'interprétation des résultats de classification. Concrètement, il s'agit de modéliser le domaine de l'interprétation des résultats à travers la perception des experts en charge des tâches d'interprétation. Rappelons par ailleurs que nous avons choisi la classification comme domaine d'application pour des raisons liées à la complexité de ce domaine et au fait qu'il n'existe pas, à notre connaissance, de travaux relatifs à l'automatisation du processus d'interprétation.

6.7 Spécification de notre Métamodèle d'Interprétation

Dans le paragraphe 6.3.3, nous expliquons qu'il existe des travaux relatifs à la (méta)-modélisation du domaine de fouille de données. Ces travaux, réalisés sous l'égide de l'OMG, ont été une source d'inspiration pour notre métamodèle. Les différents concepts utilisés dans la construction de notre métamodèle ont été extraits des travaux réalisés dans le domaine de la fouille de données et relatifs au processus d'interprétation des résultats de classification. L'intérêt d'un tel métamodèle est d'expliciter les différentes notions utilisées dans le domaine de l'interprétation, de déterminer ce qu'ils recouvrent et de mettre à jour les relations entre les différents concepts le composant.

Pour construire notre métamodèle, nous sommes partis du constat selon lequel les différents modèles d'interprétation des résultats de classification ne sont pas forcément

disjoints et qu'il existe ainsi un certain nombre de concepts partagés par ces modèles. La figure 6.6 montre la relation entre un modèle d'interprétation et un outil implémentant les méthodes de fouille de données. Le métamodèle permettra d'abstraire cette relation en définissant les concepts communs à tous les modèles d'interprétation. Cette modélisation permet de définir le fait qu'un **métamodèle d'interprétation** permet de définir les comportements de tous les *modèles d'interprétation* associés aux outils de fouille de données. Dans les sections suivantes, nous spécifions le métamodèle qui nous a permis de définir les contraintes à respecter par les modèles d'interprétation.

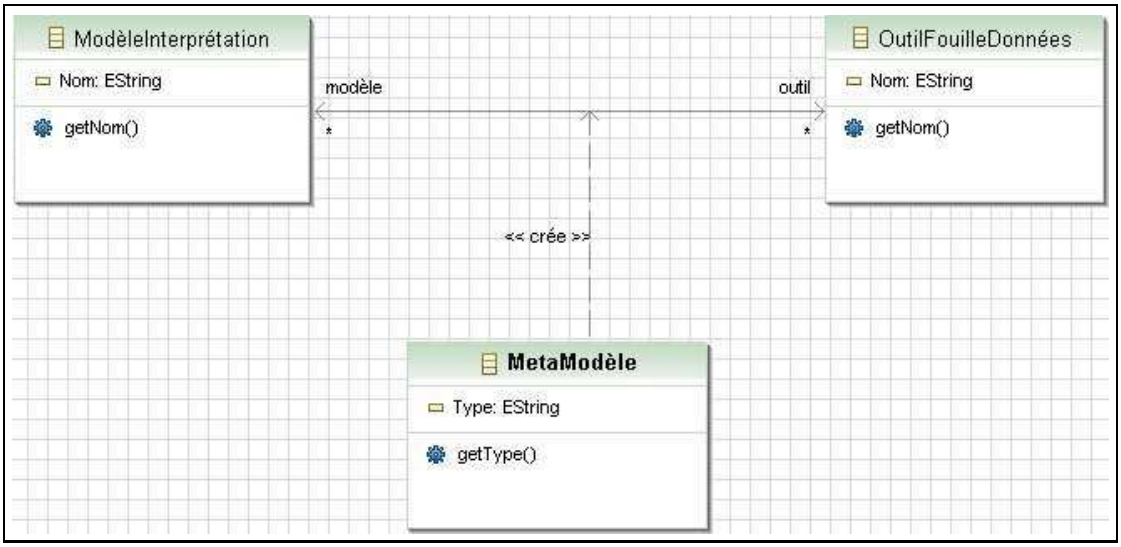

FIG. *6.6 - Relation entre outil de fouille de données, modèle d'interprétation et métamodèle*

6.7.1 Architecture générale de notre métamodèle

En se basant sur la stratégie de modélisation proposée par l'OMG et évoquée dans la section 6.3.1, nous proposons une architecture de modélisation à six (6) niveaux d'abstraction. La figure 6.7 montre une représentation en langage UML (réalisée avec le plug-in *EclipseUML* de *eclipse*) de cette architecture. Dans cette figure, nous expliquons les principaux concepts de notre métamodèle. Ainsi, le concept de *ModèleInterprétation* est **composé** d'un ensemble de concepts *ElémentInterprétation* dont chacun est **composé** d'un ensemble de concepts *Attribut*. De plus, un *ModèleInterprétation* est **décrit** suivant trois (3) axes d'interprétation (*AxesInterprétation*) ; il s'agit des axes de *structure classificatoire (StructureClassificatoire)*, de *variable* et d'*individu*. Enfin, le concept *AxesInterprétation* sert à **interpréter** le concept *ResultatClassification* qui est la spécialisation du concept *Résultat* qui caractérise les résultats de toutes méthodes de fouille de données. Voici les différents niveaux de notre métamodèle correspondant aux concepts décrits précédemment :

- *Niveau métamodèle :* Le niveau métamodèle doit permettre l'intégration des différents points de vue nécessaires à l'interprétation des résultats de classification automatique. La construction de notre métamodèle se base sur trois point de vues ou axes d'interprétation (*structure classificatoire, individus et variables*).

- *Niveau modèle :* A chaque axe d'interprétation du niveau métamodèle correspond un ou plusieurs modèles d'interprétation. Suivant la stratégie de modélisation utilisée, ces modèles peuvent être préexistants (approche *top-down*) ou peuvent être développés à partir des données (approche *bottom-up*). Par exemple, le *modèle d'interprétation par les individus* est relatif à l'axe d'interprétation *individus*.

- *Niveau concept :* Chacun des modèles du niveau précédent est décrit par un ou plusieurs concepts (ou éléments d'interprétation) du domaine. Par exemple le modèle d'interprétation par les individus est décrit par les concepts d'*Individu central*, d'*Individu outlier* ou d'*Individu aberrant*.

- *Niveau attribut :* Ces éléments permettent de caractériser les concepts décrits au niveau précédent. Un même attribut peut décrire plusieurs concepts appartenant à des modèles d'interprétation différents. *Exemple* : l'attribut *cardinalité* caractérise le concept *Classe*. Aussi l'attribut *nom* décrit aussi bien le concept *Classe* que le concept *Variable*.

- *Niveau instance :* Ce niveau correspond à l'affectation des valeurs aux attributs définis au niveau précédent.

- *Niveau objet :* Ce niveau représente le phénomène ou le système étudié. Dans notre cas, il s'agit du processus d'interprétation des résultats des méthodes de classification.

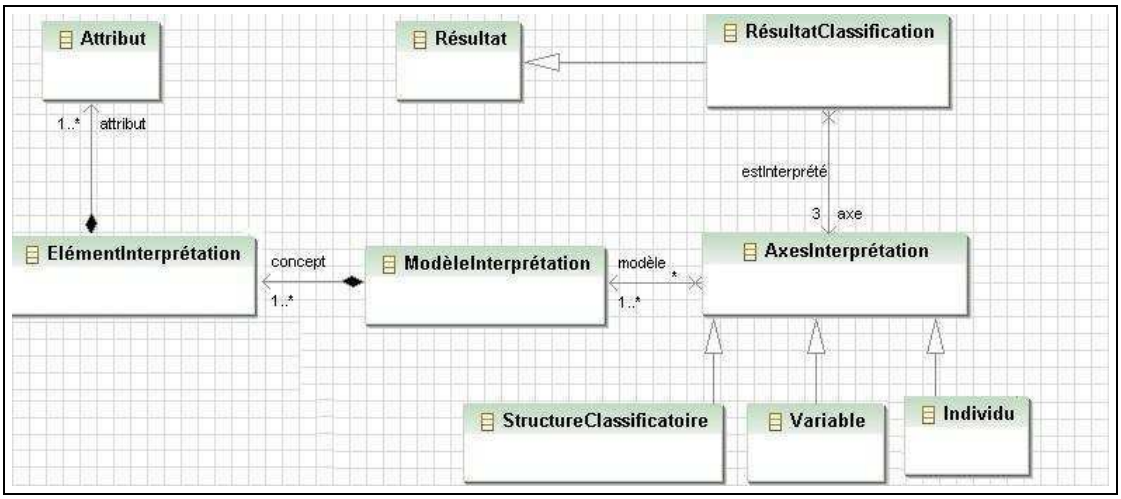

FIG. *6.7 - Architecture de notre métamodèle*

6.7.2 Spécification des concepts de notre métamodèle

Dans cette section, nous expliquons les concepts contenus dans notre métamodèle. Il s'agit de montrer les relations existantes entre ces concepts d'une part et entre ces concepts et des concepts provenant d'autres métamodèles d'autre part. Concrètement, nous définissons les différents concepts cités précédemment.

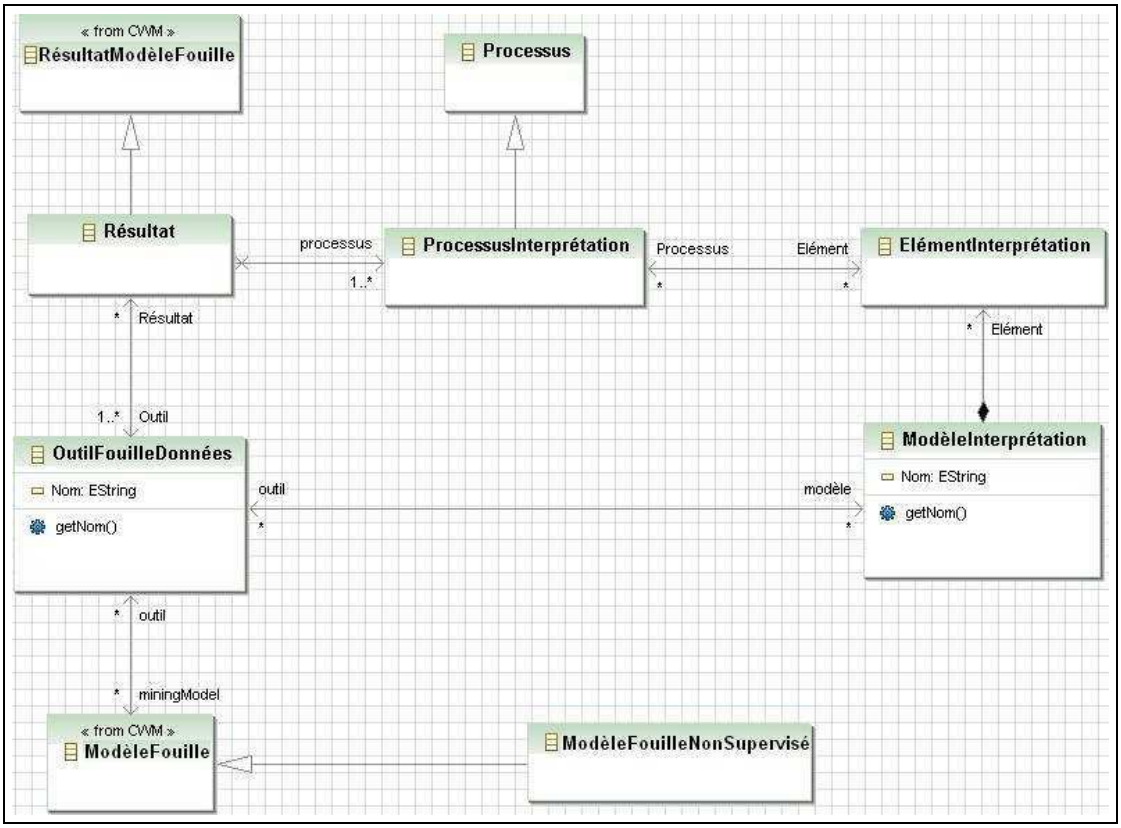

FIG. *6.8 - Le concept de processus d'interprétation*

Nous commençons par les concepts de ***ModèleInterprétation, ElémentInterprétation*** et ***Résultat***. La figure 6.8 montre la relation entre ces concepts et ceux issus de CWM qui, comme nous le rappelions, a servi de base à la réalisation de notre métamodèle. Ainsi, pour un concept *Résultat*, nous pouvons avoir plusieurs processus d'interprétation (concept *ProcessusInterprétation*) (qui hérite des comportements du concept *Processus* décrivant tous types de processus en fouille de données et issu du métamodèle CWM). Un processus d'interprétation est relatif à un résultat de classification. De plus, un *Résultat* spécifie le résultat d'un modèle de fouille de données (concept *ResultatModèleFouille*) (provenant du métamodèle CWM). Par ailleurs, pour un outil de fouille de données (concept *OutilFouilleDonnées*), nous avons un ou plusieurs modèles d'interprétation (concept *ModèleInterprétation*) correspondants. Et ce même outil implémente plusieurs modèles de fouille de données (concept *ModèleFouille*) (provenant du métamodèle CWM). Ce dernier est spécialisé par le concept *ModèleFouilleNonSupervisé* qui

caractérise les méthodes non supervisées (par exemple une méthode de partitionnement). Enfin, un processus d'interprétation se base sur un ou plusieurs éléments d'interprétation (concept *ElémentInterprétation*).

Ensuite, nous expliquons les relations entre les concepts ***ElémentInterprétation, Métadonnée*** et ***Critère***. En effet, comme nous pouvons le constater sur la figure 6.9, un modèle d'interprétation (concept *ModèleInterprétation*) est composé d'au moins un élément d'interprétation (concept *ElémentInterprétation*). Ensuite nous montrons des exemples de concepts de *métadonnées* et de *critères*. Ainsi, les concepts *MétadonnéeDescriptive* et *MétadonnéeInterprétative* héritent du concept *Métadonnée* et permettent de différencier le rôle des métadonnées utilisées pour décrire ou interpréter un résultat de classification. De même, les concepts *CritèreValidation* et *CritèreInterprétation* héritent du concept *Critère* et permettent de différencier les critères utilisés pour valider un résultat de classification et ceux utilisés pour interpréter ces mêmes résultats. Avec cette modélisation, nous sommes sûrs d'avoir les concepts génériques qui correspondant aux éléments entrant en ligne de compte dans le processus d'interprétation.

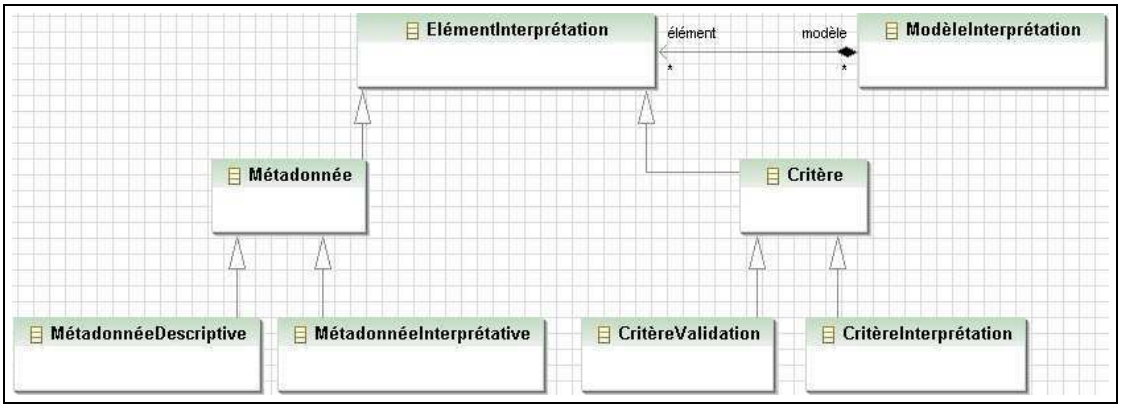

FIG. *6.9 - Le concept d'élément d'interprétation*

Puis, nous présentons le concept *AxeInterprétation* ainsi que sa relation avec d'autres concepts de notre métamodèle. Ainsi, un résultat d'une méthode de classification (concept *RésultatClassification*) peut être interprété suivant 1, 2 et/ou 3 axes : *structure classificatoire, individus et variables*. Ces trois concepts héritent donc du concept *AxeInterprétation* et permettent d'interpréter tous résultats de méthodes de fouille de données suivant l'un de ces axes. Ces trois axes d'interprétation sont aussi en relation avec le concept de *Classe*. En effet, une *structure classificatoire* est le **produit** d'une ou de plusieurs *classes*. De même, une *classe* est **décrite** par une ou plusieurs *variables*. Enfin, une *classe* est formée par la composition d'un ou de plusieurs *individus*. A chacun de ces axes d'interprétation, correspond un modèle d'interprétation de résultats. La figure 6.10 montre la description, en langage UML, de ce que nous venons d'expliquer dans ce paragraphe.

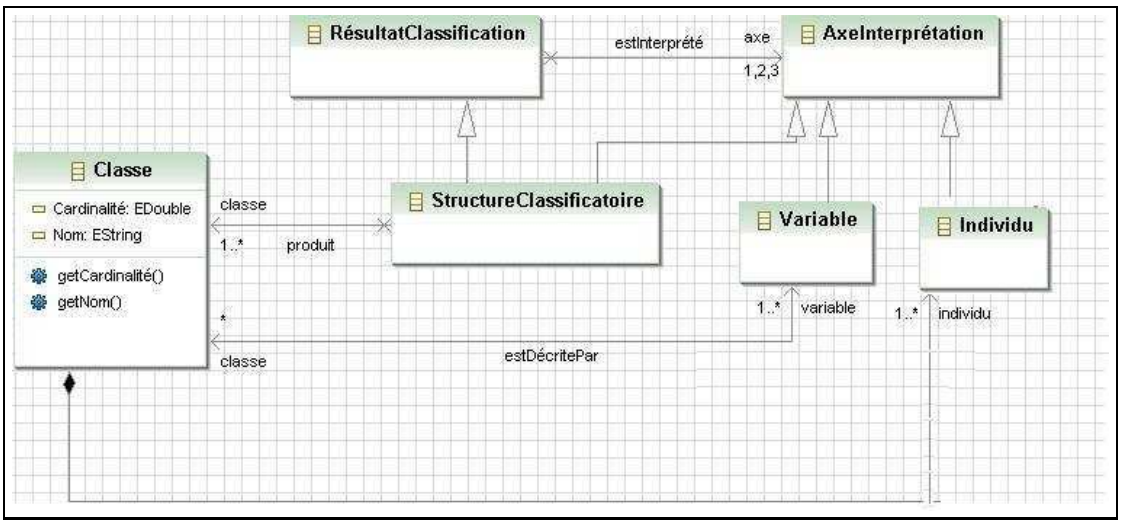

FIG. *6.10 - Le concept d'axe d'interprétation*

6.8 Exemple de Modèle d'interprétation

Dans cette section, nous montrons un modèle d'interprétation utilisant l'axe d'interprétation *Variable*. Nous présentons un modèle d'interprétation souvent utilisé par les experts dans le cadre de l'analyse des résultats de classification. Notre objectif est de montrer sur un exemple concret l'utilisation de notre métamodèle. Pour une vue complète de notre diagramme de classes cf. annexe A.

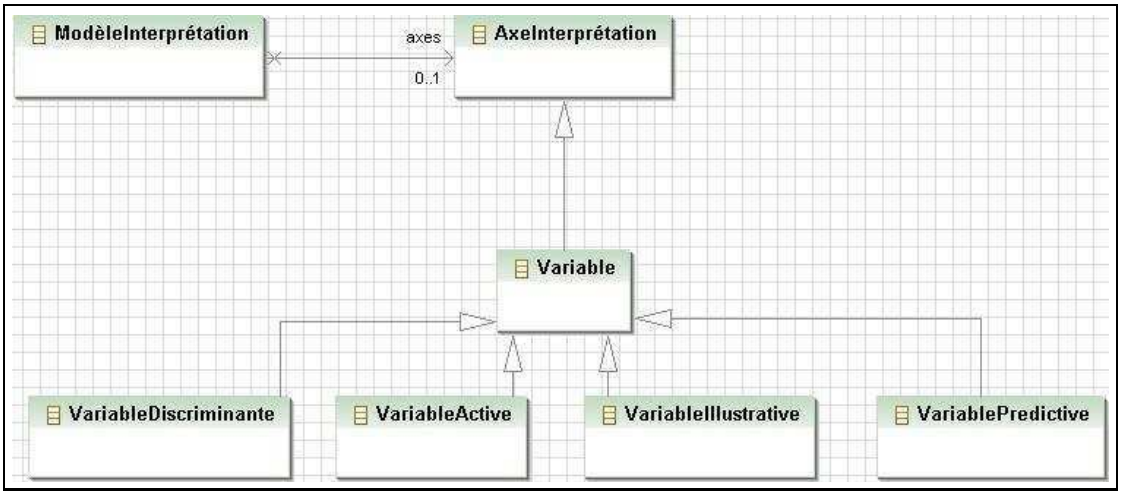

FIG. *6.11 - Modèle d'interprétation dirigé par les variables*

Ainsi, à l'issue d'une classification, les utilisateurs peuvent se poser la question de savoir quelles sont les variables qui ont été déterminantes au cours de leur classification. La figure 6.11 présente les concepts utilisés dans le modèle d'interprétation des résultats suivant l'axe *Variable*. Nous remarquons que, suivant cet axe, nous avons quatre (4)

concepts : les concepts *Variable Discriminante, Variable Active, Variable Illustrative* et *Variable Prédictive*. L'objectif dans ce modèle d'interprétation est de définir des scénarios d'interprétation relatifs aux rôles des variables lors du processus de classification. Ce rôle est déterminé grâce à la méthode de calcul utilisé dans l'outil de classification ou par l'expert du domaine de la méthode utilisée.

6.8.1 Structure de notre fichier de métadonnées

L'objectif de notre modélisation est de mettre à la disposition de l'utilisateur un fichier descriptif des résultats qui soit le plus générique et le plus sémantique possible. Pour ce faire, nous avons décidé que le fichier de métadonnées aura les composantes suivantes :

- *la composante Entête :* qui donne des informations descriptives ; ces informations proviennent du standard Dublin Core, il s'agit par exemple de la *date de création*, le *titre de l'étude*, etc.
- *la composante Module :* cette composante décrit les informations sur le module et/ou l'outil de classification. Il s'agit par exemple des *paramètres de la méthode de classification*, du *nom* et de la *version* du module ;
- *la composante Fichier :* cette composante décrit l'ensemble des fichiers (*sources* ou *résultats*) ayant servi au processus d'interprétation. Il s'agit par exemple de donner leur emplacement et/ou de pouvoir y accéder à des fins particulières ;
- *la composante Expérience :* qui permet de contenir l'ensemble des critères et/ou éléments utilisés pour valider ou interpréter les résultats de méthodes de classification.

6.8.2 Cas d'utilisation

Le cas d'utilisation utilisé dans ce paragraphe est celui qui consiste à rechercher les *variables les plus discriminantes* à l'issue d'un processus de classification. Dans ce contexte, rappelons que le processus d'interprétation s'effectue en deux phases bien distinctes :

- *la création du fichier de métadonnées :* il s'agit de la phase qui consiste à la création du fichier de métadonnées lors de l'exécution d'un algorithme de classification. Le fichier ainsi obtenu contient l'ensemble des éléments de métadonnées qui seront utiles dans le processus d'interprétation. Ce fichier peut aussi être obtenu en utilisant les fichiers résultats fournis par les méthodes de fouille de données (par exemple fichier au format PMML). Dans tous les cas ce fichier de métadonnées devra respecter la structure définie dans le paragraphe précédent ;
- *l'exploitation du fichier de métadonnées :* il s'agit d'utiliser le fichier de métadonnées obtenu précédemment en vue de l'interprétation. En effet, une fois que nous obtenons le fichier de métadonnées au format XML, il s'agit traduire les scénarios d'interprétation utilisés en langage XQUERY afin de fournir les résultats espérés.

Dans ce paragraphe, nous supposons que nous disposons du fichier de métadonnées servant de cadre d'interprétation. Ainsi, la figure 6.12 explique la manière avec laquelle une demande d'un utilisateur est prise en compte dans le cadre d'une interprétation basée sur l'axe *Variable*.
Ainsi, l'utilisateur commencera par choisir le fichier de métadonnées sur lequel il souhaite faire une interprétation. Une procédure de vérification est alors déclenchée afin de vérifier si le fichier chargé est valide du point de vue de notre structure décrité dans le paragraphe précédent.

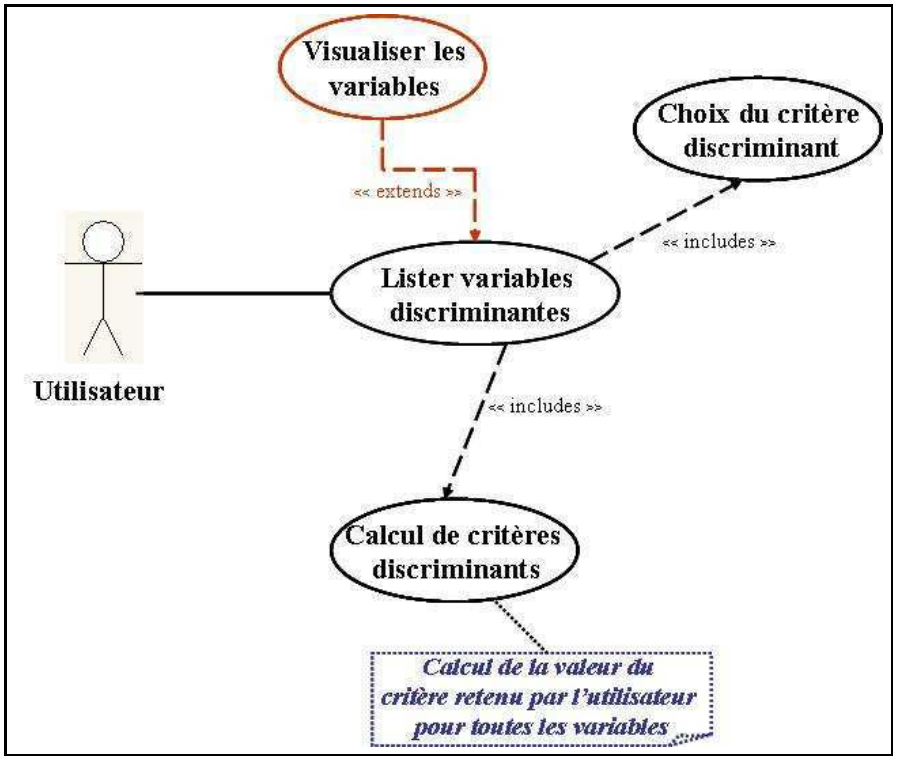

FIG. *6.12 - Scénario de cas d'utilisation*

Décrivons le cas d'utilisation qui correspond à la demande faite par un utilisateur pour connaître les variables qui ont été déterminantes lors de sa classification. Nous avons l'utilisateur qui va chercher à connaître les variables les *variables discriminantes* (action *Lister variables discriminantes*). Cette action est en relation avec d'autres actions qu'elle utilise ou qui l'étendent. Ainsi, lister les variables discriminantes lors d'un processus de classification inclut de **choisir** le *critère discriminant* et de *calculer les critères discriminants* retenus. Le premier est le fruit du choix d'un expert du domaine de la classification ou de l'utilisateur. Tandis que le second s'obtient à travers un calcul (sur les critères d'interprétation contenus dans le fichier de métadonnées) proposé par l'expert du domaine ou par l'utilisateur. De même l'action de *lister les variables discriminantes* est **étendue** grâce à la *visualisation des variables discriminantes*. Dans le paragraphe suivant, nous expliquons plus en détail les séquences par lesquelles ce cas d'utilisation est réalisé.

6.8.3 Diagramme de séquences

Nous présentons le déroulement du scénario d'interprétation défini au paragraphe précédent. Il s'agit de montrer les différentes interactions mises en œuvre pour répondre à une demande d'interprétation d'un utilisateur (cf. figure 6.13). Pour ce scénario d'interprétation, nous supposons que le fichier de métadonnées à interpréter a été validé et chargé en mémoire. Voici le déroulement de ce scénario d'interprétation :

- l'utilisateur, à travers l'interface de notre outil, choisit le scénario qui correspond à la recherche de variables discriminantes ;
- cette requête est récupérée par la méthode chargée de la traiter au niveau de notre ontologie. C'est grâce à cette dernière que nous récupérons les critères à utiliser dans l'interprétation ainsi que la formule correspondante à ce scénario d'interprétation ;
- après il s'agit de convertir ce scénario d'interprétation en langage de requêtes XQUERY ;
- le résultat de l'exécution de cette requête constitue le résultat de notre scénario d'interprétation. Ce résultat est présenté à l'interface via la même interface utilisateur.

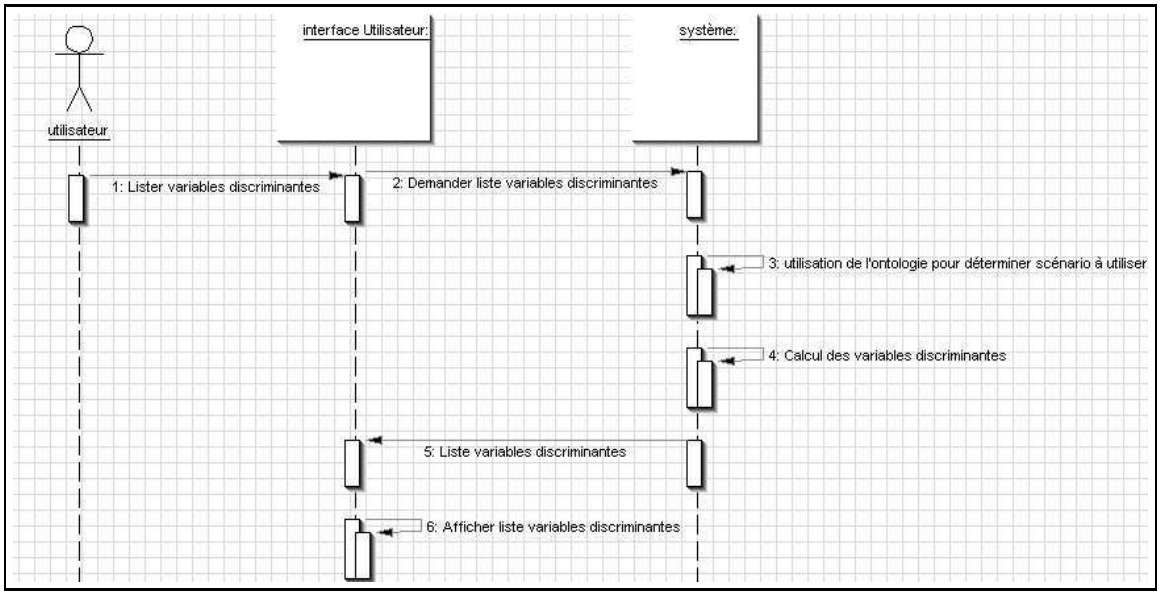

FIG. *6.13 - Diagramme de séquence d'un scénario d'interprétation*

6.9 Synthèse du Chapitre 6

A ce niveau, il convient de rappeler que pour que, les applications informatiques puissent être évolutives, adaptables, et compréhensibles par l'utilisateur final, nous avons besoin d'outils de modélisation basés sur des formalismes adéquats, simples à comprendre et faciles à utiliser. D'où la nécessité de faire appel à la métamodélisation. Ce chapitre a expliqué le processus de métamodélisation et cette métamodélisation nous a permis de nous abstraire de la complexité d'un domaine tel que l'interprétation des résultats des méthodes de classification.

Ainsi, ce chapitre a été consacré à la présentation de notre métamodèle. La première partie a été consacrée à la définition des concepts de modèle et de métamodèle. Ce chapitre a permis aussi de montrer les différences existant entre ces deux concepts. Ensuite, nous avons présenté quelques techniques de métamodélisation et nous avons expliqué celles qui nous ont été utiles dans notre travail.

La seconde partie de ce chapitre a été consacrée à la présentation de notre stratégie de modélisation d'une part et de la spécification de notre métamodèle d'autre part. Ainsi, nous expliquons les classes de notre métamodèle et les relations qui existent entre elles. Par la suite nous avons instancié notre métamodèle en définissant un modèle d'interprétation basé sur l'utilisation d'un des trois (3) axes d'interprétation (axe « *Variable* »).

Grâce à notre métamodèle MIRC, nous permettons aux utilisateurs d'interpréter, à travers leurs propres critères, les résultats de leur classification. Par ailleurs, nous avons défini trois axes d'interprétation correspondant à des scénarios d'interprétation communément appliqués par les experts du domaine à la suite d'un processus de classification. Dans les chapitres suivants, ces thèmes sont abordés de manière plus approfondie.

Chapitre 7

Architecture d'utilisation de Métadonnées pour l'Interprétation des résultats de Classification (AMIC)

Dans ce chapitre, nous expliquons l'architecture générale d'exploitation de notre métamodèle. Il s'agit de donner dans un premier temps un état de l'art succinct des différentes approches existantes en terme d'architecture logicielle. Ainsi, nous consacrons la première partie de ce chapitre à expliquer les différentes *architectures logicielles* couramment utilisées. Il s'agit des *architectures à couche*, des *architectures à base d'objets* et des *architectures hybrides*. Pour chacune d'entre elles, nous prenons un exemple concret.

La seconde partie de ce chapitre est consacrée à la spécification du modèle d'architecture que nous avons adopté. Ensuite, nous expliquons les technologies que nous avons utilisées dans la réalisation de cette architecture. Ainsi, nous abordons le langage de requêtes XQuery, le processeur Saxon, l'utilisation du standard *PMML* dans notre architecture, l'utilisation du langage de transformation *XSLT* ainsi que de l'API de traitement des fichiers XML *Jdom*.

7.1 Modèles d'architecture logicielle

Introduction

L'étape la plus délicate dans le processus de création d'une application est, sans conteste, celle relative à la définition et à la modélisation du problème à traiter. Ainsi, il existe de nombreuses architectures logicielles constituant des aides précieuses à l'élaboration d'une application. L'objectif de ces architectures est d'aider le concepteur à formuler son problème puis à le décomposer en différents objets et/ou modules.
Tous les modèles d'architecture logicielle proposent au concepteur d'applications interactives de séparer en différents composants le domaine d'application, l'interface proposée à l'utilisateur et le contrôle de l'application.

Notre objectif est de faire un bref état de l'art sur certains de ces modèles d'architecture logicielle. Ainsi, les différents modèles d'architecture logicielle peuvent être classés en trois grands groupes [Depaulis et al., 2006] :

- *Modèles linguistiques ou à couches* : ils décrivent la structure globale d'une application sous forme de couches logiques. Ils ont un niveau d'abstraction assez élevé, ce qui constitue un inconvénient puisqu'ils ne peuvent pas être utilisés efficacement de manière individuelle. Il s'agit par exemple du modèle ARCH [Bass et al., 1992] ;
- *Modèles à objets ou interacteurs* : très largement utilisés aujourd'hui, ces modèles s'inspirent du paradigme de la programmation objet. Ils proposent une décomposition modulaire de l'interface en un ensemble d'objets communicants. Il s'agit par exemple de *PAC* [Coutaz, 1987], de *MVC* [Krasner and Pope, 1988], etc.
- *Modèles hybrides* : ils ont été conçus afin de tirer le meilleur parti des modèles cités précédemment. Nous pouvons citer, entre autres, le modèle *PAC-Amodeus* [Nigay, 1994].

Bien que les terminologies employées diffèrent sensiblement, on retrouve souvent d'un modèle à l'autre, les mêmes principes de base : séparer le noyau fonctionnel (représentant les objets et les méthodes de l'application) de l'interface utilisateur (cf. figure 7.1).

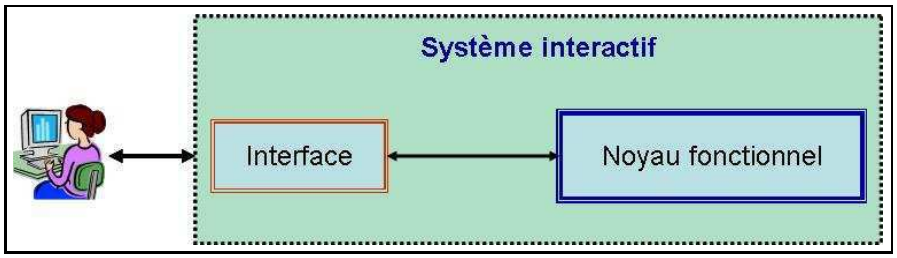

FIG. *7.1 - Système de base*

Tous les modèles abordés dans cette section permettent la décomposition d'une application interactive en différents éléments. Cette décomposition permet une conception plus simple car chaque module peut être réalisé de manière plus ou moins indépendante. Ce qui rend aisées les éventuelles modifications et accroît la fiabilité du système. Mal-

heureusement, cette décomposition n'est pas aisée à réaliser car l'interface se situe à la croisée des chemins entre l'application et l'utilisateur.

