الجمهوريــة الجزائريــة الديمقراطيــة الشعبيــة
**République Algérienne Démocratique et Populaire**
وزارة التعليم العالي و البحث العلمي
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université de 20 Août 1955-Skikda**
**Faculté des Sciences**
**Département d'informatique**
**Ref :**


**جامعة 20 أوت 1955 سكيكدة**
**كلية العلوم**
**قسم الإعلام الآلي**
**المرجع:**

## Thèse

Présentée en vue de l'obtention du diplôme de

**Doctorat en sciences**

Spécialité

**Informatique : Intelligence Artificielle**

## Titre

# Patrons de représentation et correspondance des ontologies Multipoints de vues : Conception et Exploitation pour intégrer les bases de données à base ontologique

Présentée par :

**Soumaya Kasri**

Soutenue publiquement le : 20 / 06 / 2019

**Devant le jury composé de** :

| | | |
|---|---|---|
| Bachir Boucheham | Professeur à l'université 20 Août 1955-Skikda | Président du jury |
| Smaine Mazouzi | MCA à l'université 20 Août 1955-Skikda | Examinateur |
| Hassina Seridi | Professeur à l'université de B. Mokhtar, Annaba | Examinatrice |
| Mourad Bouzenada | MCA à l'université de Constantine 2 | Examinateur |
| Djamel Eddine Saidouni | Professeur à l'université de Constantine 2 | Examinateur |
| Fouzia Magra-Benchikha | Professeur à l'université de Constantine 2 | Directrice de thèse |

**Année universitaire 2018/2019**

# *Merci à*

**ALLAH** tout Puissant, Maître des cieux et de la terre qui m'a permis de mener à bien ce travail et ce niveau d'études.

Pr. Fouzia Magra-Benchikha pour avoir accepté de m'encadrer et dirigé cette thèse et pour ses conseils, orientations et échanges tout au long de cette thèse.

Pr. Bachir Boucheham pour avoir accepté de présider le jury de cette thèse.

Dr. Smaine Mazouzi, Pr. Hassina Seridi, Pr. Djamel Eddine Saidouni et Dr. Mourad Bouzenada pour avoir accepté d'examiner mon travail.

Tous mes collègues du département informatique et plus particulièrement Samia, Amina et Hanane pour tous les bons moments passés en leur compagnie.

Toi, lecteur, qui aura le cœur de dépasser cette page.

Mon père, ma mère, mes frères et mes sœurs ainsi que mes neveux : Amani, Aya, Anas, Noor, Abederahmen, Ibtihel, Yahya et Massara. Je leur dédie cette thèse.

## *Résumé*

Ce travail de recherche concerne l'apport de l'intégration de la notion de point de vue dans les ontologies en utilisant les patrons de conception d'ontologie. L'objectif principal de ce travail est de développer un patron de conception d'ontologie d'architecture et des correspondances afin de construire une ontologie multipoints de vue. Un Framework de validation de notre patron de conception est proposé en utilisant l'extension de l'Analyse Formelle de Concepts (structures de patrons). Le patron est utilisé par la suite dans un système d'intégration hybride à base ontologie multipoints de vue afin d'offrir un accès par point de vue aux données. Les ontologies locales sont restructurées par notre patron proposé et l'Analyse Relationnelle de Concepts avant les intégrer afin de les rendre homogènes.



**Mots clés :** ontologie, intégration de données à base ontologie, patron de conception d'ontologie, analyse formelle de concepts, analyse relationnelle de concepts, structures des patrons.



## *ملخص*

يتعلق هذا العمل البحثي بالإسهام في دمج مفهوم وجهة النظر في الأنطولوجيا باستخدام أنماط تصميم الأنطولوجيا. الهدف الرئيسي من هذا العمل هو تطوير نمط تصميم هيكلي و أنماط تصميم رابطية من أجل بناء أنطولوجيا متعددة الاوجه.. يقترح إطار التحقق من نمط التصميم الخاص بنا باستخدام امتداد التحليل الرسمي للمفاهيم (بنية النمط). يُستخدم النموذج لاحقًا في نظام دمج هجين قائم على أساس الأنطولوجيا متعددة الاوجه لتوفير إمكانية الوصول إلى البيانات من وجهة نظر معينة. يتم إعادة هيكلة الأنطولوجيات المحلية من خلال نمط التصميم المقترح والتحليل العلائقي للمفاهيم قبل دمجها لجعلها متجانسة.



**.الكلمات المفتاحية**: انطولوجيا ، تكامل البيانات ، نمط تصميم الأنطولوجيا ، تحليل الشكلي للمفاهيم ، تحليل العلائقي للمفاهيم ، بنية الأنماط.



## *Abstract*

This research work concerns the contribution of the integration of the view point notion in ontologies by using ontology design patterns. The main goal of this work is to develop an architectural ontology design pattern and correspondence design patterns in order to build a multiviewpoints ontology. A validation framework of our design pattern is proposed using the extension of the Formal Concepts Analysis (pattern structures). Subsequently, our pattern is used in hybrid *Multiviewpoints Ontology-based data integration* system to provide an access by view point to data. In this system, local ontologies are restructured by our proposed pattern and the Relational Concepts Analysis to make them them homogeneous.



**Keywords**: ontology, data integration, ontology design pattern, formal concept analysis, relational concept analysis, pattern structures.

# Sommaire

# Chapitre 3 : L'Analyse Formelle de Concepts et Ontologies



# Partie2 : Contributions

# Chapitre 4 : Patron de Conception d'Ontologie Multipoints de Vue

## Chapitre 5 : Une Ontologie Multipoints de Vue pour Intégrer les Bases de Données à Base Ontologique



## Chapitre 6 : Expérimentation et Implémentation

# Liste des Figures

# Liste des Tableaux

# INTRODUCTION GÉNÉRALE

Nous présentons ici notre contexte général de recherche. Nous évoquons les problématiques au niveau de l'ingénierie des ontologies, l'accès et la modélisation par point de vue et aussi la modélisation par patron de conception. Ensuite, nous définissons l'objectif de notre thèse ainsi que les différentes contributions. Enfin, nous présentons l'organisation de notre manuscrit.

## Contexte Général de Recherche et Motivation

Depuis la naissance de l'intelligence artificielle (IA), plusieurs techniques de représentation de connaissances ont vu le jour. Elles consistent à créer des modèles pour représenter et interpréter les différents éléments du monde réel. L'IA a vu apparaître différents modèles de représentations telles que la logique de prédicats, les systèmes de production, les réseaux sémantiques, les objets, les frames, etc. Les travaux menés dans la dernière décennie ont fait émerger la notion d'ontologie en tant qu'un nouveau modèle de représentation de connaissances. Elle est définit comme conceptualisation consensuelle et partageable d'un domaine (Gruber, 1993).

Un nombre croissant d'applications de domaines variés utilisent les ontologies pour décrire de façon consensuelle, générique et formelle leurs informations pertinentes. Cette utilisation génère un gros volume de données qui doit être stocké, interrogé, traité, intégré, maintenu, etc. En conséquence, des solutions de persistance s'appuyant sur des SGBDs existants ont été proposées (Alexaki, Christophides, Karvounarakis, Plexousakis, & Tolle, 2001). Elles ont fait naître un nouveau type de bases de données appelé *Base de Données à Base Ontologique* 'BDBO'.

Le développement d'une ontologie s'appuie sur les mêmes principes que ceux appliqués pour développer un composant logiciel. Cette tâche est difficile et nécessite la mise en place de procédés élaborés afin d'extraire la connaissance d'un domaine donné (Borst, 1997) (Corcho, Fernández., Gómez-Pérez., & López-Cima., 2005). A l'heure actuelle plusieurs méthodologies de développement d'ontologies existent et qui permettent leur construction 1) à partir du début, 2) par intégration d'autres ontologies, 3) par réingénierie ou 4) par construction collaborative (Psyche, Mendes, & Bourdeau, 2003).

La modélisation de la connaissance d'un domaine, ou la conception d'une ontologie, s'avère particulièrement difficile, surtout si l'on souhaite modéliser un domaine complexe où différentes disciplines sont impliquées dans sa description. Les modèles conceptuels résultants sont hétérogènes et dépendent du point de vue du concepteur du monde et de ses habitudes de modélisation et sa terminologie usuelle (Falquet & Mottaz, 2002) (Bach, 2006) (Hemam, 2012). Cette hétérogénéité de modélisation des connaissances permet de concilier, pour les mêmes connaissances, les quatre notions antagonistes : la modélisation consensuelle des connaissances versus la modélisation par point de vue des connaissances et la modélisation par patron versus la modélisation sans patron.

En effet, l'une des caractéristiques essentielles de l'ontologie est qu'elle fournit des connaissances consensuelles et partagées par une communauté sur un domaine donné (Gruber, 1993). L'aspect consensuel qui la caractérise permet de faciliter sa réutilisation future en offrant une vue globale du domaine d'intérêt. Cependant, le développement d'une ontologie peut être abordé selon de multiples points de vue. D'après Sowa (Sowa, 1997), la construction d'ontologie est une méthode pour extraire un catalogue des choses ou des entités(C) dans un domaine(D) selon la perception (le point de vue) d'une personne qui sert d'un certain langage (L) pour le décrire. Le concept de point de vue a été utilisé avec des sens divers dans le domaine informatique tels que les bases de données, les systèmes de représentation de connaissances par objet, l'analyse et la conception, la programmation (Marino, 1993) (Marcaillou, 1995) (Benchikha & Boufaida, 2007) …etc.

La construction d'une ontologie qui modélise simultanément plusieurs aspects d'un domaine complexe est extrêmement difficile. En effet, plusieurs ontologies qui modélisent des vues particulières ou partielles sont développées séparément et coexistent avec les risques d'incohérence associés (Lisi, & Esposito, 2014) (Ou, Pekar, Orasan, Spurk, & Negri, 2008). Ce qui incite à créer une seule ontologie multipoints de vue réutilisable. C'est l'un des objectifs des travaux présentés dans cette thèse. En ingénierie ontologique, les ingénieurs d'ontologie utilisent l'héritage multiple pour exprimer le fait qu'une instance peut être attachée à plusieurs concepts. Cependant cette solution d'utiliser une relation de spécialisation multiple oblige à créer de nombreux concepts creux pour rendre toutes les combinaisons de point de vue possibles.

D'une autre part, la construction d'une ontologie de qualité est un enjeu important pour l'ingénierie des connaissances. Une ontologie de qualité facilite l'évolution et la maintenance (enrichissement, correction de bogues de conception, intégration, etc.) (Daga et al., 2010) (Gangemi, 2005). Au cours du processus de construction d'ontologie, il est fréquent certains problèmes soient récurrents (représentation des contextes, des points de vue, des relations n-aire, etc.). Puisque les outils et les langages utilisés pour représenter les connaissances (ontologies) sont moins expressifs, il est plus que probable que ces problèmes amènent à des solutions de conceptualisation similaires. Des recherches récentes considèrent les patrons comme la panacée aux problèmes de conception d'ontologie (Presutti, Daga, Gangemi, & Blomqvist, 2009) (Rodriguez-Castro, 2012) (Shimizu, Hitzler, Paul, 2018) (Alirezaie, Hammar, & Blomqvist, 2018). En effet, les patterns sont utilisés dans plusieurs domaines. Ils ont d'abord été mentionnés dans le domaine d'architecture par le mathématicien et l'architecte Christopher Alexander en 1977 (Alexander, Ishikawa, & Silverstein, 1977). En génie logiciel, un patron de conception est un concept destiné à résoudre les problèmes récurrents suivant le paradigme objet. Les patrons de conception décrivent des solutions standards pour répondre à des problèmes d'architecture et de conception de logiciels. Ils ont été formalisés pour la première fois en 1995 dans le livre du «Gang of Four» (Gamma, Helm, Johnson, & Vlissides, 1995). L'un des objectifs principaux de cette thèse est d'utiliser les patrons de conception pour l'intégration de la notion de point de vue dans l'ontologie.

## Objectifs et Contributions

Dans le web sémantique, il existe plusieurs outils et langages pour construire une ontologie comme RDF/RDFS et OWL qui sont des langages de représentation de connaissances. Cependant ces outils et langages n'offrent pas la possibilité de représenter la notion de point de vue.

Pour intégrer les points de vue dans l'ontologie, il est possible de développer un nouveau langage d'ontologie pour les supporter. Cependant, deux raisons nous ont dissuadés de nous engager dans cette démarche et ont motivé notre recherche. La première est liée à la possibilité d'appliquer le patron à n'importe quel langage d'ontologie donc nous gagnons le coût de développement d'un nouveau langage et le temps consacré pour l'apprendre et l'utiliser. La seconde raison est liée aux efforts dépensés pour adapter ou bien créer de nouveaux éditeurs ou raisonneurs pour ce nouveau langage. Pour cela, il est très souhaitable d'utiliser le même langage, formalisme, outils pour manipuler (exprimer, exploiter, raisonner) le concept multipoints de vue. Pour les raisons citées ci-dessus. Nous proposons un patron de conception d'ontologie nommé patron multipoints de vue (MV2P).

L'objectif de notre recherche est double. Nous fournissons tout d'abord un patron de conception d'ontologie pour intégrer la notion de point de vue comme une connaissance dans l'ontologie. Par la suite, ce patron sera utilisé dans un processus d'intégration hybride à base d'ontologie intégrant les points de vue comme une connaissance dans le schéma global du système d'intégration. Le but est d'offrir un accès par point de vue aux données par les utilisateurs. Les ontologies locales sont restructurées par notre patron proposé afin de les rendre structurées par point de vue.

Les questions auxquelles nous essayons de répondre sont :

1. Comment résoudre le problème d'intégration de la notion de point de vue dans la modélisation d'une ontologie ?
2. Comment valider le patron de conception d'ontologie proposé ?
3. Comment restructurer une ontologie par un patron de conception d'ontologie ?
4. Comment utiliser et quel est l'avantage d'utiliser le patron proposé dans un processus d'intégration des données ontologiques ?
5. Comment valider le travail ?

Donc, notre contribution est répartie en quatre parties.

1. Le premier apport de cette thèse est relatif à l'intégration de la notion de point de vue dans la représentation des connaissances ontologiques. Nous proposons un patron de conception d'ontologie d'architecture et un ensemble de patrons de correspondances pour intégrer les points de vue dans la structure de l'ontologie. Le patron de conception multipoints de vue (MV2P) permet une représentation multiple des concepts ontologiques à partir de laquelle chaque utilisateur de l'ontologie va pouvoir voir son propre point de vue sur un concept donné.
2. Le deuxième apport consiste à proposer un Framework de validation du patron de conception multipoints de vue. Dans la littérature, il n'existe pas de méthodologie rigoureuse pour valider un patron de conception d'ontologie, mais seulement la

vérification de certains critères, telle que la réutilisation. Pour cela, nous proposons un Framework de validation basé sur l'extension de l'analyse formelle des concepts (structure de patrons) pour spécifier formellement le patron.

3. Le troisième apport de cette thèse est relatif à la proposition de l'approche de restructuration des ontologies en se basant sur le patron MV2P et l'analyse relationnelle de concepts (ARC). Les ontologies résultantes sont des ontologies multipoints de vue et la restructuration est utilisée pour construire une harmonisation virtuelle des ontologies.
4. Le quatrième apport consiste à proposer une approche d'intégration hybride des données ontologiques en se basant sur le patron multipoints de vue et l'analyse relationnelle de concepts. Ce système d'intégration prend le patron multipoints de vue comme un modèle d'accès multipoints de vue et aussi comme un modèle d'harmonisation virtuelle pour les bases de données à base ontologique locales. Le processus d'intégration des ontologies est un processus complexe et très couteux en temps et en main d'œuvre. Il est généralement construit manuellement ou avec une grande interaction et intervention de l'ingénieur d'ontologie. L'objectif est d'aider les ingénieurs à construire une ontologie globale dans un système d'intégration en proposant une approche semi-automatique par l'utilisation des techniques de l'Analyse Formelle de Concepts (AFC).

## Organisation du Manuscrit

Ce mémoire est organisé en deux parties.
La première partie, présentant les états de l'art, comprend trois chapitres.
Le premier chapitre présente la technologie des ontologies, les bases de données à base ontologique ainsi que la notion de point de vue.
Le deuxième chapitre présente un état de l'art sur les patrons de conception ; leurs types, leur utilisation en ingénierie des ontologies. En particulier, quelques patrons pour construire, aligner et intégrer les ontologies sont présentés.
Le troisième chapitre est lui consacré aux principes de l'analyse formelle de concepts (AFC) et des applications en ingénierie ontologique. Il présente différents algorithmes et outils de construction de treillis ainsi qu'un ensemble d'extensions de l'AFC et quelques approches qui couplent l'ontologie et l'AFC.
La deuxième partie de ce manuscrit présente l'ensemble de nos contributions. Elle contient les trois chapitres suivants :
Le chapitre 4 explicite nos patrons de conception d'ontologies (le patron d'architecture : patron multipoints de vue et les patrons de correspondance qui lient les vues) et porte aussi une attention particulière à la validation du patron par l'extension de l'analyse formelle de concepts « structures de patrons ».
Dans le chapitre 5, nous nous focalisons sur la présentation de l'approche d'intégration des ontologiques proposée. Nous détaillons les différentes étapes de l'approche, en particulier restructuration des ontologies locales aux ontologies multipoints de vue en se basant sur le patron de conception d'ontologie proposé et l'ARC.

Le chapitre 6 présente une mise en œuvre de nos propositions.
Ces travaux ont donné lieu à trois publications :

1. Kasri, S. and Benchikha, F. (2016) 'Refactoring ontologies using design patterns and relational concepts analysis to integrate views: the case of tourism', Int. J. Metadata, Semantics and Ontologies, inderscience. Vol. 11, No. 4, pp.243–263.
2. Kasri, S., and Benchikha F. (2014). 'Integrating Multi-viewpoints Paradigm in Ontology Using Ontology Design Patterns'. ADBIS (2), Ohrid, Macedonia. Advances in Intelligent Systems and Computing 312, Springer. (pp 257-268).
3. Kasri, S., Benchikha, F. (2014). 'Multi-viewpoints Ontology Design Pattern and its Evaluation with Pattern Structures'.7th IADIS International Conference Information Systems, Spain.

Chapitre 1

# Ontologie et Notion de Point de Vue

## Sommaire



## 1 Introduction

Le Web sémantique est une nouvelle génération du web. Il a été introduit par Tim Berners (Berners-Lee, Hendler, & Lassila, 2001). L'idée de Tim consiste à rendre les ordinateurs capables de traiter de manière experte les pages Web et rendre les données partageables et traitables par des outils automatisés. Il s'agit de décrire ces ressources selon une spécification formelle et partageable par une communauté appelée ontologie (Gruber, 1993). Actuellement, l'utilisation des ontologies est devenue vitale. Elle rend les données accessibles aux machines sous forme de concepts.

Le mot ontologie est emprunté du latin scientifique « Ontologia », lui-même composé à l'aide onto-, tiré du grec ancien (ôn, ontos) « étant, ce qui est », et -logia, tiré du grec (logos) « discours, traité » qui désigne une discipline philosophique qui décrit la science de l'existant ou la science de l'être (Dictionnaire de l'Académie française). L'informatique a repris ce terme par analogie, pour nommer les représentations des connaissances qui permettent la spécification formelle et partageable de ce qui existe dans le monde. Le terme d'ontologie a été introduit en informatique dans les années 90 comme un ensemble de termes et de relations de base comportant le vocabulaire d'un domaine ainsi que des règles pour combiner les termes et les relations afin de définir des extensions du vocabulaire (Neches et al.,1999).

Depuis une dizaine d'années, le nombre d'ontologies qui sont développées et utilisées pour différentes applications et domaines ne cesse d'augmenter. L'ontologie est définie comme une spécification consensuelle d'une conceptualisation d'un domaine donné (Gruber, 1993). Cependant, la forte possibilité que les objets réels d'un domaine peuvent être vus différemment a entraîné le problème du caractère consensuel de l'ontologie. Un domaine peut donner lieu à plusieurs interprétations menant à la création de plusieurs ontologies différentes. Les ontologies créées sont des tentatives de modélisation de différents experts où chacun a son propre point de vue sur les objets du monde réel. Ce point de vue reflète son métier, ses connaissances et la tâche d'utilisation qu'il recherche accomplir. En conséquence, pour pouvoir intégrer dans une seule ontologie les points de vue de plusieurs experts sur un concept donné un nouveau type d'ontologie a été créé, appelé ontologie multipoints de vue qui supporte la notion de plusieurs vues pour la modélisation d'un même objet réel. Elle permet aussi aux utilisateurs selon leurs besoins de sélectionner tout ce qui les intéressent et pas plus.

Dans ce chapitre, nous présentons les principes fondamentaux des ontologies et la notion de point de vue. Dans la section 2, nous présentons quelques définitions de l'ontologie, ses caractéristiques, une classification des ontologies de domaine et quelques formalismes ontologiques existants. Nous présentons ensuite quelques applications des ontologies couplées au domaine des bases de données. Nous abordons, dans la section 3, l'intégration de la notion de point de vue dans les ontologies en présentant quelques systèmes jugés les plus représentatifs. La section quatre conclut ce chapitre.

# 2 Notion d'ontologie

Dans cette section, après avoir rappelé quelques définitions de l'ontologie, nous présentons les caractéristiques qui la distinguent des autres modèles informatiques. Une classification des ontologies de domaine sera ensuite exposée. Enfin, nous présentons quelques formalismes ontologiques existants (RDF (Brickley & Guha, 2002), OWL (Bechhoferet al., 2004) et PLIB (Pierra, 2008)) et quelques domaines d'application des ontologies.

## 2.1 Définition et caractéristiques

Selon Gruber (Gruber, 1993), en informatique, une ontologie est une spécification explicite d'une conceptualisation. Par la suite Borst et al (Borst, 1997) ont défini une

ontologie d'une façon plus élaborée comme une spécification explicite et formelle d'une conceptualisation partagée. Une spécification explicite signifie que les concepts et les relations d'un modèle reçoivent des noms et des définitions explicites. Formelle veut dire qu'elle est présentée dans un formalisme qui permet aux machines de faire des raisonnements. Conceptualisation partagée se réfère à un modèle abstrait et validé par une communauté (Studer, Benjamins, & Fensel, 1998).

Pierra (Pierra, 2008) définit aussi une ontologie de domaine comme : « une ontologie est une représentation formelle, explicite, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine sous forme de classes, de propriétés et de relations qui les lient». Cette définition met l'accent sur des caractéristiques importantes décrivant l'ontologie qui sont :

1. **Formelle** : l'ontologie est basée sur des théories formelles permettant à une machine de vérifier d'une façon automatique la consistance de certains éléments ontologiques et aussi la possibilité de faire un raisonnement automatique sur eux;
2. **Explicite** : la spécification des concepts de l'ontologie est faite explicitement et sans ambiguïté ;
3. **Référençable** : afin d'expliciter la sémantique des éléments ontologiques, chacun d'eux est associé à un identifiant unique permettant de le référencer à partir de n'importe quel environnement ;
4. **Consensuelle** : une ontologie est une conceptualisation validée et acceptée par une communauté qui fait un accord sur une conceptualisation partagée.

Stumme et al. (Stumme et al., 2003) définissent une ontologie du point de vue formel en précisant sa structure, ses composants et les types d'interactions possibles entre les composants. Donc, une ontologie est définie par $O = \{C, R, A, T, R, \_A, \leq R, \leq C\}$ avec :

- C : un ensemble de concepts ;
- R : un ensemble de relations ;
- A : un ensemble d'attributs ;
- T : un ensemble de types de données (entiers, booléens, dates, chaînes de caractères…etc.) ;
- _R : signature de relation. Elle est utilisée pour décrire les éléments de R.
- _A : signature d'attribut. Elle est utilisée pour décrire les éléments de A.
- ≤R : hiérarchie des relations (ordre partiel).
- ≤C : hiérarchie des concepts (ordre partiel).

Dans la littérature, la majorité des travaux s'accordent sur le fait que quelque soit les définitions de l'ontologie, celle-ci repose sur les composants élémentaires suivants :

- **Les concepts :** Ce sont des groupes d'individus partageant des caractéristiques communes et sont organisés dans une hiérarchie par une relation d'ordre partiel qui est la relation de subsomption (”est-une”). Dans l'exemple (voir la figure 1.1), les concepts

sont montrés par des ellipses (*Location*, *Hotel*, *Room.*…etc.). Les concepts ontologiques peuvent être classés en deux catégories (Gruber, 1993):

- Les concepts primitifs : la définition de ces concepts dans l'ontologie n'est pas complète. Elle présente seulement des conditions d'appartenance d'une instance. Pour des définitions complètes, on se base sur un savoir communautaire préexistant ou des documentations contextuelles ;
- Les concepts définis : la définition de ces concepts est exprimée par des conditions nécessaires et suffisantes en termes d'autres concepts primitifs ou définis fournis par l'ontologie.

- **Les propriétés (attributs) :** Ce sont des caractéristiques attachées aux concepts qui peuvent prendre des valeurs. Dans l'exemple de la figure 1.1 '*HasPhone*' est une propriété du concept '*Hotel*' qui prend un numéro de téléphone d'hôtel comme valeur.
- **Les relations :** Ce sont des liens entre les concepts qui expriment des interactions entre leurs instances. Dans la figure 1.1 '*hasLocation*', '*hasRooms*' par exemple sont des relations. La relation de subsomption « est-un » est une relation particulière qui permet de structurer les concepts ontologiques dans une hiérarchie. La relation de méronymie, « partie-tout » est aussi une relation particulière qui permet de structurer les concepts dans une hiérarchie.
- **Les axiomes :** Ce sont des déclarations qui sont toujours vraies. Par exemple si le prix d'allocation de chambre est réduit à 50% alors les demandes de réservation seront augmentées.
- **Les instances :** Ce sont des individus spécifiques d'un concept comme '*Sheraton_Hotel*', '*Biltemore_Hotel*' (voir la figure 1.1) qui représentent des instances du concept '*Hotel*'.

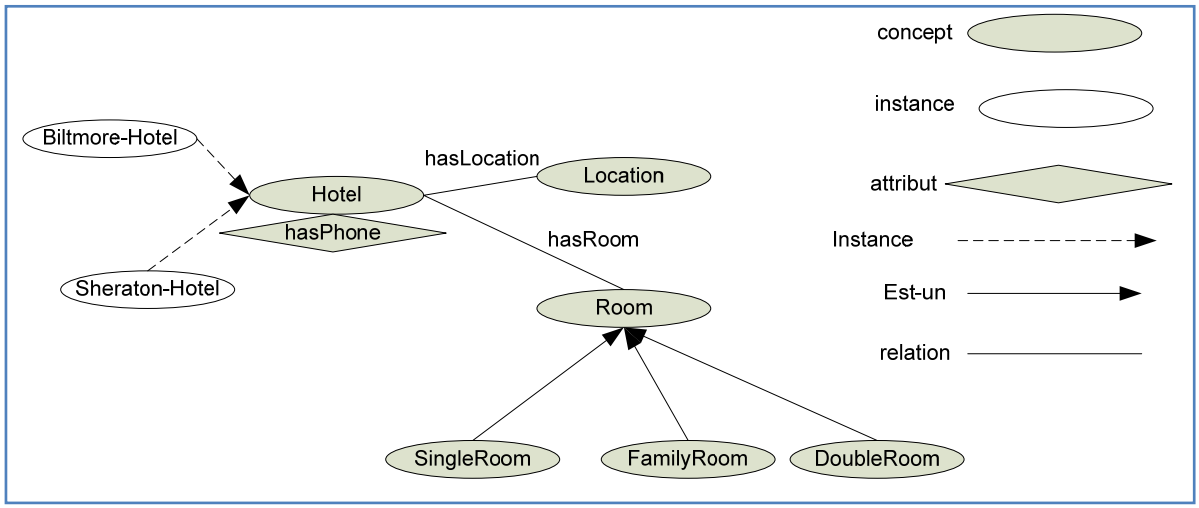



**Figure 1.1** Exemple d'un extrait d'une ontologie de tourisme

## 2.2 Types d'ontologie

En se basant sur les objets modélisés et suivant le degré d'abstraction visé, quatre types d'ontologies sont identifiés (Guarino, 1998):

- **les ontologies globales** (Top-Level Ontology) : dites aussi de haut niveau ou génériques. Ce sont des ontologies formelles et universelles qui présentent un haut niveau de généralité et d'abstraction. Elles sont généralement conçues pour une utilisation très générale et pour servir de base dans le développement des ontologies locales spécifiques à des domaines donnés. Un exemple de ce type est le projet KRAFT (Visser, Beer, Bench-Capon, Diaz, & Shave, 1999) ou WordNet[1] (Miller, 1995);

- **Les ontologies de domaine** : ce sont des ontologies qui décrivent un vocabulaire de domaines donnés (tourisme, médecine, etc.). Le niveau d'abstraction dans ces ontologies est moins élevé que celui des ontologies globales. L'ontologie des Uniprot[2] en est un exemple ;
- **Les ontologies de tâche** : ce sont des ontologies qui sont spécifiques à une tâche générique et indépendante du domaine d'application comme par exemple l'achat, la vente…etc.
- **Les ontologies d'application :** Ce sont des ontologies qui sont spécifiques à un domaine d'application précis donc elles sont souvent des spécialisations des ontologies de domaine, de tâche ou d'application.

Nous nous intéressons dans nos travaux aux ontologies de domaine. Dans la suite, nous présentons une classification de ce type d'ontologies.

### 2.3 Classification des ontologies de domaine

Deux catégories d'ontologies de domaine qui ont émergé dans la littérature sont distinguées (Pierra, 2008):(1) les ontologies linguistiques (OL) qui représentent les termes d'un domaine selon les différents langages naturels et aussi les relations entre eux (synonymes, hyponymes…etc.) et (2) les ontologies conceptuelles (OC) qui représentent les classes d'objets et leurs attributs pour un domaine donné. Deux catégories de l'ontologie conceptuelle (OC) sont distinguées : l'ontologie conceptuelle canonique(OCC) et l'ontologie conceptuelle non canonique(OCNC).

#### 2.3.1 Les ontologies linguistiques(OL)

Ce sont des ontologies qui définissent des termes d'un domaine donné éventuellement en plusieurs langues naturelles. Elles sont orientées vers les traitements automatiques de langage naturel (TALN) telle que WordNet de l'Université Princeton (Miller, 1995) qui vise à définir le sens des mots et les relations linguistiques entre eux (synonyme, antonyme, hyperonyme, etc.). WordNet utilise les relations portant des mesures de similarité pour capturer approximativement et semi-formellement les relations entre les mots. La construction de telle ontologie utilise généralement un processus d'extraction de termes dans un corpus textuel et sont classifiés par la suite par un expert de domaine (Teguiak, 2012). Plusieurs outils existent permettant de construire des ontologies à partir du texte tel que Text2Onto (Cimiano &

[1] Dictionnaire de la langue anglaise, http://www.cogsci.princeton.edu/wn
[2] Universal Protein Resource : http://www.uniprot.org/

Völker, 2005) et Terminae (Brigitte, Sylvie, & Clément, 1999). Certaines approches utilisent les OLs pour intégrer semi-automatiquement plusieurs sources de données. Le projet MOMIS (Beneventano et al., 2000) est un exemple d'une approche qui utilise une OL pour construire semi-automatiquement le schéma global à partir des différentes sources de données.

