كلية العلوم و التقنيات فاس
ⵜⴰⵖⵚⵍⵓⵍⵜ ⵏ ⵜⵜⵓⵙⵙⵏⵉ ⴷ ⵜⵓⵏⵙⵙⵉⵔⵉ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵙⵉⴷⵉ ⵎⵓⵃⴰⵎⵎⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰⵀ
Université Sidi Mohamed Ben Abdellah

Centre d'Etude Doctorale : Sciences et Techniques de l'Ingénieur
Laboratoire : Systèmes Intelligents et Applications

THESE

Présentée pour l'obtention du diplôme de

DOCTORAT

Formation doctorale : Sciences de l'Ingénieur, Sciences Physiques, Mathématiques et Informatique
Spécialité : Informatique
Par :

LAMIAE DEMRAOUI

Titre :

Capitalisation des Connaissances dans un Processus Décisionnel Multi-experts basé sur le Standard CWM

Soutenue le **Samedi 17 Février 2018** à 10h à la Faculté des Sciences et Techniques de Fès

Devant le jury :

Azeddine ZAHI	FST - Fès	Président
Mahmoud NASSAR	ENSIAS - Rabat	Rapporteur
Mohammed OUMSIS	EST - Salé	Rapporteur
Saîd OUATIK EL ALAOUI	FS Dhar Mehrez - Fès	Rapporteur
Hicham BEHJA	ENSEM - Casablanca	Examinateur
Rachid BEN ABBOU	FST - Fès	Directeur de Thèse
El Moukhtar ZEMMOURI	ENSAM - Meknès	Invité

A mes chers parents avec tout mon amour et pour tout ce qu'ils m'ont appris.
A mon frère que j'adore.

Remerciements

Je souhaite remercier en premier lieu mon directeur de thèse, **M. Rachid BEN ABBOU**, Professeur et Chef du département Informatique de la Faculté des Sciences et Techniques de Fès, pour la confiance qu'il m'a témoignée, pour m'avoir initiée à la recherche et diriger avec compétence mes travaux, de mémoire de Licence, Master à la thèse de doctorat. Nonobstant, sa relecture finale méticuleuse du rapport de ma thèse m'a sans aucun doute permis de préciser mon propos. Je lui suis également reconnaissante pour le temps conséquent qu'il m'a accordée, ses qualités pédagogiques et scientifiques, sa franchise, sa sympathie, et de son efficacité certaine que je n'oublierai jamais. J'ai beaucoup appris à ses côtés et je lui adresse ma gratitude pour tout cela.

J'adresse de chaleureux remerciements à **M. Hicham BEHJA**, Professeur et Vice-directeur de l'ENSEM de Casablanca, pour son encadrement et pour ses suggestions qui ont été prépondérants pour la bonne réussite de cette thèse. Ses conseils et son suivi avisés m'ont été toujours d'un grand secours. Je le remercie aussi pour son accueil au sein du laboratoire Recherche en Ingénierie à l'ENSEM de Casablanca, ainsi pour l'hospitalité dont il a fait preuve envers moi lors de mon séjour à l'ENSEM. Son énergie, sa dynamique et son expertise ont été des éléments moteurs pour moi. J'ai pris un grand plaisir à travailler avec lui.

Je suis particulièrement redevable envers **M. EL Moukhtar ZEMMOURI**, Professeur à l'ENSAM de Meknès. Je tiens à le remercier pour son encadrement et pour son attention à tout instant sur mes travaux. Je le remercie également pour son accueil chaleureux au sein du laboratoire MMI de l'ENSAM de Meknès, pour ses conseils précieux et son appui, à la fois, constant et amical. J'ai découvert en lui un modèle de rigueur et de générosité. Par les directives qu'il m'a prodiguée tout au long de cette période de thèse, il a su patiemment me guider et m'insuffler le goût pour la recherche et le sens du partage. Par l'exigence qu'il s'applique à lui-même, il a fortement influencé ma méthode de travail et mes recherches futures en seront, sans nul doute, grandement conditionnées.

J'associe mes remerciements à **Mme. Brigitte TROUSSE**, professeur chercheur à INRIA Sophia Antipolis de France, pour sa relecture enrichissante et minutieuse du dernier chapitre de la thèse. Ses directives, en tant qu'experte du domaine, m'ont beaucoup rajoutée.

J'adresse mes remerciements particuliers aux membres du jury, qui ont accepté aimablement d'évaluer ce travail : Je remercie **M. Azeddine ZAHI**, professeur à la Faculté des Sciences et Techniques de Fès, d'avoir accepté de présider la soutenance de cette thèse. Je le remercie également pour les discussions échangées lors des réunions durant la première année de thèse. Son enthousiasme et sa sympathie m'a beaucoup appris. Je voudrais remercier les rapporteurs de cette thèse M. Mahmoud NASSAR, professeur à l' Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes de Rabat, **M. Mohamed OUMSIS**, professeur à l' Ecole Supérieur de Technologie de Salé, et **M. Said OUATIK El Alaoui**, professeur à la Faculté des Sciences Dhar El Mehraz de Fès, d'avoir accepté de rapporter ce travail et pour l'intérêt qu'ils en ont porté.

Mes remerciements s'adressent également aux membres de l'équipe de recherche WUMEDIA pour les discussions fructueuses et les conseils pendant les différentes réunions d'équipe. J'ai le plaisir de remercier mes amis et collègues du laboratoire Systèmes Intelligents et Applications de la FST de Fès, ainsi ceux du laboratoire Modélisation Mathématique et Informatique de l'ENSAM de Meknès pour leur sympathie et leurs encouragements. Je remercie Soufiane et Oumayma pour les discussions échangées. Un grand merci à Abla, Siham, Anass, Zakaria et Brahim pour leur amitié et leur hospitalité chaleureuse au sein de l'ENSAM de Meknès. Je souhaite aussi remercier ma chère amie Lamiae pour sa compagnie précieuse, son soutien et sa disponibilité à n'importe quelle heure du jour et de la nuit.

Mes remerciements et mes pensées vont aussi à ma famille, avec cette question récurrente, « quand est-ce que tu la soutiens cette thèse ? », bien qu'angoissante en période fréquente de doutes, m'ont permis de ne jamais dévier de mon objectif final. Merci à ma cousine Hajar, pour son aide et son affection mainte fois renouvelée. Un grand merci à ma tante Majida pour ses encouragements et son hospitalité durant mon séjour à Casablanca. Je remercie aussi mon oncle Abdelghafor et ma grand-mère pour les conditions qu'ils ont mis à ma disposition pour réussir ce travail lors de mon séjour à Meknès. Un merci particulier à Maryam, Sara, Salima, Hakima, ma tante Hafida et Abdelali.... Leur amour, leur confiance et leur soutien m'ont aidé à surmonter bien des obstacles. Ce travail est autant le mien que le leur.

Je pense naturellement à mes parents si précieux et à mon frère Taha. Ils m'ont toujours aidée et encouragée. Je les remercie pour leur réconfort dans les moments de doute et pour leurs prières incessantes. Les valeurs de travail et de ténacité qu'ils m'ont inculquées ne m'auront jamais été aussi utiles. Ils m'ont donnée la force in-

térieure d'aller au bout de ce projet de recherche et ces quelques mots ne suffiront jamais à exprimer tout ce que je leur dois.

RÉSUMÉ

On assiste depuis des décennies à une montée croissante de données générés grâce au développement rapide des systèmes d'informations d'entreprises, appuyée par une grande capacité de stockage. Cette prolifération de données à grande échelle a créé de réels défis pour les entreprises et pour la communauté scientifique. L'entreposage de données (Data Warehousing) fournit une excellente approche pour transformer les données en informations utiles et fiables pour soutenir le processus de prise de décision des entreprises et aussi pour atteindre l'intelligence économique. L'un des mécanismes les plus importants et les plus critiques dans la construction et l'exploitation du data warehouse est la gestion des métadonnées qui se heurte à plusieurs problèmes tels que l'intégration, l'interopérabilité et la réutilisation.

Dans le cadre de cette thèse, nous visons à gérer et à capitaliser les connaissances dans un environnement décisionnel multi-expert afin d'aboutir à une fin de prise de décisions stratégiques au sein des organisations. Notre approche s'inscrit principalement dans l'ingénierie de connaissance avec un formalisme Orienté-Objet. Nous proposons d'intégrer les savoir-faire et les préférences des utilisateurs dans le cycle décisionnel via la notion de point de vue par extension du standard Common Warehouse Metamodel. L'objectif est de pouvoir capitaliser le point de vue et la trace de chaque utilisateur (expert) du processus décisionnel sous forme de métadonnées. Ensuite, en vue de mettre l'accent sur l'interaction et l'interdépendance entre les différentes analyses et points de vue dans le cycle décisionnel, nous formalisons un ensemble de relations entre les points de vue des acteurs afin d'améliorer la coordination, le partage de connaissances et la compréhension mutuelle entre les différents acteurs d'une analyse multipoints de vues. En conséquence, nous développons le Common Case Reuse Framework (CCReF), en se basant sur le métamodèle point de vue et sur le Raisonnement à Partir de Cas, dans le but de développer la réutilisabilité des expériences réussies et l'assistance des utilisateurs en termes de point de vue dans un processus décisionnel.

Mots-clés : Métadonnées, Intégration, Réutilisation, Capitalisation de connaissances, Interopérabilité, Processus décisionnel, CWM.

ABSTRACT

For decades, there has been a steady rise in data generated by the rapid development of enterprise information systems, supported by a large storage capacity. This proliferation of large-scale data has created real challenges for the organizations and for the scientific community. Data Warehousing provides an excellent approach to turn data into useful and reliable information to support business decision-making, and also to achieve business intelligence. One of the most important and critical mechanisms in the construction and operation of the data warehouse is the management of metadata that faces several problems such as integration, interoperability, and reusability.

In this thesis, we aim to manage and capitalize knowledge in a multi-expert decisional environment to achieve an efficient decision-making within organizations. Our approach stems primarily from the field of knowledge engineering with an object-oriented formalism. We propose to integrate the users' know-how and preferences in the decisional process via the notion of viewpoint by extension of the Common Warehouse Metamodel standard. The objective is to capitalize the user's viewpoint and the user's trace in the decisional process in the form of metadata. Then, in the aim to highlight the interaction and interdependence between the different analyzes and viewpoints in the warehouse cycle, we formalize a set of relationships between the users' viewpoint to improve coordination, sharing of knowledge and mutual understanding between the different actors of a multi-viewpoint analysis. Consequently, we develop the Common Case Reuse Framework (CCReF), based on the viewpoint metamodel and the Case-Based Reasoning, in order to enhance the reusability of successful experiences and users' assistance in the decisional process in terms of viewpoint.

Keywords : Metadata, Integration, Reutisability, Knowledge capitalisation, Interoperability, Decisional process, CWM.

LISTE DE PUBLICATIONS

JOURNAUX

- Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., (2016). **A viewpoint based extension of the common warehouse metamodel to support the user's viewpoint approach.** International Journal of Metadata, Semantics and Ontologies, 11(3), 137-154.

  Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., **On the Effective Reuse of Metadata in a Cwm Decisional Process.** Journal of Mobile Multimedia (JMM). (Article in press)

- Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., **CCReF : a Framework for Reusing Metadata and Users Experiences in a Warehouse Process.** (En cours de soumission)

CONFÉRENCES INTERNATIONALES AVEC COMITÉ DE LECTURE

— Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., (2014, May). **Towards an approach to integration of the viewpoint concept into the CWM.** In the Fifth International Conference on Next Generation Networks and Services (NGNS), Casablanca- Morocco, (pp. 352-355). IEEE.

— Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., (2014, October). **Towards integration of the users' preferences into the common warehouse metamodel.** In the Third IEEE International Colloquium Information Science and Technology (CIST), Tetuan- Morocco, (pp. 151-154). IEEE.

— Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., (2016, November). **A case-based reasoning approach to the reusability of CWM metadata.** In International Conference on Systems of Collaboration (SysCo'16), Casablanca – Morocco, (pp. 1-6). IEEE.

— Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., (2016, December). **Modeling ViewPoint Relations for CWM Metadata Reusability.** In the Sixth International Conference on Next Generation Networks and Services (NGNS'16), Rabat- Morocco.

CONFÉRENCES NATIONALES AVEC COMITÉ DE LECTURE

— Demraoui, L., Behja, H., Zemmouri, E. M., & Ben Abbou, R., (November, 2013). **Intégration de la Multi-Analyse dans le Standard CWM**, La cinquième édition des Journées Doctorales en Technologies de l'Information et de la Communication (JDTIC'13), Kénitra –Maroc

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

0.1 CONTEXTE ET MOTIVATION

L'essor des technologies de l'information a accru le volume de données disponibles de manière considérable. Désormais, la conséquence observable est que les organisations font face à un volume croissant de données hétérogènes qui transitent au sein de leurs systèmes d'information. Cette surabondance de données incohérentes et redondantes a soulevé de réels défis pour la communauté scientifique, quant à la capacité de gestion, la difficulté d'exploitation, l'accès et l'utilisation efficace de ces mêmes données pour des fins de décision (Biernacki et Waldorf 1981). Cette dynamique remarquable des corpus d'informations en expansion semble nécessiter de la part des organisations une capacité d'adaptation et de réactivité toujours plus grande. Il s'agit donc pour les organisations de passer par leur capacité à interagir efficacement avec l'ensemble des acteurs afin de fiabiliser l'ensemble d'information et les transformer en connaissance. Cette dernière devrait être capitalisée en faisant du savoir et du savoir-faire un capital humain par accumulation d'informations pertinentes dans un domaine en vue de conserver l'expérience des acteurs et pouvoir la réutiliser et la partager par la suite.

En vue de répondre à ces besoins, le nouveau rôle de l'informatique était de définir et d'intégrer une architecture qui sert de fondations aux applications décisionnelles. C'est le cas de l'informatique décisionnelle, Business Intelligence (BI) en anglais (Watson et Wixom 2007), qui représente l'ensemble des moyens, méthodes, et outils permettant de collecter, consolider, modéliser, restituer les données matérielles ou immatérielles des organisations, en vue d'offrir une aide à la décision et de permettre aux responsables de la stratégie de l'entreprise et de son opérationnalisation d'avoir une vue d'ensemble de l'activité traitée.

Le Data Warehouse (DW), entrepôt de données, représente le système d'information dédié aux applications décisionnelles. Son objectif est de permettre un traitement synthétique de l'information pour faciliter les prises de décisions. Il fournit une excellente approche pour transformer les données en informations utiles et fiables pour soutenir le processus de prise de décision des entreprises et aussi pour atteindre l'intelligence économique.

L'un des mécanismes les plus importants et les plus critiques dans la construction et l'exploitation d'un Data Warehouse est la gestion des métadonnées. En effet, ces derniers servent à construire, entretenir, gérer, et exploiter un entrepôt de données. La Figure 0.1 illustre le flux et les échanges de métadonnées dans un processus décisionnel.

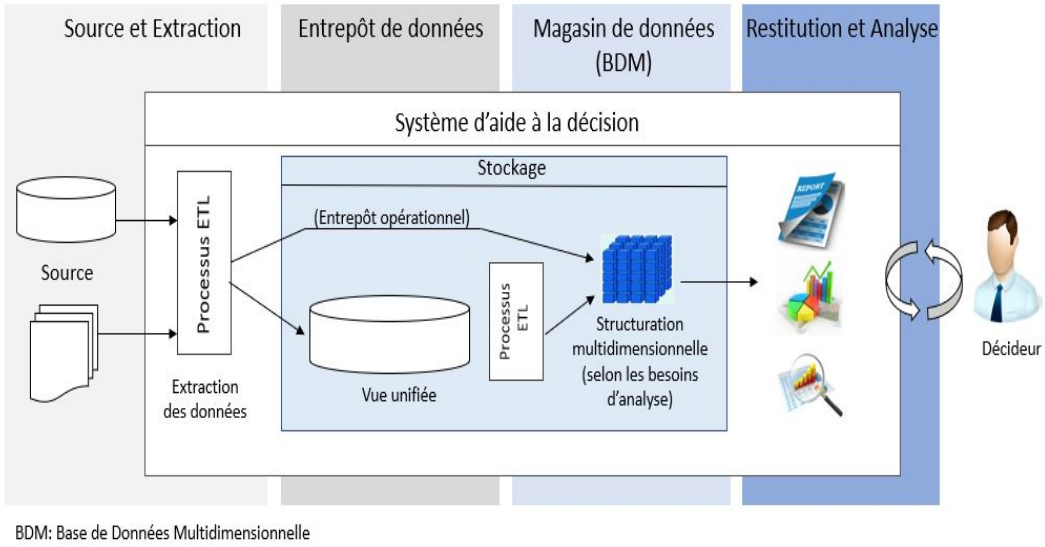

FIGURE 0.1 – Les différentes étapes de la chaine décisionnelle.

La gestion des métadonnées se heurte à trois problèmes majeurs : l'**intégration**, l'**interopérabilité** et la **réutilisation**. En effet, Afin d'interagir efficacement dans la suite décisionnelle, l'ensemble des acteurs et des outils logiciels doit être capable de partager efficacement et facilement l'ensemble des métadonnées. Ce partage nécessite une définition commune de la structure et la sémantique des métadonnées partagée afin d'assurer une meilleure intégration et interopérabilité. En effet, dans chaque phase du processus décisionnel, on trouve plusieurs acteurs et plusieurs systèmes et outils logiciels hétérogènes qui utilisent des formats de métadonnées différents. Ce qui nécessitent le plus souvent la construction de ponts complexes de transformation des métadonnées qui nécessitent d'avoir la connaissance détaillée des structures des métadonnées, ainsi que les interfaces de chaque produit à intégrer (POOLE et al. 2002) pour assurer les échanges entre différents outils logiciels.

C'est dans ce sens que Object Management Group (OMG)a mis en œuvre le Common Warehouse Model (CWM) pour pallier ces problèmes et assurer une interopérabilité et

intégration des métadonnées entre les systèmes hétérogènes. Le standard CWM fournit un cadre pour représenter les métadonnées sur les données sources et cibles, les transformations et les analyses, ainsi que les processus et les opérations qui créent et gèrent les métadonnées d'entreposage de données et fournissent des informations sur leur utilisation. Ainsi, pour assurer l'intégration et l'interopérabilité entre les différents outils logiciels de la chaîne décisionnelle, chaque outil doit consommer et générer des métadonnées conformes au métamodèle CWM.

Le CWM couvre le cycle de vie complet de modélisation, de construction et de gestion des entrepôts de données. Il définit un métamodèle qui représente les métadonnées aussi bien au niveau métier qu'au niveau technique. Cependant, nous avons constaté que l'aspect **capitalisation des connaissances** et **réutilisations** n'a pas été abordé dans le standard CWM.

0.2 DÉMARCHE ET OBJECTIFS

Dans cette thèse, nous proposons d'intégrer les connaissances, les savoir-faire et les préférences des utilisateurs dans le cycle décisionnel via la notion de point de vue. Ceci permettra d'assurer une meilleure assistance des utilisateurs et experts dans la suite décisionnelle et une bonne réutilisation des annotations et des expériences établies lors de l'exploitation du cycle décisionnel. Notre approche se positionne dans un cadre d'ingénierie de connaissance avec un formalisme Orienté-Objet.

Dans un premier temps, nous allons décrire notre approche de point de vue dans le processus décisionnel par extension du standard CWM et expliciter les objectifs à atteindre par rapport à cette définition. Le métamodèle CWM sera étendu par les nouveaux éléments de notre métamodèle point de vue afin d'assister l'utilisateur et de capitaliser ses annotations lors de l'exploitation du cycle décisionnel.

Ensuite, en vue de mettre l'accent sur l'interaction et l'interdépendance entre les différentes analyses et points de vue dans le cycle décisionnel, nous formalisons un ensemble de relations entre les points de vue des acteurs à savoir : l'équivalence, l'inclusion, et l'indépendance. Ces relations permettent d'améliorer la coordination, le partage de connaissances et la compréhension mutuelle entre les différents acteurs d'une analyse multipoints de vues, et développer la réutilisabilité en termes de point de vue des expériences réussies dans un processus décisionnel.

En conséquence, en se basant sur le métamodèle point de vue développé et sur le concept du Raisonnement à Partir de Cas, nous développons le Common Case Reuse Framework (CCReF) afin de pouvoir réutiliser les expériences et les annotations des utilisateurs lors d'une exploitation du processus décisionnel en vue de résoudre de nouvelles situations ou problèmes, et aussi de garder la trace des raisonnements et des principales décisions prises par les utilisateurs de CWM lors du processus décisionnel.

Pour résumer, les objectifs principaux de cette thèse sont :

— Intégration de la notion Point de Vue dans le processus décisionnel par extension du métamodèle CWM. L'extension permettra une meilleure assistance et capitalisation des annotations des utilisateurs durant l'exploitation du processus décisionnel.
— Coordination et partage des métadonnées et des annotations des utilisateurs : Garder la trace de raisonnement effectué pendant un travail collaboratif afin d'assurer une coordination, compréhensibilité et partage de connaissances entre les différents acteurs d'une analyse multi-points de vue.
— La réutilisation de connaissances : améliorer et encourager la réutilisation d'expériences réussies des utilisateurs en termes de points de vue, et en termes d'interactions et interdépendances entre les acteurs.
— Support des utilisateurs novices du système décisionnel : assister les utilisateurs non-experts durant leurs exploitations du processus décisionnel.

0.3 ORGANISATION DU MÉMOIRE

Après cette section introductive, ce mémoire est composé de six chapitres organisés en deux parties. Une partie état de l'art composée des trois premiers chapitres et une partie Contributions qui illustre notre approche d'intégration du point de vue et de la multi-analyse dans le cycle décisionnel. Le plan est détaillé dans les points suivants :

Le chapitre 1 présente les concepts généraux de la méta-modélisation et de l'Ingénierie Dirigée par des Modèles, notamment les notions de modèle, métamodèle et de méta-métamodèle. Aussi, le chapitre expose la notion de métadonnée et son rôle stratégique dans la prise de décision et l'interopérabilité entre les systèmes. Par la suite, nous présentons les principaux problèmes de l'intégration et l'interopérabilité des métadonnées dans le cycle décisionnel, et nous introduisons le standard CWM comme solution pour ces problèmes.

Le chapitre 2 a pour objectif de présenter le standard CWM et ses différents métamodèles et il recense les différents mécanismes de l'extension du CWM pour supporter de nouveaux concepts de différents domaines. Par la suite, nous détaillons les différents travaux de la littérature qui ont étendu le CWM pour répondre aux différents objectifs.

Le chapitre 3 présente un état de l'art sur le concept Point de Vue et son utilisation de dans la modélisation des systèmes d'information et la représentation des connaissances. Dans ce chapitre nous passons en revue les différentes définitions et approches de la notion de point de vue. Et nous proposons une analyse et comparaisons d'un ensemble de systèmes qui ont intégré la notion de point de vue, surtout pour une représentation des connaissances multipoints de vue.

Le chapitre 4 représente le premier chapitre de la partie contribution, dans lequel nous développons notre approche de point de vue dans le cycle décisionnel en se basant sur le CWM et nous explicitons les objectifs à atteindre par rapport à notre définition de point de vue. Par la suite, nous définissons un ensemble de relations pour modéliser les relations entre les points de vue des différents utilisateurs exprimés lors d'un processus décisionnel.

Le chapitre 5 développe notre Framework pour réutiliser les expériences et les annotations des utilisateurs en termes de point de vue en se basant sur le métamodèle point de vue défini dans le chapitre 4 et sur le processus de Raisonnement à Partir de Cas.

Le chapitre 6 se consacre à l'implémentation d'une étude de cas pour illustrer et évaluer notre approche.

Enfin le dernier chapitre (chapitre 7) conclut la thèse en donnant un bilan du travail effectué et les perspectives envisageables au terme de cette recherche.

Première partie

ETAT DE L'ART

1

MÉTAMODÉLISATION ET INTEROPÉRABILITÉ DES SYSTÉMES

1.1 INTRODUCTION

L'évolution permanente des besoins métier de l'entreprise et des technologies rend les systèmes informatiques de plus en plus complexes et rapidement obsolètes. Cela génère des couts de maintenance et de migration insupportables pour la plupart des entreprises.

L'ingénierie logicielle s'oriente aujourd'hui vers une approche de développement d'applications des systèmes dirigée par les modèles.Ingénierie Dirigée par les Modèles (IDM) peut être considérée à la fois comme continuité et séparation avec les travaux des approches objet. La continuité se manifeste dans le fait que les technologies objet ont été la base de l'évolution vers les modèles, tandis que le point de séparation de l'IDM par rapport aux précédents travaux était de classifier les objets en fonction de leurs différentes origines (objet fonctionnel, technologique, etc.) (Bézivin et Briot 2004). L'IDM vise donc, d'une manière plus poussée que les patterns ou les aspects de l'approche objet, à fournir un cadre qui permet d'exprimer séparément chacune des préoccupations des utilisateurs, des concepteurs, des architectes, etc. Le principe de séparation des préoccupations, est considéré comme le point de rupture par rapport aux travaux précédents. L'IDM offre un cadre méthodologique et technologique permettant d'unifier et de favoriser dans un processus homogène, l'étude des différents aspects du système. L'ingénierie dirigée par des modèles préconise l'utilisation des modèles comme entité centrale. Cette approche a contribué à une montée en puissance des modèles qui ont passé de leur stade contemplatif (visuel et documentaire) à un stade plus productif qui envisage l'utilisation des modèles au cœur du cycle de développement de ces systèmes (Blanc et Salvatori 2011).

L'enjeu de l'IDM est de capitaliser les savoir-faire de conception et de validation d'un système en capturant et en réutilisant les connexions entre ces points de vue, que ce soit de manière horizontale, par tissage de liens entre modèles, ou verticale, par transformation de modèles. Visant à automatiser une partie du processus de développement, l'IDM requiert un effort d'abstraction plus important de la part des développeurs. En contrepartie, l'IDM promet de conserver le savoir-faire de conception

proche des centres de décision grâce aux économies d'échelle dues à l'automatisation (Jézéquel, Barais et Fleurey 2009).

1.2 L'INGÉNIERIE DIRIGÉE PAR LES MODELES (IDM)

L'OMG, consortium de plus 1000 entreprises, initie la démarche IDM. Dans la littérature, les termes « démarche », « norme », et « approche » qualifient l'IDM MDA 2008. L'IDM connue en anglais sous le terme Model Driven Engineering (MDE), est une discipline du génie logiciel qui promeut les modèles en entités de première classe dans le développement logiciel (Bézivin et Briot 2004). Kent fut le premier qui a proposé le terme IDM (Kent 2002) ; par la suite, cette nouvelle discipline a fait l'objet d'un grand intérêt aussi bien de la part des équipes de recherche académiques (Diaw, Lbath et Coulette 2010).

Cette approche a pour but d'apporter une nouvelle vision unifiée permettant de concevoir des applications en séparant la logique métier de l'entreprise, de toute plateforme technique. En effet, la logique métier est stable et subit peu de modifications au cours du temps, contrairement à l'architecture technique. Il est donc évident de séparer les deux pour faire face à la complexité des systèmes d'information et aux coûts excessifs de migration technologique. Cette séparation autorise alors la capitalisation du savoir logiciel et du savoir-faire de l'entreprise. A ce niveau, l'approche objet et l'approche composant n'ont pas su tenir leurs promesses. Il devenait de plus en plus difficile de développer des logiciels basés sur ces technologies. Le recours à des procédés supplémentaires, comme le patron de conception ou la programmation orientée aspect, était alors nécessaire (Menet 2010).

L'IDM offre la possibilité de stopper l'empilement des technologies qui nécessite de conserver des compétences particulières pour faire cohabiter des systèmes divers et variés. Ceci est permis grâce au passage d'une approche interprétative où l'utilisateur a un rôle actif dans la construction des systèmes informatiques à une approche transformationnelle, où le rôle de l'individu est simplifié et amoindri grâce à la construction automatisée.

L'IDM représente aussi une forme d'ingénierie générative par laquelle tout ou partie d'une application informatique est générée à partir de modèles. Dans cette nouvelle perspective, les modèles occupent une place de premier plan parmi les artéfacts de développement des systèmes ce qui permet d'augmenter la pérennité du savoir métier de l'application (Blanc et Salvatori 2011). En outre, ces modèles doivent en contrepartie être suffisamment précis et riches afin de pouvoir être interprétés ou

transformés par des machines. Le processus de développement des systèmes peut alors être vu comme une séquence de transformations de modèles partiellement ordonnée, dont chaque transformation prend un ou plusieurs modèles en entrée et produit un ou plusieurs modèles en sortie, jusqu'à l'obtention d'artéfacts exécutables. Cette transformation des modèles nécessite la disposition d'outils flexibles pour la gestion des modèles et de langages dédiés pour leurs transformations et leur manipulation tout au long de leur cycle de vie.

Au sens large, les principaux objectifs de l'IDM sont la portabilité, l'interopérabilité et la réutilisation. Dans ce contexte l'OMG a défini une approche spécifique de l'IDM appelée Model Driven Architecture (MDA) (MILLER, MUKERJI, BELAUNDE et al. 2003) (SOLEY et al. 2000)qui vise la définition d'un cadre normatif pour l'IDM. La MDA est basée sur les problématiques suivantes :

— Spécification d'un système indépendamment de la plateforme sur laquelle ce système doit être déployé
— Spécification de plateformes
— Définition d'une plateforme pour un système
— Transformation des spécifications d'un système en un système associé à une plateforme particulière.

1.2.1 *Architecture de l'IDM*

La Figure 1.1 illustre les quatre couches de spécification de l'IDM :

Le noyau de l'architecture comporte les standards de base : Uniform Modelling Language (UML) **(**avgeriou2004software), Meta Object Facility (MOF) **(**overbeek2006meta) et CWM **(**poole2002common)spécifiés par l'OMG pour modéliser la logique métier de l'application. Ce modèle métier est spécialisé ensuite dans une technologie middleware. On retrouve les standards actuels tels que les EJB (SIERRA et BATES 2005), CORBA (SCHMIDT et KUHNS 2000), .NET (RICHTER 2002) et les Web Services (CASTAGNA, GESBERT et PADOVANI 2009). L'anneau extérieur du cercle représente les services système qui permettent de gérer la transaction, la persistance, et les évènements. Enfin, la couche spécifique au domaine se base sur des profils UML afin de proposer des Frameworks spécifiques au domaine d'application (Espace, Télécommunication, Santé, etc.).

Conceptuellement, l'intérêt fondamental de l'architecture IDM provient des différents niveaux d'abstraction : chaque niveau **n** de cette architecture permet de spécifier un système du niveau **n-1** en se focalisant sur ce qui est essentiel au système et

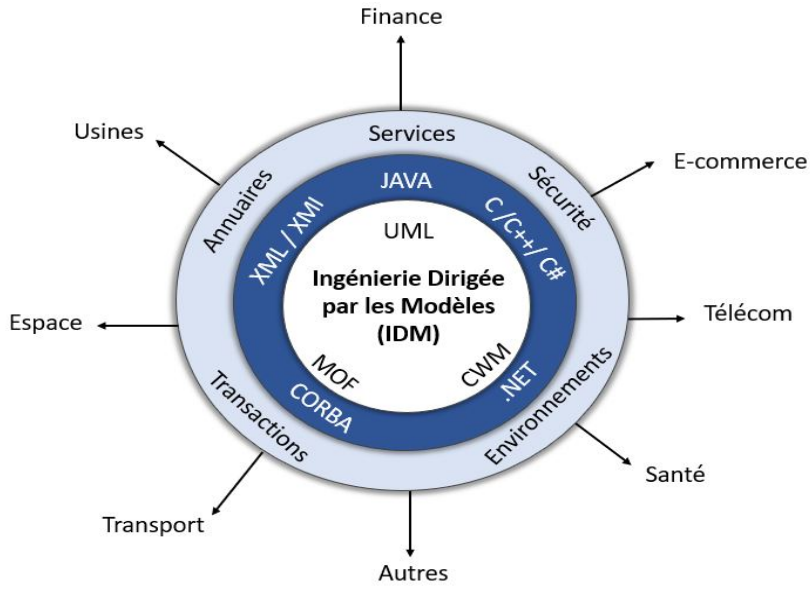

FIGURE 1.1 – Architecture de l'ingénierie dirigée par les modèles (IDM).

en s'abstrayant des spécificités de mise en œuvre comme le langage ou l'architecture cible.

1.2.2 *Modéles, métamodèles et méta-métamodèles*

La modélisation du monde réel était toujours l'une des préoccupations majeures de l'informatique. Cette modélisation consiste à créer un modèle informatique à partir du monde réel qu'on peut appeler aussi le domaine d'application. Cette section définit les concepts de base de l'ingénierie des modèles, à savoir la notion de modèle, de méta-modèle, de méta- méta-modèle, et les relations qui existent entre ces concepts.

1.2.2.1 *Modèles et Systèmes*

Indépendamment de la discipline scientifique considérée, un modèle représente une abstraction d'un système construite dans un but précis. Dans le contexte de l'IDM un modèle est défini comme une description d'un / (d'une partie) d'un système dans un langage précis qui a une syntaxe et une sémantique bien définie et qui est interprétable par un outil (KLEPPE, WARMER et BAST 2003).

On dit alors que le modèle représente le système. A titre d'exemple, une carte routière est un modèle d'une zone géographique, conçu pour circuler en voiture dans cette zone. Par conséquent, la première relation majeure de l'IDM, entre le modèle et le système qu'il représente est appelée "représentationDe" (RepresentedBy) (Atkinson et Kuhne 2003) (Bézivin et Briot 2004) (Seidewitz 2003).

Les modèles offrent de nombreux avantages dont le plus important est de spécifier différents niveaux d'abstraction, facilitant la gestion de la complexité inhérente aux applications. Les modèles possédant un niveau élevé d'abstraction sont utilisés pour présenter l'architecture générale d'une application ou sa place dans une organisation, tandis que les modèles très concrets permettent de spécifier précisément des protocoles de communication réseau ou des algorithmes de synchronisation (Menet 2010).

1.2.2.2 *Métamodèle : langage de modélisation*

Afin de rendre les modèles utilisables et leur donner une dimension opérationnelle, il est essentiel de spécifier non seulement leur sémantique, mais aussi de décrire les langages utilisés pour les représenter de manière précise. Dans une approche IDM, les spécificités techniques sous-jacentes à un modèle sont écartées. L'IDM préconise l'utilisation d'un mécanisme standard et abstrait pour définir des modèles. Ce mécanisme abstrait est dénoté par le terme métamodèle. Ainsi dans la pratique, tout modèle défini par l'intermédiaire d'une approche IDM doit posséder un métamodèle. La notion de métamodèle se définie aussi comme étant le modèle d'un modèle ; en d'autres termes, un métamodèle est un modèle qui définit le langage d'expression d'un modèle.

Un métamodèle permet de définir une trame générale des processus d'ingénierie des systèmes d'information, pouvant être réutilisable dans une organisation en définissant des modèles propres à chaque projet à l'intérieur de cette même organisation. Les métamodèles ont ainsi un rôle structurant très important. Leur principal avantage c'est de pouvoir projeter facilement des modèles vers des plateformes technologiques et passer d'un modèle à un autre.

La notion de métamodèle conduit à l'identification d'une seconde relation, liant le modèle et le langage utilisé pour le construire, appelée "conforme à" (ConformeTo). On dit qu'un modèle est conforme à un métamodèle si l'ensemble des éléments du modèle est défini par le métamodèle. Cette notion de conformité est essentielle à l'ingénierie des modèles mais n'est pas nouvelle ; à titre d'exemple : un texte est

conforme à une orthographe et une grammaire, un programme JAVA est conforme au langage JAVA et un document Extensible Markup Language (XML)est conforme à sa Document Type Definition (DTD), etc...

Pour supporter cette approche, l'OMG a défini une pyramide de méta modélisation à quatre niveaux. Ces derniers représentent les différents niveaux d'abstraction nécessaires à la modélisation des systèmes. Aussi, chaque niveau d'abstraction entretient une relation d'instanciation avec le niveau supérieur (Djebbi et Gervais 2004). L'architecture proposée est présentée par la Figure 1.2. En commençant du bas :

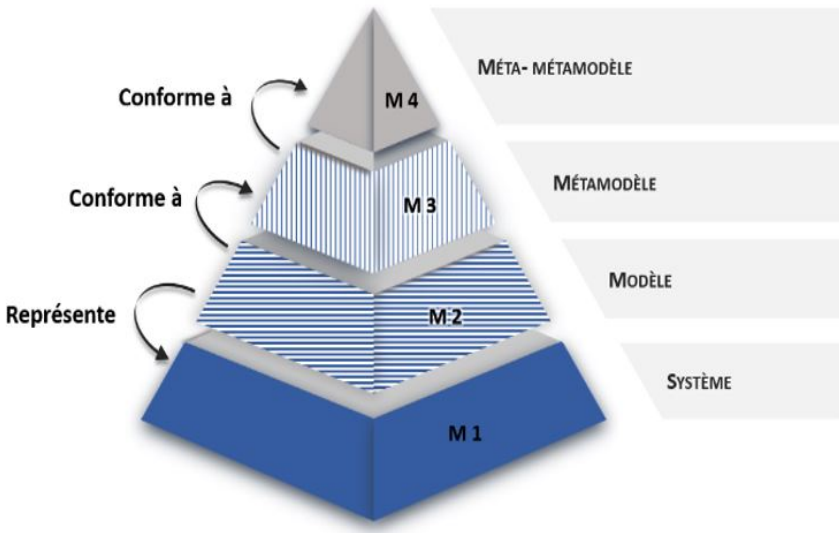

Figure 1.2 – Les quatre niveaux de la pyramide de Méta-modélisation.

— Le **niveau M0** (ou instance), représente le monde réel : à titre d'exemple, une application qui s'exécute contient les informations du monde réel que l'on souhaite modéliser. Ce sont les informations réelles de l'utilisateur, instance du modèle de M1.

— Le **niveau M1** (ou modèle), est composé des modèles d'information. Les informations de M0 sont décrites par un modèle appartenant au niveau M1, et qui représente le système du niveau M0. Les modèles du niveau M1 de même famille sont exprimés dans un langage unique dont la définition est fournie explicitement au niveau M2. C'est le cas par exemple des modèles UML qui

appartiennent au niveau M1.

— Le **niveau M2** (ou métamodèle), il définit le langage de modélisation et la grammaire de représentation des modèles de niveau M1. C'est modèles sont conformes à leurs métamodèles du niveau M2. Le méta-modèle UML décrit dans le standard UML, et qui définit la structure interne des modèles UML, fait partie de ce niveau. Aussi, les profils UML qui permettent d'étendre le méta-modèle UML appartient au niveau M2 (DJEBBI et GERVAIS 2004). Un méta-métamodèle quant à lui, permet de décrire un modèle de métamodèles ; autrement dit c'est un langage de description de métamodèles (par exemple le méta métamodèle MOF (OMG 2008). A titre d'exemple, UML est un métamodèle qui offre des concepts permettant de décrire les différents modèles (Diagramme de classe, Diagramme de cas d'utilisation) d'un système.

— Le **niveau M3** (ou méta-méta-modèle) est composé du langage unique de définition des métamodèles, appelé MOF. C'est un méta-métamodèle qui définit la structure de tous les métamodèles qui se trouvent au niveau M2. Le MOF est réflexif, c'est-à-dire qu'il est conforme à lui-même et qu'il s'auto-décrit, ce qui permet de dire que le niveau M3 est le dernier niveau de la hiérarchie. Le niveau M3 correspond donc aux fonctionnalités universelles de modélisation logicielle, alors que le niveau M2 correspond aux aspects spécifiques des différentes familles, chaque aspect étant pris en compte par un méta-modèle spécifique.

En outre, la technique de métamodélisation bénéficie des avantages de la modélisation, notamment la séparation des préoccupations pour définir un système qui est une propriété de plus en plus utilisée dans le développement de systèmes informatiques afin d'en maîtriser la complexité. Actuellement, plusieurs métamodèles sont disponibles, certains, très généraux qui ne sont pas liés à un domaine particulier, et c'est le cas d'un des plus célèbres d'entre eux, UML (J. RUMBAUGH et D. RUMBAUGH 2005), d'autres, au contraire, sont plus spécifiques, on parle alors de Domain Specific Languages (DSL) (LÉDECZI et al. 2001), comme :

- **SPEM**

Software Process Engineering Metamodel (SPEM), que l'on peut traduire par le métamodèle d'ingénierie des procédés logiciels, est un métamodèle visant à décrire le processus de développement logiciel. SPEM fait partie des langages de modélisation proposés par l'OMG. Il fournit une définition des termes usuels et une description de la notion de composant de processus. C'est autant un métamodèle de conception de processus qu'un Framework conceptuel qui met à disposition les outils et les

concepts pour modéliser, documenter, présenter, gérer et rendre concret le développement. SPEM a connu deux versions : la version 1.1 (Kruchten et Cook 2002) de janvier 2005 et la version 2.0 (S. OMG et Notation 2008)adoptée le 3 mars 2007.

Ce métamodèle a la particularité d'être défini à la fois comme un métamodèle à part entière et comme un profil UML. De manière simplifiée, SPEM décrit un processus comme étant composé d'activités qui représentent les unités de travail Figure 1.3. Elles précisent les opérations à appliquer sur les produits de travail ; de produits de travail qui englobent les résultats ou les pré–requis des activités ; et de rôles qui définissent les responsabilités par rapport aux produits de travail et à la réalisation des activités.

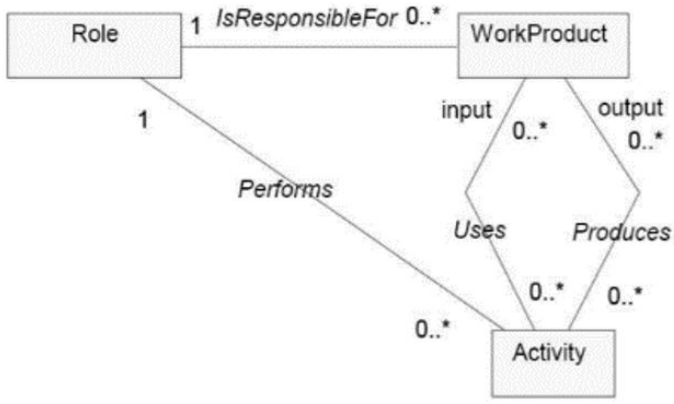

Figure 1.3 – Modèle conceptuel de SPEM : activités, rôles et produits de travail.

Les activités sont réalisées par un rôle unique et peuvent produire ou utiliser un nombre quelconque de produits de travail. Un même rôle peut réaliser plusieurs activités et aussi d'être responsable de plusieurs produits de travail. Une structure plus détaillée des éléments du métamodèle SPEM est fourni par la spécification de l'OMG (Combemale et al. 2006).

- **SYSML**

System Modeling Language (SysML) s'est créé d'une étroite collaboration entre l'INCOSE (International Council on Systems Engineering) et l'OMG. Après plusieurs versions, orientées autour de la modélisation des concepts des systèmes d'information (Sanford 2003), la version finale de SysML a été adoptée en mai 2006. Il s'agit d'un langage de modélisation graphique dont la sémantique semi-formelle est définie dans OMG. SysML est définit comme un métamodèle pour l'ingénierie système capable d'of-

frir un support pour la modélisation de multiples processus et méthodes. Il supporte la spécification, l'analyse, la conception et la validation et vérification d'une large gamme de systèmes. Il s'agit en fait d'un profil UML qui hérite donc des caractéristiques d'UML mais plutôt orienté vers les systèmes contrairement à UML est centré sur la conception de logiciels, Ce métamodèle fournit : une sémantique flexible et plus expressive, un langage « réduit » facile à apprendre et à appliquer, des tables d'allocations propres aux concepts des systèmes d'information (allocation d'exigences, de fonctions, etc.), et une représentation orientée modèles, vues et points de vue qui permet une décomposition selon les intérêts.

Etant donné que l'on peut définir un métamodèle spécifique pour chaque domaine, le risque est grand de les voir se multiplier de façon anarchique. C'est pour résoudre ce problème qu'est apparu le concept de méta métamodèle qui facilite l'interopérabilité entre différents métamodèles.

1.2.2.3 *Méta-métamodèle : langage de Méta modélisation*

De la même manière qu'il est nécessaire d'avoir un métamodèle pour interpréter un modèle, pour pouvoir interpréter un métamodèle il faut disposer d'une description du langage dans lequel il est écrit : un métamodèle pour les métamodèles. C'est naturellement que l'on désigne ce métamodèle particulier par le terme de méta métamodèle. En pratique, un méta métamodèle détermine le paradigme utilisé dans les modèles qui sont construit à partir de lui. De ce fait, le choix d'un méta métamodèle est un choix important et l'utilisation de différents méta métamodèles conduit à des approches très différentes du point de vue théorique et du point de vue technique.

Afin de se soustraire au problème de la définition des méta métamodèles et ainsi éviter d'avoir à définir un méta-méta-métamodèle, l'idée généralement retenue est de concevoir les méta métamodèles de sorte qu'ils soient autodescriptifs, c'est-à-dire qui se définissent eux-mêmes, à titre d'exemple le MOF. Dans le cas d'XML, c'est une DTD XML pour les DTD qui joue le rôle de méta métamodèle.

Comme nous avant cité précédemment, chaque niveau de l'architecture de l'IDM est décrit par un ensemble de données. Les données d'un niveau n-1 constituent une instanciation du modèle du niveau n qui en spécifie la sémantique. Aussi, les données et le modèle d'un même niveau n sont sémantiquement équivalents. Chaque description de données est caractérisée formellement dans un métamodèle (modèle de donnée) ; on parle ici des métadonnées (metadata).

1.2.3 *Outils de méta-modélisation*

Plusieurs outils de méta-modélisation font l'objet de nombreuses expérimentations dans la communauté IDM. Nous exposons dans cette section les outils que nous avons pu tester et qui serviront pour atteindre l'objectif de cette thèse qui est pour la modélisation et la manipuler du standard CWM.

1.2.3.1 *MetaEdit+*

MetaEdit+ représente le plus ancien outil de méta-modélisation, c'est un outil méta-CASE développé à l'Université de Jyväskylä, au début des années 1990, dans le cadre du projet 'MetaPHOR' (Kelly, Lyytinen et Rossi 1996). Il est depuis commercialisé par l'entreprise MetaCASE. MetaEdit+ (Tolvanen et Rossi 2003, (Muller, Fleurey et Jézéquel 2005) permet de définir explicitement un métamodèle et au même niveau de programmer tous les outils nécessaires, allant des éditeurs graphiques aux générateurs de code, en passant par des transformations de modèles. MetaEdit+ a été construit autour du langage de méta-modélisation GOPRR [1] . Grâce à ce langage de méta-modélisation, il était possible de spécifier un métamodèle statique et de dériver un éditeur graphique pour le modèle (MetaCase 2017). Cet outil est destiné à la création et l'outillage de nouveaux langages spécifique aux domaines (Kelly et Tolvanen 2008). C'est un environnement multi-utilisateur qui peut fonctionner sur de nombreux postes de travail (càd. Des clients connectés à un serveur par un réseau) grâce à un « Object Repository ». Il offre comme le montre la Figure 1.4 ci-dessous un ensemble d'outils intégrés : éditeurs graphiques de diagrammes, de tableaux et de matrices, navigateurs de modèles (Browsers), outils de développement de méthodes, générateur de code et de rapports, et API et import / export XML.

La spécification d'une fonctionnalité de génération de code s'effectue avec un langage de script (appelé Merl) qui permet de parcourir et manipuler les instances du modèle et de générer des instructions dans n'importe quel langage cible (HTML, XML, C++, Java, etc.) (Diaw, Lbath et Coulette 2010).

1.2.3.2 *TOPCASED*

TOPCASED est le fruit d'un projet collaboratif industrie-académie qui cible le développement de systèmes embarqués ayant des contraintes fortes de sûreté de fonctionnement ; son but est de fournir un atelier basé sur l'IDM pour le développement des systèmes logiciels et matériels embarqués (Farail et Vernadat 2012). L'outillage TOPCASED a pour principal objectif de simplifier la définition de nouveaux DSL ou

1. Metamodeling http ://www.metacase.com/support/45/manuals

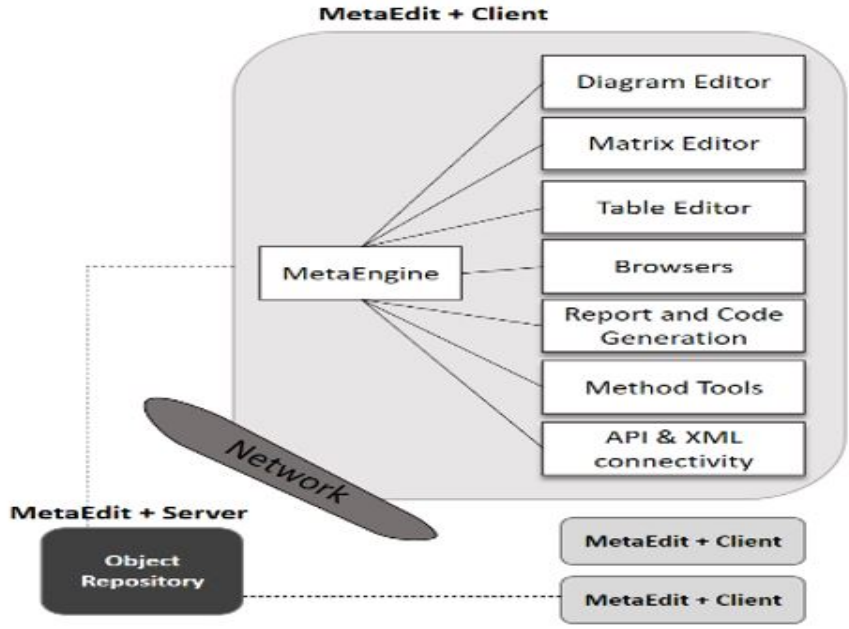

FIGURE 1.4 – Architecture générale du MetaEdit+.

de langages de modélisation en fournissant des technologies de niveau méta telles que des générateurs d'éditeurs syntaxiques (textuels et graphiques), des outils de validation statique et d'exécution de modèles, ce qui lui confère les atouts d'un véritable outil de méta-programmation. C'est une solution orientée IDM qui s'intègre dans la plateforme de développement Eclipse. Elle utilise les langages UML et SysML et met particulièrement l'accent sur la vérification, la validation et la traçabilité de modèles ainsi que la génération de code. Comme dans Kermeta, la syntaxe abstraite s'exprime avec Ecore complétée avec une syntaxe concrète définie avec un éditeur graphique. La sémantique du métamodèle permet la simulation d'exécution à l'aide d'une sorte de moteur générique composé de trois éléments :

— L'**Agenda** : gère les instances d'événements décrit dans la sémantique du méta-modèle
— Le **Contrôleur** (ou Driver) : supervise l'exécution et interagit avec l'extérieur à l'aide d'interfaces spécifiques (des API : Application Program Interface)
— L'**Interpréteur** : déclenche l'exécution des méthodes, met à jour les états des méta-objets et traite les nouvelles occurrences d'événements à traiter.