7.1.1 Les modèles à couche

Comme leur nom l'indique, ces modèles décomposent les interfaces en un petit nombre de couches. Chaque couche possède un rôle spécifique et traduit un niveau d'abstraction. Cette approche est encore utile pour comprendre le fonctionnement de l'interaction, mais elle est trop monolithique, abstraite et peu structurante. En outre, cette approche qui trouve son origine dans les interfaces textuelles est de plus en plus difficile à appliquer aux interfaces modernes. L'approche ARCH, que nous abordons dans cette section, est plus pragmatique car elle s'inspire des architectures concrètes des systèmes interactifs, avec les notions de plate-forme et de boîte à outils.

L'exemple de ARCH

Tous les modèles tentent de répondre à une préoccupation unique : la séparation de la partie interface du reste de l'application, afin de rendre cette dernière indépendante de tout matériel ou de tout outil spécifique à la présentation. Concrètement, le noyau fonctionnel ne doit avoir aucune connaissance des fonctionnalités relevant de l'interface utilisateur, ce qui permet de favoriser la réutilisation et la portabilité de l'application. C'est dans cette optique que les premiers modèles étaient des modèles à couche et l'un des précurseurs est le modèle de SEEHEIM [Green, 1985].

Dans cette section nous nous limitons au modèle ARCH [Bass et al., 1992] qui affine le modèle de SEEHEIM en s'inspirant d'avantage des boîtes à outils graphiques. Ce modèle incorpore la notion de plate-forme de développement, et décrit la nature des données qui transitent entre les différents composants. Il a été créé afin de pallier les insuffisances liées au manque de précision du contrôleur de dialogue dans SEEHEIM. Le modèle ARCH identifie cinq composants dans les systèmes interactifs (cf. figure 7.2) :

- *Le noyau fonctionnel* représente la partie non interactive de l'application. Il s'agit des objets et des méthodes propres à l'application. Ces deux concepts sont implémentés indépendemment des interactions avec l'utilisateur.
- *Le composant d'interaction* désigne l'ensemble des objets interactifs d'une boîte à outils ainsi que les communications avec les périphériques physiques. Comme nous pouvons le constater, cette composante dépend de la plate-forme sur laquelle elle est implémentée.
- *Le contrôleur de dialogue* est responsable du séquencement des tâches et du maintien de la cohérence entre les différentes vues d'un même objet. Il autorise et contrôle l'enchaînement des interactions de la couche présentation.
- *Le composant de présentation* permet de rendre le reste de l'application indépendante de toute plate-forme par une abstraction logique des objets et des méthodes

d'interaction.

- *L'adaptateur du domaine* implémente les tâches, relatives au domaine, dans lesquelles l'utilisateur intervient. Ces tâches comprennent la réorganisation des données du domaine dans des buts de manipulation interactive, ou la détection des erreurs sémantiques.

L'un des avantages de cette décomposition est qu'elle permet à une application de mieux supporter d'éventuelles modifications. Malheureusement on peut reprocher à ce modèle une description superficielle du fonctionnement de ses différentes couches.

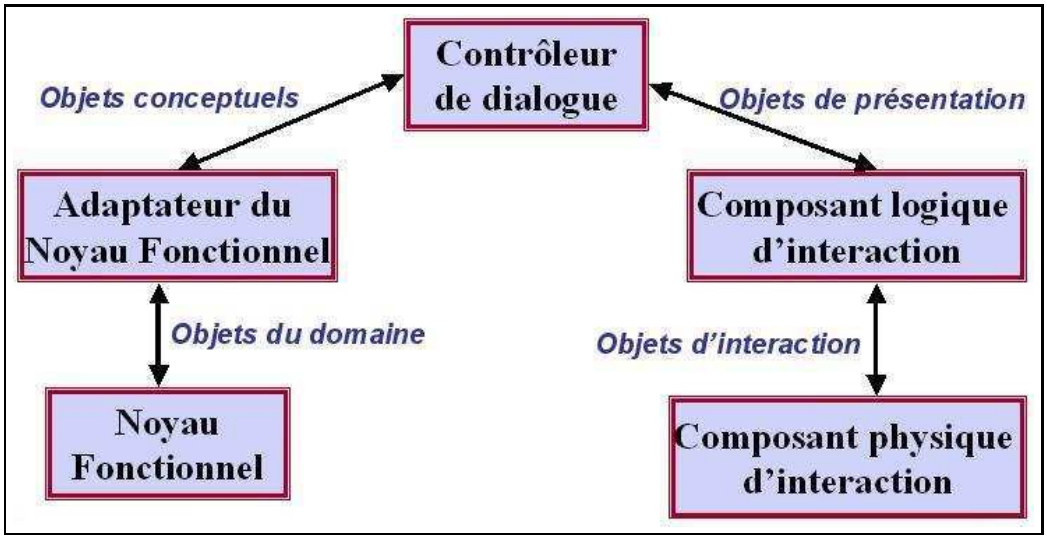

FIG. *7.2 - Architecture du modèle* ARCH

7.1.2 Les modèles à objets

L'essor des paradigmes liés à la Programmation Orientée Objet (POO) ont conduit à repenser les modèles d'architecture logicielle. Ainsi, au lieu de séparer les composants « application » et « interface », les modèles répartis (connus sous le nom de modèles à objets) utilisent les caractéristiques de la POO pour scinder l'application. Les modèles à objets décrivent un autre style de décomposition pour les interfaces utilisateur en modélisant le système de façon homogène. Cette approche permet de décrire des aspects tels que la modularité, le parallélisme et la distribution. Dans cette section nous étudions le modèle MVC [Krasner and Pope, 1988] qui est l'un des plus utilisés et que nous avons adopté dans cette thèse.

L'exemple de MVC

Le modèle MVC (Modèle, Vue, Contrôleur) [Krasner and Pope, 1988] a été introduit comme architecture de référence dans l'implémentation des interfaces utilisateur de l'environnement SMALLTALK. L'approche MVC est ainsi à la base d'un grand nombre de

modèles à base d'objets. Ainsi, à l'intérieur de ce modèle, l'interface de l'application est décomposée en un ensemble de trois modules indépendants et capables de communiquer entre eux par message (cf. figure 7.3).
Les composantes de cette approche sont :

- *Le modèle :* c'est le noyau fonctionnel. Cette composante comprend les structures de données et les méthodes de l'application. Elle est la seule qui puisse communiquer avec d'autres modèles.
- *La vue :* elle maintient une représentation du modèle perceptible par l'utilisateur, qu'elle met à jour à chaque changement d'état du modèle. C'est la représentation externe du modèle (celle qui est affichée à l'écran) et est en général constituée d'objets graphiques. La dissociation de la composante « modèle » et de la composante « vue » permet à un modèle de représenter plusieurs vues.
- *Le contrôleur :* c'est la composante qui reçoit et qui interprète les événements de l'utilisateur. Il s'agit de gérer les entrées de l'utilisateur transmises par l'intermédiaire de la vue et les communiquer au modèle.

Rappelons que chacune des composantes du modèle MVC est un objet à part entière. Au sens de la programmation orientée objet, une classe de Modèle peut être compatible avec plusieurs classes de Vues et de Contrôleurs. Par exemple, une métadonnée représentant un entier, comportant une valeur minimale et une valeur maximale, peut être représentée par une *jauge*, un *curseur*, un *champ de saisie*, etc.

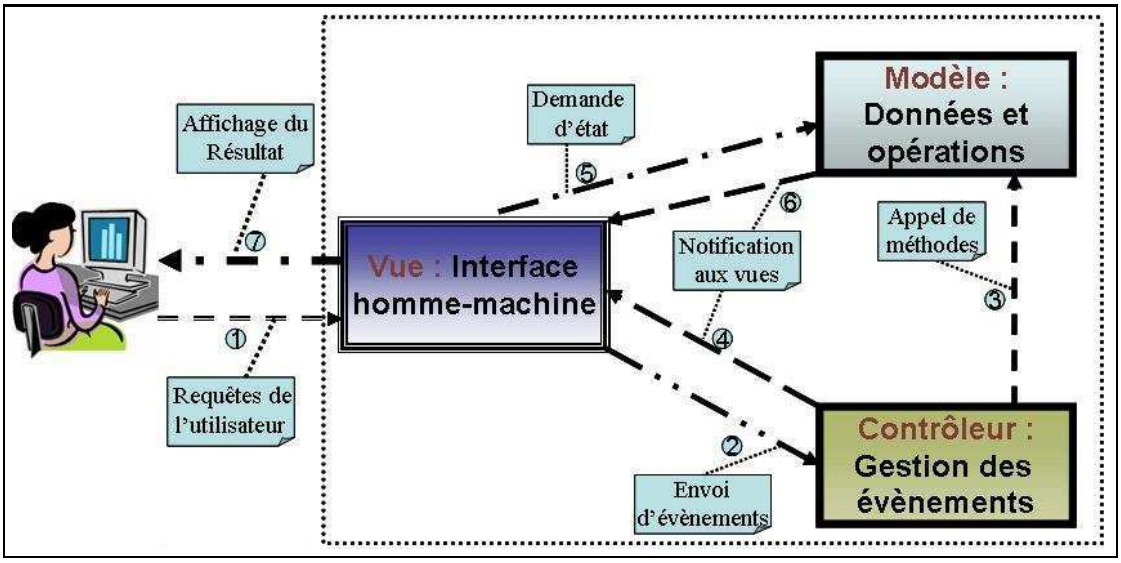

FIG. *7.3 - Architecture MVC (Model-View-Controller)*

Comparé aux autres modèles, MVC est le seul à dissocier la gestion des entrées de celle des sorties dans chacun de ses objets. Ce qui permet conceptuellement de décrire l'apparence ou le comportement en entrée d'un objet interactif de façon indépendante. Si cette séparation est très avantageuse du point de vue de la modifiabilité, elle est souvent

difficile à mettre en œuvre. En effet, dans la plupart des composants visuels, la manière dont les entrées sont interprétées est très liée à leur affichage. MVC ne décrivant pas de protocole de communication entre les deux composants, la plupart des boîtes à outils existantes préfèrent fusionner la *Vue* et le *Contrôleur* en une entité unique [Fowler, 1999]. Grâce à la séparation entre la partie sémantique et la partie présentation, cette architecture a suscité l'intérêt des concepteurs du langage *Java*. En effet, ce modèle d'architecture facilite la modifiabilité, la conception itérative et est compatible avec les langages à objets. Ce modèle d'architecture est par exemple à la base de la boîte à outils *Swing* du langage *Java*. Malheureusement, cette architecture présente aussi quelques inconvénients. Ainsi, son fonctionnement induit un échange très important de messages entre les différentes composantes et ce, même pour des actions très simples. Pour plus de détails sur cette architecture, se référer aux travaux réalisés par SCHMUKER dans [Schmuker, 1986].

7.1.3 Les modèles hybrides

Les modèles hybrides tentent de réconcilier les deux types de modèles décrits dans les sections 7.1.1 et 7.1.2 en reprenant les avantages de chacun. Dans cette section, nous évoquons l'exemple du modèle PAC-AMODEUS. Cependant il existe d'autres modèles hybrides tels que H^4 [Girard et al., 1995], [Guittet, 1995] ou multi-couche [Fekete, 1996].

L'exemple de PAC-AMODEUS

Le modèle PAC-AMODEUS [Nigay and Coutaz, 1993], [Nigay, 1994] est un modèle hybride puisqu'il repose sur une extension du modèle ARCH selon une approche objet. Ce modèle reprend les cinq niveaux fonctionnels du modèle ARCH et structure le Contrôleur de Dialogue avec une hiérarchie d'objets *PAC* [Coutaz, 1987]. L'adaptateur de domaine et la présentation de ARCH communiquent directement avec chaque objet *PAC* à travers son abstraction (pour le premier) et sa présentation (pour le second).
L'objectif de *PAC-Amodeus* est de combiner les avantages du modèle ARCH, qui intègre des aspects de génie logiciel comme la modifiabilité et la portabilité, et du modèle *PAC*, qui permet de structurer efficacement le contrôleur de dialogue jusque-là monolithique. Le modèle *PAC-Amodeus* a été employé pour modéliser des plate-formes multimodales. Les entrées sont décrites par des symboles et des grammaires, et des mécanismes de fusion de haut-niveau sont implémentés dans le dialogue par des objets *PAC*.

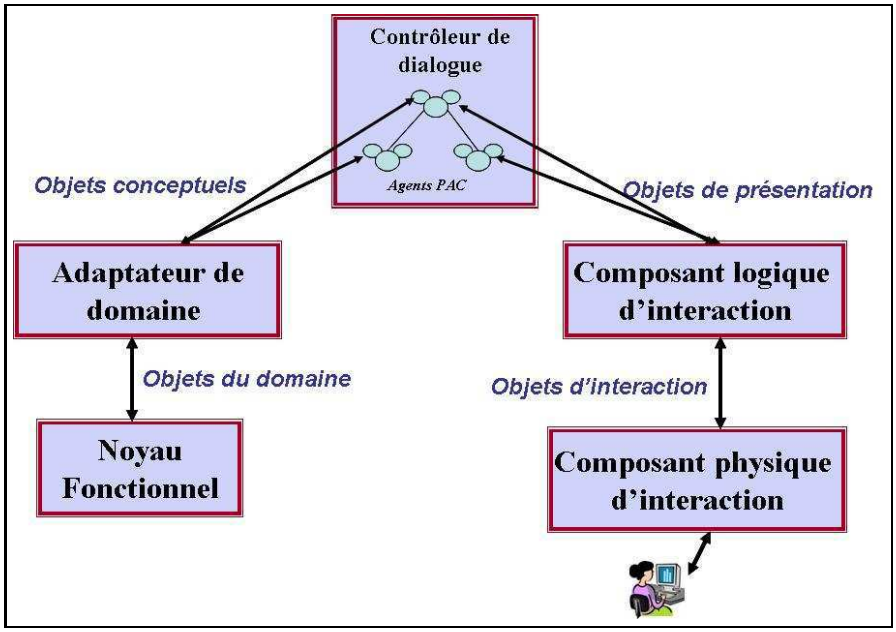

FIG. *7.4 - Architecture du modèle* PAC-AMODEUS

Synthèse de la section 7.1 et positionnement

Dans la première partie de ce chapitre, nous avons évoqué les différents modèles d'architecture logicielle les plus utilisés. Pour chacun d'entre eux, nous avons pris l'exemple d'un représentant de ces modèles pour mieux expliquer leur fonctionnement.
Puis, nous avons expliqué pour chacun de ces modèles les critiques qui leur sont faits ainsi que le choix que nous avons fait dans nos travaux.

Ainsi, pour les *modèles à couche*, on leur reproche de ne s'en tenir qu'à une description superficielle du fonctionnement de ses différentes couches. En effet, comme nous l'expliquons dans la section 7.1.1, ce type de modèle est trop abstrait et est peu structurant. Par ailleurs, ce type de modèle est difficile à mettre en œuvre de nos jours compte tenu de l'évolution des interfaces.

Pour les modèles à objets, la principale critique qui leur est faite se situe au niveau du nombre important de messages échangés entre les différentes composantes et ce, même pour des tâches simples. Heureusement que ces modèles s'appuient sur la notion de décomposition des applications informatiques en des *objets*, ce qui constitue un net avantage.

Enfin, les modèles hybrides ont pour objectif de prendre les meilleurs aspects de ces deux modèles pour en construire un nouveau.

Dans le cadre de cette thèse, nous avons fait le choix d'utiliser un exemple de modèle à base d'objets. Il s'agit du *Modèle-Vue-Contrôleur* qui offre l'avantage d'être compatible avec les langages à objet utilisés dans la réalisation des modules de cette thèse. Ce type de modélisation offre l'avantage de réaliser une modélisation flexible, robuste et facilement évolutive car le modèle de données, les contrôleurs et les vues sont indépendantes.

7.2 Spécification de notre modèle d'architecture

Nous avons adopté la spécification du modèle MVC dans la réalisation de notre application. La figure 7.5 montre l'architecture globale d'exploitation de notre métamodèle dédié à l'interprétation des résultats des méthodes de classification. A l'image du modèle MVC, cette architecture est composée de trois parties : le *noyau de l'application*, l'*interface utilisateur* et le *contrôleur d'évènement*.

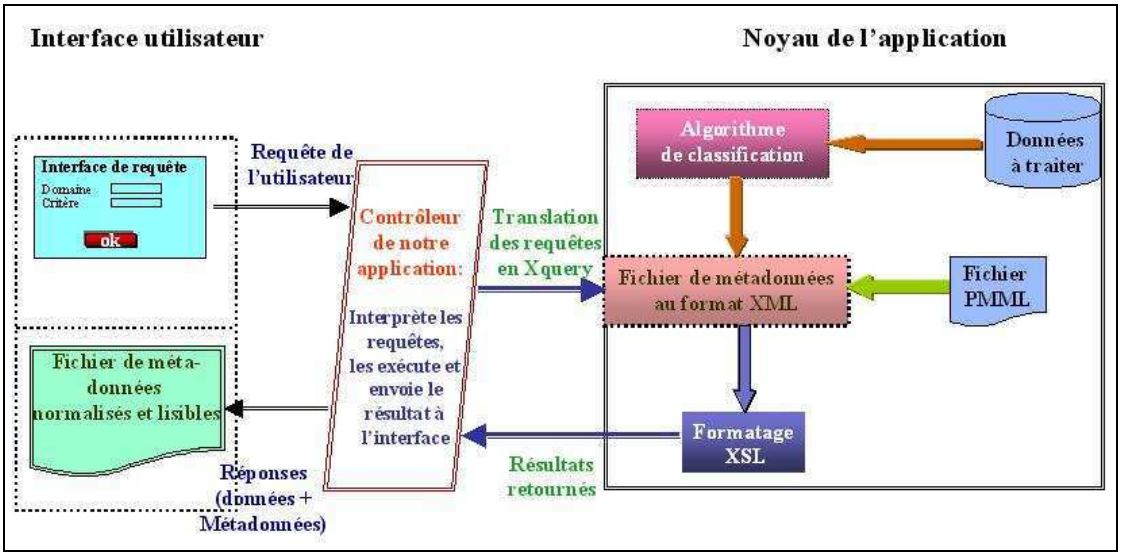

FIG. *7.5 - Architecture globale d'aide à l'interprétation des résultats de classification*

Explication de notre architecture

Dans ce paragraphe, nous évoquons le fonctionnement général de notre application à travers cette architecture. Ainsi, nous présentons le *noyau fonctionnel*, le *contrôleur* et l'*interface utilisateur* (expliquée en détail à la section 9.3).

- ***Le noyau fonctionnel (Modèle) :*** Il contient le cœur de l'application (les données et les traitements qui s'y réfèrent) et ce, indépendamment de la plate-forme utilisée. A ce niveau nous avons développé une bibliothèque de traitements de notre métamodèle d'interprétation. Cette bibliothèque permet le traitement des (méta)données présentées sous format XML. Ainsi, pour une méthode de classification donnée, pour en extraire les métadonnées nécessaires à l'interprétation de ses résultats, nous avons besoin de connaître la structure globale de l'algorithme. En effet, sans la connaissance de cette structure, il est difficile d'extraire la valeur des variables qui peuvent être déterminantes dans le processus d'interprétation.

  Notre application est aussi capable de traiter des résultats de classification exprimés en langage PMML (ce concept est abordé dans la section 3.1.5). Ceci est un avantage car PMML est devenu un standard incontournable dans la description

des résultats de méthodes de fouille de données. Tous les traitements réalisés dans cette composante ont pour but de générer un fichier de métadonnées contenant la description détaillée des résultats de la méthode de classification utilisée. Ce supplément d'information va servir à l'interprétation des résultats de la méthode et ce, à travers l'utilisation des requêtes XQuery (cf. § 7.4).

- ***Le contrôleur :*** Cette composante permet, d'une part, le traitement des informations envoyées par l'utilisateur à travers l'interface. D'autre part, il envoie à cette même interface les résultats obtenus par le noyau fonctionnel. Dans notre architecture, cette composante est basée sur le processeur Saxon (cf. § 7.4). Ce qui permet de traduire les requêtes utilisateur et/ou les scénarios d'interprétation prédéfinis. C'est aussi dans cette composante qu'on retrouve la partie de notre application qui permet de transformer en utilisant le langage XSL (cf. § 7.5) les résultats, envoyés par le noyau, en des documents plus facilement compréhensibles et conviviaux à lire.
- ***L'interface utilisateur (Vue) :*** C'est une représentation visuelle des données de l'application. Cette composante écrite avec la bibliothèque SWT[1] (sous eclipse), permet aux utilisateurs d'avoir une interface conviviale et facile à utiliser. Évidemment cette interface est fortement dépendante du système d'exploitation. Dans le modèle MVC, à chaque fois qu'il y a un changement dans le modèle, ce dernier doit avertir les vues qui dépendent de lui. Ceci a des avantages certains :
  - la possibilité d'attacher plusieurs vues distinctes à un même modèle pour obtenir plusieurs présentations ;
  - la possibilité de changer l'implémentation d'un modèle (optimisation, réorganisation, etc.) sans rien changer aux vues et aux contrôleurs.

Nous abordons dans l'annexe A l'ensemble des *patrons de conception* (**design patterns**) que nous avons utilisés pour modéliser les composantes de notre modèle d'interprétation.

Place des métadonnées dans notre architecture

Rappelons que l'utilisation de cette architecture devra permettre de répondre aux attentes liées :

- à ***la recherche des données*** ;
- à ***l'analyse*** et à ***la validation des résultats*** ;
- et à ***l'interprétation des résultats***.

C'est ce dernier point qui nous intéresse plus particulièrement dans le cadre de cette thèse. A travers ces trois activités nous avons considéré trois types de métadonnées.

- ***Les métadonnées d'accès :*** il s'agit des métadonnées permettant de localiser les données ou de les rechercher suivant les decriptions générales. Elles sont consti-

[1] Standard Widget Toolkit : bibliothèque graphique libre pour le langage Java

tuées principalement des éléments du Dublin Core [Core, 2006], éventuellement des éléments relatifs au contexte de la collecte des données [Papageorgiou, 2003] dans le cas de données provenant d'enquêtes statistiques par exemple. Ainsi, pour ces derniers, il est recommandé d'avoir les éléments suivants :

- le *titre/contenu* de la description incluant souvent les éléments suivants : la population statistique, la couverture géographique, l'unité d'observation, les classifications et les standards appliqués ;
- *les désignations des lignes/colonnes* dans les tables et les éléments de graphe ;
- les définitions des labels ;
- *les unités de mesure* ;
- *la référence temps/période* ;
- *la comparabilité* sur le temps (coupures en séries, données oubliées) ;
- *la source des données* ;
- toute information sur le *copyright*, les restrictions d'usage ;
- *les contacts* pour des informations complémentaires ;
- la *description des méthodes* utilisées dans la collecte, la révision, le *calcul* et l'estimation des statistiques ;
- l'information sur les *sources des erreurs* ;
- la *description de l'objectif des statistiques*, concepts, variables et standards utilisés.

Comme nous pouvons le constater, il s'agit principalement de métadonnées qui nous renseignent sur les processus de collecte, d'agrégation et/ou de classification des données.

- ***Les métadonnées de validation :*** ces éléments de métadonnées fournissent des informations relatives aux différents critères de validation utilisés par les méthodes de classification. Ces métadonnées permettent de rechercher ou de déterminer les classes valides.
- ***Les métadonnées interprétatives :*** il s'agit des métadonnées qui vont nous aider à interpréter les classes obtenues. Elles sont constituées par des informations liées, par exemple, au contexte des données classifiées et aux critères d'interprétation utilisés par les différentes méthodes de classification.

Il existe aussi d'autres auteurs qui se sont attelés à ce travail de classification de métadonnées, parmi eux citons les travaux de LAPLANT dans [LaPlant et al., 1996].

7.3 Utilisation des standards dans notre architecture

Dans la seconde partie de ce chapitre, nous expliquons les différentes technologies utilisés dans la réalisation de notre architecture. Ainsi, nous abordons le langage *PMML*, le processeur SAXON, le langage de requêtes XQUERY, l'API JDOM, le langage de transformation *XSLT*. Pour chacune de ces technologies, nous expliquons en quoi elles ont été

utiles dans la réalisation de notre architecture. Ainsi, la prise en charge du langage *PMML* dans notre outil, a permis d'avoir un modèle qui s'adapte aux différents modules de fouille de données produisant des résultats sous ce format. Grâce au processeur SAXON, nous sommes en mesure de traiter efficacement les requêtes XQUERY. Et l'utilisation de ce langage de requête a permis incontestablement de faciliter l'implémentation des scénarios d'interprétation élaborés par les experts du domaine de la classification. Pour pouvoir *parser* les documents au format XML, l'utilisation de JDOM a permis d'avoir les avantages de DOM tout en évitant la lourdeur liée à ce type d'analyseur syntaxique (parser). En effet JDOM est un parser dédié exclusivement au langage et conçu pour faciliter le traitement de documents XML avec *Java*. Enfin, le langage de transformation *XSLT* a permis de transformer les documents au format XML.

7.3.1 Utilisation de PMML

Dans ce paragraphe, nous montrons un *exemple tronqué* d'un fichier PMML tiré de [DMG, 2006] et qui décrit des données traitées avec la classification basée sur les prototypes (*clustering center-based*). Le but est d'expliquer la différence entre l'approche PMML et notre approche. En effet, nous utilisons les documents PMML comme données d'entrée pour la création de certaines métadonnées de notre fichier de métadonnées. Pour pouvoir utiliser ces fichiers PMML de manière efficiente, nous avons besoin de connaître les informations pertinentes dans un fichier PMML dans le cadre de l'interprétation. Comme nous l'expliquons au chapitre 6, l'extraction des contenus de ces balises se fait grâce au parser que nous avons implémenté.

```
<?xml version="1.0" encoding="windows-1252" ?>
<PMML version="2.0">
 <Header copyright="Copyright IBM Corp. 2002, All Rights Reserved">
  <Application name="IBM DB2 Intelligent Miner" version="6.2"/>
 </Header>
 <DataDictionary numberOfFields="5">
  <DataField name="PETALLEN" displayName="PETALLEN" optype="continuous"/>
  <DataField name="PETALWID" displayName="PETALWID" optype="continuous"/>
  <DataField name="SEPALLEN" displayName="SEPALLEN" optype="continuous"/>
  <DataField name="SEPALWID" displayName="SEPALWID" optype="continuous"/>
  <DataField name="SPECIES" displayName="SPECIES" optype="categorical">
   <Value value="setosa"/>
   <Value value="versicolor"/>
   <Value value="virginica"/>
  </DataField>
 </DataDictionary>
 <ClusteringModel modelName="neuclu" modelClass="centerBased" functionName="clustering"
  algorithmName="neuralNetwork" numberOfClusters="6" x-quality="0.01154717854016473">
  <MiningSchema>
   <MiningField name="PETALLEN"/>
    ..............................
```

```
  </MiningSchema>
    ..............................
  <ComparisonMeasure kind="distance">
   <squaredEuclidean/>
  </ComparisonMeasure>
  <ClusteringField field="PETALLEN" compareFunction="absDiff"/>
    ..............................
  <Cluster name="0">
   <Extension name="Cluster">
    <Cluster>
     <Partition name="0" size="50">
      <PartitionFieldStats field="PETALLEN">
       <Counts totalFreq="50" invalidFreq="0" missingFreq="0"/>
      </PartitionFieldStats>
       ..........................
     </Partition>
    </Cluster>
   </Extension>
    ..............................
 </ClusteringModel>
</PMML>
```

7.4 Le langage de requêtes XQuery

XQuery est un langage déclaratif puissant pour extraire des données XML. En effet, il permet de lire des fragments XML, d'y effectuer des recherches et de générer de nouveaux fragments XML. Ainsi, une requête XQuery est une composition d'expressions évaluées dans un certain contexte. Chaque expression a une *valeur* (qui est une séquence ordonnée de 0 ou de plusieurs items). Un *item* est un nœud de l'arbre d'un document ou une *valeur atomique*. Une *valeur atomique* est une instance ayant un type atomique XML Schema[2] (par exemple : nombres, chaînes, dates, etc.). Un nœud pouvant être de type *Document*, *Element*, *attribut*, etc. Les expressions n'ont pas d'effets de bord (par exemple, pas de mise-à-jour). Avec ce langage, il est possible de faire des opérations arithmétiques ($1+2$, $\$X$ modulo 5, etc.), des opérations telles que la concaténation, l'union, l'intersection ainsi que des opérations booléennes.
Pour récapituler, il convient de savoir qu'une requête XQuery est formée d'un prologue composé d'une suite de déclarations et de définitions et d'un corps composé d'une expression dont la valeur est le résultat de la requête. Voici un exemple de requête XQuery qui calcule la somme par ligne d'une matrice de données représentée dans un fichier XML :

```
<Resultat>
{
for $a in doc("toto.xml")//variable/ligneMatrice
return
<somme>{sum(for $i in $a/valeurMatrice
return $i
)}
```

[2]Standard qui fait de XML un langage de définition de données métier. XML Schema est un standard permettant de définir la structure d'un document XML

```
</somme>
}
</Resultat>
```

A travers cet exemple, nous voyons que ce langage est fortement typé car toute expression a un type construit à partir du système de types de XML Schema. Pour utiliser ce langage, nous avons besoin d'un processeur XQuery. Ce dernier peut être un outil « standalone » ou être intégré dans un « service Web » ou encore dans une base de données XML ou enfin une librairie qu'on intègre dans un logiciel. Dans ce travail, nous avons choisi le processeur Saxon. Saxon est une collection d'outils pour le traitement de documents XML. Une description de ses principales composantes est décrite dans [Saxon, 2006], il s'agit d'un processeur XSLT, d'un processeur XPath (version 2.0), d'un processeur XQuery (version 1.0), d'un processeur XML Schema (utilisé pour la validation d'un schéma). Et pour permettre à une application utilisant cette technologie d'utiliser ces différents processeurs sans changer de code, Saxon utilise l'API JAXP. Pour ce qui est de l'utilisation de ce processeur dans notre application, il faut savoir que Saxon offre deux API pour le traitement XQuery (sa propre API native et l'API XQJ [XQJ, 2006]). Par ailleurs, il convient de rappeler à ce stade que nous utilisons Saxon à travers une API Java. En effet, l'utilisation des requêtes XQuery doit être transparente aux utilisateurs.

Par ailleurs rappelons que ce langage peut traiter un document ou une collection de documents sous forme de fichiers, de bases de données XML ou d'arbres DOM [3] (abordé dans la section 7.6.2). En se basant sur les expressions XPath 2.0[4] (XPath est un langage avec une syntaxe non XML, permettant de parcourir un document XML selon des axes (fils, parent, etc.)), ce langage permet de faire des requêtes selon la structure des documents ou selon leurs contenus. Pour utiliser ce langage, comme pour le langage SQL, nous pouvons utiliser la ligne de commande ou une interface. Dans ce travail, nous avons utilisé une interface. Pour conclure, nous pouvons dire que le langage de requêtes XQuery est une expression qui lit une séquence de fragments XML ou de valeurs atomiques et qui retourne une séquence de fragments XML ou de valeurs atomiques. De plus, les formes principales que peuvent prendre une expression XQuery sont :

- des expressions de chemins ;
- des constructeurs ;
- des expressions FLWOR[5] ;
- des expressions de listes ;
- des conditions ;
- des expressions quantifiées ;

[3] Document Object Model

[4] Recommandation candidate du W3C, ce langage est intensément utilisé par d'autres langages et technologies XML tels que XSLT

[5] For, Let, Where, Order By et Return

- des expressions de types de données ;
- des fonctions.

Par ailleurs, rappelons que XQUERY a déjà fait l'objet de nombreuses implémentations. Ainsi, de grands groupes comme IBM, Microsoft, Oracle ou encore Apple ont été parmi les premiers à implémenter les premières versions de XQUERY. De plus, et comme le montre le processeur que nous avons utilisé, le monde du libre n'est pas en reste. Toutes choses qui confortent le sentiment que ce langage est promu à un bel avenir. En effet, ce langage confère au langage XML la faculté de recherche et de sélection des bases de données. Avec ce langage, nous pouvons exécuter des opérations de création, de suppression ou de jointure mais pas de mise à jour pour l'instant.

7.5 Le langage de transformation XSLT

XSLT (eXtensible Stylesheet Language Transformations) [Kay, 2001], [Tidwell, 2001], [Fitzgerald, 2001] est un langage déclaratif de transformation de documents *XML* dans d'autres formats de représentation. Par exemple, ce langage peut être (et est) utilisé pour transformer des documents XML en un fichier HTML ou WML. Il peut aussi être utilisé pour créer des documents prêts à l'impression et ce grâce à XSL-FO.

Le principe de fonctionnement est simple dans sa description : il s'agit de traiter des documents XML par des règles XSLT et de produire un assemblage des fragments produits [Rigaux and Amann, 2002]. En effet, le principe de base du fonctionnement d'un environnement XML/XSLT est la séparation entre le processus de traitement des données et celui de leur publication. Dans nos travaux, les fichiers XML obtenus le sont au moment de la création des métadonnées. A ce fichier nous appliquons des règles de transformation XSLT pour obtenir des documents au format HTML plus faciles à lire et à comprendre pour les utilisateurs. Dans le § 7.5.1, nous commençons par donner quelques notions de base de ce langage de transformation. Dans les § 7.5.2 et § 7.5.3, nous expliquons quelques avantages à utiliser ce langage et nous donnons un exemple concret d'utilisation.

7.5.1 Notions de base

Commençons par rappeler qu'un document XSLT n'est autre qu'un document XML car c'est un arbre constitué d'une imbrication de balises ouvrantes et fermantes. Néanmoins, il utilise un espace de nommage particulier, celui constituant à faire précéder toutes les balises de ce langage par le mot clé *xls*. Ceci, afin que le processeur puisse les distinguer des balises à insérer dans les documents de sortie.

Un autre aspect fondamental en XSLT est celui relatif à la notion de règles (appelées *template*) qui est centrale dans ce langage. Pour comprendre cette notion, nous reprenons les aspects avancés dans [Rigaux and Amann, 2002] :

- une règle s'applique toujours dans le contexte de l'un des nœuds du document source ;

- l'application de la règle consiste à produire un fragment de document qui peut combiner du texte et des données extraites du document source.

Concrètement, une règle est un élément XML défini par la balise *xsl : template* dont le contenu est le corps de cette règle et définit le texte produit à chaque fois que la règle s'applique. D'une manière générale, un programme XSLT est constitué de plusieurs règles dont chacune peut s'appliquer à différentes parties d'un document XML. Chaque règle produit un sous-arbre spécifique quand elle rencontre un élément déclencheur et l'assemblage de ces sous-arbres constitue le résultat de la transformation.

7.5.2 Les avantages de XSLT

L'un des avantages de ce langage est qu'il est accessible par un utilisateur n'ayant pas des bases de programmation car c'est un langage déclaratif et non procédural. Comme nous l'expliquons dans la section précédente, ce langage est utilisé pour obtenir des documents aux formats divers et variés. Ainsi, grâce aux transformations XSLT, nous pouvons obtenir plusieurs versions différentes d'un même document XML de base :

- un format destiné à l'affichage pour un site web ;
- un format destiné à l'affichage pour site WAP permettant de consulter sur un téléphone mobile les mêmes informations que celles affichées sur le site Web ;
- un document PDF, grâce à sa composante XSL-FO, contenant les mêmes informations et prêt à être imprimé.

Il existe bien d'autres avantages de ce langage relatifs à sa puissance de traitement et à sa facilité de prise en main.