### 2.3.2 Les ontologies conceptuelles (OC)

Contrairement à l'ontologie linguistique, l'ontologie conceptuelle est une ontologie visant à représenter des concepts généraux sans préciser le langage de formalisation. Il y a deux types de concepts qui peuvent être identifiés dans une ontologie conceptuelle : les concepts canoniques ou primitifs qui n'ont pas des définitions complètes et les concepts définis ou non canoniques qui sont des concepts « pour lesquels une ontologie fournit une définition axiomatique complète au moyen de conditions nécessaires et suffisantes exprimées en termes d'autres concepts primitifs ou eux-mêmes définis » (Gruber, 1993). L'ontologie Dublin Core[3] est un exemple d'une ontologie conceptuelle. Le tableau suivant est tiré du travail (Pierra, 2008) faisant une comparaison entre une ontologie conceptuelle et ontologie linguistique.

| | **OL** | **OC** |
|---|---|---|
| **Elément** | Mot | Concept |
| **Indentification des éléments** | Mot | GUI |
| **Définition des éléments** | Phrase | Modèle |
| **Taille de l'ontologie** | Large | Minimale |
| **Relations** | Formelles+Linguistiques | Algébriques |
| **Contenu** | Eléments primitifs + Définitions Conservatives | Eléments primitifs + Définitions Conservatives |
| **Focus** | Orienté Classe | Orienté Propriété |
| **Développement** | Semi-automatique | Manuel |
| **Usage de l'ontologie** | Assisté par ordinateur | Automatique |

**Tableau 1.1** Comparaison entre ontologie linguistique et ontologie conceptuelle (Pierra, 2008)

[3] http://dublincore.org/

Une ontologie conceptuelle peut être de type canonique ou non canonique :

- **Les Ontologies Conceptuelles Canoniques (OCC**) : ce sont des ontologies qui ne contiennent que des concepts primitifs dits aussi atomiques c.-à-d. des définitions sans redondance. Les ontologies OCC décrivent chaque concept de domaine d'une manière unique. Pour définir des ontologies canoniques, des modèles d'ontologie ont été proposés comme RDFS (Brickley & Guha, 2002) et PLIB (Pierra, 2003) (Pierra, 2008) Ces ontologies présentent une base formelle pour une modélisation et un échange efficace des connaissances d'un domaine donné. L'ontologie Dublin Core (Weibel, 1997) est un exemple type d'ontologie canonique.
- **Les Ontologies Conceptuelles Non Canoniques (OCNC) :** Ce sont des ontologies qui contiennent dans leur définition des duplications de concepts et des relations d'équivalence entre les concepts. Ces relations d'équivalence permettent de définir des concepts non canoniques. Par conséquent, ces ontologies contiennent à la fois des concepts primitifs et définis. Elles permettent de définir des concepts identiques différemment en utilisant des opérateurs ensemblistes (union, intersection, différence) et d'autres opérateurs portant sur les propriétés pour restreindre leurs valeurs (restrictions) ou sur la cardinalité.

Les trois ontologies OL, OCC et OCNC peuvent être combinées en un modèle en oignon (Jean, Pierra, & Ait-Ameur, 2007). Dans ce qui suit nous décrivons les différentes couches de ce modèle.

### 2.3.3 Modèle en oignon

L'ontologie linguistique, canonique et non canonique peuvent être combinées en un modèle en couche appelé modèle en oignon (Jean, Pierra, & Ait-Ameur, 2007) (*cf* Figure. 1.2). Ce dernier définit une ontologie de domaine en couches (OCC, OCNC, et OL) complémentaires.

La couche de base de ce modèle est constituée par la couche canonique OCC qui fournit une base formelle pour modéliser et échanger d'une manière efficace des connaissances d'un domaine. Sur cette première couche, une seconde couche non canonique OCNC est construite. Celle-ci permet de représenter à la fois les conceptualisations d'un domaine et les correspondances entre elles. Finalement, l'ontologie linguistique forme la dernière couche qui représente les concepts d'un domaine en langage naturel ou en multilingue.

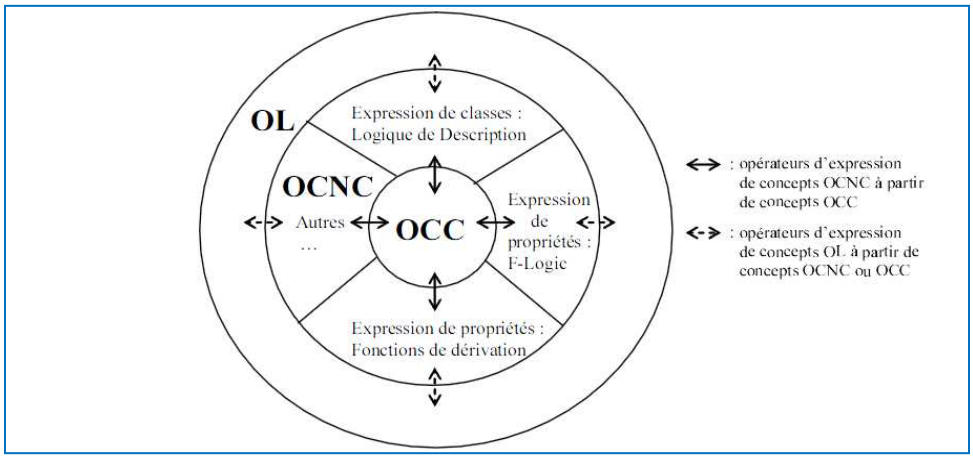





**Figure 1. 2** Le modèle en oignon pour les ontologies de domaine (Jean, Pierra, & Ait-Ameur, 2007)

Quelque soit le type de l'ontologie utilisé, l'expression de ses composants requiert un modèle ou formalisme ontologique qui permet un traitement automatique par une machine. De nombreux formalismes ont été proposés dans la littérature. Notre contexte de travail inclut les deux domaines de recherche : les bases de données et le web sémantique. Pour cela, nous avons choisi d'étudier : PLIB (Pierra, 2003) (Pierra, 2008) issu du domaine des bases de données et RDF/RDFS (Brickley & Guha, 2002) et OWL (Bechhoferet al., 2004) issus du domine du web sémantique.

### 2.4 Formalismes ontologiques

Naturellement, pour qu'une ontologie soit exploitable par les machines, il est nécessaire d'utiliser un langage de représentation formel pour spécifier les connaissances. Le web world wide consortium (W3C[4]) du web sémantique offre différents modèles de représentation pour les ontologies telles que RDF, RDFS et OWL...etc. D'autres modèles orienté ingénierie ont été aussi proposés comme PLIB (Pierra, 2003) (Pierra, 2008) dans le domaine des bases de données. Dans ce qui suit, nous décrirons le modèle PLIB (Pierra, 2008), RDF/RDFS (Brickley & Guha, 2002) et OWL (Bechhoferet al., 2004). Ces derniers diffèrent selon leur objectif d'utilisation.

Les modèles RDF/RDFS et PLIB sont adéquats pour définir des ontologies pour la gestion et l'échange des données. Ils permettent de définir la sémantique des concepts de manière unique en se basant sur les OCC. A son tour, le formalise OWL a été proposé pour la description d'ontologies en permettant la définition des correspondances entre vocabulaires et aussi la possibilité de déduction et d'inférence. Pour cela, OWL définit des concepts primitifs et définis.

#### 2 .4.1 Le langage RDF (Resource Description Framework) et RDF-Schéma (RDFS)

Il s'agit d'un langage qui permet de décrire les données sous forme d'un ensemble de triplets. Un triplet RDF (Brickley & Guha, 2002) est constitué de trois types de composants (sujet, prédicat, objet) :

- Le sujet est la ressource à décrire. Il est nécessairement identifié par une URI[5]. Une URI est une chaine de caractères utilisée pour identifier une ressource sur Internet.
- Le prédicat est une propriété utilisée pour caractériser le sujet. Le prédicat lui-même est considéré comme une ressource, et identifié par une URI.
- L'objet est une donnée ou une autre ressource identifiée par une URI.

(*Sheraton_Hotel,hasName,*"*SheratonAlge*r") , (*Sheraton_Hotel,hasLocation,Alger*) sont des exemple de triplets RDF représentés en figure 1.3.

[4] World Wide Web Consortium : consortium international pour la standardisation et la promotion des technologies du Web, http://www.w3.org/

[5] http://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/

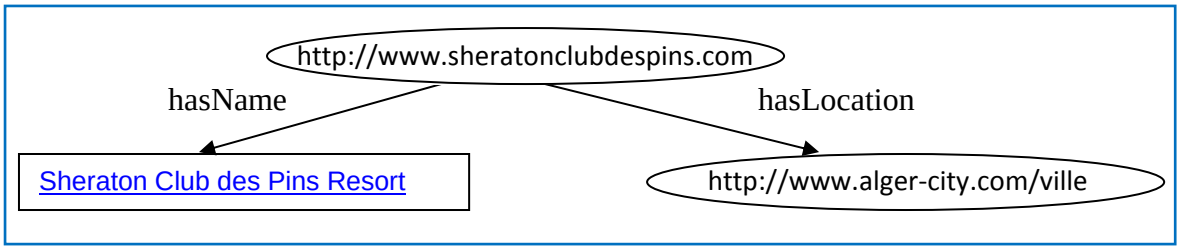



**Figure 1.3** Graphe RDF

RDF est le modèle de représentation de données le plus simple. Cependant, il définit très peu des prédicats et a très peu de sémantique et d'inférence. Il a donc été très rapidement complété par RDF-Schéma (Brickley & Guha, 2002).

RDF-Schéma (RDFS) est une extension sémantique de RDF. RDFS est un méta-modèle qui permet de spécifier les types de ressources par des classes (rdfs:Class) et de spécifier leurs propriétés (rdfs:property). La définition de classes et propriétés du langage RDFS est similaire aux systèmes de types des langages orientés objet. Il permet de "typer" une ressource à une propriété ou une classe définie en RDF Schéma (rdf:type). Il offre aussi le moyen de spécifier le typage des propriétés en indiquant leurs domaines (rdfs:domain) et leurs co-domaines (rdfs :range). Les membres d'une classe sont appelés des instances de la classe. Le langage RDFS permet aussi d'organiser les classes en une hiérarchie en utilisant rdfs:subClassOf et d'organiser les propriétés en une hiérarchie de propriétés en utilisant rdfs:subPropertyOf. Par exemple, la propriété '*hasRoom*' est définie avec un domaine '*Hotel*' et une valeur de type rdfs:Literal et aussi le domaine de la propriété '*hasRoom'* (voir la Figure 1.4) est la classe '*Hotel'* et son co-domaine est la classe '*Room*'.

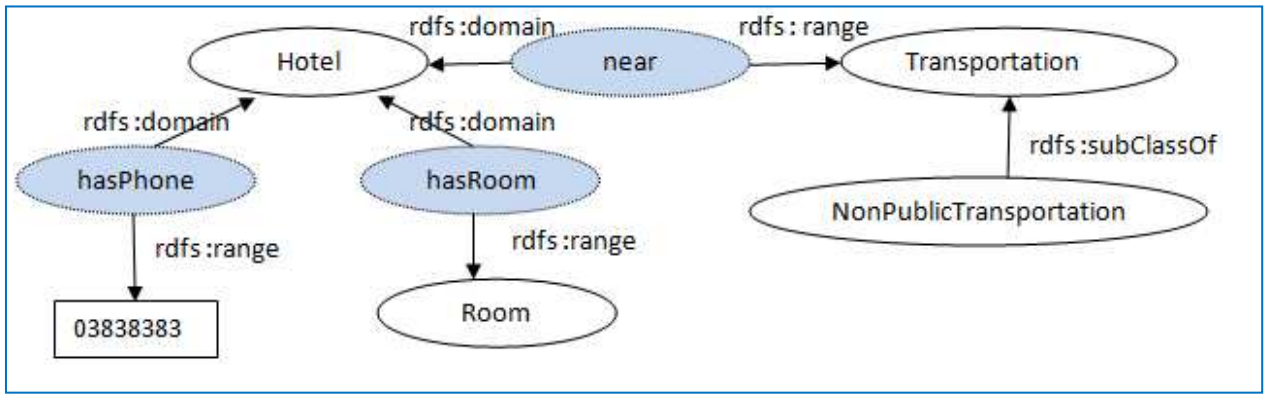



**Figure 1.4** Graphe RDFS

La puissance d'expression de RDF et de RDFS est très limitée. En effet, il est souvent nécessaire d'exprimer que deux classes sont disjointes, de créer des classes par combinaison ensembliste d'autres classes (intersection, union, complément), de mettre des contraintes de fonctionnalité et des contraintes de cardinalité maximale et minimale sur les propriétés, etc.. Ces limites ont donné naissance à un nouveau langage plus expressif et plus descriptif pour la représentation des ontologies, nommé OWL (Web Ontology Language). OWL a été recommandé par le W3C en 2004, jusqu'à l'arrivée de OWL 2.

### 2.4.2 Le langage OWL (Ontology Web Language (Bechhoferet al., 2004))

En 2004, une première version du langage OWL (Web Ontology Language) est apparue. Elle a été proposée comme standard normalisé par W3C pour la représentation des connaissances dans le domaine du web sémantique. OWL permet de définir des concepts en utilisant l'élément owl:Class. Une classe représente des ensembles d'individus définis par (owl :Individual) . OWL distingue deux types de propriétés. Celui reliant deux concepts (owl:ObjectProperty) et celui reliant un concept à un littéral (owl:DataProperty). Aussi, les classes peuvent être définies comme des combinaisons ensemblistes d'autres classes (intersection, union ou complément). OWL permet aussi de définir des axiomes. Par exemple, en utilisant un axiome owl:equivalentClass, on peut affirmer que deux classes sont équivalentes. Dans la littérature, trois sous-langages ont été associés à OWL, classés par ordre d'expressivité croissante (OWL Lite ⊂ OWL DL ⊂ OWL Full). Le pouvoir d'expression de OWL Lite reste encore limité. Il supporte seulement les contraintes simples comme la contrainte de cardinalité qui est limitée aux valeurs 0 et 1. Pour compléter ses lacunes, une deuxième version d'OWL appelée OWL DL a été proposée. OWL-DL garantit la décidabilité des raisonnements (Bizer, Heath, & Berners-Lee, 2009). Mais il reste incompatible avec RDFS. Pour cela, une troisième version appelée OWL Full a été proposée. Celle-ci permet aux utilisateurs de combiner à la fois la liberté syntaxique de RDF et le pouvoir d'expression d'OWL. Cependant, OWL Full reste peu utilisé à cause de l'indécidabilité des raisonnements. Les langages OWL-Lite, OWL- DL et OWL- Full sont des langages orientés web sémantique. Dans la section suivante, nous présentons le langage PLIB qui a été conçu pour décrire des ontologies canoniques pour modéliser, échanger et référencer des catalogues informatisés de composants ou objets techniques préexistants.

### 2.4.3 Le langage PLIB (Pierra, 2008)

Le langage PLIB (Parts LIBrary : série de normes ISO 13584) est un langage qui a été défini dans le cadre du projet PLIB pour la description d'ontologie canonique qui définit les catégories de composants industriels et leurs instances afin de les rendre échangeables entre fournisseurs et utilisateurs. Chaque concept de l'ontologie est identifié de manière unique par un GUI (Globally Unique Identifier) (Pierra, 2008). PLIB se concentre sur les propriétés pour identifier et définir précisément les concepts en associant à chacun une définition, une note, un nom, une image ou un document afin de fournir une ontologie canonique. Dans PLIB, toute instance possède une classe de base. Une instance est définie par (Pierra, 2005):

- un identifiant d'objet ;
- une référence à la classe de base de l'objet (par l'intermédiaire GUI) ;
- une liste de couples :
    - Référence à une propriété applicable à la classe (par son GUI).
    - Valeur de la propriété (type simple, identifiant d'objet ou collection).

PLIB utilise le langage de spécification de données EXPRESS (Schenk & Wilson, 1994) pour décrire les catalogues. EXPRESS est un langage d'expression de contraintes très puissant

qui assure l'intégrité des ontologies échangées. Dans ce contexte, un catalogue a une ontologie qui définit les catégories de composants industriels. Une ontologie conceptuelle canonique peut être crée par des constructeurs comme:

- Constructeur de classes de définition (item_class) : les classes créées par ce constructeur sont identifiées par un BSU (Basic Semantic Unit). Elles contiennent les propriétés essentielles.
- Constructeur de classe de représentation (functional_model_class) : les classes contiennent les propriétés qui n'ont de sens que par rapport à un point de vue donné;
- Le constructeur de classe de vue (functional_view_class) qui définit la perspective ou le point de vue selon lequel sont définies les propriétés des classes de représentation.
- Les propriétés en PLIB sont définies avec le constructeur (non_dependent_Pdet) et identifiées par un (BSU).

Le modèle PLIB fournit des types de données primitifs tels que les entiers ou les réels. Le type (class_instance_type) permet d'utiliser une classe comme un type de données. Deux opérateurs sont disponibles pour organiser les classes dans une hiérarchie. Le premier est l'opérateur (is_a) pour l'héritage simple et (case_of) pour la subsomption multiple. Ce dernier permet la construction modulaire d'ontologies de domaine où Une classe peut hériter d'une partie des propriétés des classes subsumantes. Pour réaliser l'instanciation multiple (multi-instanciation[6]) PLIB offre le mécanisme d'agrégation d'instance. Pour cela, il propose de distinguer les propriétés essentielles d'une classe de celles qui dépendent d'un point de vue sur le concept représenté. Cette distinction est mise en place via les trois constructeurs de classes suivants : (item_class), (functional_model_class) et (functional_view_class). Ainsi, une instance peut appartenir non seulement à sa classe de base mais également aux classes de représentation attachées à cette classe de base par la relation (is_view_of).

### 2.5 Langage d'interrogation des ontologies : SPARQL

La nécessité d'accéder facilement aux connaissances d'ontologie favorise le développement de langages d'interrogation d'ontologie. Un ensemble de langages de requêtes a été développé dans le contexte du Web Sémantique. Ces langages sont classés en sept catégories (SPARQL, RQL, langages inspirés de XPath, XSLT ou XQuery, langages avec règles réactives, langages déductifs et autres langages) (Bailey, Bry, Furche, & Schaffert, 2005). Le langage de requête SPARQL est une recommandation du W3C pour interroger les ontologies en format RDF, et devient par la suite le langage de requête standard (Jorge, Marcelo, & Claudio, 2009). Pour cette raison, il a été pris en charge par la plupart des outils de gestion et d'interrogation d'ontologie. Un exemple simple de requête SPARQL est donné ci-dessous :

[6] Le mécanisme d'instanciation multiple permet à un individu d'être une instance directe d'un ou de plusieurs concepts

```
PREFIX ex: <http://www.semanticweb.org/kasri/ontologies/2017/9/exempleHotel#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?s
WHERE {
    ?s ex:hasName "Sheraton Club des Pins Resort" ;
    rdf:type ex:Hotel;

}
```

SPARQL est un langage de requête basé sur la mise en correspondance des patrons de triplets. Le patron de triplet est un triplet RDF dans lequel un ou plusieurs des éléments (sujet, prédicat ou objet) peuvent être une variable commençant par "?". Le patron de triplet sert à trouver par un processus de mise en correspondance les triplets RDF qui satisfont le modèle qu'il définit. Prenons par exemple le patron de triplet (?s, rdf:type, ex :Hotel). La mise en correspondance de ce patron de triplet consiste à trouver les valeurs de ?s pour lesquelles le triplet appartient au graphe RDF. La combinaison de plusieurs patrons de triplets donne un patron de graphe (Calì, Frosini, Poulovassilis, & Wood, 2014).

Une requête SPARQL peut être de quatre types différents, SELECT, ASK, CONSTRUCT, ou DESCRIBE.

- SELECT: ce type de requête ne retourne que les informations qui vérifient les contraintes spécifiées dans la requête.
- ASK: retourne vrai ou faux en fonction de la présence ou non du patron de triplets demandé dans le document RDF.
- CONSTRUCT: ce type de requête retourne les réponses, vérifiant un ensemble de contraintes sous forme de graphe RDF.
- DESCRIBE: ce type de requête retourne des informations sur les résultats de la requête sous forme de graphe RDF.

Après la présentation de tous les formalismes de description d'ontologies, on peut remarquer que ces modèles d'ontologie partagent des points en commun. En effet, chacun utilise les classes et les propriétés pour conceptualiser un domaine donné. Chaque concept est identifié par un identifiant unique permettant d'être référencé par une autre ontologie. Les classes sont organisées en hiérarchie. Certaines différences entre ces modèles sont aussi relevées. Par exemple, le langage de définition RDFS et PLIB sont canoniques, c'est-à-dire que les concepts sont primitifs aves des définitions sans redondances. Par contre, OWL utilise les deux types de concepts primitifs et définis (non canonique). RDFS et OWL définissent la hiérarchisation des classes en utilisant la relation de subsomption. De plus, OWL utilise aussi l'équivalence. A son tour, PLIB utilise la relation est-un- cas-de pour hiérarchiser les concepts. Pour la multi-instanciation RDFS et OWL utilisent l'héritage multiple. Dans le modèle PLIB, la multi-instanciation est remplacée par l'agrégation d'instances. Pour les opérateurs, chacun possède des opérateurs spécifiques aux problèmes à résoudre (operateur d'union pour OWL DL et de modularité pour PLIB). Et pour les contraintes, OWL DL permet de mettre des contraintes de cardinalités et PLIB permet de mettre des contraintes d'intégrités.

## 2.6 Quelques domaines d'application

Dans le contexte de notre recherche, nous nous intéressons particulièrement à l'utilisation des ontologies dans le domaine de représentation et d'accès aux données. Traditionnellement, l'utilisation d'une base de données est primordiale pour le stockage de données de grande taille. D'après Germán (Germán, 2004), l'utilisation des bases de données garantie l'indépendance l'intégrité et la sécurité des données. Elle permet aussi l'administration, l'accès et la récupération des données. Cependant, avec l'apparition du web sémantique et exactement l'apparition de l'ontologie, plusieurs propositions réclament celle-ci pour son caractère formel et sémantique. Le coté formel de l'ontologie permet l'automatisation et parfois l'amélioration des tâches comme le partage et l'accès aux donnés hétérogènes et distribuées.

Les ontologies et les bases de données sont similaires : les classes, les propriétés et les axiomes peuvent être interprétés par des tables, des attributs, et des contraintes respectivement. Aussi l'interrogation peut être effectuée par SPARQL ou SQL respectivement.

Actuellement, Plusieurs approches (Broekstra, Kampman, & Van, 2002) (Fankam, Bellatreche, Dehainsala, Ait-Ameur, & Pierra, 2009) (Hamaz & Benchikha, 2017) ont été définies pour coupler les ontologies avec les bases de données pour atteindre d'une part une structuration et gestion des données efficace avec une capacité de stockage importante en utilisant les bases de données et d'autre part une sémantique explicite et partageable avec les ontologies. Les travaux de recherche étudiant le couplage entre base de données et ontologie ont suivi d'une façon générale trois orientations de recherche : base de données à base ontologique, accès aux données à base ontologique, et intégration à base ontologique. Dans ce qui suit nous présentons ces orientations de recherche.

### 2.6.1 Base de données à base ontologique

Au cours des dernières années, les ontologies sont utilisées dans de nombreuses applications. Pour une base de données, l'utilisation d'ontologie permet la spécification sémantique de ses données de manière explicite (Pan & Heflin, 2003). Dans d'autres applications, les ontologies sont généralement définies par des langages de description textuelle et sont sauvegardées dans des fichiers. Cependant, avec la forte volumétrie de données, ces fichiers sont difficilement exploitables et traitables en mémoire centrale. Dans ce cas, la base de données est le meilleur support pour stoker ces ontologies (Alexaki, Christophides, Karvounarakis, Plexousakis, & Tolle, 2001) (Broekstra, Kampman, & Van, 2002) (Ma, Su, Pan, Zhang, & Liu, 2004). C'est dans ce contexte qu'un nouveau type de bases de données, appelé Bases de Données à Base Ontologique (BDDO), est apparu.

Dehainsala (Dehainsala, Pierra, & Bellatreche, 2007) définit une base de données à base ontologique comme une source de données contient :

1. Une Ontologie représentée par un ensemble de concepts (classes, propriétés…etc)
2. Les données qui représentent les instances de l'ontologie.

3. Un ensemble des liens entre les données et leurs concepts qui définissent leur sémantique.

En se penchant sur la littérature, trois modèles de stockage ont été définis pour la persistance des ontologies et des données les référençant (Fankam, Bellatreche, Dehainsala, Ait-Ameur, & Pierra, 2009): vertical, binaire et horizontal. Nous les présentons dans ce qui suit.

- **Modèle de stockage vertical :** Dans la représentation verticale aussi appelée générique, les données sont stockées dans une unique table verticale de trois colonnes correspondant à un triplet (sujet, prédicat, objet). Cette table est dite table de triplets pour stocker l'ontologie où les triplets représentent respectivement l'identifiant de la ressource ontologique (classe, propriété ou instance ontologique), un prédicat de la ressource, et la valeur du prédicat. Chaque ressource ontologique R est définie par des triplets de l'une des deux formes suivantes :
  1. la forme *(R, rdf:type, entité)* : ces triplets permettent d'indiquer le type de ressource. Ce type est une des entités définies en RDFS (*rdfs:Class, rdfs:Property..)*
  2. la forme *(R, attribut, valeur)* : ces triplets permettent de caractériser les ressources définies en leur assignant des valeurs d'attributs RDFS.

  Cette représentation facilite l'insertion de nouveaux triplets dans la base de données. Cependant, lorsque les données deviennent de taille importante, l'interrogation devient coûteuse et complexe car elle impose un nombre élevé d'opérations d'auto-jointure. Et aussi dans la plupart du temps, la modification d'un triplet nécessite la modification d'un ensemble de triplets.
- **Modèle de stockage horizontal :** Il consiste à représenter le modèle d'ontologie en utilisant le modèle relationnel ou relationnel-objet supporté par le SGBD sous-jacent à la BDBO. Avec cette représentation, une table horizontale par classe ontologique est créée. Dans cette table, une colonne correspond à une propriété qui a comme valeur au moins une seule instance de la classe correspondante. Le schéma de la BDBO est défini en fonction des concepts du modèle d'ontologies supporté. Par exemple dans le modèle RDFS, on trouve les tables suivantes : *class, subclass, domain, range, property, subproperty.*

  Cette approche est efficace pour les requêtes portant sur les propriétés puisque chaque table se rapporte à une classe distincte. Cependant, il reste la question des propriétés associées aux valeurs nulles car certains individus n'ont pas certaines propriétés. Donc leur existence dans les tables créées serait très coûteuse. Les BDBO Sesame (Broekstra, Kampman, & Van, 2002), RSTAR (Ma, Su, Pan, Zhang, & Liu, 2004), OntoDB (Dehainsala, Pierra, & Bellatreche, 2007) et KAON (Park et al., 2007) utilisent cette représentation.
- **Modèle de stockage binaire :** Dans la représentation binaire, les relations sont décomposées en deux catégories : les tables unaires pour les classes de l'ontologie (table par classe) et les tables binaires pour les propriétés de l'ontologie (table par

propriété). Cette approche binaire se décline en trois variantes selon la façon dont elle est adoptée pour représenter l'héritage:

- une table unique pour toutes les classes de l'ontologie ;
- une table par classe avec héritage de table (si un SGBD relationnel-objet est utilisé) ;
- une table par classe sans héritage de table.

L'approche binaire a l'avantage d'être efficace en temps de réponse pour des requêtes simples. Mais, les requêtes complexes imposent un nombre élevé d'opérations de jointure. Cette représentation est adoptée par les BDBO RDFSuite (Alexaki, Christophides, Karvounarakis, Plexousakis, & Tolle, 2001) et IBM SOR (Lu et al., 2007).

Quelques BDBOs combinent ces approches dans une approche hybride comme Jena2 (Wilkinson, Sayers, Kuno, & Reynolds, 2003) qui combine les approches verticale et binaire, et DLDB (Pan & Heflin, 2003) qui combine l'approche binaire et horizontale.

### 2.6.2 Accès aux données à base ontologique

Aujourd'hui, plusieurs applications sont confrontées au problème d'accès aux sources de données existantes et ont besoin d'un mécanisme flexible à la fois puissant et efficace. Dans ce contexte, les ontologies sont largement considérées comme un outil pour résoudre ce problème et offrir un accès sophistiqué aux données.

L'Accès aux Données à Base Ontologique « ADBO » est considéré comme une approche importante et prometteuse (Kontchakov, Rezk, Rodriguez-Muro, Xiao, & Zakharyaschev, 2014) (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2017) pour la nouvelle génération des applications sémantiques. Selon Roman Kontchakov (Kontchakov, Rezk, Rodriguez-Muro, Xiao, & Zakharyaschev, 2014), ADBO permet de :

1. Donner une vue conceptuelle de haut niveau aux données,
2. Fournir à l'utilisateur un vocabulaire pratique pour les requêtes,
3. Enrichir les données incomplètes avec des connaissances de base, et
4. Prendre en charge les requêtes sur des sources de données multiples et éventuellement hétérogènes.

Trois types d'ADBO sont distingués en fonction de la puissance expressive des logiques de description[7] (LD).

- ADBO pour les bases de données: certains LD, tels que la famille des logiques DL-Lite (et OWL2QL), permettent de réduire les requêtes conjonctives sur les ontologies aux requêtes de premier ordre sur les bases de données relationnelles standard (Calvanese et al., 2011) (Poggi et al., 2008) (Artale, Calvanese, Kontchakov, & Zakharyaschev, 2009);

[7] Une famille de formalismes de représentation de connaissances basés sur la logique

- ADBO pour les moteurs datalog: d'autres DL englobant la famille des logiques EL (OWL2 EL), Horn-SHIQ et Horn-SROIQ, supportent une réduction de données et peuvent être utilisés avec les moteurs de données (Rosati, 2011) (Lutz, Toman, & Wolter, 2009) (Ortiz, Rudolph, & Simkus, 2011);
- ADBO pour la LD expressif tels que ALC ou SHIQ qui nécessitent des techniques spéciales pour répondre aux requêtes conjonctives (Hustadt, Motik, & Sattler, 2007) (Lutz, 2008) (Bailey, Bry, Furche, & Schaffert, 2005).

Calvanese et al. (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2017) proposent une approche d'accès à base d'ontologie dans un système d'intégration de données. Ce système nommé système d'ADBO ( ontology based data access OBDA) contient deux niveaux (voir la figure 1.5) :

- Le niveau intentionnel : il est présenté par la spécification (*J*) du système et est exprimé par le triplet (*O, S, M*), où (*O)* est une ontologie, *S* est le schéma de la source de données, et (*M*) est le mapping entre (*S)* et (*O)*. Plus précisément, (*M)* consiste en un ensemble d'assertions de mapping. Chaque assertion de mapping connecte une requête sur le schéma de la source à une requête sur l'ontologie
- Le niveau extensionnel : il est exprimé en termes de sources de base données (*D*) selon le schéma (*S*).

Dans un système d'ADBO, la réponse aux requêtes présente le service principal fourni par le système. L'utilisateur envoie des requêtes en se référant uniquement à l'ontologie et en masquant le détail de mise en œuvre de la source de données.