1.2.3.3 *Kermeta*

Kermeta est considéré comme un méta-atelier pour l'Ingénierie des Langages Dirigée par les Modèles (Jézéquel, Barais et Fleurey 2009). C'est un langage de métamodélisation associé à un environnement de développement de métamodèles exécutables. Il est basé sur EMOF (Essential MOF), une variante du méta-méta-modèle MOF intégrée avec l'environnement Eclipse (Kermeta 2017). En plus de la spécification de la structure des méta-modèles, Kermeta permet de décrire leurs comportements. Kermeta a été lancée en 2005 (Muller, Fleurey et Jézéquel 2005) dans l'équipe Triskell de l'IRISA [2] . Il est utilisé par INRIA [3] et ses partenaires académiques ou industriel dans le monde entier (CNRS [4], INSA [5] de l'université de Rennes. . .) pour répondre à une variété de besoins dont :

— L'édition de méta-modèles EMOF décrits sous forme textuelle dans l'environnement Eclipse ;
— L'ingénierie de langages et de DSL incluant une spécification statique et dynamique,
— L'ajout de contraintes de « bonne formation » (en OCL (Object Constraint Language)) aux méta-modèles EMOF, ces contraintes seront vérifiées par l'interpréteur Kermeta lors du chargement du modèle ;
— La transformation et la simulation de modèles et de méta-modèles.

Kermeta permet ainsi de définir et d'outiller de nouveaux langages en améliorant la manière de spécifier, simuler et tester la sémantique opérationnelle des méta-modèles. Il définit le comportement de chaque concept de manière impérative. Kermeta apporte aussi un complément aux méta-diagrammes UML sous la forme d'une méta-spécification directement opérationnelle. Le méta-modèle de Kermeta est composé de deux packages *Core* et *Behavior* correspondant respectivement à Ecore (la partie structurelle) et à une hiérarchie de méta classes représentant des expressions impératives (partie comportementale) Figure 1.5.

On peut associer chaque expression à une opération et une propriété. Les expressions Kermeta permettent de définir un comportement. En ce qui concerne la sémantique statique, Kermeta utilise d'abord le langage Object Constraint Language (OCL) pour exprimer des contraintes sur le méta-modèle. Ensuite, pour la sémantique d'exécution, le langage utilise des méthodes directement rattachées aux méta-classes. Ce langage d'action capture de manière opérationnelle les règles de déclenchement des opérations du méta-modèle à exécuter. Dans cette approche par métaprogrammation,

2. Institute for research in computer science and random systems
3. Institut national de recherche en informatique et en automatique
4. Centre national de la recherche scientifique
5. Institut National des Sciences Appliquées

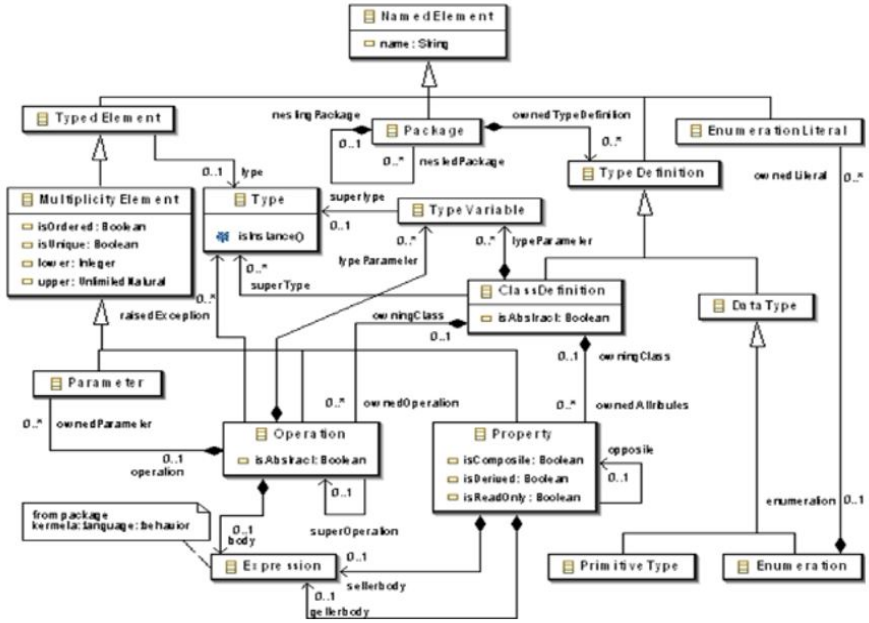

FIGURE 1.5 – La partie Core du métamodele Kermeta.

la sémantique d'exécution est ainsi décrite dans le corps des opérations sous forme d'un programme informatique. Elle prend la forme de séquences d'instructions dans le corps des méthodes rattachées aux méta-classes à l'aide d'un langage de programmation spécifique inspiré de Java et qui intègre le paradigme orienté aspects.

1.2.3.4 *Ecore*

Ecore est un méta-méta-modèle très proche du MOF. Il est en fait un sous ensemble du MOF.La Figure 1.6 illustre les méta-classes d'Ecore.

Les métamodèles conformes à ce méta-métamodèle sont toujours préfixés par un "E". Ils se composent d'EClass contenant des EAttribute et des EReference.

— EClass : représente une classe modelée, et contient : un nom, zéro ou plusieurs attributs, zéro ou plusieurs références.
— EAttribute : représente un attribut modelé. Un attribut a un nom et un type.
— EReference : représente une extrémité d'une association entre classe. Elle a un nom, un flag booléen pour indiquer si elle a du contenu ou non et une référence type.

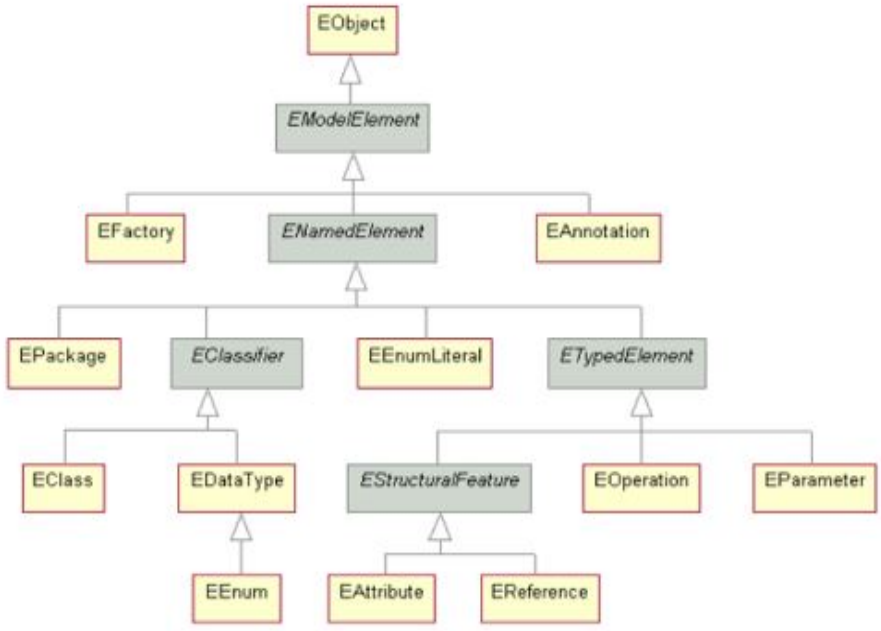

FIGURE 1.6 – Les métaclasses du métamodèle Ecore.

— EDataType : représente le type d'un attribut qui peut être primitif ou de type objet. Les EDataType dans Ecore sont (EBoolean, Echar, EFloat, EString, EByteArray, EBooleanObject, EFloatObject, EJavaObject).
— EPackage : Cette classe apporte des facilités pour accéder aux métadonnées Ecore du modèle. Elle contient des accesseurs aux EClasses, EAttributes et EReferences implémentées dans le modèle.
— EFactory : Comprend une méthode « Create » pour chacune des classes du modèle d'entrée. Cela va permettre de créer des instances (des objets) des classes de l'application.

Avec ces concepts, exprimés à un bon niveau d'abstraction, Ecore permet de définir uniquement la structure des modèles et métamodèles. Malgré le manque d'expressivité de ce langage de méta-modélisation plusieurs outils ont été développés en se basant sur ECore, on cite : TopCased (FARAIL et VERNADAT 2012), Eclipse Modeling Framework (EMF) (MERKS et al. 2003), GMF (Graphical Modeling Framework) (CONSORTIUM et al. 2007), ou encore Tiger (EHRIG et al. 2005), etc. La plupart de ces outils se limitent à des fonctionnalités d'édition textuelle ou graphique et offrent parfois la possibilité de faire de la vérification.

1.2.3.5 *EMF*

EMF, signifie Eclipse Modeling Framework, est une plate-forme de modélisation et de génération de code qui facilite la construction d'outils et d'autres applications basées sur des modèles structurés. Il permet le développement rapide et l'intégration de nouveaux plug-ins Eclipse. Le métamodèle d'EMF se base dans sa définition sur Ecore présenté en dessus Figure 1.7.

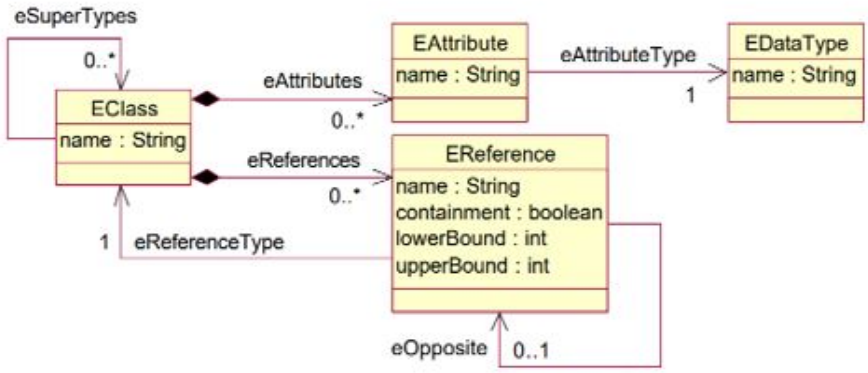

FIGURE 1.7 – les métaclasses du métamodèle EMF.

EMF est composé d'un ensemble de briques appelées plug-ins (Figure 1.8) :

— Le *métamodèle Ecore* : représente un canevas de classes pour décrire les modèles EMF et manipuler les référentiels de modèles.
— Le *EMF.Edit* : représente un canevas de classes pour le développement d'éditeurs de modèles EMF,
— Le modèle de génération *GenModel* qui permet de personnaliser la génération Java.
— Le *JavaEmitterTemplate* qui est un moteur de template générique.
— Le *JavaMerge* qui est un outil de fusion de code Java.

Le rôle principal d'EMF est d'accepter en entrée des modèles ou des fichiers et de générer en sortie du code correspondant à des outils / plug-in manipulant les données fournies en entrée. EMF fournit plusieurs fonctionnalités, dont la plus importante est la génération automatique d'un simple éditeur graphique qui permet l'édition des modèles sous forme d'arborescence en d'autres termes il permet de générer automatiquement à partir d'un méta-modèle un éditeur graphique offrant une

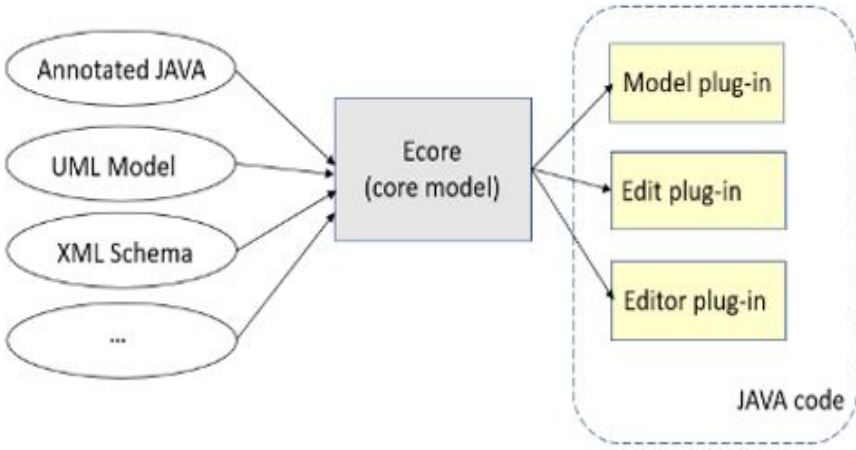

FIGURE 1.8 – Architecture d'Eclipse Modeling Framework (EMF).

vue arborescente d'un modèle. Chaque nœud de l'éditeur représentera une instance d'une méta-classe. Ensuite, la génération des interfaces de manipulation des modèles consiste à fournir des interfaces graphiques génériques pour manipuler des modèles. La génération de code améliore la productivité de développement d'application par l'automatisation de la génération de code à partir du modèle. Cette génération de code est paramétrée à l'aide d'un métamodèle appelé GenModel. Le modèle de génération est construit automatiquement à partir du modèle Ecore en associant un élément de paramétrage correspondant à chaque élément du modèle. Les informations détenues par ces éléments de paramétrage concernent différents aspects qui ont un impact sur la génération : règles de nommage, localisation des fichiers générés, mode de visualisation et d'édition, etc.

Le principal avantage de cette approche de génération paramétrée par un modèle, est le fait qu'elle ne surcharge les éléments de modélisation par d'autre détails relatifs à la génération, aussi elle et peut appliquer plusieurs générations au même modèle. Par contre, son inconvénient majeur concerne l'expression de la sémantique d'exécution qui est totalement transparente aux utilisateurs.

1.2.4 *Tableau comparatif*

Le tableau comparatif suivant est limité aux outils que nous avons pu manipuler concrètement, il s'agit de : EMF, Kermeta, TopCased, et MetaEdit+. La première partie

du tableau contient des critères généraux relatifs à chaque outil : le nom de l'outil, l'origine ou le créateur, l'année de création, les objectifs et les grandes fonctionnalités générales de l'outil (d'après ses créateurs). Tandis que, l'autre partie du tableau contient les critères de comparaison introduits : Langages et formalismes de métamodélisation, Caractéristique de la démarche de construction de l'outil d'exécution et Caractéristiques de l'outil généré.

TABLE 1.1 – Outils de méta-modélisation et critères de comparaison.

Critères \ Outils		MetaEdit+	TopCased	Kermeta	EMF
Généralités	**Nom**	MetaEdit+	Toolkit in Open Source for Critical Applications & Systems Development	KermetaKernel Metamodeling	ConceptBase.cc Eclipse Modeling Framework.
	Origine	Project 'MetaPHOR' Université Jyvaskla Finlande	Constortium / pôle de compétitivité à Toulouse-France	Projet Triskell, IRISA, France	Projet Eclipse GMT, IBM
	Data de création	1990	2004	2005	2002
	Objectifs de l'outil	Modélisation, Définition de nouveaux DSM- génération d'éditeurs graphiques -Édition de générateurs de codes	Développement d'applications critiques et de systèmes	Spécification de métamodèles - Exécution de modèles	Édition de métamodèles syntaxique, génération de code pour la vérification de code Java.
	Fonctionnalités de l'outil	Modélisation, édition graphique de modèles, conception de méthodes, usage de méthodes, générations de code pour vérification, transformation ou exécution de modèles	Mise en œuvre de la première branche du cycle en V pour l'ingénierie du logiciel et/ou matériel	Edition textuel, interprétation, débogage, conversion (depuis/vers Ecore), vérification de contraintes OCL, génération de code pour l'exécution de modèles	Modélisation, Vérification syntaxique génération de code pour la vérification de modèles
Langages de formalismes de métatmodélisation	**Nom**	Ecore	GOPPRR	Kermeta + MOF	Ecore / UML
	Type	Formalise de métamodélisation	Formalise de métamodélisation	Formalise de métaprogammation	Formalisme de métamodélisation

	Concepts	Eattribute, EdataType, Epackage, Efactory, Ereference,	Graphe, Objet, Rôle, Propriété, Port, Relation	Class, operation, attribute, reference, Affectation, litteral ...	Eclass, Eattribute, EDataType, Epackage, Efactory, Ereference
Caractéristique de la démarche de construction de l'outil d'exécution	**Niveau de la complexité de la génération de code**	Élevé, car l'implémentation nécessite des compétences de programmation à part la maitrise de langage de Script Merl	Moyen, grâce aux transformations entre le niveau conceptuel et le code	Élevé : nécessite une maitrise du langage Kermeta et de la programmation orientée aspect	Moyen, à cause des transformations semi automatiques qui génère en partie le code d'exécution
	Niveau de complexité du processus global	Moyen, nécessite la maitrise d'au moins un langage de programmation pour la génération de code. Guidage faible au niveau de la création du code	Élevé, spécifique à une domaine particulier, peu utilisé en d'autres	Élevé : pas de guidage dans le processus. Chaque étape nécessite un effort cognitif important pour la réaliser	Moyen, (-) Plusieurs diagrammes difficiles à personnaliser. (+) intégration automatisée OCL, EMF.
Caractéristiques de l'outil généré		Moyennement facile à maintenir et à porter	Maintenabilité insuffisante : l'interpréteur doit être réécrit pour chaque nouveau langage	Maintenabilité et portabilité insuffisante	Moyennement facile à maintenir et portabilité dépend de celle de la plateforme Eclipse

1.3 MÉTADONNÉES : OUTIL D'INTÉGRATION ET D'INTEROPÉRABILITÉ

1.3.1 *Définition*

Le terme 'metadata' fut employé pour la première fois en 1969 par un informaticien américain Jack E. Meyers dans le but de définir des architectures informatiques appelées plus tard métamodèles. Par la suite, l'année 1995 a connu la fondation de la 'Metadata Coalition', association qui regroupe 53 compagnies d'informatique (dont IBM et Microsoft) et d'autres organismes, dans le but de définir un ensemble de spécifications standards pour l'interchangeabilité et la prise en charge des métadonnées par des outils logiciels. Devant ce début de prolifération de standards de métadonnées, l'années 1997 a connu la naissance des recommandations et des normes pour la mise en place ou le développement de standards de métadonnées afin d'assurer pour ces derniers l'interopérabilité, le mapping, l'évolution, et la flexibilité.

Les métadonnées sont traditionnellement définies comme des données décrivant d'autres données (Greenberg 2003). Le terme metadata est surtout utilisé pour désigner l'information, elles sont perçues comme l'information de fond qui décrit le contenu, la qualité, les conditions et autres propriétés et caractéristiques des données.

Les métadonnées sont aussi appelées des méta informations (De Miguel, Jourdan et Salicki 2002)dont la sémantique est spécifiée formellement par des règles relatives à une donnée dans un contexte particulier. La sémantique d'une métadonnée peut comprendre la description des concepts manipulés dans le contexte, des contraintes de validité, des règles de représentation, de transformation, de configuration et de documentation (Duval et al. 2002). Il s'agit donc de la sémantique formelle, documentaire ou automatisée.

Leur domaine d'application est très vaste. Les métadonnées représentent un moyen efficace pour décrire et identifier tout contenu et ressources utiles pour l'apprentissage ; elles permettent, pendant la conception mais également l'exécution de l'apprentissage, de rechercher, sélectionner, retrouver, combiner, utiliser et modifier des ressources.

Les métadonnées ont quatre principales utilisations permettant une réduction des coûts et du temps et une amélioration des performances humaines (Laforcade 2004) :

— La **catégorisation**, permet d'organiser les informations qui doivent être standardisées en catégories pour les rendre efficaces.
— La **taxonomie**, qui s'agit de structurer et d'organiser les catégories de métadonnées en groupes ordonnés de relations, afin de permettre l'organisation des contenus et aussi de capturer les relations entre catégories.
— La **réutilisation**, lorsque les contenus et leurs métadonnées deviennent davantage structurés et que leur granularité décroît, la réutilisation du contenu augmente. Il est même possible de réutiliser des blocs d'information dans des buts complètement différents ou en les mettant dans des contextes différents en choisissant seulement pour ceux qui sont nécessaires, le bon médium.
— L'**assemblage dynamique**, l'information peut seulement être réutilisée selon son degré de flexibilité mais également assemblée dynamiquement dans le « bon matériau » pour la bonne personne, dans le bon format de média, dans le bon langage, délivré au bon endroit, sur le bon dispositif et au bon moment. Les métadonnées sont organisées formellement dans un métamodèle.

1.3.2 *Les types de métadonnées*

Les métadonnées sont des données sur les données. Les données font référence à la ressource qu'on gère. L'importance de métadonnées se manifeste dans le fait qu'elles permettent aux utilisateurs de gérer les ressources d'une manière plus efficace. Elles représentent un maillon essentiel pour le partage de l'information et l'interopérabilité entre les systèmes. Les métadonnées représentent donc l'ensemble de toutes les données disponibles pour une ressource, mais qui ne sont pas nécessairement contenues dans cette ressource, par exemple : le nom de la ressource, la date de sa modification, le nom du dossier qui la contient, etc.

Les métadonnées sont extraites des ressources lorsqu'elles sont importées (assimilées). En outre, l'ajout de métadonnées permet de classer davantage les ressources. Il existe deux types fondamentaux de métadonnées :

- **Métadonnées techniques**

Ce type de métadonnées est utile pour les applications logicielles qui traitent de ressources numériques qui ne doivent pas être gérées manuellement. Les métadonnées techniques peuvent être déterminées automatiquement par des logiciels, et peuvent être changées lorsque la ressource est modifiée. Les métadonnées techniques disponibles pour une ressource dépendent largement du type de fichier de celle-ci. Par exemple : la taille d'un fichier, débit d'un fichier audio, dimension d'une image…

- **Métadonnées descriptives**

Les métadonnées descriptives sont des métadonnées qui concernent le domaine d'application, par exemple l'activité dont provient une ressource. Les métadonnées descriptives ne peuvent pas être automatiquement déterminées. Elles doivent être créées manuellement ou de manière semi-automatique. Par exemple, un appareil photo avec GPS peut suivre automatiquement la latitude et la longi-

tude auxquelles a été prise une photo et ajouter ces informations aux métadonnées de l'image. A cause du coût élevé de la création manuelle des informations de métadonnées descriptives, des normes ont été définies pour faciliter l'échange des métadonnées entre les systèmes logiciels et les structures.

1.4 LES ÉLÉMENTS CONSTITUTIFS DES MÉTADONNÉES

Lors de l'analyse de différents exemples de métadonnées, plusieurs caractéristiques communes pourront être identifiées, : chaque description de métadonnées est composée d'un ensemble d'éléments, ainsi que de la valeur de ces éléments. Ces derniers sont appelés des instances qui matérialisent les métadonnées (classes, attributs, propriétés, etc) par un ensemble de valeurs. Les éléments sont organisés dans le cadre d'un schéma de métadonnées qui représente un ensemble d'éléments porteurs d'une définition sémantique précise, le schéma peut être standardisé, aussi les éléments des métadonnées sont préalablement spécifiés en utilisant un langage déterminé, tel que : XML schéma, RDF (Resource Description Framework), OWL (Web Ontology Language), etc. La Figure 1.9 illustre les trois éléments constructifs des métadonnées et le lien entre eux : l'ensemble des valeurs des attributs (instances), la définition des éléments (schéma) et le langage de définition du schéma (langage de description) (Amir 2009).

1.4.1 *La gestion des métadonnées*

En vue de garantir une meilleure utilisation et exploitation des métadonnées au sein des organisations, une stratégie de gestion des métadonnées est nécessaire. La création des métadonnées reste une tâche coûteuse et qui prend beaucoup de temps. Pour être vraiment utile, une fois stockées, les métadonnées doivent être centralisées et faciles à entretenir. La gestion des métadonnées a comme principaux objectifs de :

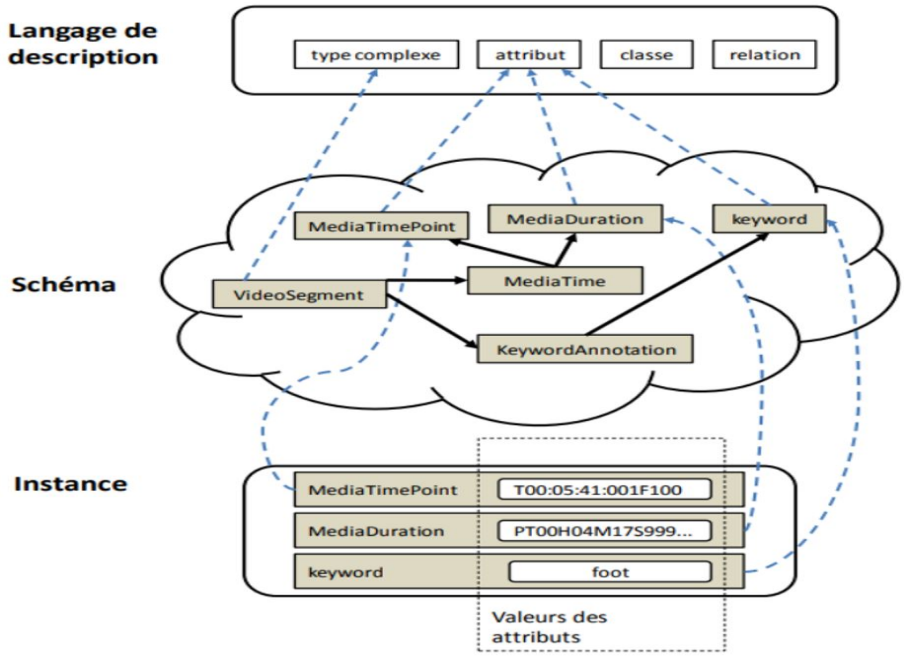

FIGURE 1.9 – Les éléments consécutifs des métadonnées et les liens entre eux (AMIR 2009).

— Favoriser la conformité des métadonnées pour permettre le partage des métadonnées entre les applications d'une organisation. Les métadonnées définies pour une application peuvent être copiées et facilement adaptées pour une autre application.
— Fournir une méthode commune et centralisée de recherche et de gestion de différentes collections de métadonnées.

Ces deux objectifs permettent de réduire les coûts du développement et de maintenance des métadonnées en favorisant la standardisation et la réduction des re-

dondances. En outre, lorsque ces objectifs sont atteints, les métadonnées peuvent fournir des informations significatives et précises, par exemple, l'analyse de l'impact des changements techniques au sein d'une organisation, ou bien des rapports techniques complets sur les systèmes d'application d'une organisation.

1.4.2 *L'interopérabilité des métadonnées*

Le terme interopérabilité désigne l'accès uniforme et transparent à des données hétérogènes à l'aide de mécanismes assurant un fonctionnement coordonné et coopéré entre les différents systèmes d'information (ECKERT, PFEFFER et VÖLKER 2010). L'auteur dans (EUZENAT 2001)définit cinq niveaux d'interopérabilité dans le contexte des bases de données :

— *Encodage* : la capacité de créer une représentation en caractère
— *Lexical* : la capacité de créer une représentation en mots ou symboles
— *Syntaxique* : la capacité de faire une représentation sous forme des phrases structurées
— *Sémantique* : la capacité de créer une signification propositionnelle d'une représentation.
— *Sémiotique* : la capacité de créer une signification pragmatique d'une représentation.

Une définition plus générale de l'interopérabilité est donnée par (HASLHOFER et KLAS 2010), qui la considère comme une propriété qualitative des métadonnées qui permet aux systèmes et aux applications de fonctionner avec – ou d'utiliser ces métadonnées dans la limite du système.

L'interopérabilité se réalise à trois niveaux techniques complémentaires :

— Une description des ressources avec des sémantiques communes issues de différents jeux de métadonnées standardisés ;
— Un contexte générique d'implémentation de ces descriptions dans des langages structurés, interprétables par les machines ;
— Des protocoles informatiques d'échange de ces données normalisées.

Dans le contexte de notre travail, nous nous intéressons à l'interopérabilité des métadonnées dans les systèmes décisionnels que nous abordons dans les sections suivantes.

1.4.3 *Métadonnées et standardisation*

L'interopérabilité implique aussi un certain niveau de standardisation et de normalisation. L'utilisation de standards permet aux utilisateurs d'avoir une terminologie commune permettant la réalisation de recherche efficace pour la découverte des données dans les catalogues. Les métadonnées reposant sur les standards permettent d'avoir un même niveau d'information et d'éviter la perte de connaissance sur les données. Ces standards ont été développés dans le but d'organiser, de représenter et de décrire l'information numérique. La Tableau 1.2 repositionne différents standards bien connus de métadonnées selon les niveaux d'interopérabilité.

Le Consortium W3C, World Wide Web Consortium, , instance internationale permanente chargée de l'avenir du Web composée surtout d'industriels de l'informatique et des TIC (Technologies de l'Information et de la Communication), travaille activement sur l'interopérabilité du Web au travers de l'élaboration et de la diffusion des protocoles et des formats syntaxiques et structurels, de HTTP (Hypertext Transfer Protocol) et HTML (Hypertext Markup Language) à RDF et au Web sémantique. Les premières versions du langage HTML prévoient déjà

TABLE 1.2 – Le positionnement de divers standards de métadonnées selon les niveaux d'interopérabilité

	Standards Traditionnels	**Standards récents**
Jeux de données	Marc	Dublin Core MARC-XML MODS EAD LOOM
Cadre générique d'implémentation	ISO 2709 ISAD (G)	XML RDF Espace de nom
Protocoles	WAIS FTP Z29.50	HTTP OAI-PMH SRU/SRW

des métadonnées dans l'en-tête du fichier. Cependant, ces éléments souffrent d'une standardisation insuffisante des usages et valeurs, et sont sous-utilisées, tant par les producteurs de sites que par les outils de recherche. De plus, ils ne peuvent décrire que des documents HTML et sont peu utilisables en ingénierie de l'information.

En ce qui concerne la description des ressources, le jeu d'élément Dublin Core a donc été créé en 1995 par le DCMI, Dublin Core Metadata Initiative [6], groupe international de professionnels du milieu documentation-bibliothèques et d'autres origines (musées, IETF [7] ...) réuni à Dublin dans l'Ohio. Il correspond à une description « généraliste » s'appliquant à tout type de ressource, pour améliorer la recherche d'information dans le cadre d'une large utilisation. Le DCMI,

6. Dublin Core Metadata Initiative, http ://www.dublincore.org
7. Internet Engineering Task Force, http ://www.ietf.org/

devenu instance permanente, continue à travailler à l'adaptation de ce jeu aux différents contextes d'usage, au travers de groupes thématiques ouverts et d'un congrès/atelier de travail (workshop) annuel.

D'autres grands organismes ou regroupements d'organismes, souvent américains et leaders dans le traitement de l'information de leur domaine d'activité, ont élaboré en parallèle des jeux sémantiques plus spécifiques, répondant à des besoins particuliers de description et de gestion de métadonnées. Citons la FGDC, Federal Geographic Data Committee pour les métadonnées géospatiales, le RLG, Research Library Group, et l'OCLC, On Line Computer Library Center, pour les métadonnées liées au patrimoine culturel et à la préservation, l'Université de Bekerley pour l'archivistique et la fédération des grandes universités américaines dans les programmes «Digital Library Initiative» [8] ; l'IPTC, International Press and Telecommunication Council pour la standardisation des métadonnées images, la Library of Congress pour le développement d'ensembles à orientation bibliographique et la maintenance de nombreux autres jeux, les éditeurs commerciaux de livres et revues et les éditeurs d'outils pédagogiques.

Les organismes normalisateurs et standardisâtes jouent un rôle important dans le processus de développement et de diffusion. Au sein de l'ISO, l'Organisation internationale de normalisation, plusieurs comités techniques sont impliqués dans la normalisation des standards actuels. Le NISO, National Information Standards Organization « identifie, développe, maintient et publie des standards techniques pour gérer l'information dans un monde évolutif et de plus en plus numérique » en lien étroit avec le monde industriel, et s'intéresse fortement aux métadonnées (Woolcott 2017).

8. Digital Library Initiative, http ://www.dli2.nsf.gov/

D'autres organismes travaillent à un niveau plus global, autour de l'usage, de l'implémentation et de l'interopérabilité des métadonnées. Ainsi, l'OCLC est porteur d'une réflexion conceptuelle et développe des outils de type passerelles entre les jeux de métadonnées. Le DDC[9], Digital Curation Center , association d'universités du Royaume Uni soutenue par le JISC, Joint Information Systems Committee, a une activité de recherche, d'expertise et de conseil sur les bonnes pratiques de traitement du document numérique, d'abord destinée aux universités du pays, mais dont l'intérêt dépasse largement ce cadre. Aujourd'hui, plusieurs dizaines de jeux de métadonnées sont reconnus comme des standards, normalisés ou en cours de normalisation, et largement utilisés dans le monde.

Dans ce qui pourrait apparaître comme un foisonnement d'initiatives, la majorité de ces ensembles sont plutôt complémentaires dans leurs objectifs, pour l'objet décrit ou en termes de métier. Presque tous sont implémentés en XML, et les autres évoluent rapidement dans ce sens. En France, malgré l'absence de projet global sur la communication scientifique, ces standards sont utilisés par les grands opérateurs comme l'ABES, la BNF, l'INIST et certaines unités. L'AFNOR, représentant de l'ISO pour la normalisation, participe à leur adaptation aux besoins et spécificités françaises.

Toutefois, il n'existe pas de standard universel et unique pour rencontrer toutes les fonctions, ou même chacune d'entre elles, qui peuvent être exercées grâce aux métadonnées. En pratique, les schémas de métadonnées ont souvent de multiples fonctions. Des applications sont développées par de très nombreuses institutions pour répondre à des besoins particuliers, ce qui ne dispense pas de devoir créer, dans certains cas, un profil d'application propre.

9. Digital Curation Center http ://www.dcc.ac.uk/

1.4.4 *Les métadonnées dans le processus décisionnel*

L'environnement d'entreposage de données et du business analyse est souvent décrit en termes d'une chaîne d'information (Information Supply Chain) (Kimball 1996). Cette métaphore reflète le fait que l'information provient dans cet environnement de sources de données hétérogènes, et passe à travers une séquence de raffinements et de transformations afin de produire l'information utile pour les décideurs (Figure 1.10).

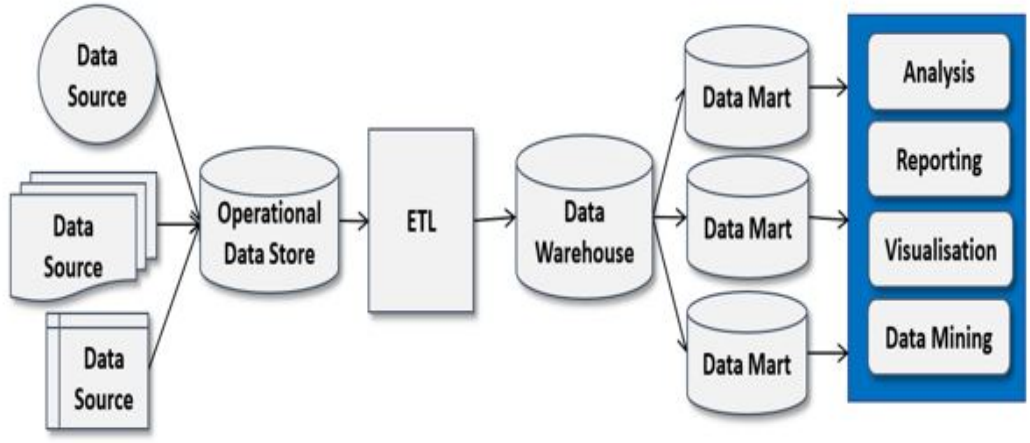

Figure 1.10 – Le flux de métadonnées dans la chaîne décisionnelle.

La première étape de la chaîne décisionnelle se charge de récupérer les données depuis les différentes sources hétérogènes. Cette étape peut prendre différents noms, y compris : l'extraction, la transformation et le chargement de données (Extract Tranform Load (ETL)) ; la normalisation des données ; la construc-

tion de l'entrepôt de données ; le nettoyage et le rassemblement des données, etc. L'étape prévoit l'acquisition de données depuis les différents systèmes transactionnels, la transformation de ces données en certains formats communs, et l'entreposage des données transformées dans une base de données. Cette base de données améliore considérablement les données transformées afin de servir de l'information stratégique pour les décideurs. Cette base de données spécialisée de l'information stratégique est souvent appelée un entrepôt de données. Ce dernier représente le point de stockage de toutes les données utilisées par le système pour analyser les informations. Il assure une étanchéité entre le système opérationnel et le système décisionnel. Son unicité permet une centralisation et une uniformisation de l'information. Les entrepôts de données ont souvent une nature dimensionnelle ; pour cela, ils organisent les données d'une manière uniforme selon les différentes dimensions du business, par exemple : comptes, produits, régions géographiques, unités de vente, etc. Ces dimensions servent comme des clés de recherche pour identifier les données. Par exemple, un business analyste se sert d'un entrepôt de données pour comparer les ventes d'un produit particulier selon différentes régions géographiques et pour une période spécifique. Après la phase d'entreposage de données, l'utilisation des data marts (magasins de données) et des outils de reporting comme des clients du data warehouse prennent lieu.

D'une part, les datamarts peuvent être considérés comme un sous ensemble du Data Warehouse (DW). De ce fait, leur différence se situe dans le fait qu'un datamart réponds à un besoin métier plus spécifique que l'entrepôt de données. Il extrait généralement une partie de l'information disponible dans l'entrepôt de données, ou bien réorganise éventuellement cette information d'une façon différente.

D'autre part, les outils de reporting et d'analyse avancée peuvent être liés directement à l'entrepôt de données ou bien aux data marts. Ces outils ajoutent

une valeur considérable à l'information fournit par l'entrepôt de données. Bien que l'entrepôt de données définisse la vision dimensionnelle de l'information, les outils de reporting et d'analyses fournissent des capacités bien spécifiques, telles que la manipulation de données dimensionnellement organisées et les opérations ou les visualisations spécialisées. Par exemple, l'utilisateur d'une analyse financière avancée peut effectuer des analyses statistiquement complexes à base de l'information dimensionnelle. Les outils de visualisation et de reporting avancés ajoutent de la valeur en permettant à l'utilisateur final de visualiser les résultats de manières différentes, en utilisation des différents types de graphiques graphes, codes colorés, alertes, ou des constructions et des images visuellement multidimensionnelles que l'utilisateur final peut manipuler (tourner, pivoter, reformer, retailler, etc.).

L'ensemble des data marts, des outils d'analyse avancés, et des outils de reporting et de visualisation forment la dernière étape de la chaîne décisionnelle dont laquelle l'information stratégique est transformée en connaissances, de telle sorte qu'elles se présentent de la façon la plus lisible possible dans le cadre de l'aide à la décision.

Afin d'interagir efficacement dans la chaîne décisionnelle, l'ensemble des produits et outils logiciels, qui interfèrent lors de chaque phase de la chaîne, doit être capable de participer à l'échange et le partage des métadonnées à travers cette chaîne. Chaque outil doit avoir une compréhension suffisante de la source et de la nature des données à consommer afin de les transformer en informations pertinentes pour la prise de décision. Ce partage de données, nécessite une définition commune de la structure (l'organisation et le type de données) et la sémantique des données. Comme les données sont généralement définies par les métadonnées, avoir une définition commune de métadonnées est un préalable nécessaire pour parvenir à l'intégration au niveau des données.

Les métadonnées sont largement reconnues comme étant le facteur le plus important dans la réalisation de l'intégration transparente et l'interopérabilité entre des applications et produits logiciels. Toutes les étapes et les outils logiciels de la chaîne décisionnelle dépendent des métadonnées pour décrire les données qu'ils consomment et transforment.

Cependant, il est presque impossible pour la plupart des produits et systèmes logiciels de la chaîne décisionnelle de partager facilement et efficacement l'ensemble des métadonnées. La plupart des produits provenant de différents fournisseurs ont des modèles de métadonnées (ou métamodèles) et des interfaces différentes. Cette différence entraîne des coûts considérables et réduit le retour sur investissement tant pour les fournisseurs que pour les organisations clientes qui tentent d'intégrer des produits, des outils et des applications.

Pratiquement, la plupart des efforts d'intégration nécessitent la construction de ponts (bridges) de métadonnées (Poole et al. 2002), qui représentent une partie de logiciel capable de traduire les métadonnées d'un produit logiciel donné à un format requis par un autre produit logiciel (Figure 1.11).

La construction de ces ponts nécessite d'avoir la connaissance détaillée des structures des métadonnées, ainsi que les interfaces de chaque produit à intégrer ; Cependant, elle s'avère complexe, coûteuse, et entraîne la création de nombreux modules logiciels qui assurent essentiellement la même fonction mais ne peuvent pas être facilement réutilisés dans d'autres efforts d'intégration.

Par conséquent, la nécessité d'avoir un métamodèle commun qui définit l'intégrité du domaine, et permet aux produits et composants logiciels de comprendre les instances de ce métamodèle (les métadonnées partagées), est fortement recommandée. Chaque produit fait correspondre les métadonnées partagées à sa représentation interne de métadonnées.

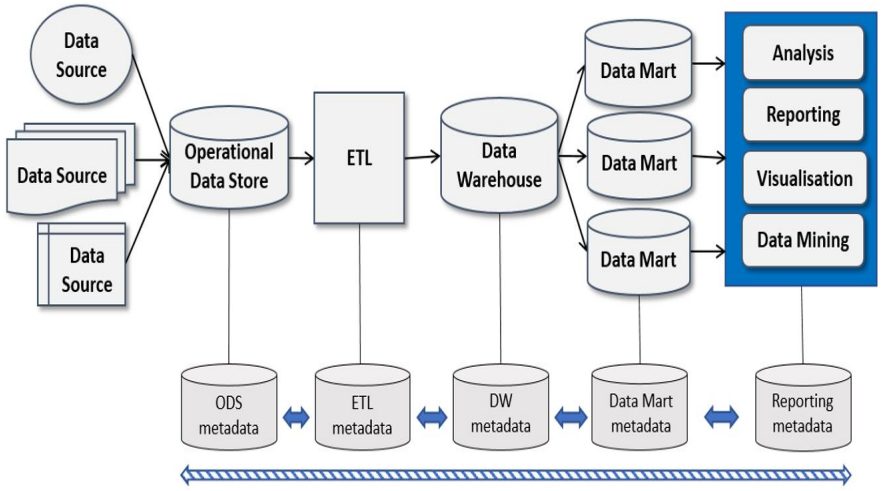

FIGURE 1.11 – Echange des métadonnées dans la chaîne décisionnelle à base de ponts. (POOLE et al. 2002)

Le besoin d'avoir une définition de métadonnées globale et comprise universellement est partiellement adressée par l'utilisation d'un répertoire ou un référentiel de métadonnées (POOLE et al. 2002), qui permet de stocker, de contrôler et de mettre à la disposition du reste de l'environnement tous les composants de métadonnées pertinents.

De ce fait, l'OMG a défini le CWM (METAMODEL 2003) pour permettre l'intégration des outils d'entreposage de données et du business analyse, basé sur l'échange et le partage des métadonnées (Figure 1.12).

Le CWM couvre le cycle de vie complet de modélisation, de construction et de gestion des entrepôts de données. Il définit un métamodèle qui représente les métadonnées aussi bien au niveau métier qu'au niveau technique.

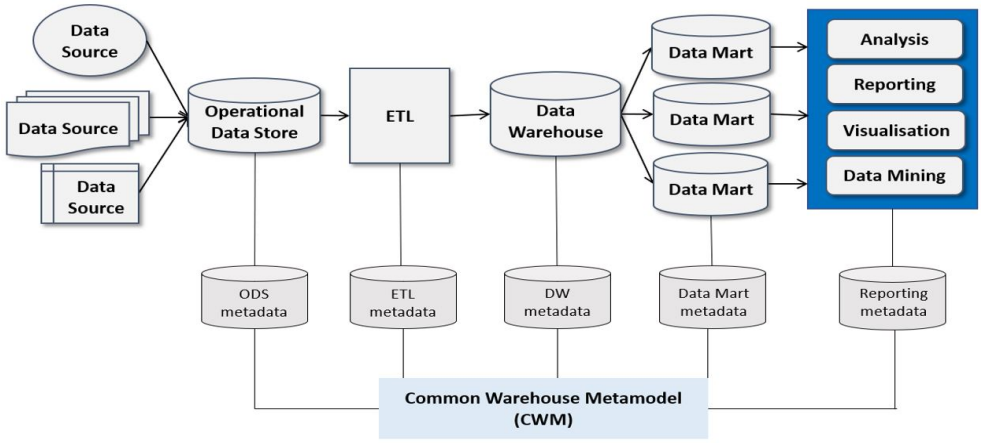

FIGURE 1.12 – L'intégration des métadonnées dans la chaîne décisionnelle via le CWM.

1.5 SYNTHÈSE

Lors du développement d'un système, de nombreux domaines d'expertise peuvent travailler ensemble, coopérer et collaborer en vue d'obtenir une solution satisfaisante. En vue de faire cohabiter les différentes disciplines en les intégrant au sein d'un même ensemble de processus, il semble nécessaire de s'appuyer sur un langage commun. Le moyen le plus sûr et contenant le moins d'ambiguïté et de divergences de compréhension reste sans aucun doute l'utilisation et le partage des modèles (ESTEFAN et al. 2007).
C'est dans ce sens que l'OMG initie l'approche IDM, un thème en pleine expansion aussi bien dans le monde académique que dans le monde industriel. L'architecture IDM préconise le modèle comme unité centrale de tout processus, et se base sur la transformation de modèle pour aboutir à une solution technique sur une plateforme de notre choix à partir de modèles métiers indépendants de toute

plateforme.

Dans ce chapitre nous avons introduit les notions de base de l'IDM à savoir : le modèle, le métamodèle et le méta métamodèle. Nous avons exposé le rôle et les différents types de métamodèle existants dans le domaine de l'IDM. Ensuite, nous avons présenté les différents outils permettant d'implémenter les métamodèles. Nous avons ainsi pu constater que l'EMF est l'outil adéquat pour implémenter notre approche Pour implémenter notre approche nous utilisons l'outil de méta-modélisation EMF puisqu'il utilise UML comme langage de méta modélisation et à cause de sa facilité d'utilisation par rapport aux autres outils de méta-modélisation. Aussi, son principal avantage de cette approche est que la génération de code est faite en respectant les compléments ajoutés manuellement.

Par la suite, nous avons éclaircie le rôle des métadonnées dans la gestion, le partage de l'information et l'interopérabilité entre les systèmes. Ces métadonnées jouent aussi un rôle primordial dans le partage des ressources dans le cycle décisionnel qui présente l'intérêt de notre approche. Vers la fin, nous avons éclaircie le rôle du métamodèle CWM dans l'intégration et l'interopérabilité des métadonnées dans les processus décisionnels.

Dans le chapitre suivant, nous développons en détail le standard CWM et ses différents métamodèles et aussi les différents techniques pour l'étendre en vue d'intégrer plusieurs domaines.

2

LE COMMON WAREHOUSE METAMODEL (CWM)

2.1 INTRODUCTION

Afin d'intéragir efficacement dans la chaîne décisionnelle, l'ensemble des produits et outils logiciels, qui interfèrent lors de chaque phase de la chaîne, doit être capable de participer à l'échange et le partage des métadonnées à travers cette chaîne. Chaque outil doit avoir une compréhension suffisante de la source et de la nature des données à consommer afin de les transformer en informations pertinentes pour la prise de décision. Ce partage de données, nécessite une définition de métadonnées globale et comprise universellement. C'est dans ce but, que le CWM a été adopté pour répondre aux besoins d'intégration de métadonnées et d'interopérabilité. Dans ce chapitre, nous développons en détails le standard CWM et ses différents composants. En outre, nous exposons les différents mécanismes d'extension fournis par le CWM pour intégrer différents domaines. Par la suite, nous présentons les travaux qui ont étendu le CWM pour répondre à plusieurs objectifs et nous dressons un tableau récapitulatif pour avoir une vision globale sur les problématiques, les aspects qui manquent dans le standard CWM et l'objectif de l'extension pour les résoudre.

2.2 LE STANDARD CWM

2.2.1 *Définition et architecture du métamodèle*

Le CWM est destiné à l'intégration des outils d'entreposage de données et du business analyse, basé sur l'échange et le partage des métadonnées. CWM est un métamodèle complet qui représente le domaine d'entreposage de données et de business analyse. En tant que méta-modèle, CWM fournit à la fois la syntaxe et la sémantique nécessaires pour construire des métadonnées (par exemple, des modèles ou des instances du métamodèle) qui décrivent tous les composants du cycle décisionnel. Le CWM est composé d'un ensemble de méta-modèles reliés entre eux, dont chacun représente un sous-domaine du processus décisionnel. Il est caractérisé par une architecture de paquetage modulaire construite à la base du formalisme orientée-objet Figure 2.1.

Le métamodèle est organisé en 21 paquets séparés et regroupés en cinq couches, dont les métamodèles (paquets) appartenant à une couche particulière dépendent uniquement des méta-modèles qui appartiennent à une couche inférieure à celui-ci, afin d'éviter le couplage des paquets du même niveau ou d'un niveau inférieur à un niveau supérieur (POOLE et al. 2002).

La construction du métamodèle sous forme de paquetage avait pour objectif de maximiser et de faciliter l'utilisation du CWM. Le comité du CWM a compris dès le départ qu'aucun outil ne supportera tous les concepts de CWM. Afin de rendre l'utilisation du CWM assez simple que possible, la structure des paquets a été construite sans accouplement horizontal et avec un couplage vertical assez faible que possible. De plus, aucune dépendance n'existe le long du plan horizontal des paquets. Cela signifie que l'implémentation d'un outil avec le CWM n'aurait besoin que des paquets verticaux correspondants à cet outil.

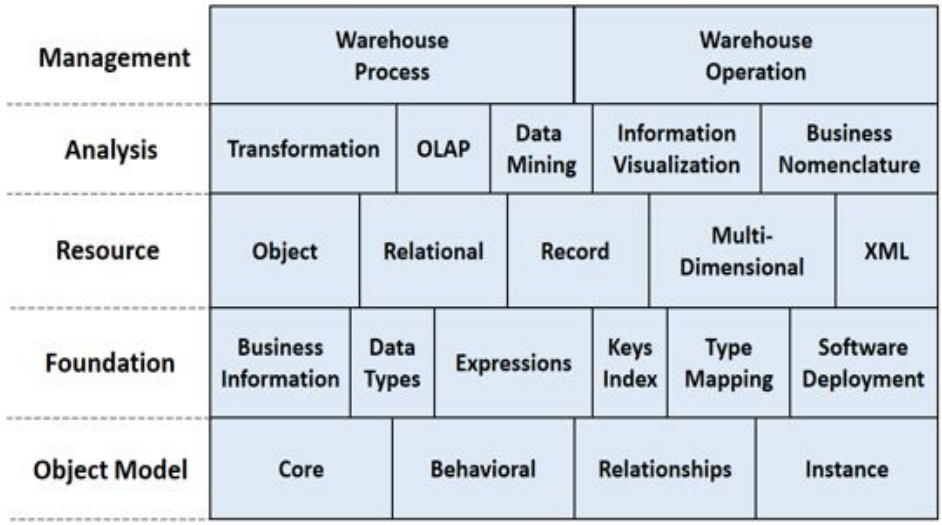

FIGURE 2.1 – l'architecture des paquets du CWM.