7.5.3 Exemple

A présent, prenons un exemple de document XML dont on souhaite afficher le contenu au format HTML. Ce document décrit un ensemble de livres. Ce fichier XML est structuré par les éléments *Titre, Auteur, Quantité et Description* qui décrivent un livre quelconque.

```
1  <?xml version="1.0" encoding="UTF-8"?>
   <?xml-stylesheet type="text/xsl" href="book.xsl"?>
   <BOOKS>
     <BOOK>
       <TITLE>UML 2.0</TITLE>
       <AUTHOR>James Rumbaugh, Ivar Jacobson, Grady Booch</AUTHOR>
       <QUANTITY>5</QUANTITY>
       <Description>
        Ce livre decrit clairement et completement les concepts UML.
       </Description>
     </BOOK>
     <BOOK>
       <TITLE>Data mining</TITLE>
       <AUTHOR> Pieter Adriaans, Dolf Zantinge</AUTHOR>
       <QUANTITY>12</QUANTITY>
```

```
      <Description>
       This is the first book to offer a comprehensive introduction to data mining.
      </Description>
    </BOOK>
</BOOKS>
```

Notre objectif est d'obtenir une représentation HTML de ces informations. Pour ce faire, nous faisons appel au langage de transformation XSLT pour montrer qu'à partir de ce document, nous pouvons par exemple avoir des sorties destinées à plusieurs supports. Dans notre exemple, nous nous limitons au format HTML. Voici le fichier de transformation permettant d'obtenir le document HTML que nous voulons.

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <!-- by Abdourahamane Baldé (inria) -->
3  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4     <xsl:output method="html" encoding="iso-8859-1"/>
5       <xsl:template match="BOOKS">
6          <HTML>
7            <body>
8              <h2 align="center" >
9                  <font color="blue" size="5">LISTE DES LIVRES</font>
10                 </h2>
11                 <table border="5" cellspacing="2" cellpadding="8"
12                    bgcolor="#ccffaa" align="center">
13                   <thead align="center" bgcolor="silver">
14                     <th>TITRE</th>
15                     <th>AUTEURS</th>
16                     <th>QUANTITE</th>
17                     <th>DESCRIPTION</th>
18                   </thead>
19                   <xsl:for-each select="BOOK">
20                      <tr align="center">
21                        <xsl:for-each select="child::*">
22                           <td align="center">
23                               <font color="blue" size="3">
24                                 <xsl:apply-templates select="self::node()"/>
25                               </font>
26                           </td>
27                        </xsl:for-each>
28                      </tr>
29                   </xsl:for-each>
30                 </table>
31           </body>
32         </HTML>
33      </xsl:template>
34 </xsl:stylesheet>
```

L'application d'un programme XSLT consiste à parcourir les nœuds d'une arborescence XML, et à appliquer des règles de transformation à ces nœuds. Ainsi, la ligne 4 explique le format de sortie que nous voulons (HTML dans cet exemple). C'est ce qui explique

l'utilisation des balises HTML dont le rôle est de donner des directives de mise en forme au navigateur qui se charge d'afficher le document. La ligne suivante indique le nœud sur lequel doivent s'appliquer les transformations. Les lignes suivantes sont des balises de mise en page HTML. Ensuite pour chaque nœud « BOOK », on sélectionne chacun de ses nœuds fils et pour chacun de ces nœuds fils on affiche son contenu (lignes 19 à 29). Avec cette transformation, nous obtenons la figure 7.6.

LISTE DES LIVRES

TITRE	AUTEURS	QUANTITE	DESCRIPTION
UML 2.0	James Rumbaugh, Ivar Jacobson, Grady Booch	5	Ce livre decrit clairement et completement les concepts UML.
Data mining	Pieter Adriaans, Dolf Zantinge	12	This is the first book to offer a comprehensive introduction to data mining.

FIG. *7.6 - Sortie du document* book.xml *après application de la transformation XSLT*

Le programme XSL présenté ci-dessus est un document XML interprété par un processeur XSLT (dans cet exemple nous avons utilisé le processeur XSLT de *Mozilla Firefox*). En effet, nous indiquons dès la ligne 2 de notre fichier XML le fichier de transformation XSLT à utiliser. Ceci permet de réaliser des transformations dynamiques. Enfin, rappelons que nous pouvons utiliser ce fichier de transformation à tout document XML ayant la même structure que le fichier *book.xml*. C'est ce que nous faisons dans le cadre de cette thèse. En effet, notre fichier de transformation se base sur la structure de notre méta-modèle pour définir l'ensemble des transformations à effectuer sur les documents XML, résultats de notre approche.

7.6 Outils pour le traitement des flux XML

Pour le traitement des documents XML, il existe plusieurs API sur le marché, chacune avec ses qualités et défauts. Dans cette section, nous prenons l'exemple de trois (3) API utilisées dans le traitement des flux XML. Il s'agit de *SAX, DOM et JDOM*. Pour les deux premières (*SAX* et *DOM*), elles sont indépendantes des langages et des implémentations existent pour la majorité des langages de programmation. Quant à *JDOM*, elle est dédiée exclusivement au langage de programmation *Java* et nous expliquons en détail tous ces aspects dans les paragraphes suivants.

7.6.1 L'API SAX (Simple API for XML)

Cette API est un analyseur événementiel c'est-à-dire que son traitement se fait au fil du flux entrant. En XML, un évènement peut être une balise ouvrante, une balise fermante, l'ouverture ou la fermeture d'un document XML, etc. Ce type d'analyseur utilise des événements pour piloter le traitement d'un fichier XML. Il a été développé par un groupe indépendant pour fournir une alternative (allégée) à DOM. SAX fournit un

modèle événementiel pour l'analyse d'un document XML. Le principe de fonctionnement est le suivant : pendant que l'analyseur lit le document XML, il envoie des informations au programme. Par exemple, à chaque fois que l'analyseur rencontre une balise de début ou de fin, il en informe le programme. L'avantage de SAX est que de très gros documents peuvent être traités sans nécessiter de très grandes quantités de mémoire. L'inconvénient est que la lecture est séquentielle.

En utilisant cette API, nous n'avons pas besoin de garder en mémoire une grande quantité d'information. En effet, SAX possède les avantages de permettre une lecture linéaire, elle est peu coûteuse et simple. Néanmoins, utiliser ce type de parser basé sur les événements peut se révéler compliqué parce que les applications doivent être préparées à ignorer les résultats au risque de rencontrer une erreur fatale. De même, ne application peut être dépendante d'une information située vers la fin du document, ce qui complique l'utilisation de certaines requêtes de type Xquery. Par ailleurs, SAX ne permet pas de modifier (ajout, modification, suppression) un document XML. Nous ne pouvons pas effectuer d'accès direct à un élément (accès séquentiel oblige). Pour toutes ces raisons, SAX ne convient pas du tout à notre travail car selon le scénario d'utilisation, nous avons besoin d'un accès direct à des balises précises.

7.6.2 L'API DOM (Document Object Model)

DOM transforme l'arborescence XML en arborescence du langage cible. Concrètement, DOM reproduit l'arborescence du document XML en mémoire. Avec DOM, les programmeurs peuvent construire des documents, naviguer dans leur structure, ajouter, modifier ou supprimer soit des éléments soit du contenu. Son inconvénient majeur réside dans la lourdeur du traitement. En effet, tant que l'intégralité du document n'a pas été analysée et ajoutée dans la structure arborescente, les données du fichier d'entrée ne se trouvent pas dans un état utilisable. Ce qui peut être handicapant lorsque l'on traite un document très volumineux. Mais une fois la structure créée en mémoire, il est possible d'accéder aléatoirement à différentes parties du document. Pour ce faire, DOM fournit quelques méthodes pour modifier la structure du document. Cet accès aléatoire autorisé en fait un bon outil pour les applications d'édition ou de navigation de documents. Ce qui aurait pu être notre cas mais malheureusement la lourdeur de son traitement nous a dissuadé de l'utiliser.

Toutefois, il convient de rappeler que cette API est une spécification du W3C qui entend proposer une API permettant de modéliser, de parcourir et de manipuler un document XML. Son principal rôle est de fournir une représentation en mémoire d'un document XML sous la forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour). Les autres aspects à connaître sur cette API est qu'elle est définie pour être indépendante du langage dans lequel elle sera implémentée. Ainsi, DOM n'est spécifique à aucun langage de programmation.

7.6.3 JDom : API de traitement de fichiers XML dédiée au langage Java

Comme nous l'expliquons dans les sections précédentes, la mise en œuvre de DOM est rigide et gourmand en mémoire d'une part ; SAX est une API événementielle qui ne permet pas d'accès direct au document d'autre part. C'est ce qui explique la création d'une API alternative, plus simple et uniquement dédiée à la programmation java. Il s'agit de l'API Jdom. JDOM est une méthode pour représenter un document XML pour facilement le lire, le manipuler et l'écrire.

Cette API propose les fontionnalités suivantes :

- la lecture de fichiers XML à partir de fichiers, arbres DOM, flux SAX ;
- la création de documents XML en permettant de naviguer dans leur structure, de modifier, d'ajouter ou de supprimer leur contenu ;
- l'exportation d'arbre XMLJDOM sous la forme de fichier, arbre DOM, flux SAX ;
- la transformation XSLT ;
- la construction ;
- etc.

JDOM présente de grandes similitudes avec DOM en ce sens qu'il représente un document XML via une structure arborescente. Cependant, il s'en distingue parce que JDOM est spécifiquement conçu pour JAVA et que du point de vue du développeur JAVA, il s'avère beaucoup plus pratique à utiliser. JDOM est aussi liée à SAX car il utilise des collections SAX pour analyser les fichiers XML.

Parmi les avantages liés à l'utilisation de cette API, nous pouvons citer :

- la non nécessité de représenter le document XML entièrement en mémoire ;
- la complexité de XML est transparente à l'utilisateur ;
- la possibilité de convertir un document lu en un arbre DOM ou de générer des événements SAX ;
- la création des éléments de manière indépendante avant de les grouper pour créer un document JDOM ;
- la capacité à prendre en charge les espaces de nom ;
- l'ajout de DTD aux documents créés ;
- la possibilité de générer le document final indenté ou non ;
- etc.

Ce sont toutes ces raisons qui nous ont conduit à utiliser cette API pour le traitement de nos documents XML.

Exemple Voici un exemple simple d'utilisation de cette API pour la création d'un document XML :

```
package base;
import org.jdom.*;
import org.jdom.output.XMLOutputter;
import org.jdom.output.Format;

public class Essai
{
   public static void main(String[] args)
   {
      Element racine = new Element("Personne");
      Document document = new Document(racine);
      Element nom = new Element("Nom");
      nom.setText("Baldé");
      Element prenom = new Element("Prenom");
      prenom.setText("Abdourahamane");
      racine.addContent(nom);
      racine.addContent(prenom);
try
{
XMLOutputter affiche = new XMLOutputter(Format.getPrettyFormat());
affiche.output(document, System.out);
}
catch (java.io.IOException e){}
     }
}
```

On commence par importer le paquet *jdom*. Ensuite, nous instantions un objet de type *Element* à l'aide du constructeur *Element(String)* (qui prend le nom de l'élément en paramètre). Après avoir créé la racine de notre document (*Personne*), un objet *Document* est instancié avec l'élément *Personne* passé comme argument. Ensuite, il suffit de rattacher à l'élément *Personne* d'autres éléments le définissant. Ainsi, nous définissons deux nouveaux éléments *Nom* et *Prenom* auxquels nous affectons des valeurs à travers la méthode *setText()*. Il ne reste alors plus qu'à visualiser le document final sur l'écran ou de l'envoyer vers un fichier. C'est la classe *XMLOutputter* qui permet de faire ce type de traitement.

A travers cet exemple, nous voyons la simplicité de la mise en place de cette API par rapport à DOM. En effet la création d'un nœud ne nécessite pas le recours à l'objet *Document*. Nous voyons ainsi que la création d'un fichier XML est très simple. Il suffit de construire chaque élément puis de les ajouter les uns aux autres de manière logique. Pour faire des transformations, il suffit d'utiliser par exemple l'API JAXP (Java API for XML Processing)[6] qui facilite les transformations XSLT (cf. section 7.5) sur un document JDOM. Voici le résultat de cette création :

```
<?xml version="1.0" encoding="UTF-8"?>
<Personne>
      <Nom>Baldé</Nom>
      <Prenom>Abdourahamane</Prenom>
</Personne>
```

[6]Cette API permet d'analyser et de transformer des documents XML en implémentant SAX, DOM et XSLT.

7.7 Synthèse du Chapitre 7

Dans ce chapitre, nous avons commencé par expliquer les différents modèles d'architecture logicielle existants (cf. section 7.1). Nous avons expliqué les raisons pour lesquelles le modèle *MVC (Modèle-Vue-Contrôleur)* correspondait mieux à nos attentes dans ce travail. Ensuite, nous avons présenté une spécification de notre modèle d'architecture. A travers cette spécification, nous avons tenu à expliquer la place des métadonnées dans notre approche ainsi que l'utilisation des technologies telles que XQuery, *XSLT*, *JDOM* ou Saxon.

Le principal objectif de ce chapitre était de valider le métamodèle explicité au Chapitre 6. En effet, à travers l'architecture développée dans ce chapitre, nous expliquons la manière avec laquelle se déroule notre chaîne de traitement. Tout commence par la production d'un fichier de métadonnées contenant les informations sur lesquelles seront appliqués nos scénarios d'interprétation. Ces scénarios d'interprétation sont traduits dans le langage de requêtes XQuery. Le résultat de ces scénarios sont donnés sous la forme de fichiers XML transformés à l'aide du langage de transformation XSLT pour produire des documents HTML.

Chapitre 8

Ontologie de la classification

Introduction

L'ontologie d'un domaine est censée expliciter les concepts qui existent dans le monde du discours ainsi que leurs relations. En ce sens, une ontologie ne retiendra du domaine à modéliser qu'un nombre fini de relations entre les objets de ce domaine. Ce qui implique la définition de l'objectif pour lequel l'ontologie est censée répondre. Dans le cadre de nos travaux, notre ontologie a pour objectif d'aider à l'interprétation *automatique* des résultats de classification. Ainsi, nous ne retiendrons comme concepts et relations de notre ontologie que ceux qui pourraient nous aider à atteindre cet objectif. Ce chapitre, consacré à la description et à l'utilisation de notre ontologie, se décompose de la manière suivante :

- la section 8.1 présente les raisons pour lesquelles nous pourrons être amenés à utiliser une ontologie ;
- la section 8.2 présente les objectifs de notre ontologie ;
- la section 8.3 est relative au processus de conception et de création de l'ontologie dédiée au domaine de la classification automatique. Dans cette section, nous abordons les outils utilisés pour développer notre ontologie. Dans cette même section, nous présentons les principes sur lesquels nous nous sommes appuyés dans le développement de notre ontologie. Cette section aborde aussi l'ensemble des concepts, des propriétés et des axiomes utilisés dans notre ontologie ;
- la section 8.4 aborde les notions de l'API *Jena* qui nous permet de manipuler

les éléments de notre ontologie ainsi que la modélisation et l'implémentation des concepts de notre ontologie ;

- la section 8.5 donne quelques pistes de réflexion sur les choix à faire au moment de l'évaluation d'une ontologie.

Mais avant, nous apportons des éléments de comparaison entre les concepts de *systèmes à base de connaissances* et *ontologie*.

Différences entre systèmes à base de connaissances et ontologies

Ce paragraphe est l'occasion pour nous de faire le point sur la différence qu'il y a entre un système à base de connaissances et une ontologie. Ces différences ont été avancées par R. Mizoguchi. Pour lui, les ontologies fournissent une spécification des connaissances et des modèles dans le système, tandis que les bases de connaissances conventionnelles sont utilisées pour la résolution de problèmes. Le rôle d'une ontologie envers une base de connaissances est de donner des définitions des concepts utilisés dans la représentation des connaissances. De plus, les ontologies permettent de spécifier les contraintes entre les concepts, afin de rendre la base de connaissances plus consistante et transparente. En effet, ces deux propriétés sont nécessaires pour pouvoir partager et réutiliser la connaissance, elles représentent le cœur des systèmes d'intelligence artificielle.

8.1 Contextes d'utilisation d'ontologie

D'une manière générale, un des aspects les plus recherchés lors de l'utilisation d'une ontologie est de permettre le partage de l'information entre différentes sources ou entités. Pour que cette utilisation soit effective, il peut être nécessaire d'opérationnaliser l'ontologie en lui dotant d'un certain nombre d'axiomes. Selon le type de langage utilisé pour écrire une ontologie, nous avons un pouvoir d'expressivité plus ou moins important. Ainsi, une ontologie écrite dans un langage basé sur la logique de description a un grand pouvoir d'expressivité sémantique. Et comme nous l'expliquons dans la section 4.1.3, pour qu'une ontologie soit opérationnelle, elle doit être porteuse d'une sémantique formelle. Grâce à cette sémantique, l'ontologie permettra d'effectuer certains raisonnements :

- l'appartenance à une classe : si x est instance d'une classe C, et C est sous-classe de D, alors x est une instance de D.
- l'équivalence de classes : si la classe A est équivalente à la classe B, et que la classe B est équivalente à la classe C, alors la classe A est équivalente à la classe C.
- la consistance : supposons que x est une instance de la classe A, et supposons également que :
  - A est une sous-classe de $B \cap C$
  - A est une sous-classe de D
  - B et D sont disjointes.

Utiliser ce type de raisonnement peut, par exemple, nous montrer qu'il existe une inconsistance puisque A devrait être vide, alors qu'il possède une instance x. Grâce à cette propriété, nous pouvons nous apercevoir qu'une ontologie est inconsistante par exemple.

- la classification : si certaines propriétés ont été déclarées comme suffisantes pour les membres d'une classe A, et si un individu x satisfait ces conditions, alors x est instance de A.

Nous voyons, à travers les propriétés évoquées précédemment que l'utilisation d'une ontologie peut aider à ajouter de la sémantique sur des concepts d'un domaine et de définir des contraintes fortes pour un domaine et un objectif donnés. Comme dans la plupart des ontologies, ces *systèmes de raisonnements structurels* sont très souvent utilisés et à juste titre d'ailleurs. Dans le cadre de nos travaux, nous nous attachons à utiliser d'autres raisonnements basés sur l'objectif de notre ontologie. Ainsi, nous avons, entre autres, les axiomes suivants :

- Une *DiscriminantVariable* (c'est-à-dire variable discriminante) est une variable pour lequel le critère (*Criterion*) a une valeur supérieure à un certain seuil (*threshold*) ;
- Une *variable discriminante pour une classe* est une variable pour laquelle le critère (*Criterion*) est supérieur à un certain seuil (*threshold*) fixé pour le *Criterion* de la classe considérée ;
- Une *classe est homogène* si les dissimilarités entre les *individus* de cette classe sont faibles et leurs *similarités* fortes.

Nous pouvons avoir autant d'axiomes que l'autorise une meilleure description du domaine de l'interprétation des résultats de classification.

8.2 Objectifs de notre ontologie

Dans cette section, nous rappelons le contexte général qui pousse à développer l'ontologie d'un domaine. Ainsi, selon Noy, on développe une ontologie pour les objectifs suivants [Noy and McGuinness, 2001] :

- partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels ;
- permettre la réutilisation du savoir sur un domaine ;
- expliciter ce qui est considéré comme implicite sur un domaine ;
- distinguer les connaissances sur un domaine avec les traitements mis en œuvre pour les rendre opérationnelles ;
- analyser la connaissance sur un domaine.

Partager la compréhension commune de la structure de l'information entre les personnes ou les fabricants de logiciels est une des raisons les plus courantes qui conduit à développer des ontologies [Musen, 1992], [Gruber, 1993]. Pour appuyer leurs propos, ils ont pris l'exemple d'un ensemble de sites Web contenant de l'information médicale ou fournissant des services de e-commerce en médecine. Si ces sites partagent et publient tous la même ontologie, qui est à la base des termes qu'ils utilisent, alors les agents informatiques peuvent extraire et agréger l'information de ces différents sites. Les agents peuvent utiliser cette information agrégée pour pouvoir répondre aux interrogations des utilisateurs ou comme données d'entrée pour d'autres applications.

Permettre la réutilisation du savoir sur un domaine semble aussi être un autre objectif sur l'utilisation d'ontologie. En effet, une fois qu'une ontologie sur un domaine est créée, alors celle-ci est réutilisée par tous les membres de la communauté du domaine. De plus, la description partielle d'un domaine, suivant les objectifs, peut aider à la création d'une ontologie générale du domaine en question. De même, une ontologie générale peut être spécifiée pour un domaine d'intérêt particulier. C'est ce que nous faisons dans le cadre de cette thèse en nous appuyant sur l'ontologie du domaine de la fouille de données orientée pour les *Grids* (cf. [Cannataro and Comito, 2003])

Rendre explicites les postulats d'un domaine sous-jacents à une implémentation est un objectif intéressant car cette explicitation rend possible la modification en cas d'évolution du savoir sur le domaine. En effet, par nature, les postulats implicites sur un domaine codés dans un langage de programmation deviennent difficiles à deviner, à comprendre mais aussi à modifier. Cette explicitation permettra ainsi d'aider les nouveaux utilisateurs à apprendre la signification des termes du domaine.

Distinguer les connaissances sur un domaine avec les traitements mis en œuvre pour les rendre opérationnelles est une autre finalité d'une ontologie. En effet, nous pouvons décrire la tâche de configuration d'un produit à partir de ses constituants, en respectant les spécifications requises et implémenter un programme qui réalisera cette configuration indépendamment des produits et de leurs composants [McGuinness and Wright, 1998]. On peut alors développer une ontologie des parties composantes et des caractéristiques d'un PC et appliquer l'algorithme pour configurer des PC sur mesure. On peut aussi utiliser le même algorithme pour configurer des ascenseurs si cet algorithme est alimenté par une ontologie des parties composantes des ascenseurs [Rothenfluh et al., 1996].

Analyser le savoir sur un domaine est possible dès que la spécification des termes du domaine est faite. L'analyse formelle des termes est extrêmement précieuse aussi bien quand on veut réutiliser les ontologies existantes que quand on veut les étendre [McGuinness et al., 2000].

Dans nos travaux, ***l'objectif principal*** de notre ontologie est de définir une base de connaissances exploitable par notre application et qui permette d'aider à l'interprétation. Mais cet objectif ne se limite pas seulement à clarifier les concepts utilisés dans le domaine de la classification ; car dans ce cas de figure l'utilisation d'un vocabulaire contrôlé aurait été une formalisation suffisante. Il s'agit, pour nous, de définir les différents concepts, propriétés et axiomes pouvant aider à formaliser le domaine de la classification. Cette formalisation est l'une des conditions pouvant permettre la définition des liens sémantiques entre les concepts du domaine de la classification. *Concrètement*, les axiomes définis sur cette ontologie doivent permettre, par exemple, le classement de nouveaux critères d'interprétation et ce, en se basant uniquement sur la description faite par l'utilisateur sur le critère qu'il désire utiliser. De plus, notre ontologie doit être capable de vérifier si un critère donné peut être utilisé de manière pertinente dans la description des résultats obtenus.

8.3 Processus de création de l'ontologie

Dans la section 4.1.3, nous avons expliqué que la mise en place d'une ontologie se déroule en trois étapes principales. Sa conception passe par une première étape d'abstraction. Il s'agit de déterminer quelles sont les connaissances à représenter ainsi que le degré de précision et/ou de granularité. La seconde étape consiste à déterminer le formalisme de représentation à adopter pour représenter les connaissances découvertes. D'une manière générale, dans le processus de construction d'une ontologie un certain nombre de questions méritent réflexion. Ainsi, il convient de déterminer s'il vaut mieux partir de rien ou partir d'une ontologie existante ou encore partir d'un corpus. Ensuite s'en suivent d'autres questions relatives au degré d'automatisation à adopter dans le processus de construction (manuel, semi-automatique ou automatique), à l'intégration (s'il y a lieu) d'ontologies existantes, ainsi de suite. Ces types d'interrogation sont ceux auxquels est confronté un concepteur d'ontologie. Il s'agit pour lui d'y apporter des réponses afin de « baliser son chemin ».

Par ailleurs, comme nous le soulignons dans la section 4.1.4, il existe plusieurs méthodes et outils de construction d'ontologie. Ces méthodes et outils sont utilisés suivant les contraintes au moment du développement de l'ontologie d'une part, et suivant les objectifs fixés d'autre part. Dans le cadre de *nos travaux*, compte tenu de l'absence d'un point de départ, nous avons construit notre ontologie *manuellement* à partir des informations tirées de la littérature et à travers les échanges avec les experts du domaine de la classification. Ce qui est en soi une tâche longue et fastidieuse. En effet, une fois que les concepts de base sont définis, il s'agit ensuite de définir leurs propriétés. En considérant le domaine de la classification et dans l'objectif d'aider à l'interprétation des résultats de méthodes de classification, nous avons défini les concepts et les propriétés de notre ontologie, respectivement dans les sections 8.3.2 et 8.3.3.

Quant à l'outil utilisé pour la création de cette ontologie, nous avons utilisé Protégé-2000 qui a, entre autres, la particularité de maintenir un espace de nommage unique pour tous ces cadres. Mais il faut savoir que c'est un outil qui est sensible à la casse, ainsi *Critère* et *critère* sont deux classes différentes. L'ontologie générée par l'outil PROTÉGÉ2000 est codée dans des fichiers RDF ou OWL compréhensibles par la machine (cf. à l'annexe C pour le fichier de description de notre ontologie en OWL). Nous expliquons dans cette section ce processus de création d'ontologie. Dans ce sens, nous apportons quelques remarques sur l'outil PROTÉGÉ2000. PROTÉGÉ2000 est un système qui décrit les ontologies de manière déclarative en utilisant la notion de *frame* (cf. figure 8.1). En effet, PROTÉGÉ2000 permet aux utilisateurs d'indiquer de manière explicite à quelle classe appartiennent les individus qu'ils créent. Aussi, il convient de rappeler que la création d'une ontologie ne se limite pas simplement à la création des classes et des relations entre elles, mais il s'agit aussi de mettre en œuvre des axiomes. D'ailleurs la création d'une classe, en programmation objet, suppose que le programmeur réalise les décisions conceptuelles en s'appuyant sur les propriétés opérationnelles de la classe tandis qu'un concepteur d'ontologie le fait en s'appuyant sur les propriétés structurelles de la classe.

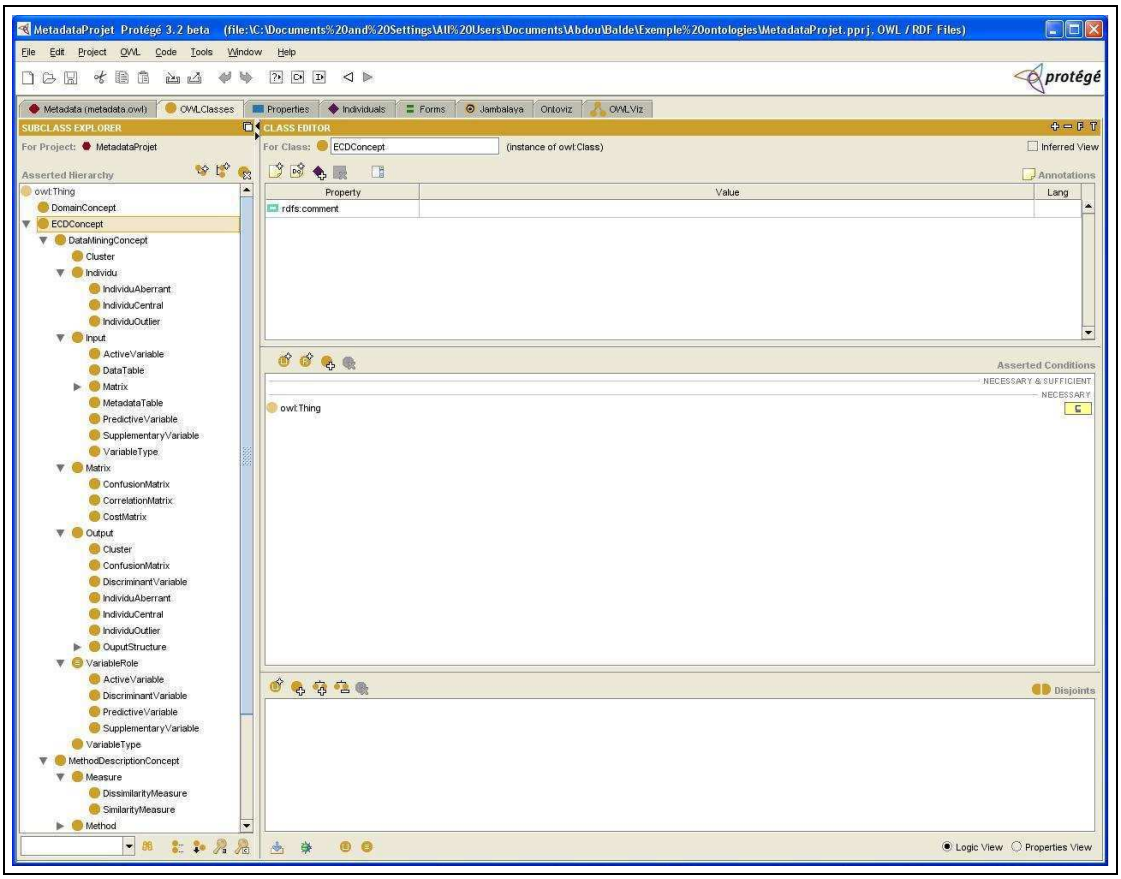

FIG. *8.1 - Une vue du logiciel* PROTÉGÉ2000 *version 3.2*

Pour résumer, il convient de rappeler que développer une ontologie s'effectue de la manière suivante :

- définir les classes dans l'ontologie ;
- arranger les classes en une hiérarchie taxinomique (*sous-classe, super-classe*) ou autre ;
- définir les attributs et décrire les valeurs autorisées pour ces attributs ;
- définir les axiomes de l'ontologie : ces axiomes doivent tenir compte de l'objectif poursuivi ;
- renseigner les valeurs pour les attributs des instances.

C'est ce que nous tâchons d'expliquer dans les sections relatives à la définition de nos concepts, relations et axiomes. En définitive, développer une ontologie s'apparente à définir un ensemble d'informations et leur structure pour qu'elles soient utilisées par d'autres applications.

8.3.1 Stratégies de développement d'ontologie

D'une manière générale, la méthodologie de développement d'une ontologie est similaire, au niveau supérieur, à celle d'une hiérarchie orientée-objet. En effet, l'ontologie se concentre sur les aspects déclaratifs. Ce qui signifie que les descriptions sont faites de façon déclarative dans la plupart des cas afin d'assurer les caractères formels et explicites de l'ontologie. Néanmoins, il existe différentes approches pour la mise en œuvre des hiérarchies de classe. C'est ce qu'explique USCHOLD dans [Uschold and Grüninger, 1996]. Ainsi nous pouvons avoir :

- la stratégie de développement de *haut en bas ou top-down* qui consiste à commencer par définir les concepts les plus généraux du domaine et ensuite les spécialiser au fur et à mesure de la construction de l'ontologie ;
- la stratégie de développement de *bas en haut ou bottom-up* qui consiste à commencer par définir les classes les plus spécifiques puis de regrouper ces classes en concepts plus généraux ;
- la stratégie *mixte* qui consiste à combiner ces deux approches ; ainsi on commence par définir les concepts les plus saillants puis, ces derniers sont généralisés ou spécialisés selon les cas.

Dans le cadre de nos travaux, nous optons pour une stratégie *mixte*. En effet, compte tenu des difficultés auxquelles nous avons été confrontés, il semblait préférable de partir par des concepts connus et admis par tous et puis d'essayer de voir si ces concepts étaient à généraliser ou à spécialiser. Notons que, pour des questions pratiques, nous avons supposé que notre ***ontologie n'évolue pas***. Dans la section suivante, nous définissons ces concepts avant d'aborder dans les sections suivantes les propriétés et les axiomes utilisés pour rendre notre ontologie opérationnelle.

8.3.2 Éléments de notre ontologie

A. Gómez-Pérez rappelle, dans [Gómez-Pérez, 1999], les informations à définir pour la construction d'une ontologie. Il s'agit des notions de :

- *Concepts :* connus aussi sous le nom de *termes* ou de *classes*, ils correspondent aux abstractions pertinentes d'un segment de la réalité, retenues en fonction des objectifs fixés et de l'application envisagée pour l'ontologie. Selon [Gómez-Pérez, 1999], ces concepts peuvent être classifiés suivant plusieurs dimensions : *niveau d'abstraction*, *l'atomicité* et le *niveau de réalité*. A ce niveau, il convient de remarquer qu'un *Concept* est indépendant de la façon dont on le nomme. En effet, nombreux sont ceux qui passent un temps énorme dans les problèmes terminologiques quand ils développent une ontologie. Il semble donc nécessaire de faire la distinction entre une question terminologique et une question ontologique.
- *Relations :* ce sont des notions qui traduisent les associations (pertinentes) existant entre les concepts présents dans le segment analysé du domaine d'études. Ces relations incluent les associations suivantes : *généralisation*, *spécialisation*, *agrégation* ou *composition*.
- *Fonctions :* il s'agit des cas particuliers de *relations*, dans lesquelles un élément de la relation, le n^{ieme} (*extrant*) est défini en fonction des $n-1$ éléments précédents (*intrants*).
- *Axiomes :* ils constituent des assertions, acceptées comme vraies, à propos des abstractions du domaine traduites par l'ontologie.
- *Instances :* ce sont des concepts qui représentent la définition de l'extension de l'ontologie ; ces objets véhiculent les connaissances (statiques, factuelles) à propos du domaine analysé.

Comme nous l'expliquons dans la section 8.1, il faut savoir que l'une des raisons pour utiliser une ontologie dans un système d'information s'explique par le besoin (ou le désir) de partager des connaissances avec le minimum d'ambiguïté. Ainsi, l'ontologie peut être vue comme une composante apportant une dimension sémantique aux systèmes d'information qui leur faisait jusqu'à présent défaut.

Principes de développement d'ontologie

L'objectif de ce paragraphe est de rappeler les principes théoriques sur lesquels nous nous basons pour la conception de notre ontologie. Ces principes sont ceux énoncés par N. Noy dans [Noy and McGuinness, 2001] :

- il n'y a pas qu'une seule façon correcte pour modéliser un domaine, il y a toujours des alternatives viables. La meilleure solution dépend presque toujours de l'application à mettre en place et des évolutions qu'il faut anticiper ;
- le développement d'une ontologie est nécessairement un processus itératif ;

- les concepts dans une ontologie doivent être très proches des objets (physiques ou logiques) et des relations du domaine à modéliser. La plupart du temps, il s'agit des noms (concepts) ou verbes (relations) dans des phrases qui décrivent le domaine considéré.

Ainsi, à travers ces principes, il a fallu considérer les concepts les plus adaptés à la tâche que nous nous sommes fixés. Par ailleurs, NOY rappelle qu'une ontologie est un modèle du monde réel et que les concepts de l'ontologie doivent pouvoir refléter cette réalité. Tâche à laquelle nous nous sommes attelés lors du développement de notre ontologie. Le paragraphe suivant explicite les concepts de notre ontologie.

Spécification de nos concepts

Dans ce paragraphe, nous spécifions l'ensemble des concepts de notre ontologie. Commençons par les deux concepts les plus généraux de notre ontologie. Il s'agit des concepts *DomainConcept* et *ECDConcept* :

- ***DomainConcept*** représente l'ensemble des concepts éventuels du domaine analysé. L'avantage de ce concept est de permettre, à l'avenir, la prise en charge d'une ontologie du domaine analysé afin de faciliter une meilleure interprétation sémantique des classes obtenues. Ce point de vue n'est pas abordé dans le cadre de cette thèse mais il n'en constitue pas moins une piste de réflexion très intéressante dans le cadre de l'aide à l'interprétation automatique des résultats des méthodes de fouille de données.
- ***ECDConcept*** représente l'ensemble des concepts que nous utilisons dans le cadre de ce travail. C'est donc le concept de base auquel se rattache tous les autres concepts.

La figure 8.1 donne une vue d'une partie de notre ontologie développée dans l'outil PROTÉGÉ2000. Les concepts suivants sont ceux qui se situent au 2^{ieme} niveau de notre ontologie ; il s'agit des concepts suivants : *DataMiningConcept et ResultInterpretingConcept*. Comme leur nom l'indique, ces concepts décrivent, selon notre point de vue, les méthodes de fouille de données.

- ***DataMiningConcept*** définit l'ensemble des concepts qui semblent nécessaires dans le processus d'aide à l'interprétation. Encore une fois, le but n'est pas de décrire tous les concepts du domaine de la fouille de données ou de la classification automatique mais de créer un cadre pouvant être enrichi par la suite.
- ***ResultInterpretingConcept*** définit des concepts souvent utilisés dans le cadre de l'interprétation.

S'en suivent les concepts spécifiés à partir de ceux définis précédemment. Ainsi, au niveau du *DataMiningConcept*, nous avons les concepts suivants :

- ***Individual*** qui est relatif à la description des individus (appelés aussi objets) en entrée des méthodes de fouille de données. A travers ce concept, nous pouvons avoir accès à l'*axe d'interprétation « Individu »*.
- ***Input*** qui exprime les connaissances à conserver sur les entrées des méthodes. Dans ce contexte, cette ontologie pourrait aider à répondre aux questions du type : *quel est l'outil adapté à tels types de données*. Dans notre travail, ce concept va servir à la description de l'historique sur les données traitées en permettant par exemple de retrouver les métadonnées issues des bases de données. Ce concept, pour mieux refléter notre point de vue, a fait l'objet d'une certaine spécialisation. Ce qui a donné les concepts suivants : ***ActiveVariable, PredictiveVariable, SupplementaryVariable, DataTable, MetadataTable, VariableType***.
- ***OutputStructure*** qui est relatif à la description des sorties des méthodes. Ce concept permet de conserver l'information sur les structures proposées par les méthodes de classification.
- ***VariableType*** qui définit les valeurs possibles pour les types de variables. Le rôle de ce concept est de définir des contraintes relatives au typage des variables.
- ***VariableRole*** qui permet de définir les rôles que peuvent avoir les variables lors d'une interprétation. Compte tenu de l'intérêt porté à ce concept, nous spécifions en définissant trois nouveaux concepts qui sont : ***ActiveVariable, PredictiveVariable*** et ***SupplementaryVariable*** qui comme, leur nom le laisse supposer, sont relatifs aux caractérisations des variables suivant leurs rôles lors du processus de classification.
- ***Method*** qui explique ce qu'est une méthode de fouille de données. Il s'agit, à partir de la spécification de ce concept, de pouvoir classer automatiquement de nouvelles méthodes de fouille de données à partir de leur description. Ce concept étant très générique, nous avons décidé de le spécifier en définissant deux nouveaux concepts : ***SupervisedMethod*** et ***UnSupervisedMethod*** qui sont plus proches de la réalité de nos travaux. Aussi, pour le concept *UnSupervisedMethod*, nous définissons le concept ***Clustering*** qui, pour nous, n'est pas une instance mais bien un concept. Nous considérons, à ce niveau, comme instance les algorithmes développés dans le cadre des méthodes de *clustering*.
- ***Parameter*** qui est relatif à la description des paramètres des méthodes.
- ***Measure*** qui concerne les mesures utilisées lors du processus de traitement. En effet, nous savons que le choix d'une mesure peut avoir un impact sur les résultats de certaines méthodes. Donc il semble important de pouvoir intégrer cet aspect dans notre processus d'interprétation. Ce concept fait l'objet d'une spécialisation en deux nouveaux concepts qui répondent à la problématique de nos travaux. Il s'agit des concepts ***SimilarityMeasure*** et ***DissimilarityMeasure*** qui décrivent mieux notre façon de modéliser les mesures de distance entre objets.

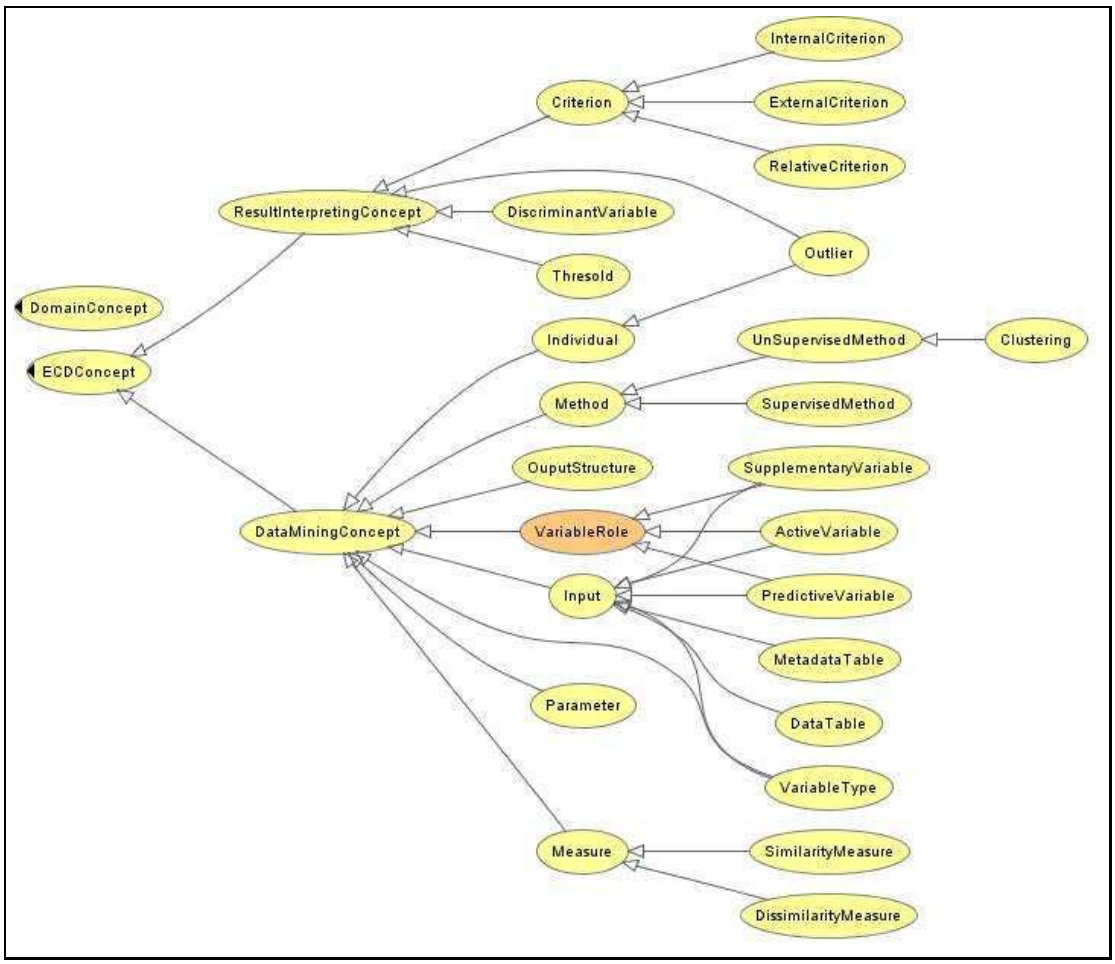

FIG. *8.2 - Représentation de notre ontologie*

Enfin, la spécialisation du concept *ResultInterpretingConcept* nous donne les concepts suivants :

- ***Criterion*** est le concept qui définit ce qu'est un critère d'interprétation ou de validation. En effet, comme nous l'expliquons au Chapitre 6, il existe divers critères d'interprétation et/ou de validation mais ceux-ci peuvent être classés en trois grandes catégories. Ce qui a conduit à la création de nouveaux concepts plus spécifiques qui sont : ***ExternalCriterion, InternalCriterion, RelativeCriterion***.
- ***Threshold*** indique le concept exprimant l'interprétation d'un résultat de classification à travers la coupure par rapport à un seuil défini par l'utilisateur ou par l'outil de classification.
- ***DiscriminantVariable*** est le concept définissant la notion de *variable discriminante*.
- ***Outlier*** exprime la notion d'objets ayant une caractéristique très différente des autres membres de la classe.

Une vue d'ensemble de ces concepts est donnée, à travers le plug-in *OWLViz* disponible dans PROTÉGÉ2000, dans la figure 8.2.

8.3.3 Propriétés de notre ontologie

Les propriétés, appelées aussi *rôles* ou *attributs* dans certains travaux, décrivent les caractéristiques des *Concepts* ainsi que les liens qui peuvent les unir. Dans nos travaux, nous définissons l'ensemble des propriétés utilisées pour décrire les concepts définis dans la section précédente. Par ailleurs, il est important de rappeler que nous n'abordons pas les propriétés intrinsèques de nos concepts, l'objectif est de se limiter uniquement aux relations entre les concepts. Car il existe dans le langage OWL, d'autres catégories de propriétés : les propriétés d'objets liant des individus à d'autres individus et les propriétés de types de données liant des individus à des valeurs de données. Dans notre ontologie développée sous PROTÉGÉ2000, nous avons utilisé les notions de *ObjectProperty* et de *DatatypeProperty* pour décrire l'ensemble des propriétés de notre ontologie.

- ***Describe :*** cette propriété exprime la relation entre les concepts *Parameter* et *Method*. En effet, chaque concept *Parameter* décrit un concept *Method*. Il s'agit d'ailleurs d'une relation inverse de la relation *isDescribedBy*.
- ***isDescribedBy :*** comme expliqué précédemment, cette relation est l'inverse de la relation *Describe*.
- ***hasType :*** cette propriété explique le fait que pour chaque concept *VariableRole*, il existe un type prenant ses valeurs à travers le concept *VariableType*.
- ***hasCriteria :*** à travers cette propriété, on exprime le fait que les concepts *VariableRole* et *Criterion* sont liés. Ainsi, le concept *VariableRole* est décrit par un ou plusieurs critères.
- ***hasGlobalCriteria :*** exprime les relations entre les concepts *Method* et *ExternalCriterion*. En effet, les méthodes de fouille de données contiennent généralement des critères externes permettant de décrire leurs résultats. Il s'agit par exemple de la description des résultats issus du logiciel WEKA à travers le critère du *f-mesure*.
- ***hasResult :*** exprime le fait que les méthodes de fouille de données produisent une ou plusieurs structures différentes. Cette propriété fait le lien entre les concepts *Method* et *OutputStructure*.

Néanmoins, voici un exemple écrit en *OWL*, exprimant le fait que les classes *Criterion, Method, Parameter* et *VariableRole* partagent la même propriété *Name*.