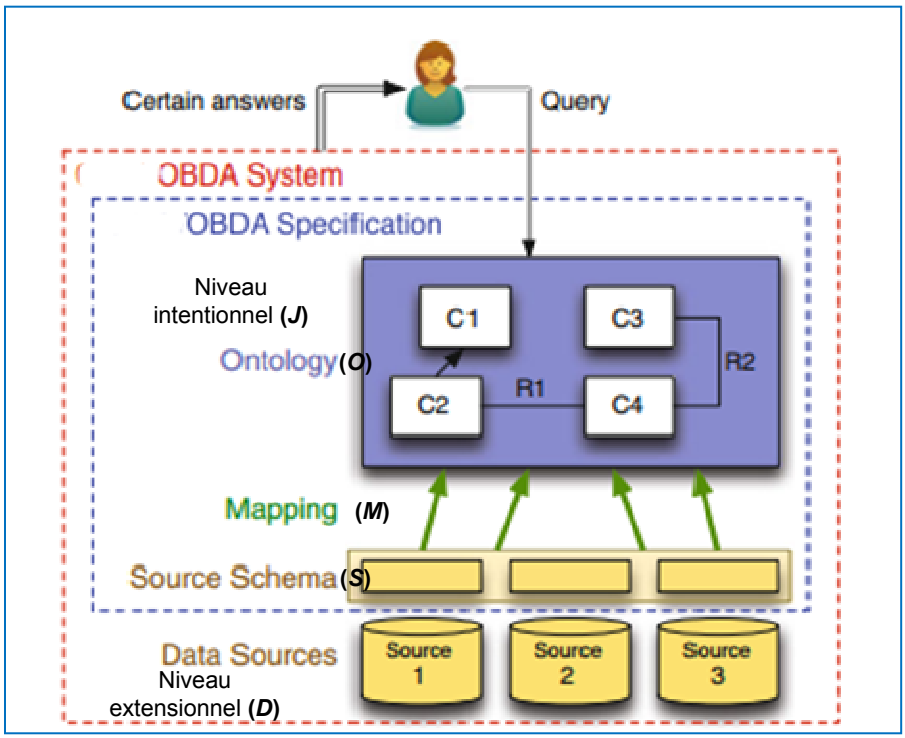



**Figure 1.5** Architecture d'un système d'accès aux données à base ontologique (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2017)

### 2.6.3 Intégration des données à base ontologique

L'intégration des données a pour objectif de permettre une combinaison des données qui résident dans des sources hétérogènes et autonomes. Elle fournit une vue unifiée et un accès transparent à ces sources. Cette vue unifiée est appelée le schéma global du système. L'intégration de données est un problème de recherche qui a été largement étudié depuis plusieurs années (Rahm & A-Bernstein, 2001) (Bakhtouchi, Bellatreche, Jean, & Ait-Ameur, 2012) (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2017). D'après Gómez-Giraldo (Gómez, 2005), un système d'intégration a pour but de fournir un schéma cohérent des données provenant de multiples sources d'informations autonomes, réparties et hétérogènes, de manière à faciliter aux utilisateurs l'accès et l'interrogation de ces données, comme s'ils accédaient à une seule source de données.

Le problème majeur des systèmes d'intégration est l'hétérogénéité des données à intégrer. Cette hétérogénéité peut provenir de deux causes principales : structurelle ou sémantique. Dans ce contexte, l'intégration est utilisée pour masquer cette hétérogénéité par le traitement de différents conflits structurels et sémantiques.

- **Hétérogénéité structurelle** : les conflits structurels entre les données surviennent lorsque des concepts équivalents sont modélisés différemment. Les conflits structurels peuvent être des conflits dans le choix de type de données, des conflits dans le choix de nom des attributs, des conflits d'agrégation, des conflits de généralisation, etc.
- **Hétérogénéité sémantique** : les conflits sémantiques entre les données proviennent du fait que les sources de données sont conçues par différents concepteurs ayant des interprétations, des significations et des points de vue différents.

Dans le domaine de l'intégration de données, les ontologies peuvent contribuer à résoudre les problèmes d'hétérogénéité sémantique. L'ontologie décrit formellement les concepts d'un domaine ainsi que les relations entre eux. Dans la littérature, il existe trois types d'intégration en fonction de la façon dont les ontologies sont utilisées (Wache et al., 2001) (Ekaputra, Sabou, Serral, Kiesling, & Biffl, 2017): intégration par une seule ontologie, par multiple ontologies et hybride (voir Figure 1.6).

1. **Intégration par une seule ontologie :** Cette approche utilise une ontologie unique partagée et globale. Chaque source à intégrer est liée à cette ontologie globale. Dans cette approche, le processus d'intégration comprend deux étapes (Ekaputra, Sabou, Serral, Kiesling, & Biffl, 2017): (i) définir une seule ontologie globale *G* et (ii) transformer les données source de *A*, *B* et *C* en une ontologie globale *G*. Avec ce type d'intégration, seulement les données qui sont représentées dans l'ontologie sont accessibles. Ce processus d'intégration est généralement difficile à gérer à cause des changements fréquents des données. Chaque fois qu'un changement survient dans l'une des sources de données, il faut décider s'il faut appliquer le changement à l'ontologie globale *G* ou non. Si tel est le cas, donc il faut garantir la mise à jour de l'ontologie globale ainsi que tous les mappages vers toutes les sources de données.

2. **Intégration par multiple ontologies :** Dans cette approche, chaque source de données est liée à sa propre ontologie indépendamment des autres sources. Un ensemble de mappings entre ces ontologies est établi permettant l'identification des correspondances ente leurs concepts. Le processus d'intégration comprend trois étapes (Ekaputra, Sabou, Serral, Kiesling, & Biffl, 2017): (i) créer les ontologies locales $L_A$, $L_B$ et $L_C$ pour les sources de données A, B et C, respectivement, (ii) transformer les données source de *A*, *B* et *C* en fonction de leurs ontologies respectives des sources locales et (iii) établir des correspondances sémantiques entre les ontologies associées. Chaque ontologie locale est accessible indépendamment. Cependant, l'inconvénient de cette approche est qu'il est difficile de définir et de maintenir les correspondances sémantiques entre les ontologies impliquées, en raison de la granularité variable des ontologies locales. De plus, chaque addition d'une nouvelle source de données nécessite une nouvelle ontologie et des mapping sémantiques supplémentaires sur toutes les ontologies locales existantes.
3. **Intégration hybride :** Ce type d'intégration combine les deux types précédents. En effet, chaque source est liée à sa propre ontologie. Un vocabulaire commun partagé entre les différentes ontologies locales est utilisé pour les connecter. Dans cette approche, le processus d'intégration comprend trois étapes (Ekaputra, Sabou, Serral, Kiesling, & Biffl, 2017): (i) définir un vocabulaire partagé *V* qui contient les termes/concepts de base du domaine, (ii) créer les ontologies locales: $L_A$, $L_B$ et $L_C$ en utilisant et/ou en étendant le vocabulaire partage *V* pour les sources de données *A*, *B* et *C* respectivement, et (iii) transformation/annotation des données de source *A*, *B* et *C* conformément aux ontologies locales $L_A$, $L_B$ et $L_C$. Ce type hybride offre deux moyens d'accès aux données: (i) un accès direct aux ontologies locales et (ii) un accès au vocabulaire partagé, où le système interroge chaque ontologie locale et fusionne les résultats. Cependant, d'un système de ce type nécessite un effort considérable.

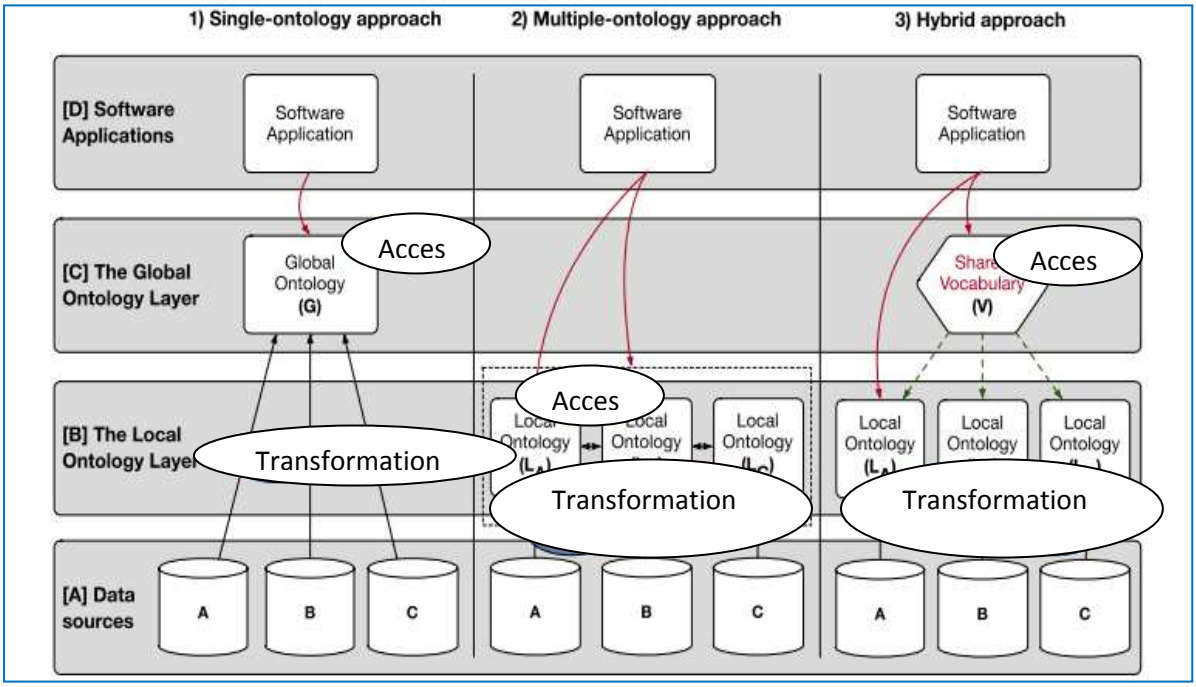



**Figure 1.6** Intégration des données à base d'ontologie (Ekaputra, Sabou, Serral, Kiesling, & Biffl, 2017)

Parmi les systèmes d'intégration fondés sur une ontologie, on peut citer :

- **Graube et al. (Graube, Urbas, & Hladik, 2016)** : les auteurs proposent un système d'intégration par une seule ontologie dans le web de données liées et ouvertes pour les données statiques (par exemple, des données RDF) et les données transitoires (par exemple, des données de capteur provenant de services Web) basées sur la fonctionnalité SPARQL endpoints[8]. Ils proposent une méthode pour acquérir des données transitoires (par exemple, des services Web contenant des données de capteur). Une évaluation de la solution proposée offre des performances suffisantes pour accéder aux données transitoires comme alternative aux solutions actuellement disponibles (SSN (Simple Notification Service), SensorML [9] et Linked Sensor Middleware[10], par exemple).
- **MASTRO (Calvanese et al., 2011)**: est un système d'intégration de type ADBO (Accès aux Donnes à Base Ontologique) (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2017). Dans ce système, l'ontologie globale fournit une interface ou une vue conceptuelle commune et abstraite du domaine d'application. Cette ontologie est formalisée dans la logique de description DL-LiteA. Seulement le TBox (schéma d'ontologie/niveau conceptuelle/ niveau intentionnel) de l'ontologie est matérialisé. Le niveau extensionnel reste dans les bases de données. Pour lier le niveau intentionnel par le niveau extensionnel MASTRO utilise un ensemble de mappings de type GAV.
- **Aarnio et al. (Aarnio, Seilonen, & Friman, 2014)** : les auteurs proposent un système d'intégration hybride pour prendre en charge la surveillance conditionnelle dans les systèmes d'automatisation. Ils effectuent un processus de transformation est effectué en quatre étapes à partir de données locales vers RDF :
    1. Transformation automatique des données locales aux données temporaires en format RDF.
    2. Transformation de données temporaires en instances des ontologies locales.
    3. Utilisation de l'outil SILK[11] pour relier les données d'ontologies locales
    4. Développement et exécution des ensembles de règles sur les ontologies locales pour déduire des nouvelles informations.

  L'approche est évaluée avec un ensemble de requêtes SPARQL

Selon la littérature, deux observations principales concernant l'utilisation des ontologies comme un schéma global dans un système d'intégration de données peuvent être signalées :

- Les ontologies apportent une description lisible et une sémantique forte qui conduit à une modélisation conceptuelle claire facilitant l'interopérabilité sémantique des sources hétérogènes dans un système d'intégration de données.

[8] https://www.w3.org/wiki/SparqlEndpoints
[9] http://www.opengeospatial.org/standards/sensorml
[10] http://open-platforms.eu/library/deri-lsm/
[11] http://silk-framework.com/

- L’ontologie qui est utilisée comme un schéma global est soit une ontologie de domaine existante, soit une ontologie fondamentale. Elle fournit un vocabulaire commun qui est utilisé comme une interface unique pour l’intégration sur laquelle un utilisateur peut exprimer ses requêtes sans connaissances préalables des sources locales. Ce vocabulaire est utile pour faciliter la communication entre différents concepteurs de connaissances dans un domaine donné.

# 3 Ontologie et notion de point de vue

Le concept de point de vue a été utilisé sous différents termes tels que perspective, rôle, vue, aspect, dimension, contexte...etc. Un point de vue est défini dans le dictionnaire ROBERT par : "un endroit où l'on doit se placer pour voir un objet le mieux possible ou encore comme étant une manière particulière dont une question peut être considérée". Dans (Marino, 1993), l’auteur considère une perspective comme une position «conceptuelle » dans laquelle un observateur regarde un objet.

Une des premières références qui ont considéré la notion de point de vue au terme de perspective se trouve dans le travail de Minsky (Minsky, 1975) avec une connotation spatiale. Les observateurs observent un même univers de discours produisant des points de vue qui peuvent être considérés de différentes manières (Benchikha & Boufaida, 2007):

- Points de vue uniformes : tous les acteurs (observateurs) ont la même vision de l’univers de discours et produisent des représentations équivalentes.
- Points de vue complémentaires : chaque acteur qui observe le même monde observé par les autres acteurs en voit une partie. Chaque point de vue est une représentation partielle et cohérente du monde.
- Points de vue comparables : les acteurs produisent des représentations comparables au sens plus général/spécifique.

Plusieurs équipes de recherche ont travaillé depuis plusieurs années sur l'intégration de la notion de point de vue dans la représentation des connaissances par objet (TROPES (Marino, 1993), VBOOL (Marcaillou, 1995), MVDB (Benchikha & Boufaida, 2007)). Ces travaux résolvent des problèmes en représentation par objet en relation avec le fait qu’un objet est perçu selon différentes perspectives. TROPES (Marino, 1993) est un système de représentation de connaissances par taxonomie des objets multipoints de vue. Il décompose le monde en concepts (familles) ; la description de chaque concept peut être selon différents points de vue. Les classes des différents points de vue peuvent être reliées par une ou plusieurs passerelles. Dans TROPES, Une passerelle établit une relation d’inclusion ensembliste entre des classes de points de vue différents. Dans VBOOL (Marcaillou, 1995), le point de vue est considéré comme un mécanisme permettant à plusieurs utilisateurs partageant un modèle commun d’accéder à un sous ensemble d’informations du modèle selon leurs besoins. Le modèle MVDB (Benchikha & Boufaida, 2007) offre un moyen de construire un ensemble de schéma de point de vue (ViewPoint) en utilisant la dépendance ‘***Vp-Extension***’ à partir d’un schéma référentiel. Les différents schémas de point de vue sont liés par des relations de type

(***vp_dependency***). Nous citerons dans ce qui suit quelques définitions du concept point de vue en représentation de connaissance.

Dans cette section nous confronterons quelques définitions de la notion de point de vue afin de relever une vue panoramique de l'utilisation de cette notion surtout pour la représentation des connaissances

- **Définition 1. (Marino, 1993):** Le point de vue est défini explicitement comme « une position conceptuelle de laquelle un observateur regarde un objet».
- **Définition 2. (Marcaillou, 1995):** Le point de vue est défini en se basant sur la définition de vue. Une vue est «une abstraction partielle du modèle et correspond donc à un sous modèle » ; et un point de vue est «est la vision qu'a un utilisateur du modèle et correspond à la combinaison de plusieurs vues, éventuellement réduite à une seule vue ».
- **Définition 3. (Bach, 2006):** Le point de vue d'une personne correspond à « un contexte ou une situation, où des connaissances à propos d'un objet, d'un concept ou d'une entité sont exprimées et considérées comme valides et vraies selon ce point de vue».
- **Définition 4. (Benchikha & Boufaida, 2007) :** Un point de vue est « une abstraction d'une certaine perception (vision) des objets du monde ». C'est une spécification d'une hiérarchie de classes appelée schéma Point de Vue (schéma-PV). L' ensemble de schémas point de vue est appelé schéma multi-point de vue.

### 3.1 Intégration de la notion de point de vue en ingénierie ontologique

Les travaux récents ont discuté la notion de point de vue dans l'ingénierie ontologique. Cependant, les premières solutions proposées intègrent ce concept comme un mécanisme de multi-représentation des objets. Benslimane et al. (Benslimane et al., 2006) définissent un langage d'ontologie contextuel pour supporter la représentation multiple d'ontologie. L'idée principale de leur travail consiste à adapter le mécanisme d'estampage proposé par Ribière et Dieng (Ribière & Dieng, 1997) pour répondre aux besoins de multi-représentation dans les graphes conceptuels. Dans ce qui suit, nous présenterons quelques travaux que nous avons jugés comme pertinents dans le contexte de notre recherche.

#### 3.1.1 Modèle de connaissance multipoints de vue (Falquet & Mottaz, 2002)

Ce modèle se base sur la présentation d'une ontologie multipoints de vues où les concepts (termes) sont rattachés à de nombreuses définitions formelles avec au maximum une définition par point de vue. Les concepts sont organisés dans une hiérarchie générique/spécifique et leurs positions dépendent de leurs définitions. Un domaine est défini comme une unité thématique. Les points de vue sont pris en compte à l'intérieur d'une «conceptualisation ou aussi vision » d'un domaine (voir la figure 1.7).

Un point de vue est un ensemble de définitions, chacune d'entre elles étant reliée à un concept différent. On peut envisager plusieurs types de relations entre points de vue :

- Points de vue par « proximité » / niveau : Par exemple dans le domaine de Zoologie, une vision peut contenir les deux points de vue du concept « cheval » : le point de vue « enfant », la classification : cheval → mammifère → vertébré →animal et dans le point de vue « zoologue » cheval → onguligrade avec nombre de doigts impair → onguligrade → … Le point de vue « enfant » représente donc un sous-ensemble du point de vue « zoologue ».
- Point de vue = vue partielle : Par exemple pour le domaine des voitures, dans le point de vue « mécanicien » les véhicules sont définis par (type de moteur, le diamètre des roues, etc) et dans le point de vue « vendeur », ils sont définies par (type de véhicule, la taille, la catégorie de prix, etc.).
- Points de vue avec chevauchement : Par exemple dans le domaine des immeubles, une partie des définitions de concepts peut être identique pour un architecte et pour un ingénieur en génie civil, alors que certains autres concepts seront définis différemment
- Points de vue avec ordres de classification différents : Par exemple dans le domaine des boissons. On peut avoir un point de vue avec l'ordre « alcoolisée/non alcoolisée » ou un point de vue « gazeuse/non gazeuse », ou un autre avec « à base de fruits/à base de lait/à base d'eau », etc.

Cependant, il n'y a pas une contradiction ou incompatibilité entre les points de vue de la même vision.

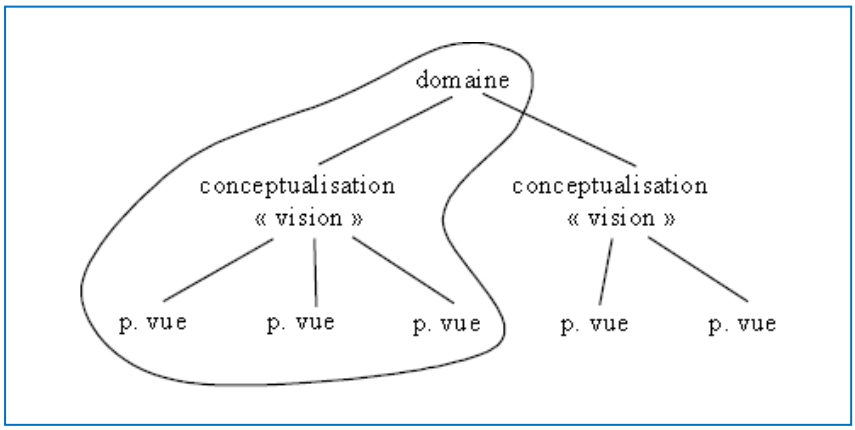



**Figure 1.7** Domaines, visions et points de vue (Falquet & Mottaz, 2002)

Les concepts sont définis en utilisant un langage formel 'ConcepTerm' (Berthet et al., 1994) qui fait partie de la famille des logiques descriptives (sous ensemble des logiques terminologiques de type ALCNR (Buchheit, Donini Francesco, & Schaerf, 1993)). Un concept peut être défini de trois manières :

- par conjonction ou disjonction de deux concepts
- par négation d'un concept
- par un lien, appelé rôle, vers un autre concept

Le rôle peut être quantifié universellement et l'ajout d'une contrainte d'inclusion est possible. Pour créer une ontologie qui contient plusieurs concepts, il est nécessaire de réifier

leurs définitions, ce qu'on peut faire sous forme de graphes orientés étiquetés, de la manière suivante :

- un concept (défini par conjonction, disjonction ou négation) est un nœud du graphe. Il est étiqueté par le nom du concept et par l'opérateur logique utilisé. L'arc de type "composition" est utilisé pour lier le concept aux concepts qui le composent et par un arc de type "généralisation" à son concept générique.
- un arc de type "rôle" est utilisé pour représenter un concept défini par un rôle. Cet arc porte le nom du rôle et pointe vers un autre concept. Sur l'arc, une étiquette qui porte le quantificateur all et une contrainte de nombre d'occurrences minimum et maximum est utilisée.

La figure 1.8 présente la définition de quaternion. "Un quaternion est un nombre formé d'au moins et d'au plus (indication (1,1)) une partie réelle et d'au moins et d'au plus une partie vectorielle.

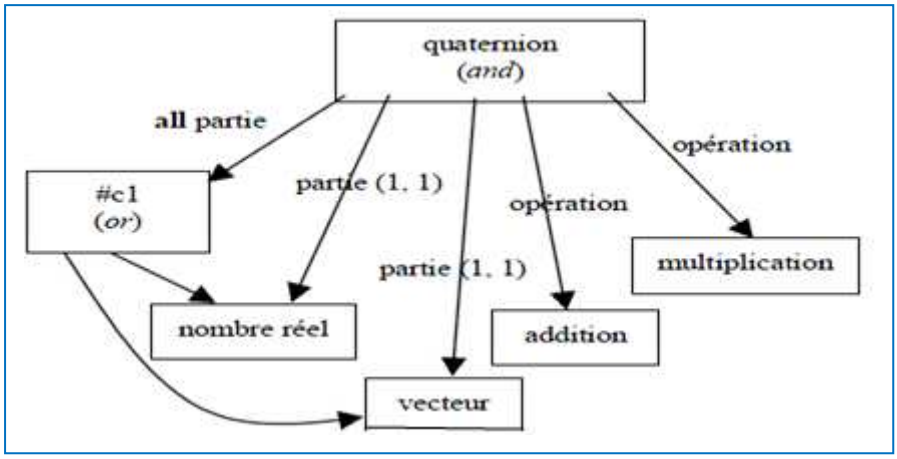



**Figure 1.8** Exemple de définition (Falquet & Mottaz, 2002)

Cette réification permet par exemple le stockage des définitions de concepts dans une base de données relationnelle. Le diagramme ci-dessous montre l'insertion des objets Concept et LienSémantique (rôle dans LD) dans le modèle orienté-objet.

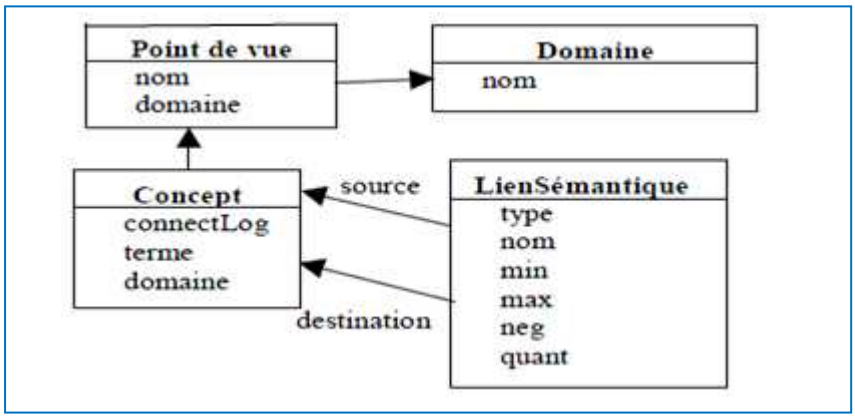



**Figure 1.9** Diagramme de classe (Falquet & Mottaz, 2002)

Les auteurs offrent les relations suivantes pour stocker les définitions dans une base de données relationnelle :

relation **concept** (concept_id, terme, connecteurLogique, pointDeVue)

relation **lienSem** (source, destination, type, min, max, negation, all)

relation **distance** (concept_id1, concept_id2, distance)

La table distance n'est pas présentée dans le modèle. Les auteurs ne nous indiquent pas comment calculer la distance entre deux concepts et quel est le type (sémantique /métrique) et le rôle de cette distance. Concernant les liens, il existe seulement des liens sémantiques entre les concepts. L'ajout des liens entre les points de vue peut aussi compléter le modèle pour assurer la cohérence dans la même vision et entre les visons et aussi pour ajouter une capacité indispensable de raisonnement sur l'ontologie.

### 3.1.2 Modèle MVP (Bach, 2006)

Dans le cadre du Web sémantique, Bach (Bach, 2006) offre une représentation des connaissances exploitant la notion de point de vue: représentation des ontologies en prenant en compte différents points de vue. D'après Bach (Bach, 2006): « dans le Web sémantique où il y' a une diversité des sources de connaissances et différentes catégories d'utilisateurs, une ontologie devrait être construite dans un environnement multipoints de vue mais tout en gardant un certain niveau de consensus ». L'ontologie construite dans cet environnement et de cette manière est appelée ontologie multipoints de vue.

Bach [9] adapte deux types de points de vue présentés dans (Ribière & Dieng, 1997):

- Points de vue perspective : Ce sont des descriptions consensuelles d'un même objet par différents ingénieurs d'ontologie ou différents experts. Ces vues sont complémentaires et forment une vision cohérente du monde.
- Points de vue opinion : Ce sont des vues non consensuelles de certains ingénieurs d'ontologie, experts, annotateurs ou utilisateurs finaux. Ces vues représentent indépendamment les unes des autres des visions incomplètes ou contradictoires.

Le modèle multipoints de vue de Bach (ou le modèle MVP) introduit quelques entités pour intégrer la notion de point de vue (voir la figure 1.10) qui sont:

- **Point de vue** ($VP_i$): Un point de vue est une entité dans le modèle multipoints de vue. Il correspond à une vue perspective composée de certaines caractéristiques (propriétés ou attributs) d'un concept (classe ou elle est représentée par une entité/nœud bleu dans la figure 1.10) qui sont pertinentes par rapport à ce point de vue. Un point de vue correspond aussi à un contexte particulier où les individus (objets réels) sont décrits et utilisés. Les concepts d'un point de vue sont classifiés par une relation de subsomption (spécialisation). Un concept (classe) peut être

sous-concept directe de plusieurs concepts selon plusieurs points de vue différents. Les points de vue, comme les autres entités dans le modèle MVP, sont référencés par des URIs. Bach suppose que les connaissances au niveau conceptuel dans une entreprise dans le cadre du Web sémantique soient cohérentes et compatibles. Ainsi, les points de vue utilisés dans les définitions des classes sont complémentaires et forment une vision cohérente du monde. Dans le modèle, il existe un point de vue général, nommé $\boldsymbol{VP_G}$ où les connaissances, les descriptions, les définitions à propos des entités (sauf les individus) dans le modèle sont considérées valides et vraies selon ce point de vue général.

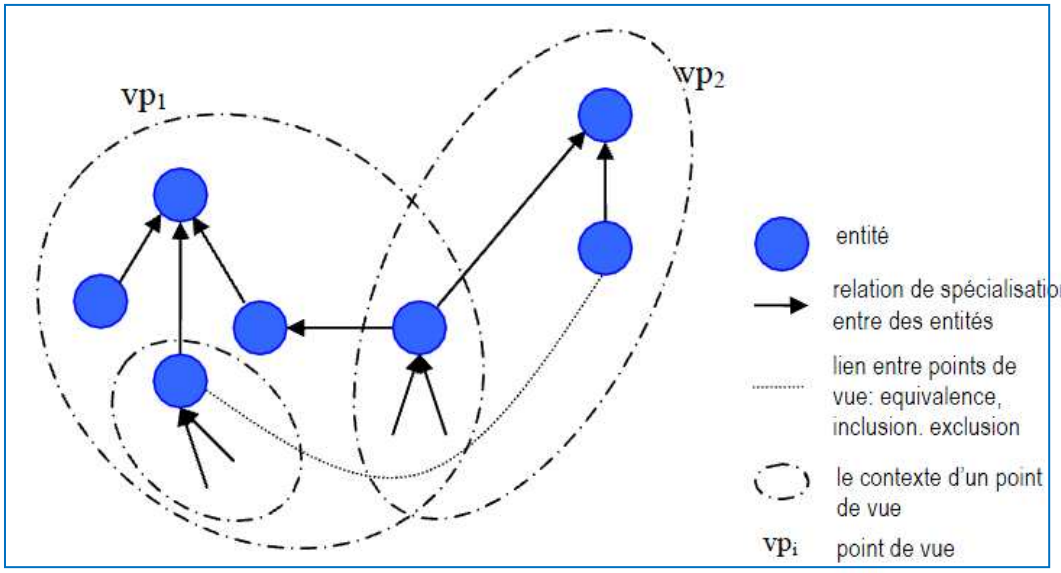



**Figure 1.10** le modèle multipoints de vue de Bach (Bach, 2006)

- **Classe** : Elle correspond à un ensemble d'individus (objets réels), qui ont les mêmes caractéristiques définies par la classe. Une classe peut être (1) l'intersection, ou l'union d'autres classes ; (2) le complément, l'équivalence ou la disjonction d'une autre classe ; (3) l'énumération des instances ; ou (4) la restriction de certaines propriétés d'une classe. Elle peut être définie comme sous-classe d'une autre classe selon un point de vue.
- **Propriété (Relation)** : ce sont les caractéristiques des classes. Une propriété peut être aussi définie comme sous-propriété d'une autre propriété ou de plusieurs propriétés. Elle peut être sous-propriété de plusieurs propriétés.
- **Liens entre points de vue (passerelles)** : ce sont les passerelles d'équivalence, d'inclusion et d'exclusion entre des classes définies comme sous-classes d'autres classes selon des points de vue différents et aussi entre des propriétés et des instances.
    - Le lien d'équivalence permet d'exprimer que deux classes utilisées et vues chacune dans un contexte différent, selon un point de vue différent ont le même sens.

- Le lien d'inclusion permet d'exprimer que le sens de la première classe inclut celui de la deuxième classe.
- Le lien d'exclusion permet d'exprimer que l'intersection des instances de deux classes est vide.

Ces passerelles permettent de raisonner avec les classes ainsi que les individus définis dans le modèle en entier, à travers des points de vue différents.

- **Instance (Individu)**: c'est un objet du monde réel. Une instance peut être rattachée à un ou plusieurs points de vue. Cela permet d'exprimer la multi-représentation d'un objet réel. Si une instance n'est pas associée à un point de vue, elle est considérée décrite selon le point de vue général $\boldsymbol{VP_G}$.
- **Annotation** : selon les points de vue, les annotations annoncées par les utilisateurs finaux sont différentes et subjectives sur les objets réels. Elles peuvent être contradictoires entre les points de vue.