Suivant ces considérations, le CWM est un métamodèle complet de niveau M2 de l'architecture de méta modélisation (voir section 1.3) divisé en un ensemble de métamodèles étroitement liés. Les cinq couches du métamodèle CWM sont définis comme suit (POOLE et al. 2002) :

— La **couche Object Model** : représente une sous couche d'UML, utilisée comme le métamodèle de base pour CWM. Elle contient les paquets qui définissent les concepts fondamentaux pour l'ensemble du métamodèle, les relations, et les contraintes requises par le reste des paquets du CWM. Ces concepts fournissent un environnement clair pour définir les autres paquets du CWM, ce qui permet une meilleure concentration sur l'objectif

individuel de chaque paquet et une minimisation des dépendances entre les paquets. La couche *Object Model* se compose de quatre métamodèles :

- **Core** : contient les classes et les associations de base utilisées par les autres paquets du CWM. Le métamodèle ne dépend d'aucun autre paquet du CWM. Le Core intègre l'infrastructure UML de base nécessaire pour définir les data store non-Orienté-Objet, tels que les bases de données relationnelles et les fichiers d'enregistrement, sans introduire exclusivement les concepts orientés objet. Ce paquet contient également les classes support et les types de données utilisés largement par les autres paquets du CWM.
- **Behavioral** : rassemble les classes qui fournissent les caractéristiques comportementales des systèmes orientés-objet aux autres classes du CWM, tels que les opérations et les procédures.
- **Relationships** : décrit la manière dont les objets du CWM sont liés entre eux. CWM définit deux types de relation, à savoir l'association et la spécialisation ou la généralisation. Comme les classes du CWM sont définies dans le niveau M2 de l'architecture de méta-modélisation, les classes Generalization, Association, and AssociationEnd fournissent des instances spécifiques aux autres classes du CWM qui devront être reliées et classées dans des hiérarchies au niveau M1 de l'architecture.
- **Instance** : définit les éléments de modélisation nécessaire pour représenter les instances de certains éléments de modélisation. Par exemple, l'association entre une classe et un objet spécifique, comme une instance de cette classe, peut être modélisée en utilisant les constructeurs du métamodèle Instance.

— La **couche Foundation** : contrairement à la couche Object Model destinée à un usage générale, cette couche étend la couche *Object Model* et fournit des services spécifiques aux paquets du CWM appartenant aux couches supé-

rieures. La particularité de la couche Fondation est que, en définissant ces concepts à un niveau très abstrait de l'architecture CWM, nous nous assurons que ces concepts ne soient définis qu'une seule fois et qu'ils peuvent être réutilisés dans des contextes plus spécifiques. La couche Fondation se compose des métamodèles suivants :

- **Business Information** : Ce paquet fournit des services à usage général aux autres paquets de CWM pour l'enregistrement des faits d'un environnement business intelligent dans lequel un échange d'entrepôt de données a eu lieu. Cependant, le paquet ne fournit pas une représentation complète de l'environnement du business, mais uniquement les services que les fondateurs du CWM ont estimé nécessaires pour accomplir un échange de métadonnées dans le cycle décisionnel.
- **Data Types** : étend quelques éléments de modélisation de la couche Object afin de définir de nouveaux éléments pour représenter les types de données de base nécessaire lors d'un échange de métadonnées. Ce paquet fournit l'infrastructure requise pour supporter la définition des types de données primitives (Integer, Boolean, etc.) et structurées (XML, Resource Description Framework (RDF), etc.).
- **Expressions** : une expression dans les langages de programmation désigne une combinaison ordonnée de valeurs et d'opérations qui pourra être évaluée pour produire une autre valeur, un ensemble de valeur, ou un fait. Ainsi, les expressions peuvent être utilisées pour décrire les relations logiques entre leurs valeurs composantes ou les étapes d'une séquence algorithmique donnée. Puisque l'un des objectifs majeurs du CWM est de promouvoir l'échange de métadonnées dans différents environnements, la capacité d'échanger des expressions est primordiale. Malheureusement, cette fonctionnalité n'existe pas souvent dans les systèmes d'entreposage de données. Dans la plupart des cas, les expéditeurs et les récepteurs partagent généralement la syntaxe ou la séman-

tique. Les classes du paquet Expression permettent l'échange ou l'enregistrement d'une expression du texte ou bien même le nom du langage dont ce texte est écrit.

- **Keys Indexes** : contient les classes et les associations qui représentent les notions de Clés et d'Indexes. Ces concepts sont fondamentalement importants pour la construction des modèles de structure de base de données relationnelles et aussi pour l'enregistrement des bases de données orientée multidimensionnelle (par exemple, la définition d'une clé unique pour une dimension OLAP).
- **Software Deployment** : ce paquet enregistre la manière dont les parties logicielles et matérielles sont utilisées dans un entrepôt de données. Les outils de gestion des entrepôts compatibles avec le CWM peuvent utiliser les informations fournies par ce paquet pour localiser les composants matériels et logiciels de l'entrepôt. Le paquet représente une grande partie des configurations opérationnelles utiles pour les autres paquets du CWM. Cependant, il ne peut pas être utilisé comme un modèle complet ou à usage général pour toute configuration de traitement de données.
- **Type Mapping** : comme mentionné précédemment dans le paquet Data Types, la clé importante d'un échange d'entrepôt de donnée réussi est la capacité de transférer correctement les données définies par les différents types de systèmes de divers produits logiciels. Au contraire du paquet Data Type qui définit les types de données, le paquet Type Mapping définit la notion de type de système comme une collection de types de données et aussi l'échange et le transfert des types de données entre les différents systèmes.

— **La couche Resource** : les paquets de cette couche décrivent la structure des ressources de données qui agissent comme sources ou cibles pour les échanges des métadonnées dans CWM. La couche contient les paquets qui

décrivent les bases de données et les applications Orientées-Object, les systèmes de gestion de base de données relationnelles et traditionnelles, les sources de données orientée-enregistrement comme les fichiers et les modèles d'enregistrement, les bases de données multidimensionnelles créées par les outils OLAP, et les flux ou les fichiers XML. La couche Resource se compose des métamodèles suivants :

- **Object** : Le CWM contient déjà un modèle d'objet parfaitement adapté (dans la couche Objet Model) dont les paquets peuvent être utilisés directement pour créer des descriptions de ressources de données orientées-objet. Le paquet Object de cette couche présente une extension de la couche Object Model en vue de prendre en charge les fonctionnalités supplémentaires qui ne sont pas couvertes par les paquets de la couche Object Model.
- **Relational** : Contient les classes et les associations qui représentent les métadonnées des sources de données relationnel, tel que : les tables, les colonnes, les vues, les procédures, etc.
- **Record** : fournit l'infrastructure requise pour décrire une large variété des structures de données orientées enregistrement.
- **Multidimensional** : contient les classes et les associations dédiées pour les bases de données multidimensionnelles qui représentent les structures des supports physiques créées et utilisées par les outils OLAP.
- **XML** : définit les classes du métamodèle CWM requises pour supporter la description des documents XML comme étant des ressources de données dans le DW.

— **La couche Analysis** : cette couche supporte les activités d'entreposage de données qui ne sont pas directement liées à la description des données sources et cibles. La couche contient les métamodèles suivants :

- **Transformation** : contient les classes et les associations qui représentent les métadonnées utiles pour les outils de transformation de données. Le métamodèle prend aussi en charge l'extraction, la transformation et le chargement (ETL) et les services de lignage de données.
- **OLAP** : contient les classes pour la visualisation et la manipulation des données d'entrepôt (cubes, dimensions, etc.).
- **Data Mining** : contient les descriptions des résultats des activités du data mining par la représentation des modèles et des valeurs d'attributs utilisées dans cette exploitation.
- **Information Visualization** : contient les classes et les associations qui représentent des métadonnées d'outils de visualisation d'informations.
- **Business Nomenclature** : contient les classes et les associations qui représentent les métadonnées dans les taxonomies et les glossaires commerciales. Il capture les concepts d'entreprise génériques sous la forme d'un vocabulaire structuré.

— **La Couche Management** : cette couche fournit les fonctionnalités requises pour supporter la gestion et l'exploitation quotidienne d'un DW au moyen de flux d'informations. Les paquets de cette couche font du CWM une partie active et bien intégrée dans l'environnement décisionnel. En outre, ils servent comme base pour construire d'autres activités de gestion d'entreposage de données plus élaborées en utilisant les mécanismes d'extension fourni par CWM qu'on évoquera dans de chapitre. Cette couche se compose des métamodèles suivants :

- **Warehouse Process** : ce paquet décrit le flux d'information dans le data warehouse. Les flux d'information s'expriment sous forme de transformations présentées par le paquet Transformation et peuvent être documentés lors d'une activité de transformation (Transformation Activity)complète ou au niveau de l'une des étapes de transformation

(Transformation Steps). Ce métamodèle permet de modéliser les processus spécifiques d'entreposage de données, tel que le processus ETL.

- **Warehouse Operation** : enregistre les évènements dans le DW, tels que : les exécutions des transformations, les mesures et les demandes de modification. Le paquet définit aussi des éléments de modélisation qui sont utilisés pour construire des métadonnées définissant les opérations spécifiques, telles que les événements planifiés et leurs interdépendances. Ces métadonnées sont utiles surtout pour les outils ETL.

2.2.2 *Fonctionnalités du CWM*

Le métamodèle CWM fournit un grand nombre de fonctionnalités et de caractéristiques en s'inspirant du domaine d'entreposage de données, on cite :

— **La réutilisation des concepts de UML**
Le métamodèle CWM possède comme base l'Object Model basé sur une version du métamodèle UML dans laquelle les aspects qui correspondent aux scénarios du cycle décisionnel sont représentés. Le reste du métamodèle CWM est construit à la base de l'extension du métamodèle Object Model.

Plusieurs types d'objets et d'associations du Core UML sont représentés par l'Object Model du CWM. Le cas échéant, les éléments du Object Model sont sous typés pour fournir plus d'objets spécifiques pour le métamodèle CWM, normalement avec des attributs ou des associations supplémentaires. Tous les types d'objets CWM sont directement ou indirectement des sous-types de types de modèles d'objets appropriés, et héritent donc de leurs attributs et associations.

La réutilisation des concepts d'UML présente de nombreux avantages. Il permet à la spécification de CWM de tirer profit des importants investissements dans

le développement et le raffinement du métamodèle UML. La prise de conscience générale des concepts UML devrait aider à comprendre la spécification CWM et son modèle de base Objet Model. Elle permet également une intégration facile des modèles UML comme partie des métadonnées de l'entrepôt de données. Le principal avantage de cette fonctionnalité est la réutilisation des associations et des concepts communs à travers les différents niveaux du standard.

— **La modularité**
Le métamodèle CWM est divisé en un ensemble de couches et de paquets. Cela permet d'une part une meilleure compréhension du métamodèle en le divisant en petites unités, d'autre part, la structure de CWM permet aux utilisateurs et aux développeurs de choisir uniquement les paquets du métamodèle qui répondent à leurs besoins et d'ignorer ceux qui ne sont pas utiles pour l'implémentation. La capacité d'implémenter et d'utiliser que des parties du métamodèle global est une caractéristique très importante du métamodèle CWM qui devrait réduire la barrière d'accès aux outils qui cherchent à utiliser CWM comme moyen d'échange et d'interopérabilité de leurs métadonnées (Poole et al. 2002).

— **Un modèle générique**
Une attention particulière a été prise par l'OMG pour s'assurer que le métamodèle CWM a été rendu aussi générique que possible pour couvrir toutes les étapes du cycle décisionnel et que seules les informations partageables entre différentes implémentations ont été incluses dans le métamodèle. La possibilité de partage de l'information a été vérifiée et raffinée en examinant les besoins en métadonnées de plusieurs implémentations différentes ainsi que d'une large gamme de configurations représentatives d'entrepôt de données.

— **Séparation entre les modèles logiques et physiques du DW**
Une autre fonctionnalité importante du CWM est qu'il permet de maintenir une séparation complète entre les modèles logiques et physiques dans un entrepôt de

donnée. Cette fonctionnalité offre aux utilisateurs de CWM le plus grand niveau de flexibilité en fournissant des vues logiques pilotées par les utilisateurs finaux et les besoins physiques des technologies de base de données, ceci leur permet d'isoler les exigences du modèle logique de l'utilisateur final des limitations physiques des systèmes de base de données spécifiques.

2.2.3 *Les standards de base du CWM*

L'objectif principal du CWM est de permettre l'échange les métadonnées entre les différents outils, plateformes et répertoires d'entreposage de données et du business intelligence dans un environnement hétérogène et distribué. Le CWM se base dans sa construction sur trois principaux standards de l'OMG :

— **UML** : représente le standard de modélisation de l'OMG.
— **MOF** : le standard de l'OMG pour la définition des métamodèles et de leurs éléments.
— **XMI**(XML Metadata Interchange) : représente le standard de l'OMG pour l'échange des métadonnées

2.2.3.1 *LE MOF*

Le MOF fait partie des standards définis par l'OMG et il peut être vu comme un sous-ensemble d'UML. Le MOF représente le quatrième niveau de la pyramide de la méta-modélisation. Il s'auto-définit en termes de classes, d'associations, de paquetages, de types de données. Cela lui permet aussi de pouvoir définir d'autres méta-modèles. Le MOF et l'UML constituent le cœur de la technologie MDA du fait qu'ils permettent de créer des modèles technologiquement neutres. Le MOF spécifie la structure et la syntaxe de tous les métamodèles comme UML, CWM et SPEM, autrement dit, il sert comme modèle pour ces métamodèles (Specification 2001). Il spécifie aussi des mécanismes d'interopérabilité entre ces

métamodèles. Grâce à ces mécanismes d'échanges, le MOF peut faire cohabiter plusieurs métamodèles différents. Il définit pour chaque méta-modèle (ANDRE 2004) :

— Un modèle abstrait d'objets MOF génériques et leurs associations.
— Un ensemble de règles pour exprimer un méta-modèle MOF à l'aide d'interface IDL (Interface Definition Language).
— Un ensemble de règles sur le cycle de vie et la composition des éléments d'un méta-modèle MOF.
— Une hiérarchie d'interfaces réflexives permettant de découvrir et de manipuler des modèles basés sur des méta-modèles MOF dont l'interface n'est pas connue.

2.2.3.2 *UML*

Le langage UML a été adopté par l'OMGcomme standard de modélisation de système informatique en novembre 1997. Les concepts définis par l'UML sont très proches de ceux définis par le MOF. Ainsi, le MOF utilise les représentations graphiques de l'UML. L'UML a apporté au domaine de la méta-modélisation sa notation graphique et ses concepts objet. Le langage UML permet la modélisation de systèmes indépendamment de toute démarche ou de plateforme.

Le CWM contient le paquet Object Model basé sur le métamodèle UML. Ce paquet représente une version de l'UML dont seulement les aspects pertinents pour le scénario d'entreposage de données sont représentés. De plus, le métamodèle CWM représente une extension de l'Object Model de l'UML. Toute classe de CWM hérite des classes de l'Object Model. En outre, la notation UML est utilisée dans la représentation des diagrammes du métamodèle CWM et les contraintes additionnelles dans le métamodèle CWM sont représentées par l'OCL, comme définies dans la spécification UML.

2.2.3.3 *XMI*

XML Metadata Interchange (XMI) est le langage d'échange entre le monde des modèles et le monde XML. C'est le format d'échange standard entre les outils compatibles MDA. XMI décrit comment utiliser les balises XML pour représenter un modèle UML en XML. Cette représentation facilite les échanges de métadonnées entre les différents outils ou plates-formes de modélisation.

CWM utilise XMI comme son mécanisme d'échange de métadonnées. En d'autres termes, la force et la flexibilité de XMI se manifeste non seulement pour échanger les métadonnées décisionnelles mais aussi pour échanger les instances du métamodèle CWM lui-même (METAMODEL p. d.).

2.2.4 *Les mécanismes d'extension du CWM*

Bien que les métamodèles du standard CWM reflètent la problématique d'un domaine particulier du cycle décisionnel, et doivent être aussi une description complète de ce domaine ; dans certaines situations le besoin de représenter pour échanger ou produire de nouveaux éléments ou des métadonnées qui ne sont pas représentés dans ce métamodèle s'impose. Dans cette situation, le métamodèle peut fournir des constructions standard pour définir de nouveaux éléments. La façon dans laquelle ces éléments doivent être interprétés ainsi que leur signification sémantique ne peut pas être déduite du métamodèle CWM, elles dépendent plutôt des outils participant à l'échange des métadonnées dans le cycle décisionnel.

Dans d'autres situations, le métamodèle CWM peut être étendu pour intégrer les concepts d'un nouveau domaine. Et puisque les métamodèles, en général, doivent être formulés selon un langage formel. L'extension d'un métamodèle existant avec des nouveaux concepts signifie que le langage de ce métamodèle

doit fournir ses propres mécanismes pour faire cette extension.

Aussi, comme nous avons cité dans le chapitre précédent, une solution d'intégration de métadonnées basée sur un modèle devrait fournir des moyens standard pour étendre les modèles, ce qui est nécessaire pour définir des métadonnées hautement spécifiques aux produits non compatibles avec CWM. Il est également nécessaire de faire une extension du métamodèle pour construire de nouveaux métamodèles pour représenter d'autres sous-domaines supplémentaires à ajouter à la solution globale de la chaîne logicielle.

Le CWM utilise la sémantique et les mécanismes d'extension standard fournis par UML pour étendre les modèles et définir de nouveaux métamodèles représentant de nouveaux sous-domaines. Les mécanismes d'extension UML se composent des éléments de modélisation Tagged Value, Stereotype et Constraint. Ces éléments de modélisation sont définis dans le métamodèle Core de la couche Object du CWM. Par conséquent, CWM définit lui-même trois mécanismes d'extension pour définir de nouveaux métamodèles, à savoir

2.2.4.1 *La technique d'héritage*

Le métamodèle CWM est construit lui-même à la base des sous classes (héritage) d'UML, donc l'utilisation de la même technique pour l'étendre semble très sensible. En fait, l'équipe de conception de CWM a préféré ce mécanisme d'extension, et tous les paquets d'extension CWM utilisent exclusivement cette technique.

2.2.4.2 *La technique des TaggedValue et des Stéréotypes*

Dans certains cas, l'utilisation de l'héritage comme technique d'extension du métamodèle s'avère compliquée surtout quand il s'agit d'une situation simple où on aura besoin d'ajouter que quelques attributs pour réaliser l'extension. Les

classes de base Stereotype et TaggedValue sont créées pour répondre à cette situation :

— **TaggedValue** (valeur étiquetée) représente un simple mécanisme d'extension qui permet l'ajout d'attributs à n'importe quel élément de modèle. Les TaggedValues sont enregistrés comme paires nom-valeur. Par exemple, le nom d'une école peut être enregistré comme "Ecole = FST" ; Ecole est le nom, et FST est la valeur. Malgré que la TaggedValues est un mécanisme d'extension utile et facile à utiliser, son utilisation peut empêcher l'échange.
— **Stéréotypes** : représentent une forme de mécanisme d'extension qui permettent d'étiqueter une classe existante du CWM avec un nom qui reflète plus précisément le rôle qu'elle joue actuellement. Ils représentent des annotations que l'on applique aux éléments de modélisation pour les spécialiser. Comme pour les TaggedValues, les Stéréotypes peuvent limiter l'échange des métadonnées s'ils sont utilisés de manière inadéquate. Ce mécanisme d'extension (TaggedValues et Stéréotypes) est souvent utilisé pour des petites extensions, par exemple lors de l'ajout des attributs supplémentaires pour les modèles objets, qui ne sont pas suffisamment signifiante pour nécessiter la création d'un modèle spécifique.

2.2.4.3 *Extension basée sur XMI*

Les échanges des métadonnées dans le CWMsont réalisés à l'aide de fichiers XML qui sont conformes à la spécification XMI de l'OMG. Les techniques d'extension de XMI peuvent être utilisées avec n'importe quel échange du CWM. C'est une extension complémentaire généralement utile pour les développeurs du CWM qui utilisent les extensions XMI.

Comme les échanges de métadonnées du CWM sont réalisés à l'aide des fichiers XML conformes à la spécification XMI d'OMG, les techniques d'extension natives

d'XMI peuvent être utilisées avec n'importe quel échange CWM. Les extensions XMI prennent en charge la puissance de modélisation complète du standard XMI et ne souffrent pas des problèmes d'incomplétude contrairement aux extensions par les Stéréotypes et les Tagged Values.

Cependant, le principal inconvénient des extensions XMI, est qu'elles n'agissent pas directement sur l'échange des métadonnées CWM. Bien que cette technique puisse générer des extensions dans l'outil de réception, elle représente en quelque sorte un passage sous CWM et peut ne pas être directement visibles via un logiciel CWM qui n'est pas préconditionné pour comprendre la sémantique des extensions spécifiques transmises. Aussi que les extensions XMI soient techniquement complètes et générales, les détails de leur livraison restent assez techniques. Cela dit, l'utilisation des extensions XMI pourrait être considérée comme une technique destinée aux programmeurs professionnels intéressés par la construction des définitions d'échange CWM spécifiques à un outil.

Ainsi, le choix de la technique d'extension adéquate dépend de l'objectif à atteindre par cette extension, de l'environnement dans lequel cette extension aura lieu, et des compétences des parties prenantes pour réaliser cette extension.

2.2.5 *Standards basés sur le CWM*

Il existe des outils et des standards qui se basent dans leurs constructions sur le standard CWM, parmi eux on cite :

2.2.5.1 *XELOPES (eXtEnded Library fOr Prudsys Embedded Solutions)*

XELOPES est un produit de la société PrudSys (Thess et Bolotnicov 2004). Il représente une bibliothèque open source de fouille de données embarquées implémentée dans diverses langages (Java, C++, C#). XELOPES a été construit

sur la base du standard CWM et offre par ailleurs plusieurs avantages dont :

— Support du standard CWM : possibilité d'échanger des données avec d'autres applications décisionnelles. Support des standards décisionnels, comme : PMML, JOLAP, JMI sont supportés.
— Indépendance à la plateforme : peut être exécuté sur diverses plateformes grâce aux différentes implémentations disponibles.
— Indépendance à la source de données : le logiciel peut accéder à divers types de sources de données incluant les fichiers plats, les bases de données. Aussi, en plus de la possibilité de traiter des données changeantes, il est capable de traiter de larges volumes d'informations. XELOPES est un outil complet et tout aussi riche. Son architecture complète, robuste et complexe témoigne de sa particularité qui réside essentiellement en la capacité, contrairement à d'autres, d'accéder à des données situées dans un entrepôt.

2.2.5.2 *Pentaho*

C'est une solution d'informatique décisionnelle considérée comme le leader des suites de BI dans le monde de l'Open Source Figure 2.2. C'est une plateforme décisionnelle extrêmement complète et autosuffisante, qui permet non seulement d'utiliser les différents outils décisionnels open source depuis une interface unique et simple d'utilisation, mais aussi d'étendre et de combiner leurs fonctionnalités grâce à l'utilisation d'un moteur de workflow. Elle repose sur JFreeReport pour les rapports, Weka pour le datamining, PDI pour l'ETL, et Mondrian pour le serveur OLAP (Bouman et Van Dongen 2009). Pentaho porte sur toute la chaîne décisionnelle et utilise différents outils et composants :

— Pour la collecte et l'intégration : l'outil ETL Kettle (Pentaho Data Integration).

— Pour la diffusion : un serveur d'application Pentaho BI Server, déployé sur un serveur d'application tel que JBoss ou Tomcat
— Pour la présentation : l'outil de design de rapports Pentaho Report Designer

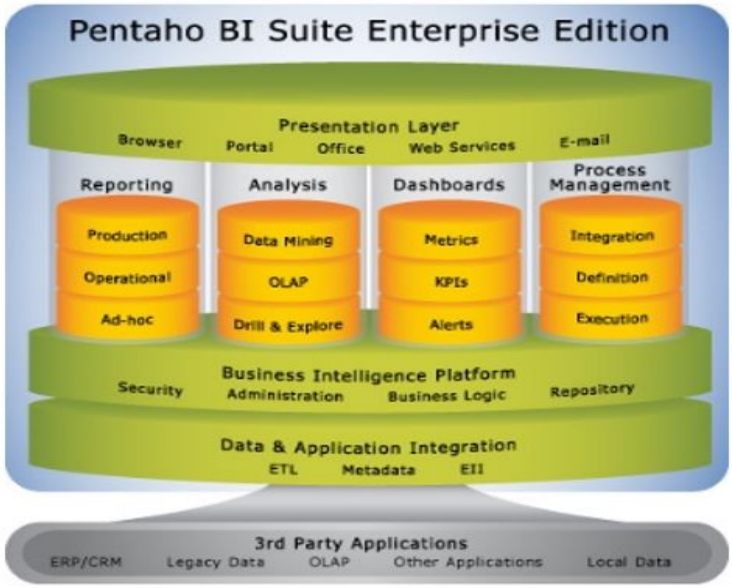

FIGURE 2.2 – Architecture de Pentaho.

2.3 EXTENSION DU CWM : TRAVAUX ASSOCIÉS

Dans le but de fournir plus de support de métadonnées pour l'entreposage de données, (MELCHERT et al. 2005)ont discuté les domaines d'application de métadonnées dans lesquels le CWM devrait être étendu. En se basant sur cette étude, plusieurs approches ont utilisé les mécanismes d'extension présentés ci-

dessus pour étendre le métamodèle CWM afin de construire leurs propres métamodèles pour répondre à des objectifs divers. En conséquence, différents travaux ont utilisé la technique d'héritage pour étendre le CWM : (ZHAO et HUANG 2006) ont exploité une approche pour développer un système intelligent qui supporte le raisonnement automatisé sur les métadonnées du CWM, puisque ce dernier ne supporte pas les capacités de version. Chaque version de métadonnées dans CWM doit être enregistrée dans un fichier XML différent, ce qui rend la capture des informations lors de l'évolution des métadonnées difficile. Pour cela, afin de fournir un support pour le développement des composants des systèmes d'entreposage de données, et aussi d'améliorer la fiabilité de l'intégration des métadonnées dans ces systèmes, l'approche a proposé une extension et une formalisation du métamodèle CWM et des métadonnées basées sur ce dernier en termes d'une logique formelle de la famille des Logiques de Description. Le métamodèle Core de la couche Object Model du CWM a été étendu par six nouvelles classes (VersionedModel, CompositeModel, PrimitiveModel, Trace, HorizontalTrace, EvolutionTrace) pour permettre au CWM de supporter l'historique de l'évolution des métadonnées en capturant et en rassemblant les informations nécessaires pour raisonner sur cette évolution, afin d'établir une cohérence lors de l'évolution des métadonnées.

L'augmentation croissante des volumes de données dans les organisations, conjuguée à une évolution régulière d'information, engendre des préoccupations importantes autour de la qualité des données. Néanmoins, le métamodèle CWM ne fournit aucun support pour les problèmes de qualité de données. Dans ce sens, (ZHAO et HUANG 2006) ont proposé une extension du métamodèle CWM dans le but de fournir des directives de modélisation pour stocker des spécifications formelles de règles de qualité de données. L'extension a concerné les métamodèles de plusieurs couches du CWM, à savoir les couches : Fondation, Object Model, Analysis, et Management, qui ont été étendues par des nouvelles classes (dictionnary, Enumeration Rules, Data Quality Rules, etc.) afin d'atteindre

l'applicabilité aux activités de qualité de données à la fois dans un contexte opérationnel et dans un contexte d'entreposage de données. Dans le même contexte, (Olbrich 2010) a présenté une extension du CWM pour répondre aux problèmes de qualité de données. Cette extension présente un nouveau concept pour diffuser et stocker les données avec leurs informations de qualité de données décrite. Dans ce but, les paquets Resource de la couche Fondation et les paquets Core et Instance de la couche Object Model ont été étendus par deux nouvelles classes (DataQuality et DQWindow) pour répondre aux problèmes de qualité de données dans le but de stocker et d'analyser la masse de qualité d'information de manière efficace.

Aussi, afin d'établir et d'activer le lignage des données et des métadonnées dans le processus décisionnel, (Santana et Carvalho Moura 2004) ont étendu le paquet Transformation de la couche Analysis du CWM pour intégrer les métadonnées capturées lors des processus de transformation en utilisant les métadonnées CWM. Le CWM a été étendu par des nouveaux éléments et par de nouvelles capacités de transformation. L'extension a concerné plusieurs paquets du CWM, principalement les paquets : OLAP, Relational, et Transformation. Cependant, seulement le paquet Transformation a été étendu par des nouvelles classes qui représentent toutes les propriétés nécessaires de permettre le lignage des données et des métadonnées, et aussi les mécanismes des métadonnées pour assurer leur interopérabilité. De surcroit, (Di Tria, Lefons et Tangorra 2012) révèlent aussi que le métamodèle CWM ne fournit pas les éléments nécessaires pour supporter le lignage des métadonnées et ne permet pas de représenter les métadonnées nécessaires pour les systèmes de réponses à requêtes approximatives. Pour cela, DI TRIA a défini un nouveau métamodèle OL2AP supplémentaire (OnLine Approximate Analytical Processing) par extension du paquet Relational de la couche Resources du CWM. Le métamodèle a été ajouté à la couche Analysis du CWM. Le nouveau métamodèle a pour but non seulement d'intégrer les métadonnées capturées durant les processus de transformation pour définir les

métadonnées à utiliser, mais aussi de représenter les métadonnées standard pour les systèmes de réponses aux requêtes approximatives en vue de réduire les données stockées dans les entrepôts de données.

De plus, le processus de préparation de données dans le système décisionnel s'avère nécessaire pour appliquer les algorithmes du datamining, les outils ETL existants offrent une riche fonctionnalité de traitement de données, cependant ils ne couvrent pas toutes les étapes nécessaires. Pour cela, (BATASOVA et al. 2015) proposent une approche qui étend les capacités des outils ETL existants en étendant les métamodèles Transformation et Data Mining de la couche Analysis du CWM pour pallier les problèmes de préparation de données pour le processus de data mining.

Aussi, (BATASOVA et al. 2015) considèrent le CWM comme le standard de Data Mining le plus profond ; de là, le paquet Datamining a été étendu afin de construire XELOPES, librairie universelle de Data Mining, qui implémente les méthodes et les algorithmes pour résoudre les problèmes du Data Mining.

Sur un autre plan, les infractions en matière de sécurité et de confidentialité des entrepôts continuent à poser une menace en raison de la grande sensibilité de l'information qui peut y être découverte, récemment, un certain nombre de solutions de sécurité des entrepôts de données ont été proposés : Dans son travail, Soler a proposé L'extension SECRDW (SeCure Relational Data Warehouse) du paquet Relational de la couche Resource du CWM (SOLER, VILLARROEL et al. 2006) (SOLER, TRUJILLO et al. 2008) dans l'objectif de construire un modèle en étoile qui représente les règles de sécurité et d'audit capturées durant la phase de modélisation conceptuelle de l'entrepôt de données. L'extension proposée a permis de définir des nouvelles classes et contraintes de sécurité et d'audit pour chaque élément du modèle relationnel qui permettent de représenter tous les besoins d'audit et de sécurité capturés durant la phase de modélisation concep-

tuelle de l'entrepôt. En se basant sur ceci, (Blanco et al. 2015)ont développé une architecture MDA automatique pour sécuriser l'entrepôt de données et définir des règles de passage formelles entre le modèle conceptuel de l'entrepôt de données et le modèle logique. Dans cette approche, le paquet OLAP de la couche Analysis a été étendu par des aspects de sécurité dans le but de définir les contraintes de sécurités dans la couche de métadonnées qui connecte le répertoire des DWs avec les outils OLAP. L'extension a été nommée SECMDDW (Secure Multidimensional Data Warehouse) et présente une proposition pour développer les applications OLAP sécurisées : elle permet de spécifier à la fois les aspects structurels et de sécurité pour des applications OLAP, principalement pour les cubes et les dimensions. Par la suite, un travail plus récent de El OUAZZANI (El Ouazzani et al. 2016)propose une approche pour remédier aux problèmes de confidentialité des données dans l'entrepôt de données. Le métamodèle proposé étend le paquet relationnel du CWM avec des nouvelles classes pour définir les permissions des utilisateurs. L'extension propose une solution basée sur le profil de l'utilisateur qui consiste à définir les autorisations d'accès en fonction du rôle de l'utilisateur en utilisant les droits d'accès définis dans les sources, générer le niveau de sensibilité de chaque objet dans le DW selon ces autorisations, tracer l'accès et détecter les tentatives de violation des droits d'accès sur une donnée sensible (données avec un niveau de sensibilité élevé). L'objectif de cette solution est de réduire la vulnérabilité des données dans un DW et d'aider le propriétaire du DW à bien gérer le contrôle d'accès des utilisateurs.

Par ailleurs, beaucoup de recherches portent sur l'utilisation de DW avec les Systèmes d'information géographiques (SIG), mais il manque encore de consensus sur la conception des schémas dimensionnels spatiaux des entrepôts de données géographiques, Geographic Data Warehouse (GDW), et de métamodèles pour fournir une intégration sémantique parmi les métadonnées de ces technologies. Visant à adresser cela, (Silva et al. 2010) ont présenté le métamodèle GDW en se basant sur l'extension du paquet Relational du CWM. Le métamo-

dèle GDW a pour but de faciliter sa réutilisation et son extension dans d'autres études en vue de rendre la spécification des schémas des entrepôts de données géographiques facile pour permettre de normaliser les données géométriques et d'améliorer le temps de réponse des requêtes et les besoins de stockage dans le GDW. De plus, (NASCIMENTO FIDALGO et al. 2010)ont proposé le GeoDWFrame en étendant le paquet OLAP du CWM afin de guider la conception des schémas dimensionnels géographiques. L'extension vise à minimiser la redondance des données géographiques et, en outre, à fournir un support naturel aux services OLAP et AGIS (Analytical Geographical Information Service).

De plus, en ce qui concerne l'aspect multidimensionnel du CWM, plusieurs travaux ont souligné que le métamodèle multidimensionnel du CWM représente les aspects multidimensionnels d'une façon générale et ne prend pas en compte tous les objets d'une base de données multidimensionnelle. De plus, selon Medina (MEDINA et TRUJILLO 2002) le métamodèle CWM est assez générique pour représenter toutes les particularités de la modélisation multidimensionnelle à un niveau conceptuel et aussi très complexe pour être manoeuvré par les utilisateurs finaux et les concepteurs. Dans ce sens, l'approche de MIDOUNI (MIDOUNI, DARMONT et BENTAYEB 2009) a étendu le métamodèle multidimensionnel du CWM par de nouvelles classes de façon à prendre en compte, dans un seul métamodèle, tous les composants d'une base de données multidimensionnelle. Le métamodèle proposé vise à apporter des solutions au problème de la modélisation multidimensionnelle de données complexes, en l'occurrence les données médicales du projet MAP (Médecine d'Anticipation Personnalisée). L'objectif est de proposer un modèle multidimensionnel pour les données biomédicales et de généraliser ce modèle vers un métamodèle pour entrepôts de données médicales. Aussi, le DWEB (Data Warehouse Engineering Benchmark) a été proposé par (DARMONT 2007) en étendant le CWM dans le but de pouvoir modéliser les grands types de schémas multidimensionnels qui sont populaires dans les environnements ROLAP (Relational OLAP), à savoir les schémas en étoile, en flocon de neige (avec

des dimensions hiérarchiques) et en constellation (avec des tables de faits multiples et des dimensions partagées). Le DWEB est plutôt destiné aux concepteurs et aux chercheurs, car il permet en premier lieu d'évaluer l'impact de différents choix architecturaux ou de techniques d'optimisation sur les performances d'un système donné. Egalement, Prat a cité que le métamodèle OLAP du CWM n'est pas reconnu actuellement comme étant le métamodèle multidimensionnel standard par la communauté de recherche (PRAT, AKOKA et COMYN-WATTIAU 2006), du fait qu'il manque de concepts importants qui se trouvent dans d'autres métamodèles multidimensionnels. Aussi, il a mentionné que le métamodèle OLAP intègre également certains concepts de modélisation périphérique à la modélisation multidimensionnelle et / ou qui sont liés à l'implémentation, c.à.d. qui appartiennent au niveau physique. Pour cela, afin de permettre l'interaction de l'utilisateur et la visualisation des schémas multidimensionnels au moyen d'une notation graphique, l'approche de Prat soutient que les concepts logiques et physiques doivent rester séparés et que le nombre de concepts multidimensionnels doit être relativement limité. L'approche propose un métamodèle multidimensionnel unifié qui partage de nombreux concepts avec le métamodèle OLAP du CWM.

De même, l'approche de KOZMINA (KOZMINA et SOLODOVNIKOVA 2011) a étendu le paquet OLAP du CWM par les classes « Acceptable Aggregation », pour enregistrer les informations sur les fonctions d'agrégation (SUM, AVG, COUNT, MIN, MAX). Par la suite, SOLODOVNIKOVA (SOLODOVNIKOVA, NIEDRITE et KOZMINA 2015) a proposé récemment une approche pour propager les exigences des DW modifiées dans les schémas d'entrepôt de données. L'approche prend en charge les versions des schémas du DW et emploie le métamodèle des exigences formalisées et le métamodèle des multiversions de l'entrepôt pour identifier les changements nécessaires dans un entrepôt de données. Dans cette approche, un Framework pour l'évolution du DW a été proposé. Le Framework supporte trois types de changement du schéma du DW différenciés selon leurs

opérations de changement : physique, logique et sémantique. Les changements physiques (c.à.d. l'ajout d'un nouvel attribut à une dimension) fonctionnent avec des objets de base de données et une instance du modèle de niveau physique de l'entrepôt de données. Les changements logiques (c.à.d., la connexion d'une dimension à une table de faits) modifient principalement une instance du modèle de niveau logique du DW. Tandis que les changements sémantiques (c.à.d., la modification de la signification d'un attribut) peuvent ajuster une instance du modèle de niveau sémantique du DW, ainsi qu'une instance du modèle de niveau logique. Le niveau physique des métadonnées est basé sur le paquet Relational du CWM, alors que le niveau logique est basé sur le paquet OLAP étendu par deux éléments SchemaVersion et VersionTransformation pour représenter les versions multiples du schéma du data warehouse.

Quant à THAVORNIUN, il est bien connu que le CWM est principalement axé sur les métadonnées relatives à l'entrepôt de données. L'objectif de son approche (Thavornun 2015) est de représenter d'autres métadonnées que le CWM ne couvre pas (tel que : par exemple, les spécifications de l'algorithme d'extraction, les caractéristiques et les données des données dictionnaire) pour prendre en charge les systèmes automatisés de découverte de connaissances. Pour cela, un outil de gestion de métadonnées pour les systèmes de découverte de connaissances a été proposé en étendant le métamodèle CWM. En ce qui concerne l'extension du CWM en utilisant le mécanisme des TaggedValues and Stéréotypes ; (M. Tavac et V. Tavac 2013) ont défini un nouveau modèle pour la construction des modèles de transformation MDA en se basant sur l'extension des paquet Core et Relational de la couche Object Model et Resources du CWM.

Le tableau suivant représente un récapitulatif des différents travaux qui ont étendu le CWM présentes dans la littérature. Nous exposons la problématique, l'objectif de l'extension, le couche et les métamodèles du CWM qui ont été étendus, et les nouveaux éléments ajoutés.

TABLE 2.1 – Synthèse des travaux qui ont étendu le métamodèle CWM

Travaux	Problématique	L'objectif de l'extension du CWM	Paquets de CWM étendus	Nouveaux éléments ajoutés
ZHAO et HUANG 2006	CWM manque de la sémantique précise pour représenter les capacités de version	- Formaliser le métamodèle CWM en termes d'une logique formelle dans le but de détecter les incohérences. - Étendre le CWM pour supporter l'enregistrement de l'évolution de l'information afin de maintenir les cohérences durant l'évolution des métadonnées.	La couche Object Model : - Le paquet Core	Les classes : - VersionedModel - CompositeModel - PrimitiveModel - Trace - HorizontalTrace - EvolutionTrace
GOMES, FARINHA et TRIGUEIROS 2007	Le métamodèle CWM ne fournit aucun support pour les problèmes de qualité de données	- Proposer un métamodèle pour la qualité et le nettoyage des données pour les contextes opérationnels et d'entreposage de données, spécialement pour ETL, par extension du CWM.	La couche Fondation : - Les paquet DataType, Expression, Business Information, et Software Deployement La couche ObjectModel : - Le paquet Core La couche Analysis : - Les paquets Business Nomenclature et Transformation La couche Mangement : - Le pquet Warehouse Operation.	Les classes : - Dictionary - Enumeration Rules - Data Quality Rules - Data Cleaning Execution - Pending DC Execution - ...
OLBRICH 2010	Les modèles traditionnels des métadonnées et l'entrepôt de données ne fournissent pas un stockage de qualité de données efficace.	- Étendre le métamodèle CWM afin de stocker et d'analyser la masse d'informations de qualité dans le but de gérer la qualité des données pour les données de diffusion.	La couche ObjectModel - Le paquet Core - Le paquet intance La couche Resource : - Le paquet Relational	Les classes : - Aggragaor - Dispatcher - BlackBox

DI TRIA, LEFONS et TANGORRA 2012	- Absence de métadonnées pour les systèmes de réponse à requêtes approximatives dans le CWM - Le CWM ne fournit pas le support pour le lignage de métadonnés dans l'entrepôt	- Définir un nouveau métamodèle par extension CWM pour représenter les métadonnées standard pour les systèmes de réponse à requêtes approximatives en vue de réduire les données stockées dans l'entrepôt de données	La couche Resource : - Le paquet Relational	Les classes : - MiningETL ArrayStreamDataProvider MiningETLArray-StreamInformer MiningETLArrayStream
BATASOVA et al. 2015	Le processus de données présenté par le CWM ne supporte pas la préparation de données sans un processus additionnel	Fournir une approche qui combine les capacités des outils existants pour le traitement des données et les transformations requises pour les algorithmes du data mining	La couche Analysis : - Le paquet Transformation - Le paquet Datamining	Les classes : MiningETL ArrayStreamDataProvider MiningETLArray- StreamInformer MiningETLArrayStream
THESS et BOLOTNICOV 2004	le standard CWM est très complexe et nécessite la connaissance d'autres standards de l'OMG	Construire une bibliothèque universelle de datamining pour implémenter les méthodes et les algorithmes du datamining	La couche Analysis : - Le paquet Datamining	Les classes : - Model - Setting - Attribute - DataAccess - Algorithms - Transformation - Automation
SOLER, TRUJILLO et al. 2008	Le CWM ne permet pas de modéliser et de représenter les aspects de sécurité des données et des mesures d'audit	- Etendre le CWM pour supporter les règles de sécurité et d'audit capturées pendant la phase de modélisation conceptuelle de DW dans le but d'améliorer le support de sécurité et faciliter l'établissement d'un mécanisme de contrôle d'accès normalisé pour l'entrepôt de données.	La couche Analysis : - Le paquet Relational	Les classes : - SSchema - Scatalog - Stable - Scolumn - SecurityProperty - AuditConstraint - ARConstraint - AURConstraint - SecurityConstraint - UserProfile

Blanco et al. 2015	Absence de la représentation des données de sécurité et des mesures d'audit dans le CWM	- Développer une architecture MDA complète pour le développement des applications OLAP sécurisée par extension du CWM.	La couche Analysis : - Le paquet OLAP	Les classes : -Role - CubePermission - MemberPermission SecurityPermission DimensionPermission
El Ouazzani et al. 2016		- Extension du CWM pour remédier aux problèmes de confidentialité des données dans l'entrepôt de données et définir les permissions des utilisateurs.	La couche Analysis : - Le paquet Relational	
Midouni, Darmont et Bentayeb 2009	Le CWM ne fournit les fonctionnalités pour représenter les données de systèmes d'informations géographique dans l'entrepôt de données	- Proposer un métamodèle pour aider à la spécification des schémas d'entrpôt de données géographiques par extension du CWM.	La couche Analysis : - Le paquet Relational	Les classes : Fact table - Dimension table TAttribute - Degenerated Common Spatial
Darmont 2007	Peu de bancs d'essai sont présents pour le contexte des entrepôts de données et de l'analyse en ligne OLAP	- Concevoir un DWEB, pour générer des entrepôts de données synthétiques ad hoc, les charges de requêtes d'alimentation et d'analyse associées pour mesurer le temps de réponse du système.	La couche Analysis : - Le paquet OLAP La couche Resource : - Le paquet Relational	Les classes : - DataWarehouse - Tuple - Fact Table
Kozmina et Solodovnikova 2011	Le métamodèle CWM ne permet pas de représenter les préférences des utilisateurs au niveau logique du DW	- Proposer une approche pour fournir aux utilisateurs les recommandations sur les rapports d'entreposage de données.	La couche Analysis : - Le paquet OLAP	Les classes : - AcceptableAggregation

SOLODOVNIKOVA, NIEDRITE et KOZMINA 2015		- Proposer un Framework pour propager les exigences d'entrepôt de données présentes dans les schémas d'entrepôt de données.	La couche Analysis : - Le paquet OLAP	Les classes : - SchemaVersion - VersionTransformation
THAVORNUN 2015	Le CWM supporte uniquement les métadonnées dédiées pour l'entreposage de données	- Présenter un outil de gestion de métadonnées pour les systèmes de découverte de connaissances.	La couche ObjectModel : - Le paquet Core La couche Analysis : - Le paquet Transformation - Le paquet Datamining	Les classes : - Evidence - UserAction - DataProperty
M. TAVAC et V. TAVAC 2013	–	- Construire un modèle pour les modèles de transformation MDA.	La couche ObjectModel : - Le paquet Core La couche Resource : - Le paquet Relational	Les stereotypes : - MovedCols - Association - Inheritance - Renamed

2.4 SYNTHÈSE

Dans ce chapitre, nous avons fourni une présentation détaillée du CWM, comme étant le standard destiné à l'intégration et l'interopérabilité des outils d'entreposage de données et du business analyse, basé sur l'échange et le partage des métadonnées. Le métamodèle CWM offre des fonctionnalités et une architecture de paquetage complète pour représenter le domaine d'entreposage de données et de business analyse.

Cependant, les constructions et la sémantique du CWM manquent de plusieurs aspects de data warehousing et du BI. C'est dans ce sens que le métamodèle CWM fournit trois mécanismes d'extension pour définir de nouveaux éléments, à savoir : l'Héritage, les Tagged Values et Stéréotypes, et les extensions à base d'XMI. Le choix de la technique d'extension pour étendre le métamodèle dépend de l'objectif à atteindre et de l'environnement dans lequel cette extension s'effectuera.

Par la suite, nous avons présenté les différents domaines dont l'extension du CWM a été considérée comme utile pour répondre à différents problèmes. D'où, nous avons constaté que l'aspect capitalisation des connaissances et réutilisations n'a pas été abordé dans le standard CWM. Ces derniers seront intégré dans CWM via la notion point de vue. L'objectif du chapitre suivant est de remonter aux origines de la notion point de vue en présentant les différents travaux qui l'ont implémenté.

3

REPRÉSENTATION MULTIPOINTS DE VUE DE CONNAISSANCES

3.1 INTRODUCTION

Représenter consiste à faire une abstraction d'une réalité riche en information. Cette abstraction dépend de l'utilisateur ou l'agent qui la réalise, de son domaine d'intérêt et de ses connaissances préalables. Ainsi, lorsque plusieurs utilisateurs (observateurs, agents...) travaillent sur un même univers de connaissance, ils observent des éléments et relations différents. La plupart des systèmes informatique négligent cette variété de perceptions et ils offrent des outils pour créer un modèle unique du monde, une représentation étroite où chaque utilisateur doit filtrer ce qui l'intéresse, un modèle mono-utilisateur, conçu pour un seul utilisateur du système, une seule vision du monde.

Par opposition à cette approche mono-point de vue, mono-utilisateur (monodisciplinaire) nous allons présenter dans ce travail l'approche multi points de vue d'utilisateur (multidisciplinaire) qui permet de modéliser une même réalité selon des points de vue et des perceptions différentes. La notion de point de vue a souvent été utilisée dans un sens très large, certainement pas limité à son domaine d'application ; On la trouve sous différentes appellations : perspective, contexte, opinion, vue etc. Cette diversité de sens reflète un flou constant dans la définition de ce qui est pris en compte et de ce qui est généré lors de la définition d'un point de vue.

Les concepts de vue et de point de vue ont été étudiés dans plusieurs domaines liés au traitement de l'information : bases de données, représentation des connaissances, analyse et conception, langages de programmation, outils de Génie logiciel, etc. Dans l'approche objet, ces concepts ont été employés sous différents noms qui reflètent leurs variétés d'utilisation, à savoir : vue, point de vue, rôle, perspective, aspect, sujet, morceau, critère, etc. Dans la suite de cette section, nous abordons l'utilisation de la notion de vue/point de vue dans les principaux domaines liés au développement du logiciel : représentation de connaissances, génie logiciel, langages de programmation et langages/méthodes de modélisation. Dans ce chapitre nous proposons un état de l'art sur les systèmes qui ont intégré la notion de point de vue, surtout pour une représentation des connaissances multipoints de vue. Les travaux abordés ne sont pas exhaustifs. En effet, nous allons centrer notre discussion sur la modélisation et la formalisation des points de vue dans un contexte de multi-experts.

3.2 LE POINT DE VUE DANS LA REPRÉSENTATION ORIENTÉE OBJET DE CONNAISSANCES

3.2.1 *Le Point de Vue en représentation de connaissances*

Le concept de point de vue est apparu dès les années 70 dans plusieurs systèmes dédiés à la représentation orientée objets des connaissances. Ce concept s'est présenté sous différents noms, à savoir : vue, point de vue, rôle, perspective, aspect, sujet, morceau, critère, etc. Le modèle de la représentation des connaissances par objets porte sur deux notions principales :

— La **notion de classe** : représente un ensemble d'objets dans le monde réel ayant des caractéristiques communes ; et les classes sont organisées en une hiérarchie avec des relations de subsomption entre elles.

— La **notion d'héritage** : les objets appartenant à une classe (les instances) héritent de toutes les propriétés (attributs) de cette classe ; les sous classes héritent de toutes les caractéristiques des classes ancêtres.

Dans ces modèles la notion de point de vue est généralement appelée perspective. Elle est intégrée sous forme de multi-héritage : un objet (une instance) est associé (hérité) à plusieurs classes selon différents points de vue. Ceci, constitue la principale faiblesse de ces approches, vu les problèmes associés à l'héritage multiple.

Plusieurs systèmes de représentation des connaissances par objets ont été développés en intégrant la notion de point de vue :

Le langage KRL (Knowledge Representation Language) (Daniel G Bobrow et Winograd 1977) développé pour la compréhension du langage naturel, fut le premier système adressé à la représentation multipoints de vue des connaissances. Les points de vue sont représentés au niveau des instances, où chaque instance est liée à sa classe de base et à ses points de vue. L'extension de KRL pour la linguistique a donné OWL II (W. A. Martin 1978), et son extension pour le développement de programme a donné PIE (Personal Information Environment) (Goldstein et Daniel G Bobrow 1980). L'amélioration de KRL, LOOPS (D. Bobrow et MJ Stefik 1982), (Mark Stefik et Daniel G Bobrow 1985) utilise la relation d'agrégation pour implémenter les points de vue ; et ceci en considérant des objets composites dont les composants représentent les différents points de vue.