```
<owl:DatatypeProperty rdf:ID="Name">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#Name"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Criterion"/>
        <owl:Class rdf:about="#Method"/>
        <owl:Class rdf:about="#Parameter"/>
        <owl:Class rdf:about="#VariableRole"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
```

8.3.4 Axiomes de notre ontologie

Avant d'expliquer les axiomes utilisés dans nos travaux, il convient de rappeler un aspect important dans la réalisation de notre ontologie. En effet, comme nous l'expliquons dans la section 8.2, l'objectif de cette ontologie n'est pas d'expliciter tous les concepts, axiomes et propriétés susceptibles d'être rencontrés dans le cadre d'une ontologie répondant à des besoins importants. Il s'agit pour nous, à travers quelques essais, de montrer l'intérêt d'utiliser une ontologie pour améliorer l'automatisation de l'aide à l'interprétation des résultats issus de méthodes de classification.

Voici une portion de notre ontologie, écrite en *OWL*, qui montre les axiomes de type *owl : disjointWith*. En effet, chaque déclaration de ce type exprime le fait que les deux descriptions de classe concernées n'ont aucun individu en commun. Un autre axiome utilisé dans cette ontologie est *rdfs : subClassOf*. Dans les deux cas, ce type de déclaration exprime une définition partielle de ce qu'est une classe (ou concept) : elles imposent une condition nécessaire mais non suffisante sur la classe.

```
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="SupervisedMethod">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Method"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
    <owl:Class rdf:ID="UnSupervisedMethod"/>
  </owl:disjointWith>
</owl:Class>
```

Dans cet exemple, nous voyons que les classes *SupervisedMethod* et *UnSupervisedMethod* sont des sous-classes de *Method* mais sont disjointes entre elles. Ce qui signifie qu'il ne peut y avoir deux instances identiques provenant de ces deux classes. L'exemple suivant exprime un autre axiome sur les valeurs que peut prendre la classe *VariableType*. De plus, cette classe est une sous-classe des classes *Input* et *DataMiningConcept*.

```
<owl:Class rdf:about="#VariableType">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <VariableType rdf:ID="Binaire"/>
        <VariableType rdf:ID="Categorique"/>
        <VariableType rdf:ID="Interval"/>
        ........................
      </owl:oneOf>
    </owl:Class>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Input"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#DataMiningConcept"/>
  </rdfs:subClassOf>
</owl:Class>
```

En clair, les axiomes expriment des contraintes sur les *propriétés*, les *classes* ou les *instances*. Par exemple, un axiome de propriété peut définir simplement l'existence d'une propriété (forme minimale d'un axiome de propriété). D'autres types d'axiomes existent, il s'agit des axiomes liés au domaine de l'ontologie. Nous donnons des exemples dans la section 8.1. En effet, grâce à ce type d'axiomes, les raisonnements sur la base de connaissances que représente l'ontologie deviennent plus faciles. Nous verrons d'ailleurs dans le paragraphe suivant la différence entre les ontologies et les systèmes à base de connaissances.

8.4 Utilisation de l'API *Jena*

Jena est une API qui permet de gérer des ontologies écrites dans les langages *RDF-Schema*, *DAML+OIL* ou *OWL* et de faire des raisonnements en utilisant les connaissances de l'ontologie. Dans *Jena*, une ontologie est un *Model*. Et plus particulièrement un *OntModel*, qui est une spécialisation de *Model*. Cette classe contient les méthodes spécifiques aux ontologies :

- création de classe : il existe des méthodes permettant ce type d'action, il s'agit de la méthode *createClass* ;
- création de propriété : de même que la méthode précédente, il existe d'autres méthodes permettant de créer les propriétés pour des objets (*ObjectProperty*).

Une description plus détaillée se trouve dans [Jena, 2006]. Concrètement *Jena* est une bibliothèque de classes Java qui facilite le développement d'applications pour le Web sémantique qui permet la manipulation de déclarations du type :

- lecture et écriture RDF/XML ;
- stockage en mémoire ou sur disque de connaissances RDF ;
- langage d'interrogation d'une base RDF ;
- gestion d'ontologies écrites en OWL par exemple.

Nous pouvons, à partir des méthodes des éléments *OntClass* et *ObjectProperty* obtenus à travers les méthodes correspondantes de l'API : *createClass* et *ObjectProperty*, faire des traitements du type : *quels sont les concepts qui ont la propriété p ?* Le but est de pouvoir inférer de nouvelles connaissances à partir de la description de notre ontologie.

Modélisation de notre ontologie en utilisant des patrons de conception

Pour utiliser notre ontologie dans cette approche, nous avons modélisé les différents concepts, propriétés et axiomes de notre ontologie en utilisant des *patrons de conception* connus en ingénierie des connaissances. Nous présentons le *design pattern* ***Stratégie*** (cf. annexe A pour l'ensemble des design patterns) qui a servi à modéliser les concepts de notre ontologie relatifs à la recherche de *variables discriminantes* par exemple. Il s'agit de montrer d'une part l'intérêt de leur utilisation et d'autre part l'utilisation de notre ontologie dans notre modèle d'interprétation. Le pattern ***Stratégie*** permet de déléguer à d'autres classes ou interfaces l'implémentation des algorithmes. En effet, ce pattern est utilisé lorsqu'il existe plusieurs façons d'implémenter un algorithme et que le choix de cet algorithme doit se faire au moment de l'exécution. L'idée est de rendre des objets interchangeables grâce au polymorphisme. La seule contrainte est que ces objets doivent implémenter la même interface. Dans notre cas, ce pattern va nous permettre de définir et d'ajouter de nouveaux scénarios d'interprétation basés sur le calcul des valeurs d'un certain nombre de critères d'interprétation. Comme nous ne pouvons connaître à l'avance tous les scénarios d'interprétation à utiliser, cette conception utilisant le pattern ***Stratégie*** semble la mieux adaptée (cf. figure 8.3). L'utilisation de ce pattern de conception est réalisée par exemple pour la définition de scénarios d'interprétation (l'interface *Scenario-Discriminant* va jouer le rôle de ce patron de conception). Ainsi, pour ajouter un nouveau scénario qui vient d'être défini dans l'ontologie, il suffit de spécialiser l'interface *ScenarioDiscriminant* et d'utiliser ce nouveau scénario dans la classe *RésultatClassification*. Et comme, dans ce contexte, c'est la composition que nous utilisons au lieu de l'héritage, nous n'avons pas de problème lié, entre autres, à la duplication de codes et à la difficulté de maintenance. Par ailleurs, il convient de rappeler qu'il existe des travaux visant à mettre en place des *patrons de conception ontologiques* qui permettent une modélisation cohérente et efficace des différentes tâches communes utilisées et partagées par la majorité des ontologies (cf. aux travaux de [Blomqvist, 2005], [Blomqvist and Sandkuhl, 2005] pour plus de renseignements).

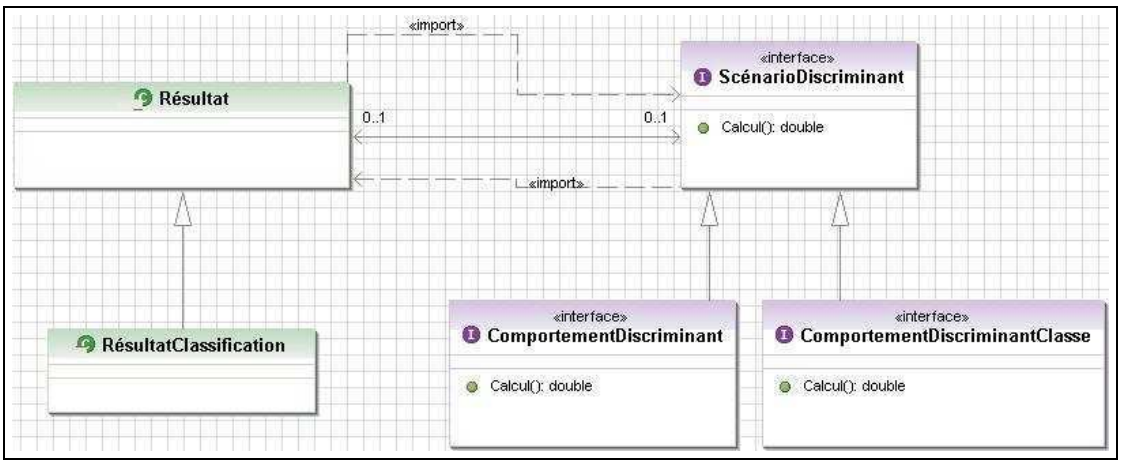

FIG. *8.3 - Utilisation du pattern Stratégie*

8.5 Evaluation d'ontologie

L'objectif de cette section est d'expliquer qu'il ne suffit pas simplement de concevoir et d'opérationnaliser une ontologie mais il faudrait aussi et surtout l'évaluer pour s'apercevoir si l'ontologie satisfait ou non les objectifs de départ. Et pour évaluer une ontologie, il existe plusieurs stratégies connues à ce jour (pour plus de détails cf. [Brank et al., 2005, Hartmann et al., 2005]). L'une de ces méthodes est de faire une évaluation en se basant sur les connaissances d'un expert du domaine. D'une manière générale, pour évaluer une ontologie, les travaux réalisés à ce jour se basent sur un certain nombre de critères qu'une ontologie est censée vérifier. Dans la construction d'une ontologie, l'aspect de la validation de l'ontologie est primordial. En effet, la validation consiste à vérifier la cohérence de l'ontologie d'un point de vue système. Il peut aussi s'agir de l'audit des connaissances et d'une validation par les experts du domaine ou d'une validation par les utilisateurs [Grüninger and Fox, 1995], [Gòmez-Pérez, 1999], [Gòmez-Pérez et al., 2003].

8.6 Synthèse du Chapitre 8

Dans ce chapitre, nous avons abordé les notions liées à la spécification de notre ontologie dans le cadre de l'aide à l'interprétation de résultats. A ce niveau, il convient de rappeler qu'une ontologie facilite la représentation des connaissances déclaratives, ce qui permet au système de modifier son comportement en modifiant la connaissance qu'il possède. Ces connaissances permettent au système d'expliquer son comportement. Dans ce chapitre, il s'agissait pour nous d'expliquer de quelles manières nous pouvons utiliser l'ontologie du domaine de la classification à des fins d'aide à l'interprétation automatique des résultats de méthodes de classification.

Chapitre 9

Applications de notre approche

Introduction

Comme nous l'expliquons depuis les chapitres précédents, les éléments de métadonnées extraits lors du processus de classification doivent aider à interpréter convenablement les résultats des méthodes de classification. Ces éléments sont de deux types : les métadonnées correspondant aux informations fournies par l'utilisateur qui a effectué la classification et celles qui sont associées aux données et aux résultats de cette classification. Voici des exemples de métadonnées correspondant au 1^{er} type : le *nombre de classes souhaitées*, le *nombre d'exécutions maximum*, les *informations sur l'auteur*, la *description de la méthode de classification*, la *distance utilisée*, *l'unité de mesure utilisée* pour certaines valeurs de données, etc. Les métadonnées liées aux données et aux résultats sont par exemple : les *paramètres de la méthode* de classification, le *nombre d'individus* dans une classe, la *source des données originales*, la *description des variables*, *l'usage de chaque variable* (active, prédictive, etc.), les *valeurs des critères* d'hétérogénéité et/ou d'isolation pour chaque classe, la *contribution* de chaque variable dans la construction des classes de la partition, etc.

L'objectif de ce chapitre est d'expliquer le fonctionnement de notre application. Pour ce faire, nous choisissons de prendre deux jeux de données (les *Iris et les données météorologiques de* 60 *stations de Chine*) sur lesquels nous appliquons l'algorithme de classification dynamique *Sclust* [Chavent et al., 2006], la méthode des nuées dynamiques ainsi que deux méthodes (KMeans et EM) implémentées dans l'outil WEKA. Le but est de montrer nos contributions en matière d'interprétation des résultats issus de ces diverses méthodes. Ce chapitre s'articule autour des sections suivantes :

- la section 9.1 présente le processus de traitement de notre application ;
- la section 9.2 présente les critères d'interprétation que nous utilisons dans le cadre de cette application pour expliquer l'avantage de notre approche (ces critères n'étant évidemment pas exhaustifs) ;
- la section 9.3 présente les différentes fonctions de notre outil d'interprétation ;
- la section 9.4 présente le premier scénario d'interprétation que nous implémentons dans le cadre de cette thèse ;
- la section 9.5 présente le second scénario d'interprétation que nous utilisons dans notre outil afin d'aider à l'interprétation des résultats ;
- la section 9.6 aborde l'un des avantages de notre approche à savoir la facilité de modifier et d'ajouter un scénario d'interprétation ;
- la section 9.7 donne les résultats obtenus en utilisant notre approche sur les résultats issus de divers modules de classification.

9.1 Présentation du processus de traitement de notre approche

Pour aider l'utilisateur à interpréter ses résultats, nous avons expliqué tout le long de cette thèse notre approche. Dans cette section, nous présentons la chaîne de traitement qui permet de mettre en œuvre notre approche. Ainsi, il existe deux phases distinctes dans cette chaîne de traitement (cf. figure 9.1) :

1. La ***première phase*** est relative à la création d'un fichier de métadonnées contenant les critères généraux qui seront utilisés dans la phase d'interprétation, cette phase peut être *manuelle*, *semi-automatique* ou *automatique*. En effet, deux stratégies se sont imposées à nous lors de la recherche d'une bonne solution pour la création du fichier de métadonnées : *la première*, de loin la plus *improbable*, est d'inclure une partie de notre code dans celui des algorithmes utilisés pour la classification ; *la seconde* consistait à récupérer les métadonnées dans le fichier résultat fourni par l'algorithme de classification (ce qui suppose de notre part la connaissance de la structure des fichiers résultats de ces algorithmes). C'est cette dernière solution qui a retenu notre approbation car elle est de loin la plus *réaliste*, la plus *flexible* et la plus *facile* à mettre en œuvre. Cependant, il convient de rappeler que ces deux solutions sont réalisables avec notre approche.

2. La ***seconde phase*** est relative au processus d'interprétation basé sur l'utilisation du fichier de métadonnées obtenu à l'étape précédente. Cette phase, réalisée grâce à notre outil, est complètement *automatique*. Il s'agit, comme nous l'expliquons dans la section 3.1.4 pour le cadre supervisé, de trouver un moyen de représenter les critères généraux qui vont servir à calculer tous les autres critères généraux intervenant dans les divers scénarios d'interprétation. Pour implémenter notre approche, nous nous sommes limités à l'utilisation des critères d'interprétation qui utilisent la décomposition de l'inertie totale en inertie intra-classe et en inertie interclasse.

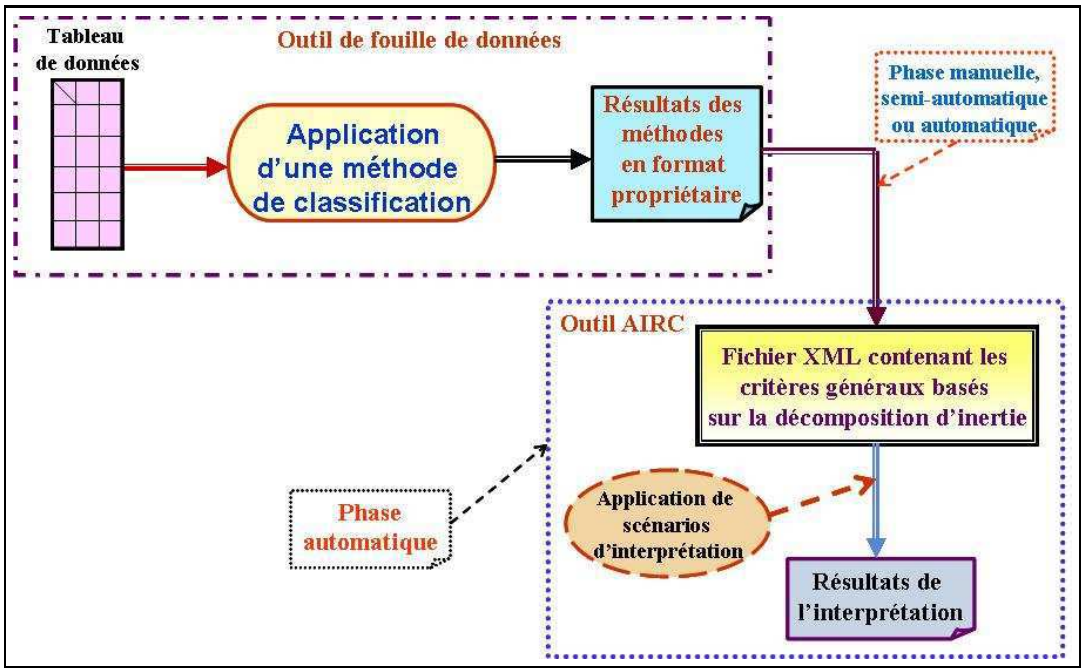

FIG. *9.1 - Processus de traitement de notre application*

9.2 Critères généraux permettant le calcul des critères d'interprétation

L'objectif de cette section est d'expliquer les critères généraux que nous utilisons pour interpréter les résultats des méthodes de classification. Comme nous l'expliquons dans la section 3.4, il existe plusieurs approches d'interprétation des résultats d'une méthode de classification. Dans le cadre de nos travaux, nous avons fait le choix de nous baser sur les interprétations utilisant les notions ***de décomposition d'inertie***. Mais rappelons que notre approche s'applique à tous types de critères généraux d'interprétation à partir du moment où ces critères peuvent être utilisés dans notre métamodèle. Nous verrons d'ailleurs à travers les exemples abordés dans ce chapitre que les résultats fournis par les méthodes de classification ne correspondent pas forcément aux critères d'interprétation que nous avons décidés d'appliquer dans le cadre de cette thèse. Il a fallu procéder à un prétraitement des résultats afin d'obtenir les données souhaitées (il s'agissait des valeurs des inerties interclasse, intra-classe et totale).

Comme nous utilisons la décomposition d'inertie pour interpréter nos résultats de classification en se restreignant aux variables, il faut savoir que la formule de base dans la décomposition d'inertie dans une partition est [Jambu, 1978] :

$$T = B + W \tag{9.1}$$

Où T est l'inertie totale, W l'inertie intra-classe et B l'inertie interclasse.

Dans les exemples qui vont suivre, nous utilisons deux types d'interprétation : *la recherche de variables discriminantes* lors d'un processus de classification et la *recherche de variables discriminantes dans la construction de chaque classe*. Rappelons que pour un résultat de classification à interpréter, nous ***utilisons notre ontologie de la classification*** afin de proposer à l'utilisateur les différents scénarios possibles pour un résultat et une méthode de classification donnés.

9.3 Présentation de notre outil : AIRC

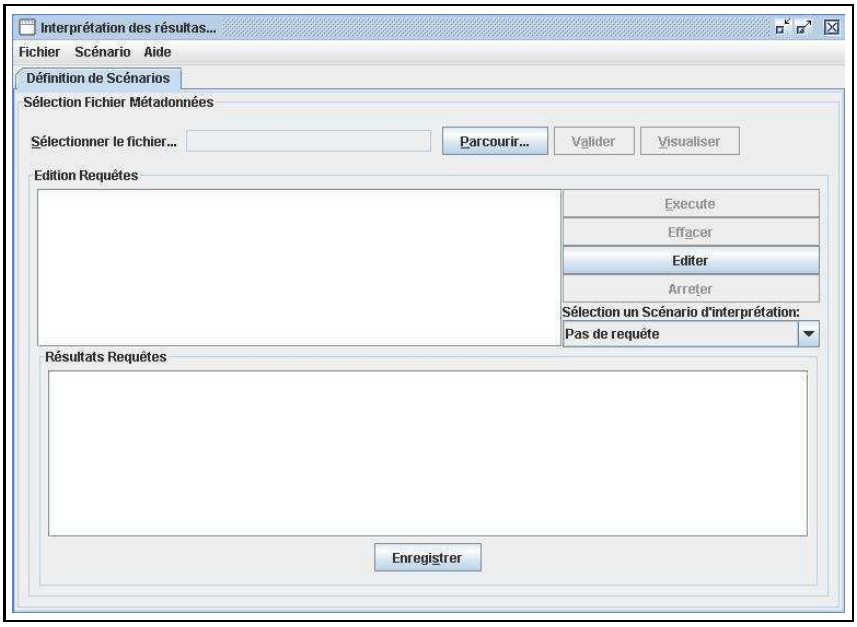

FIG. *9.2 - Présentation de notre outil*

Dans cette section, nous présentons les fonctions principales de notre outil AIRC (Aide à l'Interprétation des Résultats de Classification). Cet outil est composé de trois parties :

1. *la sélection du fichier de métadonnées* à interpréter ;
2. *l'édition des requêtes* XQUERY pour la définition des scénarios d'interprétation ;
3. *la visualisation des résultats* des scénarios d'interprétation (cf. figure 9.2).

La sélection du fichier de métadonnées se fait en cliquant sur le bouton *Parcourir*. Pour éviter de sélectionner des fichiers n'ayant pas l'extension *Xml*, nous avons mis en place un filtre pour les extensions de ce type. Ce qui permet de ne visualiser que les fichiers ayant l'extension *Xml*, ce qui ne signifie pour autant pas que les fichiers sélectionnés correspondent bien à notre modèle de fichier de métadonnées. Pour valider le fichier sélectionné, nous avons construit un parseur qui nous permet de vérifier la validité de notre fichier. Dans le cas échéant, un message d'erreur est généré.

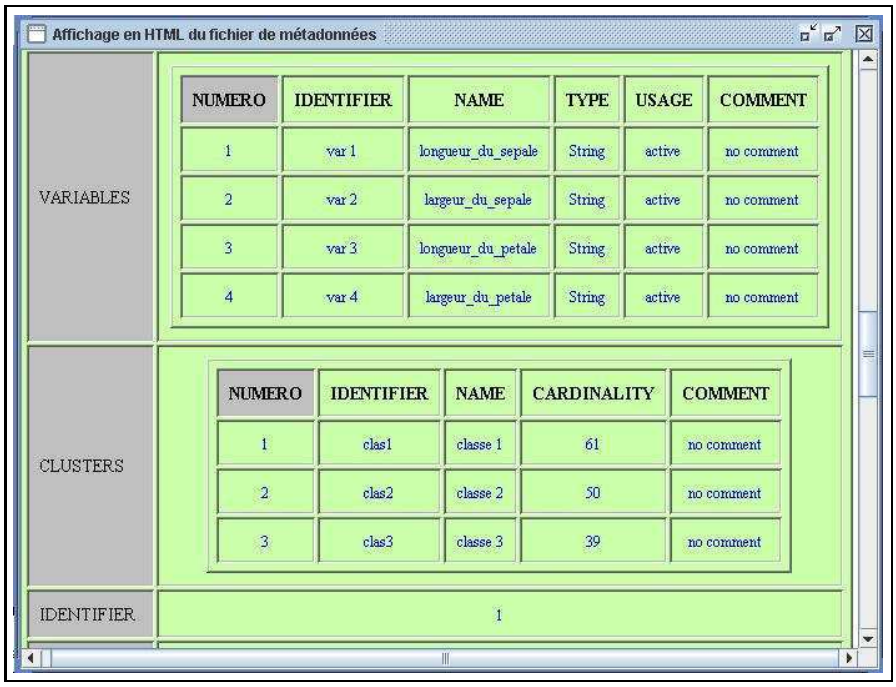

FIG. *9.3 - Visualisation du fichier de métadonnées*

Ensuite, il est possible de visualiser (en cliquant sur le bouton *Visualiser*) dans une nouvelle fenêtre le fichier de métadonnées sélectionné (cf. figure 9.3). Ceci permettrait à l'utilisateur de s'assurer de la conformité du fichier sélectionné en vue de l'interprétation.

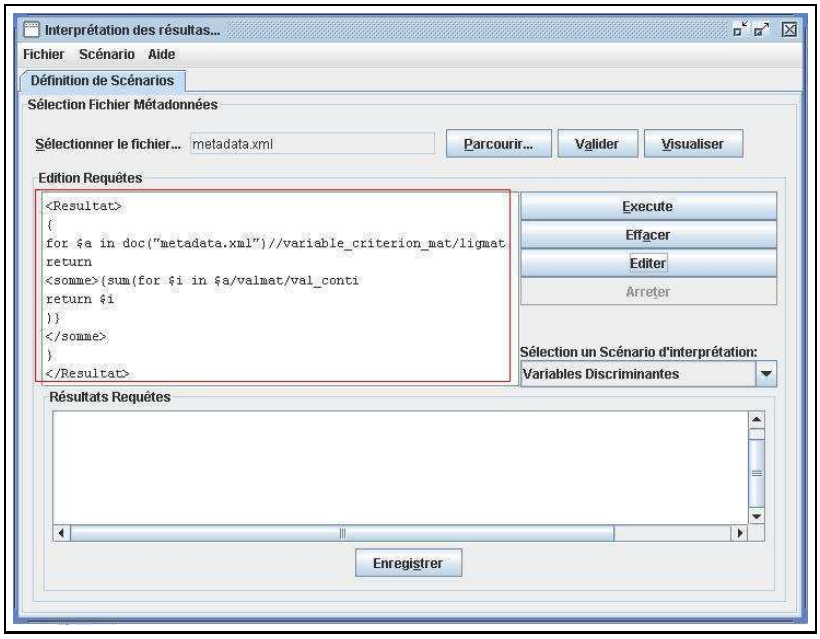

FIG. *9.4 - Édition des requêtes Xquery*

L'application des scénarios d'interprétation s'effectue dans le panneau suivant. Il est constitué d'un ensemble de boutons permettant, entre autres, d'éditer une requête et de l'exécuter (cf. figure 9.4). A travers cette figure, nous voyons qu'il suffit que l'utilisateur sélectionne le scénario d'interprétation désiré pour voir la requête correspondante dans la zone de texte. Pour modifier sa requête, il suffit de cliquer sur le bouton *Editer*. Ensuite, il aura loisir de faire les modifications qu'il souhaite et de les enregistrer ensuite (cf. figure 9.5).

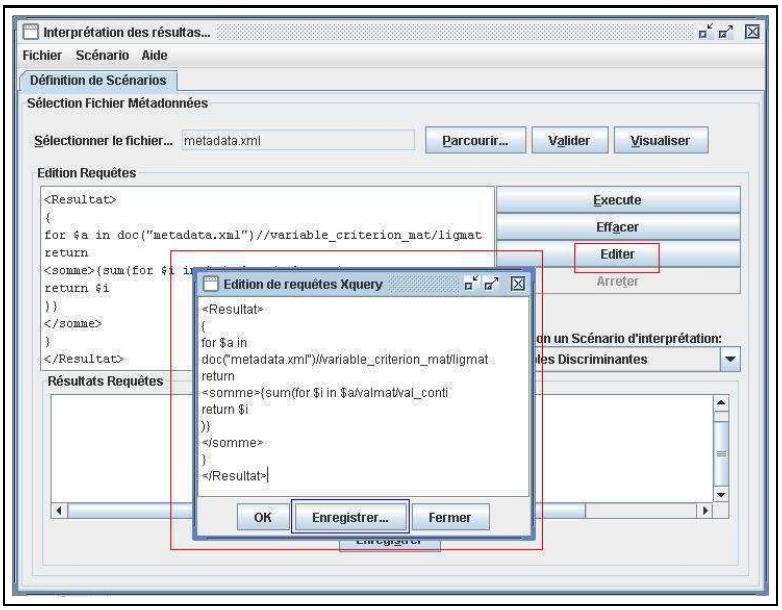

FIG. *9.5 - Fenêtre d'édition des requêtes Xquery*

Enfin, il est important d'apporter quelques précisions sur l'utilisation de notre outil. En effet, les *fichiers de métadonnées* à inclure pour l'interprétation sont des fichiers *Xml* avec une structure précise. Ce fichier est obtenu à l'aide de notre application qui analyse les résultats issus des méthodes de fouille de données. Il peut aussi être obtenu en incluant notre algorithme à l'intérieur des méthodes pour lesquelles nous voulons extraire des métadonnées.
Cette structure déjà définie dans au § 6.8.1, est reprise ici :

- la composante *Entête* contient les informations générales sur l'application traitée. Cette composante est composée d'éléments issus de la norme du Dublin Core ;
- la composante *Module* décrit les informations sur les modules de classification utilisés ainsi que les paramètres des méthodes considérées ;
- la composante *Fichier* décrit l'ensemble des fichiers utilisés dans le cadre de ces méthodes. Il s'agit par exemple des fichiers relatifs aux sources de données ou aux résultats des méthodes. Notre application permet de maintenir le lien avec ces différents fichiers ;
- la composante *Expérience* contient la description des critères généraux ainsi que leurs valeurs. La seule contrainte existante sur ces éléments est celle relative au fait que chaque critère doit avoir un *identifiant*, un *nom*, une *valeur* et un *espace de commentaire*.

9.4 Premier scénario d'interprétation

Soit R le *pouvoir discriminant moyen* des variables vis-à-vis d'un ensemble de classes, B^j, W^j et T^j, respectivement, les parts d'inertie de la variable j dans l'inertie inter-classe, intra-classe et totale, voici les différentes interprétations que nous pouvons avoir ([Jambu, 1978]) :

- *Pouvoir discriminant de la variable par rapport à une classe* : pour chaque variable j et chaque classe k, on définit :

$$COR(j,k) = \frac{B_k^j}{T^j} \tag{9.2}$$

  Avec : $\sum_{k=1}^{k} COR(j,k) = COR(j)$

- *Contribution relative de la variable j et de la classe k à l'inertie interclasse* :

$$CTR(j,k) = \frac{B_k^j}{B_k} \tag{9.3}$$

  Avec : $\sum_{k=1}^{k} CTR(j,k) = CTR(j)$. Cet indice est complémentaire à l'indice précédent : il permet de déterminer les variables qui caractérisent le mieux chaque classe.

- *Contribution des variables :* pour chaque variable j, on définit :

$$COR(j) = \frac{B^j}{T^j} \tag{9.4}$$

  Cet indice représente la part d'inertie de la variable j pris en compte dans la construction des classes. Il mesure ainsi le pouvoir discriminant de la variable j. Ainsi, si $COR(j) >$ (respectivement $<$)R, la variable j est plus (respectivement moins) discriminante que la moyenne puisque R représente le pouvoir discriminant « moyen » des variables.

- *Contribution relative à l'inertie interclasse et/ou intra-classe :* il s'agit de la contribution relative de la variable j à l'inertie interclasse et/ou intra-classe de la partition.

$$CTR(j) = \frac{B^j}{B} \text{ avec } \sum CTR(j) = 1. \tag{9.5}$$

Les deux derniers indices ont tendance à varier dans le même sens. Ceci est vrai en particulier dans le cas où la partition a des classes de volume analogue.
Une règle d'interprétation souvent utilisée est de voir la variation de ces deux derniers indices.

Règle 9.1. *Si $COR(j)$ est faible (respectivement fort) et $CTR(j)$ est fort (respectivement faible), alors la variable j, malgré une forte (respectivement faible) contribution à l'inertie de la partition, est peu (respectivement très) discriminante.*

Ce type de règle peut être utilisé dans notre application pour voir de manière automatique les variables qui répondent à ce type de contraintes. Pour que ces règles puissent être implémentées dans notre outil, il a fallu les traduire dans le langage de requêtes XQuery.

9.5 Second scénario d'interprétation

D'autres travaux reposant sur la décomposition de l'inertie pour interpréter les résultats de classification ont vu le jour. Il s'agit par exemple des récents travaux réalisés par M. Chavent dans [Chavent et al., 2006] et qui généralise certains critères proposés par G. Celeux [Celeux et al., 1989] pour une partition P d'un ensemble n de points x_s de R^p munis de poids p_s et implémentés par une classification dynamique. Ainsi, comme dans [Jambu, 1978], ils procèdent à la décomposition de l'inertie totale TSS en inertie intra-classe WSS et en inertie interclasse BSS dont la formule est la suivante :

$$TSS = WSS + BSS \tag{9.6}$$

Avec : $TSS = \sum_{s=1}^{n} d^2(x_s, G_i)$ où G_i est le prototype de la classe i
$WSS = \sum_{i=1}^{k} \sum_{s \in C_i} d^2(x_s, G_i)$
$BSS = \sum_{i=1}^{k} n_i d^2(G_i, G)$

Dans ce cadre, pour trouver la qualité d'une partition P, il suffit d'appliquer la formule suivante :

$$Q(P) = \frac{BSS}{TSS} = 1 - \frac{WSS}{TSS} \tag{9.7}$$

Avec cette formule, nous pouvons avoir les deux règles suivantes :

Règle 9.2. *Pour avoir la contribution de chaque variable j dans la construction de la partition P, on compare $Q_j(P)$ à $Q(P)$. Si pour une variable j, nous avons $Q_j(P) > \alpha\ Q(P)$ ($\alpha > 1$), alors cette variable fait partie des variables ayant un fort pouvoir discriminant.*

Règle 9.3. *Cette règle exprime la contribution de chaque variable j dans la construction de chaque classe k. Pour ce faire, nous comparons $Q_j^k(P)$ à $Q_j(P)$. Si pour une variable k, nous avons $Q_j^k(P) > \alpha\ Q_j(P)$ ($\alpha > 1$), alors cette variable fait partie des variables ayant un fort pouvoir discriminant dans la construction de la classe k.*

Ces deux règles implémentées dans notre outil, sont traduites en langage XQuery pour permettre leur utilisation. Ce sont ces règles que nous appliquons afin d'expliquer notre approche (sections 9.7.2 et 9.7.4). D'autres critères utilisant les notions d'inertie sont dans [De Carvalho et al., 2006].

9.6 Ajout et/ou modification d'un scénario d'interprétation

L'avantage de notre approche est qu'il est facile de modifier et d'ajouter un nouveau scénario d'interprétation. Dans cette section, nous présentons cette caractéristique de notre approche. Ainsi, ajouter un scénario d'interprétation utilisant la décomposition de l'inertie totale est très facile. En effet, l'utilisateur doit traduire son scénario d'interprétation en une requête XQuery qui fera intervenir les critères généraux contenus dans notre fichier de métadonnées. La seule contrainte dans ce cas de figure est la connaissance initiale que doit avoir l'utilisateur sur le langage de requêtes XQuery. De même, un utilisateur peut modifier un scénario d'interprétation déjà défini en ajustant par exemple le seuil suivant son gré ou en modifiant la formule de calcul utilisée dans le dit scénario.

D'un point de vue technique, notre implémentation se base sur l'utilisation de certains *patrons de conception* tels que *Strategie, Observateur, Adaptateur, etc.* qui permettent de rendre robuste et flexible notre application (ces *patrons de conception* sont spécifiés dans l'annexe A). Ainsi, il est possible de modifier plus facilement la base sur laquelle nous avons réalisé notre expérimentation (c'est-à-dire l'utilisation des critères généraux basés sur la décomposition de l'inertie).

9.7 Expérimentations

L'objectif de cette section est de présenter les résultats obtenus dans le logiciel WEKA ainsi que ceux issus des *méthodes de nuées dynamiques* sur le traitement de données numériques. Il s'agit aussi de présenter les résultats obtenus avec l'algorithme SCLUST sur la classification de *données symboliques*. Ensuite nous présentons l'outil AIRC et nous montrons sa spécificité par rapport aux précédentes méthodes. Mais avant, nous expliquons dans la section suivante les jeux de données utilisés dans le cadre de cette expérimentation.

9.7.1 Les jeux de données

Dans nos travaux, nous utilisons deux jeux de données :

1. Le premier est un jeu de données très connu dans le domaine de la fouille de données. Il s'agit des ***Iris*** qui sont constitués par 150 individus décrits par 4 variables continues qui sont : *longueur du sépale*, *largeur du sépale*, *longueur du pétale* et *largeur du pétale*. L'objectif de cette classification est de voir si on est capable de retrouver les trois classes naturelles des Iris à savoir les *Iris Setosa*, les *Iris Versicolor* et les *Iris Virginica*. Ce jeu de données est celui utilisé dans l'expérimentation des méthodes de nuées dynamiques et du logiciel WEKA.
2. Le second est un jeu de ***données symboliques*** qui décrit les données récueillies dans des stations météorologiques des régions de la Chine. Ces individus sont au nombre de 60 objets symboliques décrits par 12 variables de type *intervalle* qui représentent les mois de l'année. Le nombre de classes obtenues est de 6.

Notons que nous utilisons ces deux jeux de données dans l'outil AIRC afin de comparer l'apport de notre approche par rapport aux méthodes précédemment citées.

9.7.2 Expérimentation dans les Méthodes de Nuées Dynamiques

Comme nous le voyons dans le listing montré dans la figure 9.6, il n'est pas possible de réaliser une interprétation automatique de ces résultats. Pour effectuer cette interprétation, nous avons procédé à un prétraitement des résultats contenus dans ce listing afin de l'adapter à notre métamodèle d'interprétation.

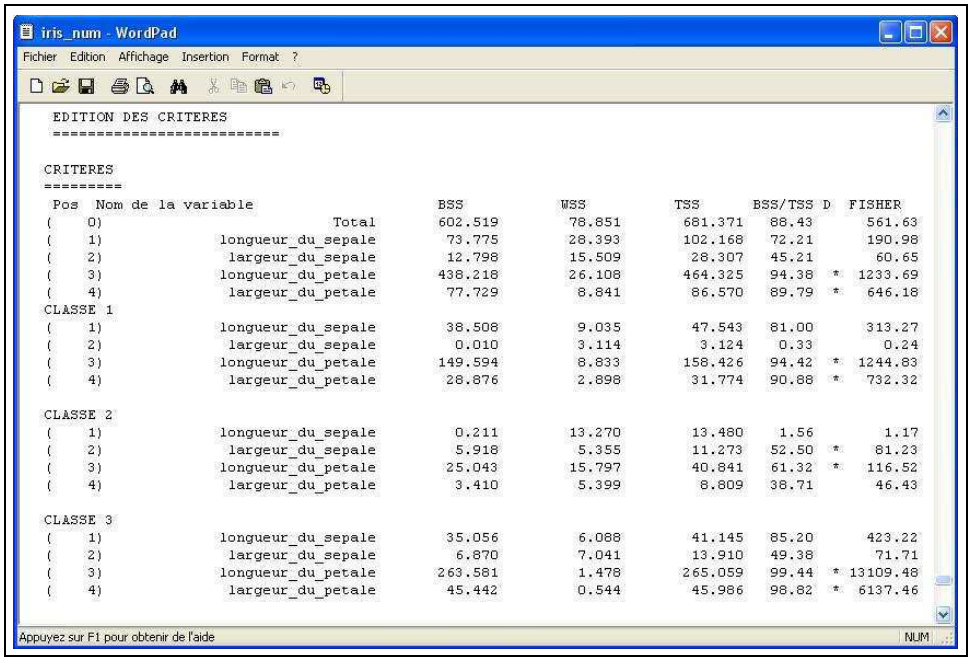

FIG. *9.6 - Listing Nuées Dynamiques sur les Iris*

Grâce à ce prétraitement, nous pouvons calculer, pour chaque variable, sa contribution à l'inertie intra-classe W_k^j, à l'inertie interclasse B_k^j et à l'inertie totale T_k^j(cf. section 3.4). Il s'agit de voir le pouvoir discriminant de chaque variable dans la construction de chaque classe de la partition (règle 9.3). Voici la traduction en langage XQUERY de la règle 9.3 :