Pour implémenter le modèle MVP, Bach propose un langage d'ontologie permettant de représenter une ontologie multipoints de vue dans le cadre du web sémantique. Le langage d'ontologie proposé, nommé MVP-OWL, est une extension du langage d'ontologie OWL par l'ajout de nouvelles primitives permettant d'exprimer les concepts (classes), les faits, les annotations.

### 3.1.3 Modèle MPV (Hemam, 2012)

Hemam (Hemam, 2012) propose un modèle d'ontologie multipoints de vue en Logique de Description en s'inspirant de travaux qui traitent la représentation des connaissances par objets auxquels sont associés des points de vue (Marino, 1993). Dans ce travail le mécanisme d'estampillage utilisé dans (Benslimane et al., 2006) est adapté pour permettre la multi représentation des concepts en Logique de Descriptions. Une ontologie est définie par un quadruplet $O = <C_G, R_G, Vp, M>$, où :

- $C_G$ est l'ensemble des concepts globaux,
- $R_G$ est l'ensemble des rôles globaux
- $Vp$ est l'ensemble des points de vue
- $M$ est l'ensemble des passerelles.

Un point de vue est une description partielle d'un univers de discours selon une perception particulière. Un point de vue est défini par un triplet $VP_K = <C_L, R_L, A_L>$, où :

- $C_L$ est l'ensemble des concepts locaux
- $R_L$ est l'ensemble des rôles locaux
- $A_L$ est l'ensemble des individus locaux

Un *Concept global (noté* $\boldsymbol{C^{\hat{0}}}$ *)* est un concept vu par l'ensemble des points de vue avec des propriétés communes. En Logique de Description étendue par le mécanisme d'estampillage, un *Concept global* peut être formé en utilisant les constructeurs de conjonction et de disjonction ainsi que les constructeurs de restriction de cardinalité.

Un *Concept local* (noté $\boldsymbol{vp_i: Concept_i}$ est un concept qui est vu et décrit selon un point de vue donné. Les concepts locaux sont organisés par une relation de *subsomption* (⊑) nommée *Hiérarchie locale*. Ils sont aussi liés par des canaux de communication, appelé *Passerelles*, qui permettent de représenter des liens consensuels entre les concepts locaux de différents points de vue. Dans ce modèle deux types de *Passerelle* sont définis. *Passerelle unidirectionnelle* ( $\overset{\sqsubseteq}{\rightarrow}$ ) qui représente une relation d'inclusion et *Passerelle bidirectionnelle* qui représente ( $\overset{\equiv}{\Leftrightarrow}$,$\overset{\perp}{\Leftrightarrow}$) une relation d'égalité ou bien une relation d'exclusion. Un *Rôle global* (défini par $\boldsymbol{R^{\hat{0}}}$ ($vp_i$: C, $vp_j$: D) où R est un nom de rôle global, C et D sont deux concepts locaux définis dans deux points de vue différents) est une relation entre deux concepts locaux définis dans deux points de vue différents. Un *Rôle local* (défini par $\boldsymbol{vp_i: R}$ (C, D) où $\boldsymbol{R}$ est le rôle et (C, D) sont deux concepts locaux du point de vue $\boldsymbol{vp_i}$) est une relation entre deux concepts locaux définis dans le même point de vue.

Un langage d'ontologie, nommé MVP-OWL, permettant de représenter une ontologie multipoints de vue est proposée dans (Hemam, 2012). MVP-OWL est une extension du langage d'ontologie OWL par l'ajout de nouvelles primitives permettant d'exprimer les concepts (classes), les faits et les annotations.

### 3.2 Discussion

L'étude que nous avons menée sur l'intégration de la notion de point de vue dans les ontologies a permis de lever certaines limites de ces modèles. Il s'agit de limitations liées à :

1. **La conception de l'ontologie** : Comme les composants logiciels, les ontologies sont destinées à être utilisées et réutilisées dans les différents systèmes informatiques. En particulier les ontologies qui sont actuellement très utilisées pour accéder aux données larges et complexes comme le web de données liées et ouvertes. Ce type de système nécessite des ontologies bien structurées et de qualité afin de faciliter la compréhension, l'utilisation par l'utilisateur final, la réutilisation, et l'évolution des jeux de données. Nous avons constatés que les modèles existants ne prennent pas en considération la structuration ni la qualité de l'ontologie créée.
2. **La consistance et la cohérence de l'ontologie** : la consistance et la cohérence de l'ontologie est définie dans (Stojanovic, 2004) comme suit « une ontologie est cohérente par rapport à son modèle – d'ontologie – si et seulement si, elle satisfait les contraintes spécifiées par ce modèle ». Cependant tous les modèles présentés n'utilisent pas des contraintes pour garantir la consistance. Pour (Falquet & Mottaz, 2002)**,** l'ontologie finale peut contenir différents points de vue dans une même vision mais sans contradiction ou incompatibilité entre eux. Cependant, le modèle n'utilise aucune contrainte pour assurer cette proposition. Les contraintes utilisées dans le modèle sont des restrictions de cardinalité.
3. **L'évolution du modèle** : les ontologies évoluent pour répondre à différents besoins des utilisateurs. Dans les modèles de point de vue existants, cette évolution engage une gestion des changements d'ontologie tout en maintenant la cohérence des concepts entre et dans les points de vue. Cette maintenance implique une réorganisation de la structure de

l'ontologie. Et comme nous avons vu, les modèles présentés ne prennent pas en compte la structure de l'ontologie résultante et créent des ontologies complexes, difficiles à gérer et lourdes à réorganiser et classifier.

4. **La validation et l'exploitation du modèle** : comme tout modèle proposé, les modèles de point de vue nécessitent une validation afin de démontrer leur faisabilité d'intégrer et d'exploiter correctement la notion de point de vue avec une concordance structurelle et sémantique. La validation est une activité importante qui assure que le modèle répond à l'attente et satisfait bien les besoins et les objectifs spécifiés au départ. Cette validation peut être formelle en appliquant des méthodes formelles ou opérationnelles en prouvant le modèle par une implémentation ou une étude de cas. Seulement le modèle de (Falquet & Mottaz, 2002) a montré qu'il est possible de créer diverses vues dans leur système en combinant un ensemble de schémas de nœuds de base. Les autres travaux sont des modèles théoriques et sont présentés sans aucune validation.

# 4 Conclusion

De nos jours, la notion d'ontologie est utilisée dans de nombreux domaines d'application incluant le web de données, le commerce électronique, le traitement du langage naturel, etc. Dans ce chapitre nous avons rappelé des notions de base sur les ontologies ainsi que quelques applications couplant les ontologies et les bases de données. Nous avons présenté les trois orientations de recherche étudiant ce couplage qui sont : base de données à base ontologique, accès aux données à base ontologique, et intégration à base ontologique. Nous avons porté une attention particulière à la notion de point de vue dans les ontologies, étant donné que nous nous intéressons à son intégration pour construire une ontologie multipoint de point de vue.

Les problématiques inhérentes à l'intégration de la notion de point de vue dans les ontologies et les modèles de point de vue potentiels proposés dans la littérature ont suscité notre motivation à orienter les objectifs de la thèse vers la création d'une ontologie multipoints de vue bien structurée, de qualité et réutilisable. Afin d'atteindre cet objectif, nous utilisons les patrons de conception d'ontologie qui seront présentés dans le chapitre prochain. L'utilisation du patron nous permettra de créer un modèle facile à exploiter, facile à évoluer et facile à valider formellement en utilisant une extension de l'analyse formelle de concepts (*cf* chapitre 3).

Chapitre 2

# Patrons de Conception d'Ontologie et Ingénierie des Ontologies

## Sommaire



## 1 Introduction

Dans l'ingénierie ontologique, le développement d'ontologie doit s'appuyer sur des principes similaires à ceux utilisés en génie logiciel. Dans la littérature, plusieurs méthodes de construction d'ontologie existent (Méthode On-To-Knowledge (Sure, Tempich, & Vrandecic, 2006), Méthode Methontology (Corcho, Fernández., Gómez-Pérez., & López-Cima., 2005) Méthode ARCHONTE (Bachimont, Isaac, & Troncy, 2002)…etc.) et elles s'appuient sur un processus de développement découpé en trois principales phases : (1) la spécification sémantique, (2) la conceptualisation, (3) et la formalisation. Dans certaines méthodes, le processus se termine par une étape d'implémentation. En outre, le développement d'une ontologie de qualité est un enjeu important pour l'ingénierie ontologique. Une ontologie de qualité facilite l'évolution, la maintenance (enrichissement, correction de bogues de conception) et la réutilisation.

Dans le monde informatique, le concept de réutilisation fut présenté pour la première fois par McIlroy en 1967 (Pietro-Diaz & Jones, 1987). Il propose un catalogue renfermant des composantes logicielles modulaires qui peuvent être assemblées pour construire un tout comme en mécanique. La réutilisation est communément acceptée comme un moyen pour exploiter les architectures et les besoins communs à plusieurs applications. Tout comme l'ingénierie

logicielle vise la réutilisabilité des composants logiciels, l'ingénierie ontologique cherche à favoriser la réutilisation de modèles de connaissances. De la branche de l'ingénierie ontologique, a émergé la réingénierie ontologique comme un nouveau champ de recherche actif. La réingénierie d'ontologies est le processus qui consiste à réutiliser un modèle conceptuel d'une ontologie déjà implémentée pour le reconstruire ou le lier à un autre en cours d'implémentation (InterOp, 2006).

Les besoins en réutilisation de modèles de connaissances du domaine ont conduit à définir la notion de patrons de conception d'ontologie. Au cours du processus de construction d'ontologie, il est fréquent que certains problèmes soient récurrents (représentation des contextes, des points de vue, des relations n-aire…etc.) et il est plus que probable que ces problèmes amènent à des solutions de conceptualisation similaires. C'est ce qu'expriment les patrons de conception d'ontologie. Ils sont des modèles généraux des solutions conceptuelles et réutilisables après une adaptation pour créer des solutions conceptuelles spécifiques. Ils caractérisent une bonne pratique de modéliser certains types de connaissances. Ils peuvent être utiles pour développer des ontologies plus rapidement en aboutissant à un résultat de meilleure qualité (Presutti, Daga, Gangemi, & Blomqvist, 2009) (Rodriguez-Castro, 2012) (Shimizu, Hitzler, Paul, 2018) (Alirezaie, Hammar, & Blomqvist, 2018).

Dans ce chapitre, nous présentons les patrons de conception d'ontologie. La section 2 présente les patrons de conception d'ontologie, leur définition, leurs types, et leurs intérêts dans l'ingénierie ontologique. Nous abordons, dans la section 3, quelques applications des patrons de conception en ingénierie des otologies notamment pour la construction d'une ontologie de qualité, pour l'alignement des ontologies et dans le processus d'intégration des sources de données. Dans la section 4, nous présentons une discussion en évaluant les différents patrons présentés dans la section 3. La section 5 conclut le chapitre.

## 2 Patrons de conception d'ontologie

En littérature, Design pattern est souvent traduit par patron de conception, Bonne pratique, Modèle de conception, Motif de conception, etc. Les patrons sont utilisés dans plusieurs domaines. Ils ont d'abord été mentionnés dans le domaine d'architecture par le mathématicien et l'architecte Christopher Alexander en 1977 (Alexander, Ishikawa, & Silverstein, 1977). Ce dernier s'est interrogé sur l'évaluation objective de la qualité en architecture. Il s'est posé les questions suivantes : En quoi une conception est-elle bonne et une autre mauvaise ? Quels aspects les différencient ? Alexander a réussi à déterminer que les bonnes constructions qui résolvent le même problème ont des points communs (qualifiés de patrons) (Shalloway, & Trott, 2002).

La figure 2.1 illustre le cas de deux solutions apportées à un problème de délimitation de chemin d'accès. Cette constatation permet de définir un patron comme une solution décrivant un problème récurrent dans un environnement, puis le cœur de cette solution qui sera réutilisable indéfiniment, quelle que soit la mise en pratique choisie Alexander, Ishikawa, & Silverstein, 1977).

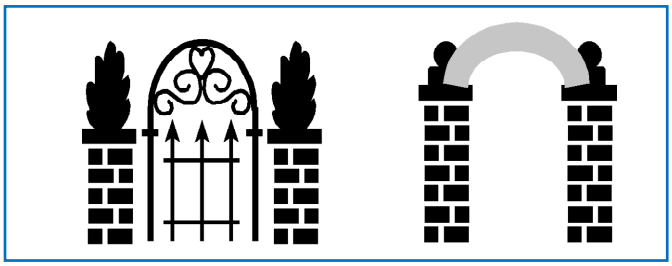


**Figure 2.1** Deux structures différentes pour le même problème Alexander, Ishikawa, & Silverstein, 1977).

En génie logiciel, un patron de conception est un concept destiné à résoudre les problèmes récurrents suivant le paradigme objet. Les patrons de conception décrivent des solutions standards pour répondre à des problèmes d'architecture et de conception des logiciels. Ils ont été formalisés pour la première fois en 1995 dans le livre du «Gang of Four» (Gamma, Helm, Johnson, & Vlissides, 1995). La figure 2.2 présente le patron composite (Gamma, Helm, Johnson, & Vlissides, 1995). Ce patron permet la conception et la construction des structures arborescentes entre les objets (des objets simples et des objets composites) et les traiter uniformément. Cette solution permet de simplifier l'ajout de nouveaux composants et fournit un modèle extensible.

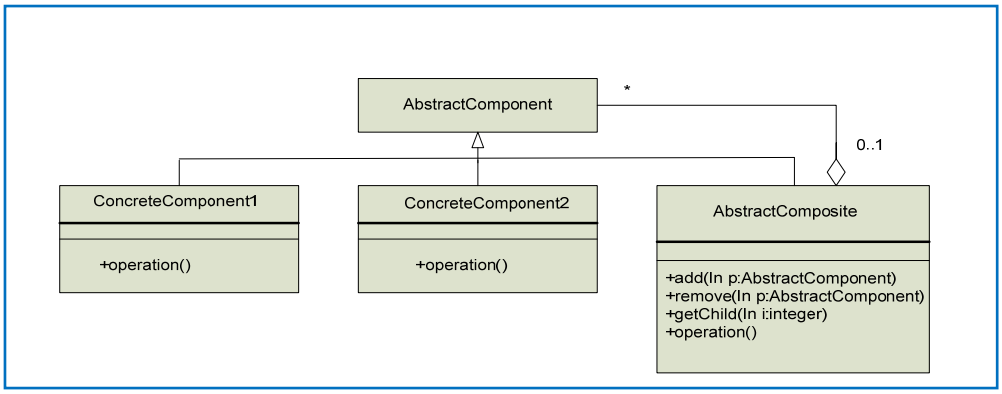



**Figure.2.2** Patron composite (Gamma, Helm, Johnson, & Vlissides, 1995)

Supposons que nous voulions concevoir une hiérarchie de composants graphiques qui seront manipulés suivant la même interface qu'ils soient unitaires ou composites. La conception du système présenté dans la figure 2.3 permet de dessiner une image graphique en implémentant le patron composite. Une image graphique est composée de lignes, rectangles, textes et images. Une image peut être composée d'autres images, lignes, rectangles et des textes.

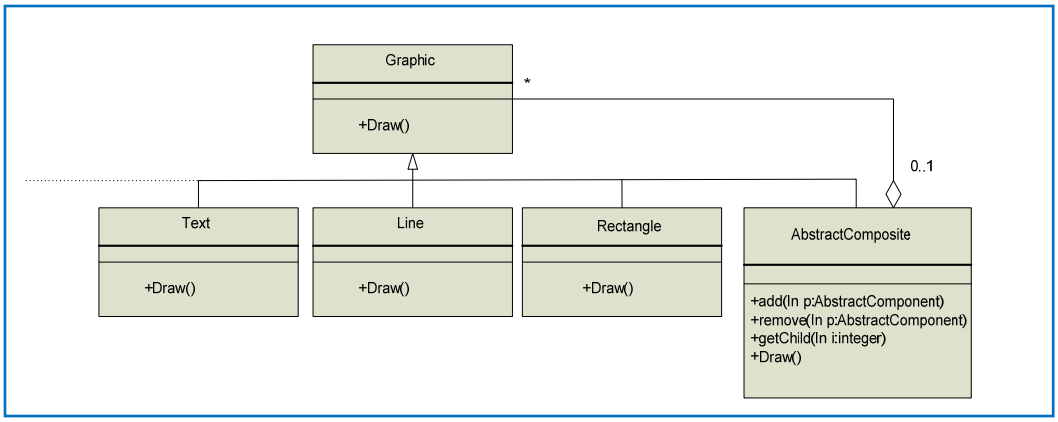



**Figure 2.3** Implémentation du patron composite pour un système de dessin des images (Gamma, Helm, Johnson, & Vlissides, 1995)

La figure 2.4 présente le patron stratégie (Gamma, Helm, Johnson, & Vlissides, 1995). C'est un patron de comportement qui définit une famille d'algorithmes, encapsule chacun d'eux et les rend interchangeables. Stratégie permet aux algorithmes de varier indépendamment des clients qui les utilisent.

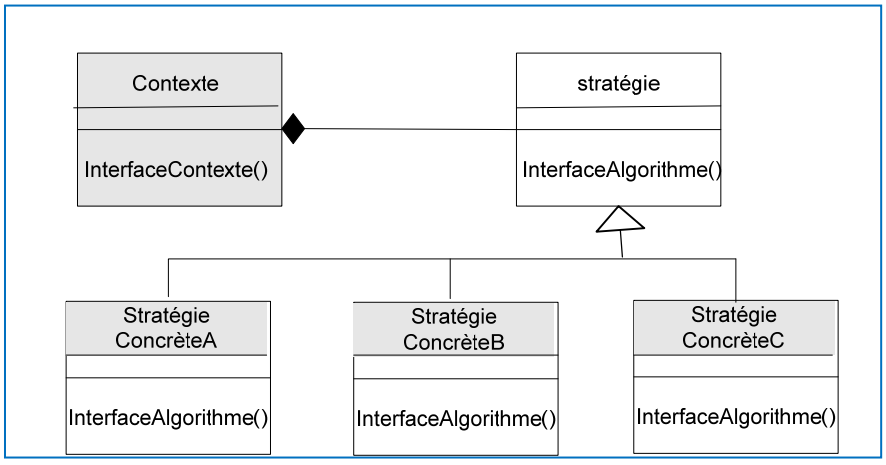



**Figure 2.4** Patron Stratégie (Gamma, Helm, Johnson, & Vlissides, 1995)

Comme en ingénierie du logiciel, l'ingénierie de connaissance et plus précisément ontologique a adopté les design patterns pour servir la représentation d'entités selon des modèles génériques et réutilisables.

Dans le cadre du projet NeOn, financé par l'Europe, Presutti et al. (Presutti, Daga, Gangemi, & Blomqvist, 2009) ont développé différents types de patron de conception d'ontologies liés à la modélisation des ontologies. L'idée a été inspirée du patron Presittu et al. (Presutti, Daga, Gangemi, & Blomqvist, 2009) défini comme suit :

« Un patron de conception d'ontologies est une solution modélisée permettant de résoudre un problème de conception d'ontologie récurrent. Il est un *objet d'information* qui exprime un *schéma de patron de conception* ou une *enveloppe de patron de conception*. Un *schéma* peut être satisfait seulement par une *solution de conception* qui est une combinaison particulaire des

*éléments ontologiques*. Cette *solution* remplit certains *rôles d'élément* définis par le patron de conception ».

Cette définition est basée sur les éléments suivants :

- **Un objet d'information** : c'est une pièce d'informations codée sur une entité pouvant être interprété par un agent. Il dépend de l'encodage ainsi que de la réalisation concrète. Des exemples d'objets d'information sont: une composition musicale, un texte, un mot, une image, etc.
- **Un schéma de patron de conception** : c'est la description d'un patron de conception d'ontologie. Il inclut les rôles, les tâches et les paramètres nécessaires pour résoudre un problème de conception d'une ontologie. Par exemple, un schéma de patron de conception pour la création de contenu fournit des instructions pour cloner, choisir, composer, ou spécialiser une ontologie existante ou une collection d'éléments ontologiques, éventuellement selon les bonnes pratiques.
- **Une enveloppe de patron de conception** : c'est une description structurelle du patron de conception d'ontologie. Il identifie les éléments nécessaires à mettre en œuvre pour décrire le patron de conception. Par exemple, une enveloppe pour le patron de conception d'ontologie « **partof** » est définie avec un rôle d'élément pour la relation méréologique et deux rôles d'élément (partie et tout) pour deux entités de même catégorie.
- **Un élément ontologique** : c'est un élément (identifié) d'ontologie. Les éléments ontologiques sont également modélisés dans le méta-modèle de définition d'ontologie (ODM) (Kendall, Dutra, Jacobs, & Schwab, 2005), le vocabulaire de métadonnées d'ontologie (OMV) (Haase, Brockmans, Palma, Euzenat, & dÕAquin, 2007). Par exemple : les classes OWL, les propriétés OWL et restrictions OWL.
- **Un rôle d'élément** : Un rôle d'élément est un concept qui classifie un élément ontologique.
- **Une solution de conception** : c'est une situation qui inclut uniquement les expressions formelles et leurs relations. Par exemple, l'utilisation du patron de conception partie/tout afin de modéliser les villes d'un pays est une solution de conception.

Nous considérons un patron de conception d'ontologie (Ontology Design Pattern ODP) comme une des solutions qui est jugée appropriée pour plusieurs conceptualisations, selon plusieurs ingénieurs d'ontologie, dans un problème comparable.

### 2.1 Types des patrons de conception d'ontologie

Un design pattern est indépendant du domaine et du langage d'ontologie. Cependant, les efforts actuels de cataloguer les patterns ne concernent que le développement en langage OWL (patterns implantés en OWL). Un catalogue de patterns est une collection des patterns qui sont classifiés et décrits selon un vocabulaire commun dénotant l'expertise des ingénieurs d'ontologie. Ce vocabulaire peut aider à la compréhension rapide des solutions proposées, sans entrer dans les détails. Ces patterns sont certifiés, validés et prêts à l'utilisation. Les ODPs

(Ontology Design Patterns) sont classifiés en six familles (types) (Gangemi & Presutti, 2009), comme montré dans la figure 2.5.

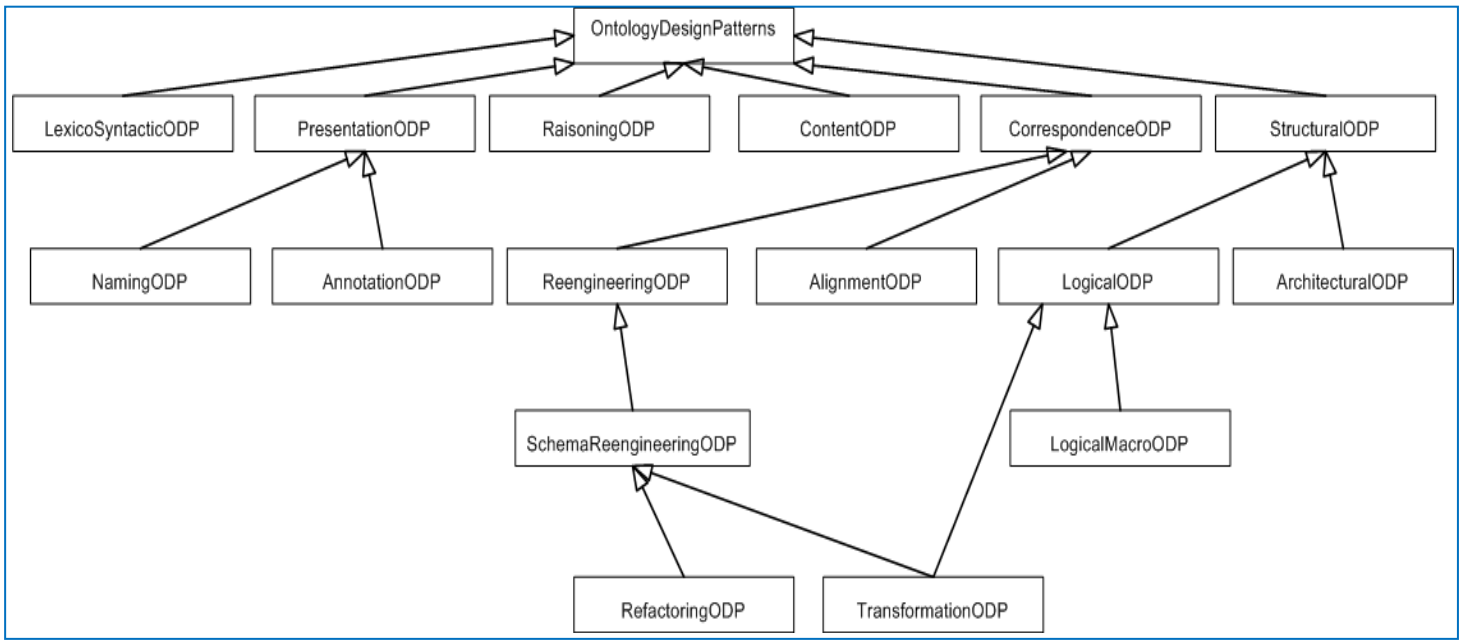



**Figure 2.5** Les types de patrons de conception d'ontologie (Gangemi & Presutti, 2009)

Dans ce qui suit, Certains types sont présentés sans exemples [1] puisque ils restent théoriques.

1. Dans les ODPs structurels, on distingue deux types de pattern :
   (1) les patterns d'architecture qui affectent la forme générale de l'ontologie, et dont le but est de contraindre la construction d'une ontologie et répondre à la question «comment l'ontologie devrait ressembler ». Par exemple, le patron d'architecture 'View Inheritance' (Rodriguez-Castro, 2012) qui résoud le problème de présentation des concepts selon plusieurs critères alternatifs permettant de classer leurs abstractions. Les classes ('Criterion_i') sont des critères d'abstraction alternatifs du concept 'TargetDomainConcept', les autres classes sont des affinements ou combinaisons de critères d'abstraction (*cf* figure 2.6).

[1] http://ontologydesignpatterns.org/wiki/Main_Page

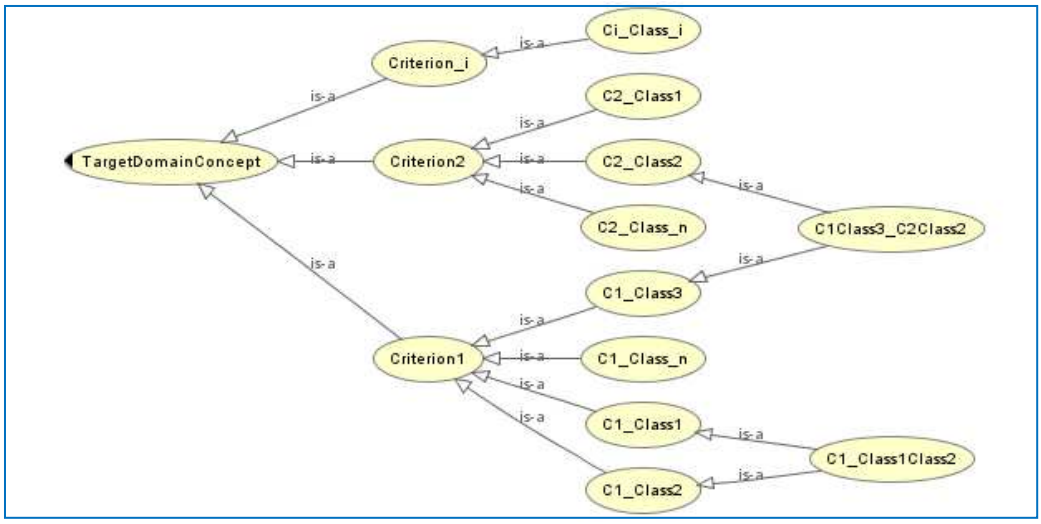



**Figure 2.6** Patron d'architecture d'ontologie « View Inheritance » (Rodriguez-Castro, 2012)

(2) le deuxième type concerne les patterns logiques exprimés en termes d'un vocabulaire logique. Ils aident à résoudre les problèmes de conception où les primitives du langage de représentation utilisé ne soutiennent pas directement certaines constructions logiques. Par exemple, Le patron logique de la relation symétrique « n-aire » est créé pour résoudre le problème de modélisation des distances entre deux points. La solution nécessite une entité supplémentaire (relation ou attribut) pour représenter la distance entre ces points (*cf* la figure 2.7).

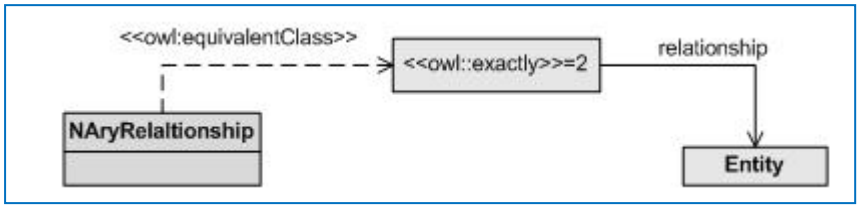



**Figure 2.7** Patron logique de la relation symétrique « n-aire »[2]

2. Les ODPs de correspondance sont des modèles pour découvrir les liens sémantiques entre deux ontologies existantes. Par exemple, le patron 'Class Union' (Scharffe, Zamazal, & Fensel, 2013) est utilisé pour résoudre le problème qu'une classe d'une ontologie est l'union de deux classes dans une autre ontologie. La classe ' CanadianCitizenByBirth' de l'ontologie (o2) est l'union de deux classes 'PersonBornInCanada' et 'PersonWithCanadienParent' de l'ontologie (o1) (*cf* la figure 2.8)

[2] http://ontologydesignpatterns.org/wiki/Submissions:Symmetric_n-ary_relationship

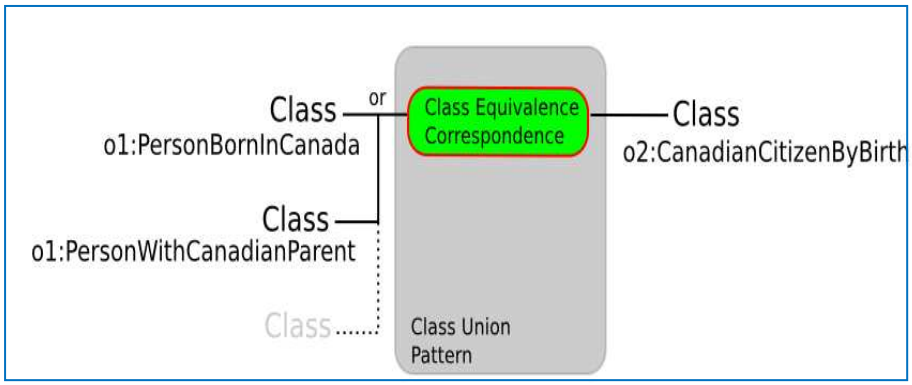



**Figure 2.8** Patron de correspondance « Class Union » (Scharffe, Zamazal, & Fensel, 2013)

3. Les ODPs de contenu sont des modèles de solution orientés domaine (tourisme, biologie…etc.) pour résoudre les problèmes de conception des classes et des propriétés dans une ontologie. Par exemple, le patron de contenu 'SmartHome_Event' (Alirezaie, Hammar, & Blomqvist, 2018) est créé pour résoudre le problème de représentation des événements dans le contexte d'environnements intelligents (*cf* la figure 2.9). Un événement ('SmartHomeEvent') est représenté sous la forme d'une manifestation ('Manifestation') ou d'un événement complexe (ComplexEvent'). Une manifestation représente l'occurrence d'une situation spécifique d'un objet capturé directement ("le téléviseur est allumé"), alors qu'un événement complexe est défini en fonction des conditions préalables (par exemple, "regarder la télévision" se produit lorsque "le téléviseur est allumé" et "quelqu'un est assis sur le canapé"). La condition préalable d'un événement complexe est représentée sous la forme d'une situation (DUL: Situation).