Ensuite, ROME (Carré 1990) utilise l'héritage multiple pour implanter les points de vue. Le langage est fondé sur la sous-classification, dont une classe contient l'identité d'un objet donné, puis des sous-classes définies pour chaque point de vue à représenter. La notion de point de vue intervient pour exprimer la préférence pour les caractéristiques d'une sous-classification par rapport à une

autre. En ROME, la sélection d'un point de vue permet de lever des conflits entre les caractéristiques (attributs et méthodes) qui portent le même nom, mais qui sont issues de sous-classifications différentes. Ces conflits surgissent tant en héritage multiple qu'en représentation multiple. La résolution des conflits ne s'effectue pas de façon statique, en faisant des choix arbitraires (selon l'ordre des superclasses par exemple), mais se fait de façon dynamique en prenant en compte le contexte induit par le choix d'un point de vue ce qui présente le principal inconvénient de ROME.

L'extension FROME « Frame en Rome » pour le langage de Frames (DEKKER 1994) répond à ce problème en intégrant le contrôle de type statique. Cette extension retient la notion de point de vue du langage ROME pour la description et la manipulation multi-point de vue de concepts. La notion de point de vue est utilisée avantageusement pour résoudre des types de conflits survenant dans les frames. Intuitivement, un point de vue dans FROME correspond à une partie de la hiérarchie des classes, décrivant un ensemble de caractéristiques ayant trait à une partie spécifique du domaine de connaissances représentées. Plus radicalement, chaque classe FROME peut être interprétée comme un point de vue sur les frames qu'elle décrit.

Dans le cadre de la représentation multi-points de vue, les systèmes ROME et FROME distinguent deux relations différentes pour associer un objet à une classe : le lien d'instanciation et le lien de représentation. Le lien d'instanciation relie l'objet à la classe dont il est instance (à la façon des langages de programmation par objets). Le lien de représentation associe l'objet aux classes qui le représentent dans différents points de vue. De plus, il existe une relation d'exclusion mutuelle entre classes qui permet de spécifier que deux classes ne peuvent représenter un même objet, c'est-à-dire qu'elles déterminent deux points de vue incompatibles pour cet objet.

Par la suite, une nouvelle génération des systèmes de représentations multi-points de vue de connaissances est représentée par le modèle TROPES (Mariño 1993) (Ribière 1999) qui rajoute aux entités classiques de la modélisation objets (classe, instance, attributs et facettes) deux autres éléments supplémentaires à savoir : le concept et le point de vue. Chaque concept, correspond à un ensemble d'individus (objets dans le monde réel), et associé à un ou plusieurs points de vue. Les concepts sont disjoints, ils ne partagent pas les mêmes objets. La combinaison de concept et point de vue donne naissance à une hiérarchie de spécialisation de classes (sous forme d'un arbre), qui sont définies selon le point de vue et à propos du concept en question. Une classe ne peut donc spécialiser qu'une seule autre classe (pas de multi héritage). La combinaison de concept et point de vue crée une hiérarchie de spécialisation de classes définies selon le point de vue et à propos du concept en question. Une classe ne peut donc spécialiser qu'une seule autre classe (pas de multi héritage). Un objet (instance) peut instancier plusieurs classes de différents points de vue (multi représentation). Les classes appartenant aux différents points de vue d'un même concept sont reliées par des liens, appelés passerelles. Cela permet d'exprimer des relations ensemblistes entre des ensembles d'instances possibles des classes de points de vue différents, et permet d'effectuer des raisonnements entre des points de vue tels que la vérification de la cohérence (une instance définie dans un point de vue doit satisfaire toutes les contraintes de ce point de vue, et aussi celles définies dans un autre point de vue si ces deux points de vue sont connectés par une passerelle).

La Figure 3.1 montre un exemple de représentation multi point de vue avec FROME, qui peut se comparer avec la même représentation en KRL et en TROPES. Le système ROME et son extension pour le langage de Frames FROME diffèrent de TROPES au niveau de la flexibilité dans le modèle de la représentation des objets. Tandis que dans TROPES, les points de vue sont prédéfinis et fixés pendant la construction du modèle, ROME permet d'ajouter de nouveaux points de vue

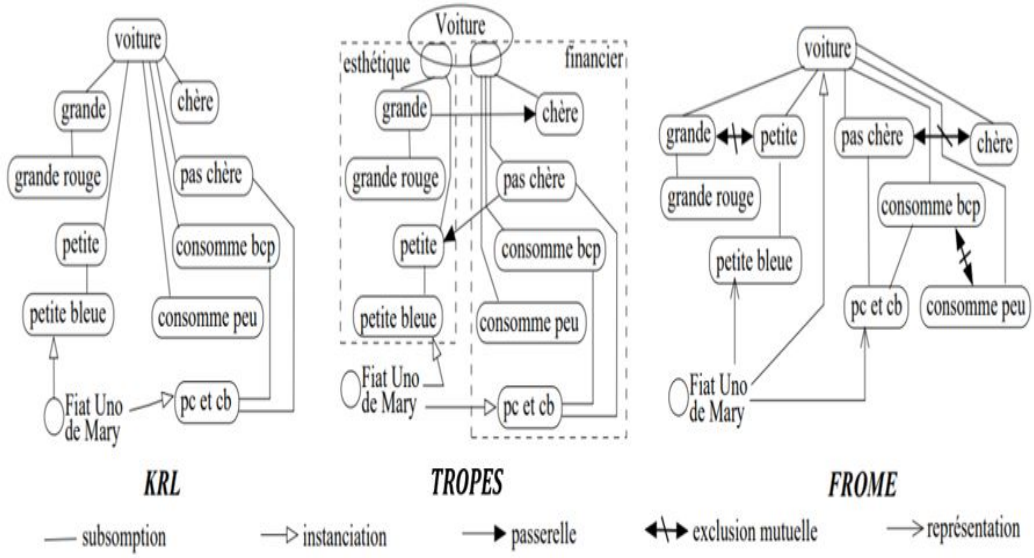

FIGURE 3.1 – Point de vue en Représentation de Connaissances.

et aussi de nouvelles représentations de l'objet pendant son utilisation via des liens de représentation. Cela permet d'avoir un système de représentation d'objets incomplets et évolutifs : les définitions des objets sont précisées, modifiées ou changées lorsqu'ils évoluent dans le temps.

3.2.2 *Le Point de Vue dans la modélisation orientée objet*

La technologie objet a apporté de nombreuses améliorations au domaine du génie logiciel au niveau de la productivité, la fiabilité et la réutilisabilité des logiciels. Cependant, elle a tendance à ne pas intégrer l'acteur et sa vision par rapport au système ou à l'intégrer dans une étape tardive. L'intégration de la notion de

vue et de point de vue dans l'approche objet permet de prendre en compte les acteurs du système dès les premières étapes d'analyse et de conception. De ce fait, elle permet aux différents experts de se focaliser sur leur domaine d'expertise et de projeter leurs systèmes de valeurs sur le système (Nassar 2004). L'introduction de la notion de point de vue dans la modélisation orientée objet consiste en l'élaboration d'un modèle unique partageable accessible suivant plusieurs points de vue. L'intérêt de cette approche apparaît au niveau de la cohérence des données, de la suppression de certaines redondances, de l'enrichissement de l'approche multi-modèles et de la définition des droits d'accès (Marcaillou 1995) :

— **Cohérence** : Au point que l'approche multivues se base sur l'élaboration d'un modèle unique partageable accessible selon plusieurs points de vue, ceci permet de centraliser les données du modèle et permet à l'utilisateur d'un sous-modèle de voir les modifications apportées répercutées dans les autres sous-modèles. En conséquence, les problèmes d'incohérence dus au partage des données entre les modèles partiels sont évités.
— **Enrichissement de l'approche multi-modèles** : La notion de point de vue associée aux utilisateurs permet de voir le modèle selon plusieurs points de vue. Ceci revient à donner des "éclairages" différents sur la même information, ce qui permet à l'utilisateur de se focaliser sur un sous-modèle du modèle global spécifique à son besoin. Donc la notion de point de vue induit indirectement un enrichissement de l'approche multi-modèles.
— **Droits d'accès** : Les utilisateurs d'un système n'ont pas toujours les mêmes besoins et les mêmes responsabilités, ce qui met en évidence la nécessité de définir des droits d'accès aux informations du modèle. L'intégration de l'approche multivues dans la modélisation par objet permet de prendre en considération cette exigence. En effet, les points de vue se traduisent par la définition de droits d'accès sur le modèle. Ils permettent donc de masquer des vues à certains utilisateurs.

— **Centralisation de la connaissance** : Dans un modèle, certaines connaissances doivent être partagées entre des utilisateurs différents. Pour assurer ce partage, l'approche multi-modèles procède généralement par une duplication de ces connaissances dans les différents sous-modèles. Cependant, cette répartition des informations implique la nécessité de prendre en charge la mise à jour de ces informations pour garder leur cohérence. Afin de surmonter ces problèmes, l'approche multivues utilise un modèle unique accessible selon plusieurs points de vue.
— **Evolutivité** : Un intérêt important de l'approche multivues est la possibilité de masquer puis de retrouver des points de vue à des instants donnés. Ceci veut dire qu'il faut conserver les modèles d'analyse tout au long du cycle de vie d'un système mais sans qu'ils soient actifs en permanence. Donc les points de vue permettent de répondre au besoin d'évolutivité temporelle. Ils offrent aussi la possibilité de faire évoluer les droits d'accès des utilisateurs du modèle.

- **VBOOM-VBOOL-VUML-VxUML**

Dans l'objectif de définir et de mettre en oeuvre une approche de développement orientée points de vue. Le projet VBOOM (View Based Object Oriented Methodology) a donné lieu à plusieurs travaux de recherche qui ont visé l'introduction du concept de vue et point de vue dans le cadre d'une modélisation par objets. L'objectif principal était d'apporter une plus grande souplesse, flexibilité et rigueur dans la modélisation à objets dans un contexte de Génie Logiciel. Pour cela, l'extension VBOOL (Marcaillou 1995) du langage à objet Eiffel (Meyer 1992) avec intégration du point de vue a été définie. L'implantation de la notion de point de vue dans VBOOL est réalisée par une nouvelle relation appelée visibilité. Cette relation est proche de l'héritage, elle permet de sélectionner les informations contenues dans une classe et de les filtrer vers les classes dérivées (Marcaillou 1995). Figure 3.2 illustre les concepts clés de VBOOL :

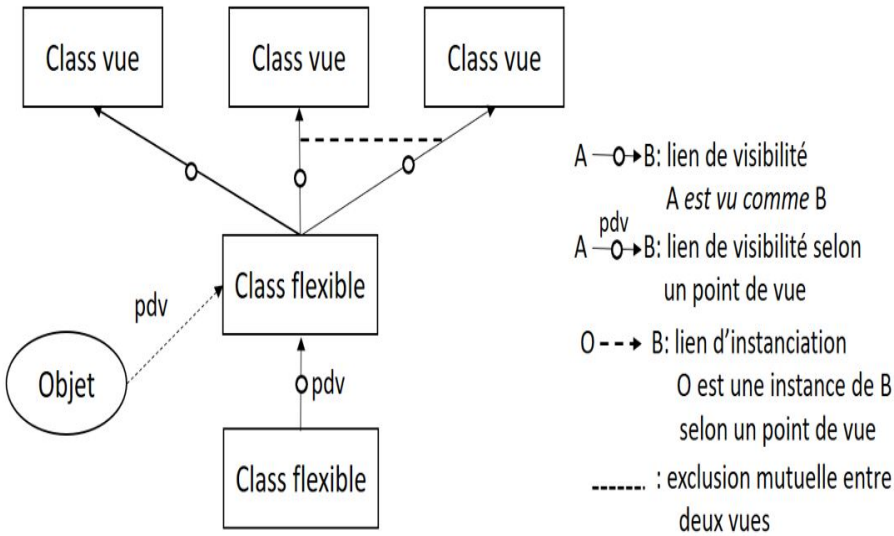

FIGURE 3.2 – Les concepts clés de VBOOL.

— Une *vue* représente une abstraction partielle du modèle, correspondant donc à un sous-modèle.
— Un *point de vue* représente la vision qu'a un utilisateur du modèle et correspond à la combinaison de plusieurs vues, éventuellement réduite à une seule.
— Une *classe flexible* est une classe dans laquelle est déclaré un ensemble de vues constitué d'au moins deux éléments. N'importe quelle classe est susceptible de devenir une vue puisqu'une classe quelconque X peut établir une relation de visibilité vers une classe Y qui devient alors une vue de X. Le principe de base de l'utilisation de la visibilité est lié à sa sémantique

exprimée par ‘est vu comme’. Cette relation est très proche de la relation d’héritage dont la sémantique est "est un".

La relation de visibilité dans VBOOL permet une souplesse et une flexibilité lors de la définition des points de vue en exposant à chaque utilisateur les informations pertinentes pour son activité sous la forme de vue ; contrairement au modèle TROPES où les points de vue sont fixes et prédéfinis. Par contre, l’expérience a permis de mettre en évidence plusieurs limites de VBOOM ou de sa mise en oeuvre : subjectivité dans la définition des vues, rigidité de l’héritage multiple pour prendre en compte l’évolution d’un modèle à bases de vues, complexité d’une programmation avec le langage dédié VBOOL, manque d’outils supports performants, etc. (NASSAR 2004). Ceci a engendré une grande réflexion qui a mené à une nouvelle approche d’analyse et de conception multi -vues appelée VUML (View based Unified Modeling Language) par reconsidération de l’approche développée dans VBOOM et en conservant l’objectif d’une modélisation multivues à granularité fine. VUML a pour objectif d’offrir un formalisme et une démarche pour mener une modélisation objet par points de vue de l’analyse jusqu’à la génération de code. Par rapport à l’approche initiale adoptée dans VBOOM, VUML conserve l’objectif d’une modélisation multivue à granularité fine.

La singularité de VUML réside dans le fait d’associer de façon unique une vue à chaque type d’acteur (utilisateur) du système. L’implantation et la gestion de l’évolution des vues sont réalisées par l’intermédiaire de la délégation (au lieu de l’héritage multiple mis en oeuvre dans VBOOL). Dans VUML, l’ajout principal à UML est le concept de classe multivue qui est composée d’une base (partie partagée par tous les points de vue) et d’un ensemble de vues (extensions de la base). VUML (View based UML) offre des mécanismes pour gérer les droits d’accès aux caractéristiques des classes multivue, spécialiser une classe multivue, spécifier les dépendances entre les vues, assurer la cohérence du modèle en cas de mises à jour, administrer les vues à l’exécution, etc. A l’instar d’UML, la séman-

tique de VUML est décrite par un métamodèle, des règles de bonne formation exprimées en OCL (O. OMG 2005), et des descriptions textuelles informelles. Les définitions informelles des concepts clés de VUML sont :

— Un *acteur* : représente une entité logique ou physique qui interagit avec le système.
— Un *point de vue* : représente la vision d'un acteur sur le système (ou sur une partie de ce système). En d'autres termes, il exprime une perspective selon laquelle un système peut être modélisé pour décrire les besoins de l'acteur considéré. Un point de vue unique est associé à un acteur.
— Une *vue* : entité de modélisation (statique). Elle correspond à l'application d'un point de vue sur une entité donnée (classe, et par généralisation le système entier). Par simplification de langage, nous dirons qu'une vue est associée à un acteur en considérant comme implicite l'entité sur laquelle le point de vue de l'acteur s'applique.
— Une *classe multivue* est une unité d'abstraction et d'encapsulation composée d'une base (partie commune accessible par tous les acteurs de la classe multivue) et d'un ensemble de vues représentant les besoins et les droits d'accès des acteurs. Chaque vue correspond à un seul acteur. A l'exécution, un objet multivue est une instance d'une classe multivue. Une seule de ses vues est active, et correspond à un point de vue qui est le point de vue de l'utilisateur courant du système.

VUML offre un formalisme pour modéliser un système logiciel par une approche combinant objets et points de vue. Sur le plan méthodologique, VUML propose une démarche centrée utilisateur qui permet d'intégrer la notion de point de vue dans le processus de développement des systèmes. Dans cette approche, plusieurs modèles de conception modélisant les besoins de chaque acteur interagissant avec le système sont développés par différents concepteurs. Ces modèles doivent être ensuite composés (fusionnés) pour produire un modèle VUML

unique représentant le modèle de conception global du système. L'ajout d'un point de vue (par exemple pour prendre en compte un nouvel acteur) peut nécessiter le développement d'un modèle de conception supplémentaire qui devra être composé à son tour avec le modèle VUML existant.

La Figure 3.3 ci-dessous illustre la structure statique d'une classe multivue. Une telle classe est composée d'une base (stéréotypée « base ») qui a le même nom que la classe UML correspondante, et des vues (stéréotypées « view ») reliées à la base par une relation nommée viewExtension. La relation viewExtension est une relation de dépendance (et non pas une relation d'héritage) au sens où les valeurs des attributs de la base sont implicitement partagées par les vues ; en outre, une caractéristique (attribut, méthode) de vue peut redéfinir une caractéristique de la base.

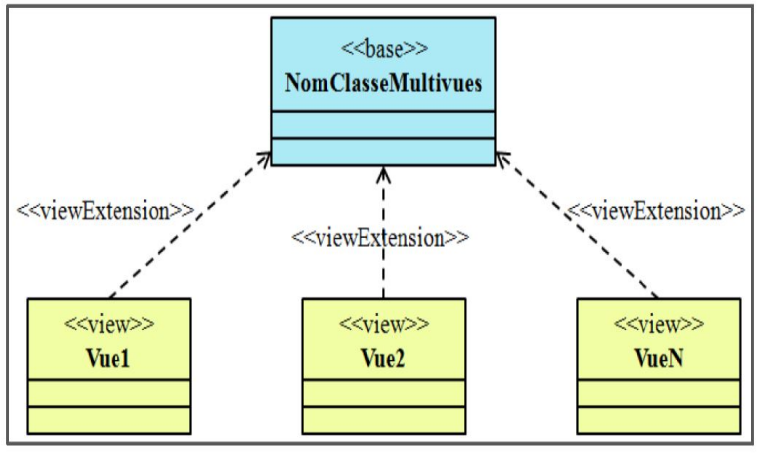

FIGURE 3.3 – Structure statique d'une classe multivue.

Les vues d'une classe multivue peuvent naturellement être dépendantes les unes des autres, et sont reliées par une relation de dépendance stéréotypée par « viewDependency ». La dépendance « viewDependency » entre les vues vue 1 et vue 2 indique que des données de la vue vue1 (source de la dépendance) dépendent de certaines données de la vue vue2 (cible de la dépendance). La Figure 3.4 présente un exemple du concept VUML.

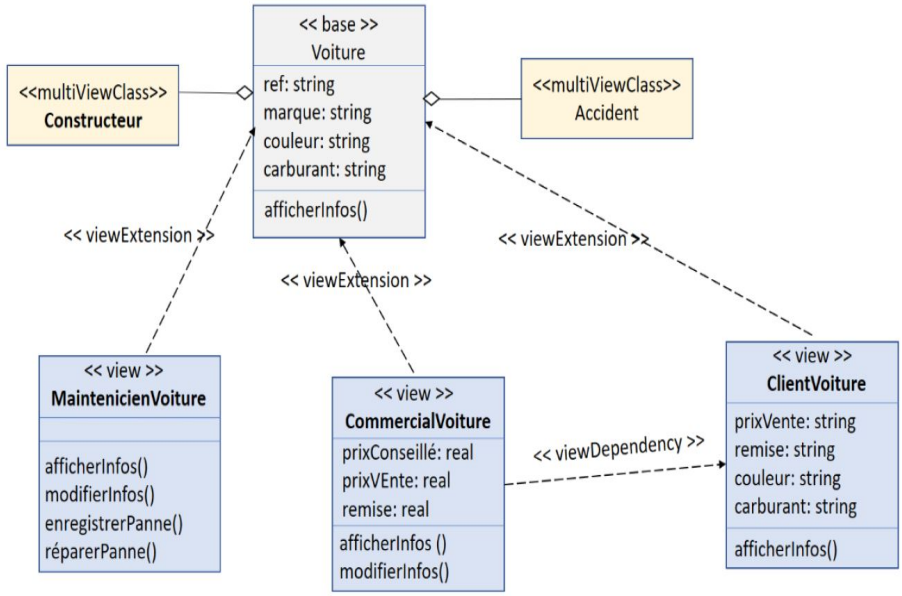

FIGURE 3.4 – Les concepts de la classe multivue VUML.

A ce jour, en proposant la notion de classe multivue, VUML traite les aspects structuraux liés à la composition des vues et au partage des données statiques sans prendre en compte la manière dont ces vues vont réagir, ni comment les synchroniser afin de représenter le comportement des objets multivue (instances

d'une classe multivue) (ANWAR 2009). L'approche de VxUML (LAKHRISSI 2010) cherche à combler ce manque en dotant le profil VUML de nouveaux mécanismes permettant d'exprimer le comportement d'un système. Il s'est concentré pour cela sur le comportement des objets multivue décrit par des machines à états qui nécessitent des adaptations des concepts de modélisation UML. L'approche a introduit la notion de sonde d'événements pour spécifier des communications implicites entre les objets-vue à travers des observations d'événements. Ceci permet de découpler des spécifications qui sont a priori fortement interconnectées, de les concevoir séparément, puis de les intégrer sans avoir à les modifier.

3.3 LE POINT DE VUE DANS LES LANGAGES DE PROGRAMMATION

La notion de point de vue a été intégrée dans plusieurs langages à objets en utilisant les relations classiques du paradigme objet, notamment : l'héritage, la délégation et la composition. Aussi, plusieurs travaux ont tenté d'intégrer les points de vue dans les langages à prototypes (BARDOU 1998a). En second lieu, le concept du point de vue est apparu dans les langages de programmation, à savoir : la programmation par sujet (HARRISON et OSSHER 1993), programmation par aspect (KICZALES et al. 1997), programmation par vues (MILI et al. 1999), Programmation par objets structurés en contextes CROME (DEBRAUWER 1998) (VANWORMHOUDT 1999)et la programmation par points de vue en langages à prototypes.

3.3.1 *Le Point de Vue dans la programmation par sujet*

La programmation par sujets (Subject Oriented Programming ou SOP) est une nouvelle technique de séparation des préoccupations (métier, technologiques, règles de gestion, etc.) qui s'introduit par le concept de subjectivité qui permet d'identifier un ensemble de spécifications et de comportements reflétant la per-

ception du monde réel qui correspond à une vision générique d'un acteur (Harrison et Ossher 1993). Le point de vue dans la programmation par sujet porte le nom de « sujet », dont chaque sujet représente une préoccupation particulière. Un sujet représente une collection d'états et de comportements d'un groupe d'objets/classes ou fragments de classes liés entre eux grâce à l'héritage ou aux autres relations Figure 3.5. Ces objets reflètent une perception du monde réel. Dans ce contexte, la composition (connue aussi sous le nom d'intégration) de l'ensemble des sujets considérés produit le système final.

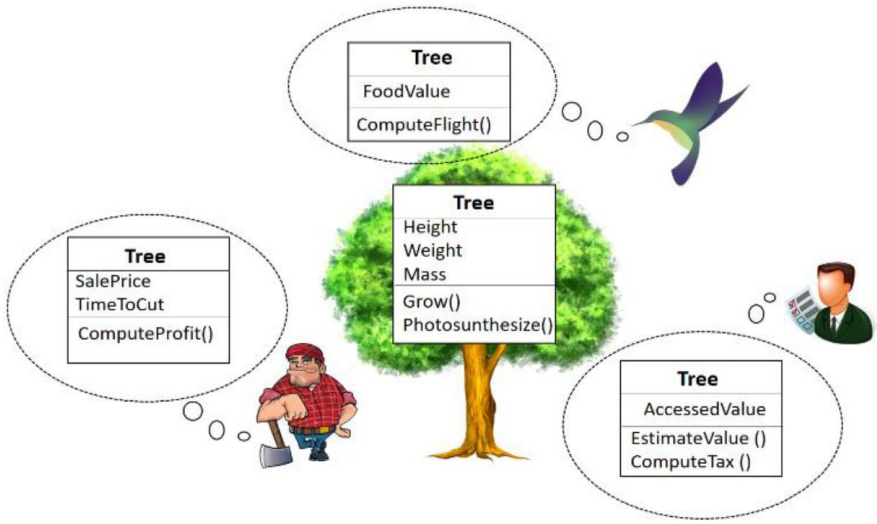

Figure 3.5 – Un arbre orienté-objet selon plusieurs vue subjective (point de vue). (Harrison et Ossher 1993)

Pour effectuer l'intégration de l'ensemble des points de vue (sujets) trois types de règles de composition sont utilisés (Ossher et al. 1996) : les règles de mise en correspondance, les règles de combinaison, et les règles de combinaison et de mise en correspondance en même temps. Chaque règle de composition concerne

plusieurs points de vue destinés à être intégrés. Enfin, la définition des règles de composition est encapsulée dans un ou plusieurs modules de composition. Ce paradigme offre un certain nombre d'avantages dans la conception et l'implémentation de systèmes complexes (Hachani 2006). Cependant les objets partagés par les développeurs peuvent avoir des définitions relatives aux vues subjectives des développeurs ; Un unique sujet cohérent définit une seule préoccupation ; Les fonctionnalités d'un système peuvent être étendues par le biais d'un nouveau sujet encapsulant l'évolution attendue (modifications ou extensions), à condition de préciser les règles de composition (Amroune 2014).

3.3.2 *Le Point de Vue dans la Programmation par aspects*

La Programmation Orientée Aspect (POA) (Kiczales et al. 1997)permet d'encapsuler les exigences non fonctionnelles d'un système sous forme d'aspects. Ces exigences non fonctionnelles représentent les qualités ou les contraintes imposées pour mieux satisfaire les besoins fonctionnels d'un système, par exemple : aspects de performance, de sécurité, de persistance, de journalisation, d'authentification, etc. Le point de vue dans la POA prend le nom d'aspect et représente une entité logicielle qui capture une préoccupation transversale d'un système. Par rapport à l'orienté objet, l'aspect permet aux programmeurs d'encapsuler des comportements qui affectaient de multiples classes dans des modules réutilisables (Figure 3.6).

La POA permet donc d'encapsuler dans un module les préoccupations qui se recoupent avec d'autres. Le principe est de coder chaque problématique séparément et de définir leurs règles d'intégration pour les combiner en vue de former un système final. Le point de vue dans la POA permet une meilleure réutilisation, moins de couplage entre les modules, ainsi qu'une meilleure séparation entre les préoccupations d'une manière générale.

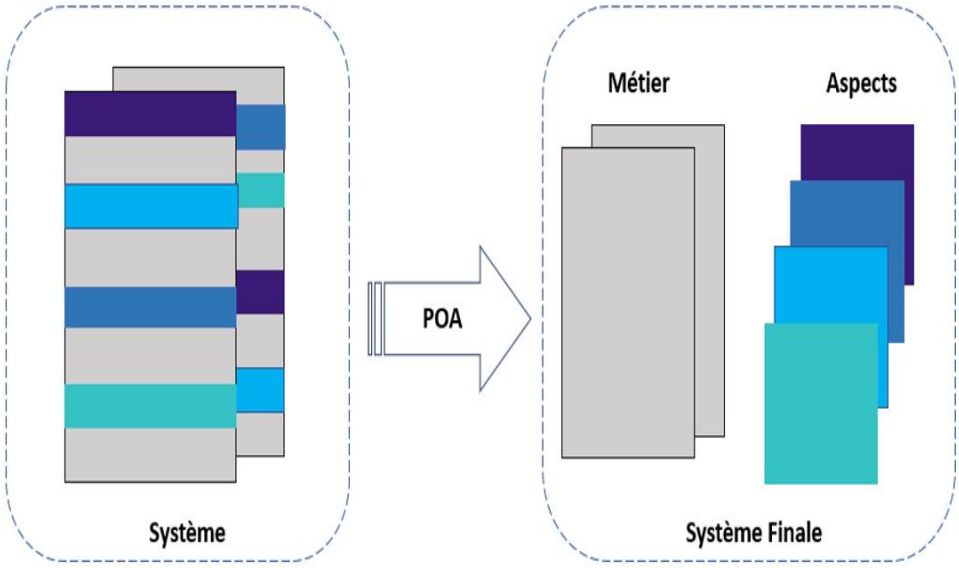

FIGURE 3.6 – Les concepts de la Programmation par Aspects.

3.3.3 *La programmation par rôles ou par points de vue*

La programmation par rôles ou par point de vue (BARDOU 1998b)permet de déstructurer la notion d'objet par l'introduction de vues subjectives : les rôles qu'ils jouent ou les points de vue qu'ils représentent. Dans cette approche, les programmes sont structurés en groupes d'objets. Ces groupes représentent les différents points de vue du système à développer. Ce qui distingue cette approche de l'approche objet, est qu'une entité du monde réel, dans cette approche, est représentée par un ou plusieurs objets. Selon le point de vue considéré, une entité peut jouer plusieurs rôles. Chaque objet modélise un rôle particulier que peut jouer une entité. L'ensemble des préoccupations de l'application représentent les diffé-

rentes vues subjectives du système. Chaque préoccupation est représentée par les différents rôles et leurs interactions. La programmation par rôles ou par points de vue présente aussi certaines limites liées à la séparation des préoccupations à cause des problèmes de partage de données et de recouvrement de diverses définitions entre préoccupations distinctes. En plus, il est à signaler, la difficulté de mise en oeuvre de l'intégration de propriétés d'objets représentant une même entité du monde réel mais appartenant à des groupes d'objets différents. sectionLe Point de Vue dans l'Ingénierie des Exigences Le principal but de l'Ingénierie d'Exigence (IE) est l'obtention d'un ensemble complet et cohérent d'exigences. Dans ce domaine, les vues et les points de vue représentent les compétences, les objectifs et les rôles de chaque participant dans le processus d'IE. Dans ce contexte, l'approche multipoints de vue permet une meilleure élicitation collective des exigences (Nuseibeh, Kramer et Finkelstein 1994), (Sommerville et Pete Sawyer 1997). Le principal objectif du point de vue dans ces approches est de guider la découverte, l'analyse et la gestion des exigences. Nous présentons cidessous l'exemple de la méthode PREVIEW l'une des méthodes les plus connues utilisant la notion de point de vue dans l'ingénierie d'exigence.

- **La méthode PREVIEW**

La méthode Preview (Process and Requirements Engineering VIEWpoint) (Sommerville, Peter Sawyer et Viller 1998)a été développée dans le cadre du projet REAIMS (Requirements Engineering Adaptation and Improvement for Safety and dependability). Son objectif était d'améliorer le processus d'élicitation au sein de quatre partenaires industriels. Le point de vue (PV) dans cette méthode est défini comme une entité d'encapsulation de quelques informations concernant les exigences d'un système futur. Ces informations peuvent être découvertes à travers un processus d'analyse, à partir des discussions avec les parties prenantes, ou à partir des connaissances organisationnelles et du domaine. Un point de vue dans la méthode PREview est représenté par une structure à six composants :

— Le *nom* : identifie le PV
— Le *centre d'intérêt du PV* : appelé aussi Focus, définit le cadre de définition du PV.
— Les *préoccupations du PV* : reflétent les buts organisationnels et les contraintes qui guident le processus d'analyse.
— Les *sources des PVs* : sont des identifications explicites des sources d'information associées aux points de vue. Elles définissent explicitement les sources des exigences liées à ce point de vue. Elles peuvent être des individus, des rôles, des groupes ou encore des documents.
— Les *exigences du PV* : sont un ensemble de besoins résultant du processus d'analyse du système en se basant sur le centre d'intérêt du PV. Les exigences peuvent être exprimées en termes de fonctions système, les nécessités des utilisateurs ou des contraintes résultantes du domaine de l'application ou à partir des considérations organisationnelles.
— L'*historique du PV* : enregistre les changements du PV comme une aide pour la traçabilité. Il inclut les changements du centre d'intérêt, des sources et des besoins encapsulés dans le PV.

La notion de point de vue dans les approches d'IE présente les avantages de minimiser l'incomplétude des exigences, réduire la complexité du problème en séparant les préoccupations et enfin augmenter la traçabilité en associant les exigences aux points de vue (Amroune 2014). Cependant, l'identification des points de vue lors du processus d'élicitation des exigences reste un problème crucial. Bendjenna (Bendjenna et al. 2010)traite ce problème en proposant une approche qui combine les notions but, scénario et point de vue pour présenter un intérêt particulier.

La Figure 3.7 présente le métamodèle de l'approche proposée : Les parties prenantes (utilisateur, organisation ...) commencent par décrire leurs buts (objectifs). Un but est défini généralement par une Préoccupation Fonctionnelle (PF)

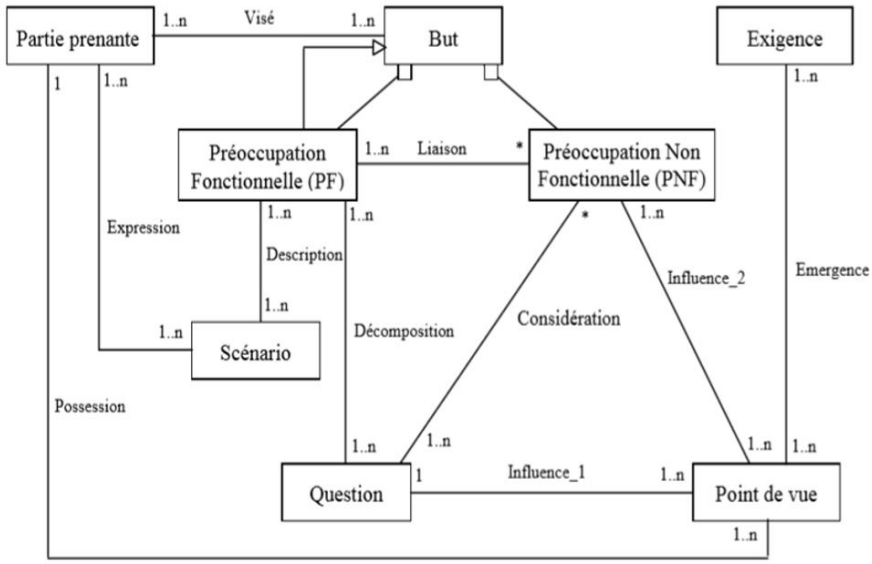

FIGURE 3.7 – Métamodèle pour combiner but, scénario et point de vue dans l'IE.

et un ensemble qui peut être vide de Préoccupation Non-Fonctionnelle (PNF). Une PF représente un but métier, par exemple la fabrication d'un produit. Un scénario est une représentation textuelle qui décrit une concrétisation possible d'une PF, il agit comme moyen pour créer une vision partagée entre l'analyste et les autres parties prenantes. Ensuite, à partir de chaque PF,un ensemble de questions est explicité. Chacune de ces questions va être posée par l'analyste afin de couvrir ou faire face à une caractéristique particulière de la PF à laquelle elle est attachée. L'identification de ces questions permet de s'assurer que toutes les exigences nécessaires pour satisfaire une PF vont être élicitées et spécifiées. Enfin, les parties prenantes expriment leurs points de vue sur la façon de réagir à chaque question en considérant la ou les PNF(s) associées. Ces points de vue

contiennent les exigences qui concrétisent les objectifs initiaux. La définition d'un point de vue doit répondre à trois interrogations principales : Pourquoi ces exigences sont nécessaires (question) ? quelles sont les contraintes du système qui sont associées à ces exigences (PNF(s)) ? , et comment cela peut être réalisé (les exigences) ?

3.4 LE POINT DE VUE DANS LES GRAPHES CONCEPTUELS

Le système VIEWS (Davis 1987)a intégré la notion de point de vue sous forme de vue. VIEWS permet de représenter des objets structurés en décrivant plusieurs vues contenant des éléments communs. Une vue est définie par un réseau de parties, de relations et de contraintes, ce qui permet de cerner les relations qu'un objet peut posséder. Le modèle est semblable à celui de TROPES, dans le fait que les points de vue sont prédéfinis, les utilisateurs du système ne peuvent pas ajouter ou modifier les vues et les points de vue. La notion de point de vue dans le domaine des graphes conceptuels (Sowa 1983) est défini par Ribière, qui selon elle le point de vue prend une dimension contextuelle et personnelle. Deux éléments sont distingués :

— Le *focus* (point de focalisation) : représente le contexte de travail de l'expert et de son objectif. Plusieurs experts peuvent partager le même focus.
— L'*angle de vue* : représente les capacités et les compétences de l'expert. L'angle de vue est propre à chaque expert.

A partir de cette définition, deux modèles sont proposés pour représenter des connaissances multi-expertes à travers la représentation et la gestion de multiples points de vue dans le formalisme des graphes conceptuels. Le premier modèle, C-VISTA (Ribière 1999), permet la représentation des terminologies et des ontologies personnelles de plusieurs experts. Il propose un moyen de relier entre elles ces terminologies ou ontologies personnelles. Le second modèle, CG-VISTA (Ribière et Dieng-Kuntz 2002), permet d'indexer par des points de vue

et de faire cohabiter les différentes analyses ou descriptions des experts d'un même objet, tout en fournissant, à travers la définition de chaque point de vue, le moyen d'interpréter de façon pertinente les connaissances formalisées. Il permet également de partitionner la base de connaissances et de mettre en place un accès filtré à cette base selon un point de vue utilisateur. Le formalisme des graphes conceptuels proposé par Sowa se compose des éléments de base tels que *des concepts, des relations, des types de concept*, et des types de relation. Les types de concepts ou de relations sont organisés dans un treillis (une hiérarchie) avec des liens sous-type entre eux. Ce type de lien est exploité et étendu dans ces travaux pour représenter et intégrer la notion de point de vue. Cette nouvelle relation de sous typage point de vue est proche du principe de VBOOL qui définit un lien de visibilité entre les classes, et aussi du modèle TROPES qui définit des passerelles entre des concepts définis selon différents points de vue. Comme le modèle de ROME, ce modèle permet aussi d'exprimer des représentations multiples d'un concept : un concept instancie un type de concept basique et est relié à plusieurs concepts orientés via des liens de représentation. Cependant la représentation du point de vue dans cette approche reste limitée. Elle s'appuie sur le concept de graphes conceptuels dont il manque des langages de requêtes. Ainsi, elle représente des limitations liées à l'interopérabilité entre les différents points de vue (Hicham Behja 2009).

3.5 LE POINT DE VUE DANS EN ECD

Dans le domaine de l'Extraction de Connaissances à partir de Données (ECD), la notion de point de vue a été initiée par Behja et al. (Hicham Behja, Trousse et Abdelaziz Marzak 2005), au sein de l'équipe projet AxIS à l'INRIA Sophia Antipolis, principalement pour l'annotation du processus d'ECD en termes de point de vue de l'analyste.

Tenant compte des travaux de (Trousse 1998) sur la gestion des points de vue en conception coopérative, et partant du constat qu'un processus d'ECD est un processus complexe qui nécessite l'intervention de plusieurs analystes ayant chacun ses préférences, son domaine de compétence et ses objectifs (son point de vue), un point de vue du processus d'ECD décrit *une vision particulière qu'a l'analyste sur son raisonnement et/ou sur les données manipulées. Il correspond, d'une part, à un savoir-faire selon un point référentiel de connaissance du domaine (connaissance du domaine analysé) et d'autre part, à définir un plan d'exécution établi lors d'une analyse d'ECD (connaissance sur le domaine de l'analyste)* (Hicham Behja 2009).

Aussi, Behja se base dans sa définition d'un point de vue sur la spécification d'un ensemble de critères génériques qui intègre des connaissances du domaine analysé et du domaine de l'analyste, à savoir : le contexte, l'objectif, la dimension, et les variables illustratives.

L'objectif d'une telle modélisation est de faciliter l'analyse et l'utilisation d'un processus d'ECD en termes de point de vue. Dans cette modélisation, Behja définit une vue par *l'ensemble des données (métadonnées) obtenues par transformation d'une vue à l'étape précédente correspondant à un point de vue donné*. Initialement, la vue est composée des données brutes à analyser. Les point de vue selon Behja sont reliés par des relations d'équivalence ou de subsomption.

Pour permettre la réutilisation des analyses faites selon les points de vue des analystes, un format de métadonnées est proposé pour annoter le processus d'ECD en termes de points de vue et de vues. Une activité d'analyse d'ECD guidée par le point de vue est défini *par un processus de génération, transformation et enrichissement de vues*. Ces dernières sont annotées par des métadonnées permettant de garder la trace de leur génération. La Figure 3.8 illustre un exemple d'une succession de vues générées lors de l'analyse des fichiers log http du serveur Web d'une plateforme d'enseignement à distance (E. Zemmouri, Behja et Marzak

2010) :

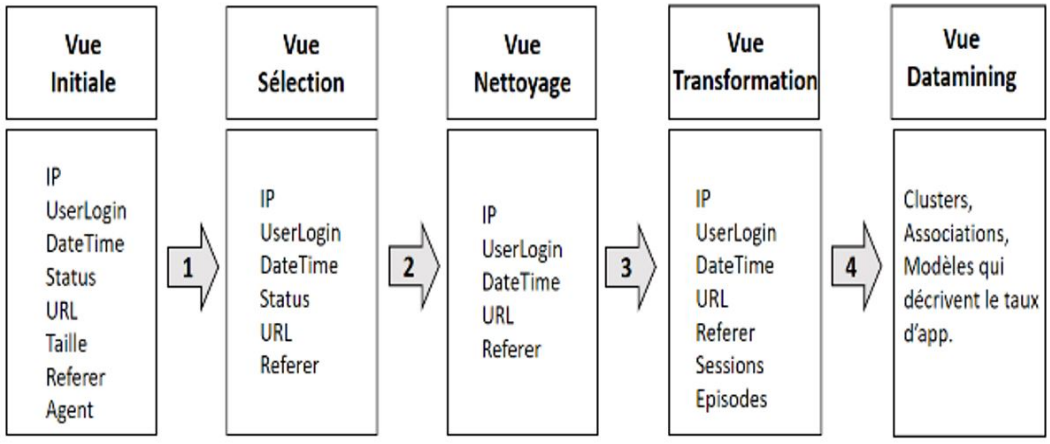

FIGURE 3.8 – Exemple de vues générées lors de l'analyse de fichiers log http.

Motivé aussi par le besoin de disposer d'un outil qui soit ouvert et spécialisable par les experts pour mener des analyses multi-vues en ECD, Behja propose une plateforme à objet spécialisable ainsi que des composants de bases réutilisables basés essentiellement sur l'utilisation des patrons de conception (GAMMA 1995). Cette plateforme est conçue pour le développement de prototypes, l'intégration de nouveaux composants et l'extension des composants existants pour s'adapter au mieux aux exigences spécifiques des analystes. Elle intègre un modèle du processus d'ECD composé de trois grandes étapes : prétraitement, fouille de données et post-traitement ; un modèle du point de vue (Figure 3.9) et un modèle d'intégration du point de vue en ECD .

L'intégration du point de vue dans un processus d'ECD nécessite, à la fois, la spécification des attributs du domaine analysé et du domaine de l'analyste du

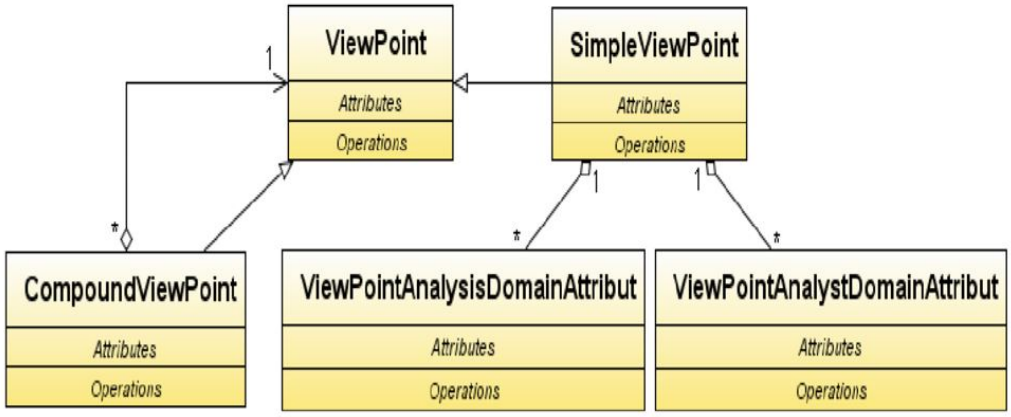

FIGURE 3.9 – Modèle du point de vue en ECD. (Hicham BEHJA 2009)

point de vue. Ces attributs permettent d'une part, de cerner le domaine à analyser et d'autre part de définir un plan d'exécution selon le point de vue.

L'approche de Behja fournit une modélisation de haut niveau du processus ECD . Cependant elle présente plusieurs limites et reste abstraite et difficile à implémenter dans des cas pratiques d'ECD . Pour pallier ceci, l'approche de Zemmouri (E. ZEMMOURI 2013), qui se situe dans la continuité des travaux de Behja et al., envisage essentiellement une caractérisation multi-critères de la notion de point de vue en ECD. Cette caractérisation permet une indexation plus précise des connaissances du processus et une assistance de l'utilisateur expert tout au long du cycle de conception et d'exécution du processus d'ECD multi-vues. Le point de vue dans cette approche est défini par « *une interface permettant (1) d'accéder à un sous-ensemble de connaissances de domaine (domaine analysé et domaine de l'analyste) et conduisant l'analyste à atteindre son objectif d'analyse, et (2) de capturer la logique du raisonnement et la trace des décisions prises par l'analyste lors d'une analyse*

d'ECD (i.e. capturer la sémantique du processus en termes du point de vue de l'analyste) ». L'approche propose un modèle du point de vue pour une caractérisation multi-critères du point de vue en ECD Figure 3.10 ; le modèle décrit le point de vue en ECD en termes d'un ensemble de critères génériques (domaine analysé, domaine de l'analyste et contexte de l'analyse) identifiés à base du modèle de référence CRISP-DM et formalisé sous forme d'une ontologie OWL. Ces critères sont indépendants de la tâche et du domaine d'application. Ils permettent de modéliser la vision de l'analyste sur les données analysées, l'objectif d'analyse et une partie de l'expertise nécessaire aux nombreuses prises de décisions effectuées pendant l'analyse.

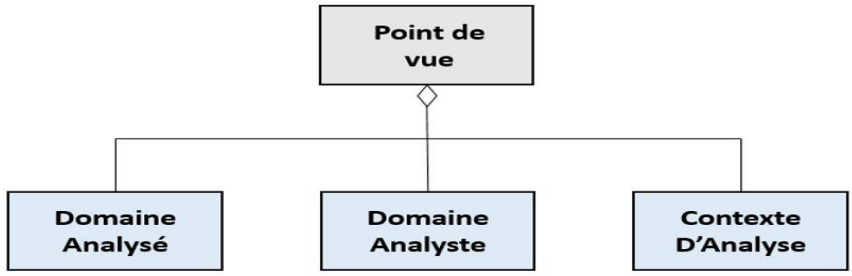

FIGURE 3.10 – Le modèle de point de vue en ECD. (E. ZEMMOURI 2013)

Le point de vue domaine analysé correspond au système ou domaine d'application étudié. Ce point de vue concerne les connaissances statiques sur le domaine (et les données) qui sont indépendantes de la tâche d'extraction (classification, clustering, associations, ...). Le point de vue domaine de l'analyste concerne tout ce qui est propre à la résolution du problème de l'extraction et les connaissances mises en oeuvre par l'analyste (expert) pour cette résolution. Finalement, le point de vue contexte d'analyse représente les objectifs opérationnels du problème (projet) de l'extraction et l'environnement dans lequel se déroule le processus d'ECD. Aussi, afin de garantir un partage de connaissances entre

les différents acteurs d'une analyse multivue et d'assurer une meilleure réutilisabilité, coordination et compréhensibilité entre les analystes, un ensemble de relations sémantiques entre points de vue (équivalence, inclusion, conflit, exigence et indépendance) avec un ensemble de mécanismes de raisonnement et d'inférence ont été développés. La modélisation du point de vue en ECD devrait favoriser la coordination et la compréhension entre les différents experts d'une analyse multi-vues. Elle favorise également la réutilisation d'une analyse selon un point de vue donné.

3.6 LE POINT DE VUE EN ONTOLOGIES

Une ontologie capture et structure la connaissance dans un domaine et, ce faisant, elle capture la signification des concepts qui sont spécifiques pour ce domaine. Il existe généralement plusieurs façons d'appréhender les connaissances d'un domaine, c'est-à-dire différents points de vue selon lesquels ces connaissances peuvent être représentées. Ainsi, le même domaine peut avoir plus d'une ontologie, où chacune d'entre elles est décrite selon un point de vue ou une perception particulière.

Dans une ontologie qui confère à un même univers de discours plusieurs descriptions partielles existent telles que chacune est relative à un point de vue. Ce besoin de prise en compte de connaissances multipoints de vue, au sein d'une même ontologie, provient essentiellement d'un environnement multi-expert et multidisciplinaire où plusieurs personnes ou groupes de personnes diversifiés coexistent et collaborent entre eux. Chaque expert a ses intérêts particuliers et perçoit différemment les entités conceptuelles du même univers de connaissances à représenter. Ainsi, les différents experts peuvent être en désaccord sur la signification des concepts du domaine et sur les termes pour les désigner. En vue de pouvoir intégrer, dans une seule ontologie, les avis ou les définitions de plusieurs

experts sur un concept donné du domaine, Falquet et Mottaz (Falquet et Mottaz Jiang 2001) ont introduit la notion d'ontologies multipoints de vue, qu'ils définissent par « *une ontologie dans laquelle chaque concept [du domaine de discours] peut avoir plusieurs définitions différentes, chacune d'entre elles représentant un point de vue différent (selon le point de vue, un concept peut également être situé à plusieurs emplacements différents dans la hiérarchie [de subsomption])*».

Dans ce contexte, les points de vue sont pris en compte « à l'intérieur d'une conceptualisation/vision » du domaine (Figure 3.11), et il ne doit pas y avoir de conflits entre deux points de vue d'une même conceptualisation, ce qui constitue la principale limite de ce modèle, vu que les conflits entre experts sont parfois enrichissants (Gesche 2008).

L'idée du modèle proposé est que pour définir un concept, chaque expert exprime son avis en proposant une définition qui est rattachée au concept et qu'ensuite, on essaie de se diriger progressivement vers un consensus en rattachant chaque définition au point de vue adéquat. L'objectif est d'aboutir à une ontologie où, pour chaque concept, on a au maximum une définition par point de vue. Dans le même contexte, au lieu de représenter des concepts dans une ontologie selon plusieurs points de vue par un modèle de l'ontologie multi-points de vue, le formalisme C-OWL (Bouquet et al. 2003) a contextualisé des ontologies en OWL pour intégrer la notion de point de vue dans les ontologies du Web sémantique.