```
<Resultat>
{
let $n :=doc("metadataKmeans.xml")//ComposantExperience/Variables/Variable
let $k :=doc("metadataKmeans.xml")//ComposantExperience/Classes/Classe
for $b in doc("metadataKmeans.xml")//ComposantExperience/CRITERIA/criteria
let $bss := for $val in doc("metadataKmeans.xml")//ClasseVariableMatrix/LigMat/value
 Where //ComposantExperience/CRITERIA/criteria/="BSS"
let $tss := $val in doc("metadataKmeans.xml")//ClasseVariableMatrix/LigMat/value
 Where //ComposantExperience/CRITERIA/criteria/="TSS"
let $B :=sum(for $i in $bss/ContinuValue)
let $T :=sum(for $j in $tss/ContinuValue)
where $bss/ContinuValue div $tss/ContinuValue >$B/ContinuValue div $T/ContinuValue
AND ($k/Identifier =$n///ComposantExperience/Classes/Classe/Identifier)
return  $n/Name
}
</Resultat>
```

Le résultat de cette requête est montré dans la figure 9.7. D'autres critères d'interprétation peuvent être calculés à partir du moment où ils font intervenir, comme dans cet exemple, les notions d'inertie. Cet exemple est aussi valable pour d'autres critères généraux non basés sur les inerties. En effet, le plus difficile dans notre approche est d'identifier les règles à appliquer.

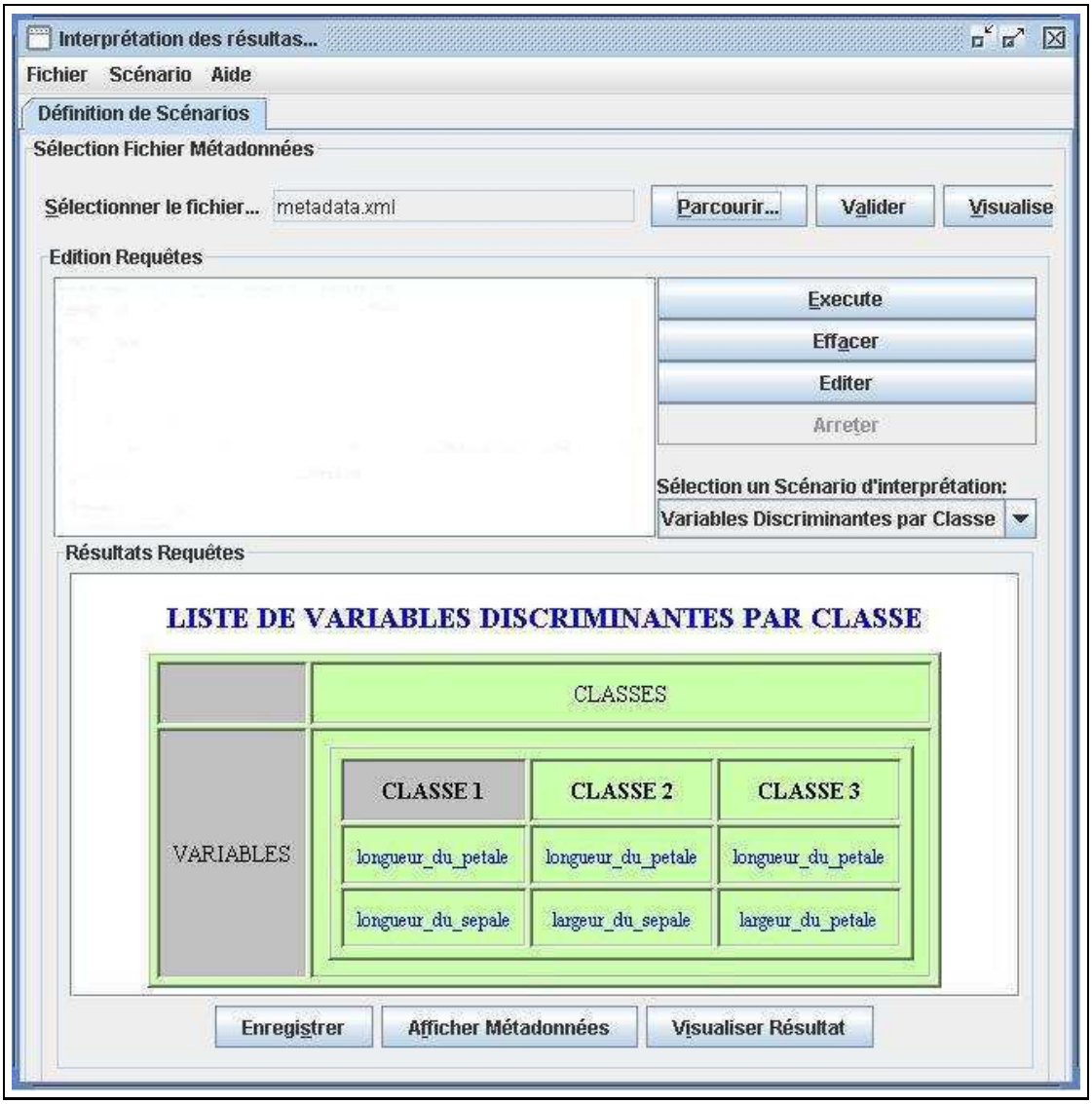

FIG. *9.7 - Contribution à l'inertie interclasse par variable (AIRC)*

9.7.3 Expérimentation dans Weka

Description

Développé par des groupes de chercheurs[1] de l'université de Waikato (Nouvelle Zélande), Weka est un *logiciel libre* entièrement écrit en langage java. Concrètement, il s'agit d'un outil implémentant un ensemble d'algorithmes issus des méthodes d'apprentissage, de reconnaissance des formes et de la fouille de données. Il est composé de quatre modules[2] accessibles au lancement (cf. figure 9.8) :

[1] Ces chercheurs sont issus pour la plupart des domaines de la fouille de données, de l'apprentissage et de la reconnaissance des formes

[2] La version décrite ici est la version 3.4

FIG. *9.8 - Fenêtre de lancement de Weka*

- Le menu SIMPLECLI qui permet d'utiliser des modules du logiciel en ligne de commande. La prise en main de ce logiciel par ce menu semble complexe vu les nombreux paramètres à fournir pour l'utilisation des classes implémentées.
- Le menu EXPLORER qui permet d'analyser un jeu de données à travers une interface graphique permettant de paramétrer et de visualiser les résultats. Ainsi, ce menu permet de lancer une méthode à partir d'un fichier ARFF[3]. Cette commande ne permet de traiter qu'une méthode de fouille sur un jeu de données à la fois. Nous verrons dans la section 9.7 que les critères d'interprétation obtenus dans le cadre des méthodes supervisées sont de loin les plus pertinents que dans le cadre non supervisé. Ceci s'explique pour la simple raison que le logiciel WEKA est fortement dédié à la classification supervisée. Dans nos travaux, nous montrons que nous pouvons utiliser une nouvelle approche permettant de rendre ces outils plus flexibles et plus intuitifs dans le processus d'interprétation.
- Le menu EXPERIMENTER qui permet de configurer des expériences complètes et complexes par l'analyse de plusieurs jeux de données via différentes méthodes de traitement. Il s'agit de comparer des méthodes traitement issues de la classification supervisée ou des stratégies d'utilisation de ces méthodes. Les résultats de ces comparaisons sont enregistrés dans des fichiers textes au format *csv*. Néanmoins, la difficulté de ce type de comparaison réside dans le fait que les critères d'interprétation utilisés ne sont pas forcément les plus pertinents pour réaliser la comparaison. De plus, il n'est pas possible d'ajouter de nouveaux critères de validation ou d'interprétation.

[3]ARFF est le format d'entrée des données traitées sous Weka

- Le menu KnowledgeFlow dont l'activation permet la réalisation d'un enchaînement d'analyses à travers une interface graphique. Il s'agit de créer les liens entre les entrées et les sorties de différents modules implémentés dans le logiciel. Il s'agit de créer une chaîne de traitement complète d'analyse d'un jeu de données à travers l'utilisation de diverses méthodes.

Dans nos travaux, nous avons expérimenté notre application sur les modules *Explorer* et *Experimenter* afin de montrer l'intérêt de l'utilisation de notre approche. En effet, pour comparer par exemple plusieurs algorithmes sur divers jeux de données, Weka utilise l'ensemble les critères montrés dans le tableau 9.1. Ce qui pose un problème évident de compréhension de la part des utilisateurs car ces critères sont difficiles à comprendre par ces utilisateurs. De plus, suivant les algorithmes utilisés, certaines valeurs de ces

Key_Dataset	Key_Run	Key_Fold
Key_Scheme	Key_Scheme_options	Date_time
Number_of_training_instances	Number_incorrect	Number_correct
Number_of_testing_instances	Number_unclassified	Percent_correct
SF_mean_scheme_entropy	Percent_unclassified	Kappa_statistic
Mean_absolute_error	SF_entropy_gain	IR_precision
Root_relative_squared_error	SF_prior_entropy	SF_scheme_entropy
Root_mean_squared_error	Time_training	Percent_incorrect
SF_mean_entropy_gain	KB_information	KB_mean_information
KB_relative_information	True_positive_rate	Num_true_positives
False_positive_rate	Num_false_positives	True_negative_rate
Num_true_negatives	False_negative_rate	Num_false_negatives
Relative_absolute_error	IR_recall	F_measure
SF_mean_prior_entropy	Time_testing	Summary

Tab. *9.1 - Tableau des critères utilisés dans le module* Experimenter *de Weka*

critères sont nulles car ne pouvant être calculées. C'est pourquoi nous choisissons de ***ne définir aucun critère d'interprétation à l'avance mais de pouvoir créer un cadre permettant de les définir tous***. Nous verrons, à travers les différents exemples abordés dans ce chapitre que ceci est encore plus vrai dans le cadre des méthodes non supervisées. En effet, il n'y pas de normalisation au niveau des sorties de ces méthodes, ce qui peut désarçonner un *utilisateur* cherchant à interpréter ses résultats. De même, une fois que ces résultats sont affichés, il n'est plus possible de faire des traitements ultérieurs sur ceux-ci. Pour essayer de faire des modifications sur le calcul des critères d'interprétation ou tout simplement en ajouter de nouveaux, l'utilisateur se trouve dans l'obligation de faire ces modifications dans le code du programme. Ce qui n'est pas chose aisée, loin s'en faut. L'objectif de notre outil est d'éviter aux utilisateurs ces écueils rencontrés dans la majorité des outils de fouille de données d'une part. D'autre part, il

s'agit de leur donner des résultats ayant une structure identique et ce, quelle que soit la méthode de classification utilisée.

Nous avons expliqué que pour évaluer les méthodes dites supervisées, plusieurs critères d'interprétation sont utilisés dans WEKA. Ces critères permettent à des utilisateurs avertis de valider et d'interpréter les résultats de leurs méthodes de classification supervisée. Avec notre outil AIRC, il est possible de conserver ces informations dans l'élément appelé *Critères Globaux* qui décrit l'ensemble des critères relatifs aux méthodes dans les divers outils de fouille de données.

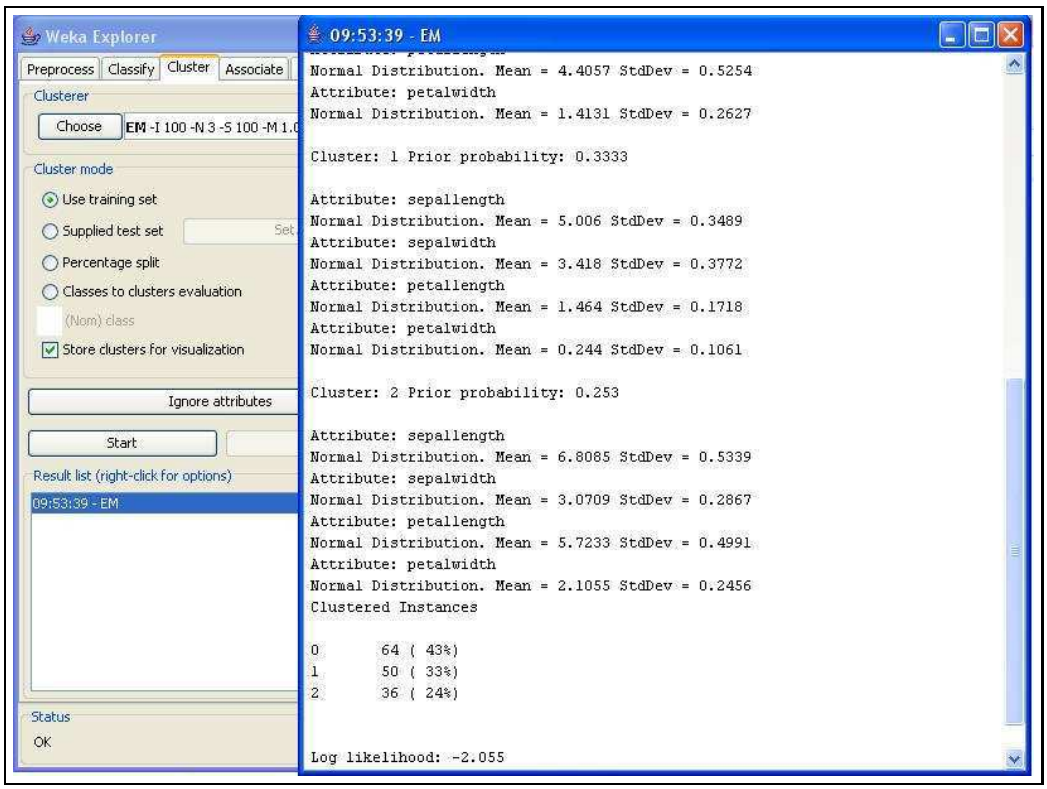

FIG. *9.9 - Résultat de l'application de EM sur les Iris (*WEKA*)*

Par contre, dans le cadre des méthodes non supervisées, WEKA semble être en retrait dans la structuration uniforme des sorties des différentes méthodes. Ce qui pose un problème évident d'automatisation du recueil de ces données. En effet, comme nous allons le voir dans les figures 9.9 et 9.10, les structures des résultats fournis par Weka sont différents d'une méthode à l'autre. Voici un exemple illustrant ce défaut de structuration au niveau des méthodes non supervisées. Pour le calcul de la standard déviation, les deux algorithmes (KMeans et EM) ne procèdent pas au même type de calcul. De ce fait, nous avons été obligés de normaliser les valeurs des standards déviation de la méthode KMeans en *multipliant les valeurs données dans le listing par le carré du rapport* $n-1/n$ *où* n *est l'effectif de chaque classe.* Ce qui nous a rendu la tâche ardue dans l'extraction des informations que nous jugions pertinentes dans le cadre d'un processus d'interprétation.

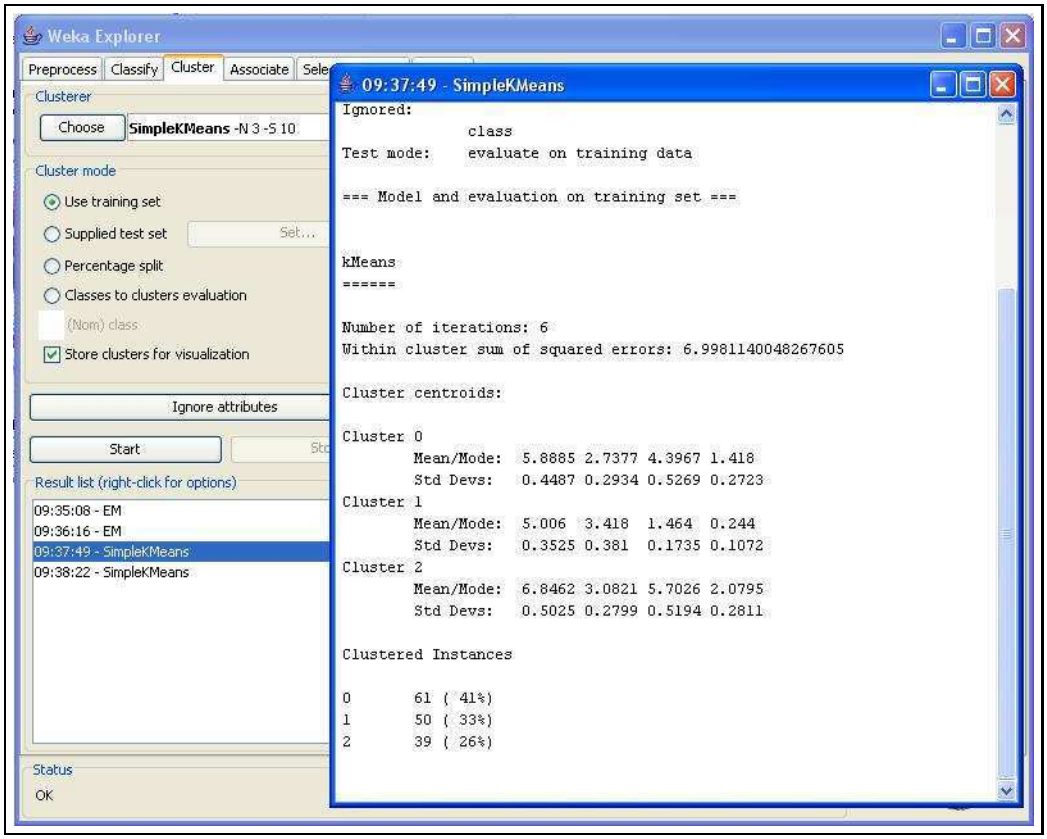

FIG. *9.10 - Résultat de l'application de KMeans sur les Iris (*WEKA*)*

Par ailleurs, pour interpréter ces résultats en utilisant les notions d'inertie, nous les avons transformés pour obtenir les critères d'inertie intra-classe, interclasse et totale. Pour l'inertie intra-classe, il a fallu normaliser les *standards déviations* obtenus par les méthodes de WEKA afin de calculer l'inertie intra-classe (W).

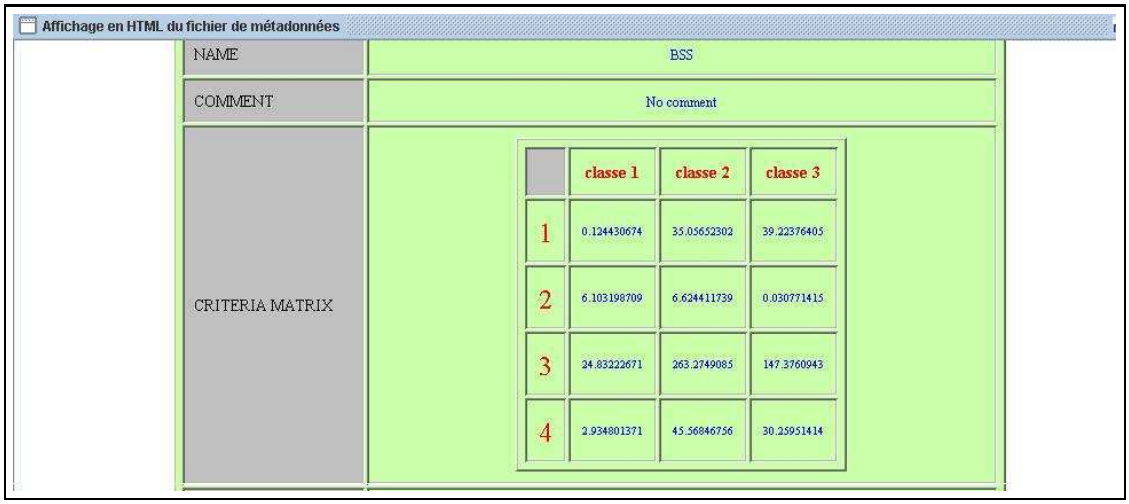

FIG. *9.11 - Valeurs des critères de l'inertie interclasse pour chaque variable (AIRC)*

La formule de transformation utilisée est la suivante : pour chaque variable j et chaque classe k :

$$W_k^j = n_k(S_k^j)^2 \text{ avec } (S_k^j)$$

les *standard déviations* définis dans ces méthodes de l'outil Weka et n_k les effectifs de la classe k.
En ce qui concerne le critère interclasse, nous utilisons la formule suivante :

$$B_k^j = n_k(g_k^j - g^j)^2$$

où g_k^j est la moyenne de la variable j dans la classe k et $g^j = \frac{\sum_{k=1}^{n} n_k * g_k^j}{n}$ représente la moyenne par variable j. Ces valeurs sont obtenues en utilisant les variables *Mean/Mode* des listings précédents. Après ce prétraitement, nous obtenons le fichier de métadonnées (figures 9.11, 9.12 et 9.13) contenant les critères que nous venons de décrire. C'est à partir de ce fichier que les requêtes XQUERY sont appliquées pour obtenir les interprétations souhaitées.

IDENTIFIER	2
NAME	TSS
COMMENT	No comment
CRITERIA MATRIX	

	classe 1	classe 2	classe 3
1	12.20433207	41.14507929	48.81900153
2	11.2682123	13.73730074	3.0078438
3	41.48964332	264.7499188	157.627596
4	7.383638783	46.13156772	33.26216811

FIG. *9.12 - Valeurs des critères de l'inertie totale pour chaque variable (AIRC)*

A travers ces deux exemples, nous constatons qu'il est impossible de réutiliser ces résultats pour en faire une interprétation de manière automatique. En effet, ces listings contiennent des informations qui ne peuvent généralement pas aider l'utilisateur *lambda* dans la signification des classes qu'il a construites. C'est ce que nous tentons de lui apporter à travers notre outil d'aide à l'interprétation.

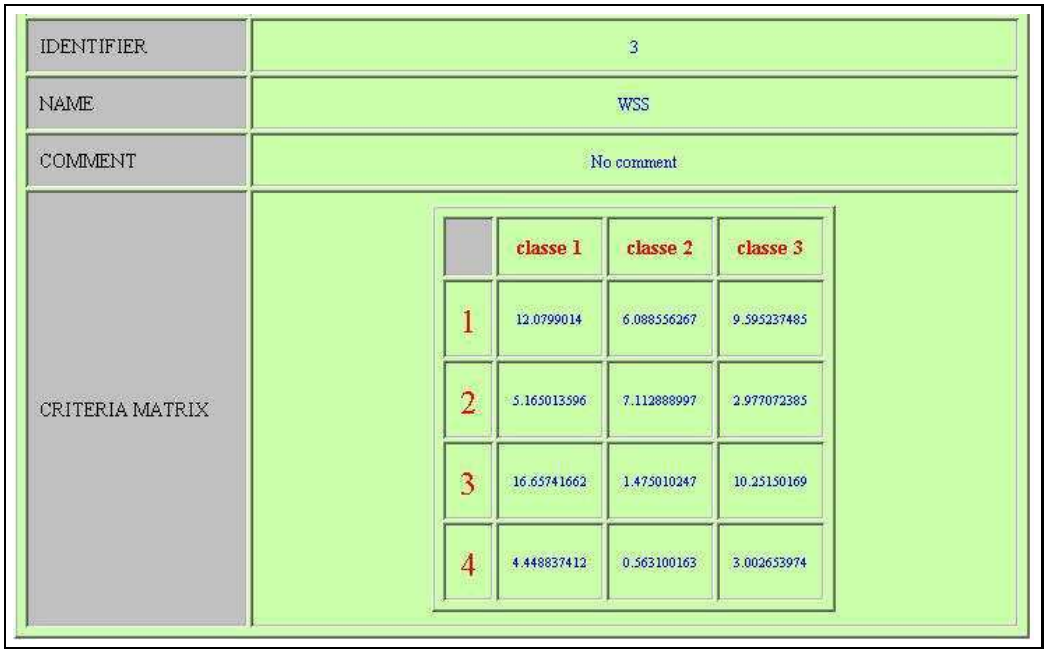

FIG. *9.13 - Valeurs des critères de l'inertie intra-classe pour chaque variable (AIRC)*

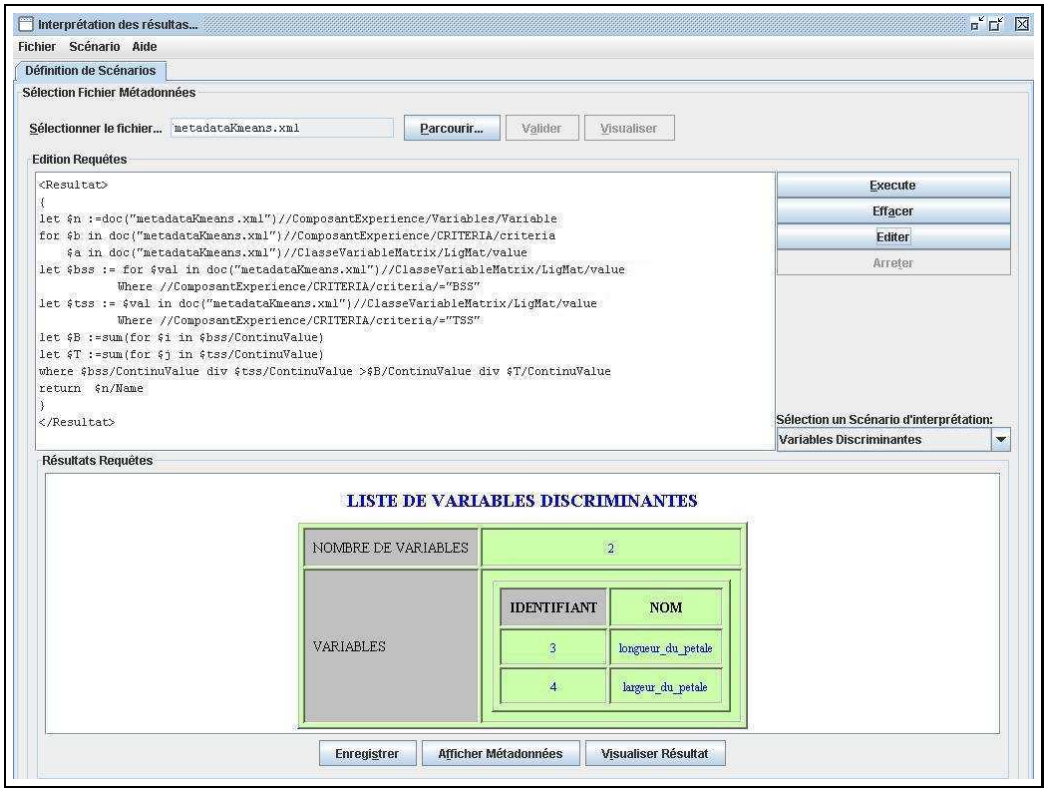

FIG. *9.14 - Interprétation : variables discriminantes (AIRC)*

L'utilisation du scénario qui consiste à rechercher les variables les plus discriminantes dans l'obtention de la partition donne les résultats montrés dans la figure 9.14.

En traduisant la règle 9.2 dans le langage XQUERY, nous obtenons le résultat suivant :