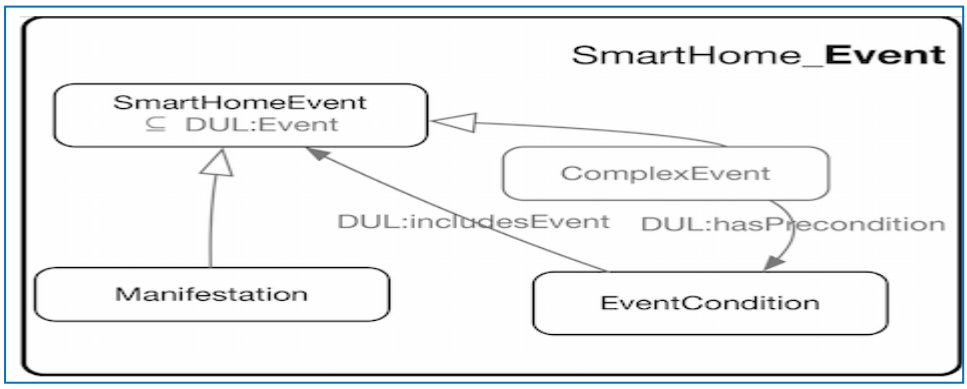



**Figure 2.9** Patron de contenu« SmartHome_Event » (Alirezaie, Hammar, & Blomqvist, 2018)

4. Les ODPs de raisonnement appliquent les ODPs logiques pour obtenir des résultats de certains raisonnements, basés sur le comportement implémenté dans un moteur de raisonnement.
5. Les ODPs de présentation font face à la convivialité et la lisibilité des ontologies à partir d'un point de vue d'utilisateur. On distingue deux types : (1) les patterns de

nommage qui présentent des conventions sur l'espace du nom «namespace », (2) les patterns 'annotation comme les étiquettes et les commentaires RDF.

6. Les ODPs lexico-syntaxiques sont des structures linguistiques ou des schémas qui sont constitués de certains types de mots suivant un ordre spécifique, et qui permettent de généraliser et d'extraire des conclusions sur le sens qu'ils expriment. Ils sont utiles pour associer des patterns logiques et des patterns de contenu simples avec des phrases du langage naturel. Par exemple le patron lexico-syntaxique 'superClass' (Maynard, Funk, & Peters, 2009) est créé pour résoudre le problème d'indiquer une superclasse (utilisée pour générer des informations ontologiques à partir des textes) (*cf* figure 2.10). NP signifie 'Noun phrase', CN signifie ('class of' ou 'group of'…etc.), CATV est une classification des verbes[3], PUNCT est la ponctuation et NPlist est une liste jointe de NPs ("X, Y and Z"). <sub> et >sup> signifie les sous-classes et les superclasses respectivement.

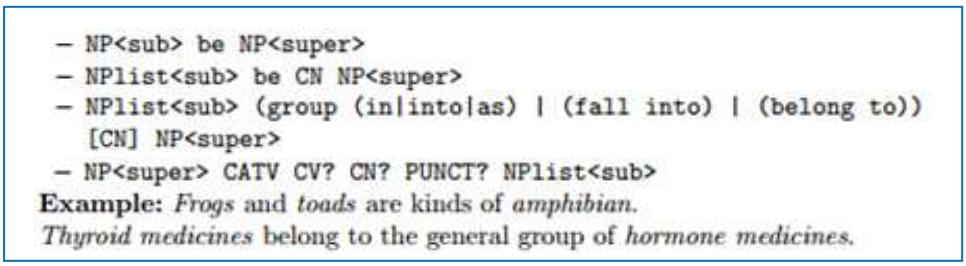



**Figure 2.10** Patron lexico-syntaxique « superClass » (Maynard, Funk, & Peters, 2009)

A L'état actuel de notre étude, des dizaines de patrons de conception d'ontologie sont crées. Pour cela, un effort énorme est dispensé dans le cadre du projet NeOn Projet[4] pour les classifier selon leur type (contenu, raisonnement…etc). Un portail sémantique ODP en ligne, "OntologyDesignPatterns.org", est créé pour soumettre les patrons. Il contient maintenant 210 patrons (voir le tableau 2.1). La majorité de ces patrons sont des patrons de contenu et les statistiques suivantes offertes par le portail "OntologyDesignPatterns.org" affirment cette conclusion.

| **Type de patron** | **Nombre de patron soumis** |
|---|---|
| Patrons de contenu | 155 |
| Patrons réingénierie | 12 |
| Patrons d'alignement | 14 |

[3] classify in/into, comprise, contain, compose (of ), group in/into, divide in/into, fall in/into, belong (to).
[4] (http://www.neontoolkit.org)

| | |
|---|---|
| Patrons structurel | 18 |
| Patrons architecturel | 1 |
| Patron de lexico-syntaxique | 20 |

**Tableau 2.1** Statistiques des patrons soumis dans le portail "OntologyDesignPatterns.org"

En effet, Le nombre élevé de ce type de patrons est justifié par sa similarité avec l'ontologie. Une ontologie est une conceptualisation d'un domaine d'intérêt. Et même le patron de contenu est une conceptualisation d'une notion particulaire dans le domaine d'intérêt (solution conceptuelle d'un problème conceptuel récurrent).

## 2.2 Intérêts des patrons de conception d'ontologie

L'un des défis les plus difficiles et les plus attentifs dans le domaine de conception et développement des ontologies est l'aspect de réutilisabilité, qui devient de plus en plus important dans un monde de données liées et ouvertes. Pour supporter et faciliter la réutilisation, Gangemi a introduit les patrons de conception d'ontologies (ODPs) (Gangemi, 2005) et des idées étendues ont été présenté dans le groupe de travail de W3C[5] sur les meilleures pratiques et déploiement du Web sémantique. Ainsi, parmi les apports inéluctables des patrons de conception :

1. Faciliter la réutilisation : L'utilisation des patrons de conception d'ontologie permet la réutilisation des concepts ontologiques issus des expériences des autres ingénieurs d'ontologie expérimentés dans la modélisation des problèmes récurrents dans domaine donné.
2. Faciliter le développement des ontologies : Des expériences dans l'ingénierie ontologique prouvent que les ODP améliorent le cycle de développement d'ontologies car la réutilisation des ODP rend le processus de développement de l'ontologie plus facile, réduit les erreurs dans les ontologies et fournit un moyen plus élégant de produire des ontologies (Poggi et al., 2008).
3. Faciliter la communication entre les développeurs d'ontologies : Les patrons de conception d'ontologie fournissent un vocabulaire partagé en assurant une meilleure communication entre les ingénieurs d'ontologie.
4. Maitriser la complexité des domaines modélisés : Grâce à ce vocabulaire fourni, les ingénieurs d'ontologie peuvent modéliser un domaine avec une meilleure abstraction.
5. Développer des ontologies modulaires et robustes : Les patrons de conception d'ontologie forment un catalogue de meilleures pratiques à partir desquelles des ontologies plus larges peuvent être élaborées.

[5] https://www.w3.org/2005/10/03-swbp-minutes.html

6. Faciliter l'évolution: Grâce à l'aspect de modularisation du patron de conception d'ontologie, des ontologies modulaires peuvent être créées avec un meilleur niveau de structuration, ce qui permet de faciliter leur évolution.

# 3 Ontologies et patrons de conception d'ontologie

Dans cette section, nous présentons quelques patrons de conception d'ontologie proposés en ingénierie des ontologies notamment pour la construction d'une ontologie, pour l'alignement des ontologies et pour l'intégration des ressources à base d'ontologie.

## 3.1 Patrons de construction d'ontologie

Plusieurs travaux ont visé l'amélioration de la conception d'ontologie par l'utilisation des patrons. Mortensen et al. (Mortensen, Horridge, Musen, & Noy, 2012) ont fait une étude de cas sur l'utilisation des modèles de conception d'ontologie pour développer des ontologies biomédicales. Ils concluent que l'utilisation des patrons facilite la construction de l'ontologie et réduit les erreurs de conception. Dans ce qui suit, nous présentons des descriptions de quelques patrons d'ontologie proposés dans la littérature en mettant l'accent sur :

1. **Le type de patron utilisé** : structurel, de correspondance, logique, de contenu, etc.
2. **Le problème récurrent à résoudre** : selon le projet de développement.
3. **Le formalisme utilisé** : le langage utilisé pour décrire le patron.
4. **La validation proposée** : tout patron de conception a besoin d'une évaluation ou validation avant de l'utiliser.
5. **La réutilisation** : elle demande un mécanisme claire et une adaptation adéquate afin d'assurer l'utilisation du patron dans les différents projets et dans des contextes similaires (projet de construction d'ontologie avec le même problème récurrent de conception).

### 3.1.1 Patron de conception d'ontologie « objet avec plusieurs états » (García-Castro & Gómez-Pérez, 2013)

Les auteurs créent et appliquent le patron de conception d'ontologie « objet avec plusieurs états » dans le développement de l'ontologie « ALM iStack[6] ». Dans ce patron, un objet peut avoir plusieurs états dans le temps. Une personne par exemple peut être célibataire, mariée et plus tard peut devenir divorcée ou veuve. Ce patron de contenu permet de modéliser les différents états d'un objet et de mettre des restrictions sur cet objet. La figure 2.11 présente ce patron

[6] https://delicias.dia.fi.upm.es/ontologies/alm-istack.owl

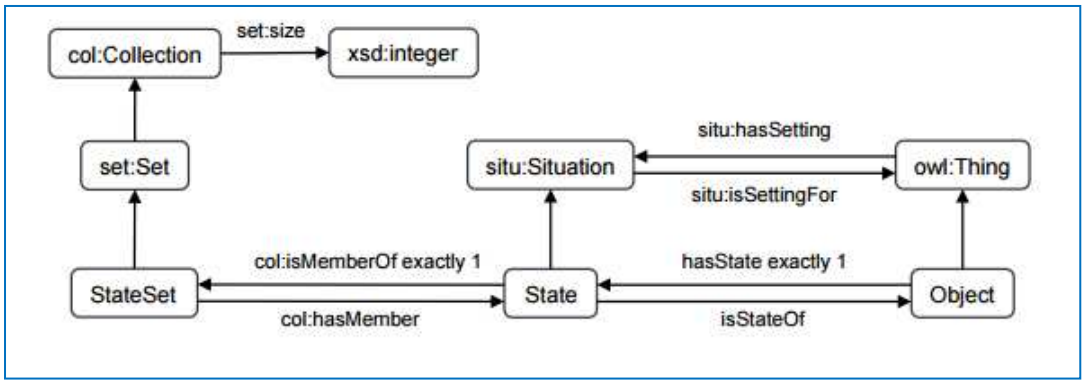



**Figure 2.11** Patron de conception d'ontologie « objet avec plusieurs « états » (García-Castro & Gómez-Pérez, 2013)

Comme on peut le voir, le patron contient trois classes principales : (1) la classe 'Object', sous classe de 'owl:Thing' qui présente les objets,(2) la classe 'State', sous classe de 'situ :Situation' qui présente les états d'objet et (3) la classe 'StateSet', sous classe de 'col :collection' et représente les ensembles des états. Le patron contient aussi des propriétés d'objet permettant de relier des objets et des états avec des restrictions de cardinalité (exactly 1). L'utilisation de classe 'owl:Thing' indique que le langage utilisé est OWL. Pour illustrer la façon d'utiliser et d'instancier le patron, les auteurs prennent un exemple de trois classes et propriétés avec les exigences d'ontologie suivantes (*cf* figure 2.12) :

1. Les états d'objets possibles sont: StateA, StateB et StateC.
2. Un objet peut avoir trois états différents.
3. Les objets dans StateA doivent avoir au moins une valeur pour la propriété property1.
4. Les objets dans StateB doivent avoir au plus une valeur pour la propriété property2.
5. Les objets dans StateC doivent avoir exactement une valeur pour la propriété property3.

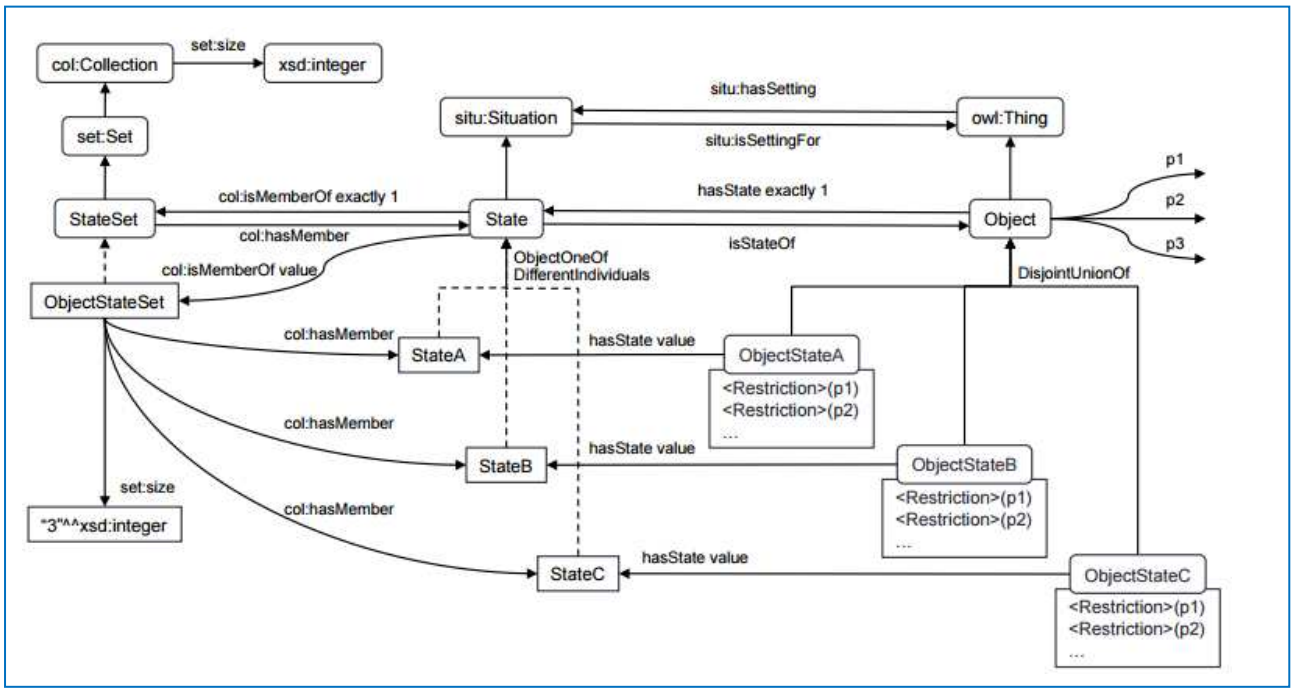



**Figure 2.12** Instanciation du patron (García-Castro & Gómez-Pérez, 2013)

Le patron est appliqué pour construire l'ontologie ALM iStack. Il est utilisé pour décrire les entités d'un logiciel de gestion de cycle de vie d'une application. Les « défauts logiciels » sont l'un des principaux concepts de cette ontologie. Une fois qu'un défaut est enregistré sur la plateforme ALM iStack, il doit avoir un certain statut et doit satisfaire un ensemble de restrictions. La figure 2.13 illustre un extrait de cette ontologie.

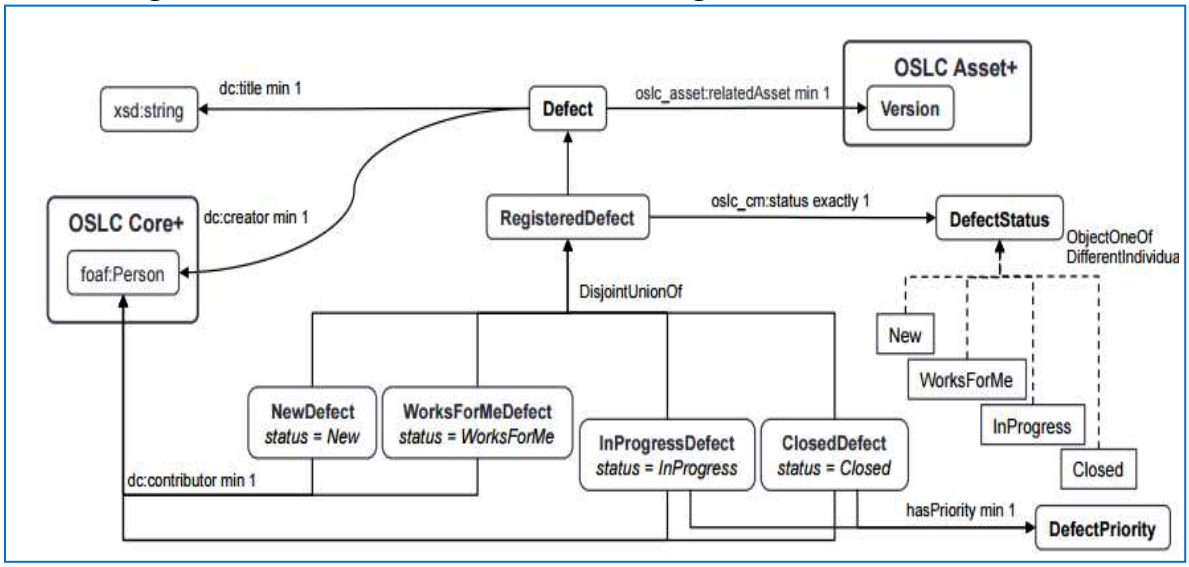



**Figure 2.13** Un extrait de l'ontologie ALM iStack (García-Castro & Gómez-Pérez, 2013)

Cependant les auteurs n'offrent aucune validation seulement une application du patron pour construire l'ontologie' ALM iStack'. Pour sa réutilisation, les étapes suivantes sont fournies pour instancier le patron :

1. Représenter tous les états possibles de l'objet comme des instances de la classe d'état en utilisant le patron de partition de valeur (Rector, 2005).
2. Définir l'ensemble des états, qui inclut tous les états, et sa taille.
3. Définir des classes pour représenter l'objet dans chacun des états.
4. Appliquer des restrictions état-spécifique à ces classes.
5. Définir la classe « Object » comme une union disjointe de ces classes.

Ce patron est aussi en ligne en tant qu'ontologie OWL réutilisable[7].

### 3.1.2 Patron de conception d'ontologie « LoSe» (Glöckner & Ludwig, 2016)

Dans le domaine des services logistiques (services de gestion-régulation de la production), les objectifs de base des activités logistique sont : offrir le bon produit, avec la bonne information, au bon endroit, au bon moment, dans la bonne qualité, avec une bonne quantité et pour le bon prix. Un patron de contenu pour un service orienté industrie nommé LoSe est proposé. La composition des services logistiques de plusieurs fournisseurs est une tâche difficile en raison de l'écart sémantique entre les différentes formulations et descriptions

[7] https://delicias.dia.fi.upm.es/ontologies/ObjectWithStates.owl

utilisées dans les IT-System[8] (Information Technology System). Avec l'utilisation du patron de conception d'ontologie pour les services logistiques, les auteurs montrent que l'écart sémantique est réduit. Les différents concepts ontologiques et non ontologiques du domaine de la logistique sont analysés et leurs principaux sont intégrés dans le patron de conception d'ontologie. La vue schématique de ce patron est présentée à la figure 2.14. Un service logistique, présenté par la classe 'LogisticsService', est mesuré par la classe 'ServiceLevelAgreements'. Il a des contraintes obligatoires (telles que des régulations légales) et non obligatoires (Environnemental, sociale et économique) ainsi que certaines capacités. Les services logistiques consomment des ressources, présentées par la classe 'Resource', pour effectuer des transformations. Ils sont connectés entre eux par des flux (classe 'flow') de type (information, marchandise, contrôle). Les transformations de prix (classe 'cost'), de qualité (classe 'quality'), de quantité (classe 'quantity), de localisation (classe 'location') et de temps (classe 'time') sont effectuées par une ressource active.

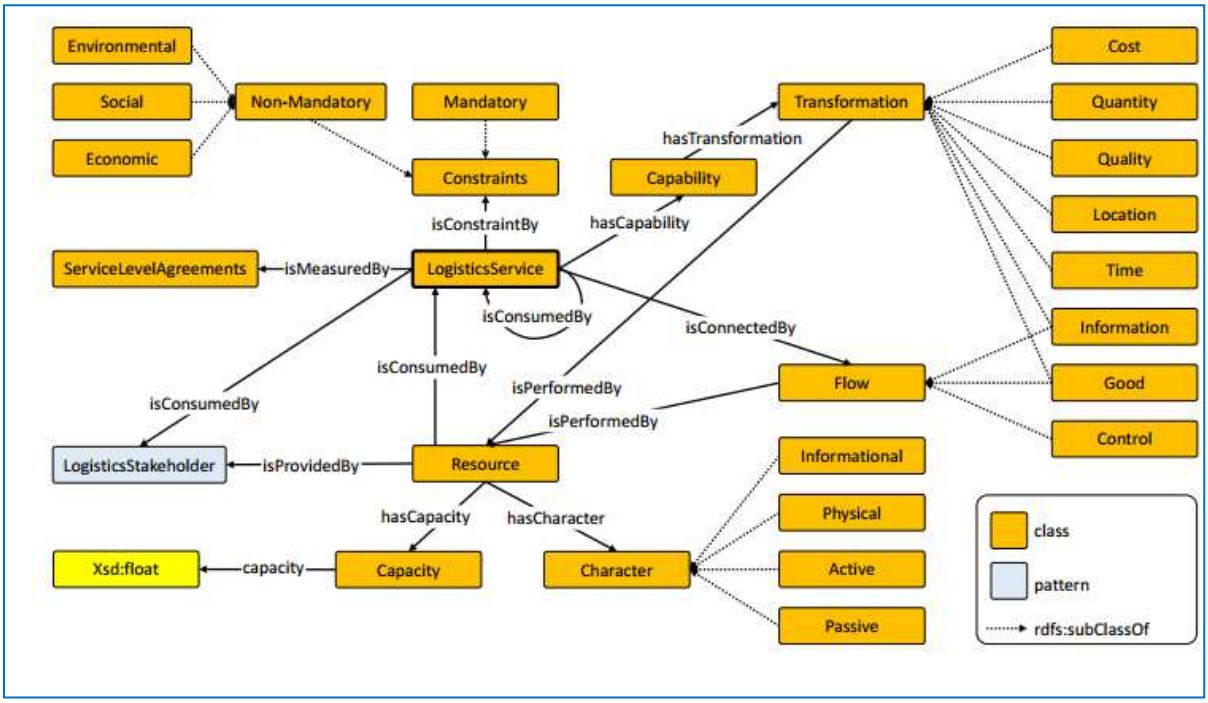



**Figure 2.14** Patron de conception d'ontologie du service logistique LoSe (Glöckner & Ludwig, 2016)

Le patron est formalisé en OWL 2 (McGuinnes, & van Harmelen, 2012) et exprimé en logique de description (Baader, 2004). L'évaluation du patron LoSe est réalisée à l'aide du 'Framework for Evaluation in Design Science Research '(FEDS) de (Venable, Pries-Heje, & Baskerville, 2016) en appliquant la stratégie simple et rapide et le scénario illustratif de l'approche 'Design Science Research Evaluation' [9] (Peffers, Rothenberger, Tuunanen, & Vaezi, 2012). Concernant la réutilisation du patron, les auteurs ne donnent pas de

[8] désigne tous les systèmes informatiques (y compris les programmes informatiques, logiciels, bases de données, matériel informatique et documentation associée), ainsi que les sites Web, y compris les processus de traitement de données, d'information, de conservation de documents, de gestion de comptes et de gestion des stocks.
[9] concerne l'évaluation des résultats de la science de la conception, y compris la théorie et les artefacts.

mécanisme ou même un ensemble d'étapes à suivre pour guider la construction d'une ontologie par ce patron.

### 3.1.3 Patron de conception d'ontologie « Winston-Part-Whole » (Shimizu, Hitzler, Paul, 2018)

Les auteurs proposent un patron logique, nommé "Winston-Part-Whole ", pour modéliser la relation (partie/tout)10 dan l'ontologie en se basant sur le travail de Winston "A Taxonomy of Part-Whole Relations" (Winston, Chaffin, & Herrmann, 1987). Ce patron permet de présenter la relation entre paiement et shopping (activité/activité), la relation entre tranche et tarte (portion/masse), la relation entre arbre et forêt (collection/membre)…etc. La figure 2.15 présente le patron.

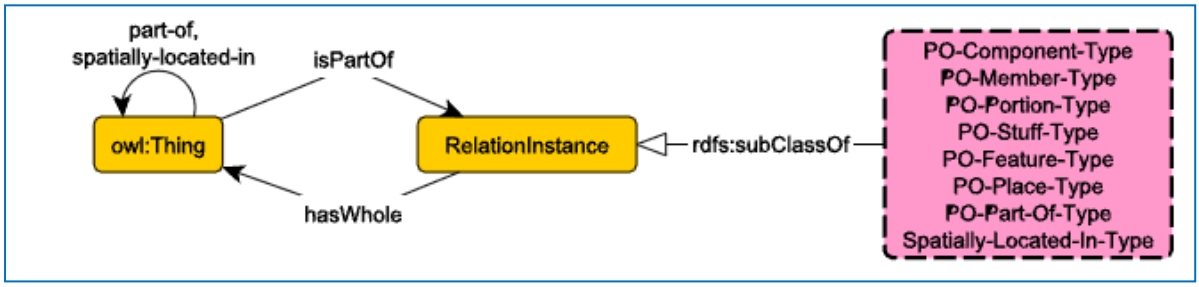



**Figure 2.15** Le patron de Winston-Part-Whole (Shimizu, Hitzler, Paul, 2018)

Pour les différents types de la relation partie/tout, les auteurs utilisent les propriétés OWL suivantes :

- composant- objet intégral: po-component
- membre-collection: po-member
- potion-masse: po-portion
- chose-objet: po-stuff
- fonctionnalité-activité: po-feature
- lieu-zone: po-place

Ils utilisent aussi d'autres relations connexes, identifiées et discutées par Winston comme la relation topologique 'spatially-located-in' et la relation part-of (comme une sous propriété de la relation partie/tout). Le patron est décrit en OWL et en logique de description. Aucune validation du patron n'est offerte. Cependant, un exemple d'application du patron dans le domaine de la science des matériaux est réalisé.

Dans la littérature, Il existe également des outils qui supportent la conception d'ontologie basée sur les patrons comme Le plugin eXtreme Design pour NeOn Toolkit (outil XD)[11]. L'éditeur d'ontologie Protégé 3.4.4[12] avait pris en charge la réutilisation automatique des

[10] Une relation a été étudie en philosophie, en linguistique, systèmes d'information géographique..etc.
[11] http://www.neon-toolkit.org
[12] http://protege.stanford.edu/download/register.html

patrons logiques suivants les bonnes pratiques: «Partition de valeur», «Enumération», «Relation N-aire», «Liste OWL» et «Liste RDF». Récemment, quelques méthodes et outils sont apparus pour guider et soutenir la réutilisation des ODPs, comme XD (pour eXtreme Design) (Daga et al., 2010) (Presutti, Daga, Gangemi, & Blomqvist, 2009). Dans cette méthode, l'activité de réutilisation des ODPs est divisée en huit tâches:

1. Identifier les exigences à traiter;
2. Identifier les modèles de conception disponibles;
3. Diviser et transformer le problème, sélectionner un problème partiel;
4. Lier le problème partiel sélectionné aux modèles de conception;
5. Sélectionner des patrons;
6. Appliquer (réutiliser) les modèles de conception sélectionnés et les composer;
7. Évaluer et réviser en respectant le problème partiel;
8. Intégrer des solutions partielles.

### 3.2 Patrons de correspondance pour l'alignement des ontologies

L'alignement consiste à la recherche des similarités entre les différents schémas locaux afin de construire l'ensemble des correspondances. Trouver des similarités entre les sources nécessite un processus d'alignement (matching). L'objectif consiste à trouver des correspondances entre des schémas en entrée et générer un mapping (correspondance) en sortie. Une méthode l'alignement est définie comme une opération de matching (Shvaiko & Euzenat, 2005) qui prend en entrée O et O' (schéma/ontologie) et qui retourne un ensemble de mapping A' ou chaque élément de A' est défini comme un 5-uplet : <id, e, e', n, R> où :

- id : est l'identifiant du mapping
- e et e' : sont les entités mises en relation (éléments XML, classes, etc.)
- n : est une mesure de confiance (valeur entre 0 et 1)
- R : est une relation : équivalence (=), plus général (⊇), disjonction (⊥)…..

Cette opération peut prendre en compte des paramètres p (un alignement préliminaire A, un seuil, pondérations…) et des ressources externes r (thésaurus, dictionnaire….) »

On peut distinguer différentes familles de méthodes par rapport au type de relation entre leurs alignements produits en sortie. En effet, la grande majorité des méthodes s'intéresse seulement à la relation d'équivalence (=). Cependant d'autres méthodes distinguent les relations de disjonction (⊥), plus général (⊇)…etc

De nombreuses méthodes d'alignement dédiées aux ontologies ont vu le jour cette dernière décennie(Otero-Cerdeira, Rodriguez-Martinez, & Gomez-Rodriguez, 2015) (Wang,

Bhagavatula, Neumann, Wilhelm, & Ammar, 2018) (Shvaiko & Euzenat, 2005) (Kasri & Benchikha, 2011). Cependant la majorité de ces méthodes n'offrent que des correspondances simples (équivalences ou subsomption). Des travaux spécifiques aux patrons d'alignement ont été présentés dans (Scharffe, Zamazal, & Fensel, 2013). Les patrons d'alignement sont des patrons de correspondances qui permettent de créer des liens sémantiques entre deux ontologies modélisant des domaines similaires. Ces patrons sont principalement destinés à aider l'utilisateur lors de la création des correspondances complexes. Scharffe (Scharffe, 2009) propose une bibliothèque de 35 patrons d'alignement d'ontologie (cf. figure 2.16) et chaque patron est décrit selon le template (nom, URI, Alias, Problème, contexte, solution, correspondance, exemple).