En suivant le même principe, Bach (T. L. Bach 2006) a proposé un modèle pour la représentation d'ontologies multipoints de vue appelé MVP, et une extension du langage d'ontologie OWL appelée MVP-OWL. L'objectif de ce modèle vise à construire et exploiter un Web Sémantique dans une organisation hétérogène, comportant différentes sources de connaissances et différentes catégories d'utilisateurs, en construisant une ontologie multipoints de vue permettant d'ex-

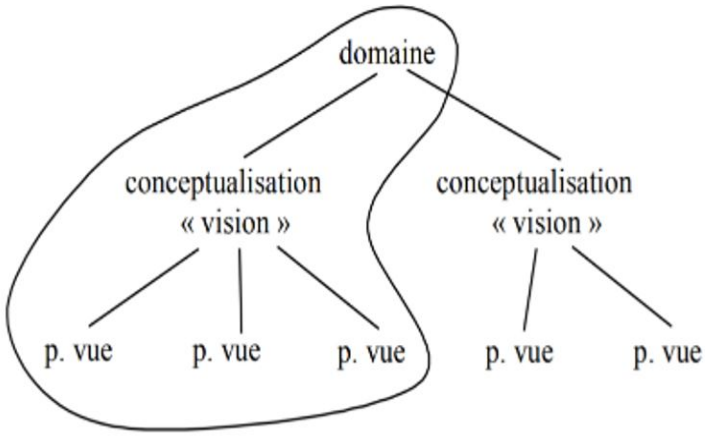

FIGURE 3.11 – Points de vue selon une conceptualisation du domaine. (FALQUET et MOTTAZ JIANG 2001)

primer des terminologies, des engagements ontologiques et des connaissances. Le point de vue dans le modèle MVP de Bach est introduit comme une entité supplémentaire en plus des entités classiques (classe, relation et instance), et il est défini selon deux niveaux de représentions des connaissances :

— Au niveau conceptuel (terminologique), un point de vue correspond à une vue perspective composée de certaines propriétés (attributs) d'une classe (concept). Ces propriétés constituent les informations pertinentes selon ce point de vue. Ainsi, un point de vue joue le rôle d'un filtre des caractéristiques d'une classe.
— Au niveau assertionnel, un point de vue correspond à un contexte particulier où les instances (objets) sont décrites et utilisées. Les descriptions à propos d'un objet selon un point de vue constituent des vues opinions. Elles sont cohérentes et valides dans le contexte défini par le point de vue.

Le modèle MVP permet l'héritage multiple selon plusieurs points de vue dans la définition d'une classe : une classe peut être sous-classe directe de plusieurs classes selon plusieurs points de vue différents. Comme une classe peut avoir plusieurs sous-classes selon plusieurs points de vue. Aussi, les relations entre les différents points de vue dans MVP sont définies comme des passerelles entre des classes, propriétés ou instances appartenant à des points de vue différents. Le principal objectif de ces passerelles est de pouvoir relier les classes décomposées différemment selon des points de vue différents et permettre donc de raisonner avec les classes et instances définis dans le modèle en entier, à travers des points de vue différents.

Par la suite, le point de vue selon (M. Boufaida et Fouzia 2007) est fondé sur la conjonction acteur/information. Le terme de point de vue est défini comme "une position conceptuelle mettant en liaison, d'une part, un acteur qui observe et, d'autre part, un monde qui est observé". Les acteurs peuvent observer un même univers de discours produisant des points de vue qui peuvent être considérés de différentes manières :

— *Points de vue uniformes* : dans ce cas, tous les acteurs ont la même vision de l'univers de discours et produisent des représentations équivalentes.
— *Points de vue complémentaires* : dans ce cas, chaque acteur voit une partie du monde observé. Chaque point de vue est une représentation partielle et cohérente du monde. Les différentes représentations qui découlent des différents acteurs sont alors complémentaires.
— *Points de vue comparables* : dans ce cas les acteurs produisent des représentations comparables au sens plus général/spécifique.

En se basant sur le même travail, (Hemam et Z. Boufaida 2009) ont défini un langage d'ontologies capable de supporter une représentation multi-points de vue basé sur le formalisme des Logiques de Descriptions (LDs). Récemment, (Djezzar et Z. Boufaida 2015)ont proposé un processus de classification d'un

individu dans une ontologie multi- points de vue. Cette classification s'effectue selon un ou plusieurs points de vue et permet de trouver, pour chaque point de vue, des concepts locaux dans lesquels l'individu est susceptible d'être une instance. L'objectif de tous ces travaux est de :

- Répondre aux problèmes de la multi-représentation.
- Fournir une meilleure visibilité et accès aux éléments de l'ontologie (concept, rôles et Individu).
- Permettre un développement modulaire collaboratif parmi les différentes communautés du même domaine.
- Bénéficier de la représentation multi-point de vue de connaissance afin de permettre leur évolution.

Comme les modèles de TROPES et MVP, DJEZZAR et BOUFAIDA définissent des passerelles pour relier les points de vue. Ils différencient entre : passerelle d'inclusion avec plusieurs sources, passerelle d'inclusion bidirectionnelle, et passerelle d'exclusion bidirectionnelle. Aussi, cette approche diffère de celle de C-OWL dans le fait que C-OWL est concernée par la façon de raisonner sur et les alignements entre différentes ontologies, par contre cette approche représente la façon dont une nouvelle et unique ontologie est définie de manière à ce que différents points de vue soient inclus.

En vue de traiter la diversité du contenu généré par les utilisateurs dans une approche du web sémantique, (DESPOTAKIS 2013) propose le ViewS (Viewpoint Sémantique) pour modéliser, représenter, capturer et analyser la sémantique des points de vue des utilisateurs. Un point de vue est défini par *"le focus et la collecte des déclarations pertinentes intégrées dans une partie du contenu généré par l'utilisateur "*. Le focus du point de vue désigne les aspects et les caractéristiques mentionnés dans les énoncés faits par l'utilisateur en tant que vision ou perception du domaine. Le point de vue est représenté par le tuplet : ensemble d'utilisateurs (U), ensemble d'objets digitaux (O), ensemble des énoncés faits par l'utilisateur (S),

ensemble d'ontologies (Ω) qui représente une ou plusieurs dimensions reliées au domaine ou le domaine lui-même, un ensemble d'annotation ontologiques (C) qui représentent la sémantique du point de vue, et une sémantique représentation du focus (F) de point de vue. Les ensembles U, O, S et Ω forment l'entrée du processus de modélisation du point de vue, tandis que le C et le F constituent la sortie.

La Figure 3.12 illustre les relations entre les concepts du point de vue :

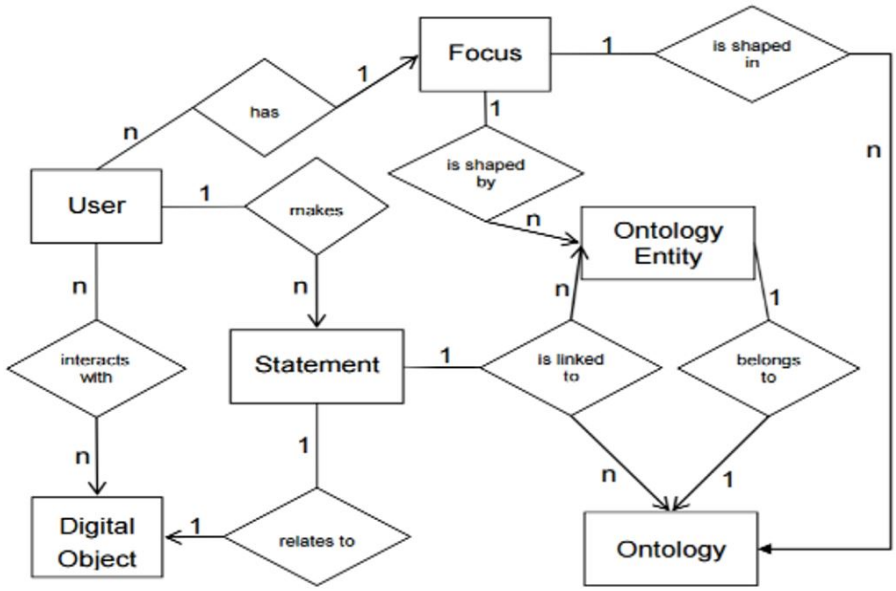

FIGURE 3.12 – Les relations entre les concepts du Point de View dans l'approche ViewS. (DESPOTAKIS 2013)

Le principal objectif de l'approche est d'aborder trois défis principaux : (a) permettre une meilleure compréhension des utilisateurs en capturant la sémantique

de leurs points de vue ; (b) représenter formellement les points de vue des utilisateurs en capturant le focus du point de vue et identifier la projection des modèles d'utilisateurs sur le domaine d'intérêt ; Et (c) permettre l'exploration de la diversité en fournissant des méthodes intelligentes d'analyse et de comparaison des points de vue.

3.7 SYNTHÈSE

A titre de synthèse, nous constatons que la notion de point de vue est présente sous le nom de perspective ou vision dans la majorité des approches, surtout dans les approches orienté- objet. Dans le tableau ci-dessous, nous synthétisons les différentes approches de représentation des connaissances multi-points de vue. Nous utilisons comme critères de comparaison : la caractérisation des points des vues et des vues dans le modèle, le formalisme et le langage utilisés pour implémenter ou représenter le modèle, et les liens définis pour relier les différents points de vue.

TABLE 3.1 – Synthèse des différentes approches de représentation multipoints de vue des connaissances

Approches		Points de vue	Liens entre points de vue	Formalisme/ Language	Nouveaux concepts
Représentation de connaissances	**KRL**	Perspectives	Les points de vue sont indépendants	Orienté-Object	- Classe de base - Classe de spécialisation
	ROME	Perspectives		Orienté-Object	- Classe d'instanciation - Classe de représentation
	FROME	Perspectives	Exclusion mutuelle	Orienté-Object	- Concept - Point de vue - Passerelle
	TROPES	Perspectives	Les passerelles	Orienté-Object	- Classe flexible - lien de visibilité
Modélisation Orientée Object	**VBOOL/VBOOM**	- Vue = perspective définie sur l'objet - Point de vue= une composition de vues sur l'objet.	Des vues peuvent être en exclusion mutuelle	Orienté-Object	- Classe flexible - Lien de visibilité
	VUML	- Point de vue= vision d'un acteur sur le système - Vue : application d'un point de vue sur une entité donnée.	Relations de dépendance entre les vues	Orienté-Object	- MultiViewsClass - ViewExtenson - AbstractView - Base - View - ViewDependency
	VxUML	Extension de VUML		Orienté Object	Sonde d'événements état multivue
Langages de programmation	**Programmation par sujet**	- Point de vue= sujet= perception générique d'un acteur	- Règles de mise en correspondance - Règles de combinaison - Règles de combinaison et mise en correspondance en même temps	Orienté Object	-Sujet

	Programmation par aspects	- Point de vue= Aspect= capture les préoccupations transversales d'un système		Orienté Object	-Aspect
	Programmation par rôle	- Point de vue= abstraction représentant partiellement un univers du discours contribuant à l'élaboration de représentation multiple de celui-ci		Orienté Object	-Structure représentante - Relation de visibilité
Graphes Conceptuels	**C-VISTA / GG-VISTA**	- Point de vue caractérisé par un focus et un angle de vue - Deux types de points de vue : point de vue perspective et point de vue opinion	- Lien d'équivalence - Lien d'inclusion - Lien d'exclusion	Graphes conceptuels	- Concept basique - Concept v-orienté - sous-typage point de vue - Lien point de vue
Ontologies	C-OWL	- Point de vue = contexte	Passerelles entre entités : - Equivalent - Compatible - Incompatible - Dedans - Dessous	Ontologie	- Mapping - BridgeRule - Correspondence
	MVP	- Point de vue comme perspective. - Point de vue comme interprétation contextuelle	Passerelles entre classes : - Equivalence - Inclusion - Exclusion	Ontologie	- Viewpoint - ClassWithViewpoint - belongToViewPoint - onViewPoint

	ViewS	Le focus et la collecte des déclarations pertinentes intégrées dans une partie du contenu généré par l'utilisateur	Passerelle	Ontologie	- User - Focus - Ontology Entity - Ontology - Statement - Digital Objet
	Ontologie Multi Point de vue	Une position conceptuelle mettant en liaison d'une part un acteur qui observe et d'autre part un monde qui est observé	Passerelle d'inclusion avec plusieurs sources, passerelle d'inclusion bidirectionnelle, et passerelle d'exclusion bidirectionnelle.	Ontologie	- Multi-viewpoints ontology - Viewpoint - Global concept - Stamps - Local concept. - Global role - Local role - Local hierarchy - mlti-instanciation -Bridge rule
Extraction de connaissances à partir des données (ECD)	**Plateforme object pour ECD**	Point de vue : perspective d'un analyste - Vue = résultat d'une transformation	- Lien d'équivalence - lien de subsomption	Orienté Object RDF	-ViewPoint - SimpleViewPoint - CompoundViewpoint - View
	ECD multi-point de vue	-Point de vue = une interface permettent (1) d'accéder à un sous-ensemble de connaissances de domaine et conduisant l'analyste à attendre son objectif d'analyse, et (2) de capturer la logique du raisonnement et la trace des décisions prises par l'analyste lors d'une analyse	- équivalence, inclusion, conflit, exigence et indépendance	Ontologie du web sémantique	-Domaine analysé - Domaine d'analyste - Contexte d'analyse

Deuxième partie

CONTRIBUTIONS POUR L'INTÉGRATION DE LA NOTION POINT DE VUE DANS UN SYSTÈME DÉCISIONNEL MULTI-EXPERT BASÉ SUR LE STANDARD CWM

4

NOTRE APPROCHE DE POINT DE VUE DANS UN SYSTÈME DÉCISIONNEL BASÉ SUR CWM

4.1 INTRODUCTION

La notion de point de vue varie selon son contexte d'utilisation. C'est une notion polysémique utilisée dans plusieurs domaines de l'informatique avec des sens différents. Dans ce chapitre nous allons développer notre approche de point de vue en CWM. Nous allons dans un premier temps décrire notre approche point de vue dans le cycle décisionnel en se basant sur le CWM et expliciter les objectifs à atteindre par rapport à cette définition. Par la suite, en se basant sur notre définition du point de vue, nous proposons une extension du CWM par les nouveaux éléments de notre métamodèle point de vue. Aussi, lors d'un processus multi-analyse de point de vue, il est important de mettre l'accent sur l'interaction et l'interdépendance entre les différentes analyses selon différents points de vue. De ce fait, nous définissons aussi dans ce chapitre un ensemble de relations pour modéliser les relations entre les points de vue des différents utilisateurs exprimés lors d'un processus décisionnel.

4.2 DESCRIPTION DE NOTRE APPROCHE DE POINT DE VUE

Comme mentionné dans le chapitre 3, la définition du point de vue varie selon le domaine et le contexte d'utilisation. Notre approche de point de vue dans le processus décisionnel en termes de CWM se situe dans un contexte orienté-objet

avec un formalisme d'ingénierie de connaissances.

L'intégration de la notion du point de vue dans la modélisation orientée-objet des systèmes permet de prendre en compte les acteurs du système dès les premières étapes d'analyse et de conception. Elle consiste en l'élaboration d'un modèle unique, partageable, et accessible suivant plusieurs points de vue. De ce fait, elle permet aux différents experts de se focaliser sur leur domaine d'expertise et de projeter leurs systèmes de valeurs sur le système à modéliser en vue d'améliorer la productivité, la fiabilité et la réutilisabilité des systèmes logiciels.

Le formalisme d'ingénierie de connaissance de notre approche tire profit des travaux de Behja (Hicham Behja, Trousse et Abdelaziz Marzak 2005) (Hicham Behja 2009)et Zemmouri (E. Zemmouri 2013) qui ont introduit le concept point de vue en ECD. Selon Behja (Hicham Behja 2009), le point de vue représente « une vision particulière qu'a l'analyste sur son raisonnement et /ou sur les données manipulées », tandis que Zemmouri a spécifié de plus le point de vue de l'analyse et il la définit par une « interface qui permet d'accéder à un sous-ensemble des connaissances de domaine, conduit l'analyste à atteindre son objectif d'analyse et de capturer la logique du raisonnement et de la trace des décisions prises par l'utilisateur lors d'une analyse ECD ».

En continuité avec les approches qui précédent, nous proposons d'intégrer la notion de point de vue dans le processus décisionnel par extension du CWM. Le cycle décisionnel est un système multi-vues souvent mené par une variété d'experts et d'utilisateurs qui interagissent et manipulent les métadonnées et les connaissances circulant lors des échanges de données dans ce cycle avec différents rôles et profiles. Chacun de ces utilisateurs manipule les différentes connaissances et savoir-faire et participe dans une ou plusieurs phases du développement et de maintenance de l'entrepôt de données et peut étendre les fonctionnalités fournies par la base du métamodèle CWM. Cet accès est différent dans

le sens qu'il respecte les besoins, les objectifs, les préférences et les différentes perspectives des utilisateurs sur les différentes entités de l'entrepôt de données, en bref, il doit respecter les différents points de vue.

Dès lors, on distingue six catégories d'utilisateur qui interagissent dans le cycle décisionnel :

— **Vendeurs d'outils et de plateforme d'entreposage de données** (warehouse platform and tool vendors) : responsable de la construction, l'exécution de l'architecture de l'entrepôt de données et le contrôle de la gestion des données.
— **Fournisseur de services professionnels** (professional service providers) : responsable de la configuration et de la gestion des ressources matérielles et logicielles requises par un entrepôt de données.
— **Développeur d'entrepôts de données** (warehouse developers) : responsable du développement de différents composants fonctionnels des applications des entrepôts de données, y compris l'extraction des programmes des systèmes sources, les applications ETL, les fonctions de nettoyage des données, les fonctions de gestion du système, par exemple l'automatisation des charges, les fonctions d'acquisition de données et d'autres.
— **Administrateur d'entrepôts de données** (warehouse administrators) : responsable de l'intégration et de la coordination des métadonnées et des données à travers les différentes sources de données ainsi que la gestion des sources de données, la conception de la base de données physique, l'exploitation, la sauvegarde et la récupération, la sécurité, les performances et le réglage.
— **Utilisateur final** (end user) : exploite l'entrepôt de données par des rapports et des requêtes.
— **Gestionnaire de la technologie de l'information** (information technology managers) : responsable de l'implémentation et de la maintenance de l'infrastructure technologique d'une organisation. Il surveille les besoins opéra-

tionnels de l'organisation, recherche des stratégies et des solutions technologiques en vue de construire un système rentable et efficace pour atteindre ces objectifs.

4.2.1 *Notre définition du point de vue*

Notre principal objectif est de capturer et de garder la trace des décisions prises par chaque utilisateur lors de l'exploitation du cycle décisionnel en vue d'assurer une importante coordination et compréhensibilité entre les acteurs d'une analyse multi-vues et une meilleure réutilisation des métadonnées et des connaissances lors du processus décisionnel. Par la suite, nous souhaitons pouvoir comparer, partager et réutiliser ces points de vue. Ainsi nous définissons le point de vue par :

— Un **point de vue** est *« la perspective ou la vision d'un acteur sur le processus décisionnel ou sur une partie de ce processus »*. Le point de vue fait référence à un ensemble de données qui permettent de caractériser la situation dans laquelle s'inscrit l'interaction entre l'utilisateur et le système. Chaque point de vue correspond à une catégorie d'utilisateur et permet de capturer les connaissances expertes pertinentes selon 1) le Contexte d'analyse, 2) l'Objective d'analyse, et 3) la Préférence de l'utilisateur exprimée durant son exploitation du processus décisionnel (Figure 4.1) :

- Le **contexte d'Analyse**, décrit l'environnement dans lequel une analyse du processus décisionnel se déroule. Il représente le domaine d'application (par exemple : santé, finance, marketing, production, ingénierie...). Nous définissions un contexte par un nom et une description.
- L'**objectif de l'analyse**, décrit l'objectif d'un utilisateur d'une vision opérationnelle (métier). L'objectif d'analyse est exprimé d'une manière infor-

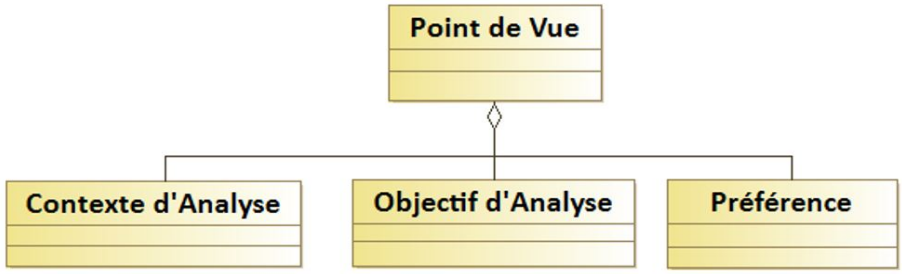

FIGURE 4.1 – Les composants de notre modèle Point de Vue.

melle en langage naturel par l'acteur, par exemple « Améliorer la qualité d'un cours ». Nous considérons qu'un objectif est défini par un nom, une description et un degré d'intérêt (DOI) qui exprime l'intérêt de l'utilisateur pour accomplir cet objectif.

- La **préférence**, décrit ce qu'un utilisateur envisage d'accomplir dans le système. Elle met l'accent sur les données intéressantes que l'utilisateur voudrait durant une analyse du processus décisionnel. Par exemple, une préférence peut être une synthèse annuelle des informations sur la moyenne des étudiants dans chaque cours, ou simplement année = 2017. Nous considérons qu'une préférence est définie par un nom, et une description.

4.2.2 *Notre métamodèle Point de Vue*

En dépit des fonctionnalités offertes par le CWM et qui font de lui un standard puissant pour représenter les phases de la chaîne décisionnelle (section 2.2), son métamodèle et sa sémantique ne permettent pas de représenter et d'intégrer l'aspect utilisateur avec ses préférences et ses points de vue.

Le métamodèle CWM se base sur la modélisation orientée-objet et sur l'expressivité et le pouvoir d'UML pour définir les structures et les relations complexes des métadonnées. Le CWM utilise le concept d'héritage orienté-objet, comme mentionné dans le chapitre précédent, pour étendre les éléments de modélisation d'UML en vue de définir de nouveaux éléments de modélisation qui représentent des concepts d'entreposage de données et du business analyse. Le CWM utilise UML dans trois rôles principaux :

— **UML est utilisé comme un équivalent du métamodèle MOF** : UML ou les parties qui correspondent au modèle MOF, à la notation UML, et à OCL sont utilisés respectivement comme langage de modélisation, notation graphique et langage de contrainte.
— **UML est utilisé comme le métamodèle de base** : spécialement un sous-ensemble UML représenté par le paquet Object Model, est utilisé comme base du CWM dont d'autres métamodèles héritent des classes et des associations.
— **UML est utilisé comme le métamodèle orienté-objet pour CWM** : UML, en particulier le package ObjectModel, est utilisé pour représenter des sources de données orientées objet. Néanmoins, le CWM utilise uniquement la modélisation de structure statique d'UML pour modéliser les métadonnées partagées pour l'entreposage de données et pour le business intelligence. La modélisation de cas d'utilisation d'UML qui met l'accent sur la fonctionnalité du système tel qu'il apparaît chez les utilisateurs extérieurs et la modélisation comportementale d'UML qui met l'accent sur le comportement des objets du système, y compris leurs interactions, événements, contrôles et flux de données, sont inexistants dans le métamodèle CWM.

Cependant, le CWM utilise uniquement la modélisation structurelle statique d'UML pour modéliser le partage des métadonnées pour l'entreposage de donnée et le business analyse. Beaucoup de concepts clés d'UML n'existent plus dans le mé-

tamodèle CWM, à savoir : La modélisation des cas d'utilisation d'UML ,qui met l'accent sur les fonctionnalités du système telles qu'elles apparaissent chez les utilisateurs extérieurs, représentée par l'élément Use Case, l'entité Actor du métamodèle UML qui représente une personne, une organisation, ou un système en dehors de l'architecture du modèle mais qui interagit avec, et la modélisation comportementale de l'UML qui porte sur le comportement des objets du système, y compris leurs interactions, les événements, le contrôle et le flux de données.

Afin d'intégrer la notion point de vue dans le cycle décisionnel et garder la trace de l'utilisateur (ses objectifs, ses préférences, son domaine d'expertise...) durant une exploitation du processus décisionnel, nous étendons du CWM en utilisant ses mécanismes d'extension présentés dans le chapitre 2.

Dans la mesure où notre approche de point de vue prends une perspective d'ingénierie de connaissances avec un formalisme orientée-objet, et vue que le CWM est construit à base d'UML, nous nous inspirons de l'approche VUML introduite par Nassar (Nassar 2004) dans le but d'étendre le métamodèle CWM pour supporter l'approche point de vue. Le métamodèle VUML étend le métamodèle UML avec de nouveaux éléments de modélisation. Le concept de VUML se base sur la notion de classe multivue constituée d'une base et d'un ensemble de vues spécifiques reliées à la base par une relation d'extension. Elle permet de stocker et de restituer l'information selon le profil de l'utilisateur. Elle offre des possibilités de changement dynamique de point de vue et d'expression des dépendances entre vues. VUML tourne autour de trois concepts clés : la classe multivue (MultiViewClass), la base (Base) et la vue (View). La classe Multivue est définie comme l'élément de modélisation qui représente les besoins et les droits de l'utilisateur. Elle est composée d'une base et d'un ensemble de vue. La classe Base est l'entité core qui inclue les spécifications communes à tout type d'acteur. Tandis que la classe Vue représente l'entité de modélisation qui correspond à l'application d'un point de vue à une entité donnée. Une vue est associée à un

acteur en considérant comme implicite l'entité sur laquelle le point de vue de l'acteur s'applique.

Notre métamodèle point de vue est composé des classes suivantes :

— **MultiViewDecisionalProcess** : capture les caractéristiques des utilisateurs exprimées durant l'exploitation de processus décisionnel. C'est une structure composée d'une classe de base (Base) reliée à un ensemble de point de vue par une relation de dépendance VPExtension.
— **Base** : représente un composant de la classe MultiViewDecisionalProcess. Elle décrit les caractéristiques structurelles et comportementales communes à tous les utilisateurs de la classe MultiViewDecisionalProcess. La classe Base est reliée à la classe ViewPoint par une relation de dépendance (VPExtension).
— **ViewPoint** : c'est la classe qui représente la perspective d'une catégorie d'utilisateurs sur tout le processus décisionnel ou sur une partie de ce processus. Elle modélise les caractéristiques structurelles et comportementales spécifiques à un utilisateur particulier. Elle est connectée à la classe Base par la relation VPExtension. Chaque point de vue d'utilisateur a un contexte, un objectif d'analyse et une classe Preference qui représente la préférence de chaque utilisateur, exprimée durant l'exploitation du processus décisionnel.
— **VPExtension** : définit la relation entre la classe Base et la classe Point de Vue d'une classe MultiViewDecisionalProcess.
— **VPDependency** : définit la relation entre les différents points de vue des utilisateurs d'un processus décisionnel. Cette notion sera détaillée par la suite.
— **WarehouseAdministratorVP**, **WarehousePlatformVP**, **EndUserVP**, **WarehouseDeveloperVP**, **InformationTechnologyManagerVP**, et **Professional ServiceProviderVP** représentent le point de vue de chaque catégorie d'uti-

lisateurs représentée précédemment.

Afin d'intégrer le concept point de vue dans le processus décisionnel en termes de CWM, nous étendons le métamodèle CWM par les nouvelles classes et concepts de notre métamodèle Point de Vue. L'extension concerne les paquets *Core* de la couche *Object Model*, et le paquet *Business Information* de la couche *Fondation*. La Figure 4.2 illustre un fragment des métamodèles *Core* et *Business Information* : Le

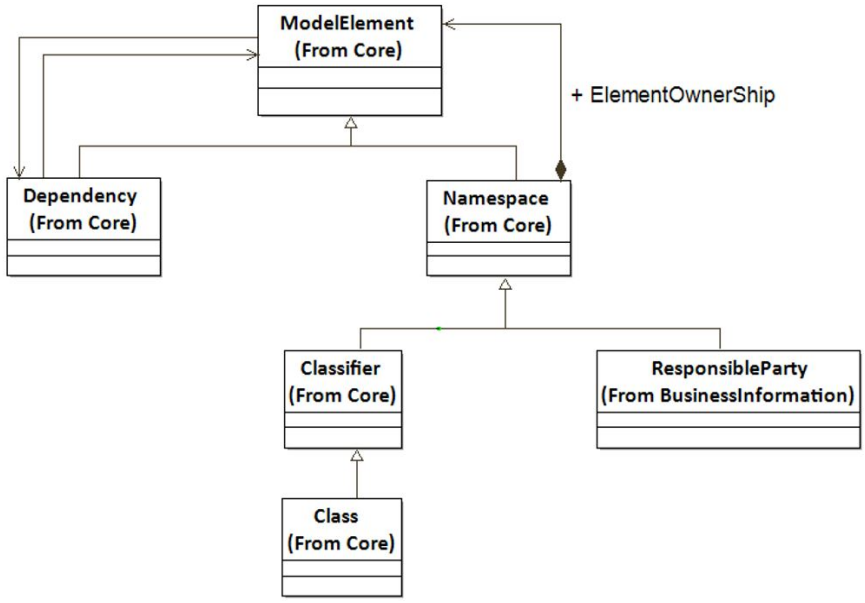

FIGURE 4.2 – Un fragment du métamodèle Core et Business Information du métamodèle CWM.

métamodèle *Core* définit les éléments de modélisation de base. Il représente aussi le *Core* du langage UML (voir chapitre 2). Ce métamodèle ne dépend d'aucun autre paquet du CWM. Il contient les classes et les associations de base utilisées par les autres paquets de la couche *Object Model* et par tous les autres paquets

appartenant aux autres couches du CWM. Il est constitué de nombreuses classes dont nous nous limitons seulement aux classes concernées par notre extension :

— **ModelElement** : est un élément qui représente une abstraction extraite du système à modéliser. Il représente dans le métamodèle toute entité qui porte un nom dans un modèle. *ModelElement* est la base de toutes les métaclasses dans CWM ; en d'autres termes, toutes les autres métaclasses du CWM sont des sous classes du *ModelElement*. Cette dernière est définie par les attributs :

- Name : c'est une chaîne de caractère qui définit le nom du *ModelElement*.
- Visibility : spécifie la visibilité de ce *ModelElement* dans le *NameSpace* qui le contient.

— **Namespaces** : ou espace de nom en français, représente une partie du modèle qui contient un ensemble des éléments du modèle. Le nom de chaque *ModelElement* appartient à un *Namespace* doit être unique. La classe *Namespace* spécialise la classe ModelElement, elle représente donc un *ModelElement* qui peut contenir d'autres *ModelElements*. Le *Namespace* contient l'attribut :

- OwnedElement : un ensemble de *ModelElement* possédé par le *Namespace*. L'attribut *Visibility* du *ModelElement* énonce si l'élément est visible en dehors du *Namespace*.

Namespace et *ModelElement* sont reliés par la relation de composition ElementOwnership, qui permet au *Namespace* d'avoir des *ModelElements*, et par conséquent, d'autres *Namespaces*. Cette association est l'un des principaux mécanismes de structuration au sein du CWM. La relation *ElementOwnership* est réutilisée largement par l'ensemble des métamodèles du CWM pour indiquer les relations de propriété entre les classes à tous les niveaux du métamodèle CWM.

— **Classifier** : est un élément qui décrit les caractéristiques structurelles et comportementales ; il peut prendre différentes formes, à savoir : classe, type de donnée, interface, composant, et d'autres qui sont définis dans les autres

paquets du CWM. Le Classifier est souvent utilisé comme un type qui peut déclarer une collection de caractéristiques (Feature) (les attributs, les opérations, et les méthodes). Il porte un nom unique dans le *Namespace* qui l'inclut. C'est une métaclasse abstraite. Comme la classe Classifier spécialise la classe Namespace, elle peut déclarer d'autres *Classifier* imbriqués dans sa portée.

— **Class** : le terme *Class* est utilisé par le CWM avec une signification identique à celle de la classe en UML, c'est une spécification abstraite des méta-objets qui intègre leurs états, leurs interfaces, et leurs comportements (d'une façon informelle). La classe est un *Classifier* ayant des instances multiples. Une Classe dans CWM peut avoir trois types de caractéristiques : les attributs, les opérations, et les références.

— **Dependency** : la dépendance énonce que l'implémentation ou le fonctionnement d'un ou plusieurs éléments nécessite la présence d'un ou plusieurs autres éléments. Dans le CWM la dépendance est une relation dirigée d'un client/s à un fournisseur/s (supplier), à condition que le client dépend de ce fournisseur, en d'autres termes : l'élément client nécessite la présence et la connaissance de l'élément fournisseur. Une classe *Dependency* dans CWM est caractérisée par les attributs suivants :

- *Kind* : contient la description de la nature de la relation de dépendance entre le client et le fournisseur. La liste des valeurs possibles que cet attribut peut prendre est illimitée.
- *Client* : représente l'élément concerné par l'élément fournisseur. Dans certains cas la direction de la relation n'est pas importante et sert seulement à faire la différence entre les deux éléments.
- *Supplier* : c'est l'inverse du client. Il désigne l'élément qui n'est pas concerné par un changement. Généralement, le supplier est l'élément le plus général.

Le métamodèle *Business Information* de la couche *Foundation* est aussi concerné par notre extension. Ce dernier fournit des services généraux à tous les paquets des autres couches du CWM en vue de définir des informations orientées-commerciales pour les éléments du modèle. Il est également conçu pour répondre aux besoins des systèmes d'entreposage de données et du business intelligence (Poole et al. 2002). Cependant, sa principale faiblesse et qu'il ne fournit pas une représentation complète du domaine du *Business Intelligence*, de plus, il reste un métamodèle à usage général.

Le paquet *Business Information* dépend du paquet Core de la couche *Object Model*. Il prend en charge la notion de parties responsables (Responsible Parties) et des informations sur la façon de les contacter. La classe *ResponsibleParty* permet aux personnes ou aux unités organisationnelles, telles que le service IT, d'être identifiées comme responsables de tout élément du modèle (*ModelElement*). Il peut être utilisé pour représenter des organigrammes complets, en permettant de lier la responsabilité des éléments du modèle dans un système d'information aux entités organisationnelles responsables de ces éléments. Dans notre approche, nous étendons la classe *Responsible Party* par les classes qui représentent les catégories des utilisateurs qui interagissent dans un processus de décision.

Notre métamodèle Point de Vue est construit en utilisant l'héritage comme technique d'extension du CWM pour étendre les paquets *Core* et Business Information en vue d'intégrer le point de vue de l'utilisateur dans le cycle décisionnel via le CWM. Le métamodèle Point de Vue étend le métamodèle CWM avec de nouveaux éléments de modélisation pour représenter l'approche point de vue. La classe *ResponsibleParty* est spécialisée par les classes qui représentent les catégories des utilisateurs qui interagissent dans le cycle décisionnel : *Warehouse Platform, Professional Service Provider, Warehouse Developer, Warehouse Administrator, End User*, et *Information Technology Manager*. La classe *Multiview Decisional Process* représente une spécialisation de la classe *Classifier* ; elle est composée

d'une classe *Base* et des classes *Point de vue* (ViewPoint). La classe *Point de Vue* est connectée à la *Base* par une relation de dépendance particulière (*VPExtension*). Chaque point de vue est associé à un responsable de parties unique. Les points de vue des utilisateurs sont reliés avec des relations de dépendance *VPDependency* qu'on développera dans le chapitre qui suit. Chaque point de vue est caractérisé par un *Contexte* et un *Objectif* d'analyse et une classe *Préférence* qui représente les préférences de l'utilisateur exprimées au long du processus décisionnel.Les classes *WarehousePlatformVP, ProfessionalServiceProviderVP, EndUserVP, WarehouseDeveloperVP, WarehouseAdministratorVP, et InformationTechnologyManagerVP* représentent le point de vue de chaque catégorie d'utilisateur. Ces classes spécialisent la classe point de vue. L'absence d'une catégorie d'utilisateur engendre l'absence de son point de vue.

Notre approche de point de vue par extension du CWM est illustrée dans la Figure 4.3, les classes du métamodèle point de vue sont représentées en gris, tandis que les classes du métamodèle CWM sont représentées en blanc.

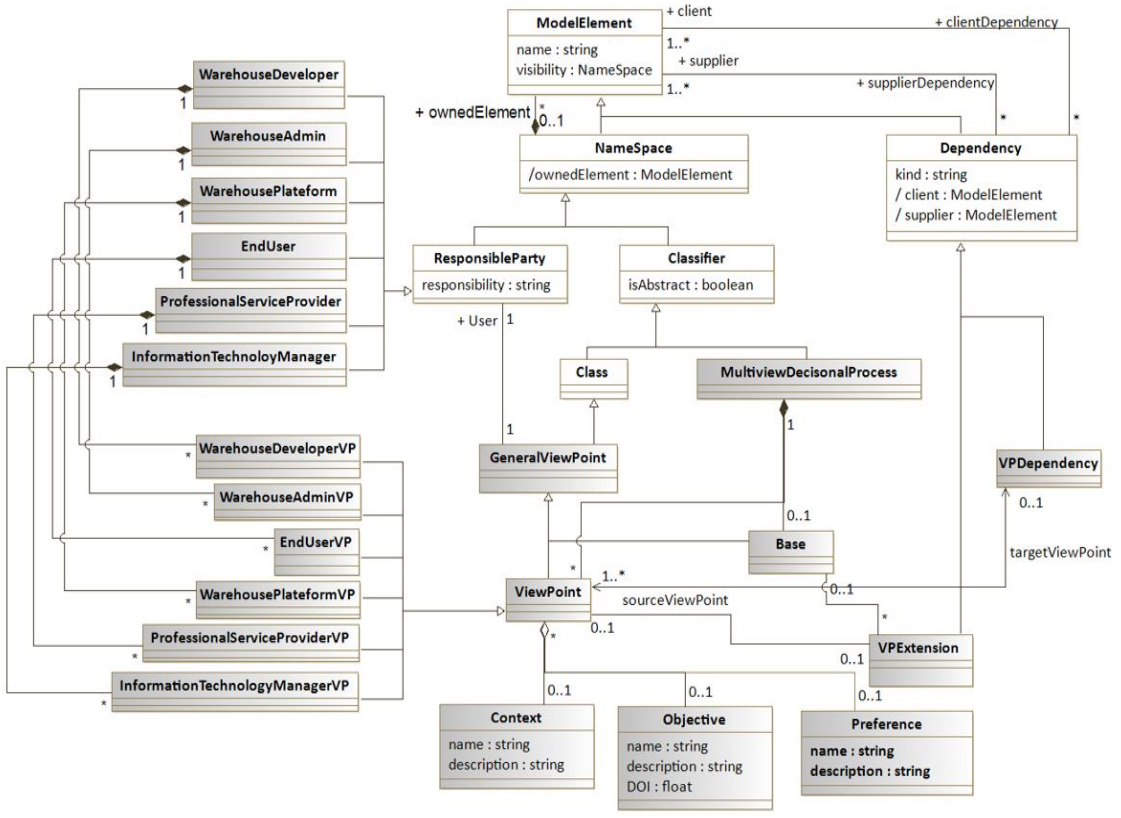

FIGURE 4.3 – Notre extension du CWM pour supporter la notion Point de Vue.

La description des principales classes de notre métamodèle point de vue est résumée dans le Tableau 4.1 ci-dessous. Chaque instance de classe est appelée méta objet et représente une construction du système à modéliser.

TABLE 4.1 – Description des classes de notre métamodèle Point de Vue

Métamodèles	**Classes**	**Description**
Core	ModelElement	Elément qui représente une abstraction du système à modéliser
	NameSpace	C'est un ModelElement qui peut contenir d'autre ModelElement
	Classifier	Elément qui décrit les caractéristiques comportementales et structureelles d'un élément de modèle
	Dependency	C'est une relation entre deux éléments. Un changement dans un élément (l'élément indépendant) engendre un changement dans l'autre élément (l'élement dépendant)
Busssiness Information	ResponsibleParty	C'est une classe qui permet d'identifier une personne ou une organisation d'être responsable d'un Model Element
ViewPoint	MultiViewDecisionnalProcess	C'est une classe qui représente le point de vue de l'utilisateur exprimé durantn le processus décisionnel. Elle spécialise la classe Classifier, et elle est composée d'une classe Base et un ensemble de points de vue (ViewPoint)
	GeneralViewPoint	C'est une classe qui spécialise l'élément de modélisation Class
	ViewPoint	Représente un point de vue ResponsibleParty qui corresponds à la perspective d'une catégorie d'utilisateur dans le processus décisionnel, ou une partie de ce processus. C'est aussi un composant de la classe MltiViewDEcisionalProcess qui spécialise la classe GeneralViewPoint. La classe ViewPiont modélise les caractéristiques structurelles et comportementales spécifique à un utilisateur.
	Context	Décrit l'environnement dans lequel se déroule l'analyse
	Objective	Décrit l'bojectif d'un utilisateur d'une vision opérationnelle
	Preference	Décrit ce que l'utilisateur d'une vison opérationnelle
	Base	C'est un élément qui spécialise la classe GeneralViewPoint. Elle décrit les caractéristiques structurelles et comportementales communes à tous les utilisateurs d'un MultiViewDecisionalproceess. La classe Base est reliée à la classe ViewPoint par la relation de dépendance (VPExtension)
	VPExtension	C'est une relation de dépendance entre la classe ViewPoint et la classe Base d'une classe MultiViewDecisinalProcess

	VPDependency	C'est une relation entre les point de vue (ViewPoints). Les informations du point de vue source dépendent des informations des points de vue cible
ViewPoint	WarehousePlatforme	Représentent les catégories des utilisateurs qui manipulent le cycle décisionnel
	ProfessionalServieProvider	
	WarehouseDeveloper	
	EndUser	
	InformationTechnologyManager	
	ProfessionalServieProvider	
	WarehousePlaformVP	Représente le point de vue de chaque catégorie d'utilisateur
	ProfessionalServiceProviderVP	
	WarehouseDeveloperVP	
	WarehouseAdministratorVP	
	EndUserVP	
	InformationTechnlologyManagerVP	

- Exemple illustratif

Pour illustrer notre approche, nous considérons l'exemple d'une organisation qui voudrait implémenter un entrepôt de données dans un environnement scientifique pour analyser des publications (Figure 4.4).

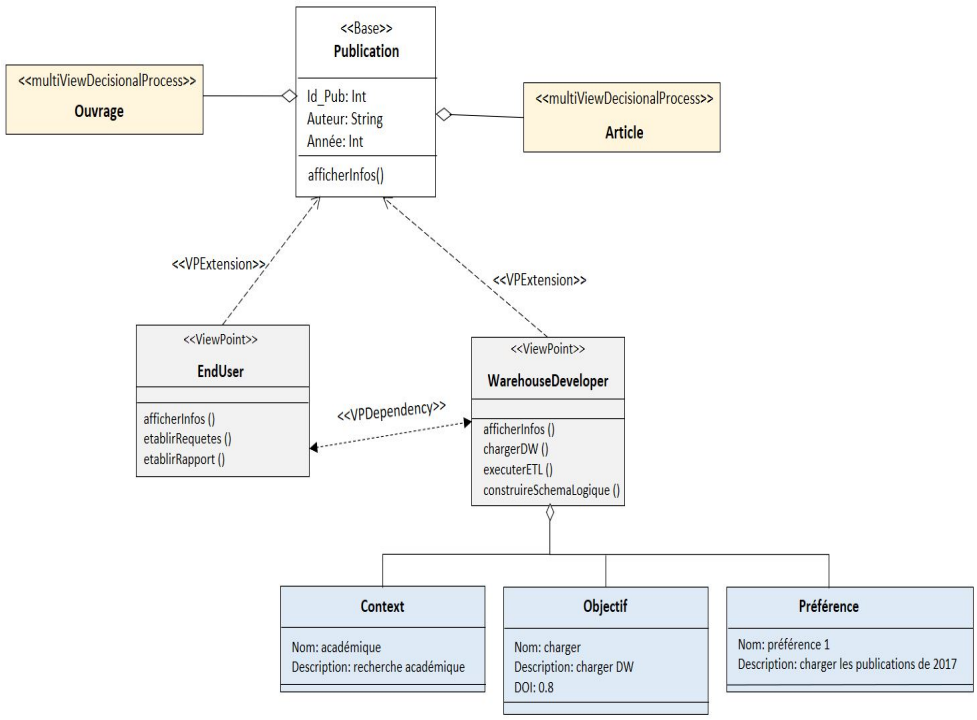

FIGURE 4.4 – Modèle point de vue pour un système de gestion de publications.

Chaque classe de notre modèle peut devenir une classe *MultiViewDecisionalProcess*, selon les points de vue des utilisateurs sur cette classe. Dans ce modèle

point de vue, nous spécifions seulement deux points de vue associés aux *Warehouse Developer* et End User. La classe Publication est constituée d'une base et de deux points de vue correspondant à ces deux acteurs. La base contient les informations (attributs et méthodes) partagées par ces acteurs. La classe *WarehouseDeveloper* décrit la spécificité d'une publication selon le point de vue d'un développeur d'entrepôt de données, tandis que la classe *EndUser* décrit les spécificités d'une publication selon le point de vue d'un utilisateur final. On peut remarquer que la méthode afficherInfos ( ) de la base est redéfinie dans les deux points de vue. Le point de vue du développeur de l'entrepôt de données définit son contexte, son objectif et sa préférence.

4.3 NOTRE DÉFINITION DES RELATIONS ENTRE LES POINTS DE VUE

Dans la plupart des systèmes de modélisation, les relations entre les objets jouent un rôle important dans l'expression de la dépendance entre les entités d'un domaine particulier. Comme nous avons cité dans le chapitre 3, les relations entre les points de vue ont apparu sous plusieurs noms dans différentes approches pour répondre à des objectifs divers.

Dans notre approche du point de vue en termes de CWM, nous envisageons d'exprimer l'interaction et l'interdépendance entre les différents points de vue lors du processus décisionnel. Pour cela nous définissons dans cette section un ensemble de relations entre points de vue afin d'améliorer la coordination et la compréhension mutuelle entre les points de vue des utilisateurs, de permettre la réutilisation des analyses effectuées dans un processus décisionnel basé sur CWM et d'apporter une assistance méthodologique pour les analystes de ce processus.

En se basant sur les travaux de la littérature, nous définissons trois types de relations entre les points de vue lors d'une exploitation d'un processus déci-

sionnel en se basant sur les attributs qui composent le point de vue, à savoir : contexte, objectif, et préférence. L'objectif principale de ces relations est d'assurer une meilleure coordination et compréhension mutuelle entre les utilisateurs et de permettre la réutilisabilité des analyses durant un processus décisionnel basé sur le CWM en termes de point de vue. Dans ce sens, nous définissons ci-dessous trois types de relation entre les points de vue d'un processus décisionnel basé sur le CWM.

4.3.1 *La relation d'équivalence*

Soient VP1 et VP2 deux points de vue différents VP1 est équivalent à VP2, si VP1 et VP2 ont le même objectif.

La relation d'équivalence entre deux points de vue permet d'identifier les points de vue de deux utilisateurs ayant le même objectif, mais utilisés dans des contextes différents. Pour illustrer ceci, nous considérons les points de vue de deux utilisateurs finaux, qui ont comme objectif de demander les rapports sur la vente d'un produit donné, l'un des utilisateurs exprime son point de vue dans un contexte de Marketing, tandis que l'autre l'exprime dans un contexte d'administration d'un système particulier Figure 4.5. Les deux points de vue des utilisateurs finaux sont équivalents car ils ont le même objectif (demander les rapports sur la vente d'un produit donné).

En conséquence, la relation d'équivalence entre les points de vue sert pour la réutilisation de l'expérience et l'assistance des utilisateurs dans un processus décisionnel

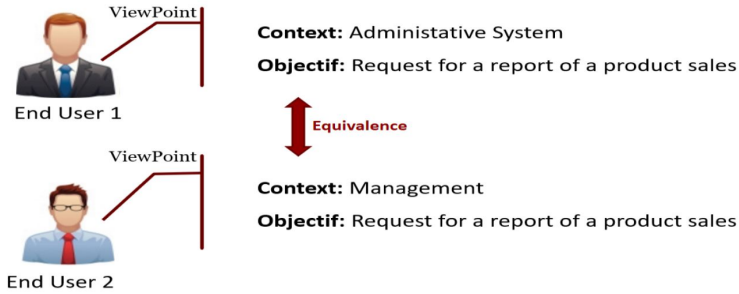

FIGURE 4.5 – Exemple de la relation d'équivalence entre deux points de vue.

4.3.2 *La relation d'inclusion*

Soient VP1 et VP2 deux points de vue différents. VP1 inclut VP2, si VP2 utilise partiellement VP1. En d'autres termes, la satisfaction de l'objectif de VP1 implique la satisfaction de l'objectif VP2.

À titre d'exemple, nous considérons les points de vue de deux utilisateurs du processus décisionnel basé sur le CWM : le développeur de l'entrepôt de données (Warehouse Developer) et l'administrateur de l'entrepôt de données (Warehouse Admin). Le premier utilisateur a comme objectif de point de vue de développer le design logique d'un entrepôt de données, tandis que le deuxième utilisateur a comme objectif de définir le schéma physique d'un entrepôt de données (Figure 4.6).

Alors, on dit que le point de vue du développeur du DW implique le point de vue de l'administrateur du DW.

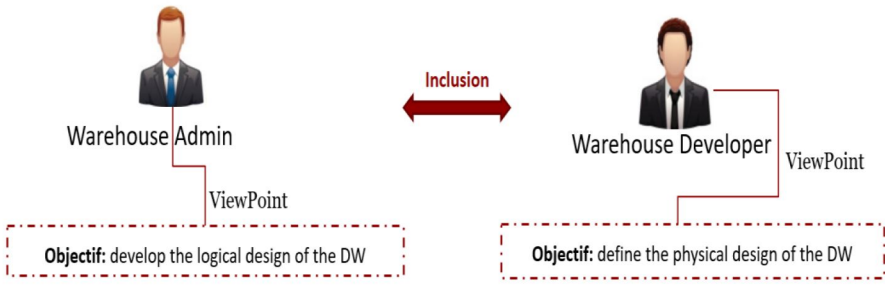

FIGURE 4.6 – Exemple de la relation d'inclusion entre deux points de vue.

4.3.3 *La relation d'indépendance*

Soient VP1 et VP2 deux points de vue différents. VP1 et VP2 sont indépendants, si l'existence de VP1 n'influence pas l'existence de VP2.

Par exemple l'existence du point de vue d'un utilisateur final n'affecte pas l'existence du point de vue d'un développeur de DW. La relation d'indépendance entre les points de vue est utile dans le cas où deux points de vue sont exprimés lors des phases différentes du cycle décisionnel, dont l'existence d'un point de vue d'utilisateur n'influence pas l'existence de l'autre point de vue. Ceci implique que deux analyses peuvent être menées sans que les résultats de l'une influencent l'exécution de l'autre.