```
<Resultat>
{
let $n :=doc("metadataKmeans.xml")//ComposantExperience/Variables/Variable
for $b in doc("metadataKmeans.xml")//ComposantExperience/CRITERIA/criteria
    $a in doc("metadataKmeans.xml")//ClasseVariableMatrix/LigMat/value
let $bss := for $val in doc("metadataKmeans.xml")//ClasseVariableMatrix/LigMat/value
 Where //ComposantExperience/CRITERIA/criteria/="BSS"
let $tss := $val in doc("metadataKmeans.xml")//ClasseVariableMatrix/LigMat/value
 Where //ComposantExperience/CRITERIA/criteria/="TSS"
let $B :=sum(for $i in $bss/ContinuValue)
let $T :=sum(for $j in $tss/ContinuValue)
where $bss/ContinuValue div $tss/ContinuValue >$B/ContinuValue div $T/ContinuValue
return  $n/Name
}
</Resultat>
```

9.7.4 Expérimentation dans Sclust

Le module Sclust [Chavent et al., 2006] est dédié à la classification dynamique basée sur des prototypes qui sont les représentants des classes. Le critère de classification optimisé est basé sur la somme des proximités entre les individus et les prototypes des classes. Dans le cas de l'utilisation de ce module dans le cadre des objets symboliques, l'algorithme prend en entrée un ensemble de données symboliques décrites par des variables catégoriques multi-valuées, les variables intervalles et modales incluant les données manquantes. Les sorties contiennent les partitions, la valeur des critères, le nombre d'itérations avant convergence, les statistiques, la table de classification avec la liste des objets symboliques appartenant aux différentes classes et la description des classes.

SCLUST est un algorithme, dit de nuées dynamiques [Celeux et al., 1989], permettant de partitionner un ensemble de données en un nombre k (prédéfini) de classes homogènes. La principale différence de cet algorithme avec celui de *K-means* est que, dans SClust, les objets peuvent être décrits par plusieurs variables symboliques (variables intervalles, modales, etc.). En appliquant cet algorithme sur les données décrivant les données météorologiques de 60 régions de Chine en 1988, on obtient les résultats décrits dans la figure 9.15. Il convient de rappeler que pour la plupart des algorithmes basés sur la classification d'objets symboliques, comme SCLUST, les valeurs de l'inertie interclasse B ne sont pas données. Il s'agit de déduire ces valeurs à partir de la formule 9.1.

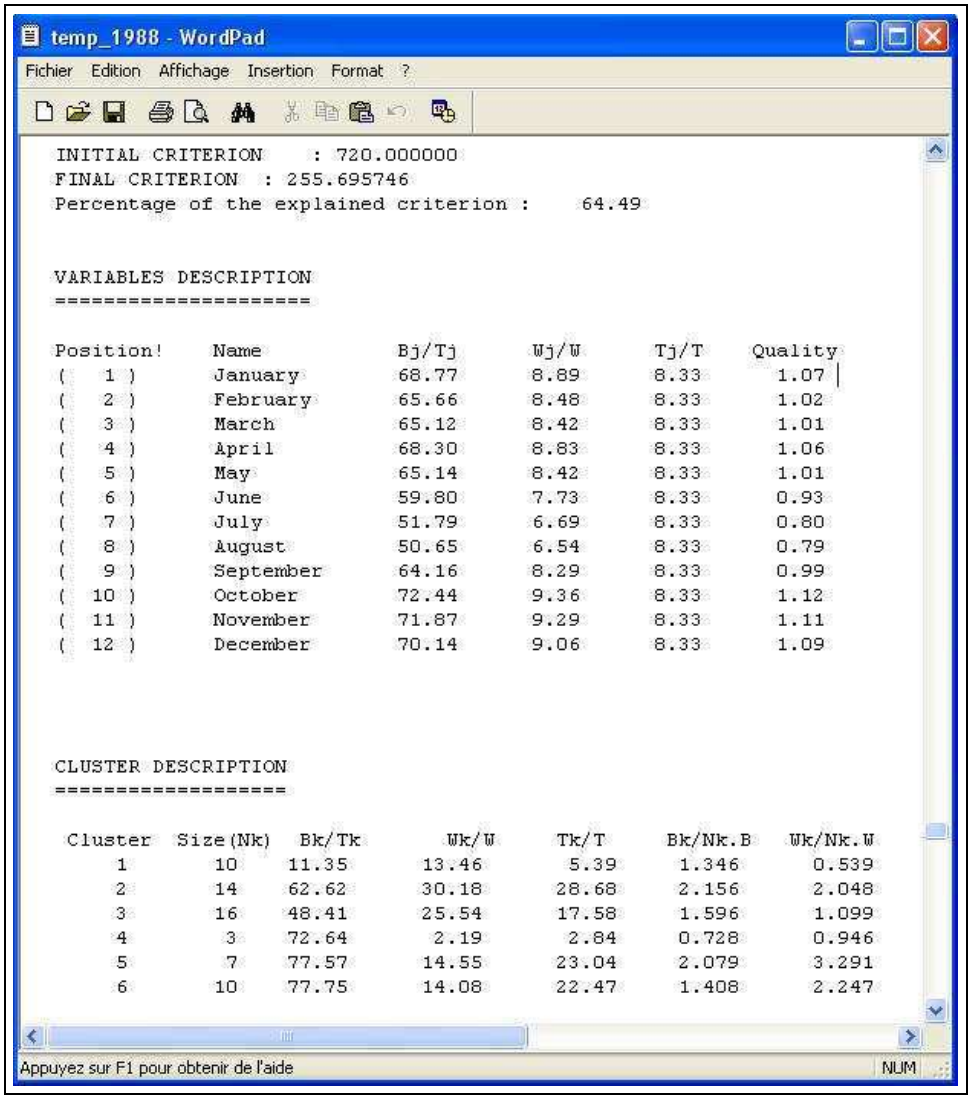

FIG. *9.15 - Listing de la méthode Sclust sur les données de température (tiré de* SCLUST*)*

A travers ce listing, on imagine aisément les difficultés rencontrées par les utilisateurs à la recherche de l'information significative leur permettant de tirer profit de leurs résultats. Une façon d'interpréter ce type de résultat implémenté dans notre outil est de rechercher les variables ayant une forte contribution dans la séparation des classes de la partition. Autrement dit nous cherchons les variables dont le $\frac{B^j}{B}$ est supérieur à $\frac{B^j}{T^j}$ (B étant l'inertie interclasse totale, B^j la contribution de la variable j à l'inertie interclasse et T^j étant la contribution de la variable j à l'inertie totale). La règle utilisée ici est la même que celle évoquée dans la règle 9.3. En transformant cette règle en une requête XQUERY, on obtient les résultats donnés dans la figure 9.16. Bien évidemment le seuil (α) choisi dans notre cas peut être modifié à travers notre outil. Enfin, pour définir ce qu'est une variable discriminante, ***l'utilisateur peut choisir sa propre règle*** en utilisant les requêtes XQUERY adéquates à sa demande.

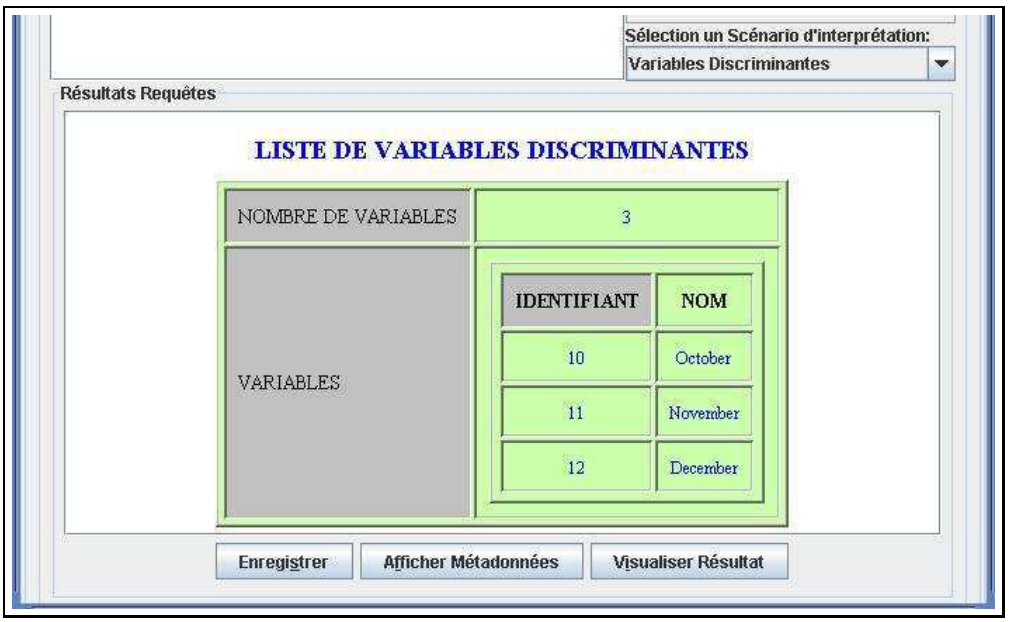

FIG. *9.16 - Variables discriminantes suivant leur contribution à l'inertie interclasse (AIRC)*

9.7.5 Expérimentation dans AIRC

Dans les sections précédentes nous avons expliqué que l'avantage de notre outil est qu'il permet de faire des traitements en *offline* sur les résultats issus des méthodes de classification. Pour ce faire, nous avons mis en place une application permettant de gérer les traitements de requêtes XQUERY. Ainsi, comme nous l'expliquons dans les § 9.7.4, § 9.7.2 et § 9.7.3, nous donnons la possibilité aux utilisateurs de comprendre les résultats qu'ils ont à leur disposition. Ceci est facilité par la mise en place de scénarios prédéfinis tels que la recherche de variables discriminantes (cf. figure 9.14). Ces scénarios sont proposés à l'utilisateur grâce à notre ontologie du domaine de la classification.

Par exemple, dans le cas de l'utilisation des données *Iris*, le scénario consistant à rechercher les variables les plus discriminantes donne les mêmes résultats pour chacune des méthodes utilisées dans ce chapitre. C'est-à-dire, nous avons toujours les mêmes variables ayant un plus grand pouvoir discriminant : *longueur du pétale* et *largeur du pétale*. Notre interface offre des métadonnées correspondantes à l'interprétation en cours. Ceci est réalisé grâce au bouton *Afficher Métadonnées*.

Il convient de remarquer ici que nous n'avons donné que deux scénarios d'interprétation parmi beaucoup d'autres. En effet, ***l'objectif de cette section n'est pas de lister tous les scénarios d'interprétation avec leurs résultats mais de montrer qu'il existe un cadre dans lequel ces scénarios peuvent être exécutés.*** De plus, grâce à l'ontologie de la classification expliquée au Chapitre 8, nous pouvons définir avec une sémantique claire un ensemble de scénarios d'interprétation couramment utilisés suivant les méthodes et les outils de classification.

9.8 Synthèse du Chapitre 9

L'objectif de ce chapitre était de montrer, à travers des exemples, les différences qui existent entre notre approche et certains outils dédiés aux méthodes de fouille de données et de la classification en particulier. A travers les sections de ce chapitre, nous avons expliqué les insuffisances des outils tels que *Sclust* et *Weka*, qui n'offrent pas de cadre satisfaisant pour l'aide à l'interprétation des résultats issus des méthodes de classification. Notamment parce que ces méthodes ne fournissent pas de résultats structurés permettant de procéder à une automatisation de leur traitement. Or, à l'heure où les efforts tendent vers des données de plus en plus structurées, il semblait nécessaire d'offrir un cadre unique de représentation des résultats des méthodes de classification.

C'est ce que nous proposons à travers notre approche et l'outil développé dans ce sens. Pour appuyer nos propos, nous avons fait une expérimentation comparative entre les informations fournies par certains outils et le nôtre. Il s'avère qu'il est plus facile de comprendre les résultats des méthodes de classification avec notre approche. A travers ces exemples, nous avons pu mettre en évidence l'utilité d'avoir une approche générique ne tenant pas compte de la structure d'une seule méthode mais susceptible d'être utilisée par tous types de méthodes de classification.

Conclusion de la partie III

Dans cette partie, il s'agissait d'expliquer nos contributions dans le domaine de l'aide à l'interprétation des résultats de classification. Cette partie a été l'occasion pour nous d'expliquer notre approche basée sur l'utilisation d'un métamodèle d'interprétation des résultats de classification dans le Chapitre 6. Ensuite, le Chapitre 7 fut consacré à l'explication de notre architecture destinée à exploiter notre métamodèle d'interprétation. Puis dans le Chapitre 8, il s'agissait d'expliquer les concepts, les relations et les axiomes de notre ontologie dédiée à la classification. Enfin, dans le Chapitre 9, nous montrons, à travers des exemples concrets, l'utilité de notre approche comparativement à d'autres méthodes de classification implémentées dans divers outils.

Rappelons que l'objectif de nos travaux est d'aider les utilisateurs à mieux interpréter leurs résultats de classification. En effet, à travers nos différentes études bibliographiques, nous avons constaté qu'il était très difficile pour un utilisateur, même expert, de comprendre les listings fournis en résultats des méthodes de classification. Et par la même occasion d'interpréter les résultats dont il dispose. Pour pallier ces insuffisances, nous avons proposé une nouvelle ***approche basée sur l'utilisation de métadonnées permettant de structurer les résultats des méthodes de classification automatique***.

Conclusions Générales et Perspectives

Synthèse de nos travaux

Il n'est nullement besoin de rappeler aujourd'hui que, vue la masse de données dont disposent les entreprises, le domaine de l'Extraction de Connaissances à partir de Données (ECD) connaît un succès retentissant. Ceci s'explique, en grande partie, par le fait qu'il semble impossible, pour les « décideurs » des entreprises de procéder à un traitement manuel de leurs données avant de prendre une décision de marketing par exemple. En effet, la masse de données est telle qu'un traitement de celle-ci semble systématiquement voué à l'échec. Le cas échéant, les informations tirées de ces masses de données deviennent très rapidement obsolètes vu le temps qui y sera consacré. C'est ce qui explique, en partie, le besoin pour ces entreprises de se tourner vers des techniques qui permettraient de fournir des informations pertinentes et dans des délais acceptables.

Et comme nous l'expliquons au Chapitre 3, le processus d'ECD se décompose en trois phases principales : le *prétraitement des données* qui permet d'avoir des données utilisables par la machine, ensuite nous avons la phase de *traitement des données* (appelée *fouille de données*) dans laquelle nous appliquons les techniques de fouille de données adaptées aux données à traiter et enfin la *phase d'interprétation des résultats* de ces méthodes. Cette dernière phase consiste à mettre sur des résultats numériques ou symboliques de l'information sémantique correspondant aux besoins de l'utilisateur. Or, à travers nos études bibliographiques et nos différents entretiens avec des experts du domaine de la fouille de données, il ressort qu'il n'existe pas de structure standard de sortie pour ces méthodes. En effet, les listings proposés par ces méthodes sont dépendants des outils dans lesquels elles sont implémentées. Ce qui pose problème pour un utilisateur n'étant pas familier à l'outil et/ou à la méthode utilisée. Dans la plupart des cas, on fournit à l'utilisateur des représentations graphiques de ces résultats et des listings contenant un ensemble de critères difficilement compréhensibles.

Dans le § 3.1.3, nous avons abordé les différentes techniques de fouille de données utilisées dans le cadre de l'extraction de connaissances. Pour chacune de ces techniques, nous avons expliqué leur rôle ainsi que les critiques qui leur sont faites. Ainsi, nos travaux répondent à un but simple : ***aider les utilisateurs à interpréter automatiquement*** les résultats issus des méthodes de fouille de données en général et de la ***classification automatique*** en particulier. En effet, contrairement aux méthodes dites *supervisées*, les résultats des méthodes *non supervisées* donnent des résultats difficilement interprétables par les utilisateurs. Ceci est dû au fait que les listings fournis par les outils qui implémentent ces méthodes sont difficilement exploitables. Il n'est donc pas possible de traiter automatiquement ces résultats. De plus, les multitudes de critères utilisés dans la validation et l'interprétation ne facilitent pas non plus cette tâche d'interprétation.

En partant de ces constats, nous avons mis en place une nouvelle approche qui consiste à utiliser les métadonnées comme éléments de capitalisation de la connaissance au cours du processus de classification. Cette nouvelle approche a abouti à la création d'un *métamodèle d'interprétation des résultats de classification* abordé au Chapitre 6. L'objectif de ce métamodèle est de permettre la mise en place d'une approche générique modélisant les données utiles au processus d'interprétation de tous types de méthodes de classification. Cette métamodélisation a été rendue possible grâce à nos études bibliographiques sur les différentes techniques d'interprétation (cf. section 3.4) et les interactions avec les experts du domaine de la classification. Cette métamodélisation tient compte de trois axes d'interprétation que nous avons recensés : *l'interprétation basée sur les individus*, *l'interprétation basée sur les variables* et *l'interprétation basée sur la structure classificatoire*. A partir de cette décomposition, nous avons montré que notre approche pouvait s'appliquer à tous types de méthodes de classification. Et comme nous l'expliquons en introduction du Chapitre 6, nous n'avons pas cherché à créer de nouveaux critères d'interprétation. Il s'agissait de trouver une approche permettant de généraliser les critères déjà existants. Cette généralisation a été rendue possible grâce à la mise en place de notre métamodèle d'interprétation.

Pour exploiter notre métamodèle, nous avons défini une architecture d'exploitation de ce dernier. En effet, la création d'une architecture permet de fixer le cadre dans lequel notre métamodèle peut être exploité et sous quelles contraintes. La mise en place de cette architecture a nécessité de faire des choix tenant compte de la spécificité de notre problématique et des solutions que nous avions proposées. Ceci a abouti à mettre en œuvre une architecture logicielle basée sur le modèle *MVC (Modèle-Vue-Contrôleur)*. Cette architecture permet de séparer les traitements de l'interface graphique (cf. Chapitre 7). Dans ce chapitre, nous avons explicité les technologies que nous avons utilisées pour pouvoir faire fonctionner notre approche. A cet égard et vues les réalités auxquelles nous avons été confrontés, il semblait évident que nos *sorties* devraient être dirigées vers un fichier *XML*. Et ceci pour la simple raison que ce langage semble le mieux correspondre à nos attentes : langage disposant d'un langage de requête (en l'occurrence XQuery), facile à

utiliser, accepté par une large communauté. Néanmoins, il faut savoir que la majorité des outils de classification n'utilisent pas de sorties à extension XML, ce qui a posé un certain nombre de problèmes lors de nos travaux. En effet, pour récupérer les informations dont nous avions besoin pour l'interprétation, deux options s'offraient à nous :

- La première consistait à utiliser notre code dans les codes des algorithmes de classification : ce qui n'est pas du tout souhaitable car ceci demande une connaissance approfondie du code dans lequel nous voulons réaliser cette insertion. De plus, cette option semble difficile à mettre en œuvre et demande un surcroît de travail à chaque modification de code dans l'un ou dans l'autre cas.
- La deuxième option consistait à récupérer les fichiers générés par ces modules de classification et à les utiliser dans notre outil en entrée de notre application. Ceci aussi pose un autre problème lié au manque et à la diversité de structuration des résultats des méthodes de classification. En effet, comme nous l'expliquons tout au long de ce rapport, les listings proposés par la majorité des modules de classification ne sont pas exploitables automatiquement.

De ces deux options, nous avons opté pour la seconde. Cependant nous avons été contraints de réaliser des prétraitements afin de fournir en entrée de notre application des fichiers ayant une structure exploitable par notre outil.

Afin d'illustrer notre approche, nous avons défini avec des experts du domaine de la classification, un certain nombre de scénarios d'interprétation couramment utilisés à la suite d'un processus de classification. Il s'agissait pour nous de prendre un exemple concret d'interprétation sur des méthodes de classification utilisant la décomposition de l'inertie totale en inertie interclasse et en inertie intra-classe. Pour rendre cette aide à l'interprétation automatique, nous avons défini une ontologie minimale du domaine de la classification composée par des concepts, des propriétés et des axiomes de ce domaine. Le but de cette ontologie est d'aider à la définition des liens sémantiques entre les différents concepts utilisés dans les modèles d'interprétation. Le dernier chapitre de nos travaux a été l'occasion pour nous d'expliciter, à travers une application concrète, notre approche. Il s'agissait aussi pour nous, de montrer l'intérêt que pourrait avoir une vision jusque là jamais abordée. Bien sûr, l'outil que nous avons développé n'est qu'un prototype mais il n'en demeure pas moins qu'il constitue un exemple concret de la manière dont on peut aider les utilisateurs à interpréter les résultats de leurs méthodes de classification. Ce travail de thèse a donné lieu à quelques publications [Baldé, 2004], [Baldé et al., 2004], [Baldé and Aufaure, 2005] et [Baldé et al., 2006].

Perspectives

Comme nous l'expliquons dans la section précédente, nos travaux ont pour but d'aider à l'interprétation automatique des résultats de méthodes de classification. L'approche que nous avons utilisée et expliquée tout le long de ce travail consiste à capitaliser, en *métadonnées*, les connaissances acquises au cours du processus de classification. La réalisation de cette approche n'aura pas été simple et ce, compte tenu de l'hétérogénéité des structures des résultats des méthodes de classification. La prise en compte de cet aspect a conduit à la mise en place d'un métamodèle d'interprétation. Hormis l'interprétation visuelle proposée par la majorité des outils de classification, nous avions remarqué qu'il n'existe pas, à notre connaissance, de travaux relatifs à l'automatisation du processus d'interprétation des résultats issus des méthodes de classification. Ce qui explique qu'à ce jour, il existe *plusieurs perspectives* à nos travaux :

- Étendre et mieux expliciter l'ontologie minimale de la classification de telle manière qu'elle puisse répondre à toutes les attentes relatives au processus d'interprétation ;
- Définir une plus grande palette de scénarios de manière à pouvoir tenir compte de toutes les possibilités d'interprétation de tous types de méthodes de classification ;
- Faire une étude plus précise des besoins des utilisateurs en terme d'attentes à la suite d'un processus d'interprétation ;
- Intégrer les techniques de visualisation pour une meilleure lecture des métadonnées, car il semble évident qu'une lecture graphique d'informations est souvent beaucoup plus *significative* qu'une longue description ;
- Étendre notre approche à d'autres méthodes de la fouille de données. En effet, nous avons montré que notre approche permettait aussi d'aider à l'interprétation des méthodes dites supervisées (comme dans le cas de l'outil Weka).

Bibliographie

[Aberer and Read, 1998] Aberer, K. and Read, B. J. (1998). Metadata for web databases. In *Proceedings of the* 11^{th} *ERCIM Database Research Group Workshop*, pages 49–57.

[Adamo, 2000] Adamo, J. M. (2000). *Data Mining for Association Rules and Sequential Patterns*. Springer Verlag.

[Adriaans and Zantinge, 1996] Adriaans, P. and Zantinge, D. (1996). *Data Mining*. Addison-Wesley.

[Agrawal et al., 1993] Agrawal, R., Imielinski, T., and Swami, A. N. (1993). Mining associations between sets of items in massive databases. In *ACM-SIGMOD, Conference on Management of Data*, pages 207–216. ACM Press.

[Agrawal and Srikant, 1994] Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Proceedings of the* 20^{th} *International Conference on Very Large Databases*, pages 487–499. Morgan Kaufmann.

[Atzeni et al., 1998] Atzeni, P., Mecca, G., Merialdo, P., and Sindoni, G. (1998). A logical model for metadata in web bases. citeseer.ist.psu.edu/atzeni98logical.html.

[Aussenac-Gilles et al., 2000] Aussenac-Gilles, N., Biebow, B., and Szulman, S. (2000). Revisiting ontology design : a method based on corpus analysis. In *Proceedings of the conference EKAW'2000*, pages 172–188. Springer LNCS 1937.

[Bachimont, 2000] Bachimont, B. (2000). Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances. In Charlet, J., Zacklad, M., Kassel, G., and Bourigault, D., editors, *Ingénierie des connaissances : évolutions récentes et nouveaux défis*, pages 305–323. Eyrolles.

[Bachimont, 2001] Bachimont, B. (2001). Modélisation linguistique et modélisation logique des ontologies : l'apport de l'ontologie formelle. In *Actes des journées francophones d'Ingénierie des Connaissances (IC'2001)*, pages 349–368. Presse Universitaire Grenobloise.

[Baldé, 2004] Baldé, A. (2004). Extraction of metatada on the prototypes resulting from objects clustering. In *International workshop on symbolic data analysis*, Paris, France. University PARIS-IX Dauphine.

[Baldé and Aufaure, 2005] Baldé, A. and Aufaure, M.-A. (2005). How can metadata contribute to add semantic information to clusters ? *Journal of Symbolic Data Analysis*, 3(1) :32–44.

[Baldé et al., 2004] Baldé, A., Lechevallier, Y., and Aufaure, M.-A. (2004). Extraction de métadonnées sur les prototypes issus de la classification d'objets. In *11èmes Rencontre de la Société Francophone de Classification*, pages 95–98, Bordeaux. SFC'2004.

[Baldé et al., 2006] Baldé, A., Lechevallier, Y., Trousse, B., and Aufaure, M.-A. (2006). Utilisation de métadonnées pour l'aide à l'interprétation de classes et de partitions. In *Actes des 6ème journées Extraction et Gestion des Connaissances (EGC 2006), Revue des Nouvelles Technologies de l'Information (RNTI-E-6).*

[Baldonado et al., 1996] Baldonado, M., Chang Chen-Chuan, K., Gravano, L., and Paepcke, A. (1996). Metadata for digital librairies. architecture and design rationale. Technical report, Stanford University. Report SIDL-WP-1996-0051.

[Barboni et al., 2001] Barboni, P., D'angliolini, G., Fanfoni, L., Paolucci, M., and Sulsenti, G. (2001). Esploris : exploring multi-source statistical information systems through metadata. In *New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 209–218.

[Bargmeyer and Gillman, 2000] Bargmeyer, B. E. and Gillman, D. W. (2000). Metadata standards and metadata registries : An overview. In *Proceedings of the International Conference on Establishment Surveys II.*

[Bass et al., 1992] Bass, L., Faneuf, R., Little, R., Mayer, N., Pellegrino, B., Reed, S., Seacord, R., Sheppard, S., and Szczur, M. R. (1992). A metamodel for the runtime architecture of an interactive system. *ACM Special Interest Group Computer-Human Interface bulletin*, 24(1) :32–37.

[Bechhofer et al., 2001] Bechhofer, S., Horrocks, I., Goble, C., and Stevens, R. (2001). Oiled : a reason-able ontology editor for the semantic web. In *Proceedings of KI2001, Joint German/Austria Conference on Artificial Intelligence*, pages 396–408. Springer Verlag LNAI 2174.

[Behja et al., 2005a] Behja, H., Trousse, B., and Marzak, A. (2005a). Plateforme objet d'évaluation orientée point de vue d'un système d'information. *RSTI série ISI - Ingénierie des systèmes d'information - Numéro spécial Méthodes avancées de développement des systèmes d'information.*, 10(6) :107–134.

[Behja et al., 2005b] Behja, H., Trousse, B., and Marzak, A. (2005b). Prise en compte des points de vue pour l'annotation d'un processus d'extraction de connaissances à partir de données. In Pinson, S. and Vincent, N., editors, *Actes des 5ème journées Extraction et Gestion des Connaissances (EGC 2005), Revue des Nouvelles Technologies de l'Information (RNTI-E-3)*, volume 1, pages 245–256. Cépaduès-Editions.

[Ben Ahmed, 2005] Ben Ahmed, W. (2005). *SAFE-NEXT : Une approche Systémique pour l'extraction de connaissances de données.* PhD thesis, Ecole Centrale Paris.

[Ben Ahmed et al., 2002] Ben Ahmed, W., Mekhilef, M., Bigand, M., and Page, Y. (2002). Revue des méthodes et techniques de classification automatique en data mining. Cahier de recherche du Laboratoire LGI, Ecole Centrale Paris.

[Ben Mufti, 2006] Ben Mufti, G. (2006). *Validation d'une classe par estimation de sa stabilité*. PhD thesis, Université Paris-Dauphine.

[Ben Mustapha et al., 2006] Ben Mustapha, N., Aufaure, M.-A., and Zghal, H. B. (2006). Vers une approche de construction de composants ontologiques pour le web sémantique - synthèse et discussion. In Boussaid, O. and Trousse, B., editors, *Proceedings of the EGC'06 Workshop "Fouille de données complexes dans un processus d'extraction"*.

[Berry and Linoff, 1996] Berry, M. J. and Linoff, G. (1996). *Data mining techniques for marketing, sales and customer support*. John Willey & Sons Inc.

[Bertrand and Ben Mufti, 2006] Bertrand, P. and Ben Mufti, G. (2006). Loevinger's measures of rule quality for assessing cluster stability. *Computational statistics and data analysis*, 50(4) :992–1015.

[Bi and Lamb, 2001] Bi, Y. and Lamb, J. (2001). Facilitating integration of distributed statistical databases using metadata and xml. In *Proceedings of the International Conference on New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 1–7.

[Bi et al., 1999] Bi, Y., Murtagh, F., and McClean, S. (1999). Metadata and xml for organising and accessing multiple statistical data sources. In *Proceedings of Conference of the International Association for Statistical Computing*, pages 393–404.

[Biébow and Szulman, 1999] Biébow, B. and Szulman, S. (1999). Terminae : a method and a tool to build of a domain ontology. In Benjamins, V., Fensel, D., and Pérez, A., editors, *Proceedings of International Workshop on Ontological Engineering on the Global Information Infrastructure*, pages 25–30. Springer.

[Blàzquez et al., 1998] Blàzquez, M., Fernández-Lòpez, M., Garcia-Pinar, J. M., and Gòmez-Pérez, A. (1998). Building ontologies at the knowledge level using the ontology design environment. In *Proceedings of Knwoledge Acquisition Workshop (KAW'98)*.

[Blomqvist, 2005] Blomqvist, E. (2005). Fully automatic construction of enterprise ontologies using design patterns : Initial method and first experiences. *Lecture Notes in Computer Science*, 91(3761) :1314–1329.

[Blomqvist and Sandkuhl, 2005] Blomqvist, E. and Sandkuhl, K. (2005). Patterns in ontology engineering : Classification of ontology patterns. In Chen, C. S., Filipe, J., Seruca, I., and Cordeiro, J., editors, *ICEIS (3)*, pages 413–416.

[Bock and Diday, 2000] Bock, H.-H. and Diday, E. (2000). *Analysis of Symbolic Data. Exploratory methods for extracting statistical information from complex data*, chapter 4. Springer Verlag.

[Bolshakova and Azuaje, 2003] Bolshakova, N. and Azuaje, F. (2003). Improving expression data mining through cluster validation. In *Proceedings of the* 4^{th} *Annual IEEE Conference on Information Technology Applications in Biomedecine*, pages 19–22.

[Booleman, 2004] Booleman, M. (2004). The dutch metadata model : Introduction of conceptual metadata, process metadata and implementation strategy. In *Proceedings of the international conference on Statistics Investment in the Future.*

[Boussaid and Loudcher, 2006] Boussaid, O. and Loudcher, L.-R. (2006). Intégration des métadonnées dans la fouille de données. In *Actes du XXIVème Congrès INFORSID*, pages 719–734.

[Brachman and Anand, 1996] Brachman, R. and Anand, T. (1996). The process of knowledge discovery in databases : A human-centered approach. *Advances in Knowledge Discovery and Data Mining*, pages 37–58.

[Brank et al., 2005] Brank, J., Grobelnik, M., and Mladenić, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of Conference on Data Mining and Data Warehouses (SiKDD 2005).*

[Brickley and Guha, 1999] Brickley, D. and Guha, R. (1999). Resource Description Framework Specification. http ://www.w3.org/TR/REC-rdf-syntax/.

[Brisson et al., 2006] Brisson, L., Collard, M., and Pasquier, N. (2006). Ontologie et base de connaissances pour le pré-traitement et post-traitement en fouille de données. In Boussaid, O. and Trousse, B., editors, *Proceedings of the EGC'06 Workshop "Fouille de données complexes dans un processus d'extraction"*, pages 13–26.

[Burgun and Bodenreider, 2001] Burgun, A. and Bodenreider, O. (2001). Mapping the umls semantic network into general ontologies. In *Proceedings of the AMIA Annual Symposium*, pages 81–85. Springer LNAI 2473.

[Bézivin and Gerbé, 2001] Bézivin, J. and Gerbé, O. (2001). Towards a precise definition of the omg/mda framework. In *Proceedings of the Conference on Autonomous Software Egineering (ASE'01)*, pages 273–280. IEEE Computer Society Press.

[Bézivin et al., 1995] Bézivin, J., Lanneluc, J., and Lemesle, R. (1995). sNets : The Core Formalism for an Object-Oriented CASE Tool. *Object-Oriented Technology for Databases and Software Systems*, pages 224–239.

[Calvanese et al., 2001] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, N., and Rosati, R. (2001). Data integration in data warehousing. *International Journal of Cooperative Information Systems*, 10 :237–271.

[Cannataro and Comito, 2003] Cannataro, M. and Comito, C. (2003). A datamining ontology for grid programming. In *Proceedings of* 1^{st} *Workshop on Semantics in Peer-to-Peer and Grid Computing (in conj. with WWW2003)*, pages 113–134.

[Celeux et al., 1989] Celeux, G., Diday, E., Govaert, G., Lechevallier, Y., and Ralambondrainy, H. (1989). *Classification automatique des données. Environnement statistique et informatique*, pages 155–164. Dunod.

[Chandrasekaran et al., 1999] Chandrasekaran, B., Josephson, J., and Benjamins, V. (1999). What are ontologies, and why do we need them ? *IEEE Intelligent Systems*, 14(1) :20–26.

[Charlet et al., 2000] Charlet, J., Zacklad, M., Kassel, G., and Bourigault, D. (2000). *Ingénierie des connaissances, évolutions récentes et nouveaux défis*. Eyrolles.

[Chauchat, 2002] Chauchat, J.-H. (2002). Une application du data mining, ou fouille de données, au marketing : construction et mise à jour d'une segmentation des clients. 1^{er} colloque franco-libanais de statistique, actes en cours.

[Chavent et al., 2002] Chavent, M., De Carvalho, F., Lechevallier, Y., and Verde, R. (2002). New clustering methods of interval data. In *Proceedings of the International Conference of the Gesellschaft für Klassifikation (GfKl)*.

[Chavent et al., 2006] Chavent, M., De Carvalho, F., Lechevallier, Y., and Verde, R. (2006). New clustering methods of interval data. *Computational Statistics*, 21(2) :211–229.

[Corcho et al., 2001] Corcho, O., López, F. M., and Pérez, A. G. (2001). Ontoweb : Technical roadmap. www.ontoweb.org.

[Core, 2006] Core, D. (2006). DUBLIN CORE. http ://dublincore.org/.

[Cormuéjols and Miclet, 2002] Cormuéjols, A. and Miclet, L. (2002). *Apprentissage artificiel : concepts et algorithmes*. Eyrolles.

[Coutaz, 1987] Coutaz, J. (1987). Pac : an implementation model for dialog design. In *Proceedings of Interact'87*, pages 431–436. Bullinger and Shackel.

[Crawley et al., 1997] Crawley, S., Davis, S., Indulska, J., McBride, S., and Raymond, K. (1997). Meta-meta is better-better ! In *Proceedings of the IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems (DAIS'97)*. Disponible à l'adresse suivante : http ://citeseer.ist.psu.edu/44757.html.

[Cregan et al., 2005] Cregan, A., Mochol, M., Vrandečić, D., and Bechhofer, S. (2005). Pushing the limits of OWL, Rules and Protégé : A simple example. In Grau, B. C., Horrocks, I., Parsia, B., and Patel-Schneider, P., editors, *OWL : Experiences and Directions*. Disponible à l'adresse suivante : http ://dl.kr.org/owled2005/sub22.pdf.

[CWM, 2006] CWM (2006). Common Warehouse MetaModel. http ://www.omg.org/technology/cwm/.

[Daconta et al., 2003] Daconta, M. C., Obrst, L. J., and Smith, K. T. (2003). *The Semantic Web : A Guide to the Future of XML, Web Services and Knowledge Management*. Wiley.

[De Carvalho et al., 2006] De Carvalho, F., Brito, P., and Bock, H.-H. (2006). New clustering methods of interval data. *Computational Statistics*, 21(2) :231–250.

[DeMichiel et al., 2000] DeMichiel, L. G., Yalçinalp, L., and Krishnan, S. (2000). Enterprise java beans specification version 2.0 - public draft.

[Depaulis et al., 2006] Depaulis, F., Jambon, F., Girard, P., and Guittet, L. (2006). Le modèle d'architecture logicielle h4 : Principes, usages, outils et retours d'expérience dans les applications de conception technique. *Revue d'Interaction Homme Machine*, 7(1) :93–129.

[Diday, 1991] Diday, E. (1991). Des objets de l'analyse des données à ceux de l'analyse des connaissances. *Induction Symbolique-Numérique*, pages 9–76.

[Diday, 1998] Diday, E. (1998). L'analyse des données symboliques : un cadre théorique et des outils. Technical report, CEREMADE - Université Paris Dauphine.

[Diday, 2003] Diday, E. (2003). Cours dea. Université Paris-Dauphine.

[Diday et al., 2000] Diday, E., Kodratoff, Y., Brito, P. M., and Moulet, M. (2000). *Induction symbolique numérique à partir de données*. Cépaduès Editions.

[Dieng et al., 2001] Dieng, R., Corby, O., Gandon, F., Giboin, A., Golebiowska, J., Matta, N., and Ribière, M. (2001). *Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du knowledge management*. Dunod Ed., 2^{nde} edition.

[DMG, 2006] DMG (2006). Predictive model markup language. www.dmg.org.

[DOE, 2002] DOE (2002). Differential ontology editor home page. http ://opales.ina.fr/public.

[Douglas and Guha, 1990] Douglas, B. L. and Guha, R. (1990). *Building Large Knowledge-Based Systems : Representation and Inference in the CYC Project*. Addison-Wesley.

[Duineveld et al., 2000] Duineveld, A., Stoter, R., Weiden, M., Kenepa, B., and Benjamins, V. (2000). Wondertools ? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 52(6) :1111–1133.

[Everitt, 1993] Everitt, B. (1993). *Cluster Analysis*. Edward Arnold, 3rd edition.

[Famili et al., 2000] Famili, A., Shen, W., Weber, R., and Simoudis, E. (2000). Data preprocessing and intelligent data analysis. *IEEE Data Engineering Bulletin*, 23(4) :1–28.

[Farmakis et al., 2001] Farmakis, G., Kapetanakis, Y., Petrakos, G., and Petrakos, M. (2001). Architecture and design of a flexible integrated information system for official statistics surveys, based on structural survey metadata. In *Proceedings of the International Conference on New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 219–229.

[Farquhar et al., 1995] Farquhar, A., Fikes, R., Pratt, W., and Rice, J. (1995). Collaborative ontology construction for information integration. Technical report, Knowledge Systems Laboratory (Stanford).

[Farquhar et al., 1997] Farquhar, A., Fikes, R., and Rice, J. (1997). The ontolingua server : a tool for collaborative ontology construction. *International journal of Human-Computer studies*, 46(6) :707–727.

[Fayyad et al., 1996] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996). *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press.

[Fekete, 1996] Fekete, J.-D. (1996). *Un modèle multicouche pour la construction d'applications graphiques interactives*. PhD thesis, LRI, Université Paris-Sud, Orsay.

[Fernández-Lòpez et al., 1997] Fernández-Lòpez, M., Gòmez-Pérez, A., and Juristo, N. (1997). METHONTOLOGY : from ontological art towards ontological engineering. In *Proceedings of the Spring Symposium Series on Ontological Engineering(AAAI'97)*, pages 33–40. AAAI Press.

[Fitzgerald, 2001] Fitzgerald, M. (2001). *XSLT Essentials*. John Wiley and Sons, Inc.

[Fortuna et al., 2005] Fortuna, B., Mladenič, D., and Grobelnik, M. (2005). Semi-automatic construction of topic ontology. In Ackermann, M., Berendt, B., Grobelnik, M., and Svatek, v., editors, *Proceedings of the Workshop W3. The* 2^{nd} *International Workshop on Knowledge Discovery and Ontologies (KDO-2005)*, pages 23–30.

[Fowler, 1999] Fowler, A. (1999). A Swing Architecture Overview (The Inside Story of JFC Component Design). Technical report, Sun. http ://-java.sun.com/products/jfc/tsc/articles/architecture/.

[Froeschl and Grossmann, 1999] Froeschl, K. A. and Grossmann, W. (1999). Metadata as a tool to improve quality of the statistical production process. *Bulletin of the International Statistical Institute*, 91(1) :349–350. Proc. ISI, 52^{nde} Session, Helsinki.

[Fürst, 2002] Fürst, F. (2002). L'ingénierie ontologique. Rapport de recherche n^o02-07.

[Gandon, 2002] Gandon, F. (2002). Ontology engineering : a survey and a return on experience. Technical report, INRIA. Rapport de recherche 4396.

[Gerbé and Kerhervé, 1998] Gerbé, O. and Kerhervé, B. (1998). Modeling and metamodeling requirements for knowledge management. In *Proceedings of OOPSLA Workshop on Model Engineering with CIDF*. ACM.

[Girard et al., 1995] Girard, P., Pierra, G., and Guittet, L. (1995). Les interacteurs hiérarchisés : une architecture orientée tâches pour la conception des dialogues. *Revue d'Automatique et de Productique Appliquée (RAPA)*, 8(2-3) :235–240.

[Gómez-Pérez, 1999] Gómez-Pérez, A. (1999). Ontological engineering : A state of the art. *Expert Update*, 2(3) :33–43.

[Gòmez-Pérez, 1999] Gòmez-Pérez, A. (1999). Evaluation of taxonomic knowledge in ontologies and knowledge bases. In *Proceedings of the* 12^{th} *workshop on knowledge acquisition, Modeling and Management (KAW'99)*, volume 2, pages 6.1.1–6.1.18.

[Gòmez-Pérez et al., 2003] Gòmez-Pérez, A., Fernández-Lòpez, M., and Corcho, O. (2003). *Ontological engineering*. Springer-Verlag.

[Gordon, 1999] Gordon, A. D. (1999). *Classification*. Chapman and Hall, 2^{nde} edition.

[Green, 1985] Green, M. (1985). Report on dialogue specification tools. In *User Interface Management Systems (UIMS), Eurographics Seminars*, pages 9–20. Springer-Verlag.

[Grinstein et al., 2001] Grinstein, G., Trutschl, M., and Cvek, U. (2001). High-dimensional visualizations. In *Proceedings of the Visual Data Mining Workshop*.

[Grüninger and Fox, 1995] Grüninger, M. and Fox, M. S. (1995). Methodology for the design and evaluation of ontologies. In *Proceedings of the Workshop on Basic Ontological Issues on Knowledge Sharing, IJCAI'95*.

[Grossman, 2004a] Grossman, R. (2004a). Business intelligence applications in financial services : Transitioning to clusters. www.rgrossman.com.

[Grossman, 2004b] Grossman, W. (2004b). Data, metadata and statistical information. In *Presentation at Statistics : investment in the future*.

[Grossmann, 1998] Grossmann, W. (1998). Use of metadata in the statistical production process. In *Proceedings of New Techniques and Technologies for Statistics Exchange of Technology and Know-how*.

[Gruber, 1993] Gruber, T. R. (1993). A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2) :199–220.

[Gruber et al., 1995] Gruber, T. R., Guarino, N., and Poli, R. (1995). Formal ontology in conceptual analysis and knowledge representation. *Special issue of the International Journal of Human and Computer Studies*, 43(5-6) :625–640. citeseer.ist.psu.edu/guarino95formal.html.

[Gruber and Olsen, 1994] Gruber, T. R. and Olsen, G. R. (1994). An ontology for engineering mathematics. In *Proceedings of the* 4^{th} *International Conference on Principles of Knowledge Representation and Reasonning*, pages 258–269. Morgan-Kauffmann.

[Gruninger and Lee, 2002] Gruninger, M. and Lee, J. (2002). Ontology applications and design. *ACM Communication*, 45(2) :39–41.

[Guarino, 1997a] Guarino, N. (1997a). Some organizing principles for a unified top-level ontology. In *AAAI Spring Symposium on Ontological Engineering*, pages 57–63.

[Guarino, 1997b] Guarino, N. (1997b). Understanding, building and using ontologies. *International Journal of Human Computer Studies*, 46 :293–310.

[Guarino, 1997c] Guarino, N. (1997c). Understanding, building, and using ontologies : A commentary to 'using explicit ontologies in kbs development'. *International Journal of Human and Computer Studies*, 46 :293–310.

[Guarino et al., 1994] Guarino, N., Carrara, M., and Giaretta, P. (1994). An ontology of meta-level categories. In *Principles of Knowledge representation and Knowledge Reasonning : Proceedings of the* 4^{th} *international conference (KR'94)*, pages 270–280. Morgan-Kauffmann.

[Guarino and Giaretta, 1995] Guarino, N. and Giaretta, P. (1995). Ontologies and knowledge bases : towards a terminological clarification. In *Towards very large knowledge bases*, pages 25–32.

[Guarino and Welty, 2000a] Guarino, N. and Welty, C. (2000a). A formal ontology of properties. In Dieng, R. and Corby, O., editors, *Knowledge Engineering and Knowledge Management : Methods, Models and Tools. International Conference EKAW'2000*, pages 97–112. Springer-Verlag.

[Guarino and Welty, 2000b] Guarino, N. and Welty, C. (2000b). Towards a methodology for ontology based model engineering. In *International Workshop on Model Engineering (in conjunction with ECOOP 2000)*.

[Guittet, 1995] Guittet, L. (1995). *Contribution à l'Ingénierie des Interfaces Homme-Machine - Théorie des Interacteurs et Architecture H4 dans le système NODAOO.* PhD thesis, LISI/ENSMA, Université de Poitiers.

[Halkidi et al., 2001a] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001a). Clustering algorithms and validity measures. In *Proceedings of the SSDBM Conference*, pages 3–22.

[Halkidi et al., 2001b] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001b). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3) :107–145.

[Halkidi et al., 2002] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2002). Cluster validity methods : part 1. *SIGMOD Rec.*, 31(2) :40–45.

[Han et al., 2003] Han, H., Giles, C. L., Manavoglu, E., and Zha, H. (2003). Automatic document metadata extraction using support vector machines. In *Proceedings of the Joint Conference on Digital Librairies*, pages 37–48.

[Han and Kamber, 2000] Han, J. and Kamber, M. (2000). *Data Mining : Concepts and Techniques.* Morgan Kaufmann.

[Hartmann et al., 2005] Hartmann, J., Spyns, P., Giboin, A., Maynard, D., Cuel, R., Suárez-Figueroa, M. C., and Sure, Y. (2005). D1.2.3 methods for ontology evaluation. Technical report, KWEB. http ://-java.sun.com/products/jfc/tsc/articles/architecture/.

[Hellerstein, 1997] Hellerstein, J. (1997). Special issue on data reduction techniques. *Bulletin of the Technical Committee on Data Engineering*, 20(4).

[Heujst et al., 1997] Heujst, V., Schreiber, A., and Wielinga, B. (1997). Using explicit ontologies in kbs development. *International Journal of Human and Computer Studies/ Knowledge Acquisition*, 2(3) :183–292.

[Holsapple and Joshi, 2002] Holsapple, C. W. and Joshi, K. D. (2002). A collaborative approach to ontology design. *Commununications ACM*, 45(2) :42–47.

[Inmon, 1996] Inmon, W. (1996). *Building the datawarehouse.* John Wiley and Sons.

[J2EE, 2006] J2EE (2006). Java 2 Enterprise Edition Home Page. http ://-java.sun.com/j2ee/.

[Jain and Dubes, 1988] Jain, A. and Dubes, R. (1988). *Algorithms for Clustering Data.* Prentice Hall.

[Jain et al., 1999] Jain, A., Narashima Murty, M., and Flynn, P. (1999). Data clustering : A review. *ACM Computing surveys*, 31(3) :264–323.

[Jambu, 1978] Jambu, M. (1978). *Classification automatique pour l'analyse des données*, pages 245–286. Dunod.

[Jena, 2006] Jena (2006). Jena. http ://jena.sourceforge.net/tutorial/RDF_API/index.html.

[Jouini et al., 2003] Jouini, W., Ben Ahmed, W., Mekhilef, M., Bigand, M., and Page, Y. (2003). Les méthodes et techniques d'extraction de connaissances de bases de données. Cahier de recherche du Laboratoire LGI, Ecole Centrale Paris.

[Karge and Rauch, 2001] Karge, R. and Rauch, L. (2001). Central metadata system(cms) in statistical production and publication environment. In *Proceedings of the International Conference on New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 195–208.

[Kassel, 2002] Kassel, G. (2002). Ontospec : une méthode de spécification semi-informelle d'ontologies. In *Actes des journées francophones d'Ingénierie des Connaissances (IC'2002)*, pages 75–87.

[Kassel et al., 2000] Kassel, G., Abel, M., Barry, C., Boulitreau, P., Irastorza, C., and Perpette, S. (2000). Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche. In *Actes des journées francophones d'Ingénierie des Connaissances (IC'2000)*, pages 251–259.

[Kay, 2001] Kay, M. (2001). *XSLT Programmer's Reference.* Wrox, 2^{nde} edition.

[Keith, 2001] Keith, J. G. (2001). Metadata : an overview and some issues. Disponible à l'adresse suivante : http ://citeseer.ist.psu.edu/46364.html.

[Kent and Schuerhoff, 1997] Kent, J. and Schuerhoff, M. (1997). Some thoughts about a metadata management system. In *Proceedings of the* 9^{th} *International Conference on Scientific and Statistical Database Systems*, pages 174–185.

[Kiefer et al., 1995] Kiefer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42 :741–843.

[KIF, 2002] KIF (2002). Knowledge Interchange Format Home Page. http ://logic.stanford.edu/kif/dpans.html.

[Kokolakis et al., 2001] Kokolakis, G., Kouvaras, G., and Panagopoulou, G. (2001). The role of metadata in the intelligent use of the data : The user approach. In *Proceedings of the International Conference on New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 257–262.

[Krasner and Pope, 1988] Krasner, G. and Pope, S. (1988). A cookbook for using the model-view-controller user interface paradigm in smalltalk-80. *Journal of Object Oriented Programming (JOOP)*, 1(3) :26–49.

[Lamb, 1993] Lamb, J. (1993). Metadata in survey processing. In Franchet, Y., editor, *Proceedings of the EUROSTAT Statistical Meta Information Systems Workshop*, pages 103–112.

[Lamb, 2001] Lamb, J. (2001). Sharing best methods and know-how for improving generation and use of metadata. In *NTTS2001-ETK2001 Pre-proceedings*, volume 1, pages 175–194.

[Lamiri et al., 2004] Lamiri, M., Ben Ahmed, W., Mekhilef, M., Bigand, M., and Page, Y. (2004). Interprétation des résultats issus de classification automatique. Cahier de recherche du Laboratoire LGI, Ecole Centrale Paris.

[LaPlant et al., 1996] LaPlant, W., Lestina, G., and Gillman, D.W.and Appel, M. (1996). Proposal for a statistical metadata standard. U.S. Bureau of the Census Annual Research Conference.

[Le Moigne, 1999] Le Moigne, J.-L. (1999). *La modélisation des systèmes complexes.* Dunod.

[Le Pallec, 2000] Le Pallec, X. (2000). Workflow et travail collaboratif.

[Lefébure and Venturi, 1998] Lefébure, R. and Venturi, G. (1998). *Data Mining. Gestion de la relation client ; personnalisation de sites Web.* Eyrolles.

[Lindblom and Sundgren, 2004] Lindblom, H. and Sundgren, B. (2004). The metadata system at statistics sweden in an international perspective. In *Presentation at Statistics : investment in the future.* Disponible à l'adresse suivante : http ://www.czso.cz/sif/conference2004.nsf.

[Litvak et al., 2005] Litvak, M., Last, M., and Kisilevich, S. (2005). Improving classification of multi-lingual web documents using domain ontologies. In Ackermann, M., Berendt, B., Grobelnik, M., and Svatek, V., editors, *Proceedings of the Workshop W3. The 2^{nd} International Workshop on Knowledge Discovery and Ontologies (KDO-2005)*, pages 67–74.