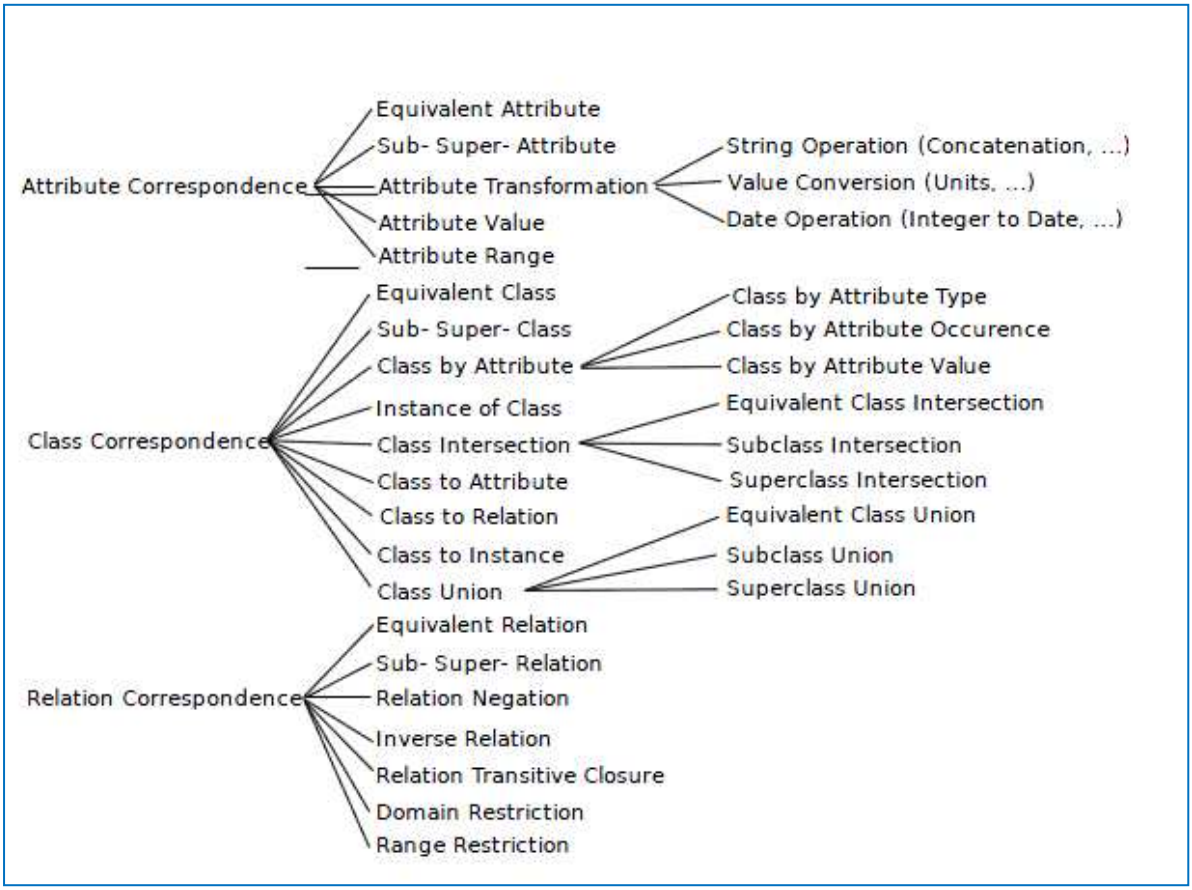



**Figure 2.16** Patron d'alignement d'ontologie (Scharffe, 2009)

Dans ce qui suit, nous présentons le patron « Relation Classe Attribut à Attribut Concaténation » (*cf* figure 2.17). Un exemple est donné en RDF / XML en utilisant le langage d'alignement (Euzenat, 2004).

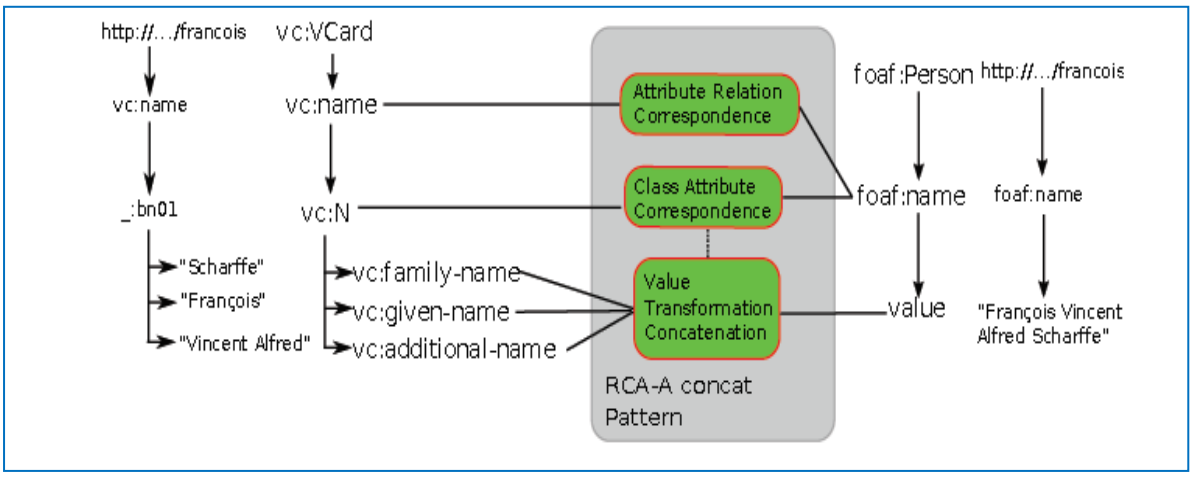



**Figure 2.17** Patron Relation Classe Attribut à Attribut Concaténation (Scharffe, 2009)

**Nom**: Patron Relation Classe Attribut à Attribut Concaténation
**URI**: http://www.omwg.org/TR/d7/patterns-library#rca-a-concat
**Problème**: Un attribut dans une ontologie a la même intension qu'une classe dans une autre ontologie. Cette classe a de nombreuses propriétés dont les valeurs concaténées forment la valeur de la première propriété de l'ontologie.
**Contexte**: il est applicable quand une relation dans une ontologie est représentée en utilisant une propriété dans l'autre ontologie.
**Solution**: Ce patron établit une correspondance entre une relation, une classe et une ou plusieurs propriétés dans une ontologie et une propriété dans une autre ontologie. La valeur de la propriété est déterminée par une fonction d'agrégation.
**Syntaxe RDF/XML**:
correspondence ::= <Cell>
<entity1><Relation rdf:about="R"/>
</entity1>
<entity2><Property rdf:about="P"/></entity2>
</Cell>
<Cell>
<entity1><Class rdf:about="C"/>
</entity1>
<entity2><Property rdf:about="P"/></entity2>
</Cell>
<Cell>
<entity1><Property><or><Collection>
<item><Property rdf:about="Q1"/></item>
. . . <item><Property rdf:about="Qn"/></item></Collection>
</or></Collection> <transf rdf:resource="transf:concat"/>
</Property></entity1>
<entity2><Property rdf:about="P"/></entity2>
</Cell>
**Exemple**:
<Cell>

```
<entity1><Relation rdf:about="vc:name"/>
</entity1>
<entity2><Property rdf:about="foaf:name"/></entity2>
</Cell>
<Cell>
<entity1><Class rdf:about="vc:N"/>
</entity1>
<entity2><Property rdf:about="foaf:name"/></entity2>
</Cell>
<Cell>
<entity1><Property><or><Collection>
<item><Property rdf:about="vc:family-name"/></item>
<item><Property rdf:about="vc:given-name"/></item>
<item><Property rdf:about="vc:additional-name"/></item><Collection>
        </or></Collection> <transf rdf:resource="transf:concat">vc:given-name"
"vc:additional-name" "vc:family-name</transf>
</Property></entity1>
<entity2><Property rdf:about="foaf:name"/></entity2>
</Cell>
```

**Patrons en relation**: Equivalent Propriété Correspondance, Propriété valeur Correspondance.

### 3.3 Patrons d'intégration des sources de données

Dans un système d'intégration, un schéma global (ontologie globale) est construit pour répondre aux requêtes des utilisateurs. Dans ce cas, les mappings (les transformations) sont employés pour décrire des relations entre le schéma global et les schémas locaux.

Un système d'intégration est défini formellement par un triplet (*G, S, M*) où *G* représente le schéma global, *S* représente le schéma de la source, et *M* représente la transformation entre G et S. L'approche GAV (Global-as-View) (Batini, Lenzerini, & Navathe, 1986) (Halevy, 2001), consiste à définir le schéma global en fonction des sources. Les éléments du schéma global sont associés aux vues sur les schémas de sources locales (la transformation *M*). La requête est définie sur le schéma global puis les définitions des vues sont utilisées pour la réécrire sous forme des requêtes distribuées sur les sources locales. L'une des limites de l'approche GAV c'est que l'évolution des schémas locaux est difficile à gérer et peut mettre en cause tout le système.

Au contraire, dans l'approche LAV (Local-as-View) (Batini, Lenzerini, & Navathe, 1986) (Halevy, 2001), on définit le schéma global et on décrit les sources par rapport à ce schéma où les éléments d'un schéma de source locale sont associés aux vues sur le schéma global. La

requête au niveau du schéma global est traitée en impliquant sa réécriture par l'utilisation des vues locales. La limite de cette approche est que le changement de schéma global implique la modification des règles de définition des schémas locaux qui doivent être redéfinies.

D'autres approches qui mixent les deux approches précédentes sont apparues comme l'approche BAV (Both-as-View) (McBrien, & Poulovassilis, 2003) qui fait un mapping bidirectionnel entre chaque source locale et le schéma global pour traiter une requête. Et aussi l'approche BGLAV (BYU-Global-Local-as-View) qui génère automatiquement le mapping et permet de basculer en mode LAV ou GAV selon la tâche à traiter l'ajout de nouvelles sources ou le traitement de requête (Xu & Embley, 2004).

L'approche de construction d'ontologie à base de ODP consiste à spécifier un ensemble de patrons de conception d'ontologies dont chacun présente une ontologie partielle qui formalise une seule notion clé (Krisnadh et al., 2015). OceanLink est un projet dans le domaine des sciences de mer qui vise à améliorer la récupération et la réutilisation des données via les ontologies, les technologies du Web sémantique, et données liées. OceanLink relève le défi d'utiliser les techniques avancées du Web sémantique en particulier les données liées et ouvertes (Bizer, Heath, & Berners-Lee, 2009) et les patrons de conception d'ontologies (ODP) (Gangemi, 2005). Dans OceanLink, une ontolgie de domaine de croisière océanographique a été développée pour construire le schéma global du système d'intégration (*cf* figure 2.18). Dans ce projet, le patron de conception OceanLink est utilisé comme un modèle pour construire le schéma global.

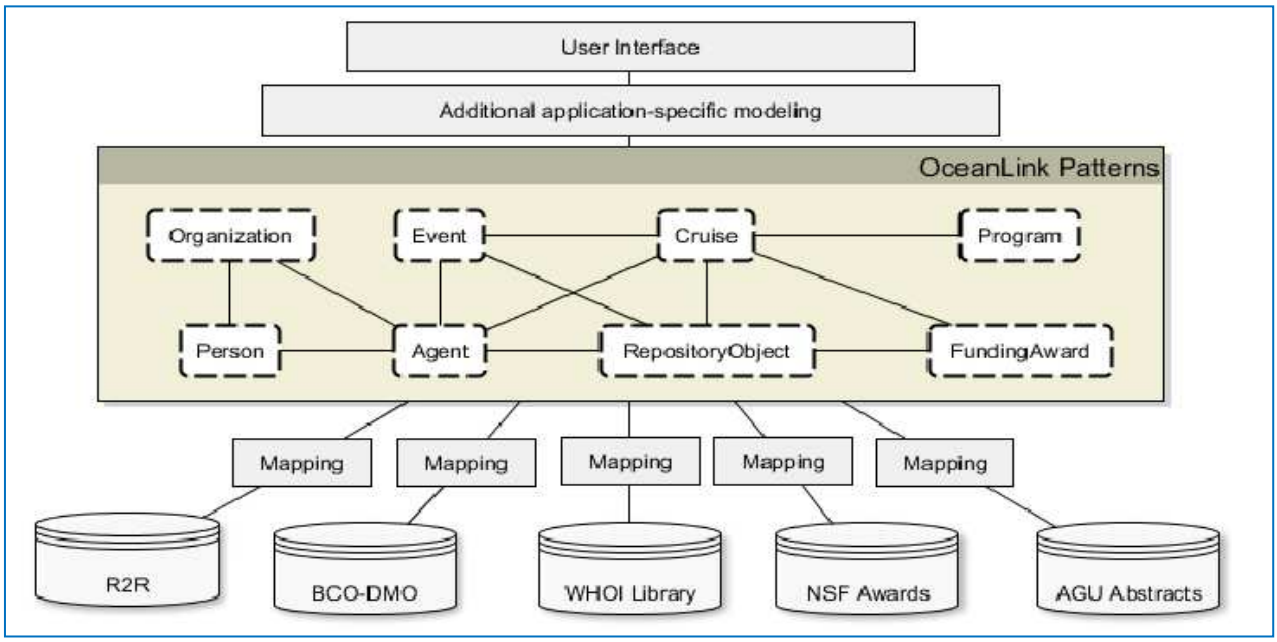



**Figure 2.18** OceanLink : intégration à base patron de conception (Krisnadh et al., 2015)

Une collection d'ODP (patron event, patron Agent, patron person et patron cruise...etc) joue le rôle d'une couche intermédiaire entre l'interface utilisateur et les sources de données. L'interface utilisateur traduit les requêtes d'utilisateur en requêtes fédérées à l'aide des vocabulaires donnés par les patrons. Les requêtes sont envoyées aux sources dont chacune est accompagnée d'une couche d'alignement (mapping) qui traduit la requête en termes de son modèle de données.

Actuellement, les données suivantes sont en cours d'intégration dans le Framework:

1. données de recherche de navires « R2R »;
2. données océanologiques, biologiques et chimiques « BCO-DMO »;
3. rapports de croisières et données de thèses de doctorat de la bibliothèque du laboratoire de biologie marine de Woods Hole de l'établissement océanographique (MBLWHOI);
4. données sur les bourses financées par la NSF; et
5. présentations de la conférence et résumés de l'Union Geophysical American (AGU).

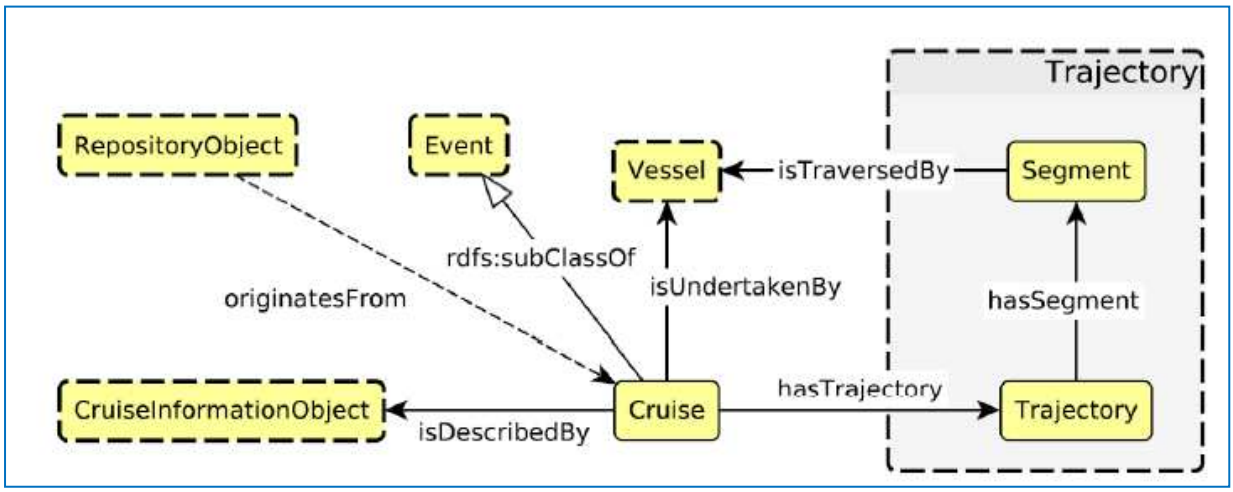



**Figure2.19** Vue générale du patron « Cruise » (Krisnadh et al., 2015)

Le patron le plus important dans la collection ODP de OceanLink est le paton « CruiseFigure ». La figure 2.19 montre sa vue générale. La relation entre les classes « Cruise », « Trajectory » et « Vessel » implique une classe interne du patron Trajectory. Puisqu'une classe « Cruise » est une sorte de « Event », la classe « Cruise » est présenté comme sous-classe de la classe générique « Event ». Les ornements du patron « Cruise » sont attachés par une instance de la classe « CruiseInformationObject ».

Le patron de contenu « Cruise » regroupe trois patrons existants: « Trajectory », « Event » et « Iinformation Object ». Cette combinaison peut rendre le patron « Cruise » un peu compliqué, tant pour les fournisseurs de données que pour les utilisateurs. Pour les aider dans la lisibilité et faciliter l'utilisation, il est souvent utile de spécifier des «raccourcis» sémantiques qui capturent certaines requêtes courantes sur le patron. Ces raccourcis, appelés \ emph {views}, peuvent être définis en fonction des besoins de l'application et généralement

sont exprimés sous forme de règles pouvant être traduites en axiomes OWL. Par exemple, La propriété « hasChiefScientist » relie une croisière et son chef scientifique:

$$\begin{aligned} &Cruise(x) \wedge providesRole(x, y) \wedge isPerformedBy(y, z) \\ &\quad \wedge Person(z) \wedge hasRoleType(y, chief\_scientist) \\ &\rightarrow hasChiefScientist(x, z) \end{aligned}$$

D'après (Krisnadh et al., 2015), avec ce Framework, l'intégration est flexible en rejoignant facilement d'autres sources de données. En outre, la modularité des patrons dans l'approche ODP permet d'étendre le schéma global assez facilement en cas de besoin.

## 4 Discussion

Malgré le fait que l'utilisation des patrons de conception d'ontologie comporte des éléments clé pour la réutilisation des ontologies, le développement d'un patron sans validation, sans processus de création et d'utilisation, sans processus clair de réutilisation et sa forte indépendance à un langage donné limitent son utilité comme moyen puissant qui nous permet de créer des ontologies réutilisables. Notre analyse des patrons de conception d'ontologie nous a amené aux constats suivants :

1. **Validation du patron :** Un patron de conception d'ontologie doit être validé avant de le réutiliser. On observe que seule les auteurs du patron « LoSe » réalisent une évaluation à l'aide du 'Framework for Evaluation in Design Science Research '(FEDS) mais les autres n'offrent aucune validation. Cependant, l'utilisation du cadre d'évaluation 'FEDS' n'offre pas une validation qui assure la solution au problème récurrent mais permet seulement de guider le développement en se basant une approche qualitative.
2. **Processus clair de création et d'utilisation du patron :** les patrons de conception sont considérés comme un moyen de capitalisation des bonnes pratiques (expériences) appliquées à la conception des ontologies. Il ne s'agit pas de fragments d'ontologie mais d'une manière standardisée permettent de résoudre un problème récurrent. Cependant la majorité des patrons de conception d'ontologie sont donnés comme un couple (problème/solution) avec un exemple d'application en cachant toute expertise et sans documentation.
3. **Processus clair de réutilisation du patron :** La réutilisation est l'avantage important derrière l'utilisation des patrons cependant les travaux présentés n'offrent pas un mécanisme clair de réutilisation. Seule les auteurs du patron « objet avec plusieurs états » offrent un ensemble des pour instancier le patron comme un processus de réutilisation.
4. **La dépendance forte conception implémentation :** Concernant le langage utilisé pour décrire et implémenter l'ontologie, les patrons étudiés sont décrits en OWL ou en logique de description. Cependant ce choix rend la phase d'implémentation d'ontologie dépendante à la phase de conceptualisation.

# 5 Conclusion

Un des buts essentiels des patrons de conception d'ontologies est d'une part de faciliter la réutilisation et d'autre part d'assurer la qualité en se basant sur des expériences tirées (bonnes pratiques) de la résolution du même problème de conceptualisation récurrent.

Dans ce chapitre, l'analyse que nous avons effectuée sur plusieurs patrons de conception d'ontologies nous ont conduit aux résultats suivants :

- La majorité des patrons de conceptions d'ontologie sont décrits en OWL. Ils sont fortement liés à un langage d'implémentation.
- Ils sont décrits sans mécanisme clair dé réutilisation.
- Ils sont proposés comme des briques ontologiques contenant des bonnes pratiques à réutiliser. Cependant aucune validation ne s'accompagne pour les prouver.

Ces trois résultats nous ont conduits à proposer un patron de conception d'ontologie (MultiViewPoints Pattern MV2P) en utilisant UML pour être indépendant de tout langage d'implémentation. Pour réutiliser le patron, nous adaptons l'opération d'imitation (Coad, 1995) (Rieu, Giraudin, & Saint Marcel, 1999). Concernant la validation, nous proposons un Framework d'évaluation en se basant sur l'extension de l'analyse formelle « structures des patrons » (Ganter & Kuznetsov, 2001) qui sera présentée dans le chapitre suivant.

Chapitre 3

# L'Analyse Formelle de Concepts et Ontologies

## Sommaire



## 1 Introduction

L'analyse formelle de concepts (AFC) est un formalisme mathématique d'analyse et de représentation des connaissances pour la fouille de données qui fournit un cadre théorique pour l'apprentissage de hiérarchies de concepts. La hiérarchie résultant de l'AFC est appelée treillis de Galois ou treillis de concepts (Ganter, 1984) (Ganter & Wille, 1999).

A la fin du 19ème siècle, la notion de treillis a été introduite comme une structure algébrique munie de deux opérateurs appelés borne inférieure et borne supérieure. Cependant, le premier ouvrage de référence de la théorie des treillis a été écrit par de Birkhoff en 1940 (Calì, Frosini, Poulovassilis, & Wood, 2014). Le treillis de Galois établissant une relation entre deux ensembles d'objets est à la base de plusieurs méthodes de classification conceptuelles. Il consiste à regrouper des objets en classes qui permettront de matérialiser les concepts du domaine d'étude. Ces objets sont discriminés en fonction de leurs propriétés communes.

Dans les dernières dizaines années, l'AFC a été utilisée avec succès dans le domaine de l'ingénierie des logiciels (Godin, Missaoui, & April, 1993) (Tilley, Cole, Becker, & Eklund, 2005). Mais récemment, Il a été montré qu'elle pouvait être utilisée pour acquérir, traiter, et modéliser des connaissances dans la construction d'une ontologie. Pour les ontologies, l'AFC peut fournir un cadre structurel et formel qui supporte un processus de classification conceptuelle. Dans la littérature il existe plusieurs travaux qui se basent sur l'analyse formelle de concepts (AFC) pour l'extraction d'ontologies à partir de textes (Bendaoud, Toussaint, & Napoli, 2010), la fusion d'ontologies Stumme & Maedche, 2001), la recherche d'information Azmeh, Huchard, Tibermacine, Urtado, & Vauttier, 2008), la sélection de services Web (Carpineto & Romano, 2005), l'extraction de connaissances (Lakhal & Stumme, 2005), …etc.

L'AFC ne permet de traiter que des attributs binaires et ne prend en compte que les relations entre les objets. Son extension, l'analyse relationnelle de concepts permet de regrouper un ensemble d'objets, non seulement par l'ensemble des attributs binaires qu'ils partagent mais aussi par les relations inter-objets. Elle permet l'extraction de concepts formels à partir d'un ensemble d'objets formels combinant des attributs binaires et des relations qu'ils entretiennent entre eux (Rouane-Hacene, Huchard, Napoli, & Valtchev, 2013). Dans les différents domaines du monde réel, les données sont bien souvent d'origine complexe et ne peuvent pas être traitées par des tables binaires. Une autre extension, les Structures de Patrons, (Ganter & Kuznetsov, 2001) permet la construction des treillis des objets complexes en ordonnant partiellement leurs descriptions.

Nous présentons dans ce chapitre un bref tour d'horizon des fondamentaux des treillis et des concepts formels auxquels nous avons eu recours pour développer notre travail. Nous introduisons, dans la section 2, les notions de base de l'analyse formelles de concepts, sa présentation, sa visualisation et les algorithmes de construction des treillis. Les sections 3 et 4 présentent respectivement les extensions qui nous intéressent; l'analyse relationnelle de concepts et les structures des patrons. Nous présentons quelques applications en ingénierie ontologique qui utilisent cette assise théorique dans la section 5. La section 6 conclut le chapitre.

## 2 Analyse Formelle de Concepts

L'analyse formelle de concepts (AFC) (Ganter & Wille, 1999) est un procédé qui permet de découvrir tous les concepts et les structurer hiérarchiquement en regroupant des objets ayant des attributs communs. La hiérarchie résultante est appelée treillis de Galois (Barbut & Monjardet, 1970) ou treillis de concepts (Ganter & Wille, 1999). L'analyse de concepts formels est une méthode mathématique appliquées qui consiste à restructurer la théorie des treillis (Birkhoff, 1940). Dans ce qui suit, nous présentons les notions fondamentales de l'AFC (Ganter & Wille, 1999) .

**Définition 3.1 (Contexte Formel).** Un contexte formel est un triplet K = (G, M, I) où G est un ensemble d'objets formels, M est un ensemble d'attributs formels et I est une relation binaire ou d'incidence entre G et M vérifiant :

- $I \subseteq G \times M$

- Une paire (g, m) signifie que l'objet g ∈ G possède l'attribut m ∈ M.

**Définition 3.2 (connexion de Galois dans un Contexte Formel).** Soit K = (G,M, I) un contexte formel. Pour tout A ⊆ G et B ⊆ M, on définit deux opérations de dérivation notées par( $'$ ):

- $A' = \{m \in M \mid g\ I\ m \text{ for all } g \in A\}$
- $B' = \{g \in G \mid g\ I\ m \text{ for all } m \in B\}$

**Définition 3.3 (Concept Formel).** Dans un contexte formel $K = (G, M, I)$, une paire $(A, B)$ où $A \subseteq G$ et $B \subseteq M$ vérifiant A′= B et $B' = A$ est appelé concept formel. $A$ est appelé l'extension du concept formel et $B$ son intension. $A$ est l'ensemble maximal d'objets partageant l'ensemble des attributs de l'ensemble $B$.

**Définition 3.4 (Relation de Subsomption).** Soit $\mathfrak{B}(G, M, I)$ l'extraits des concepts formels associés au contexte formel $\mathbb{K} = (G, M, I)$. Ces concepts formels sont ordonnés par une relation de subsomption (⊑) entre un concept $(A_1, B_1)$ et un super concept $(A_2, B_2)$ qui est défini comme suit :

$$(A_1, B_1) \sqsubseteq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \text{ et } B_2 \subseteq B_1.$$

Cette relation peut être interprétée comme une relation de généralisation/spécialisation entre les concepts formels. Un concept est plus général qu'un autre concept s'il contient plus d'objets dans son extension et ces objets ont moins d'attributs en communs. De la même façon, un concept est plus spécifique qu'un autre s'il contient moins d'objets dans son extension. Ces objets ont plus d'attributs partagés.

**Définition 3.5 (Treillis de Concept).** La hiérarchie de $\mathfrak{B}(G, M, I)$ créée par la relation d'ordre ou subsomption (⊑) associée au contexte formel $\mathbb{K} = (G, M, I)$ est appelé *treillis de concept* ou *treillis de Galois* noté $\underline{\mathfrak{B}}$(G, M, I) ou $\underline{\mathfrak{B}}(\mathbb{K})$.

### 2.1 Représentation et visualisation des concepts formels

Un contexte formel $\mathbb{K} = (G, M, I)$ peut être représenté dans un tableau ou visualisé dans un diagramme. Deux représentations sont présentées ci-dessous.

#### 2.1.1 Représentation tabulaire

Un contexte formel $\mathbb{K} = (G, M, I)$ peut être représenté dans un tableau dans lequel les lignes sont les objets formels, les colonnes sont les attributs formels et une croix ou coche est placée à la case d'intersection d'une ligne et d'une colonne si l'objet possède l'attribut correspondant sinon la case est vide. La table 3.1 présente un exemple de contexte formel nommé TourismContext représentant un ensemble d'objets de type Hôtel. Les objets du contexte sont PULLMANparis, MOLITORParis, SOFITELParis, LE ROYALMONCEAU RAFFLESParis et ANTIN TRINITEParis. Un Objet d'Hôtel possède un nombre donné d'étoiles qui sont les attributs du contexte. Ces attributs sont 1star, 2star , 3stra, 4stra et 5star.

| TourismContext | 1star | 2star | 3star | 4star | 5star |
|---|---|---|---|---|---|
| **PULLMANparis** | | | | X | |
| **MOLITORParis** | | | | | X |
| **SOFITELParis** | | | | | X |
| **LE ROYALMONCEAU RAFFLESParis** | | | | | X |
| **ANTIN TRINITEParis.** | X | | | | |

**Table 3.1** Exemple du contexte TourismContext

### 2.1.2 Représentation par diagramme

Le diagramme de Hasse est une représentation visuelle d'un ordre fini. Ce digramme peut être utilisé pour représenter un treillis de concepts $\underline{\mathfrak{B}}(\mathbb{K})$ correspondant au contexte formel $\mathbb{K}$ = (G, M, I). Les nœuds de ce digramme représentent les concepts formels $\mathfrak{B}$(G, M, I), la relation de subsomption ($\sqsubseteq$) est représentée par un arc entre deux nœuds. La représentation des concepts peut être réduite (dite aussi étiquetage réduit) pour améliorer la lisibilité du treillis, qui consiste à reporter les étiquettes des intensions sur le concept le plus général possible dans le treillis et les étiquettes des extensions sur le concept le plus spécifique.

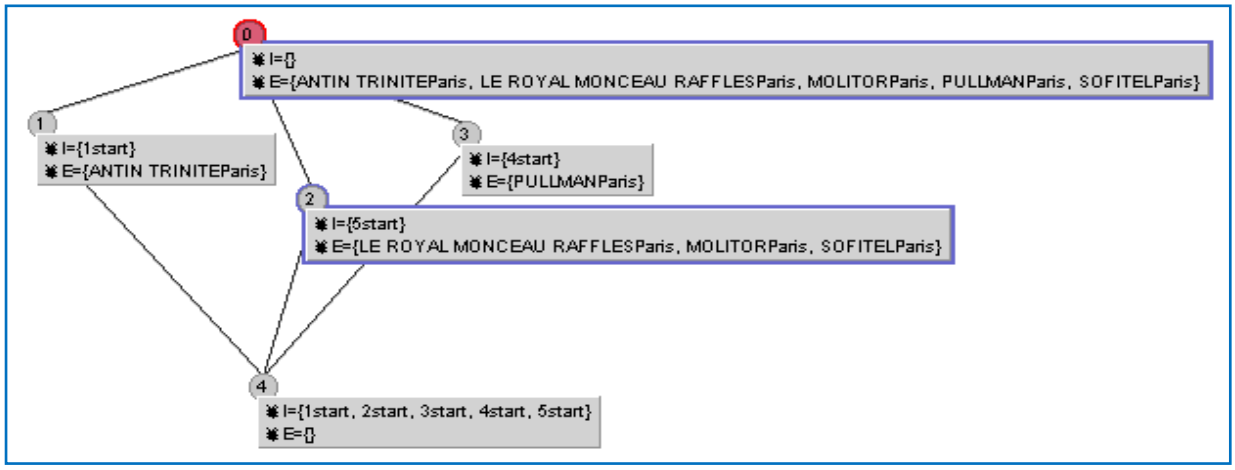



**Figure 3.1** Diagramme de Hasse du contexte formel TourismContext du tableau 3.1

## 2.2 Algorithmes et outils de construction de treillis de concepts

Dans la littérature il y a plusieurs travaux qui traitent la génération des treillis de concepts formels et la visualisation de leurs graphes sous forme de diagrammes de Hasse.

### 2.2.1 Les algorithmes de construction

Plusieurs algorithmes ont été proposés pour la construction de treillis qui sont classifiés en trois familles différentes : les algorithmes non incrémentaux, les algorithmes incrémentaux et les algorithmes d'assemblage.

1. **Les algorithmes non incrémentaux** : cette famille d'algorithmes reconstruit le treillis si l'ensemble des objets ou des attributs sont modifiés. Parmi les algorithmes de cette famille: Ganter (Ganter & Wille, 1999), Chein (Chein, 1969), Norris (Norris, 1978), Bordat (Bordat, 1986) qui ont fait l'objet des études comparatives (Kuznetsov & Obiedkov, 2002) (Godin, 1991).
2. **Les algorithmes incrémentaux** : cette famille d'algorithmes permet de modifier localement les nœuds si leurs attributs ou objets augmentent et d'insérer éventuellement de nouveaux nœuds (Godin & Valtchev, 2005).
3. **Les algorithmes d'assemblage** : cette famille d'algorithmes est basée sur la fusion de deux contextes et leurs treillis. Un exemple de ce type d'algorithme est celui de Valtchev (Valtchev & Missaoui, 2001).