4.4 REPRÉSENTATION DES RELATIONS DANS LE MÉTAMODÈLE POINT DE VUE

Afin d'illustrer les relations entre les points de vue décrite ci-dessous, nous étendons le métamodèle Point de vue présenté dans le chapitre 4 avec des nou-

velles entités qui modélisent ces relations.

Dans ce cas, la classe *VPDependency* qui spécialise la classe *Dependency* du métamodèle *Core* et qui représente les relations entre les points de vue d'un processus décisionnel basé sur le CWM est étendue par les entités *Equivalence*, *Inclusion*, et *Independence* Figure 4.7 : Cette extension permet d'exprimer la dépendance entre les points de vue des utilisateurs dans un processus décisionnel basé sur le CWM et d'établir une interaction et compréhension mutuelle entre ces utilisateurs.

4.5 SYNTHÈSE

Dans ce chapitre, nous avons défini la notion de point de vue qui se positionne dans une approche orientée-objet avec un formalisme d'ingénierie de connaissance ; De plus, nous avons présenté notre métamodèle point de vue et ses différents concepts en se basant sur la modélisation objet.

Afin d'intégrer les différents points de vue des utilisateurs durant l'exploitation du processus décisionnel, nous avons opté pour une extension du métamodèle CWM vu qu'il ne présente pas la sémantique et les composants nécessaires pour répondre à notre besoin. C'est avec cette perspective que nous avons étendu le métamodèle Core de la couche *Object Model* et le métamodèle *Business Information* de la couche *Foundation* avec les nouveaux éléments du métamodèle *Point de Vue*. Le point de vue servira pour annoter les expériences des utilisateurs et aussi de garder la trace des décisions prises lors de l'exploitation d'un processus décisionnel. Les annotations et les métadonnées capturées seront stockées par la suite dans une base de connaissance pour améliorer la réutilisation et guider les utilisateurs novices dans leurs exploitations.

Vers la fin, en vue de formaliser l'interaction et l'interdépendance entre les différentes analyses lors du processus décisionnel selon différents points de vue, nous avons présenté un ensemble de relations entre points de vue. Nous avons défini ainsi les relations d'*équivalence*, d'*inclusion*, et d'*indépendance* entre deux points de vue. Ces relations nous permettent d'améliorer la coordination, le partage de connaissances et la compréhension mutuelle entre les différents acteurs d'une analyse multipoints de vues, et la réutilisabilité en termes de point de vue des expériences réussies dans un processus décisionnel.

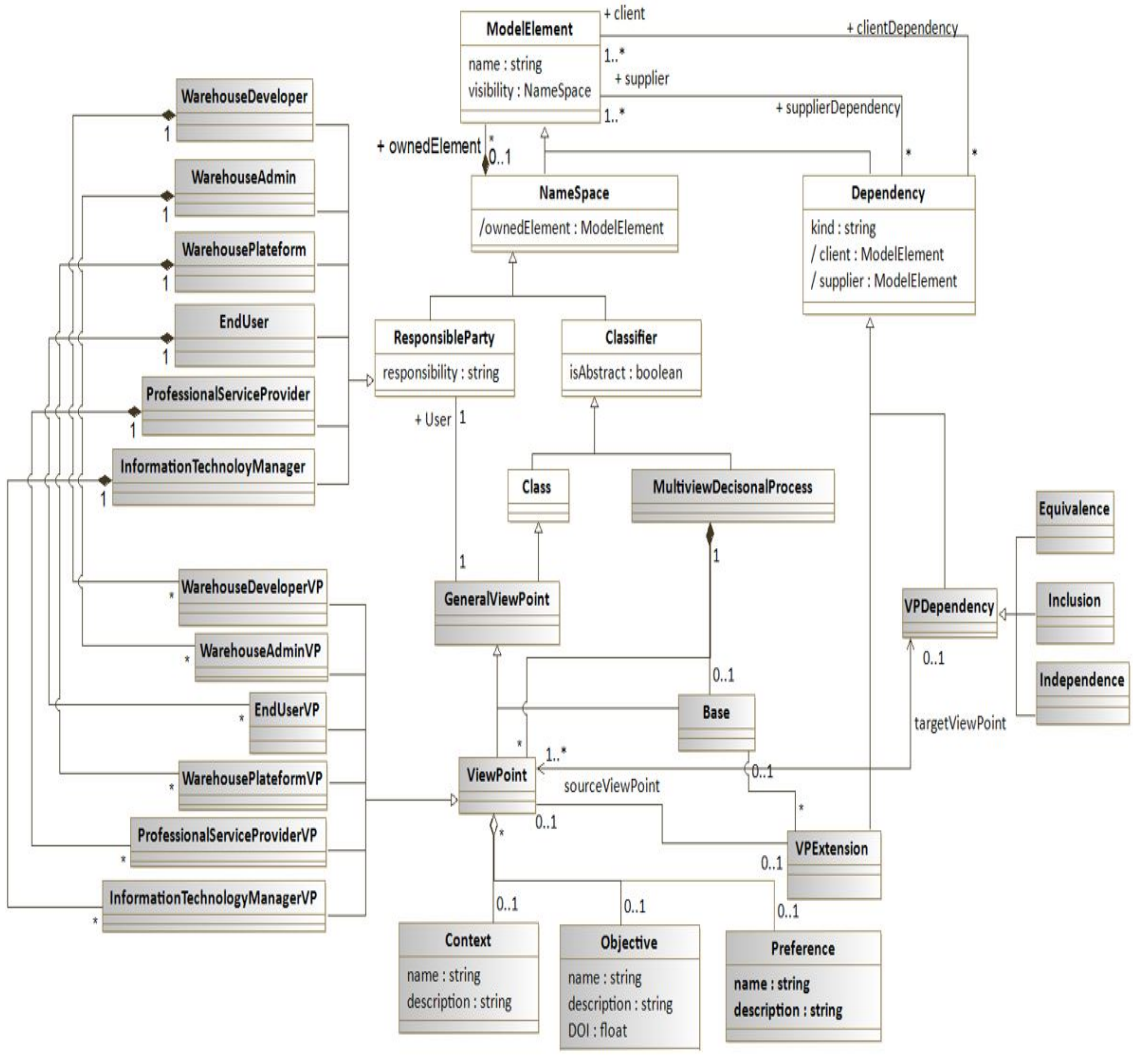

FIGURE 4.7 – Le métamodèle Point de vue étendu par les trois types de relations.

5 LE COMMON CASE REUSE FRAMEWORK (CCREF)

5.1 INTRODUCTION

Après la définition du concept point de vue dans le cycle décisionnel en termes de CWM et la modélisation des différentes relations entre les points de vue, nous envisageons dans ce chapitre de réutiliser les expériences et les annotations des utilisateurs en termes de point de vue en utilisant le Raisonnement à Partir de Cas.

5.2 LE RAISONNEMENT À PARTIR DE CAS

Le Raisonnement à Partir de Cas (RaPC) est un paradigme de résolution de problèmes issue des travaux en Sciences Cognitives et en Intelligence Artificielle, qui consiste à utiliser les solutions de problèmes passés, déjà résolus, dans le but de résoudre de nouveaux problèmes. La réutilisation peut consister en la simple ré-application de la solution précédente, ou bien en l'application d'une solution « adaptée » pour mieux répondre aux spécificités du problème.

5.2.1 *Fondements historiques*

Le RaPC est une approche de résolution de problèmes datant du début des années 1980. Ainsi, Schank (SCHANK 1983)reste le premier à formuler le paradigme de raisonnement basé sur les cas en s'inspirant des approches de Minsky

(Minsky 1975) et (Schank 1983) réalisés à la fin des années 70. En effet, Minsky a défini un réseau de noeuds et de relations entre eux ainsi que la notion de « frame (script, schéma) » qui correspond à une structure remémorée qui doit être adaptée pour correspondre à la réalité d'une nouvelle situation rencontrée. Cependant Schank a repris ces travaux et a supposé que le processus de compréhension soit un processus d'explication qui s'applique d'une manière itérative. De là, Schank reste le premier fondateur du terme « Case-Based Reasoning ». Il introduit à travers le modèle de « mémoire dynamique » un degré de généralité varié connu sous le nom de « MOPS (Memory Organization Packets) » constituant un réseau dense d'expériences. De plus, l'auteur tente d'opérationnaliser le comportement humain et l'optimiser si possible. Dans ce cadre, Gebhardt et al. (Gebhardt et al. 1997) définissent le raisonnement à partir d'expériences comme une façon naturelle de penser caractérisant la réflexion humaine sans doute plus encore que le raisonnement avec des règles. A la fin des années 80, les recherches dans le domaine du RàPC ont réellement commencé à prendre forme (J. Kolodner 1988) (Veloso 1994).

5.2.2 *Communautés en RàPC*

Le positionnement de l'approche RàPC se définit selon le type de méthode développée. On peut la situer dans le domaine de Intelligence Artificielle (IA) (machine learning) ou bien dans le domaine de l'Ingénierie des Connaissances (IC)(Figure 5.1). L'IC qui se place à l'intersection de deux communautés de recherche : l'IA et les sciences cognitives. L'IA est un domaine de recherche permettant d'élaborer des systèmes intelligents. L'IA est la « recherche de moyens susceptibles de doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains » (J.-P. Haton et M.-C. Haton 1989). Les sciences cognitives regroupent un ensemble de disciplines scientifiques dédiées à l'étude et la compréhension des mécanismes de la pensée humaine, animale

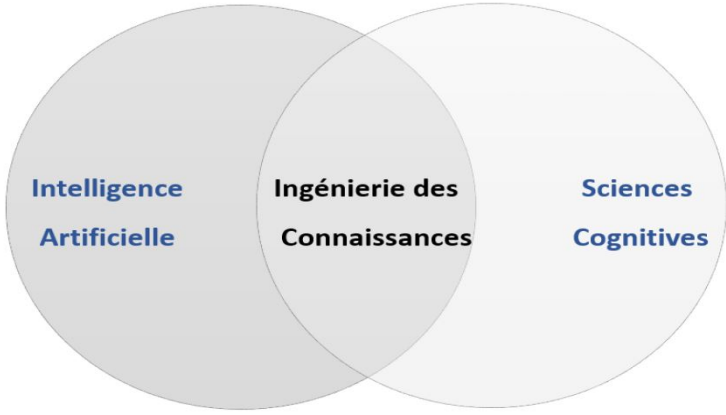

FIGURE 5.1 – L'ingénierie des connaissances à l'intersection de l'Intelligence Artificielle et des Sciences Cognitives.

ou artificielle, et plus généralement de tout système cognitif, c'est-à-dire tout système complexe de traitement de l'information capable d'acquérir, conserver, utiliser et transmettre des connaissances (LE NY 1989).

L'ingénierie des connaissances est le domaine qui correspond à l'étude des concepts, méthodes et techniques permettant de modéliser et/ou d'acquérir les connaissances pour des systèmes réalisant ou aidant les humains à réaliser des tâches se formalisant a priori peu ou pas (CHARLET, ZACKLAD et KASSEL 2000).

Le RaPC est une approche qui utilise un raisonnement par analogie (MILLE, Beatrice FUCHS et HERBEAUX 1996). Dans ce sens, le RaPC a été représenté par un modèle de carré d'analogie.

5.2.3 *Carré d'analogie*

Le carré d'analogie (Figure 5.2) permet d'établir le lien entre la description d'un cas et sa solution (la trace du raisonnement menant à la solution) ; et aussi, les liens entre la description et la solution du cas source de la base de cas et du cas cible qui est un nouveau problème à résoudre (similarité entre deux problèmes) (MILLE, Beatrice FUCHS et HERBEAUX 1996). Dans ce cas-là, la solution du cas cible est adaptée en fonction de la similarité et les descripteurs des cas sources similaires de la base de cas.

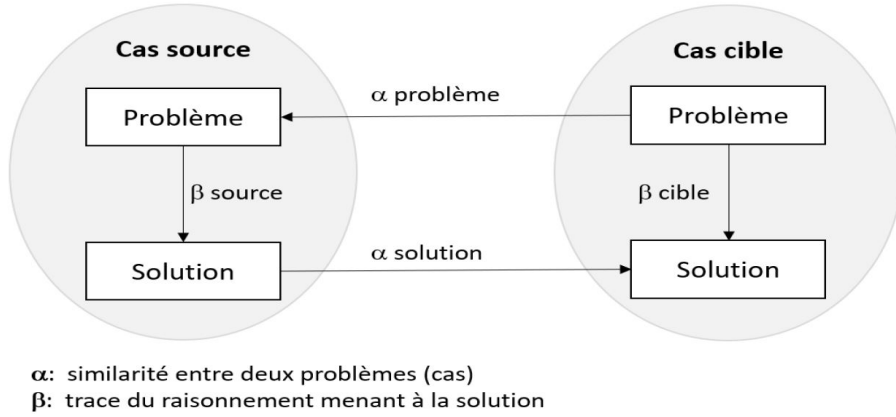

FIGURE 5.2 – Représentation du RàPC par le carré d'analogie. (MILLE, Beatrice FUCHS et HERBEAUX 1996)

— La mesure de similarité α représente la similarité du cas source sélectionné à partir des valeurs de descripteurs du problème cible.
— Les relations de dépendance β entre les valeurs de descripteurs de problème et les valeurs de descripteurs de solution mettent en évidence les descripteurs de solution qui doivent être adaptés. Les descripteurs de so-

lution dépendent des descripteurs de problèmes source qui sont différents des descripteurs de problèmes cible.

En d'autres termes, si une valeur de descripteur source dépend d'une valeur de descripteur de problème cible, une modification de la valeur du descripteur de problème entraînera une modification « analogue » à la dépendance du descripteur de solution correspondant.

Cette connaissance est nécessaire pour la phase d'adaptation. En fonction de ces dépendances et des écarts α constatés à corriger, l'adaptation permet de proposer une solution cible candidate qui pourra être vérifiée par rapport à sa conformité aux dépendances particulières qui pourraient exister entre problème et solution cible.

5.2.4 *Définition et structure du cas*

Un cas est une expérience passée permettant au système RàPC de résoudre des problèmes plus efficacement ou d'éviter les échecs passés. Cependant, selon Kolodner (J. KOLODNER 2014) la définition d'un cas, dans la base de cas, passe par les étapes suivantes :

— La synthèse qui consiste à trouver une structure permettant de satisfaire des spécifications ;
— L'analyse qui consiste à trouver le comportement associé à partir d'une structure particulière ;
— L'évaluation qui consiste à vérifier que le comportement est conforme à ce qui est attendu.

Schématiquement, un cas est composé de deux parties décrivant un problème et la solution qui lui a été appliquée. La description du problème peut recouvrir les objectifs à atteindre, les contraintes sur ces objectifs et le contexte dans lequel le problème a été posé. La solution peut regrouper la description de la solution

effectivement apportée, les étapes qui ont mené à cette solution, son évaluation et les effets de son application (Jaczynski 1998). Cette représentation dépend de la tâche à accomplir, du type des données disponibles et des connaissances du domaine.

5.2.4.1 *Structure du cas*

Généralement, deux types de cas peuvent être distingués : cas source et cas cible. Le cas source est celui dans lequel les parties « problème » et « solution » sont renseignées. En effet, c'est un cas dont on s'inspire pour résoudre un nouveau problème. Le cas source peut aussi contenir une autre partie appelée « information de qualité » (Reinartz, Iglezakis et T. Roth-Berghofer 2000). Cette partie contient des informations sur l'utilisation du cas dans le système. Quant au cas cible, c'est celui qui porte le problème et dont sa partie solution n'est pas renseignée.

Les cas dans une base de cas représentent de nombreux types de connaissances qui peuvent être stockés dans différents formats représentatifs. La représentation du cas diffère selon le type du système RaPC ; Par exemple, les cas pourraient représenter les personnes, les objets, les situations, les diagnostics, les conceptions, les plans ou les décisions, parmi de nombreuses autres représentations. Dans de nombreuses applications de RaPC, les cas sont généralement représentés comme deux ensembles non structurés de paires attribut-valeur qui représentent le problème et les fonctionnalités de la solution (Pal et Shiu 2004). Toutefois, la tâche la plus difficile est de décider ce qu'il faut représenter.

Dans ce but, un ensemble de caractéristiques est extrait ou sélectionné dans la représentation de chaque cas. Ces caractéristiques, appelées Indices déterminent dans quelle situation un cas est applicable et utile (Jaczynski 1998).

5.2.4.2 *Indexation du cas*

Les cas sont organisés dans une mémoire appelée **base de cas**. Afin de faciliter cette organisation et ainsi la recherche du cas le plus approprié au problème posé, il faut désormais les indexer. Il est à noter que lors de la recherche des cas, c'est la partie problème qui va être sollicitée. Or, cette partie est décrite par un ensemble de caractéristiques pertinentes nommées « indices ». Ces indices vont déterminer dans quels contextes et dans quelles situations les cas vont être recherchés et retrouvés pour les proposer au problème rencontré.

Alors, il faut trouver le moyen de bien manipuler ces indices pour une configuration optimale. Pour cela, plusieurs méthodes d'indexation sont proposées : manuelles ou automatiques. Dans le cas des méthodes manuelles, il est supposé que l'objectif d'utilisation des cas, et surtout des circonstances dans lesquelles les cas seront utiles, est déterminé précisément. Toutefois, les méthodes d'indexation sont de plus en plus automatisées.

D'autre part, le choix des indices dépend du domaine d'application. Comme le précise (J. Kolodner 1988), ces indices doivent être :

— Prédictifs afin de jouer un rôle déterminant dans le choix d'une solution pour un nouveau problème ;
— Suffisamment abstraits pour que le cas ait la possibilité d'être utilisé plusieurs fois pour la résolution de plusieurs problèmes ;
— Suffisamment concrets pour que le cas soit reconnu le plus rapidement possible pour la résolution d'un nouveau problème.

Plusieurs méthodes d'indexations sont représentées dans la littérature (Jaczynski 1998) (Trousse 1998) (Jaczynski et Trousse 1997, pour faciliter l'application du RaPC au sein de la classe de problèmes visée.

5.2.4.3 *Base de cas*

Le bon fonctionnement et les performances d'un système de RaPC sont fortement liés à l'organisation de sa mémoire. En effet, la mémoire qui contient tous les cas sources précédemment retenus est appelée base de cas. La base de cas est un élément majeur dans l'indexation et l'organisation des cas afin de pouvoir les retrouver facilement et efficacement. Plusieurs types d'organisation de la base de cas existent (PAL et SHIU 2004). Cependant, lors de la création d'une base de cas, trois points principaux doivent être considérés (MAIN, DILLON et SHIU 2001) :

— La structure et la représentation des cas ;
— Le modèle de la mémoire utilisée pour organiser la base de cas ;
— La sélection des indices qui sont utilisés pour identifier chaque cas.

En effet, lorsque nous évoquons la base de cas, qui représente le cœur du système de RaPC, nous lui associons forcément les cas (dans notre approche il s'agira des cas dédiés à l'exploitation du cycle décisionnel en termes de point de vue). Par la suite, afin de réaliser notre objectif, nous devons passer par les différentes phases du cycle du RaPC.

5.2.5 *Cycle du RàPC*

Le RaPC dispose d'un cycle dont le nombre de phases varie selon les différentes sources bibliographiques. En effet, Fuchs et al. (FUCHS et al. 2006)déterminent dans leur approche trois phases à savoir la remémoration, l'adaptation et la mémorisation. Les premiers auteurs ayant décrit le cycle du RaPC sont (AAMODT et PLAZA 1994) et le composent de quatre phases : la remémoration (ou recherche du cas similaire) (Retrieval), l'adaptation (ou la réutilisation du cas retrouvé) (Reuse), la révision (ou la validation du cas sélectionné) et la mémorisation (ou l'apprentissage). Récemment, Mille (MILLE 2006) a ajouté une phase préliminaire d'élaboration au début du cycle. La Figure 5.3 illustre les cinq phases du cycle RaPC.

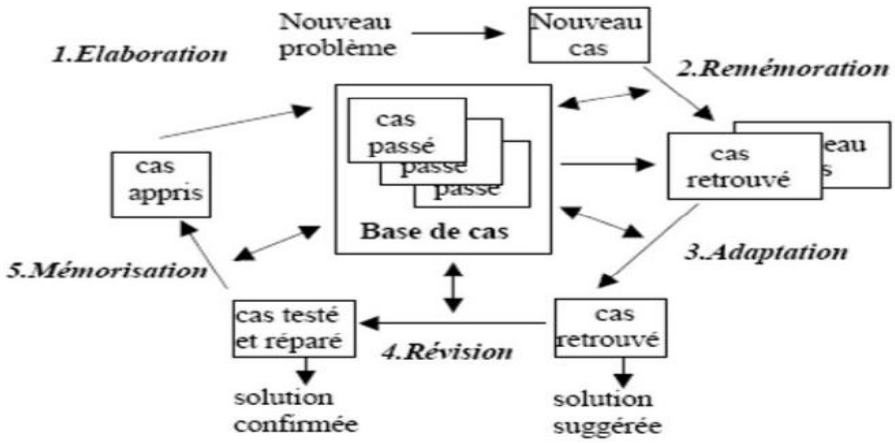

FIGURE 5.3 – Les cinq phases du cycle RàPC . (MILLE 2006)

— La **phase d'élaboration** dans laquelle le cas cible est construit en complétant ou en filtrant la description d'un problème à partir d'une description éventuellement incomplète.
— La **phase de remémoration** des cas sources à partir de la base de cas en recherchant des correspondances entre descripteurs des cas sources et du cas à résoudre (cible) ; de construire une solution au problème courant en se basant sur les cas identifiés dans la phase précédente.
— La **phase d'adaptation** consiste à construire une solution au problème du cas cible inspirée de la solution du (des) cas source(s) le(s) plus similaire(s) ;
— La **phase de révision** de la solution proposée en cas d'une éventuelle solution insatisfaisante, alors il serait possible de la corriger. Dans ce cas, la solution est évaluée dans le monde réel en s'appuyant soit sur l'utilisateur, un expert humain, les connaissances du domaine ou sur un processus automatique ;

— La **phase de mémorisation** consiste à stocker un nouveau cas résolu dans la base de cas si ce stockage est jugé opportun afin d'enrichir la mémoire du système. Et aussi de mettre à jour les éléments du raisonnement en prenant en compte l'expérience qui vient d'être réalisée, et qui pourra ainsi être utilisée dans les raisonnements futurs (Jaczynski 1998).

5.2.5.1 *Phase d'élaboration (Représentation) du cas*

D'après (Fuchs et al. 2006), une première formalisation de la phase d'élaboration est proposée. Elle est définie comme : « une étape qui consiste, à partir de l'entrée du système de RaPC, à construire le problème cible ». De là, cette approche évoque l'importance de cette phase dans le RaPC car elle permet de faciliter d'exécution et d'évaluer les différentes phases du cycle et aussi d'identifier les connaissances qui n'étaient pas prises en compte.

La phase d'élaboration ou de représentation de cas a pour rôle de mettre en place les spécifications du problème à résoudre (cas cible). De ce fait, différents mécanismes sont mis en place pour passer d'un problème souvent mal exprimé à un problème correctement défini. Dans ce sens, Jaczynski (Jaczynski 1998) définit deux étapes pour représenter les cas : le filtrage des cas sur la base des indices, puis la sélection fine des cas. La réalisation de ces filtrages et de sélections s'appuie sur la notion d'index qui permet d'organiser les cas sources au sein de la mémoire. Généralement, la méthode générale consiste à compléter ou filtrer la description d'un problème en se fondant sur la connaissance du domaine pour déduire tout ce qui est possible à partir d'une description éventuellement incomplète, et pondérer les descripteurs en fonction des dépendances identifiées entre les descripteurs du problème cible et les descripteurs de la solution recherchée (Main, Dillon et Shiu 2001).

Les cas sources et le cas cible courant partagent un même formalisme de représentation, généralement formé d'attributs, pour que les opérateurs de mise en correspondance et d'adaptation puissent être appliqués. Dans ce formalisme, Jaczynski identifie trois types d'attributs en fonction de leur rôle pour l'indexation (JACZYNSKI 1998) :

— **Attributs toujours ignorés** : par exemple dans le cadre d'une application donnée, le nom du cas et sa date de création peuvent ne jamais être pertinents comme indices.
— **Indices potentiel** : ces indices, provenant principalement de la description d'un problème, peuvent être utilisés pour déterminer l'utilité d'un cas et ses conditions d'application.
— I**ndices effectifs** ou simplement indices : ces indices déterminent précisément, pour un cas donné, les indices potentiels qui sont effectivement sélectionnés pour guider l'utilisation du cas (situation du cas). Les indices effectifs d'un cas source peuvent être déterminés à priori ou en fonction du cas cible courant.

Dans notre approche, nous tenons compte de cette phase qui représentera le point de départ de notre approche du RaPC. Une bonne élaboration du cas facilite la recherche d'un cas similaire au problème posé pour l'orienter vers une solution adaptable.

5.2.5.2 *Phase de remémoration (Retrieval)*

Cette phase consiste à rechercher dans la base de cas le ou les cas sources les plus similaires à partir de la description de la partie problème du cas cible, qui vont être utilisés pour le résoudre. Cette phase doit permettre d'obtenir la meilleure solution en se basant sur : l'identification des caractéristiques pertinentes du problème, la remémoration et la sélection des meilleurs cas parmi les cas sources. L'exécution de ces tâches est dépendante de la représentation de cas,

de leur indexation et de leur organisation de la base de cas.

Cette phase repose surtout sur le calcul de similarité afin de remémorer des cas similaires au cas cible à partir de la base de cas. Ainsi, le degré de similarité représente la fonction d'utilité/adaptabilité de la solution.

La phase de remémoration représente le domaine de recherche majeur pour les systèmes RàPC. Plusieurs techniques de remémoration sont évoquées dans ce sens (Haouchine 2009). De plus, ces techniques impliquent le développement d'une métrique de similarité qui permet de mesurer la proximité (la similitude) parmi les cas. Dans ce sens plusieurs mesures de similarité ont été établies (Liao, Zhang et Mount 1998) (Santini et Jain 1999).

5.2.5.3 *Phase d'adaptation /réutilisation (Reuse)*

La phase d'adaptation permet de résoudre un nouveau problème par la réutilisation de la solution du cas source, dans le nouveau contexte propre au cas cible. La littérature englobe plusieurs définitions de la phase d'adaptation. Selon (De Mantaras et al. 2005), la phase d'adaptation est le processus proposant une solution à un nouveau problème à partir des solutions appartenant aux cas sources remémorés. De sa part, (Béatrice Fuchs et al. 1999)considèrent l'adaptation comme un plan dont l'état initial est la solution de départ et l'état final est la solution adaptée. Tandis que (Lieber et al. 2004) considèrent que la phase d'adaptation consiste à effectuer un raisonnement par analogie : « sachant que la solution du cas cible est à la solution du cas source ce que le cas cible est au cas source, connaissant le cas source et sa solution ainsi que le cas cible, que vaut la solution du cas cible ? »

Cette phase peut se faire soit via une intervention humaine (manuelle) soit d'une manière automatique à l'aide d'algorithmes, de méthodes, de formules, de règles, etc. (J. KOLODNER 1988).

5.2.5.4 *Phase de validation (Revise)*

Au cours de la phase de révision, la solution proposée à l'issue de la phase d'adaptation sera évaluée selon plusieurs action (MILLE 1999). La phase de révision consiste donc à continuer éventuellement l'élaboration de la solution cible si besoin. Par conséquent, si avec les précédentes actions la solution est jugée insatisfaisante alors elle va être corrigée. Ces corrections peuvent être apportées par :

— L'utilisateur, qui peut donner sa propre évaluation par rapport à la solution fournie via le système de RaPC (KAROUI, Rushed KANAWATI et PETRUCCI 2006) ;
— Un expert humain, qui reflète l'expertise du domaine considéré (CORDIER, 2008).
— Un processus automatique (MALEK et KANAWATI 2004), qui part du principe de l'auto-évaluation en utilisant la base de cas.

Après l'étape de révision, le cas devient une source d'apprentissage importante pour faire évoluer les connaissances mobilisées par le raisonnement. Cependant, Il n'existe pas encore de méthode standardisée pour rendre compte de la tâche de révision dans le RaPC.

5.2.5.5 *Phase d'apprentissage (Retain)*

Cette phase consiste à incorporer ce qui est utile à retenir dans la base de cas et permet de synthétiser les nouvelles connaissances qui vont être réutilisées ultérieurement. Cet apprentissage peut s'effectuer non seulement à partir du succès mais aussi de l'échec dans la résolution du problème cible. Le stockage d'un

nouveau cas permet donc d'enrichir la base de cas permettant l'augmentation de l'expérience du système.

Il peut cependant être utile de garder la « trace » de l'ensemble du cycle avec le détail des corrections faites. Même si on n'a pas encore pu mettre à jour les connaissances du système, cette trace pourra être utilisée pour considérer ce cas comme un modèle pour « corriger » en s'inspirant de cette correction, une nouvelle adaptation qui se ferait avec le même type de similarité.

Les différentes tâches de la phase de mémorisation peuvent être résumées sous une tâche générale nommée : maintenance de la base de cas.

La maintenance de la base de cas est définie comme étant le processus d'affinement de la base de cas d'un système de RaPC. Elle met en œuvre des politiques pour réviser l'organisation ou le contenu de la base de cas afin de faciliter le raisonnement futur pour un ensemble particulier d'objectifs de performance (D. B. Leake et Wilson 1998).

Fort de ce constat, Les différentes phases du cycle de RaPC doivent être en parfaite symbiose. Elles sont interdépendantes et le choix de la méthode d'une phase agit sur le choix des méthodes des autres phases. La conception du système de RaPC se place au centre du processus ce qui influence directement le choix des mesures de similarité, la technologie d'adaptation ainsi que la maintenance de la base de cas.

5.3 CHOIX DE L'APPROCHE RÁPC

Afin de réutiliser l'expérience et les points de vue des utilisateurs au long du cycle décisionnel, nous avons choisi d'utiliser l'approche RaPC. Cette approche a

l'avantage d'être une démarche plus simple à mettre en œuvre que celles basées sur un modèle de domaine, puisqu'elle permet d'éviter les difficultés de modélisation du savoir-faire des experts (complexité des ontologies et des représentions logique...) (JACZYNSKI 1998). Aussi, le RaPC était toujours considéré comme un bon choix pour les domaines qui n'exigent pas de solutions optimales, ou dont les principes sont mal formalisés ou peu éprouvés (RASOVSKA 2006).

Le RaPC vise l'utilisation des connaissances spécifiques et pragmatiques, qui concernent les problèmes précédemment expérimentés, en les capitalisant d'une façon progressive avec le temps ; l'apprentissage sera ainsi incrémental et basé sur les expériences vécues (CORTES ROBLES 2006). En revanche, cette approche présente de nombreux inconvénients, le fait que le RaPC ne trouve pas nécessairement la solution concrète à un problème ; et parfois, juste proposer un ensemble de solutions possibles (DEVÈZE et FOUQUIN 2004) . Aussi, vu la nécessité d'une intervention et d'une mobilisation en continu des experts, lors de la capitalisation progressive des connaissances ; les experts hésitent de partager leurs connaissances, ou bien ils estiment leurs savoir-faire comme leur plus grande assurance (MEKROUD et MOUSSAOUI 2009).

Plusieurs Frameworks ont implémenté différentes architectures RàPC dans différents domaines, à savoir : jCOLIBRI (BELLO-TOMAS, GONZALEZ-CALERO et DIAZ-AGUDO 2004), jCOLIBRI2 (RECIO-GARCIA, GONZALEZ-CALERO et DIAZ-AGUDO 2014) myCBR (T. R. ROTH-BERGHOFER et BAHLS 2008) (K. BACH et ALTHOFF 2012), CAKE (MAXIMINI 2007), CBR*tools (JACZYNSKI et TROUSSE 1997) , et autres (W. A. MARTIN 1978). Dans la section suivante, nous évoquons les travaux qui ont utilisé le RaPC en vue de réutiliser l'expérience des utilisateurs dans plusieurs domaines.

5.4 RÀPC POUR LA RÉUTILISATION ET LA CAPITALISATION DE CONNAISSANCE

Les systèmes de RàPC ont pour fonction de capitaliser l'expertise terrain sous forme de connaissances dans sa mémoire, de pouvoir raisonner dans un domaine à connaissance réduite, de réutiliser des connaissances analogues pour prendre une décision et d'enrichir la mémoire en ajoutant de nouvelles connaissances d'une façon dynamique.

La réutilisation de l'expérience était toujours considérée comme étant un moyen de réduire la reprise des tâches, d'éviter les problèmes, de gagner le temps, d'accélérer les progrès, et aussi de guider les novices qui n'ont pas d'expérience.

Parmi les travaux qui ont utilisé le raisonnement à partir de cas (RàPC) dans le but de réutiliser l'expérience on cite : (Simić, Kurbalija et Budimac 2003) explorent l'idée de la réutilisation de l'expérience acquise depuis des jeux précédents afin d'identifier les déplacements stratégiquement importants pour un joueur donné en utilisant RaPC. Cette approche consiste à retrouver la position d'un jeu précèdent qui est très similaire à celle dans le jeu en cours. Les déplacements joués dans ce jeu précèdent sont adaptés pour générer des nouveaux déplacements pour la situation du jeu en cours. L'approche proposée permet d'identifier rapidement les zones stratégiques les plus précieuses du jeu dans les premières phases du jeu, basé sur la recherche de zones semblables dans un ensemble de parties joués. (Egyed-Zsigmond, Mille et Prié 2003 et (Althoff, Nick et Tautz 1998) présentent un modèle de trace d'utilisation qui permet la collecte et la réutilisation de l'expérience des utilisateurs en utilisant le modèle RaPC ; Le modèle de (Egyed-Zsigmond, Mille et Prié 2003 permet de collecter les actions des utilisateurs lors de l'utilisation d'un programme d'ordinateur dans une représentation homogène, très branchée et bien formalisée des utilisateurs, des procédures et des objets ; tandis que le modèle de (Althoff, Nick et

Tautz 1998) permet une assistance dans des situations non triviales qui exigent de la créativité. Le travail (K. Bach, Szczepanski et al. 2016) se base sur le RaPC pour proposer un système de décision prédictive (selfBACK) dans le but de capturer et de réutiliser les cas de patients afin de suggérer les objectifs et les plans d'activité les plus appropriés pour un patient donné. Les cas dans selfBACK consistent en différents types d'informations qui décrivent un patient et les conseils personnalisés. Dans (Ferrer et Plaza 2016), les auteurs ont utilisé RaPC pour introduire toute forme d'expérience sur le monde réel exprimé sur le web en tant que contenu contribué par l'utilisateur. L'objectif final est de réutiliser l'expérience collective pour aider les nouveaux utilisateurs à prendre une décision plus éclairée en fonction de leurs préférences, qui peuvent être différents des préférences des individus qui ont exprimé leur expérience sur le web. Reuss et al. (Reuss et al. 2016) décrivent un Framework pour l'extraction semi-automatique de connaissances pour le diagnostic à partir de cas dans le domaine de l'aéronef. L'utilisation de RaPC pour le diagnostic peut aider à éliminer les situations de diagnostic ambiguë à l'aide des connaissances expérimentales issues de problèmes résolus avec succès, et de capturer et réutiliser les cas de patients afin de suggérer les objectifs et les plans d'activité les plus appropriés pour un patient donné. (Wienhofen et Mathisen 2016) décrivent un système de décision à partir de cas dans un domaine industriel pour augmenter la vitesse de conversion numérique. La conversion numérique est le processus de découper plusieurs types de matériaux en formes, basé sur un design numérique. Le système vise à faciliter et à automatiser l'apprentissage à partir des expériences passées (c.à.d. les tâches de découpage avec des paramètres spécifiques à un design et à un matériel) dans une société spécifique qui aidera un utilisateur non expérimenté à trouver des cas similaires et à réutiliser les paramètres. L'approche se base sur le raisonnement à partir de cas qui sélectionne automatiquement le cas le plus applicable et le paramètre associé à l'utilisateur afin que tout le potentiel de la machine puisse être utilisé. Dans le domaine médical, (Polese 2014)se base sur le RaPC pour présenter un système d'aide à la décision de santé pour

les directives médicales ; Le système proposé permet d'identifier les directives médicales en exploitant l'historique clinique d'un patient, les protocoles standards, et les historiques d'autres patients. Aussi, (GUESSOUM, LASKRI et LIEBER 2014) consiste à réaliser un système d'aide à la décision pour le diagnostic de la maladie pulmonaire obstructive chronique (COPD). En se basant sur RaPC, le système dans cette approche capture l'expérience des médecins dans le but d'assister les futurs pneumologues. Pour la représentation des cas, les auteurs ont établi en collaboration avec les médecins un ensemble de symptômes (descripteurs) de l'état du patient à son arrivée à l'hôpital sur lequel repose le diagnostic. La phase de récupération (Retrieval) du processus RaPC est basée sur une mesure de similarité définie comme une moyenne pondérée des mesures locales de similarité associées à chaque attribut ; Tandis que la phase d'adaptation repose sur un ensemble de règles pour pouvoir adapter le nouveau cas à la situation voulue.

Dans (TAWFIK et J. L. KOLODNER 2016), les auteurs utilisent le RaPC pour pallier les problèmes rencontrés lors de l'implémentation des systèmes d'apprentissage par problème (PBL). Dans cette approche, le RaPC aide les apprenants à créer des bibliothèques mentales riches et facilement accessibles. Il fournit des recommandations pour tirer parti des expériences d'autres apprenants lorsque les novices manquent d'expérience et aussi d'encourager la réflexion sur l'expérience nécessaire pour favoriser l'apprentissage à partir des expériences de résolution de problèmes. Il suggère également d'autres recherches nécessaires pour comprendre la manière d'intégrer des outils numériques qui faciliteront la gestion des systèmes d'apprentissage par problèmes et aussi de maximiser pleinement les promesses des PBL. Dans (A. MARTIN et al. 2017), les auteurs utilisent RaPC afin de présenter une nouvelle approche pour récupérer les connaissances historiques des projets en tenant compte des différents points de vue et préoccupations des employés travaillant sous différents rôles. La nouveauté dans ce travail est l'introduction d'une nouvelle ontologie basée sur le raisonnement à partir de cas (OBCBR) qui utilise une ontologie d'entreprise pour améliorer l'accès aux

connaissances du projet. Dans cette approche, les caractérisations des cas sont exprimées par une partie définie de l'ontologie de l'entreprise et du domaine. Cette connaissance explicite est utilisée pour la comparaison d'un nouveau cas avec des cas précédentes lors de la phase de récupération (Retrieval) du RaPC. Aussi, dans un contexte de réutilisation d'expériences et d'analyse dans un processus d'ECD, (Hicham Behja 2009)a proposé une plate-forme objet d'ECD qui supporte des analyses multivue basées essentiellement sur l'utilisation des patrons de conceptions.

Dans les travaux cités ci-dessus, le RaPC a été utilisé dans plusieurs domaines pour réutiliser les expériences. Dans le même objectif, nous utilisons également le RaPC afin de réutiliser les annotations et les expériences de l'utilisateur lors d'un processus décisionnel. Dans la section suivante, nous développons le Common Case Reuse Framework en se basant sur le métamodèle point de vue et l'approche RaPC.

5.5 LE CCREF : FRAMEWORK POUR LA RÉUTILISATION D'ANNOTATION DANS LE CYCLE DÉCISIONNEL

En se basant sur notre métamodèle point de vue précédemment décrit (Demraoui, Hicham Behja, Abbou et al. 2014) (Demraoui, Hicham Behja, E. M. Zemmouri et al. 2016) et sur le concept du Raisonnement à Partir de Cas, nous développons dans cette section le CCReF.

L'objectif principal du CCReF est de pouvoir réutiliser les expériences et les annotations des utilisateurs lors d'une exploitation du processus décisionnel en vue de résoudre de nouvelles situations ou problèmes, et aussi de garder la trace des raisonnements et des principales décisions prises par les utilisateurs de CWM lors du processus décisionnel. Ceci permettra d'assurer une meilleure coordination

et compréhension entre les utilisateurs et d'aider les utilisateurs novices durant leurs exploitations.

Nous considérons une base de cas qui contient un ensemble initial de cas. Chaque cas se compose d'une paire (problème, solution) : le problème représente le point de vue de l'utilisateur, tandis que la solution associée représente l'annotation de l'expérience de l'utilisateur qui a exprimé ce point de vue, par exemple : les expériences des scénarios OLAP, d'ETL, datamining ..., etc. Un nouveau cas VP (point de vue) représente la partie problème qui doit être associée à la partie solution qui lui correspond (expérience de l'utilisateur).

La première phase du cycle CCReF consiste à comparer les attributs du VP avec toutes les parties problème des autres cas dans la base du cas selon une fonction de similarité. Le résultat de cette étape est le cas le plus semblable à ce VP (cas récupéré), c'est-à-dire la paire (point de vue, expérience) avec la fonctionnalité la plus similaire.

Ensuite, le cas récupéré devrait être réutilisé pour générer le cas résolu, en combinant sa partie Solution avec la partie Problème du nouveau cas VP. Le cas résolu représente donc une paire (point de vue, expérience). À ce stade, le cas résolu est retourné comme étant une solution proposée, qui pourrait être révisée par des experts. Si la solution proposée est retenue lors de la phase de Révision, elle devient alors une solution confirmée (tested case). Si cela échoue, le cas sera rejeté.

En dernier lieu, la phase Retain (Apprentissage) décide d'inclure ou non la solution confirmée (tested case) dans la base de cas. La phase d'apprentissage dépend de plusieurs critères comme nous avons cité dans la partie état de l'art. Dans notre approche, nous utilisons une valeur de seuil (th) sur la fonction de similarité pour décider le cas à retenir. Si la fonction de similarité renvoie une

valeur supérieure au seuil, le cas est ajouté à la base du cas. La Figure 5.4 illustre notre approche CCReF décrite ci-dessus.

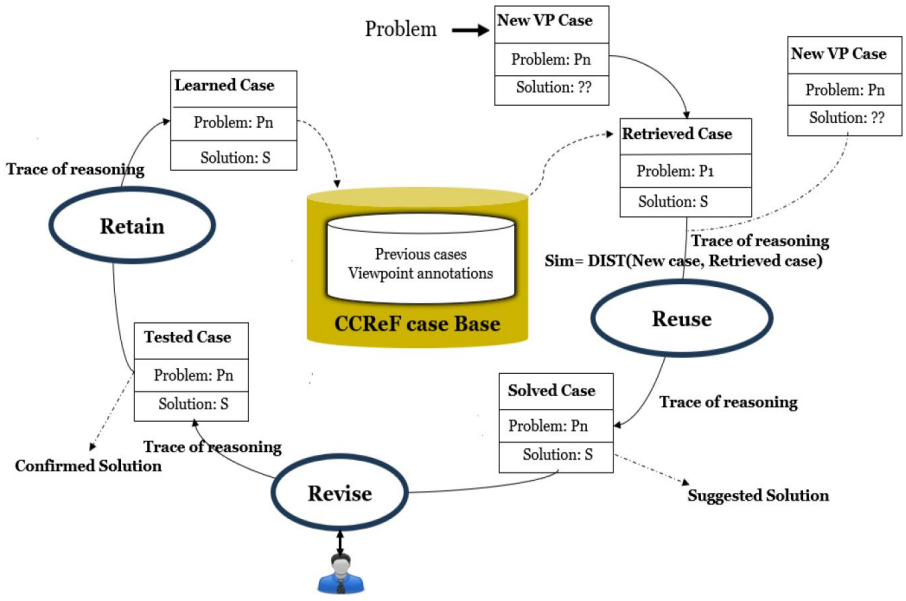

FIGURE 5.4 – Le cycle du Common Case Reuse Framework (CCReF)

5.5.1 *La représentation du cas*

La première phase du cycle CCReF est de construire la spécification du problème à résoudre. La plupart des modèles de RaPC ne comprennent pas de tâche d'élaboration du cas qui est néanmoins toujours implicite.

En effet, avant d'effectuer la remémoration d'un cas, il est nécessaire de créer un nouveau cas, de collecter des informations afin de décrire le problème, de

déterminer les indices qui permettront de rechercher un cas dans la base des cas passés.

La tâche d'élaboration ou de représentation du cas est déclenchée par un agent extérieur (utilisateur, expert, logiciel...), et a pour point de départ une requête qui donne les premières informations connues et contextuelles sur le problème à résoudre. Un nouveau cas est créé et la description du problème est ensuite complétée en collectant d'autres informations. Puis sont déterminés les indices à utiliser pour remémorer un cas passé.

La phase de représentation d'un nouveau cas est composée de deux sous-tâches principales : La première sous-tâche crée un nouveau cas, et la seconde prépare le cas en évaluant les meilleurs indices à utiliser pour la tâche remémoration.

Pour la représentation du cas pour notre contexte du point de vue de l'utilisateur dans CWM, nous avons utilisé une représentation de cas Orientée-Objet (OO), où les cas sont représentés sous forme de collections d'objets décrites par un ensemble de paires attributs-valeurs (MANAGO et al. 1994). Les représentations orientées objet sont appropriées pour des domaines complexes où différentes structures de cas peuvent se produire.

La Figure 5.5 illustre la structure de représentation de cas OO. Comme indiqué précédemment, la classe de cas est divisée en deux parties : Problème et Solution.

La partie Problème capture les attributs point de vue exprimés par un utilisateur de CWM lors une analyse OLAP. Ces attributs sont :

— **VP_id**, l'identifiant du point de vue (Integer)
— **User_id** (Integer), l'identifiant de l'utilisateur du CWM user_role (responsibleParty) qui correspond au rôle des utilisateurs du CWM qui expriment

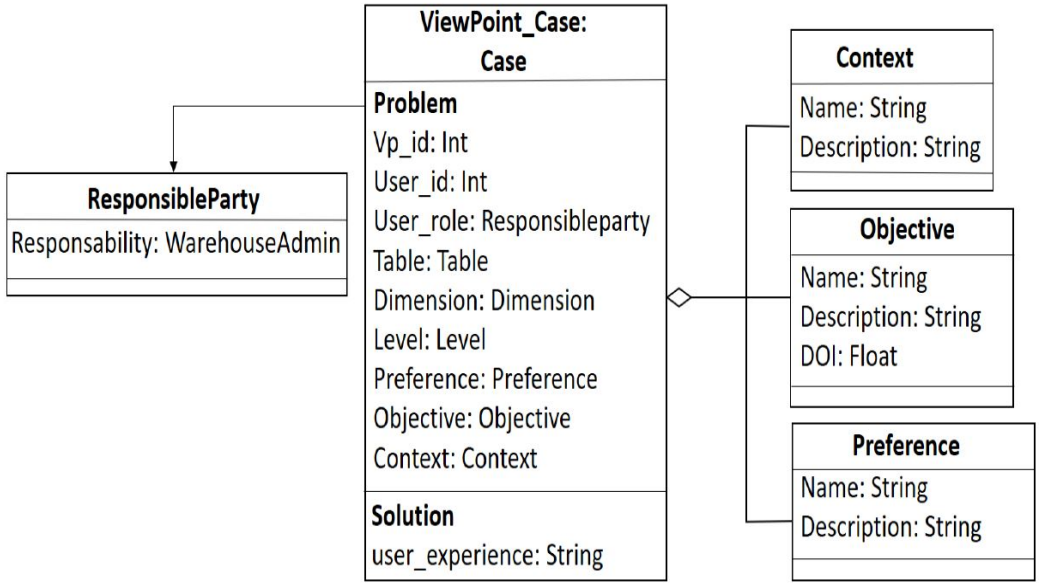

FIGURE 5.5 – La représentation Orienté-Objet du cas point de vue

ce point de vue, à savoir : Warehouse Developper, Warehouse Admin, End User, IT Manager, Warehouse Platform, and Professional Service Provider

— **Préférence** : la préférence de l'utilisateur lors du cycle décisionnel
— **Objectif** (objectif) : l'objectif de l'analyse
— **Contexte** (contexte) : le contexte de l'analyse

Le point de vue est composé d'un Contexte d'analyse décrit par les attributs nom et description ; une classe Préférence caractérisée par un nom et une description, et un Objectif d'analyse décrit par un nom, une description et un Degree Of Interest (DOI). Le degré d'intérêt exprime l'intérêt de l'utilisateur d'une CWM à réaliser cet objectif ; La valeur du DOI est spécifiée par l'utilisateur du CWM. Par exemple,

un utilisateur opère avec des valeurs de l'attribut DOI qui peuvent être comme suit : très bas, bas, moyen, élevé, très élevé. Chaque DOI peut avoir un nombre réel équivalent défini. Par exemple, si les valeurs du DOI sont dans l'intervalle [0 ; 1], le degré d'intérêt moyen correspond à la valeur numérique 0.5, tandis qu'un faible degré d'intérêt correspond à 0,2, etc.

La partie Solution représente l'expérience de l'utilisateur associée à la description du problème ; cette dernière est identifiée par un attribut String : User_experience. L'expérience de l'utilisateur pourrait être représentée par un fichier XMI qui capture la trace de l'utilisateur lors de l'exploitation d'un processus décisionnel en termes de point de vue.

- **Indexation de cas**

La deuxième sous-tâche de la tâche d'élaboration consiste à choisir les indices pertinents permettant de diriger la recherche de cas. Ceci permet d'éliminer des attributs peu pertinents qui risquent de bruiter la recherche.

Les indices d'un cas sont des combinaisons de ses importants descripteurs qui le distingue des autres cas. Lorsqu'un nouveau cas est ajouté à la base de cas, nous lui attribuons des index pour faciliter la récupération. Le problème de l'indexation est le problème de veiller à ce qu'un cas soit accessible à chaque fois que cela est approprié.

Pour notre approche, nous choisissons comme index pour le cas de point de vue : le *contexte* I, l'*objectif* O et *le rôle de l'utilisateur* R (user_role).

5.5.2 *Remémoration du cas(Retrieval)*

La phase de remémoration consiste à sélectionner les cas ayant le plus de chances de faciliter la résolution du problème. La tâche de remémoration choisit, à partir de la base des cas, le cas similaire au problème courant. Dans de nombreux systèmes, cette tâche exploite un modèle d'indexation pour identifier les points d'entrée dans la base de cas, des modèles de similarité et des modèles du domaine afin d'évaluer la ressemblance (Béatrice FUCHS 1997).

La phase de remémoration procède à la sélection progressive d'un sous-ensemble de cas, en commençant de la base de cas intégrale jusqu'à l'obtention d'un seul cas dont la solution sera réutilisée pour le problème courant. Cette phase du cycle CCReF est répétitive et à chaque étape le sous-ensemble de cas sélectionné est raffiné à l'aide de critères plus restrictifs. Chaque étape de raffinement commence par une recherche qui consiste en un appariement suivi d'une évaluation permettant de chiffrer la similarité. Les cas sont alors ordonnés en fonction de l'évaluation de similarité. La sélection choisit un ou plusieurs cas en fonction de cet ordonnancement.

La Figure 5.4 montre que les nouveaux cas sont donnés en tant qu'entrée au système CCReF sous la forme d'une partie de problème (Pn). Afin de trouver une solution à un nouveau cas, la phase de remémoration filtre dans un premier temps les cas en se basant sur plusieurs méthodes (PAL et SHIU 2004), puis dans un deuxième temps elle évalue la similarité des cas sélectionnés.