[Liu et al., 2000] Liu, B., Chen, S., and Ma, Y. (2000). Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems and their Applications*, 15(5) :47–55.

[López, 1999] López, F. M. (1999). Overview of methodologies for building ontologies. In *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5).*

[Ma and Lamb, 2001] Ma, D. and Lamb, J. (2001). User workspace for access the statistical data from the library. In *Proceedings of the International Conference on New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 249–256.

[Manco et al., 2004] Manco, G., Pizzuti, C., and Talia, D. (2004). Eureka ! : an interactive and visual knowledge discovery tool. *Journal of Visual Languages and Computing*, 15(1) :1–35.

[Marcos and Solange, 2005] Marcos, A. D. and Solange, O. R. (2005). Using taxonomies to facilitate the analysis of the association rules. pages 59–66.

[McGuinness et al., 2000] McGuinness, D. L., Fikes, R., Rice, J., and Wilder, S. (2000). An environment for merging and testing large ontologies. In Cohn, A. G., Giunchiglia, F., and Selman, B., editors, *Proceedings of the Seventh International Conference of Principles of Knowledge Representation and Reasoning (KR2000)*, pages 483–493. Morgan Kaufmann.

[McGuinness and Wright, 1998] McGuinness, D. L. and Wright, J. (1998). Conceptual modeling for configuration : A description logic-based approach. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing - special issue on Configuration*, 12(4) :333–344.

[MCV, 2006] MCV (2006). Metadata Common Vocabulary. Statistical Data and Metadata Exchange Initiative.

[Mädche et al., 2000] Mädche, A., Schnurr, H., Staab, S., and Studer, R. (2000). Representation language-neutral modeling of ontologies. In *Proceedings of the German Workshop Modellierung*, pages 128–144.

[Miller, 2000] Miller, L. (2000). Ontologies and metadata. In *Semantic Web Technologies Workshop*. A Draft Discussion.

[Mitchell, 1997] Mitchell, T. (1997). *Machine learning*.

[Mizoguchi, 2004] Mizoguchi, R. (2004). Tutorial on ontological engineering. part 2 : Ontology development, tools and languages. *New Generation Computing*, 22 :61–96.

[Mizoguchi and Ikeda, 1997] Mizoguchi, R. and Ikeda, M. (1997). Towards ontology engineering. In *Proceedings of the Joint Pacific Asian Conference on Expert Systems*, pages 259–266.

[Mizoguchi et al., 2000] Mizoguchi, R., Kozaki, K., Sano, T., and Kitamura, Y. (2000). Construction and deployment of a plant ontology. In *Proceedings of the* 12^{th} *International Conference, EKAW 2000*, pages 113–128. Lecture Notes in Artificial Intelligence.

[MOF, 2006] MOF (2006). Meta-object facility home page. http ://www.omg.org/technology/documents/formal/mof.htm.

[Mougin et al., 2003] Mougin, F., Marquet, G., Burgun, A., Guérin, E., Moussouni, F., and Olivier, L. (2003). Use of metadata for biomedical heterogeneous data sources integration. In *Proceedings of the* 2^{nd} *European Conference on Computational Biology*. Poster.

[Musen, 1992] Musen, M. A. (1992). Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research*, 25 :435–467.

[Nigay, 1994] Nigay, L. (1994). *Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales*. PhD thesis, Université Joseph-Fourier Grenoble I.

[Nigay and Coutaz, 1993] Nigay, L. and Coutaz, J. (1993). A design space for multimodal systems : Concurrent processing and data fusion. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems, Voices and Faces*, pages 172–178.

[Noirhomme-Fraiture and Rouard, 2000] Noirhomme-Fraiture, M. and Rouard, M. (2000). *visualizing and Editing symbolic objects*, chapter 7, pages 125–138. Springer Verlag.

[Noy et al., 2000] Noy, N. F., Fergerson, R., and Musen, M. (2000). The knowledge model of protégé2000 : combining interoperability and flexibility. In Dieng, R. and Corby, O., editors, *Proceedings of the* 12^{th} *International Conference on Knowledge Engineering and Knowledge Management (EKAW 2000)*, volume 1937, pages 17–32. Springer.

[Noy and McGuinness, 2001] Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101 : A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.

[NU and ECE, 1998] NU and ECE (1998). Standards for statistical metadata on internet.

[NU and ECE, 2000a] NU and ECE (2000a). *Guidelines for statistical metadata on the Internet*, volume 52.

[NU and ECE, 2000b] NU and ECE (2000b). *Terminology on statistical Metadata*, volume 53. United Nations Pubns.

[OCDE, 1997] OCDE (1997). *Main Economic indicators, sources and methods, labour and wage statistics*. Lavoisier. Statistics Directorate.

[OCDE, 2004] OCDE (2004). The role of metadata in promoting international comparisons and adherence to international statistical standards. www.oecd.org/std/metarole.htm.

[ODE, 2002] ODE (2002). Ontology design environment home page. http ://www.swi.psy.uva.nl/wondertools/html/ODE.html.

[OILED, 2002] OILED (2002). Oil editor home page. http ://oiled.man.ac.uk/.

[Olenski, 2001] Olenski, J. (2001). Pratical problems of implementing metadata standards in official statistics. In *The* 8^{th} *International Conference on Scientific and Statistical Database Management*, pages 130–148.

[OMG, 2006] OMG (2006). Object Management Group. http ://www.omg.org.

[OntoEdit, 2002] OntoEdit (2002). Ontology editor home page. http ://www.ontoprise.de/com/.

[Ontolingua, 2002] Ontolingua (2002). Ontoligua home page. http ://www.ksl.stanford.edu/software.

[OWL, 2004] OWL (2004). OWL Web Ontology Language Semantics and Abstract Syntax. http ://www.w3.org/2004/OWL/.

[Papageorgiou, 2003] Papageorgiou, H. (2003). Revision of the asso metadata model. Technical report, ASSO Project. Deliverable 4.4.

[Papageorgiou et al., 2001] Papageorgiou, H., Pentaris, F., Theodorou, E., Vardaki, M., and Petrakos, M. (2001). A statistical metadata model for simultaneous manipulation of data and metadata. *Journal of Intelligent Information Systems (JIIS)*, 17(2/3) :169–192. invited paper.

[Papageorgiou et al., 2000a] Papageorgiou, H., Vardaki, M., and Pentaris, F. (2000a). Data and metadata transformations. *Research in Official Statistics (ROS)*, 3(2) :27–43.

[Papageorgiou et al., 2000b] Papageorgiou, H., Vardaki, M., and Pentaris, F. (2000b). Quality of statistical metadata. *Research in Official statistics*, 3 :45–57.

[Papageorgiou et al., 2000c] Papageorgiou, H., Vardaki, M., and Pentaris, F. (2000c). Recent advances on metadata. *Computational statistics*, 15(1) :89–97.

[Papageorgiou et al., 2002] Papageorgiou, H., Vardaki, M., Theodorou, E., and Pentaris, F. (2002). The use of statistical metadata modelling and related transformations to assess the quality of statistical reports. In *Joint UNECE/Eurostat Seminar on Integrated Statistical Information Systems and Related Matters (ISIS 2002)*. Invited paper. Disponible à www.unece.org/stats/documents/ces/sem.47/24.e.pdf.

[Parekh et al., 2004] Parekh, V., Gwo, J.-P. J., and Finin, T. (2004). Ontology based Semantic Metadata for GeoScience Data. In *Proceedings of the International Conference of Information and Knowledge Engineering*, pages 485–490. The International MultiConference in Computer Science and Computer Engineering.

[Pellegrino, 2000] Pellegrino, M. (2000). The harmonisation of statistical metadata for the European Union : Eurostat's needs and responsibilities. In *Work session on Statistical Metadata*. Disponible à l'adresse suivante : http ://www.unece.org/stats/documents/2000/11/metis/16.e.pdf.

[Petrakos et al., 2001] Petrakos, M., Farmakis, G. E., Petrakos, G., Bisiela, P., and Koumanakos, G. (2001). A structured approach to statistical metadata base. In *Proceedings of New Techniques and Technologies for Statistics Exchange of Technology and Know-how*, pages 1–7.

[Pöyry et al., 2002] Pöyry, P., Pelto-Aho, K., and Järvi, J. P. (2002). The role of metadata int the cuber system. In *Proceedings of SAICSIT 2002*, pages 172–178.

[Protége2000, 2002] Protége2000 (2002). PROTEGE2000 ontology editor home page. http ://protege.stanford.edu/.

[Pukli, 2004] Pukli, P. (2004). Information technology for statistics. In *Presentation at Statistics : investment in the future*. Disponible à l'adresse suivante : http ://www.czso.cz/sif/conference2004.nsf.

[Pyle, 1999] Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kaufmann.

[Rahm and Do, 1997] Rahm, E. and Do, H. (1997). Data cleaning : Problems and current approaches. *Intelligent Data Analysis*, 1 :3–23.

[Rigaux and Amann, 2002] Rigaux, P. and Amann, B. (2002). *Comprendre XSLT*. O'Reilly.

[Roche, 2006] Roche, C. (2006). Terminologie et ontologie. *Revue Langages*, (157) :48–62.

[Roche, 2007] Roche, C. (2007). Dire n'est pas concevoir. In *Conférence IC 2007 : Ingénierie des Connaissances*. A paraître.

[Role, 1999] Role, F. (1999). Panorama des travaux en cours dans le domaine des méta-données. Technical report, INRIA.

[Roth et al., 2002] Roth, V., Lange, T., Braun, M., and Buhmann, J. (2002). A resampling approach to cluster validation. In *Proceedings of the International Conference COMPSTAT*, pages 123–128.

[Rothenfluh et al., 1996] Rothenfluh, T., Gennari, J., Eriksson, H., Puerta, A., Tu, S., and Musen, M. A. (1996). Reusable ontologies, knowledge-acquisition tools, and performance systems : ProtÉgÉ-ii solutions to sisyphus-2. *International Journal of Human-Computer Studies*, 44 :303–332.

[Rouard et al., 1998] Rouard, M., Noirhomme-Fraiture, M., and Bisdorff, R. (1998). Utilisation de l'etoile zoom en exploration de données statistiques. In *Proceedings of Knowledge Extraction from Statistical Data (KESDA'98)*.

[Saeki, 2000] Saeki, M. (2000). Toward formal semantics of meta-models. In *International Workshop on Model Engineering (in conjunction with ECOOP 2000)*. Disponible à l'adresse suivante : http ://www.metamodel.com/IWME00/program.html.

[Sarvas et al., 2004] Sarvas, R., Herrarte, E., Wilhelm, A., and Davis, M. (2004). Metadata creation system for mobile images. In *MobiSys '04 : Proceedings of the* 2^{nde} *International conference on Mobile systems, applications, and services*, pages 36–48. ACM Press.

[Saxon, 2006] Saxon (2006). Saxon. http ://www.saxonica.com/.

[Schmuker, 1986] Schmuker, K. J. (1986). *Object Oriented Programming for the Macintosh*. Hayden.

[Schweiger and Sklar, 1983] Schweiger, B. and Sklar, A. (1983). *Probabilistic Metrie Spaces*. North-Holland.

[SCI, 2006] SCI (2006). Metadata. http ://www.sci.gc.ca/metadata.html.

[SDMX, 2006] SDMX (2006). Sdmx standards. www.sdmx.org/standards/index.aspx.

[Sibilla and Jocteur-Monrozier, 1999] Sibilla, M. and Jocteur-Monrozier, F. (1999). Sumo : Supervision globale des systèmes. In *Actes de l'atelier Middleware et les applications internet*. Centre National d'Études Spatiales.

[Soltés, 2004] Soltés, D. (2004). New challenges and roles of metadata in text/data mining in statistics. In *Proceedings of the NEMIS 2004 Final Conference*, pages 191–201. Springler Verlag.

[Sousa and Tendeiro, 2005] Sousa, F. and Tendeiro, J. (2005). A validation methodology in hierarchical clustering. In *Proceedings of the International Symposium on Applied Stochastic Models and Data Analysis*, pages 396–403.

[Sowa, 1995a] Sowa, J. (1995a). Distinction, combination and constraints. In *Proceedings of IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, page 175.

[Sowa, 1995b] Sowa, J. (1995b). Top-level ontological categories. *International Journal of Human and Computer Studies*, 43 :669–685.

[Sowa, 2000] Sowa, J. (2000). Ontology, metadata and semiotics. In *Proceedings of the 8th International Conference on Conceptual Structures (ICCS'2000)*, pages 55–81. Springer-Verlag.

[Srikant and Agrawal, 1997] Srikant, R. and Agrawal, R. (1997). Mining generalized association rules. *Future Generation Computer Systems*, 13(2-3) :161–180.

[Stéphan, 1998] Stéphan, V. (1998). *Construction d'objets symboliques par synthèse des résultats de requêtes SQL.* PhD thesis, Université Paris Dauphine.

[Studer, 1998] Studer, B. F. (1998). Knowledge engeeneering : Principles and methods. *Data and Knowledge Engineering*, 25 :161–197.

[Sundgren, 1995] Sundgren, B. (1995). Guidelines for the modelling of statistical data and metadata. In *Proceedings of Conference of European Statisticians. Methodological material.* Disponible à l'adresse suivante : www.unece.org/stats/publications/metadatamodeling.pdf.

[Sundgren, 1999] Sundgren, B. (1999). Information systems architecture for national and international statistical offices, guidelines and recommendations. In *Proceedings of International Conference of European Statisticians, Satistical standards and studies*, volume 51.

[Sure et al., 2002] Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., and Wenke, D. (2002). Ontoedit : colllaborative ontology development for the semantic web. In *Proceedings of the International Semantic Web Conference*, pages 221–235. Springer-Verlag LNCS 2342.

[Tallur, 1982] Tallur, B. (1982). Méthode d'interprétation d'une classification hiérarchique d'attributs-modalités pour l'explication d'une variable : application à la recherche du seuil critique de la tension artérielle systolique et des indicateurs de risque cardiovasculaire. Technical Report 123, IRISA.

[Tannebaum, 2002] Tannebaum, A. (2002). *Metadata Solutions - Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand.* Addison-Wesley.

[Terminae, 2002] Terminae (2002). TERMINAE Home Page. http ://www.lipn.univ-paris13.fr/ szulman/TERMINAE.html.

[Thai and Lam, 2001] Thai, T. and Lam, H. (2001). *.Net Framework Essentials*. O'Reilly.

[Theodoridis and Koutroumbas, 1999] Theodoridis, S. and Koutroumbas, K. (1999). *Pattern recognition*. Academic Press.

[Tidwell, 2001] Tidwell, D. (2001). *XSLT*. O'Reilly, 1st edition.

[Toledo, 2005] Toledo, M. D. G. (2005). A comparison in cluster validation techniques. Master thesis.

[Troncy and Issac, 2002] Troncy, R. and Issac, A. (2002). Doe : une mise en œuvre d'une méthode de structuration différentielle pour les ontologies. In *Actes des journées francophones d'Ingénierie des Connaissances (IC'2002)*, pages 63–74.

[Tseng and Lin, 2004] Tseng, M.-C. and Lin, W.-Y. (2004). Maintenance of generalized association rules with multiple minimum supports. *Intelligent Data Analysis*, 8(4) :417–436.

[Uschold, 1995] Uschold, M. (1995). Towards a methodology for building ontologies. Technical report.

[Uschold, 1996] Uschold, M. (1996). Building ontologies : towards a unified methodolgy. Technical report.

[Uschold and Grüninger, 1996] Uschold, M. and Grüninger, M. (1996). Ontologies : principles, methods, and applications. *Knowledge Engineering Review*, 11(2) :93–155.

[Vale and Pellegrino, 2000] Vale, S. and Pellegrino, M. (2000). The metadata problem in a European Context, Eurostat. In *Proceedings of Workshop on Statistical Metadata, Working*, number 11.

[Vardaki, 2004] Vardaki, M. (2004). Metadata for symbolic objects. *Electronic Journal of Symbolic Data Analysis*, 2. Disponible à http ://www.jsda.unina2.it.

[Vassiliadis and Stavrakas, 1998] Vassiliadis, P. and Stavrakas, Y. (1998). Different perspectives of metadata for web-based information systems. In *Proceedings of the* 11^{th} *ERCIM Database Research Group Workshop*. citeseer.ist.psu.edu/14121.html.

[W3C, 2006] W3C (2006). World Wide Web. http ://www.w3c.org/.

[Warwick et al., 1994] Warwick, J. N., Tracy, L. S., Simon, R. T., Andrew, J. C., and Wes, B. F. (1994). The population biology of abalone (haliotis-species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait. Technical report, Sea Fisheries Division. Technical Report No. 48 (ISSN 1034-3288).

[Weaver et al., 2001] Weaver, M., Declcambre, L., and Maier, D. (2001). A superimposed architecture for enhanced metadata. In *Proceedings of the* 3^{th} *DELOS Network of Excellence Workshop*. Disponible à l'adresse suivante : http ://citeseer.ist.psu.edu/mathew01superimposed.html.

[WebOde, 2002] WebOde (2002). WEBODE Home Page. http ://delicias.dia.fi.upm.es/webODE/.

[Wielinga et al., 1994] Wielinga, B., Hans Akkermans, Y., Hassan, H., Olsson, O., Orsvarn, K., Schreiber, G., Terpstra, P., Van de Velde, W., and Wells, S. (1994). Expertise model definition document. Technical report, University of Amsterdam. ESPRIT Project P5248, KADSII, Document Id. KADSII/M2/UvA/026/5.0.

[Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1) :67–82.

[XMI, 2006] XMI (2006). XML Metadata Interchange : an adopted standard of the OMG. http ://www.omg.org/technology/documents/formal/xmi.htm.

[XML, 2002] XML (2002). eXtended Markup Language Specification. http ://www.w3.org/TR/REC-xml.

[XQJ, 2006] XQJ (2006). XQJ Home Page. http ://www.datadirect.com/developer/xquery.

[Yatskiv and Gusarova, 2004] Yatskiv, I. and Gusarova, L. (2004). The methods of cluster analysis results validation. In *Proceedings of the International Conference RelStat'04*, pages 75–80.

[Zaït and Messatfa, 1997] Zaït, M. and Messatfa, H. (1997). A comparative study of clustering methods. *Future Generation Computer System*, 13 :149–159.

[Zhang et al., 2003] Zhang, L., Tang, C., Zhang, A., and Ramanathan, M. (2003). VizCluster and Its Application on Clustering Gene Expression Data. *Distributed and Parallel Databases*, 13(1) :73–97.

[Zweigenbaum, 1999] Zweigenbaum, P. (1999). Encoder l'information médicale : des terminologies aux systèmes de représentation des connaissances. *Innovation Stratégique en Information de Santé*, 2(3) :27–47.

Annexes

Annexe A

Diagramme de classes de notre application

Nous présentons dans cette annexe les classes et interfaces que nous avons utilisées pour concevoir notre modèle d'interprétation des résultats de classification automatique. Nous commençons par présenter la manière avec laquelle nous avons modélisé l'acquisition des métadonnées en vue de la création du fichier de métadonnées. Ensuite nous montrons l'utilisation du modèle d'architecture MVC dans cette thèse.

A.1 Modélisation et acquisition du fichier de métadonnées

Il convient de rappeler que notre objectif tout le long du processus de modélisation a été de fournir une application extensible, robuste et facilement maintenable. Ce qui suppose d'appliquer un certain nombre de principes de *programmation orientée objet*. Ainsi, nous préférons, dans la limite du possible, *la composition* à l'héritage. De même, nous avons essayé d'éviter de *dupliquer* du code inutilement en cherchant à factoriser les codes communs. Aussi, nous avons privilégié *l'ajout de code* à sa modification et nous avons essayé de mettre en place des classes *faiblement couplées* afin d'en faciliter la maintenance. Voici quelques-unes des classes utilisées pour la réalisation de ce fichier de métadonnées :

- *Entete :* qui gère les informations relatives à la description du cadre de l'étude ; il s'agit principalement des informations tirées de la norme Dublin Core ;
- *Module :* qui gère les informations des modules de fouille de données, par exemple les *paramètres* de la méthode ;
- *Individu :* qui gère les informations relatives aux données qui sont l'objet de la classification ;
- *Classe :* décrit l'ensemble des résultats de la classification (en l'occurrence les classes) ;
- *Fichier :* qui gère la decription des fichiers de données, résultats, etc.

- *Criterion :* qui permet la gestion des critères généraux dans les méthodes de fouille de données ;
- *Variables :* gère l'ensemble des variables décrivant les données ou participant à leur classification ;
- *Matrice :* est une classe qui joue un rôle stratégique dans notre application car c'est elle qui gère les ***valeurs des critères généraux*** à utiliser dans les différents scénarios d'interprétation. Ainsi, dans nos différentes expérimentations, nous avons utilisé les valeurs de ces critères pour chaque classe et pour chaque variable.
- *MetaMatrice :* cette classe est composée par l'ensemble des classes que nous venons de décrire et d'autres classes qui n'ont pas été abordées ici.

L'un des *patrons de conception* utilisés dans cette modélisation est le *Patron de méthode*. Le ***Patron de méthode*** définit les étapes d'un algorithme et laisse la latitude aux sous-classes de fournir l'implémentation d'une ou plusieurs de ses étapes sans modifier sa structure. Il s'agit d'*encapsuler les comportements interchangeables* et d'utiliser la délégation pour décider du comportement à utiliser. Ainsi, à la suite de notre première modélisation, nous avons remarqué qu'il existait un ensemble de méthodes communes (mais dont l'implémentation est différente suivant les classes) à plusieurs classes. C'est le cas des méthodes d'écriture (*writeXml*) des données dans les fichiers *XML*. En effet, les balises utilisées dans les différentes composantes de notre fichier de métadonnées sont très différentes. En utilisant ce patron de conception, nous avons créé une *classe abstraite* contenant le *Patron de méthode* et les versions abstraites des opérations primitives utilisées (cf. *methodeAlgorithme1 et methodeAlgorithme1* dans la figure A.1) dans ce patron de méthode. Ensuite, il ne reste plus qu'à implémenter ces opérations primitives dans les sous-classes (classes concrètes). Grâce à ce *patron*, notre algorithme ne réside plus que dans une seule classe et c'est uniquement dans celle-ci qu'ont lieu les modifications éventuelles du code. Ce *patron* fournit un cadre dans lequel nous pouvons insérer de nouvelles classes utilisant certaines propriétés de notre algorithme. Ces nouvelles classes devront implémenter les opérations primitives abstraites définies dans le *Patron de méthode*.

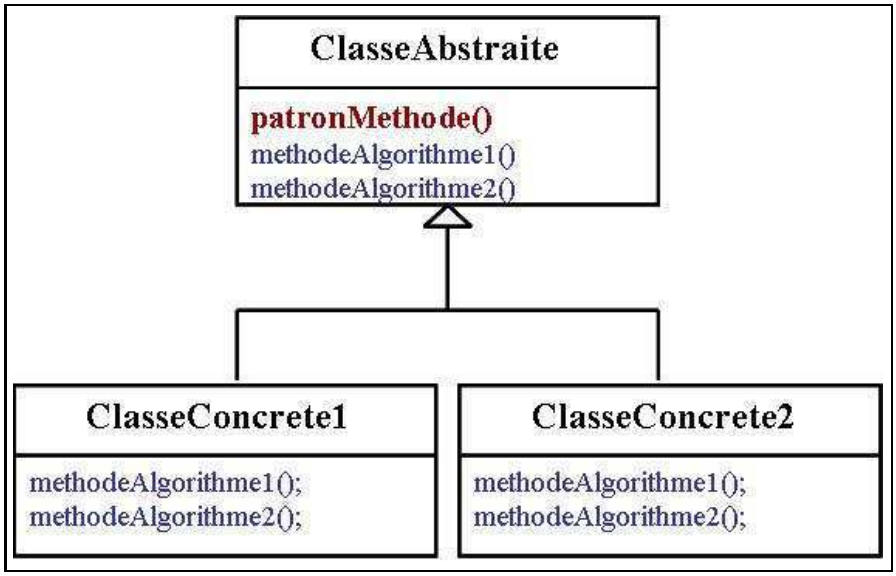

FIG. *A.1 - Schéma du Patron de méthode*

L'autre *patron de conception* utilisé dans cette application est *Stratégie*. Son utilisation s'explique par le fait que nous ne souhaitons pas définir une norme rigide de la description des critères généraux. Par exemple, durant nos expérimentations, nous avons pu travailler avec des valeurs de *critères généraux* issus de la description des classes par d'autres classes, des variables par des classes, ainsi de suite. A la différence de ce qui est fait dans le logiciel WEKA (cf. section 3.1.4) où les critères d'interprétation se basent uniquement sur les valeurs contenues dans la ***matrice de confusion*** (description des classes par des classes), l'utilisation de ce *patron de conception* permet de prendre en compte de nouveaux critères généraux qui ne soient pas uniquement basés sur la matrice de confusion. Dans la figure A.2, nous avons une description schématique du patron *Stratégie*. Il s'agit de définir les comportements (ou d'encapsuler ce qui varie) dans des interfaces (cf. partie 2 de la figure A.2) et d'implémenter ces interfaces par une classe abstraite contenant les descriptions statiques des classes concrètes (cf. partie 1 de la figure A.2). L'un des avantages est relatif à la possibilité de faire le choix, de manière dynamique (c'est-à-dire à l'exécution), du type des critères généraux à utiliser dans le fichier de métadonnées. En effet, le patron *Stratégie* permet de définir une famille d'algorithmes en encapsulant chacun d'eux et en les rendant interchangeables. Il permet ainsi à l'algorithme de varier indépendamment des clients qui l'utilisent. Ce qui garantit une certaine flexibilité lors de l'ajout de nouveaux critères généraux. Leur prise en compte est transparente aux classes concrètes qui les implémentent. De plus, nous avons une interface commune (cf. à l'interface *CriteresGeneraux* dans la figure A.2) à laquelle nous nous adressons sans nous préoccuper des implémentations éventuelles de ces méthodes.

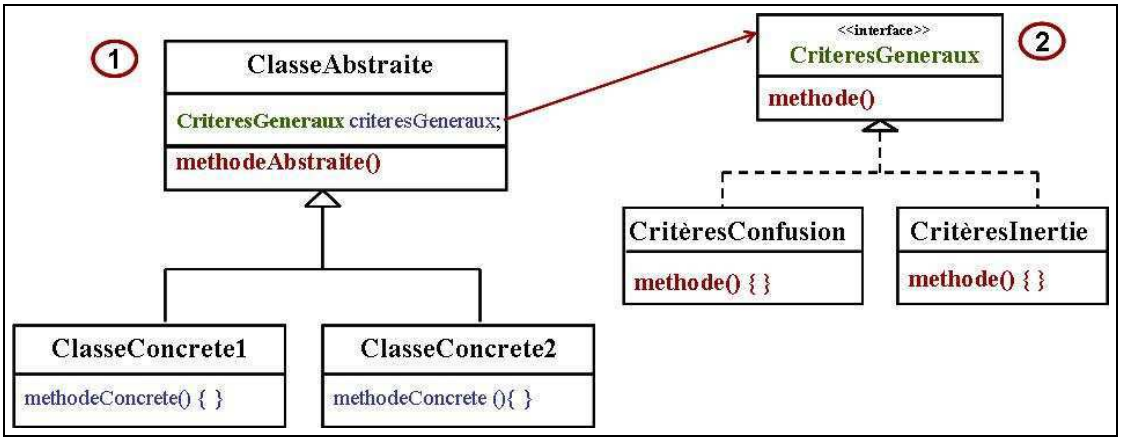

FIG. *A.2 - Schéma du patron de conception Stratégie*

A.2 Utilisation de l'architecture Modèle Vue Contrôleur (MVC)

L'architecture MVC est un ensemble de patrons de conceptions, nous présentons l'utilisation de cette architecture, à travers les patrons de conception utilisés, dans notre travail. Il convient de rappeler que l'architecture MVC est composée par :

- *Modèle :* qui contient les états, les données et la logique applicative. Il s'agit de la partie de notre application qui produit le fichier de métadonnées à interpréter ;
- *Vue :* fournit une présentation du modèle. D'une manière générale, elle reçoit directement du modèle l'état et les données qu'elle doit afficher. Cette vue est matérialisée par notre interface utilisateur ;
- *Contrôleur :* gère les entrées des utilisateurs et déterminent ce qu'elles signifient pour le modèle. Dans notre travail, cette composante est matérialisée par les classes qui gèrent les scénarios d'interprétation, qu'il s'agisse des classes de gestion des requêtes XQUERY ou des classes gérant notre ontologie de la classification.

Le premier patron de conception que nous avons utilisé dans cette architecture MVC est le patron *Stratégie* abordé dans la section précédente. Nous expliquons simplement que ce patron de conception a été utilisé pour modéliser le fonctionnement de la composante *Contrôleur*. En effet, la *Vue* et le *Contrôleur* vont mettre en oeuvre ce patron de conception dans la mesure où la *Vue* est un objet configuré avec une stratégie et que le *Contrôleur* fournit cette stratégie. De ce fait, la *Vue* n'est concernée que par les aspects visuels de l'application. Elle délègue ainsi au *Contrôleur* toutes les décisions sur le comportement de l'interface. Ainsi, lorsqu'un utilisateur choisit un scénario d'interprétation, ce dernier est interprété par le *Contrôleur* qui délègue la tâche d'abord aux classes de traitement des flux XML. Puis nous utilisons les classes de *gestion de notre ontologie* afin de choisir les données et la formule du scénario d'interprétation à utiliser.

Le second *patron de conception* utilisé est celui qui nous permet de modéliser la composante *Modèle*, il s'agit de l'*Observateur*. Ce patron permet de définir une relation entre plusieurs objets de type *un-à-plusieurs*. Ainsi, lorsqu'un objet change d'état, tous ceux qui en dépendent sont automatiquement notifiés et mis à jour. Une description schématique de ce patron de conception est donnée dans la figure A.3. En utilisant ce patron de conception, nous garantissons une totale indépendance entre notre modèle et l'interface utilisée pour l'afficher. En effet, les données sont contrôlées par l'*Observable*, ce qui évite que plusieurs objets contrôlent la même donnée.

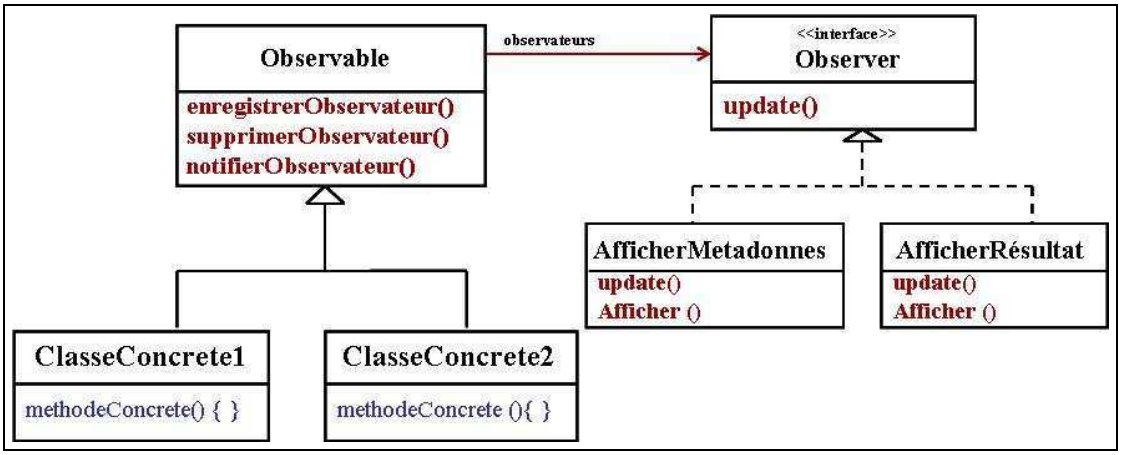

FIG. *A.3 - Schéma du patron de conception Observateur*

Annexe B

Programme de transformation XSLT des fichiers de métadonnées

Dans cette annexe, nous présentons le code de notre programme XSLT qui fait la transformation de nos fichiers de métadonnées (XML) en des fichiers HTML plus lisibles.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- by Abdourahamane Balde (inria) -->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="html" encoding="iso-8859-1"/>
        <xsl:template match="/">
                <HTML>
                        <body>
                                <xsl:apply-templates/>
                        </body>
                </HTML>
        </xsl:template>
        <!--****************************BLOC DUBLIN CORE HEADER ***************************-->
        <!-- cette section permet d'afficher les éléments de l'entête tout en éliminant les
        valeurs nulles  -->
        <xsl:template match="MLC/GlobalMetadata/Header">
                        <h2 align="center" id="Header">
                                <font color="blue" size="5">HEADER</font>
                        </h2>
        <table border="2" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
                <xsl:if test="not(Title='UNKNOWN')">
                        <tr>
                                <td bgcolor="silver">TITLE</td>
                                <td>
                                        <font color="blue" size="2">
                                                <xsl:apply-templates select="Title"/>
                                        </font>
                                </td>
                        </tr>
                        </xsl:if>
                        <xsl:if test="not(Author='UNKNOWN')">
                        <tr>
                                <td bgcolor="silver">CREATOR</td>
                                        <td>
                                        <font color="blue" size="2">
                                                <xsl:apply-templates select="Author"/>
```

```
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Date='UNKNOWN')">
        <tr>
                <td bgcolor="silver">DATE</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Date"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Contributor='UNKNOWN')">
        <tr>
                <td bgcolor="silver">CONTRIBUTOR</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Contributor"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Rights='UNKNOWN')">
        <tr>
                <td bgcolor="silver">RIGHTS</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Rights"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Subject='UNKNOWN')">
        <tr>
                <td bgcolor="silver">SUBJECT</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Subject"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Description='UNKNOWN')">
        <tr>
                <td bgcolor="silver">DESCRIPTION</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Description"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Editor='UNKNOWN')">
        <tr>
                <td bgcolor="silver">EDITOR</td>
```

```
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Editor"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Resource='UNKNOWN')">
100     <tr>
                <td bgcolor="silver">RESOURCE</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Resource"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Format='UNKNOWN')">
110     <tr>
                <td bgcolor="silver">FORMAT</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Format"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Source='UNKNOWN')">
120     <tr>
                <td bgcolor="silver">SOURCE</td>
                        <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Source"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
        <xsl:if test="not(Language='UNKNOWN')">
130             <tr>
                <td bgcolor="silver">LANGUAGE</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Language"/>
                        </font>
                </td>
                </tr>
        </xsl:if>
        <xsl:if test="not(Relation='UNKNOWN')">
140     <tr>
                <td bgcolor="silver">RELATION</td>
                <td>
                        <font color="blue" size="2">
                                <xsl:apply-templates select="Relation"/>
                        </font>
                </td>
        </tr>
        </xsl:if>
```

```
                        <xsl:if test="not(Coverage='UNKNOWN')">
                        <tr>
                                <td bgcolor="silver">COVERAGE</td>
                                <td>
                                <font color="blue" size="2">
                                        <xsl:apply-templates select="Coverage"/>
                                </font>
                                </td>
                        </tr>
                        </xsl:if>
        </table>
        </xsl:template>
        <!--************************BLOC MODULE ****************************************-->
        <!-- cette section permet d'afficher les éléments de la composante Module relative
        à la description du module utilisé: les paramètres de la méthode, etc.  -->
        <xsl:template match="MLC/GlobalMetadata/Module">
                        <h2 align="center" id="Module">
                                <font color="blue" size="5">MODULE</font>
                        </h2>
        <table border="3" bgcolor="#ccffaa" cellspacing="2" cellpadding="8" align="center">
                        <tr>
                                <td bgcolor="silver"> IDENTIFIER </td>
                                <td align="center">
                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="Identifier"/>
                                        </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">NAME</td>
                                <td align="center">
                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="Name"/>
                                        </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">VERSION</td>
                                <td align="center">
                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="Version"/>
                                        </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">COMMENT</td>
                                <td align="center">
                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="Comment"/>
                                        </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">PARAMETERS</td>
                                <td align="center">
        <table border="5" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
                                <thead align="center" bgcolor="silver">
```

```
                                        <th>IDENTIFIER</th>
                                        <th>NAME</th>
                                        <th>VALUE</th>
                                        <th>COMMENT</th>
                                </thead>
                                <xsl:for-each select="Parameters/Parameter">
                                        <tr align="center">
                                                <xsl:for-each select="child::*">
                                                <td align="center">
                                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="self::node()"/>
                                                        </font>
                                                </td>
                                                </xsl:for-each>
                                        </tr>
                                </xsl:for-each>
        </table>
                                </td>
                        </tr>
        </table>
</xsl:template>
<!--**************************BLOC FILES ************************************************-->
<!-- Cette section permet d'afficher les informations sur les fichiers de données, résultats, etc.-->
<xsl:template match="MLC/GlobalMetadata/Files">
                        <h2 align="center" >
                                        <font color="blue" size="5">LIST OF FILES</font>
                        </h2>
        <table border="5" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
                        <thead align="center" bgcolor="silver">
                                <th>NUMERO</th>
                                <th>IDENTIFIER</th>
                                <th>NAME</th>
                                <th>COMMENT</th>
                        </thead>
                        <xsl:for-each select="File">
                        <xsl:variable name="var" select="Name"/>
                                <tr align="center">
                                        <xsl:for-each select="child::*">
                                                <td align="center">
                                                        <font color="blue" size="3">
                                                                <a href ="{$var}">
                                                <xsl:apply-templates select="self::node()"/>
                                                                </a>
                                                        </font>
                                                </td>
                                        </xsl:for-each>
                                </tr>
                        </xsl:for-each>
        </table>
</xsl:template>
<!--**************************BLOC CRITERES GLOBAUX ************************************************-->
<!-- Cette section traite de la manière d'afficher les critères globaux d'interprétation utilisés-->
<xsl:template match="MLC/Experiences/Experience">
                <h2 align="center" >
                                <font color="blue" size="5">LIST OF EXPERIENCES</font>
                </h2>
                <table border="5" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
```

```
                        <tr>
                                <td bgcolor="silver"> IDENTIFIER </td>
                                <td align="center">
                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="Numero"/>
                                        </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">NAME</td>
                                <td align="center">
                                                <font color="blue" size="3">
                                                <xsl:apply-templates select="Name"/>
                                                </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">COMMENT</td>
                                <td align="center">
                                        <font color="blue" size="3">
                                                <xsl:apply-templates select="Comment"/>
                                        </font>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">GLOBAL CRITERIONS</td>
                                <td align="center">
                <table border="5" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
                        <thead align="center" bgcolor="silver">
                                <th>NAME</th>
                                <th>VALUE</th>
                                <th>COMMENT</th>
                        </thead>
    <xsl:for-each select="/MLC/Experiences/Experience/ComposantExperience/GlobalCriterionTab/GlobalCriterion">
                        <tr align="center">
                                <xsl:for-each select="child::*">
                                        <td align="center">
                                                <font color="blue" size="3">
                                                        <xsl:apply-templates select="self::node()"/>
                                                </font>
                                        </td>
                                </xsl:for-each>
                        </tr>
    </xsl:for-each>
                </table>
                        </td>
                </tr>
                        <tr>
                                <td bgcolor="silver">VARIABLES</td>
                                <td align="center">
                <table border="5" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
                        <thead align="center" bgcolor="silver">
                                <th>NUMERO</th>
                                <th>IDENTIFIER</th>
                                <th>NAME</th>
                                <th>TYPE</th>
                                <th>USAGE</th>
```

```
320                         <th>COMMENT</th>
                        </thead>
                <xsl:for-each select="/MLC/Experiences/Experience/ComposantExperience/Variables/Variable">
                    <xsl:if test="not(name='null')">
                        <tr align="center">
                                <xsl:for-each select="child::*">
                                        <td align="center">
                                                <font color="blue" size="3">
                                                        <xsl:apply-templates select="self::node()"/>
                                                </font>
330                                     </td>
                                </xsl:for-each>
                        </tr>
                    </xsl:if>
                </xsl:for-each>
                </table>
                                </td>
                        </tr>
                        <tr>
                                <td bgcolor="silver">CLUSTERS</td>
340                             <td align="center">
                <table border="5" cellspacing="2" cellpadding="8" bgcolor="#ccffaa" align="center">
                                <thead align="center" bgcolor="silver">
                                        <th>NUMERO</th>
                                        <th>IDENTIFIER</th>
                                        <th>NAME</th>
                                        <th>CARDINALITY</th>
                                        <th>COMMENT</th>
                                </thead>
                <xsl:for-each select="/MLC/Experiences/Experience/ComposantExperience/Classes/Classe">
350                         <xsl:if test="not(name='null')">
                                <tr align="center">
                                        <xsl:for-each select="child::*">
                                        <td>
                                                <font color="blue" size="3">
                                                        <xsl:apply-templates select="self::node()"/>
                                                </font>
                                        </td>
                                        </xsl:for-each>
                                </tr>
360                         </xsl:if>
                </xsl:for-each>
                                </table>
                                </td>
                                </tr>
                <xsl:for-each select="/MLC/Experiences/Experience/ComposantExperience/CRITERIA/Criteria">
                                        <tr>
                                                <td bgcolor="silver"> IDENTIFIER </td>
                                                <td align="center">
                                                <font color="blue" size="3">
370                                                     <xsl:apply-templates select="Numero"/>
                                                </font>
                                                </td>
                                        </tr>
                                        <tr>
                                                <td bgcolor="silver">NAME</td>
                                                <td align="center">
```

```
                                                        <font color="blue" size="3">
                                                                <xsl:apply-templates select="Name"/>
                                                        </font>
380                                                 </td>
                                            </tr>
                                            <tr>
                                                    <td bgcolor="silver">COMMENT</td>
                                                    <td align="center">
                                                            <font color="blue" size="3">
                                                            <xsl:apply-templates select="Comment"/>
                                                            </font>
                                                    </td>
                                            </tr>
390                                         <tr>
                                                    <td bgcolor="silver">CRITERIA MATRIX</td>
                                                    <td align="center">
                                            <xsl:for-each select="cla_var/ClasseVariableMatrix">
        <table border="5" cellspacing="2" cellpadding="10" bgcolor="#ccffaa" align="center">
                                    <thead bgcolor="silver" align="center">
                                                    <th/>
                <xsl:for-each select="/MLC/Experiences/Experience/ComposantExperience/Classes/Classe">
                                                        <th align="center">
                                                                <font color="red" size="4">
400                                                     <xsl:value-of select="Name"/>
                                                                </font>
                                                        </th>
                                                </xsl:for-each>
                                            </thead>
                                            <xsl:for-each select="LigMat">
                                                    <tr align="center">
                                                    <td>
                                                            <font color="red" size="6">
                                                            <xsl:value-of select="string(position())"/>
410                                                         </font>
                                                    </td>
                                                            <xsl:for-each select="Value">
                                                            <td align="center">
                                                    <xsl:call-template name="matrix_var">
                                                    <xsl:with-param name="var_num2" select="position()"/>
                                                    </xsl:call-template>
                                                            </td>
                                                    </xsl:for-each>
                                                    </tr>
420                                         </xsl:for-each>
                                            </table>
                                            </xsl:for-each>
                                            </td>
                                            </tr>
                                    </xsl:for-each>
                            </table>
        </xsl:template>
    <!--**************************TEMPLATES SUPPLEMENTAIRES ***************************************-->
    <!-- ce module permet d'afficher les données d'une matrice -->
430     <xsl:template name="matrix_var">
                <xsl:param name="var_num2"/>
                <xsl:variable name="no_var" select="$var_num2"/>
                                <font color="blue" size="2">
```

```
                                    <xsl:value-of select="ContinuValue"/>
                                </font>
        </xsl:template>
</xsl:stylesheet>
```