La construction du treillis de concepts implique des opérations pour la découverte des la liaison des concepts. Dans cette section, nous présentons les principes de fonctionnement des algorithmes choisis pour la construction du treillis de concepts : Bordat (Bordat, 1986) et Nourine et Raynaud (Nourine & Raynaud, 1999). Les deux algorithmes construisent à la fois les concepts et les liens entre les concepts.

- **Algorithme de Bordat (Bordat, 1986)**: Il se charge de générer le graphe de Hasse de la relation directe d'ordre entre les concepts (couverture) en plus de la génération des concepts. L'algorithme génère pour un concept $(A_1, B_1)$ les concepts $(A'_1, B'_1)$ qu'il couvre. La relation de couverture permet ainsi d'établir un ordre de subsomption entre les concepts. Un concept $(A_1, B_1)$ couvre $(A'_1, B'_1)$ si $(A'_1, B'_1)$ est successeur de $(A_1, B_1)$ directement. Le treillis est construit en parcourant la liste des concepts séquentiellement. Pour chaque concept de la liste, l'algorithme construit les concepts qu'il couvre et les insère dans la liste s'ils n'existent pas déjà,. L'algorithme s'arrête quand le parcours de la liste se termine sans l'insertion de nouveaux concepts.

---

*1.**Algorithm*** Bordat ***(In**:X extension,Y intension, **Out**: L lattice)*

---

*2:let* $c_i \leftarrow \{ X, Y \}$
*3: // initialize L*
*4:* $L \leftarrow L \cup c_i$
*5: Let covering← covering-Infimum (* $X, Y$*)*
*6:for each (A,D) concept ∈ covering do*
*7:if (A,D) ∈ L then*
*8: child=*Bordat *(A,D)*
*9: else*
*10: child=retrieve((A,D), L)*
*11: Createrows(*$c_i$ *, child)*
*12: Return(*$c_i$*)*

*1.**Function** covering-Infimum (**In**: $X$ extension , $Y$ intension, **Out**: $DirectChildren$)*

*2: Let covering*$\leftarrow \emptyset$
*3: let* $Z \leftarrow Y$
*4:while NextObject(X, Z) do*
*5:* $D \leftarrow \{object\}$
*6:* $D' \leftarrow parent(D)$
*7:* $otherObject \leftarrow object$
*8: while* $otherObject \neq$ *last(X) do*
*9:* $otherObject \leftarrow next(X)$
*10: if* $parent(otherObject) \cap D' \nsubseteq Z$ *then*
*11:* $D \leftarrow D \cup \{otherObject\}$
*12:* $D' \leftarrow D' \cap \{otherObject\}$
*13:endwhile*
*14: if* $D' \cap Z = Y$ *then*
*15:* $covering \leftarrow covering \cup \{(D, D')\}$
*16:* $Z \leftarrow Z \cup D'$
*17: if* $||DirectChildren|| = 0$ *then*
*18:* $covering \leftarrow \{(\emptyset, Y')\}$
*19: Return* $DirectChildren$

La complexité de l'algorithme est O(|A||B|$^2$|N|) où |A| étant le nombre d'objets dans le contexte, |B| le nombre d'attributs et |N| le nombre de concepts formels dans le treillis obtenu. Avec les treillis de petite taille, les performances de Bordat sont bonnes mais s'altèrent rapidement avec un grand volume de données. Pour cela, d'autres algorithmes sont proposés dont l'algorithme de Nourine et Raynaud (Nourine & Raynaud, 1999).

- **Algorithme de Nourine et Raynaud (Nourine & Raynaud, 1999)**: il est plus récent et plus performant que l'algorithme de Bordat. Il se charge de générer des concepts organisés dans un arbre lexicographique puis identifier de la relation d'ordre entre les concepts (couverture) pour créer le graphe de Hasse. L'algorithme commence par la construction d'un arbre lexicographique contenant les concepts. Cet arbre offre une structure de recherche efficace qui permet de retrouver rapidement un concept à partir de son extension. La complexité de l'algorithme est (O((|A| + |B|)|A||N|))

*1.**Algorithm*** NourineAndRaynaud ***(In**:C concepts, **Out**: L lattice)*

*2: // initialize* $count_{ci}$
*3:for each* $count_{ci}$ *do*
*4:* $count_{ci} \leftarrow 0$
*5:for each* $c_i \in C$ *do*
*6:* $condidate \leftarrow \emptyset$
*7:for each* $a \in$ *intension*$(c_i)$ *do*
*8: D*$\leftarrow$ *getExtension(a)* $\cap$ *extension*$(c_i)$
*9:* $condidate \leftarrow$ *getConcept(C, D)*
*10:* $count_{condidate} \leftarrow count_{condidate} + 1$

*11:if* $count_{condidate}$ *+| intension*$(c_i)$ *|=| intension(condidate ) | then*
*12: addrows*$(c_i$ *, condidate)*
*13:end for*
*14: Reset* $count_{ci}$
*15:end for*

### 2.2.2 Les outils de construction

Il existe plusieurs outils qui permettent d'éditer des contextes formels, et de construire le treillis de concepts associé. Ces outils permettent de tracer des treillis de concepts à partir de contextes formels avec la possibilité de l'explorer interactivement. Un ensemble des outils libre est disponible à partir de la page web FCA[1] (Villerd, 2008). Nous avons choisi dans le contexte de notre travail d'utiliser Galicia pour représenter nos treillis, car cet outil offre une représentation lisible et simple. Plusieurs algorithmes de génération de sous-hiérarchies de Galois (figure 3.2) ont été élaborés et implémentés dans Galicia (Valtchev, Grosser, Roume, & Rouane-Hacene, 2003).

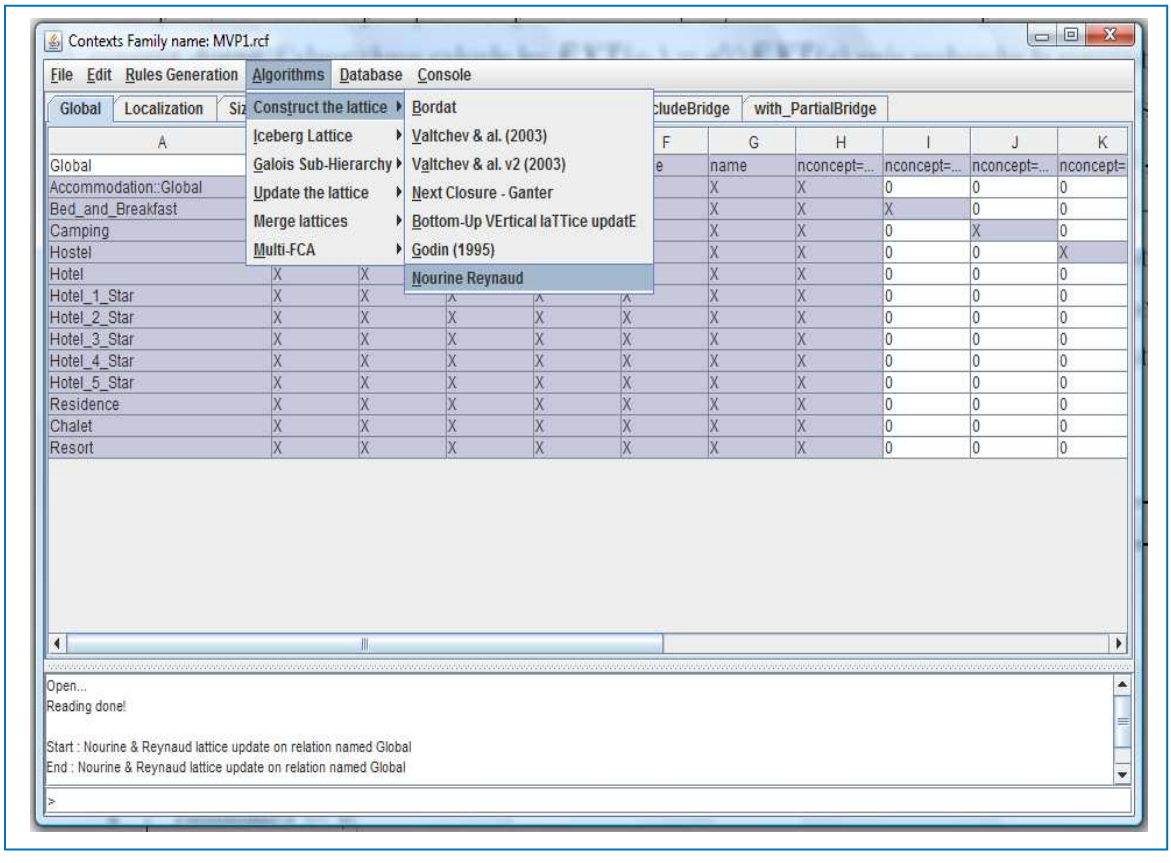



**Figure 3.2** Outil de visualisation Galicia

Nous utilisons l'AFC pour construire les hiérarchies dans l'ontologie. Cependant, l'inconvénient majeur de l'FCA est la taille du treillis qui augmente considérablement en fonction du nombre d'objets et d'attributs. En effet pour un contexte formel à |O| objet et |A| attributs, on compte O ($2^{\min(|O|, A|)}$) concepts (Villerd, 2008). A cet effet, des extensions de

[1] http://www.upriss.org.uk/fca/fca.html

l'AFC sont proposées notamment l'analyse relationnelle de concepts qui supporte d'autres relations plus que l'hiérarchie et l'extension « structures de patrons » qui sera utilisée pour valider notre patron de conception d'ontologie.

## 3 Analyse relationnelle de concepts

L'analyse relationnelle de concepts (ARC) est une extension de l'AFC qui est apparue comme une étude de relation entre les objets dans un contexte formel. U. Priss (Priss, 2006) dans ses travaux de thèse a proposé l'analyse relationnelle de concepts pour prendre en compte des relations sémantiques dans les bases de donnés lexicales qui sont représentées sous la forme de contextes formels linguistiques. Récemment, plusieurs travaux ont redéfini l'Analyse Relationnelle de Concepts (ARC) (Rouane-Hacene, 2006) (Huchard, Rouane-Hacene, Roume, & Valtchev, 2007) (Rouane-Hacene, Huchard, Napoli, & Valtchev, 2013) comme une extension de l'AFC permettant de prendre en compte, en plus des caractéristiques des objets, les relations entre ces objets. L'ARC nécessite un ou plusieurs contextes formels ainsi que des contextes relationnels qui forment une famille de contextes relationnels (FCR). Un contexte relationnel décrit une relation entre les objets de deux contextes formels (qui ne sont pas forcément différents). Dans l'approche que nous proposons dans le chapitre cinq, nous utilisons les définitions de Rouane-Hacene et al. comme suit (Rouane-Hacene, Huchard, Napoli, & Valtchev, 2013):

**Définition 3.6 (Famille de contextes relationnels).** Une famille de contextes relationnels FCR est un couple $(\mathbb{K}, \mathbb{R})$ où $\mathbb{K}$ est un ensemble de contextes formels $\mathcal{K}_i=(O_i, A_i, I_i)$ et $\mathbb{R}$ est un ensemble $r$ où $r \subseteq O_i X O_j$ où $O_i$ et $O_j$ sont des objets des contextes $\mathcal{K}_i$ et $\mathcal{K}_j$ de $\mathbb{K}$ et appelés domaine $dom(r)$ et co-domaine $ran(r)$ de $r$ respectivement. Les fonctions de domaine et co-domaine sont définies comme suit.

- $dom: R \rightarrow O$ où $dom(r) = O_{j1}$ *si est seulement si pour chaque* $(x, y) \in r, x \in O_{j1}$ ,
- $ran: R \rightarrow O$ où $ran(r) = O_{j2}$ *si est seulement si pour chaque* $(x, y) \in r, x \in O_{i2}$ .

Une manière d'assembler des fonctions en respectant leurs domaines la fonction de contexte $rel$ a été fournie.

**Définition 3.7 (Fonction des contextes $rel(\mathcal{K})$).** La famille des relations créée dans un contexte donné est définie par :

- $rel : \mathbb{K} \rightarrow \wp(\mathbb{R})$ où $rel\big(\mathcal{K} = (O, A, I)\big) = \{r \in \mathbb{R} | dom(r) = O\}$.

L'AFC classique repose sur la représentation des connaissances du monde réel en utilisant une relation binaire entre un ensemble d'objets et un ensemble d'attributs monovalués. Cependant les attributs peuvent être multi valeurs. Pour cela en ARC, les attributs multivalués sont transformés en ensembles d'attributs monovalués (binaires) par l'intermédiaire de l'échelonnage conceptuel pour former un contexte monovalué.

**Définition 3.8 (Echelonnage conceptuel).** L'échelonnage conceptuel consiste à transformer chaque attribut multivalué en un ensemble d'attributs monovalué qui forment un contexte formel monovalué appelé échelle conceptuelle de l'attribut multivalué. Ce mécanisme permet de remplacer chaque valeur d'attribut multivalué par une relation binaire et l'échelle conceptuelle est généralement définie par un expert du domaine pour attribuer les valeurs d'attributs.

Prenons l'exemple du contexte numérique multivalué suivant où les objets sont des accommodations et les attribut " surface " et " distance au centre " prennent des valeurs numériques. Les données numériques sont présentées dans la table 3.2 et 3.3

| | **Surface ($m^2$)** |
|---|---|
| **Villat1Z** | 280 |
| **Maison22E** | 120 |
| **AppartementZE3** | 90 |
| **StudioED1** | 55 |

**Table 3.2** Données numériques de l'attribut "Surface"

| | **Distance au centre (km)** |
|---|---|
| **Villat1Z** | 20 |
| **Maison22E** | 12 |
| **AppartementZE3** | 0 |
| **StudioED1** | 10 |

**Table 3.3** Données numériques de l'attribut " distance au centre "

Une échelle conceptuelle possible pour l'attribut "Surface" est donnée dans la table 3.4. Les attributs binaires de l'échelle dans cet exemple sont "280", "120", "90" et "55". La figure 3.3 présente son diagramme de Hasse.

| | **Surface<=60** | **60>Surface<=120** | **120>Surface<=180** | **Surface >180** |
|---|---|---|---|---|
| **Villat1Z** | | | | X |
| **Maison22E** | | | X | |
| **AppartementZE3** | | X | | |
| **StudioED1** | X | | | |

**Table 3.4** Echelle conceptuelle de l'attribut "Surface"

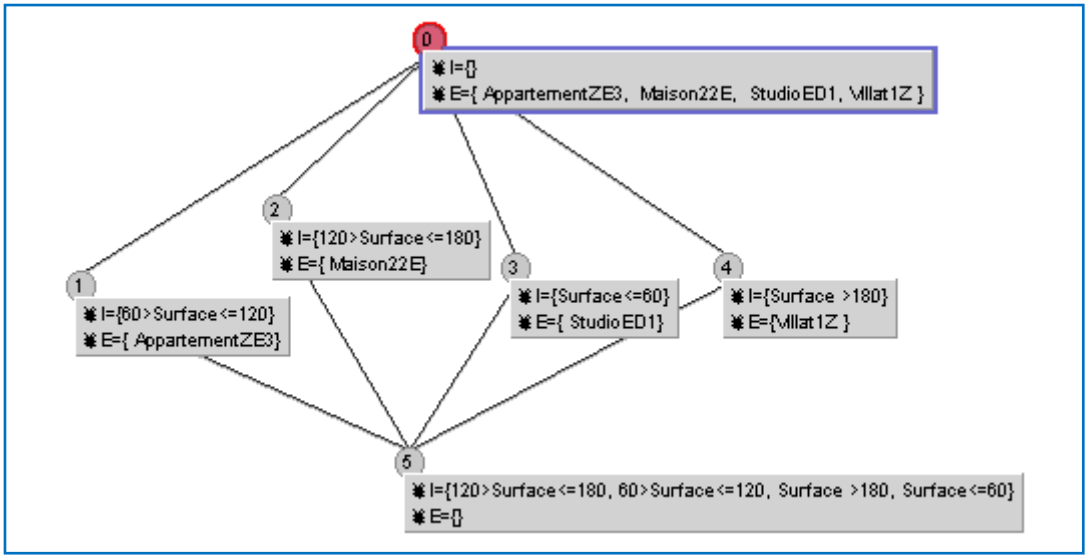



**Figure 3.3** Digramme de Hasse du tableau 3.4

De la même manière on peut définir l'échelle pour l'attribut multivalué "Distance au centre". L'échelle conceptuelle qui correspond à l'attribut est présentée dans la table 3.5 et son diagramme de Hasse dans la figure 3.4.

| | **Distance au centre <=10** | **10> Distance au centre <=20** | **20> Distance au centre <=30** |
|---|---|---|---|
| **Villat1Z** | | X | |
| **Maison22E** | | X | |
| **AppartementZE3** | X | | |
| **StudioED1** | X | | |

**Table 3.5** Echelle conceptuelle de l'attribut " Distance au centre "

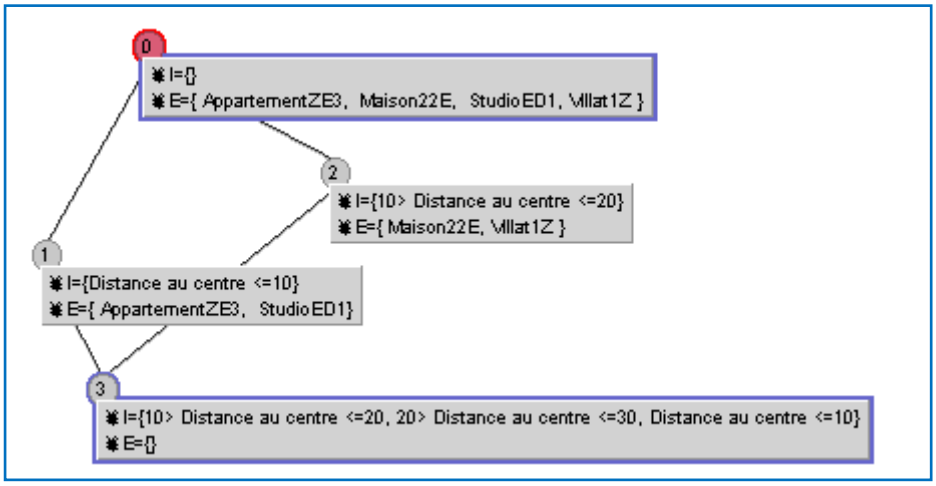



**Figure 3.4** Digramme de Hasse du tableau 3.5

Le contexte monovalué obtenu à partir du contexte multivalué des accommodations est donné dans le tableau suivant. L'ensemble des objets est le même et l'ensemble des attributs

est l'union disjointe des attributs des échelles conceptuelles. La figure 3.5 présente le contexte formelle correspondant.

| Contexte Accommodation | Surface <=60 | 60>Surface <=120 | 120>Surface <=180 | Surface >180 | Distance au centre <=10 | 10> Distance au centre <=20 | 20> Distance au centre <=30 |
|---|---|---|---|---|---|---|---|
| **Villat1Z** | | | | X | | X | |
| **Maison22E** | | | X | | | X | |
| **AppartementZE3** | | X | | | X | | |
| **StudioED1** | X | | | | X | | |

**Tableau 3.6** Contexte binaire résultant de l'échelonnage conceptuel du contexte multivalué des accommodations.

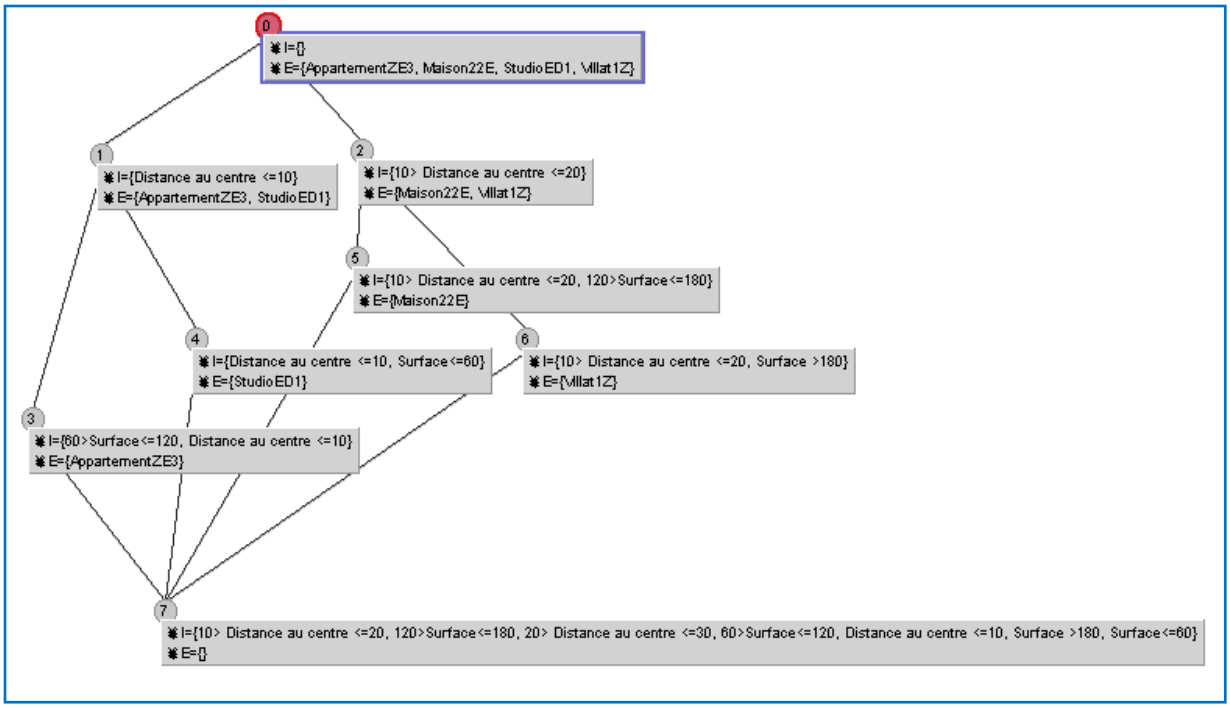



**Figure 3.5** Digramme de Hasse du tableau 3.6

L'ARC injecte les relations comme des nouveaux attributs aux objets grâce à un processus d'échelonnage relationnel. On distingue deux types d'échelonnage relationnel : un échelonnage existentiel (∃) qui consiste à considérer qu'un objet du contexte est en relation avec un autre concept s'il existe au moins un objet de son extension en relation avec lui. L'échelonnage universel (∀) consiste à considérer qu'un objet est en relation avec un autre concept si tous les objets avec lesquels il est en relation sont dans son extension. Dans notre cas, nous ne nous intéressons qu'à l'échelonnage existentiel en raison de type des relations entre les concepts ontologiques.

**Définition 2.9 (Opérateur d'échelonnage relationnel existentiel).** Etant donné $\mathcal{K} = (O, A, I), r \in rel(\mathcal{K})$, et soit $L_i$ un treillis correspondant au $\mathcal{K}_i = (O_i, A_i, I_i)$. L'opérateur d'échelonnage relationnel existentiel noté $S_{(r,\exists),L_i}$ transforme $\mathcal{K}$ au contexte dérivé $\mathcal{K}^+ = (O^+, A^+, I^+)$ où :

- $O^+ = O$
- $A^+ = \{\exists r: c | c \in L_i\}$ où chaque $\exists r: c$ est un attribut relationnel
- $I^+ = \{(O, \exists r: c) | o \in O, c \in L_i, r(o) \cap extension\ (c) \neq \phi\}$

La définition de Rouane-Hacene et al. (Rouane-Hacene, Huchard, Napoli, & Valtchev, 2013) consiste à étendre le contexte domaine d'une relation r par de nouveaux attributs relationnels, notés ∃ r: c où ∃ est un opérateur existentiel, r est la relation et c est un concept du contexte d'un ensemble d'objets O.

Le processus d'analyse relationnelle est itératif où les contextes sont modifiés et par conséquent les treillis correspondant à l'échelonnage relationnel. Il commence par la construction des contextes initiaux. Ensuite, les treillis initiaux sont modifiés dont l'effet est d'agrandir tous ces contextes au moins d'une nouvelle relation (enrichissement). Ce processus itératif s'arrête quand un point est fixe atteint. Les nouvelles relations injectées par l'échelonnage comme nouveaux attributs ont de fortes chances de générer des nouveaux concepts. Reprenons l'exemple du contexte Personne (Tableau 3.7) où les objets sont des personne et les attributs sont " aUnDiplôme ", " Maitriser#Lang ", "Maitriser#Langs", "A+Diplômes" et " AimerVoyager ".

| **ContextePersonne** | **aUnDiplôme** | **Maitriser#Langs** | **A+Diplômes** | **AimerVoyager** |
|---|---|---|---|---|
| **Mahamed** | X | | X | |
| **Mouloud** | X | X | | X |
| **Mounir** | | X | | X |
| **Mohcin** | X | | | X |

**Tableau 3.7** Contexte formelle Personne.

| **aChoisir** | **Villat1Z** | **Maison22E** | **AppartementZE3** | **StudioED1** |
|---|---|---|---|---|
| **Mahamed** | X | | | |
| **Mouloud** | | X | | |
| **Mounir** | | | | X |
| **Mohcin** | X | | X | |

**Tableau 3.8** La relation aChoisir entre le contexte Personne et Accommodation.

| ContextePersonne | aUnDiplôme | Maitriser#Langs | A+Diplômes | AimerVoyager | aChoisir : C0 | aChoisir : C1 | aChoisir : C2 | aChoisir : C4 | aChoisir :C5 | aChoisir : C6 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Mahamed** | X | | X | | X | | X | | | X |
| **Mouloud** | X | X | | X | X | | X | | X | |
| **Mounir** | | X | | X | X | X | | X | | |
| **Mohcin** | X | | | X | X | X | X | | | X |

**Tableau 3.9** Cotexte formelle Personne après l'application d'ARC

L'application du processus itératif de l'analyse relationnelle de concepts sur le contexte formel Personne est présenté dans tableau 3 .9. Le diagramme de Hasse du Contexte Personne avec la relation  a Choisir est présenté dans la figure 3.6.

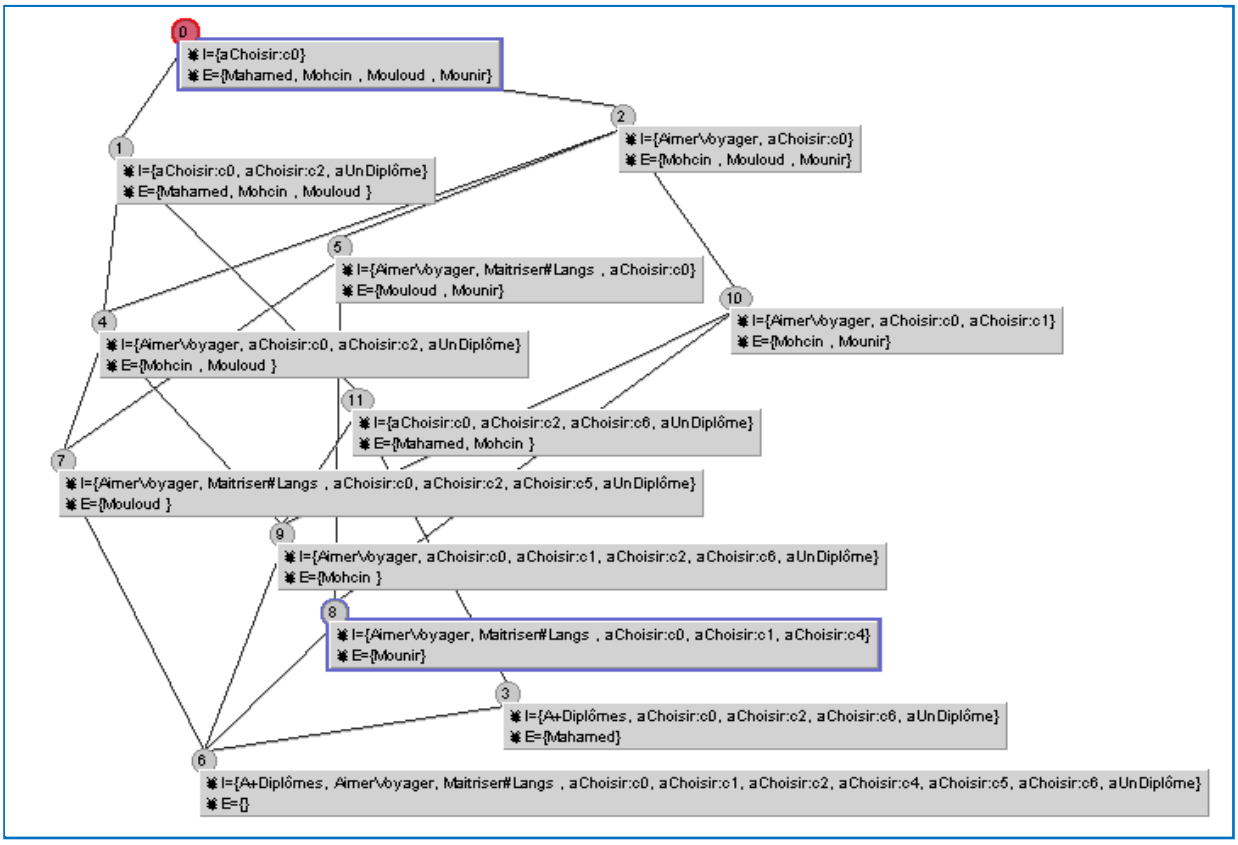



**Figure 3.6** Le diagramme de Hasse du Contexte Personne avec la relation  aChoisir

## 4 Structures de patrons

Nous rappelons ici quelques définitions générales concernant la théorie des treillis sur lesquelles s'appuient les structures de patrons (extension de l'analyse formels de concepts).

**Définition 3.10 (Infimum).** Etant donné un ensemble partiellement ordonné $(O, \leq)$ c-à-d ses éléments ne sont pas tous comparables deux à deux, $A$ un sous ensemble de $O$, et $i$ un élément quelconque de $O$. $i$ est un infimum ou borne inférieure de $A$ si et seulement si :

- $\forall\, a \in A, (i \leq a)$
- et $\forall\, c \in O, (\forall\, a \in A, c \leq a) \rightarrow (c \leq i)$

**Définition 3.11 (Inf-demi-treillis).** Un inf-demi-treillis est un ensemble muni d'une relation d'ordre tel que chaque couple $(x, y)$ d'élément admet un infimum de $x$ et de $y$.