Dans notre approche, nous employons la méthode des K plus proches voisins (K-NN) (PAL et SHIU 2004) en se basant sur la similarité entre chaque cas source et cas cible, généralement représentée par un nombre réel de l'intervalle [0,1]. Dans la remémoration du plus proche voisin, le cas récupéré est choisi lorsque la somme pondérée de ses caractéristiques (attributs) qui correspondent au cas

à évaluer est supérieure à celle des autres cas dans la base du cas. En d'autres termes, si tous les attributs sont pondérés d'une manière égale, un cas avec n caractéristiques qui correspondent au cas à évaluer sera remémoré au lieu d'un cas qui ne contient que k caractéristiques qui correspondent au nouveau cas, avec k<n.

Lors d'une situation de résolution de problèmes, l'importance des caractéristiques est exprimée en les pesant avec des poids plus fors dans le processus d'appariement des cas. Généralement, les cas remémorés sont les k plus similaires au problème cible. Par ailleurs, les cas récupérés peuvent être ceux dont la similarité avec le problème cible dépasse un seuil prédéfini (th) ; Dans notre cas, nous considérons un seuil th de 0,5.

Pour mesurer la similarité entre les cas, plusieurs approches sont présentes dans la littérature pour différentes représentations de cas (PAL et SHIU 2004). Nous définissons une mesure de similarité locale pour chaque attribut et une mesure de similarité globale qui est calculée comme une moyenne pondérée des similarités locales. Les poids correspondants aux attributs des cas sont assignés par les experts ou les utilisateurs du domaine, ils permettent d'évaluer les degrés d'importance pour chaque variable.

- **Calcul de similarité**

La similarité définit une relation d'ordre entre les cas discriminés par rapport au nouveau cas reflétant leur ressemblance. L'évaluation de similarité prend en compte les caractéristiques des cas avec des degrés d'importance différents. Cette différence d'importance reflète une relation plus ou moins étroite avec la solution. La similarité est définie dans un modèle de similarité contenant notamment une évaluation de l'importance des caractéristiques appelée également pondération. L'importance d'une caractéristique peut ne pas être connue à l'avance et peut

dépendre du contexte d'utilisation.

D'une manière générale, la similarité entre un nouveau cas C^N et un candidat dans la base de cas C^C est calculée par une mesure de similarité globale définie comme la somme des similarités de ses caractéristiques qui le composent (similarité locale) multipliées par leurs poids de pertinence attribué par l'utilisateur :

$$\mathrm{Sim}(C^N, C^C) = \sum_{1}^{n} \left(w_i \ \mathrm{sim}_i(C_i^N, C_i^C) \right) \tag{5.1}$$

Avec :

- C^N : représente un nouveau cas à évaluer
- C^C : représente le cas candidat dans la base du cas
- w_i : représente le poids associé à l'ième attribut de chaque cas
- sim_i : représente la mesure de similarité locale (c'est-à-dire mesure de similarité spécifique pour chaque attribut (descripteur) de cas).

Afin de fournir une représentation de la mesure de similarité, il est donc nécessaire d'assigner à la fois pour chaque attribut du cas à présenter, un poids pertinent et une description adéquate de la mesure de similarité locale.

La valeur attribuée aux poids permet de spécifier l'importance relative de chacun de ces attributs de cas lors l'évaluation de la similarité exprime l'importance des attributs individuels pour l'évaluation de la similarité. Plus le poids associé est grand, plus l'attribut est important avec :

$$w_i \geq 1 \quad et \quad \sum_{i=1}^{n} w_i = 1 \tag{5.2}$$

L'utilisateur affecte manuellement la valeur du poids aux indexes Les valeurs de poids sont affectées à l'index par l'utilisateur d'une façon manuelle. Dans notre approche, nous définissons trois types de mesures de similarité locales pour chaque type d'attribut de l'indice du cas (P, O et R).

En vue de pouvoir calculer la valeur de similarité entre le cas source et le cas cible, il est nécessaire de calculer la valeur de la similarité locale pour chaque descripteur (attribut) de cas. La similarité locale $\mathbf{sim_i}(\mathbf{C_i^N}, \mathbf{C_i^C})$ prends aussi une valeur dans l'intervalle $[0, 1]$, elle calcule la similarité de deux valeurs du même attribut i, l'un appartenant au cas source C_i^N et l'autre appartenant au cas cible C_i^C. Une mesure de similarité locale est symétrique, c'est-à-dire : $\mathbf{sim_i}(\mathbf{C_i^N}, \mathbf{C_i^C}) = \mathbf{sim_i}(\mathbf{C_i^C}, \mathbf{C_i^N})$.

Le calcul de la similarité locale dépend du type de l'attribut. Nous distinguons trois types de similarité locale :

— **Attribut chaîne de caractères** : La similarité pour ce type d'attributs est calculée selon plusieurs méthodes citeCorley. Dans notre approche, nous utilisons le coefficient de Jaccard (JC) Cite NiWATTNAKUL 2013 pour évaluer la similarité entre deux valeurs d'attribut (Équation 5.3).

$$JC = \frac{|S(a) \cap S(b)|}{|S(a) \cup S(b)|} \tag{5.3}$$

$S(a)$ et $S(b)$ corresponds respectivement aux attributs chaînes de caractère dans le cas a et le cas b. un coefficient de Jaccard égal à 1 indique que les deux cas possèdent des valeurs d'attributs identiques et par conséquent ils sont très similaires. Tandis qu'une valeur de coefficient de Jaccard égale à 0 indique que les deux cas ont une similarité minimale.

— **Attribut nominal** : la similarité locale pour les attributs nominaux est calculée comme suit :

$$\text{sim}(C^N, C^C) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

C^N et C^C désignent respectivement le nouveau cas (cas source) et le cas candidat (cas cible). Les attributs i et j désignent respectivement l'attribut dans le nouveau cas et celui dans le cas candidat. Par exemple, l'attribut "Warehouse Developer" et "Warehouse Admin" vont être mesurés par ce type de similarité.

— **Attribut numérique** : la mesure de similarité locale pour ce type d'attribut est définie selon la formule de la distance Euclidienne (Équation 5.4) :

$$\text{Sim}(C^N, C^C) = 1 - \text{Dist}(C^N, C^C) = 1 - \sum_{i=1}^{n} \text{dist}(C_i^N, C_i^C) \tag{5.4}$$

— C^N et C^C désignent respectivement le nouveau cas (cas source) et le cas candidat (cas cible).
— w_i est le poids assigné pour chaque attribut **i**, où : $i = 1, 2, \ldots, n$ avec n le nombre d'attribut dans le cas.
— C_i^N et C_i^C désignent respectivement l'attribut numérique dans le nouveau cas (cas source) et celui dans le cas candidat (cas cible)
La distance normalisée $dist(C_i^N, C_i^C)$ est souvent représentée par :

$$\textbf{dist}(C_i^N, C_i^C) = |C_i^N - C_i^C|$$

si $\mathbf{C_i^N} = \mathbf{C_i^C}, dist(C_i^C, C_i^N) = 0$ sinon $\text{dist}(C_i^N, C_i^C) = 1$
— Quand la **DIST** $= \mathbf{0}$, la similarité SIM atteint la valeur maximale 1, ceci signifie que les deux cas sont identiques.
— Quand la **DIST** $= \mathbf{1}$, la similarité atteint la valeur minimale 0, c'est à dire que les deux cas sont complétement différents.

L'évaluation de l'indice Contexte (C) consiste à comparer la valeur de l'attribut chaîne de caractère du nouveau cas du point de vue à analyser et la partie problème de chaque cas dans la base de cas. Nous assignons un poids de $w = 0,1$ au Contexte en considérant que l'évaluation de cet indice n'affecte pas directement

le choix du cas dans la base de cas.

L'évaluation de l'Objectif permet de calculer la valeur de similarité de ses attributs (name, description, and DOI) dans le nouveau cas et des attributs qui leurs correspondent dans la partie problème de chaque cas de la base de cas. Nous attribuons à l'Objectif un poids de $w = 0.6$, comme étant l'attribut qui représente l'important critère pour sélectionner les cas similaires. La similarité pour les attributs nom (name) et description est calculée par le coefficient de Jaccard présenté précédemment, tandis que la similarité pour l'attribut degré d'intérêt DOI est calculée selon la distance Euclidienne (Équation 5.3). Deux objectifs du cas point de vue sont similaire s'ils possèdent le même degré d'intérêt. En conséquence, les cas sélectionnés de la base seront ceux qui ont une similarité qui dépasse un certain seuil, qu'on l'a défini à $0,5$.

L'évaluation de l'indice User_Role (R) consiste à comparer les valeurs des attributs nominaux du nouveau cas du point de vue à analyser et la partie problème de chaque cas dans la base de cas. On assigne un poids élevé à l'indice User_Role $(w = 0,3)$, comme étant l'indice qui différencie le rôle des utilisateurs du cycle décisionnel et influence le choix des cas.

5.5.3 *La réutilisation / adaptation*

La réutilisation de la solution du cas remémorée représente les différences entre le cas passé et le cas actuel et la partie d'un cas remémoré qui pourrait être transféré au nouveau cas, ce qui implique que le cas le plus similaire est retenu comme étant la meilleure solution. Par conséquent, la partie solution du nouveau cas sera la partie solution (S) - c'est-à-dire l'expérience de l'utilisateur candidate

- du cas récupéré le plus similaire selon la Équation 5.2.

La réutilisation du cas remémoré s'avère simple quand le nouveau problème est identique au cas remémoré de la base de cas. Dans le cas contraire, une phase d'adaptation est nécessaire.

Dans la plupart des situations une phase d'adaptation est indispensable car le cas remémoré n'est jamais strictement identique au nouveau cas. Elle consiste à modifier la solution du cas remémoré pour prendre en compte les différences entre spécifications de problèmes. Généralement, la solution du cas remémoré est copiée et certaines parties sont substituées en fonction des spécificités du nouveau problème. Beaucoup de systèmes se contentent d'une recopie simple de la solution du cas remémoré (Lenz et al. 1996), ou d'une composition des solutions de plusieurs cas remémorés dans différents cycles de raisonnement distincts. L'adaptation ne change pas radicalement la solution, mais en modifie certaines parties et peut la réorganiser.

Dans notre cas, nous utilisons la technique de ré-instanciation pour adapter le cas remémoré. Dans cette technique la solution du nouveau problème est copiée du cas remémoré et utilisé directement.

Le cas résolu est ensuite retourné comme étant la solution proposée (Figure 5.5) et peut être révisé par des experts. De cette façon, l'expert décide si la solution est adaptée au cas cible et pertinente pour le domaine sous-jacent. Sinon, le cas résolu est rejeté. Les cas pertinents peuvent ensuite faire partie de l'état initial de la base de cas. Le retour des experts n'est pas obligatoire mais nécessaire pour améliorer la performance du raisonnement et pour déterminer la valeur de seuil selon le nombre de cas dans la base de cas à un moment donné.

5.5.4 *Phase d'apprentissage*

À ce stade, le system CCReF a comparé le nouveau cas avec la partie problème de tous les cas dans la base de cas et bien évidemment a trouvé une solution en termes de similarité d'attribut présentée ci-dessus. En outre, la solution a été révisée et reconnue valable par les utilisateurs experts - c'est-à-dire, c'est une solution confirmée (tested case). La prochaine étape consiste à décider si le cas testé sera ajouté à la base du cas ou non.

Dans ce cas, nous définissons une valeur de seuil (th) pour la fonction de similarité (DIST). La valeur de seuil détermine si un nouveau cas sera retenu ou non (learned case) dans la base du cas. Si la fonction de similarité renvoie une valeur supérieure au seuil, le cas est ajouté à la base de cas comme un nouveau cas significatif, sinon le cas sera rejeté.

A cette phase, si l'utilisateur exprime sa satisfaction des résultats obtenus suivant la description donnée au cas point de vue, le cas est enregistré et pourra être utilisé comme une expérience, sinon, le cas est enregistré comme exception, et on revient à la première étape du cycle CCReF.

5.6 SYNTHÈSE

Dans ce chapitre nous avons développé leCommon Case Reuse Framework (CCReF). En se basant sur notre métamodèle point de vue précédemment décrit et sur le concept du Raisonnement à Partir de Cas.

L'objectif principal du CCReF est de pouvoir réutiliser les expériences et les annotations des utilisateurs lors d'une exploitation du processus décisionnel en vue de résoudre de nouvelles situations ou problèmes, et aussi de garder la trace des

raisonnements et des principales décisions prises par les utilisateurs de CWM lors du processus décisionnel. Ceci permettra d'assurer une meilleure coordination et compréhension entre les utilisateurs et d'aider les utilisateurs novices durant leurs exploitations.

6 VALIDATION DE NOTRE APPROCHE DE POINT DE VUE VIA UNE ETUDE DE CAS

6.1 INTRODUCTION

Dans cette section nous validons notre approche de point de vue via l'exemple d'implémentation d'un entrepôt de données pour un environnement d'analyse de publications scientifiques pour un laboratoire de recherche.

L'implémentation de cet environnement décisionnel se base sur le CWM, le standard de l'OMG destiné à l'intégration des outils d'entreposage de données et du business analyse, afin d'assurer une meilleure intégration et interopérabilité lors de l'échange et le partage des métadonnées entre les différents composants du processus décisionnel.

6.2 VUE FONCTIONNELLE DU SYSTÈME

6.2.1 *Les acteurs*

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié, autrement dit un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données (AUDIBERT 2009). Dans le cadre de notre étude nous avons

distingué six acteurs principaux qui interagissent lors des différentes étapes du système décisionnel :

— **Développeur d'entrepôts de données** (WarehouseDevelopers) : responsable du développement de différents composants fonctionnels des applications des entrepôts de données, y compris l'extraction des programmes des systèmes sources, les applications ETL, les fonctions de nettoyage des données, les fonctions de gestion du système, par exemple l'automatisation des charges, les fonctions d'acquisition de données et d'autres.
— **Administrateur d'entrepôts de données** (WarehouseAdministrators) : responsable de l'intégration et de la coordination des métadonnées et des données à travers les différentes sources de données ainsi que la gestion des sources de données, la conception de la base de données physique, l'exploitation, la sauvegarde et la récupération, la sécurité, les performances et le réglage.
— **Utilisateur final** (EndUser) : dont sa tâche consiste à exploiter l'entrepôt de données par des rapports et des requêtes.
— **Vendeurs d'outils et de plateforme d'entreposage de données** (warehouse platform and tool vendors) : responsable de la construction, l'exécution de l'architecture de l'entrepôt de données et le contrôle de la gestion des données.
— **Fournisseur de services professionnels** (professional service providers) : responsable de la configuration et de la gestion des ressources matérielles et logicielles requises par un entrepôt de données.
— **Gestionnaire de la technologie** de l'information (information technology managers) : responsable de l'implémentation et de la maintenance de l'infrastructure technologique d'une organisation. Il surveille les besoins opérationnels de l'organisation, recherche des stratégies et des solutions technologiques en vue de construire un système rentable et efficace pour atteindre ces objectifs.

6.2.2 *Diagramme de cas d'utilisation*

Le diagramme de cas d'utilisation décrit les grandes fonctions d'un système du point de vue des acteurs, mais n'expose pas de façon détaillée le dialogue entre les acteurs et les cas d'utilisation. Bien que de nombreux diagrammes d'UML permettent de décrire un cas, il est recommandé de rédiger une description textuelle, car c'est une forme souple qui convient dans bien des situations (AUDIBERT 2009).

— **Identification des cas d'utilisation**

Un cas d'utilisation est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service (AUDIBERT 2009). Diagramme de cas d'utilisation par acteurs

— Cas d'utilisation pour l'acteur Adminstrateur et l'entrepôt de données(*WarehouseAdmin*).

L'administrateur de l'entrepôt de données a plusieurs tâches (Figure 6.1), on cite : le contrôle de l'entrepôt, la gestion et le transfert de données, le traitement des conflits des bases de données, et la conception logique et physique de l'entrepôt.

— Cas d'utilisation pour l'acteur Développeur de l'entrepôt de données

Le développeur de l'entrepôt de données (*WarehouseDeveloper*)(Figure 6.2) est chargé du dévelopement de plusieurs composants fonctionnels de l'entrepôt de données, y compris l'implémentation des applications ETL

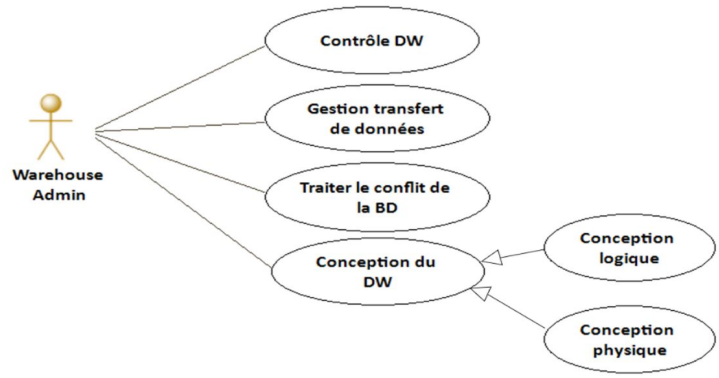

FIGURE 6.1 – Diagramme de cas d'utilisation pour le WarehouseAdmin.

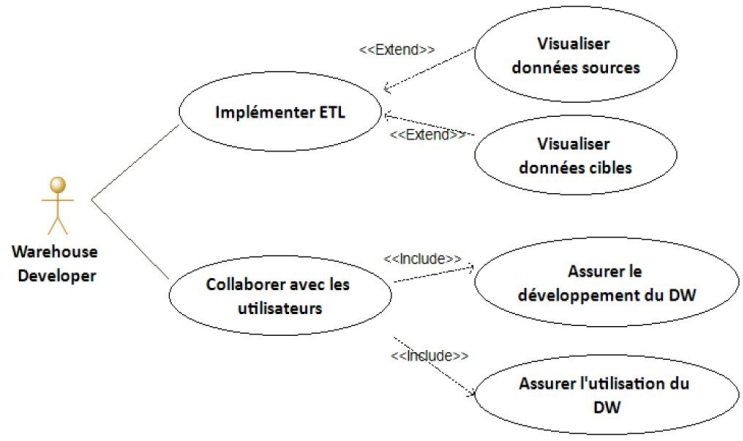

FIGURE 6.2 – Diagramme de cas d'utilisation pour le *WarehouseDeveloper*

. . .,comme il collabore avec les autres acteurs du cycle décisionnel afin d'assurer la bonne utilisation et dévelopement de l'entrepôt de données.

— Cas d'utilisation pour l'utilisateur final (*EndUser*)

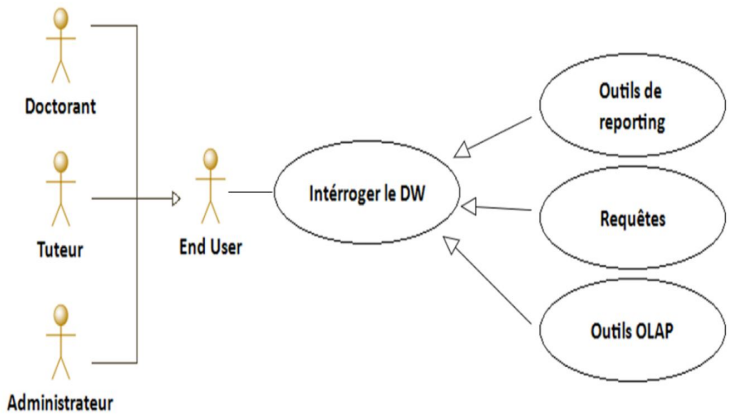

FIGURE 6.3 – Diagramme de cas d'utilisation pour le *EndUser*

La tâche fondamentale de l'utilisateur final (Figure 6.3) est l'exploitation de l'entrepôt de données via les des requêtes ou des rapport. Dans notre étude de cas, l'utilisateur final peut être un doctorant, un tuteur ou bien un administrateur.

— Cas d'utilisation pour le fournisseur professionnel du service

Le fournisseur professionnel de service (*ProfessionalServiceProvider*) (Figure 6.4)se charge de la configuration et de la gestion des ressources matérielles et logicielles requises par un entrepôt de données, il participe à l'amélioration de la productivité de l'entrepôt et il fournit le conseil techniques au clients.

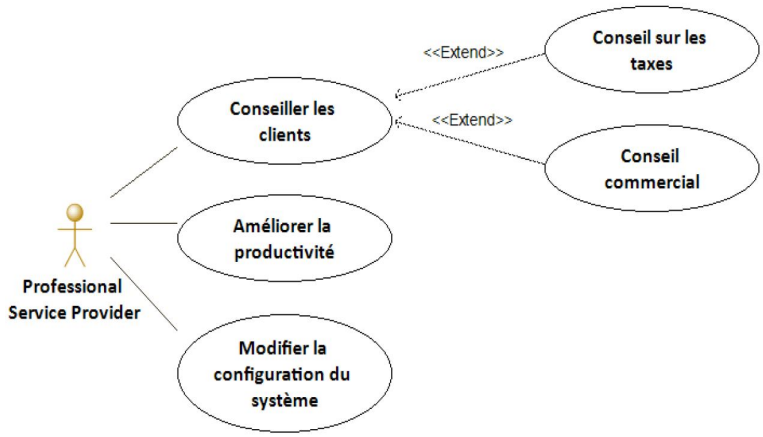

FIGURE 6.4 – Diagramme de cas d'utilisation pour le *ProfessionalServiceProvider*

— Cas d'utilisation pour le vendeur d'outils et plateformes d'entreposage de données

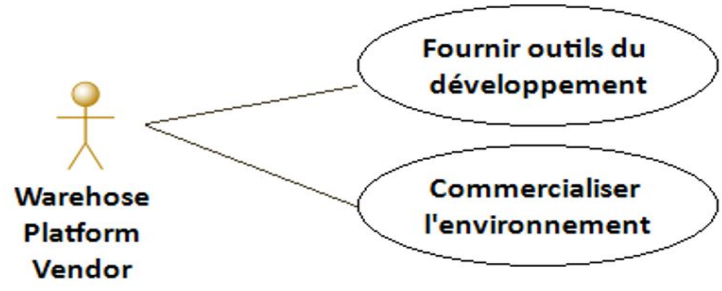

FIGURE 6.5 – Diagramme de cas d'utilisation pour le *WarehousePlatformVendor*

Le vendeur d'outil d'entrepôsage de données *(WarehousePlatformVendor)* (Figure 6.5)fournit les outils nécessaires pour construire et commercialiser l'environnement de l'entrepôsage de données.

— Cas d'utilisation pour le fournisseur professionnel du service La res-

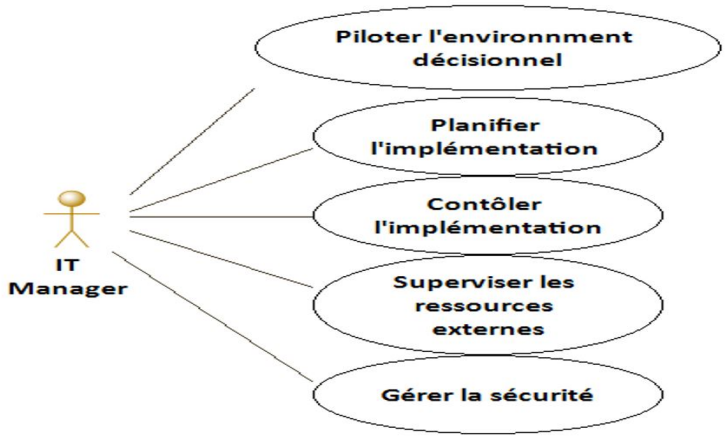

FIGURE 6.6 – Diagramme de cas d'utilisation pour le *ITManager*

ponsabilité du fournisseur de services professionnels (*ITManager*) (Figure 6.6) est de configurer et gérer les ressources matérielles et logicielles requises par l'entrepôt de données. Il pilote, planifie et contrôle l'environnement décisionnel, comme il se charge de la supervisions des ressources externes et de la gestion de sécurité au sein de l'entrepôt.

6.3 LES PAQUETS CWM UTILISÉS

Le *WarehouseAdministrator* contrôle l'environnement décisionnel et gère le transfert des données. L'administrateur d'entrepôt traite les conflits relatifs à la base de données du DW, et surveille le fonctionnement de l'entrepôt de données en s'assurant que les étapes ETL sont exécutées dans les délais qui leur sont impartis. De ce fait, les différentes couches et paquets du CWM sont impliqués dans le scénario de l'administration de l'entrepôt de données (Figure 6.7), à savoir : la couche *Management* (les paquets *Warehouse Process* et *Warehouse Operation*), la couche *Analysis* (les paquets : *Transformation, Data Mining, Information Visualization* et Business Nomenclature), et la *couche Fondation* avec son paquet *Software Deployment*.

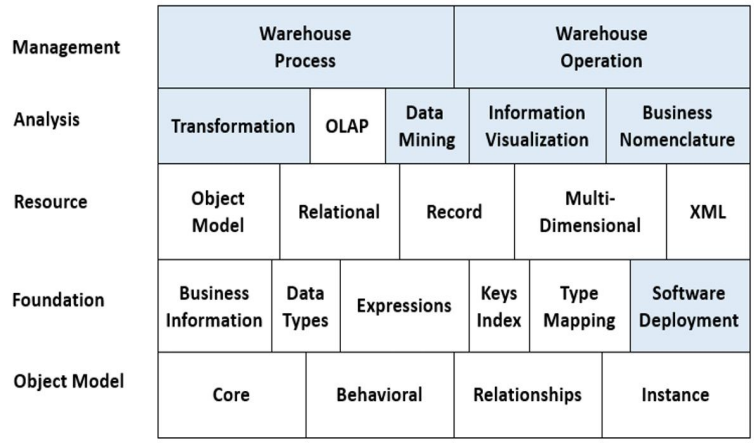

FIGURE 6.7 – Les paquets du CWM utilisés pour le scénario d'administration d'entrepôt de données

Le *Warehouse Developer* est responsable du développement de différents composants fonctionnels des applications des entrepôts de données, y compris l'ex-

traction des programmes des systèmes sources, les applications ETL, les fonctions de nettoyage des données, les fonctions de gestion du système, par exemple l'automatisation des charges, les fonctions d'acquisition de données et d'autres.

A titre d'exemple, un scénario ETL commence par la définition du modèle de *Transformation* de CWM pour le mouvement des données source vers les données cible. Les paramètres des données source, des données cibles et de la logique de transformation sont des valeurs assignées par le modèle. Les paramètres des données source dépendent des types de ces derniers qui sont présents dans les paquets Orienté-objet, Relationnel, Orienté-Enregistrement, Multi-Dimensionnel, ou bien XML de la couche Resource. Les paramètres des données cible sont choisis de la même façon. Les paramètres de la logique de transformation comprennent l'identification d'un composant de transformation et des sources de données et des cibles de données. Le composant de transformation est une méthode composée d'une large hiérarchie de composants (outils commerciaux, bibliothèques commerciales, scripts personnalisés).

Un processus ETL est réalisé par un certain nombre de composants définis à travers plusieurs packages CWM (Figure 6.8), à savoir le *Software Deployment* de la couche *Foundation*, les paquets de la couche *Resource*, les paquets *Transformation* et *Business Nomenclature* de la couche *Analysis*, et les paquets de la couche *Management (Warehouse Process et Warehouse Operation)*.

Un développeur du CWM data warehouse peut lancer un processus ETL comme une opération planifiée consistant en un certain nombre d'étapes de transformation exécutées en séquence. Par exemple, la première transformation consiste à extraire et à filtrer les données à partir d'un certain nombre de données source. Une deuxième transformation nettoie, combine ou réduit les données et les stocke dans un format normalisé dans une base de données relationnelle de l'entrepôt. Une troisième transformation sélectionne certaines lignes de la base de données relationnelle et charge leurs valeurs dans les cellules d'entrée d'une

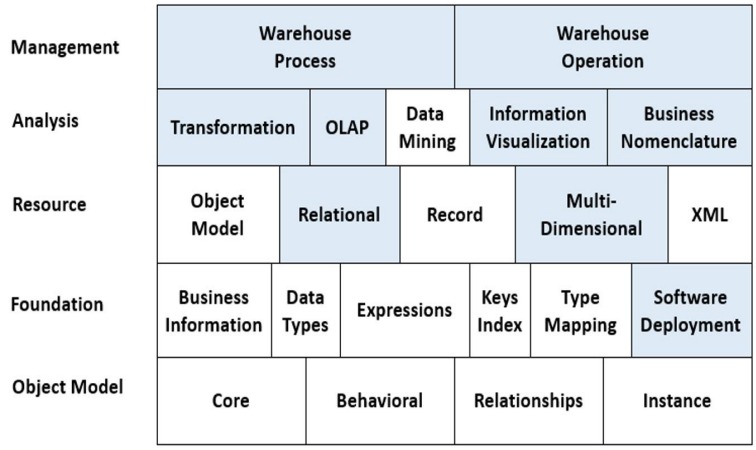

FIGURE 6.8 – Les paquets du métamodèle CWM utilisés pour le scénario de développement de l'entrepôt de données

base de données multidimensionnelle. Finalement, le processus CWM d'entreposage de données pourrait indiquer à la base de données multidimensionnelle de recalculer ses cellules agrégées en fonction des nouvelles données d'entrée.

En outre, un Utilisateur Final (End User) d'un entrepôt de données s'engage dans une session analytique avec une base de données multidimensionnelle ou relationnelle à travers la couche OLAP. L'utilisateur navigue dans les cubes et les dimensions et lance de manière sélective les requêtes OLAP. À un moment donné, l'utilisateur décide d'analyser (drill-down) une valeur consolidée à des niveaux inférieurs de détail dans une hiérarchie OLAP, éventuellement jusqu'à la valeur la plus profonde du plus bas niveau dans la hiérarchie.

En tirant profit de la capacité inhérente de CWM à préserver le lignage de données, l'utilisateur peut afficher les détails opérationnels, qui ont formé la (les)

valeur(s) d'entrée. Les sources de données originales peuvent être identifiées à partir du scénario du Warehouse Developer qui a enregistré la production de la (les) valeur (s) d'entrée.

Les paquets du CWM utilisés lors du scénario OLAP sont représentés par la Figure 6.9 :

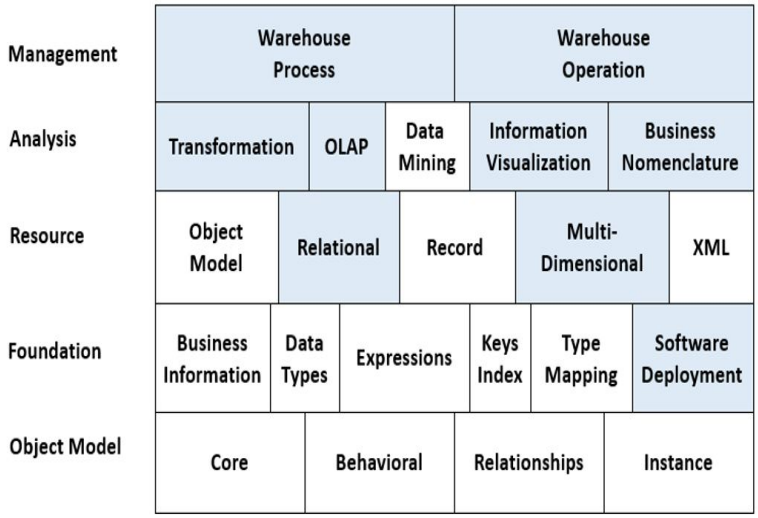

FIGURE 6.9 – Les paquets du métamodèle CWM utilisés pour le scénario OLAP

6.4 VUE STATIQUE DU SYSTÈME : DIAGRAMME DE CLASSE

Le diagramme de classe ci-dessous (Figure 6.10) représente la modélisation multidimensionnelle de notre exemple, la classe Publication représente la table de Fait que l'on a choisi à observer selon plusieurs dimensions : temps (Time), Author (Auteur), Support, Mot clé, et Document. Pour la dimension Temps, nous

nous intéressons uniquement aux années, qui représenteront le plus bas niveau de granularité. Aussi, il est possible de regrouper les années en Période. La dimension Author contient deux niveaux d'hiérarchie, qui permettent de regrouper les auteurs selon leurs statuts (Professeur, Tuteur, Doctorant, ...). La dimension

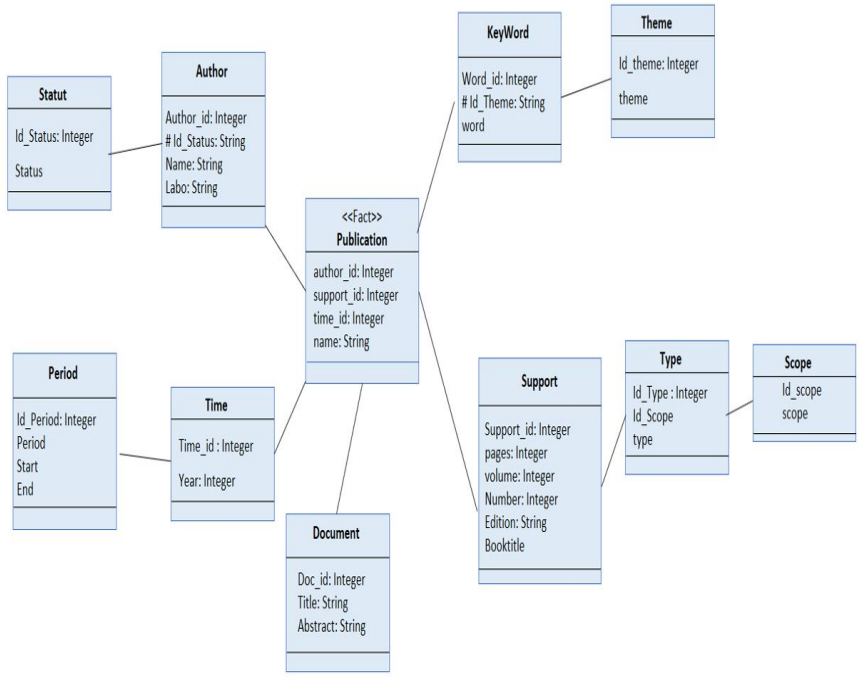

FIGURE 6.10 – Représentation multidimensionnelle de la table Publication et ses dimensions

Support spécifie le nom de la conférence, du journal, du livre ainsi que le volume, le nombre de pages, etc. Chaque support peut être agrégé selon son type (conférence, revue, thèse, ...) et selon sa portée (nationale ou internationale). La dimension Support a trois niveaux : le Support, le Type et la Portée qui contient

deux membres : international et national. Enfin, les dimensions Document et KeyWord peuvent être regroupées en thème.

6.5 CONCEPTION TECHNIQUE

Pour implémenter notre approche, nous utilisons l'outil de méta modélisation EMF puisqu'il utilise UML comme langage de méta-modélisation et à cause de sa facilité d'utilisation par rapport aux autres outils de méta-modélisation (voir chapitre 1). Les itérations du cycle de développement d'EMF se présentent selon les niveaux de la méta modélisation présentés dans le chapitre 1 (Figure 6.11). Les étapes de création d'EMF s'illustrent dans la Figure 6.12. Les paquets du mé-

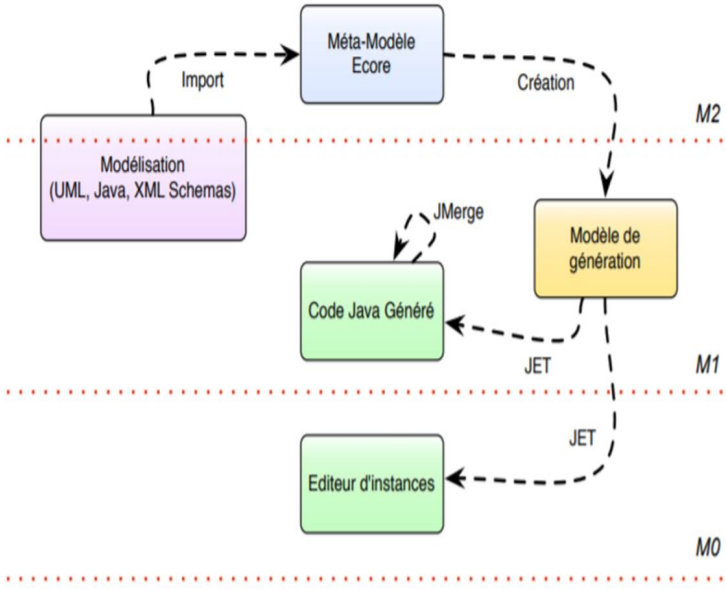

FIGURE 6.11 – La représentation des composants EMF selon l'architecture de méta-modélisation

tamodèle CWM sont importés, sous forme d'un modèle UML, XML Schema ou bien en modèle JAVA, en vue de construire le modèle Ecore d'EMF qui permettre de générer automatiquement un éditeur graphique pour l'édition des modèles sous forme d'arborescence (Figure 6.13). Chaque nœud de l'éditeur représentera

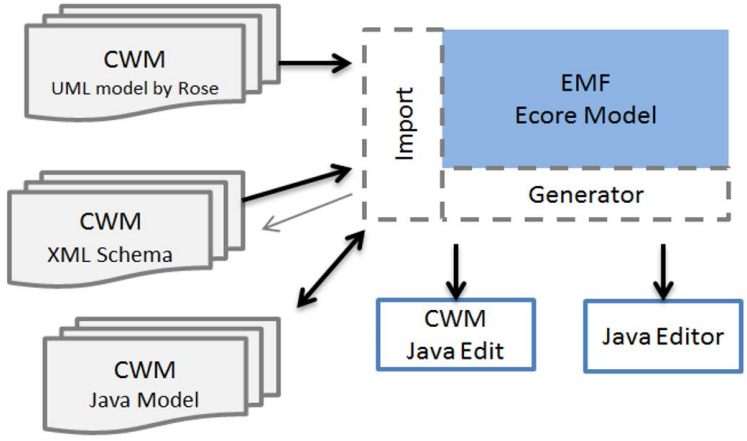

FIGURE 6.12 – Les différentes étapes d'EMF

une instance d'une méta-classe. Ensuite, la génération des interfaces de manipulation des modèles consiste à fournir des interfaces graphiques génériques pour manipuler des modèles. Enfin, la génération de code, considérée comme notre objectif principal, améliore la productivité de développement d'application par l'automatisation de la génération de code à partir du modèle. Effectivement, une fois le modèle créé « quelques clics » suffisent à cette génération. Le métamodèle CWM.ecore se génère à partir de l'importation des modèles CWM (Rational Rose, XML schéma, ou bien JAVA).

Cette génération de code est paramétrée à l'aide d'un métamodèle appelé *GenModel*. Le modèle de génération est construit automatiquement à partir du

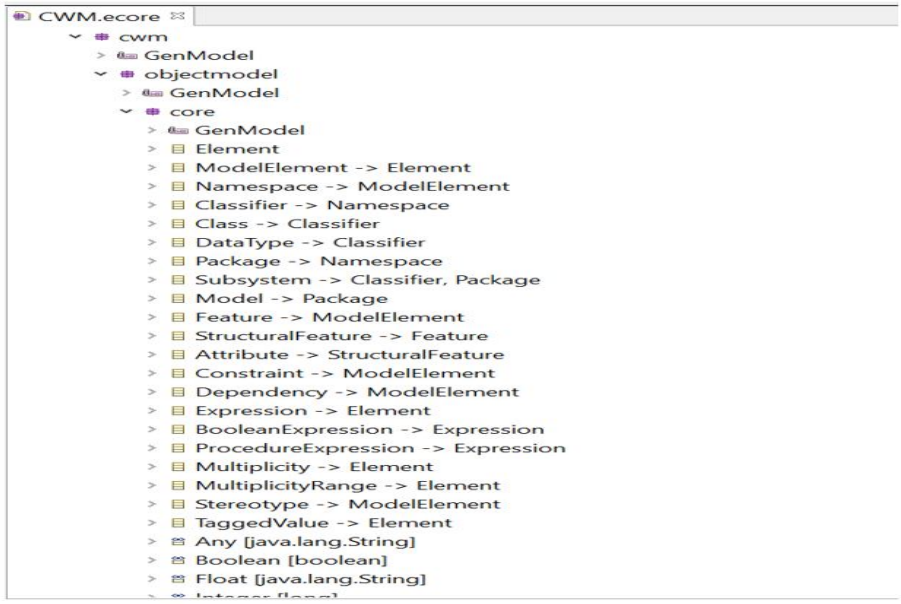

FIGURE 6.13 – Création du CWM.ecore

modèle *Ecore* en associant un élément de paramétrage correspondant à chaque élément du modèle. Les informations détenues par ces éléments de paramétrage concernent différents aspects qui ont un impact sur la génération : règles de nommage, localisation des fichiers générés, mode de visualisation et d'édition, etc.

La génération des modèles *Ecore* et *Genmodel* nous permet d'étendre le métamodèle CWM par les classes et les associations du métamodèle Point de vue définit dans le chapitre 4. L'extension concerne le métamodèle *Core* de la couche *Object Model* et le métamodèle Business Information de la couche *Foundation*. L'extension de ces deux métamodèles est illustrée dans la Figure 6.14.

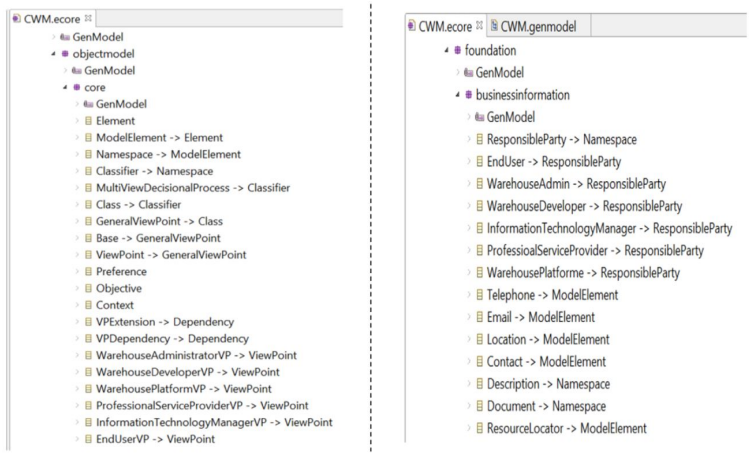

FIGURE 6.14 – L'extension des métamodèles Core et Business Information par les classes du métamodèle Point de Vue

Le métamodèle Core est étendu par les classes : *GeneralViewPoint*, *Multiview-DecisionalProcess*, *Base*, *ViewPoint* qui se compose du *Contexte*, *Objectif* et *Preference*, *VPExtension*, *VPDependency*, *Warehouse PlatformVP*, *ProfessionalServiceProviderVP*, *Warehouse DeveloperVP*, *WarehouseAdministratorVP*, *Information Technology-ManagersVP* et *EndUserVP*. Le métamodèle *Business Information* est étendu par les classes qui représentent les catégories des utilisateurs qui interagissent lors de l'exploitation du cycle décisionnel, à savoir : *WarehousePlatform*, *Professional-ServiceProvider*, *WarehouseDeveloper*, *Warehouse Administrator*, *EndUser* et *InformationTechnologyManagers*.

Ensuite, le *Generate Model* Code permet de générer les classes du domaine. De ce fait, un ensemble de classes correspondant au model CWM ont été généré (Figure 6.15) :

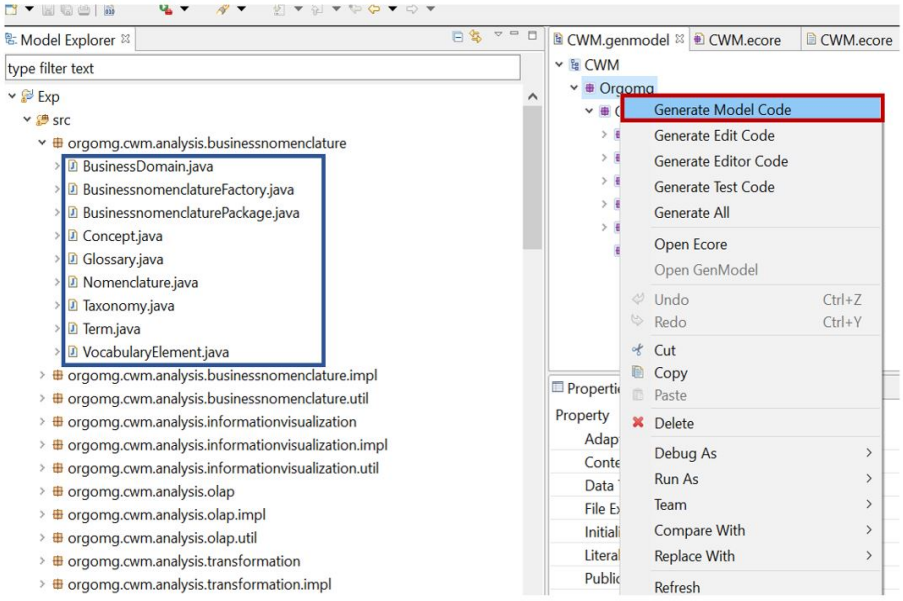

FIGURE 6.15 – La génération des classes du domaine par le Generate Model Code

Par la suite, l'outil EMF nous permet de générer automatiquement un éditeur pour construire les instances du modèle. Pour cela, deux plugins ont été ajoutés (*Exp.edit* et *Exp.editor*) et tous les codes Java ont été générés (Figure 6.17).

Une fois le modèle Point de Vue crée, on procède à la création des instances. La figure suivante (Figure 6.17) représente la création des instances pour le point de vue du *Warehouse Admin*. Ce dernier, aura comme objective de charger l'entrepôt de données dans un contexte de recherche académique.