Annexe C

Description en langage OWL de notre ontologie

Cette annexe présente la description de notre ontologie décrite en langage OWL.

```
1  <?xml version="1.0"?>
   <rdf:RDF
       xmlns="http://www.owl-ontologies.com/metadata.owl#"
       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
       xmlns:owl="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.owl-ontologies.com/metadata.owl">
     <owl:Ontology rdf:about=""/>
     <owl:Class rdf:ID="SupervisedMethod">
       <rdfs:subClassOf>
         <owl:Class rdf:ID="Method"/>
       </rdfs:subClassOf>
     </owl:Class>
     <owl:Class rdf:ID="VariableRole">
       <owl:equivalentClass>
         <owl:Restriction>
           <owl:someValuesFrom>
             <owl:Class rdf:ID="VariableType"/>
           </owl:someValuesFrom>
           <owl:onProperty>
             <owl:ObjectProperty rdf:ID="hasType"/>
           </owl:onProperty>
         </owl:Restriction>
       </owl:equivalentClass>
       <rdfs:subClassOf>
         <owl:Class rdf:ID="DataMiningConcept"/>
       </rdfs:subClassOf>
     </owl:Class>
     <owl:Class rdf:ID="ECDConcept"/>
     <owl:Class rdf:ID="InternalCriterion">
```

```
    <owl:disjointWith>
      <owl:Class rdf:ID="ExternalCriterion"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="RelativeCriterion"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Criterion"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="DomainConcept">
    <owl:disjointWith>
      <owl:Class rdf:about="#DataMiningConcept"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="DiscriminantVariable">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="ResultInterpretingConcept"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Method">
    <owl:disjointWith>
      <owl:Class rdf:about="#Criterion"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DataMiningConcept"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="OuputStructure">
    <rdfs:subClassOf>
      <owl:Class>
        <owl:oneOf rdf:parseType="Collection">
          <OuputStructure rdf:ID="Hierarchical"/>
          <OuputStructure rdf:ID="Density"/>
          <OuputStructure rdf:ID="Grid"/>
          <OuputStructure rdf:ID="Partition"/>
        </owl:oneOf>
      </owl:Class>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DataMiningConcept"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Individual">
    <owl:disjointWith>
      <owl:Class rdf:about="#Criterion"/>
    </owl:disjointWith>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#DataMiningConcept"/>
```

```
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#RelativeCriterion">
  <owl:disjointWith>
    <owl:Class rdf:about="#ExternalCriterion"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#InternalCriterion"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Criterion"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#ResultInterpretingConcept">
  <rdfs:subClassOf rdf:resource="#ECDConcept"/>
</owl:Class>
<owl:Class rdf:ID="SimilarityMeasure">
  <owl:disjointWith>
    <owl:Class rdf:ID="DissimilarityMeasure"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Measure"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Parameter">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#DataMiningConcept"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#DissimilarityMeasure">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Measure"/>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#SimilarityMeasure"/>
</owl:Class>
<owl:Class rdf:ID="MetadataTable">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Input"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#VariableType">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Input"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#DataMiningConcept"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <VariableType rdf:ID="Binaire"/>
```

```
        <VariableType rdf:ID="Categorique"/>
        <VariableType rdf:ID="Interval"/>
        <VariableType rdf:ID="Mixte"/>
        <VariableType rdf:ID="Nominal"/>
        <VariableType rdf:ID="Numerique"/>
        <VariableType rdf:ID="OrdinalContinu"/>
        <VariableType rdf:ID="OrdinalDiscrete"/>
        <VariableType rdf:ID="RatioScaled"/>
      </owl:oneOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#DataMiningConcept">
  <owl:disjointWith rdf:resource="#DomainConcept"/>
  <rdfs:subClassOf rdf:resource="#ECDConcept"/>
</owl:Class>
<owl:Class rdf:ID="UnSupervisedMethod">
  <rdfs:subClassOf rdf:resource="#Method"/>
</owl:Class>
<owl:Class rdf:ID="PredictiveVariable">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Input"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#VariableRole"/>
</owl:Class>
<owl:Class rdf:ID="Clustering">
  <rdfs:subClassOf rdf:resource="#UnSupervisedMethod"/>
</owl:Class>
<owl:Class rdf:ID="Outlier">
  <rdfs:subClassOf rdf:resource="#Individual"/>
  <rdfs:subClassOf rdf:resource="#ResultInterpretingConcept"/>
</owl:Class>
<owl:Class rdf:about="#Criterion">
  <owl:disjointWith rdf:resource="#Individual"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Measure"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:ID="Thresold"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#ResultInterpretingConcept"/>
  <owl:disjointWith rdf:resource="#Method"/>
</owl:Class>
<owl:Class rdf:about="#Measure">
  <owl:disjointWith rdf:resource="#Criterion"/>
  <rdfs:subClassOf rdf:resource="#DataMiningConcept"/>
</owl:Class>
<owl:Class rdf:ID="DataTable">
  <rdfs:subClassOf>
```

```
    <owl:Class rdf:about="#Input"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#Thresold">
  <owl:disjointWith rdf:resource="#Criterion"/>
  <rdfs:subClassOf rdf:resource="#ResultInterpretingConcept"/>
</owl:Class>
<owl:Class rdf:ID="ActiveVariable">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Input"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#VariableRole"/>
</owl:Class>
<owl:Class>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Method"/>
    <owl:Class rdf:about="#VariableRole"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:about="#ExternalCriterion">
  <rdfs:subClassOf rdf:resource="#Criterion"/>
  <owl:disjointWith rdf:resource="#InternalCriterion"/>
  <owl:disjointWith rdf:resource="#RelativeCriterion"/>
</owl:Class>
<owl:Class rdf:about="#Input">
  <rdfs:subClassOf rdf:resource="#DataMiningConcept"/>
</owl:Class>
<owl:Class rdf:ID="SupplementaryVariable">
  <rdfs:subClassOf rdf:resource="#Input"/>
  <rdfs:subClassOf rdf:resource="#VariableRole"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasGlobalCriteria">
  <rdfs:range rdf:resource="#ExternalCriterion"/>
  <rdfs:domain rdf:resource="#Method"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Describe">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="isDescribedBy"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Method"/>
  <rdfs:domain rdf:resource="#Parameter"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#isDescribedBy">
  <rdfs:range rdf:resource="#Parameter"/>
  <rdfs:domain rdf:resource="#Method"/>
  <owl:inverseOf rdf:resource="#Describe"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#hasType">
  <rdfs:range rdf:resource="#VariableType"/>
```

```
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#ActiveVariable"/>
          <owl:Class rdf:about="#PredictiveVariable"/>
          <owl:Class rdf:about="#SupplementaryVariable"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasResult">
    <rdfs:range rdf:resource="#OuputStructure"/>
    <rdfs:domain rdf:resource="#Method"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="hasCriteria">
    <rdfs:domain rdf:resource="#VariableRole"/>
    <rdfs:range rdf:resource="#Criterion"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="Identifier">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#VariableRole"/>
          <owl:Class rdf:about="#Parameter"/>
          <owl:Class rdf:about="#Method"/>
          <owl:Class rdf:about="#Criterion"/>
          <owl:Class rdf:about="#Individual"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Usage">
    <rdfs:domain rdf:resource="#VariableRole"/>
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >Active</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >Illustrative</rdf:first>
              <rdf:rest rdf:parseType="Resource">
                <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >Supplementary</rdf:first>
              </rdf:rest>
            </rdf:rest>
```

```
        </rdf:rest>
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >Predictive</rdf:first>
      </owl:oneOf>
    </owl:DataRange>
  </rdfs:range>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Name">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#Name"/>
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Criterion"/>
        <owl:Class rdf:about="#Method"/>
        <owl:Class rdf:about="#Parameter"/>
        <owl:Class rdf:about="#VariableRole"/>
        <owl:Class rdf:about="#Individual"/>
        <owl:Class rdf:about="#Measure"/>
        <owl:Class rdf:about="#Thresold"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Valeur">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Parameter"/>
        <owl:Class rdf:about="#Criterion"/>
        <owl:Class rdf:about="#Thresold"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Comment">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Criterion"/>
        <owl:Class rdf:about="#Method"/>
        <owl:Class rdf:about="#Parameter"/>
        <owl:Class rdf:about="#VariableRole"/>
        <owl:Class rdf:about="#Individual"/>
        <owl:Class rdf:about="#Measure"/>
        <owl:Class rdf:about="#Input"/>
        <owl:Class rdf:about="#DiscriminantVariable"/>
        <owl:Class rdf:about="#Outlier"/>
        <owl:Class rdf:about="#Thresold"/>
```

```
          </owl:unionOf>
        </owl:Class>
      </rdfs:domain>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>
    <owl:DatatypeProperty rdf:ID="Type">
      <rdfs:domain rdf:resource="#VariableRole"/>
      <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    </owl:DatatypeProperty>
    <VariableRole rdf:ID="Variable_1"/>
    <Parameter rdf:ID="Parametre_4"/>
    <VariableRole rdf:ID="Variable_2"/>
    <owl:Thing rdf:ID="in2"/>
    <Parameter rdf:ID="Parametre_5"/>
    <owl:Thing rdf:ID="in1"/>
    <Parameter rdf:ID="Parametre_1"/>
    <owl:Thing rdf:ID="individu_3"/>
    <owl:Thing rdf:ID="individu_1"/>
    <owl:Thing rdf:ID="no_comment"/>
    <owl:Thing rdf:ID="in3"/>
    <owl:Thing rdf:ID="individu_2"/>
    <Parameter rdf:ID="Parametre_3"/>
    <VariableRole rdf:ID="Variable_3"/>
    <VariableRole rdf:ID="Variable_5"/>
    <Parameter rdf:ID="Parametre_2"/>
    <VariableRole rdf:ID="Variable_4"/>
  </rdf:RDF>

<!--Created with Protege (with OWL Plugin 2.2, Build 311)  http://protege.stanford.edu -->
```