Une fois le point de vue du *Warehouse Admin* est créé, des annotations sont générées et sauvegardées dans la base des cas pour garder la trace du point de

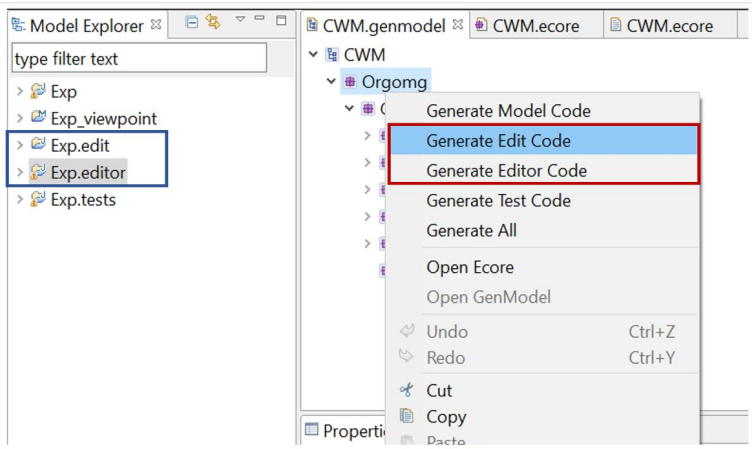

FIGURE 6.16 – La génération du.edit et .editor pour construire les instances du modèle

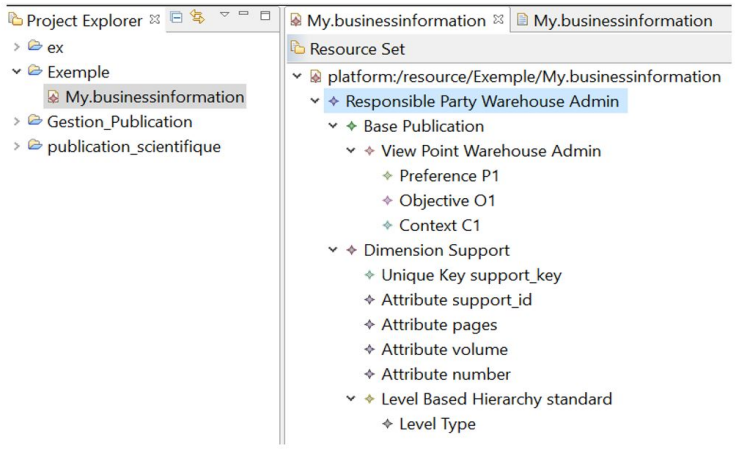

FIGURE 6.17 – Instanciation du modèle par le point de vue du Warehouse Admin

vue pour pouvoir le réutiliser par la suite. Ces annotations sont générées par un fichier XMI pour permettre l'interopérabilité (Figure 6.18).

```
<?xml version="1.0" encoding="UTF-8"?>
<orgomg.cwm.foundation.businessinformation:ResponsibleParty xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3
  <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Base" name="Publication ">
    <ownedElement xsi:type="orgomg.cwm.objectmodel.core:ViewPoint" name="Warehouse Admin ">
      <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Preference" name="P1" nameP=""/>
      <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Objective" name="O1" descriptionO="Feed the data warehouse" DOI="0,8"
       nameO="Feed"/>
      <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Context" name="C1" nameC="Academic" descriptionC="Academic research"/>
    </ownedElement>
  </ownedElement>
  <ownedElement xsi:type="orgomg.cwm.resource.multidimensional:Dimension" name="Support">
    <ownedElement xsi:type="orgomg.cwm.foundation.keysindexes:UniqueKey" name="support_key" responsibleParty="/"
     feature="//@ownedElement.1/@ownedElement.1"/>
    <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Attribute" name="support_id" responsibleParty="/"
    uniqueKey="//@ownedElement.1/@ownedElement.0"/>
    <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Attribute" name="pages" responsibleParty="/"/>
    <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Attribute" name="volume" responsibleParty="/"/>
    <ownedElement xsi:type="orgomg.cwm.objectmodel.core:Attribute" name="number" responsibleParty="/"/>
    <ownedElement xsi:type="orgomg.cwm.analysis.olap:LevelBasedHierarchy" name="standard">
      <ownedElement xsi:type="orgomg.cwm.analysis.olap:Level" name="Type"/>
    </ownedElement>
  </ownedElement>
</orgomg.cwm.foundation.businessinformation:ResponsibleParty>
```

FIGURE 6.18 – Fichier XMI des annotations générées du point de vue du Warehouse Admin

- **Réutilisation des annotations des utilisateurs**

Réutilisation des annotations des utilisateurs En vue de réutiliser les expériences et les annotations des utilisateurs au long du cycle décisionnel, nous utilisons notre Framework CCReF présenté dans le chapitre 5 en se basant sur notre métamodèle Point de Vue et sur l'approche RaPC. De ce fait, nous considérons un nouveau cas du *Warehouse Admin* pour un scénario OLAP. Les Figure 6.19 et (Figure 6.20) présentent la représentation orientée-objet du cas point de vue du *Warehouse Admin* et ses caractéristiques selon la représentation du cas point de

vue présentée dans le chapitre 5. La Figure 6.19 présente un exemple de la représentation des cas d'objet pour le point de vue d'administration d'entrepôt ; Ainsi, la Figure 6.19 montre un nouveau cas (newCase), qui contient la description d'un point de vue d'un administrateur d'entrepôt pendant une session OLAP. Un tel cas définit les trois attributs : *Context, Objective* et User_Role. Nous supposons que la partie Solution du nouveau cas n'est pas instanciée tant que le cas n'est pas encore évalué. Nous choisissons un exemple de quatre cas candidats dans

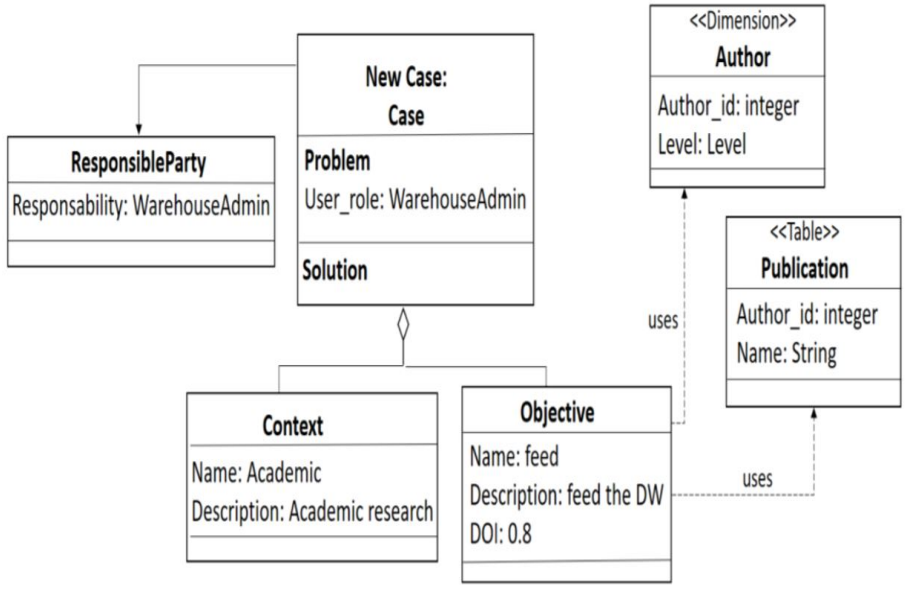

FIGURE 6.19 – Le point de vue du Warehouse Admin : Instanciation d'un nouveau cas

la base de cas (Figure 6.20). Nous nous basons sur la formule 1 du calcul de

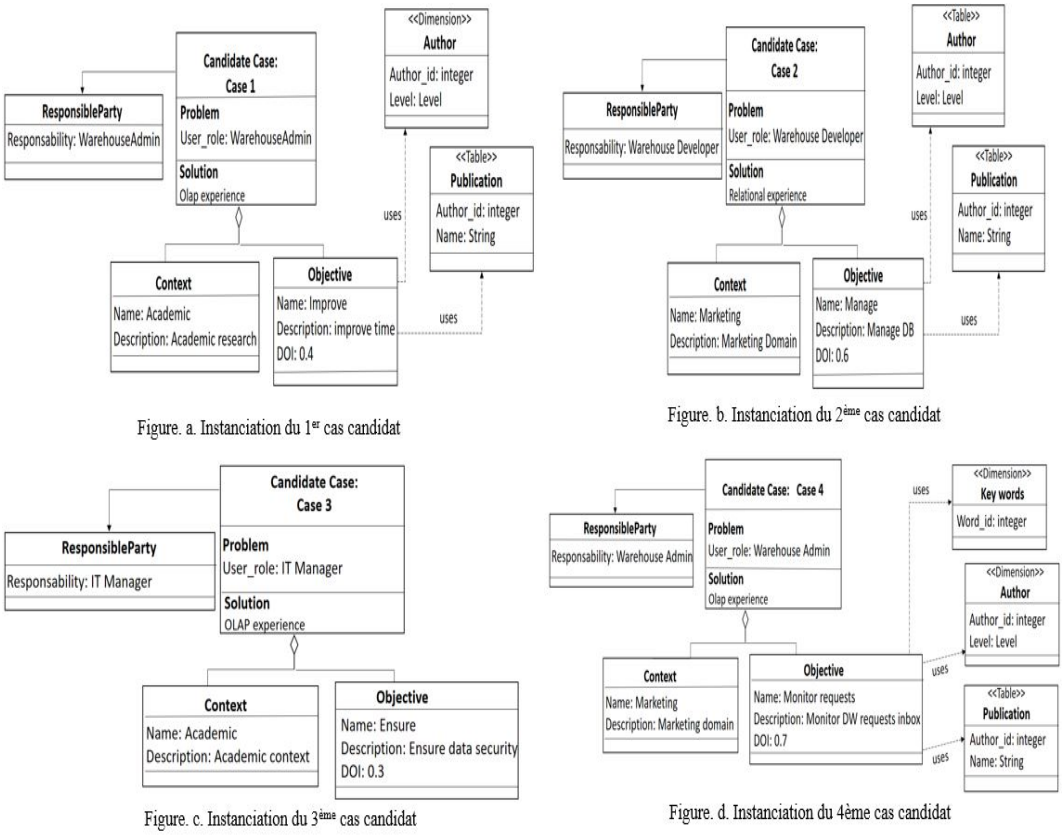

FIGURE 6.20 – Exemple d'instanciation pour quatre cas candidats Point de Vue

similarité (voir chapitre 5) pour calculer la valeur de similarité entre le nouveau cas (C^N) et chaque cas candidat (C^C) de la base de cas.

$$\mathrm{Sim}(C^N, C^C) = \sum_{1}^{n} \left(w_i \ \mathrm{sim}_i(C_i^N, C_i^C) \right) \tag{6.1}$$

Avec :

- C^N : représente un nouveau cas à évaluer
- C^C : représente le cas candidat dans la base du cas
- w_i : représente le poids associé à l'ième attribut de chaque cas
- sim_i : représente la mesure de similarité locale (c'est-à-dire mesure de similarité spécifique pour chaque attribut (descripteur) de cas).

La mesure de similarité pour le rôle de l'utilisateur : User_Role(R) est calculée entre l'attribut *responsibility* du nouveau cas et celui de chaque cas candidat en utilisant la formule de calcul de similarité pour les attributs de type Nominal (voir chapitre 5) ; le poids attribué pour User_Role est de 0.3 :

— **SIM(**C^N**,** C^C**)** = (0,3*Sim (WarehouseAdmin, WarehouseAdmin)) = 0,3* 1 = 0,3
— **SIM(**C^N**,** C^C**)** = (0,3*Sim (WarehouseAdmin, Warehouse Developer)) = 0,3* 0 = 0
— **SIM(**C^N**,** C^C**)** = (0,3*Sim (WarehouseAdmin, IT Manager)) = 0,3* 0 = 0
SIM(C^N**,** C^C**)** = (0,3*Sim (WarehouseAdmin, WarehouseAdmin)) = 0,3* 1 = 0,3

La mesure de similarité pour l'attribut *Contexte* (C) est calculée entre les attributs *name* et *description* du Context du nouveau cas et ceux du cas candidat en utilisant le coefficient de Jaccard ; le poids attribué pour l'attribut contexte est 0.1. Le calcul de similarité se fait comme suit :

— **SIM(**C^N**,** C^C**)** = (0,1*Sim (Academic, Academic)) + (0,1*Sim (Academic research, Academic research)) = 0,2
— **SIM(**C^N**,** C^C**)** = (0,1*Sim (Academic, Marketing)) + (0,1*Sim (Academic research, Marketing domain)) = 0

— **SIM(**C^N**,** C^C**)** = (0,1*Sim (Academic, Academic)) + (0,1*Sim (Academic research, Academic context)) = 0,15
— **SIM(**C^N**,** C^C**)** = (0,1*Sim (Academic, Marketing)) + (0,1*Sim (Academic research, Marketing domain)) = 0

La mesure de similarité pour l'attribut *Objectif* (O) est calculée entre les attributs de *nom, description* et *DOI* de l'objectif et ceux du cas candidat. La valeur de similarité pour les attributs *nom* et *description* est calculée en utilisant le coefficient de Jaccard, tandis que la valeur de similarité pour l'attribut *DOI* est calculée en utilisant la distance euclidienne. Nous attribuons un poids de 0.6 pour l'attribut *Objectif* comme étant l'attribut le plus important pour évaluer le cas. Par conséquent, on procède comme suit pour calculer la similarité pour l'attribut Objectif :

— **SIM(**C^N**,** C^C**)** = (0,6*Sim (feed, improve)) + (0,6*Sim (feed the DW, improve time)) + (0,6*Sim (0.8 , 0.4))= 0,36
— **SIM(**C^N**,** C^C**)** = (0,6*Sim (feed, manage)) + (0,6*Sim (feed the DW, manage DB)) + (0,6*Sim (0.8 , 0.6))= 0,48
— **SIM(**C^N**,** C^C**)** = (0,6*Sim (feed, ensure)) + (0,6*Sim (feed the DW, ensure DW security)) + (0,6*Sim (0.8 , 0.3))= 0,45
— **SIM(**C^N**,** C^C**)** = (0,6*Sim (feed, monitor requests)) + (0,6*Sim (feed the DW, monitor DW requests inbox)) + (0,6*Sim (0.8 , 0.7))= 0,66

Après avoir calculé les différentes valeurs de similarité pour les quatre cas candidats, nous avons trouvé que le premier cas candidat (cas 1) est le plus similaire à notre nouveau cas du point de vue du *Warehouse Admin*. Le tableau suivant résume les différentes opérations établies pour calculer la similarité pour le premier cas candidat selon les trois attributs qui le caractérise (*User_Role*, *Context* et *Objective*) :

Attributs	**Poids***	**Sim**$(\mathbf{C^N}, \mathbf{C^C})$	**Résultat ****

User role : responsability	0.3	0.3 $\mathrm{Sim}(wa, wa)$	0.2
Context : name and description	0.1	0.1 $\big(\mathrm{Sim}(A, A) + \mathrm{Sim}(Ar, Ar)\big)$	0.2
Objective : name, description, DOI	0.6	0.6 $\big(\mathrm{Sim}(f, imp) + \mathrm{Sim}(f_{DW}, impT) + \mathrm{Sim}(0.8, 0.4)\big)$	0.36

TABLE 6.1 – Résumé des calculs des mesures de similarité pour le cas candidat 1

— * : La somme des poids est égale à 1. Plus le poids est élevé, plus l'attribut est considéré important.
— ** La meilleure valeur est la plus grande.
— **wa** : WarehouseAdmin.
— **A** :Academic
— **Ar** : Academic research
— **f** : feed
— **imp** : improve
— $\mathbf{f_{DW}}$: feed the DW
— **impT** : improve time

Par conséquent, nous choisissons de réutiliser la solution du premier cas candidat (case 1), parce qu'elle est la plus similaire et la plus utile selon notre exemple. Une fois le cas 1 est récupéré, le CCReF adapte la solution stockée dans le cas récupéré pour répondre aux besoins du cas actuel. Nous utilisons la technique de ré-instanciation pour adapter le cas récupéré. Une fois l'adaptation terminée, il est souhaitable de vérifier que la solution adaptée prend en compte les différences entre le cas récupéré et le cas actuel, ceci est la tâche de l'utilisateur du CWM lors de la phase de révision.

7

CONCLUSION ET PERSPECTIVES

Le système décisionnel est un système multi-vues souvent mené par une variété d'experts et d'utilisateurs qui interagissent et manipulent les métadonnées et les connaissances circulant lors des échanges de données dans ce cycle avec différents rôles et profiles. Chacun de ces utilisateurs manipulent les différentes connaissances et savoir-faire et participent dans une ou plusieurs phases du développement et de maintenance de l'entrepôt de données.

Afin de fournir un accès différent dans le sens qu'il respecte les besoins, les objectifs, les préférences et les différentes perspectives des utilisateurs sur les différentes entités de l'entrepôt de données. Nous avons présenté dans ce mémoire de thèse une approche d'intégration du point de vue de l'utilisateur dans le cycle décisionnel.

Notre approche de point de vue s'inscrit dans une approche d'ingénierie de connaissance avec un formalisme orienté-objet. Dans un premier temps, nous avons décrit notre approche de point de vue dans le processus décisionnel en se basant sur le CWM et nous avons explicité les objectifs à atteindre par rapport à cette définition. Ensuite, en se basant sur notre définition du point de vue, nous avons étendu le CWM par les nouveaux éléments de notre métamodèle point de vue afin qu'il supporte la notion point de vue. Aussi, lors d'un processus multi-analyse de point de vue, il est important de mettre l'accent sur l'interaction et l'interdépendance entre les différentes analyses selon différents points de vue. De ce fait, nous avons défini un ensemble de relations pour modéliser les relations entre les points de vue des différents utilisateurs exprimés lors d'un processus

décisionnel.

En conséquence, en se basant sur le métamodèle point de vue défini et sur le processus du RàPC, nous avons développé le CCReF en vue de réutiliser les annotations et les expériences des utilisateurs en termes de ce point de vue lors d'une exploitation du processus décisionnel, de résoudre de nouvelles situations ou problèmes, et aussi de garder la trace des raisonnements et des principales décisions prises par les utilisateurs de CWM lors du processus décisionnel. En outre, ceci permettra de garantir une bonne coordination et partage des métadonnées et des annotations des utilisateurs, une meilleure réutilisation de connaissances au sein du processus décisionnel, et un support pour les utilisateurs novices du système décisionnel en assistant les utilisateurs non-experts durant leurs exploitations du processus décisionnel.

Comme tout travail de thèse, notre approche proposée est loin d'être complète, son développement et l'extension de ses fonctionnalités sont encore poursuivis. Nous proposons comme perspectives :

— Le dévelopement d'autres relations entre point de vue.
— L'amélioration des calculs de similarité pour assurer plus de pertinence.
— L'implémentation de notre framework CCReF en utilisant un outil du RàPC.
— Aussi, toute approche d'intégration de métadonnées nécessite une bonne gestion et gouvernance des métadonnées pour les organisations. C'est dans ce contexte que nous visons de projeter notre approche de point de vue dans le contexte Big data.

BIBLIOGRAPHIE

Aamodt, Agnar et Enric Plaza (1994). « Case-based reasoning : Foundational issues, methodological variations, and system approaches ». In : *AI communications* 7.1, p. 39-59 (cf. p. 146).

Althoff, Klaus-Dieter, Markus Nick et Carsten Tautz (1998). *Concepts for reuse in the experience factory and their implementation for CBR-system development*. Citeseer (cf. p. 154).

Amir, Samir (2009). « Un système d'intégration de métadonnées dédiées au multimédia. » In : *INFORSID*, p. 491-492 (cf. p. 33, 34).

Amroune, Mohammed (2014). « Vers une approche orientée aspect d'ingénierie des besoins dans les organisations multi-entreprises ». Thèse de doct. Université Toulouse le Mirail-Toulouse II (cf. p. 92, 95).

Andre, Sylvain (2004). « MDA (model driven architecture) principes et états de l'art ». In : *Conservatoire National Des Arts Et Métiers, Lyon, France* (cf. p. 58).

Anwar, Adil (2009). « Formalisation par une approche IDM de la composition de modèles dans le profil VUML ». Thèse de doct. Thèse de doctorat, Université de Toulouse (cf. p. 90).

Atkinson, Colin et Thomas Kuhne (2003). « Model-driven development : a metamodeling foundation ». In : *IEEE software* 20.5, p. 36-41 (cf. p. 15).

Audibert, Laurent (2009). *UML 2 : de l'apprentissage à la pratique...* Ellipses (cf. p. 173, 175).

Bach, Kerstin et Klaus-Dieter Althoff (2012). « Developing case-based reasoning applications using mycbr 3 ». In : *International Conference on Case-Based Reasoning*. Springer, p. 17-31 (cf. p. 153).

Bach, Kerstin, Tomasz Szczepanski, Agnar Aamodt, Odd Erik Gundersen et Paul Jarle Mork (2016). « Case representation and similarity assessment in the

selfBACK decision support system ». In : *International Conference on Case-Based Reasoning*. Springer, p. 32-46 (cf. p. 155).

Bach, Thành Lê (2006). « Construction d'un Web sémantique multi-points de vue ». Thèse de doct. INRIA Sophia Antipolis (cf. p. 104).

Bardou, Daniel (1998a). « Etude des langages a prototypes, du m ecanisme de d el egation, et de son rapport a la notion de point de vue ». In : *Th ese de doctorat en informatique, Universit e Montpellier* 2 (cf. p. 90).

Bardou, Daniel (1998b). « Roles, subjects and aspects : How do they relate ? » In : *European Conference on Object-Oriented Programming*. Springer, p. 418-419 (cf. p. 93).

Batasova, Svetlana, Maria Efimova, Ivan Kholod et Alexey Semenchenko (2015). « Preparation of distributed heterogeneous data for data mining ». In : *Soft Computing and Measurements (SCM), 2015 XVIII International Conference on*. IEEE, p. 205-207 (cf. p. 67, 73).

Behja, Hicham (2009). « Plateforme objet d'évaluation orientée point de vue d'un système d'information ». Thèse de doct. Faculté des Sciences Ben M'Sik.Casablanca. Maroc (cf. p. 98, 99, 101, 116, 157).

Behja, Hicham, Brigitte Trousse et Abdelaziz Marzak (2005). « Prise en compte des" points de vue" pour l'annotation d'un processus d'extraction de connaissances à partir de données. » In : *EGC*, p. 245-256 (cf. p. 98, 116).

Bello-Tomas, Juan Jose, Pedro A Gonzalez-Calero et Belen Diaz-Agudo (2004). « Jcolibri : An object-oriented framework for building cbr systems ». In : *European Conference on Case-Based Reasoning*. Springer, p. 32-46 (cf. p. 153).

Bendjenna, Hakim et al. (2010). « Ingenierie des exigences pour les processus inter-organisationnels ». In : *Constantine, Algerie, Toulouse, France* (cf. p. 95).

Bézivin, Jean et Jean-Pierre Briot (2004). « Sur les principes de base de l'ingénierie des modèles. » In : *L'OBJET* 10.4, p. 145-157 (cf. p. 11, 12, 15).

Biernacki, Patrick et Dan Waldorf (1981). « Snowball sampling : Problems and techniques of chain referral sampling ». In : *Sociological methods & research* 10.2, p. 141-163 (cf. p. 3).

Blanc, Xavier et Olivier Salvatori (2011). *MDA en action : Ingénierie logicielle guidée par les modèles*. Editions Eyrolles (cf. p. 11, 12).

Blanco, Carlos, Ignacio Garcia-Rodriguez de Guzman, Eduardo Fernandez-Medina et Juan Trujillo (2015). « An architecture for automatically developing secure OLAP applications from models ». In : *Information and Software Technology* 59, p. 1-16 (cf. p. 68, 74).

Bobrow, Daniel G et Terry Winograd (1977). « An overview of KRL, a knowledge representation language ». In : *Cognitive science* 1.1, p. 3-46 (cf. p. 79).

Bobrow, DG et MJ Stefik (1982). « LOOPS : data and object oriented Programming for Interlisp ». In : *Discussion papers, Proceedings of the European Conference on AI, Orsay, France* (cf. p. 79).

Boufaida, Mahmoud et Benchikha Fouzia (2007). « The viewpoint mechanism for object-oriented databases modelling, distribution and evolution ». In : *Journal of computing and information technology* 15.2, p. 95-110 (cf. p. 106).

Bouman, Roland et Jos Van Dongen (2009). *Pentaho solutions : business intelligence and data warehousing with Pentaho and MySQL*. Wiley Publishing (cf. p. 63).

Bouquet, Paolo, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini et Heiner Stuckenschmidt (2003). « C-owl : Contextualizing ontologies ». In : *International Semantic Web Conference*. Springer, p. 164-179 (cf. p. 104).

Carré, Bernard (1990). « Multiple and evolutive representation in the ROME language ». In : *Proceedings of Technology of Object-Oriented Languages and Systems* (cf. p. 79).

Castagna, Giuseppe, Nils Gesbert et Luca Padovani (2009). « A theory of contracts for web services ». In : *ACM Transactions on Programming Languages and Systems (TOPLAS)* 31.5, p. 19 (cf. p. 13).

Charlet, Jean, Manuel Zacklad et Gilles Kassel (2000). « Ingénierie des connaissances ». In : (cf. p. 141).

Combemale, Benoit, Xavier Crégut, Alain Caplain et Bernard Coulette (2006). « Modélisation rigoureuse en SPEM de procédé de développement. » In : *LMO*. T. 2006, p. 135-150 (cf. p. 18).

Consortium, Eclipse et al. (2007). *Eclipse Graphical Modeling Framework (GMF)* (cf. p. 24).

Cortes Robles, Guillermo (2006). « Management de l'innovation technologique et des connaissances : synergie entre la théorie TRIZ et le Raisonnement à Partir de Cas. Application en génie des procédés et systèmes industriels. » Thèse de doct. Institut National Polytechnique de Toulouse (cf. p. 153).

Darmont, Jérôme (2007). « Expérimentations avec le banc d'essais pour entrepôts de données DWEB ». In : (cf. p. 69, 74).

Davis, Harley (1987). *VIEWS : Multiple perspectives and structured objects in a knowledge representation language. Bachelor and Master of Science Thesis* (cf. p. 97).

De Mantaras, Ramon Lopez, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, MICHAEL T COX, Kenneth Forbus et al. (2005). « Retrieval, reuse, revision and retention in case-based reasoning ». In : *The Knowledge Engineering Review* 20.3, p. 215-240 (cf. p. 150).

De Miguel, Miguel, Jean Jourdan et Serge Salicki (2002). « Practical Experiences in the Application of MDA ». In : *International Conference on the Unified Modeling Language*. Springer, p. 128-139 (cf. p. 30).

Debrauwer, Laurent (1998). « Des vues aux contextes pour la structuration fonctionnelle de bases de données à objets en CROME ». Thèse de doct. Lille 1 (cf. p. 90).

Dekker, Lenneke (1994). « Frome : représentaion multiple et classification d'objets avec points de vue ». Thèse de doct. Lille 1 (cf. p. 80).

Demraoui, Lamiae, Hicham Behja, Rachid Ben Abbou et al. (2014). « Towards Integration of the users' preferences into the common warehouse metamodel ». In : *Information Science and Technology (CIST), 2014 Third IEEE International Colloquium in*. IEEE, p. 151-154 (cf. p. 157).

Demraoui, Lamiae, Hicham Behja, El Moukhtar Zemmouri et Rachid Ben Abbou (2016). « A viewpoint based extension of the common warehouse meta-

model to support the user's viewpoint approach ». In : *International Journal of Metadata, Semantics and Ontologies* 11.3, p. 137-154 (cf. p. 157).

Despotakis, Dimoklis (2013). *Modelling viewpoints in user generated content*. University of Leeds (cf. p. 107, 108).

Devèze, Benjamin et Matthieu Fouquin (2004). « Case-Based Reasoning ». In : *Rapport des études en spécialité SCIA, école de l'ingénieur EPTA, France* (cf. p. 153).

Di Tria, Francesco, Ezio Lefons et Filippo Tangorra (2012). « Metadata for approximate query answering systems ». In : *Advances in Software Engineering* 2012, p. 7 (cf. p. 66, 73).

Diaw, Samba, Redouane Lbath et Bernard Coulette (2010). « État de l'art sur le développement logiciel basé sur les transformations de modèles. » In : *Technique et Science Informatiques* 29.4-5, p. 505-536 (cf. p. 12, 20).

Djebbi, Olfa et Marie Pierre Gervais (2004). « Mda : Vers l'industrialisation de la construction d'applications réparties ». In : *Mémoire de DEA* (cf. p. 16, 17).

Djezzar, Meriem et Zizette Boufaida (2015). « Ontological classification of individuals : a multi-viewpoints approach ». In : *International Journal of Reasoning-based Intelligent Systems* 7.3-4, p. 276-285 (cf. p. 106).

Duval, Erik, Wayne Hodgins, Stuart Sutton et Stuart L Weibel (2002). « Metadata principles and practicalities ». In : *D-lib Magazine* 8.4, p. 1082-9873 (cf. p. 30).

Eckert, Kai, Magnus Pfeffer et Johanna Völker (2010). « Towards Interoperable Metadata Provenance ». In : *Proceedings of the ISWC workshop on Semantic Web for Provenance Management (SWPM)* (cf. p. 35).

Egyed-Zsigmond, Elöd, Alain Mille et Yannick Prié (2003). « Club♣(Tréfle) : a use trace model ». In : *International Conference on Case-Based Reasoning*. Springer, p. 146-160 (cf. p. 154).

Ehrig, Karsten, Claudia Ermel, Stefan Hänsgen et Gabriele Taentzer (2005). « Generation of visual editors as eclipse plug-ins ». In : *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. ACM, p. 134-143 (cf. p. 24).

El Ouazzani, Amina, Sara Rhazlane, Nouria Harbi et Hassan Badir (2016). « Dynamic management of data warehouse security levels based on user profiles ». In : *Information Science and Technology (CiSt), 2016 4th IEEE International Colloquium on*. IEEE, p. 59-64 (cf. p. 68, 74).

Estefan, Jeff A et al. (2007). « Survey of model-based systems engineering (MBSE) methodologies ». In : *Incose MBSE Focus Group* 25.8, p. 1-12 (cf. p. 45).

Euzenat, Jérôme (2001). « Towards a principled approach to semantic interoperability ». In : *Proc. IJCAI 2001 workshop on ontology and information sharing*. No commercial editor., p. 19-25 (cf. p. 35).

Falquet, Gilles et Claire-Lise Mottaz Jiang (2001). « Navigation hypertexte dans une ontologie multi-points de vue ». In : (cf. p. 104, 105).

Farail, Patrick et François Vernadat (2012). *Toolkit in OPen-source for Critical Applications & SystEms Development* (cf. p. 20, 24).

Ferrer, Xavier et Enric Plaza (2016). « Concept Discovery and Argument Bundles in the Experience Web ». In : *International Conference on Case-Based Reasoning*. Springer, p. 108-123 (cf. p. 155).

Fuchs, B, J Lieber, A Mille et A Napoli (2006). « Une première formalisation de la phase d'élaboration du raisonnement à partir de cas ». In : *Actes du 14ième atelier du raisonnement à partir de cas, Besançon* (cf. p. 146, 148).

Fuchs, Béatrice (1997). « Représentation des connaissances pour le raisonnement à partir de cas : Le système ROCADE ». Thèse de doct. Saint-Etienne (cf. p. 163).

Fuchs, Béatrice, Jean Lieber, Alain Mille et Amedeo Napoli (1999). « Towards a unified theory of adaptation in case-based reasoning ». In : *International Conference on Case-Based Reasoning*. Springer, p. 104-117 (cf. p. 150).

Gamma, Erich (1995). *Design patterns : elements of reusable object-oriented software*. Pearson Education India (cf. p. 100).

Gebhardt, Friedrich, Angi Voss, Wolfgang Gräther et Barbara Schmidt-Belz (1997). « Reasoning with complex cases ». In : *Reasoning with Complex Cases*. Springer, p. 11-26 (cf. p. 140).

GESCHE, Samuel (2008). « Confrontation enrichissante de points de vue-opinion Définition, modélisation et instrumentation ». In : *month* (cf. p. 104).

GOLDSTEIN, Ira P et Daniel G BOBROW (1980). « Descriptions for a Programming Environment. » In : *AAAI*, p. 187-189 (cf. p. 79).

GOMES, Pedro, José FARINHA et Maria José TRIGUEIROS (2007). « A data quality metamodel extension to CWM ». In : *Proceedings of the fourth Asia-Pacific conference on Comceptual modelling-Volume 67*. Australian Computer Society, Inc., p. 17-26 (cf. p. 72).

GREENBERG, Jane (2003). « Metadata and the world wide web ». In : *Encyclopedia of library and information science* 3, p. 1876-1888 (cf. p. 30).

GUESSOUM, Souad, Mohamed Tayeb LASKRI et Jean LIEBER (2014). « RespiDiag : a case-based reasoning system for the diagnosis of chronic obstructive pulmonary disease ». In : *Expert Systems with Applications* 41.2, p. 267-273 (cf. p. 156).

HACHANI, Ouafa (2006). « Patrons de conception à base d'aspects pour l'ingénierie des systèmes d'information par réutilisation ». Thèse de doct. Université Joseph-Fourier-Grenoble I (cf. p. 92).

HAOUCHINE, Mohamed Karim (2009). « Remémoration guidée par l'adaptation et maintenance des systèmes de diagnostic industriel par l'approche du raisonnement à partir de cas. » Thèse de doct. Université de Franche-Comté (cf. p. 150).

HARRISON, William et Harold OSSHER (1993). *Subject-oriented programming : a critique of pure objects*. T. 28. 10. ACM (cf. p. 90, 91).

HASLHOFER, Bernhard et Wolfgang KLAS (2010). « A survey of techniques for achieving metadata interoperability ». In : *ACM Computing Surveys (CSUR)* 42.2, p. 7 (cf. p. 35).

HATON, Jean-Paul et Marie-Christine HATON (1989). *L'intelligence artificielle*. Presses universitaires de France (cf. p. 140).

HEMAM, Mounir et Zizette BOUFAIDA (2009). « Représentation d'ontologies multi-points de vue : une approche basée sur la logique de descriptions ». In : *Journées Francophones d'Ingénierie des Connaissances, Tunisia* (cf. p. 106).

Jaczynski, Michel (1998). « Modèle et plate-forme à objets pour l'indexation des cas par situations comportementales : application à l'assistance à la navigation sur le web ». Thèse de doct. Nice (cf. p. 144, 145, 148, 149, 153).

Jaczynski, Michel et Brigitte Trousse (1997). « CBR* Tools : une bibliothèque objet pour l'indexation des cas par situation comportementale ». Thèse de doct. INRIA (cf. p. 145, 153).

Jézéquel, Jean-Marc, Olivier Barais et Franck Fleurey (2009). « Model driven language engineering with kermeta ». In : *International Summer School on Generative and Transformational Techniques in Software Engineering*. Springer, p. 201-221 (cf. p. 12, 22).

Karoui, Hager, Rushed Kanawati et Laure Petrucci (2006). « COBRAS : Cooperative CBR system for bibliographical reference recommendation ». In : *European Conference on Case-Based Reasoning*. Springer, p. 76-90 (cf. p. 151).

Kelly, Steven, Kalle Lyytinen et Matti Rossi (1996). « Metaedit+ a fully configurable multi-user and multi-tool case and came environment ». In : *International Conference on Advanced Information Systems Engineering*. Springer, p. 1-21 (cf. p. 20).

Kelly, Steven et Juha-Pekka Tolvanen (2008). *Domain-specific modeling : enabling full code generation*. John Wiley & Sons (cf. p. 20).

Kent, Stuart (2002). « Model driven engineering ». In : *International Conference on Integrated Formal Methods*. Springer, p. 286-298 (cf. p. 12).

Kermeta (2017). url : http://www.kermeta.org (cf. p. 22).

Kiczales, Gregor, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier et John Irwin (1997). « Aspect-oriented programming ». In : *European conference on object-oriented programming*. Springer, p. 220-242 (cf. p. 90, 92).

Kimball, Ralph (1996). « The data warehouse toolkit : practical techniques for building dimensional data warehouse ». In : *NY : John Willey & Sons* 248.4 (cf. p. 40).

KLEPPE, Anneke G, Jos B WARMER et Wim BAST (2003). *MDA explained : the model driven architecture : practice and promise*. Addison-Wesley Professional (cf. p. 14).

KOLODNER, Janet (1988). « Retrieving events from a case memory : A parallel implementation ». In : *Proc. of Case-Based Reasoning Workshop* (cf. p. 140, 145, 151).

KOLODNER, Janet (2014). *Case-based reasoning*. Morgan Kaufmann (cf. p. 143).

KOZMINA, Natalija et Darja SOLODOVNIKOVA (2011). « On implicitly discovered OLAP schema-specific preferences in reporting tool ». In : *Scientific Journal of Riga Technical University. Computer Sciences* 43.1, p. 35-42 (cf. p. 70, 74).

KRUCHTEN, P et S COOK (2002). « The Software Process Engineering Metamodel (SPEM), OMG Specification ». In : *Boston, MA : OMG* (cf. p. 18).

LAFORCADE, P (2004). « Méta-modélisation UML pour la mise en oeuvre de situations problèmes coopératives. LIUPPA ». In : *Doctorat en informatique de l'UPPA* (cf. p. 31).

LAKHRISSI, Younes (2010). « Intégration de la modélisation comportementale dans la conception par points de vue ». Thèse de doct. Université Toulouse le Mirail-Toulouse II (cf. p. 90).

LE NY, Jean François (1989). *Science cognitive et compréhension du langage*. T. 103. Presses universitaires de France (cf. p. 141).

LEAKE, David B et David C WILSON (1998). « Categorizing case-base maintenance : Dimensions and directions ». In : *European Workshop on Advances in Case-Based Reasoning*. Springer, p. 196-207 (cf. p. 152).

LÉDECZI, Ákos, Arpad BAKAY, Miklos MAROTI, Peter VOLGYESI, Greg NORDSTROM, Jonathan SPRINKLE et Gábor KARSAI (2001). « Composing domain-specific design environments ». In : *Computer* 34.11, p. 44-51 (cf. p. 17).

LENZ, Mario, Hans-Dieter BURKHARD, Petra PIRK, Eric AURIOL et Michel MANAGO (1996). « CBR for diagnosis and decision support ». In : *Ai communications* 9.3, p. 138-146 (cf. p. 169).

Liao, T Warren, Zhiming Zhang et Claude R Mount (1998). « Similarity measures for retrieval in case-based reasoning systems ». In : *Applied Artificial Intelligence* 12.4, p. 267-288 (cf. p. 150).

Lieber, Jean, Mathieu D'aquin, Sébastien Brachais et Amedeo Napoli (2004). « Une étude comparative de quelques travaux sur l'acquisition de connaissances d'adaptation pour le raisonnement à partir de cas ». In : *Actes du 12ème atelier de raisonnement à partir de cas (RàPC'04)*, p. 53-60 (cf. p. 150).

Main, Julie, Tharam S Dillon et Simon CK Shiu (2001). « A tutorial on case based reasoning ». In : *Soft computing in case based reasoning*. Springer, p. 1-28 (cf. p. 146, 148).

Malek, M et R Kanawati (2004). « A data driven CBR approach : A contineous maintenance strategy ». In : *the International Conference on Advances in Intelligent Systems : Theory and Applications*, p. 281-290 (cf. p. 151).

Manago, Michael, Ralph Bergmann, Noël Conruyt, Ralph Traphöner, James Pasley, Jacques Le Renard, Frank Maurer, Stefan Wess, Klaus-Dieter Althoff et Sylvie Dumont (1994). « CASUEL : A common case representation language ». In : *INRECA Consortium. Available on the World-Wide Web at http ://ww-wagr. informatik. unikl. de/bergmann/casuel/CASUEL toc2* 4 (cf. p. 160).

Marcaillou, Sophie (1995). « Intégration de la notion de points de vue dans la modélisation par objets : le langage VBOOL ». Thèse de doct. Toulouse 3 (cf. p. 83, 84).

Mariño, Olga (1993). « Raisonnement classificatoire dans une représentation à objets multi-points de vue ». Thèse de doct. Université Joseph Fourier (Grenoble) (cf. p. 81).

Martin, Andreas, Sandro Emmenegger, Knut Hinkelmann et Barbara Thönssen (2017). « A viewpoint-based case-based reasoning approach utilising an enterprise architecture ontology for experience management ». In : *Enterprise Information Systems* 11.4, p. 551-575 (cf. p. 156).

MARTIN, William A (1978). *Descriptions and the specialization of concepts*. Rapp. tech. MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE (cf. p. 79, 153).

MAXIMINI, Dipl-Inform Kerstin (2007). *Collaborative Agent-Based Knowledge Engine* (cf. p. 153).

MDA, OMG (2008). *Object Management Group Model Driven Architecture* (cf. p. 12).

MEDINA, Enrique et Juan TRUJILLO (2002). « A standard for representing multidimensional properties : The Common Warehouse Metamodel (CWM) ». In : *East European Conference on Advances in Databases and Information Systems*. Springer, p. 232-247 (cf. p. 69).

MEKROUD, Noureddine et Abdelouahab MOUSSAOUI (2009). « Approche Hybride pour le Diagnostic Industriel basée sur le RàPC et le Datamining : Utilisation de la Plateforme JCOLIBRI 2.1. » In : *CIIA* (cf. p. 153).

MELCHERT, Florian, Alexander SCHWINN, Clemens HERRMANN et Robert WINTER (2005). « Using reference models for data warehouse metadata management ». In : (cf. p. 64).

MENET, Ludovic (2010). « Formalisation d'une approche d'Ingénierie Dirigée par les Modèles appliquée au domaine de la gestion des données de référence ». Thèse de doct. Paris 8 (cf. p. 12, 15).

MERKS, Ed, R ELIERSICK, T GROSE, F BUDINSKY et D STEINBERG (2003). « The eclipse modeling framework ». In : *retrieved from, total*, p. 37 (cf. p. 24).

METACASE (2017). URL : http://www.metacase.com/ (cf. p. 20).

METAMODEL, Common Warehouse (p. d.). *Specification. OMG Documents, Feb. 2001* (cf. p. 59).

METAMODEL, Common Warehouse (2003). « Specification, Version 1.1 ». In : *Object Management Group* (cf. p. 44).

MEYER, Bertrand (1992). *Eiffel : the language*. Prentice-Hall, Inc. (cf. p. 84).

MIDOUNI, Sid Ahmed Djallal, Jérôme DARMONT et Fadila BENTAYEB (2009). « Approche de modélisation multidimensionnelle des données complexes : Appli-

cation aux données médicales ». In : *5èmes Journées francophones sur les Entrepôts de Données et l'Analyse en ligne (EDA 09)*. Cépaduès, p. 155-166 (cf. p. 69, 74).

Mili, Hafedh, Joumana Dargham, Ali Mili, Omar Cherkaoui et Robert Godin (1999). « View programming for decentralized development of OO programs ». In : *Technology of Object-Oriented Languages and Systems, 1999. TOOLS 30 Proceedings*. IEEE, p. 210-221 (cf. p. 90).

Mille, Alain (1999). *Tutorial CBR : Etat de l'art de raisonnement à partir de cas*. Plateforme AFIA (cf. p. 151).

Mille, Alain (2006). « Traces Based Reasoning (TBR) Definition, illustration and echoes with story telling ». In : *Rapport Technique RR-LIRIS-2006-002, LIRIS UMR* 5205 (cf. p. 146, 147).

Mille, Alain, Beatrice Fuchs et Olivier Herbeaux (1996). « A unifying framework for Adaptation in Case-Based Reasoning ». In : *Workshop on Adaptation in Case-Based Reasoning, ECAI-96*, p. 22-28 (cf. p. 141, 142).

Miller, Joaquin, Jishnu Mukerji, M Belaunde et al. (2003). « MDA guide ». In : *Object Management Group*, p. 26-27 (cf. p. 13).

Minsky, N (1975). « A framework for representing knowledge in the psychology of computer vision (Winston O., ed) McGraw-Hill ». In : *New York* (cf. p. 140).

Muller, Pierre-Alain, Franck Fleurey et Jean-Marc Jézéquel (2005). « Weaving executability into object-oriented meta-languages ». In : *International Conference on Model Driven Engineering Languages and Systems*. Springer, p. 264-278 (cf. p. 20, 22).

Nascimento Fidalgo, Robson do, Valéria Cesário Times, Joel da Silva, Fernando da Fonseca de Souza et Ana Carolina Salgado (2010). « Providing multidimensional and geographical integration based on a gdw and metamodels ». In : *Journal of information and data management* 1.1, p. 93 (cf. p. 69).

Nassar, M (2004). « VUML : une extension UML orientée point de vue ». Thèse de doct. (cf. p. 83, 86, 121).

Nuseibeh, Bashar, Jeff Kramer et Anthony Finkelstein (1994). « A framework for expressing the relationships between multiple views in requirements spe-

cification ». In : *IEEE Transactions on software engineering* 20.10, p. 760-773 (cf. p. 94).

Olbrich, Sebastian (2010). « Warehousing and Analyzing Streaming Data Quality Information. » In : *AMCIS*, p. 159 (cf. p. 66, 72).

OMG, OCL (2005). « 2.0 Specification ». In : *Object Management Group, Final Adopted Specification* (cf. p. 87).

Omg, QVT (2008). « Meta object facility (mof) 2.0 query/view/transformation specification ». In : *Final Adopted Specification (November 2005)* (cf. p. 17).

OMG, SPEM et OMG Notation (2008). « Software & systems process engineering meta-model specification ». In : *OMG Std., Rev* 2 (cf. p. 18).

Ossher, Harold, Matthew Kaplan, Alexander Katz, William Harrison et Vincent Kruskal (1996). « Specifying subject-oriented composition ». In : *Theory and Practice of Object Systems* 2.3, p. 179-202 (cf. p. 91).

Pal, Sankar K et Simon CK Shiu (2004). *Foundations of soft case-based reasoning*. T. 8. John Wiley & Sons (cf. p. 144, 146, 163, 164).

Polese, Giuseppe (2014). « A decision support system for evidence based medicine ». In : *Journal of Visual Languages & Computing* 25.6, p. 858-867 (cf. p. 155).

Poole, John, Dan Chang, Douglas Tolbert et David Mellor (2002). *Common warehouse metamodel*. T. 20. John Wiley & Sons (cf. p. 4, 43, 44, 48, 49, 56, 126).

Prat, Nicolas, Jacky Akoka et Isabelle Comyn-Wattiau (2006). « A UML-based data warehouse design method ». In : *Decision support systems* 42.3, p. 1449-1473 (cf. p. 70).

Rasovska, Ivana (2006). « Contribution à une méthodologie de capitalisation des connaissances basée sur le raisonnement à partir de cas : Application au diagnostic dans une plateforme d'e-maintenance. » Thèse de doct. Université de Franche-Comté (cf. p. 153).

Recio-Garcia, Juan A, Pedro A Gonzalez-Calero et Belen Diaz-Agudo (2014). « jcolibri2 : A framework for building Case-based reasoning systems ». In : *Science of Computer Programming* 79, p. 126-145 (cf. p. 153).

Reinartz, Thomas, Ioannis Iglezakis et Thomas Roth-Berghofer (2000). « On quality measures for case base maintenance ». In : *European Workshop on Advances in Case-Based Reasoning*. Springer, p. 247-260 (cf. p. 144).

Reuss, Pascal, Rotem Stram, Cedric Juckenack, Klaus-Dieter Althoff, Wolfram Henkel, Daniel Fischer et Frieder Henning (2016). « Feature-tak-framework for extraction, analysis, and transformation of unstructured textual aircraft knowledge ». In : *International Conference on Case-Based Reasoning*. Springer, p. 327-341 (cf. p. 155).

Ribière, Myriam (1999). « Représentation et gestion de multiples points de vue dans le formalisme des graphes conceptuels ». Thèse de doct. Nice (cf. p. 81, 97).

Ribière, Myriam et Rose Dieng-Kuntz (2002). « A viewpoint model for cooperative building of an ontology ». In : *International Conference on Conceptual Structures*. Springer, p. 220-234 (cf. p. 97).

Richter, Jeffrey (2002). *Applied Microsoft. NET framework programming*. T. 1. Microsoft Press Redmond (cf. p. 13).

Roth-Berghofer, Thomas R et Daniel Bahls (2008). « Explanation capabilities of the open source case-based reasoning tool mycbr ». In : *Proceedings of the thirteenth UK workshop on Case-Based Reasoning UKCBR*, p. 23-34 (cf. p. 153).

Rumbaugh, J et D Rumbaugh (2005). « Groundwater vistas 5.0, 1996e2005. Environmental Simulation ». In : *Inc Google Scholar* (cf. p. 17).

Sanford, A Friedenthal (2003). *UML for systems engineering request for proposal*. Rapp. tech. Technical report, OMG (cf. p. 18).

Santana, Aurisan Souza de et Ana Maria de Carvalho Moura (2004). « Metadata to support transformations and data & metadata lineage in a warehousing environment ». In : *International Conference on Data Warehousing and Knowledge Discovery*. Springer, p. 249-258 (cf. p. 66).

Santini, Simone et Ramesh Jain (1999). « Similarity measures ». In : *IEEE Transactions on pattern analysis and machine Intelligence* 21.9, p. 871-883 (cf. p. 150).

SCHANK, Roger C (1983). *Dynamic memory : A theory of reminding and learning in computers and people*. cambridge university press (cf. p. 139, 140).

SCHMIDT, DG et Fred KUHNS (2000). « An overview of the real-time CORBA specification ». In : *Computer* 33.6, p. 56-63 (cf. p. 13).

SEIDEWITZ, Edwin (2003). « What models mean ». In : *IEEE software* 20.5, p. 26-32 (cf. p. 15).

SIERRA, Kathy et Bert BATES (2005). *Head First Java : A Brain-Friendly Guide*. " O'Reilly Media, Inc." (cf. p. 13).

SILVA, Joel da, Anjolina G de OLIVEIRA, Robson N FIDALGO, Ana Carolina SALGADO et Valéria C TIMES (2010). « Modelling and querying geographical data warehouses ». In : *Information Systems* 35.5, p. 592-614 (cf. p. 68).

SIMIĆ, Dragan, Vladimir KURBALIJA et Zoran BUDIMAC (2003). « An application of case-based reasoning in multidimensional database architecture ». In : *International Conference on Data Warehousing and Knowledge Discovery*. Springer, p. 66-75 (cf. p. 154).

SOLER, Emilio, Juan TRUJILLO, Eduardo FERNANDEZ-MEDINA et Mario PIATTINI (2008). « Building a secure star schema in data warehouses by an extension of the relational package from CWM ». In : *Computer Standards & Interfaces* 30.6, p. 341-350 (cf. p. 67, 73).

SOLER, Emilio, Rodolfo VILLARROEL, Juan TRUJILLO, Eduardo FERNANDEZ-MEDINA et Mario PIATTINI (2006). « Representing security and audit rules for data warehouses at the logical level by using the common warehouse metamodel ». In : *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*. IEEE, 8-pp (cf. p. 67).

SOLEY, Richard et al. (2000). « Model driven architecture ». In : *OMG white paper* 308.308, p. 5 (cf. p. 13).

SOLODOVNIKOVA, Darja, Laila NIEDRITE et Natalija KOZMINA (2015). « Handling Evolving Data Warehouse Requirements ». In : *East European Conference on Advances in Databases and Information Systems*. Springer, p. 334-345 (cf. p. 70, 75).

Sommerville, Ian et Pete Sawyer (1997). *Requirements engineering : a good practice guide*. John Wiley & Sons, Inc. (cf. p. 94).

Sommerville, Ian, Peter Sawyer et Stephen Viller (1998). « Viewpoints for requirements elicitation : a practical approach ». In : *Requirements Engineering, 1998. Proceedings. 1998 Third International Conference on*. IEEE, p. 74-81 (cf. p. 94).

Sowa, John F (1983). « Conceptual structures : information processing in mind and machine ». In : (cf. p. 97).

Specification, Common Warehouse Metamodel (2001). « Volumes 1 & 2 ». In : *InfiniBand Trade Association* (cf. p. 57).

Stefik, Mark et Daniel G Bobrow (1985). « Object-oriented programming : Themes and variations ». In : *AI magazine* 6.4, p. 40 (cf. p. 79).

Tavac, Marek et Viliam Tavac (2013). « The general algorithm for the design of the MDA transformation models ». In : *Computational Intelligence, Communication Systems and Networks (CICSyN), 2013 Fifth International Conference on*. IEEE, p. 171-176 (cf. p. 71, 75).

Tawfik, Andrew A et Janet L Kolodner (2016). « Systematizing scaffolding for problem-based learning : A view from case-based reasoning ». In : *Interdisciplinary Journal of Problem-Based Learning* 10.1, p. 6 (cf. p. 156).

Thavornun, Varunya (2015). « Metadata Management for Knowledge Discovery ». Mém. de mast. Universitat Politècnica de Catalunya (cf. p. 71, 75).

Thess, Michael et Michael Bolotnicov (2004). « XELOPES Library Documentation Version 1.2. 5 ». In : *Prudsys AG* (cf. p. 62, 73).

Tolvanen, Juha-Pekka et Matti Rossi (2003). « MetaEdit+ : defining and using domain-specific modeling languages and code generators ». In : *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. ACM, p. 92-93 (cf. p. 20).

Trousse, Brigitte (1998). « Viewpoint management for cooperative design ». In : *INRIA, Sophia Antipolis, Action AID* (cf. p. 99, 145).

Vanwormhoudt, Gilles (1999). « CROME : un cadre de programmation par objets structurés en contextes ». Thèse de doct. Lille 1 (cf. p. 90).

VELOSO, Manuela M (1994). *Planning and learning by analogical reasoning*. T. 886. Springer Science & Business Media (cf. p. 140).

WATSON, Hugh J et Barbara H WIXOM (2007). « The current state of business intelligence ». In : *Computer* 40.9 (cf. p. 3).

WIENHOFEN, Leendert WM et Bjørn Magnus MATHISEN (2016). « Defining the initial case-base for a CBR operator support system in digital finishing ». In : *International Conference on Case-Based Reasoning*. Springer, p. 430-444 (cf. p. 155).

WOOLCOTT, Liz (2017). *Understanding Metadata : What is Metadata, and What is it For ?, by Jenn Riley. Baltimore : National Information Standards Organization, 2017. iii, 45 p. ISBN : 978-1-937522-72-8. http ://www. niso. org/apps/group_public/download. php/17446/Understanding% 20Metadata. pdf* (cf. p. 38).

ZEMMOURI, E (2013). *Représentation et gestion des connaissances dans un processus d'Extraction de Connaisssance à partir de Données multi-points de vue* (cf. p. 101, 102, 116).

ZEMMOURI, E, H BEHJA et A MARZAK (2010). « Architecture web service d'un processus d'ECD multiutilisateur ». In : *Actes de NGNS'10*, p. 171-177 (cf. p. 99).

ZHAO, Xiaofei et Zhiqiu HUANG (2006). « A formal framework for reasoning on metadata based on CWM ». In : *International Conference on Conceptual Modeling*. Springer, p. 371-384 (cf. p. 65, 72).

8

ACRONYMES

DW	Data Warehouse . . . 3
OMG	Object Management Group . . . 4
CWM	Common Warehouse Metamodel . . . 4
CCReF	Commun Case Reuse Framework . . . 6
IDM	Ingénierie Dirigée par les Modèles . . . 11
BI	Business Intelligence . . . 3
MDE	Model Driven Engineering . . . 12
MDA	Model Driven Architecture . . . 13
UML	Uniform Modelling Language . . . 13
MOF	Meta Object Facility . . . 13
CWM	Common Warehouse Model . . . 4
XML	Extensible Markup Language . . . 16
DTD	Document Type Definition . . . 16
DSL	Domain Specific Languages . . . 17
SPEM	Software Process Engineering Metamodel . . . 17
SysML	System Modeling Language . . . 18
EMF	Eclipse Modeling Framework . . . 24
XMI	XML Metadata Interchange . . . 59
OCL	Object Constraint Language . . . 22
ECD	Extraction de Connaissances à partir de Données . . . 98
IA	Intelligence Artificielle . . . 140

IC	Ingénierie des Connaissances 140
RaPC	Raisonnement à Partir de Cas 139
CCReF	Common Case Reuse Framework 6
DOI	Degree Of Interest 161
ETL	Extract Tranform Load 40
RDF	Resource Description Framework 51