**Définition 3.12 (Propriété d'idempotence).** Soit $O$ un ensemble muni d'une propriété d'une loi de composition interne $\Delta$. $\Delta$ est idempotence si et seulement si :

- $\forall\, x \in O, x \,\Delta\, x = x$

**Définition 3.13 (Propriété de commutativité).** Soit $O$ un ensemble muni d'une propriété d'une loi de composition interne $\Delta$. $\Delta$ est commutative si et seulement si :

- $\forall\, x, y \in O, x \,\Delta\, y = y \,\Delta\, x$

**Définition 3.14 (Propriété d'associativité).** Soit $O$ un ensemble muni d'une propriété d'une loi de composition interne $\Delta$. $\Delta$ est associative si et seulement si :

- $\forall\, x, y, z \in O, (x \,\Delta\, y) \,\Delta\, z = x \,\Delta\, (y \,\Delta\, z)$

Les structures de patrons sont introduites dans (Ganter & Kuznetsov, 2001) comme une généralisation des contextes formels. Une telle structure se formalise par un triplet $(G, (D,\sqcap), \delta)$ où $G$ est un ensemble d'objets, $(D,\sqcap)$ est un inf-demi-treillis d'éléments appelés patrons et $\delta$ est une fonction qui associe à tout objet $g \in G$ sa description $\delta(\mathrm{g})$ dans $D$. $(D,\sqcap)$ est un ensemble partiellement ordonné de descriptions d'objets. L'infimum $\sqcap$ est un opérateur idempotent, commutatif et associatif qui induit une relation d'ordre ou relation de subsomption ($\sqsubseteq$) où les patrons sont ordonnés par $\sqsubseteq\ d \iff c \sqcap d = c\ , \forall\, c, d \in D$. Les deux opérateurs de dérivation suivants forment une connexion de Galois. Soient $A \subseteq G$ *et* $d \in (D,\sqcap)$ :

$$A^{\square} = \prod_{g \in G} \delta(\mathrm{g})$$

$$d^{\square} = \{g \in G \mid d \sqsubseteq \delta(\mathrm{g})\}$$

Les concepts obtenus sont des couples $(A, d)$ avec $A \in G$ et $d \in (D,\sqcap)$, tels que $A^{\square} = d$ et $A = d^{\square}$ et sont ordonnés par $(A_1, d_1) \leq (A_2, d_2) \iff A_1 \subseteq A_2$ ($\iff d_2 \sqsubseteq d_1$) pour former un treillis de concepts.

Les structures de patrons permettent à l'AFC de considérer directement les données complexes. Pour cela, un infimum sur les descriptions des objets est défini et induit un ordre partiel sur des descriptions.

Kaytoue M. et al. (Kaytoue, 2011) ont utilisé des structures de patrons pour construire un treillis à partir de données numériques ou intervalles. Considérons les données du tableau suivant qui peuvent être formalisées par une structure de patron ( $G, (D,\sqcap), \delta$ ) où $G$={ $g_1, \ldots, g_5$ } et $D$ est un ensemble de vecteur d'intervalles à trois dimensions.

| | m1 | m2 | m3 |
|---|---|---|---|
| **g1** | 5 | 7 | 6 |
| **g2** | 6 | 8 | 4 |
| **g3** | 4 | 8 | 5 |
| **g4** | 4 | 9 | 8 |
| **g5** | 5 | 8 | 5 |

**Tableau 3.10** Un exemple de données numériques

Chaque composante correspond à un attribut ou colonne de la table. Par exemple, la description de l'objet $g_1$ est $\delta(g_1)$ = < [5; 5]; [7; 7]; [6; 6] >. Quand $A \subseteq G$ est un ensemble d'objets et $d \in (D,\sqcap)$ est un vecteur d'intervalles, $A^{\square}$ un vecteur d'intervalles composé pour chaque dimension du plus petit intervalle contenant tous les intervalles de la dimension correspondante des descriptions des objets de $A$. De manière duale, $d^{\square}$ retourne l'ensemble des objets décrits pour chaque dimension par un intervalle inclus dans l'intervalle correspondant de $d$.

Exemple. A partir de l'ensemble d'objets $\{g_1, g_2\} \subseteq G$et de la structure de patrons définie à partir du tableau 3.10, calculons

$$\{g_1, g_2\}^{\square} = \prod_{g_1 \in \{g_1, g_2\}} \delta(g) = \delta(g_1) \sqcap \delta(g_2)$$

= <[5; 5]; [7; 7]; [6; 6]> ⊓ <[6; 6]; [8; 8]; [4; 4]>

= <[5; 5] ⊓ [6; 6]; [7; 7] ⊓ [8; 8]; [6; 6] ⊓ [4; 4]>

= <[5; 6]; [7; 8]; [4; 6]>

$< [5;\ 6];\ [7;\ 8];\ [4;\ 6] >^{\square}$ = $\{g \in G \mid$ <[5; 6]; [7; 8]; [4; 6]> $\sqsubseteq\ \delta(g)\}$

= $\{g_1, g_2, g_5\}$

$g_1$ et $g_2$ appartiennent déjà à $< [5;\ 6];\ [7;\ 8];\ [4;\ 6] >^{\square}$ . $g_5$ appartient aussi à cet ensemble puisque sa description contient aussi des intervalles inclus dans l'intervalle correspondant de <[5; 6]; [7; 8]; [4; 6]> : nous avons donc <[5; 6]; [7; 8]; [4; 6]> $\sqsubseteq\ \delta(g_5)$. Après l'application de deux opérateurs de la connexion de Galois sur l'ensemble $\{g_1, g_2\}$, le résultat obtenu est le couple $(A, d)$ = $(\{g_1, g_2, g_5\}, < [5;\ 6];\ [7;\ 8];\ [4;\ 6] >)$ qui présente un concept formel, car $A^{\square} = d$ et $A = d^{\square}$. Le treillis de concepts pour l'exemple est donné à la figure suivante (avec conExp) et illustre l'ordre des concepts.

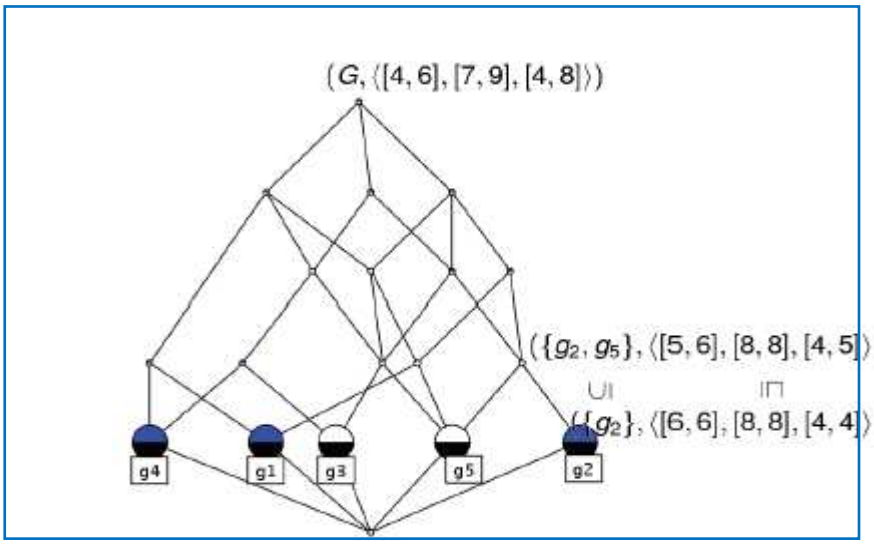



**Figure 3.7** Le treillis de concepts pour l'exemple des données numériques (Kaytoue, 2011)

Nous utilisons les structures de patrons comme un cadre formel pour valider le patron de conception d'ontologie proposé en simulant la création des points de vue d'utilisateurs comme des patrons de l'analyse formelle.

## 5 Analyse formelle de concepts et l'ontologie

L'analyse formelle de concepts est largement utilisée en ingénierie de l'ontologie. Dans ce qui suit, nous présentons quelques travaux jugés les plus représentatifs.

### 5.1 Construction des ontologies

Les ontologies jouent un rôle primordial dans le web sémantique où elles facilitent la communication entre les applications hétérogènes. Ces ontologies utilisent généralement OWL comme un langage d'ontologie. Alors que l'ontologie utilise souvent une représentation hiérarchique pour la modélisation d'un domaine, l'AFC utilise aussi la représentation hiérarchique d'ordre partiel mais avec une grande expressivité. Pour cela, plusieurs méthodes formelles de construction d'ontologies basées sur l'AFC ont été proposées.

- Guoqian Jiang et al. (Jiang, Ogasawara, Endoh, & Sakurai, 2003) ont proposé une méthode de construction d'ontologie de domaine clinique basée sur l'AFC et traitement de langage naturel (PNL). La méthode est réalisée par un plugin dans Protégé-2000. Protégé est un environnement d'édition d'ontologie développé par Stanford Medical Informatics (Jiang, Ogasawara, Nishimoto, Endoh, & Sakurai, 2005). L'AFC a été utilisée pour prendre les termes (entrée) du dictionnaire médical afin d'identifier l'ensemble des objets du domaine clinique et leurs attributs et construire le contexte formel. Enfin l'ontologie clinique est générée à partir de ce contexte formel.
- Haav (Haav, 2004) a présenté une nouvelle méthode semi-automatique qui combine un langage basé sur des règles et l'AFC pour construire une ontologie de domaine. À l'aide de l'AFC, l'ontologie initiale peut être construite comme un treillis de concepts à

partir du contexte formel. Ensuite elle est représentée comme un ensemble des règles en logique de premier ordre et visualisée dans un concepteur d'ontologie. Le concepteur d'ontologie peut aussi étendre l'ontologie par de nouveaux concepts et relations en utilisant les clauses de Horn.

- Suqin Tang et al. (Tang & Cai, 2010) ont proposé une nouvelle méthode appelée méthode de construction d'ontologie de tourisme (Tourism Ontology Construction Method noté (TOCM)) en utilisant AFC. Le TOCM comprend un module de prétraitement d'information touristique, le module AFC et le module de construction d'ontologie. .

### 5.2 Restructuration des ontologies

La restructuration d'une ontologie est une tâche difficile. En effet le repositionnement d'un l'élément ontologique peut affecter ses voisins. En réingénierie orientée objet, la restructuration à grande échelle est exécutée avec succès en utilisant l'analyse formelle de concepts (Rouane-Hacene, Fennouh, Nkambou, & Valtchev, 2010).

- Raoune et al (Rouane-Hacene, Fennouh, Nkambou, & Valtchev, 2010) ont appliqué l'analyse relationnelle de concepts comme un cadre pour la restructuration des ontologies. L'approche de restructuration proposée comprend quatre étapes: l'alignement, l'encodage, l'analyse et le reverse engineering (encodage). L'étape d'alignement détermine les correspondances entre les éléments d'ontologie d'entrée nommé ontologie de modèle (OntologyModel) OM pour éviter la redondance de codage des éléments similaires. Pendant la phase d'encodage, les éléments d'ontologie initiale OM sont transformés en une famille de contextes relationnels FCR où chaque contexte correspond à un type d'un élément dans le méta modèle d'OM, c.-à-d. des classes et des restrictions. De plus, les incidences entre les méta-éléments (liens entre les éléments dans le méta-modèle de l'ontologie) sont traduites en relations dans le FCR. Ensuite, la famille des contextes formels est transformée en un ensemble des treillis par un processus ARC. Finalement, une ontologie est générée à partir de ces treillis en filtrant potentiellement les concepts utiles.

### 5.3 Enrichissement et maintenance des ontologies

Il existe un nombre croissant d'applications qui utilisent les ontologies disponibles sur le web pour économiser l'effort de construction d'une nouvelle ontologie à partir de zéro. Cependant, souvent, les normes standard disponibles doivent être étendues pour inclure certaines connaissances spécifiques du domaine. En outre, les ontologies doivent être maintenues pour refléter les changements de la partie du monde qu'elle décrit, en particulier, lorsque l'ontologie est construite à partir d'un ensemble de données où toute modification de ces données nécessite des mises à jour des ontologies existantes.

- Dominic Looser et al. (Looser, Ma, & Schewe, 2013) présentent une approche pour enrichir et adapter une ontologie existante en utilisant une analyse formelle des concepts. L'objectif de leur recherche est de fournir une méthode pour réviser et

maintenir les ontologies existantes pour les systèmes de recrutement afin que la qualité des ontologies soit assurée et que l'appariement sémantique puisse être amélioré. Les ontologies de ce domaine doivent être enrichies adéquatement avec les connaissances détaillées spécifiques et adaptées aux besoins particuliers du marché du travail. L'objectif de cette approche est de fournir un cadre pour la mise à jour, l'enrichissement, la maintenance et la révision automatique des hiérarchies des ontologies existantes. L'approche prend de nouvelles connaissances à partir des experts de domaine et utilise FCA pour insérer de nouvelles relations et de nouveaux concepts DL dans l'ontologie existante (voir la figure 3.8).

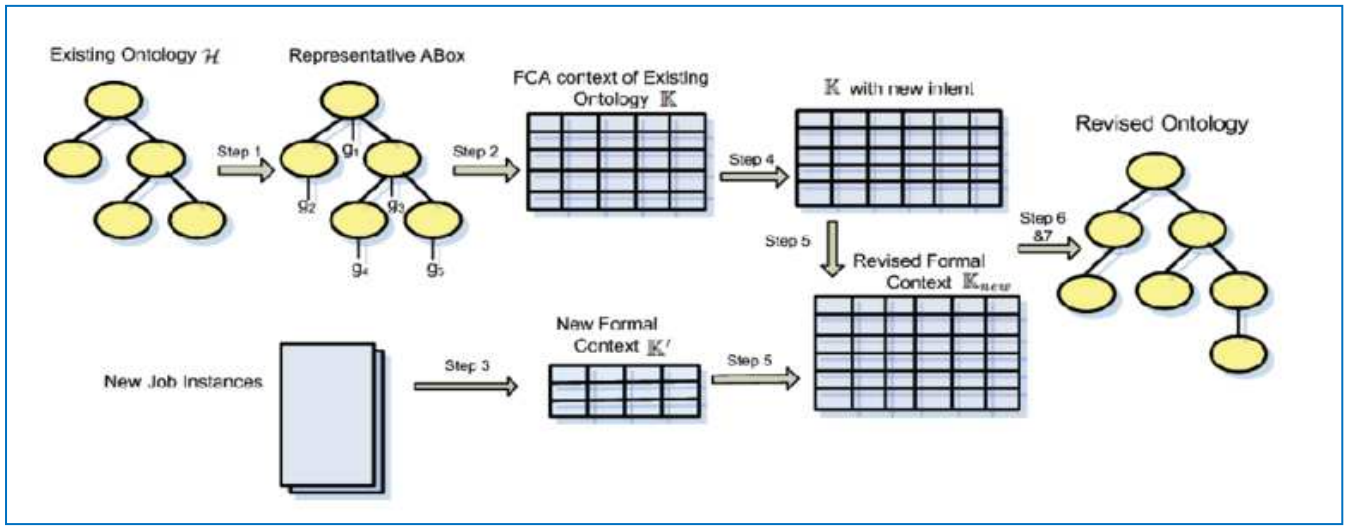



**Figure 3.8** Approche de restructuration par AFC pour l'enrichissement d'ontologie (Looser, Ma, & Schewe, 2013)

- Bendaoud et al (Bendaoud, Toussaint, & Napoli, 2010)ont proposé un système (Pactole) d'enrichissement d'ontologies à partir de textes basé sur l'analyse de concepts et l'analyse relationnelle de concepts. PACTOLE construit semi-automatiquement une ontologie à partir d'une hiérarchie de classes construites manuellement par les experts du domaine, puis l'enrichit avec des définitions extraites de toutes les ressources du domaine spécifique. PACTOLE (voir la figure 3.9) s'appuie sur trois grandes étapes. La première est l'étape de prétraitement des ressources (thésaurus, bases de données, corpus de textes, ...) pour extraire les objets du domaine. La deuxième étape est l'application des méthodes AFC/ARC. Le «schéma d'ontologie» résultant est évalué et validé par les experts du domaine. La troisième étape consiste à transformer le schéma d'ontologie à un ensemble d'expressions en Logique de Description (LD). Ces expressions sont codées par la suite en une ontologie OWL.

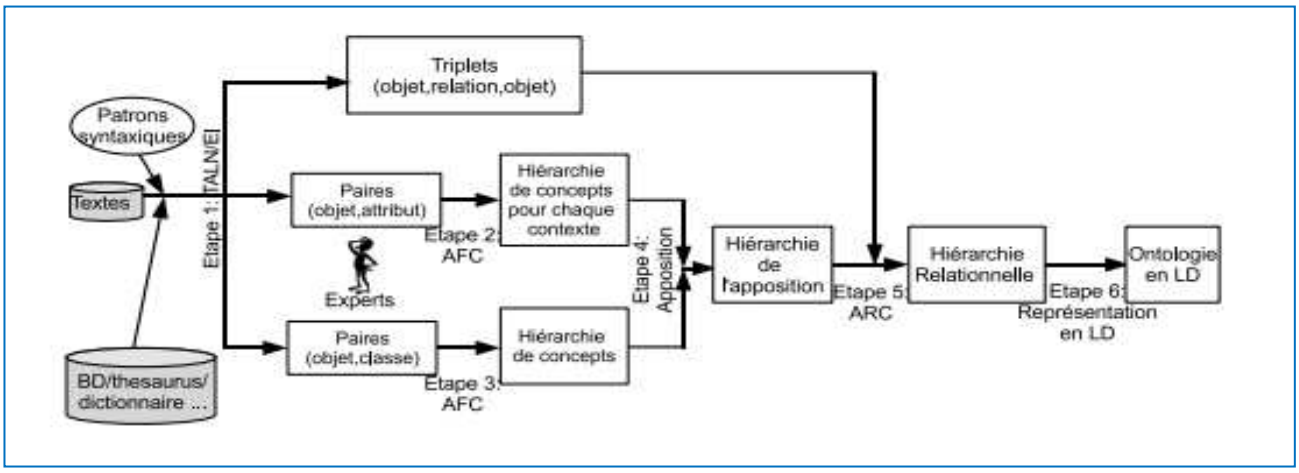



**Figure 3.9** Le système PACTOLE (Bendaoud, Toussaint, & Napoli, 2010)

### 5.4 Fusionnement des ontologies

La fusion des ontologies est un processus de génération d'une seule ontologie en intégrant des ontologies existantes. L'alignement d'ontologie est un processus impliqué dans l'intégration afin d'établir les liens de correspondance entre les ontologies.

- Gerd Stumme et al. (Stumme et al., 2003) ont proposé une méthode ascendante de fusion pour fusionner deux ontologies de même domaine en utilisant l'AFC. La méthode de fusion intègre les ontologies en trois étapes. La première étape consiste à extraire les instances à partir de deux ontologies d'entrée O1 et O2 basée sur un processus de traitement du langage naturel et deux contextes formels K1 et K2 respectivement. Ensuite, l'algorithme de FCA-MERGE (Stumme et al., 2003) prend les deux contextes formels K1 et K2 comme entrée et génère le contexte commun K de K1 et K2 et construit le treillis élagué pour le contexte K. Enfin, l'ontologie fusionnée peut être généré à partir du treillis élagué.
- Li Guan-Yu et al. (Guan-Yu, Shu-Peng, & Yan, 2011) ont présenté une méthode nommée FCA-OntMerge pour intégrer deux Ontologies basées sur l'AFC. La méthode FCA-OntMerge comporte quatre étapes pour fusionner les ontologies. La première étape consiste à unifier le format des ontologies en entrée en utilisant le format OWL dans Protege. Dans la deuxième étape, les ontologies d'entrée sont traitées par Jena pour extraire les concepts et les attributs afin de construire le contexte formel. Dans la troisième étape, un tableau de mapping est généré par l'alignement des attributs des contextes formels. La dernière étape de cette méthode consiste à intégrer les deux contextes formels, puis à construire le treillis pour le contexte fusionné et générer enfin une nouvelle ontologie.
- Olivier Cure (Cure, 2010) a proposé un algorithme pour fusionner des ontologies expressives spatiales à l'aide de l'AFC. L'entrée de l'algorithme est deux ontologies spatiales et la sortie est une ontologie spatiale fusionnée.

## 5.5 Visualisation des ontologies

Les ontologies, en tant qu'un ensemble de concepts et de propriétés d'un domaine spécifique sont très utiles dans les domaines des bibliothèques numériques, le web sémantique et la gestion d'information personnalisée. En conséquence, il existe un besoin croissant d'un outil de visualisation efficace des ontologies pour la conception, la gestion et la navigation. La visualisation des ontologies n'est pas une tâche facile. En réalité, une ontologie n'est pas une simple hiérarchie de concepts mais elle est enrichie par des rôles des relations qui lient leurs différents concepts. Chaque concept a des propriétés et probablement des instances ou individus qui peuvent varier d'un et deux à milliers. Par conséquent, il n'est pas facile de créer une visualisation qui affiche efficacement toutes ces informations et, en même temps, permet aux utilisateurs d'effectuer facilement des opérations diverses sur les ontologies.

- L'étude de Guoqian Jiang et al. (Jiang, Ogasawara, Nishimoto, Endoh, & Sakurai, 2005) se base sur l'analyse formelle de concepts (AFC) qui est un moyen idéal avec sa structure générique des algorithmes de construction des treillis qui ont un ordre partiel. Les auteurs ont développé un outil de visualisation et de modélisation pour les expressions composites de SNOMED-CT en format OWL en utilisant la technique d'AFC. Les concepts de SNOMED-CT se divisent en deux types: concepts primitifs et concepts entièrement définis. L'outil de visualisation et de modélisation a été développé sous la forme d'un plug-in Protégé appelé "OWLFCAView Tab" dans une plate-forme Protégé OWL. L'API JAVA 'Concept Explorer' version 1.2 (un logiciel open source) a été intégrée pour générer le tableau de contexte (figure 3.10) et le treillis de concepts

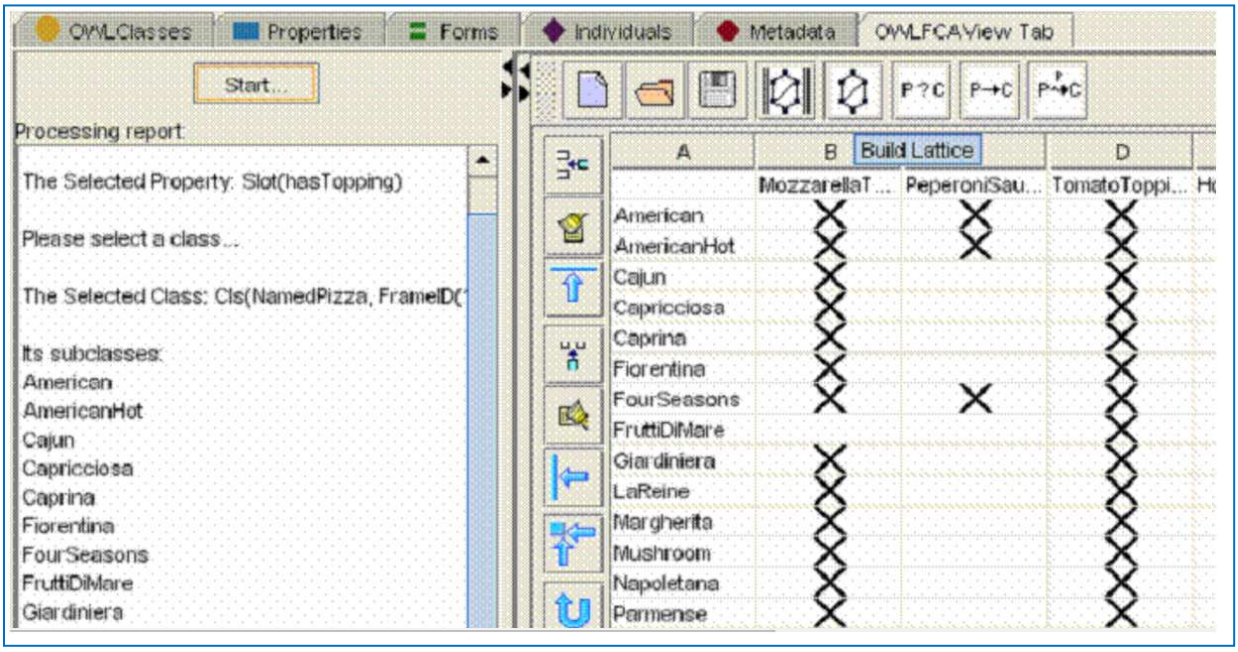



**Figure 3.10** OWLFCAView Protege Tab Plug-in with Pizza ontology: tableau de concepts(Jiang, Ogasawara, Nishimoto, Endoh, & Sakurai, 2005)

# 6 Conclusion

Dans ce chapitre, nous avons conduit une revue de la littérature pour, d'abord identifier le cadre dans lequel s'inscrit notre travail. Nous avons présenté les principes de l'analyse formelle de concepts et ses applications en ingénierie ontologique. Nous avons présenté sa représentation tabulaire et sa visualisation par le digramme de Hasse. Nous avons aussi présenté les algorithmes et les outils de construction de treillis. Deux extensions de cette méthode formelle ont été présentées : l'analyse relationnelle de concepts et les structures de patrons. Nous nous sommes concentrés sur ces deux extensions qui seront utilisées par la suite dans nos propositions.

l'AFC est considérée comme un formalisme mathématique rigoureux. Elle l'adapte afin de permettre d'analyser des données sous forme d'un contexte formel (objets x attributs) et de les structurer en un treillis de concepts facile à visualiser et interpréter. Toute discipline dont les données peuvent être codées sous forme d'ensemble (objets x attributs) peut bénéficier des avantages de l'AFC. Par ses capacités de conceptualisation, l'AFC apporte une solution formelle pour regrouper des données brutes en concepts selon des critères de similarité bien établis (partagent les mêmes attributs) et de les classifier avec une factorisation maximale (sans redondance).

Comme en Génie Logiciel, l'AFC a été utilisée avec succès en ingénierie ontologique. L'utilisation de l'AFC est motivée par :

- L'analogie entre objet/attribut/concept de treillis et individu/propriété /classe (concept) de l'ontologie
- L'analogie entre l'ordre partiel entre les concepts de treillis et la relation de subsomption de l'ontologie.
- L'AFC permet la classification des concepts dans un treillis. Ce dernier facilite le parcours et la lecture des concepts.

Dans le chapitre 4, nous présentons notre patron de conception d'ontologie multipoints de vue (MV2P) en détail. Nous utilisons l'extension de l'analyse formelle de concepts « structures des patrons » pour le valider. L'AFC classique n'est applicable que sur des objets décrits par des attributs binaires. Ainsi il faut transformer les objets du domaine, en tableaux binaires capables pour être traités par l'AFC. Cependant les objets multipoints de vue sont des vues complexes (attachés à plusieurs vues). Pour cela nous utilisons les « structures de patron ». Et comme Raoune et al (Rouane-Hacene, Fennouh, Nkambou, & Valtchev, 2010), nous utilisons l'analyse relationnelle dans un processus d'intégration de concepts afin de restructurer les données ontologiques des sources locales selon le patron multipoints de vue (dans le chapitre 5).

Chapitre 4

# Patron de Conception d'Ontologie Multipoints de Vue

## Sommaire



## 1 Introduction

La bonne conceptualisation, modularisation, compréhension, lisibilité des ontologies et la facilité de les réutiliser et d'assurer leur évolution, sont des objectifs très recherchés en ingénierie ontologique. Pour approcher de tels objectifs et favoriser une ingénierie ontologique de bonne qualité, les patrons de conceptions d'ontologie ont été adoptés ces dernières années. Les patrons de conception d'ontologie constituent un moyen efficace pour la conception des ontologies réutilisables dans des domaines complexes. Ils améliorent leur qualité, leur compréhension et facilitent leur évolution (Presutti, Daga, Gangemi, & Blomqvist, 2009) (Rodriguez-Castro, 2012) (Shimizu, Hitzler, Paul, 2018) (Alirezaie, Hammar, & Blomqvist, 2018). L'idée de notre travail de recherche est de proposer des patrons de conception d'ontologie pour intégrer la notion de point de vue. Cette dernière n'est pas nouvelle (*cf.* chapitre 1). Elle a été déjà définie dans les bases de données, dans la représentation des connaissances, dans les ontologies et plusieurs approches par point de vue ont été proposées. L'intégration de la notion de point de vue dans les systèmes informatiques est nécessaire pour l'acquisition et la représentation des connaissances dans un cadre de multi-expertises, multi-métiers ou multidisciplinaires.

Comme présentés dans le chapitre 2, les patrons de conception d'ontologie ODPs (Ontology Design Patterns) sont conçus comme des guidelines partagées et réutilisables. Ils sont classés en six différents types. En particulier, les « ODPs de contenu » sont des patrons conceptuels réutilisables visant à modéliser des ontologies valides par rapport à un domaine donné. Les « ODPs d'architecture » concernent la forme globale de l'ontologie et leur instanciation dans un domaine donné (tourisme, biologie, etc.) fournit des « ODPs de contenu ». Les « ODPs de correspondance » permettent de créer des associations sémantiques entre deux modélisations de domaines similaires.

Le but de ce chapitre est triple. Nous proposons d'abord un patron de conception d'architecture qui intègre la notion de point de vue comme une connaissance dans l'ontologie. La deuxième contribution consiste à proposer un Framework de validation du patron proposé basé sur l'extension de l'AFC (structures de patrons). Dans la troisième contribution, nous présentons un ensemble de patrons de conception de correspondance pour lier les différentes vues dans une ontologie. Ces patrons facilitent la modélisation des passerelles ou ponts entre les vues d'un même objet réel.

Dans ce chapitre, nous offrons une description détaillée comment les patrons sont créés et la façon de les utiliser. La section 2 motive l'intégration de la notion de point de vue dans les ontologies en utilisant les patrons de conception d'ontologies. Nous identifions et discutons, dans la section 3, notre méthodologie d'intégration de cette notion en précisant les différentes exigences et les différents éléments d'un modèle multipoints de vue. Dans la section 4, nous présentons nos patrons d'architecture multipoints de vue et de correspondance. Pour valider le patron multipoints de vue, nous utilisons l'extension de l'analyse formelle de concepts, 'les structures des patrons', dans la section 5. Dans la section 6, nous illustrons notre validation en appliquant le patron sur un cas d'étude. Dans la section 7 nous présentons le cycle de vie d'un patron de conception d'ontologie. La section 8 conclut ce chapitre

## 2 Motivation

A l'opposition de la vision monolithique des approches traditionnelles de représentation de connaissances où le monde est unique et les observateurs le perçoivent tous de la même façon, l'approche multi point de vue permet de modéliser la même réalité selon des points de vues différents. Dans le domaine de la représentation des connaissances, les points de vue sont utilisés comme un moyen de structuration d'une base de connaissances selon différents points de vue (Marino, 1993) (Marcaillou, 1995) (Benchikha & Boufaida, 2007). Dans le cadre du Web sémantique où l'ontologie est utilisée pour représenter des connaissances d'un domaine de discours, la notion de point de vue est aussi intégrée (Falquet & Mottaz, 2002) (Bach, 2006) (Hemam, 2012). L'idée consiste à structurer une ontologie en modules représentant chacun un point de vue. Comme en représentation des connaissances, les points de vue dans l'ontologie peuvent être des hiérarchies sur un ensemble d'objets (sous ontologie). Il peut donc y avoir plusieurs hiérarchies sur les mêmes objets. Chaque hiérarchie présente un point de vue particulier. Ces points de vue peuvent ne pas être indépendants mais liés par des liens de dépendances ou des passerelles. Les objectifs visés par l'intégration des points de vue dans la représentation d'ontologie sont multiples :

- Permettre la multi-représentation des objets : La représentation des objets réels consiste à faire une abstraction de leur réalité. Le résultat naturel de cette représentation est multiple. Celle-ci provient de la multiplicité des regards portés sur ces objets.

- Apporter la modularité des ontologies : la modularisation est la décomposition d'une ontologie monolithique en composants plus petits et interdépendants appelés modules (Abbes, Scheuerman, Meilender, & D'aquin, 2012). Elle est considérée comme un moyen de structurer et d'organiser une ontologie en des briques indépendantes, autonomes et réutilisables. Chaque brique ou module est inhérent à un point de vue.

- Maîtriser la complexité de représentation de la connaissance : la notion de point de vue permet une représentation simplifiée de la connaissance. Cela permet de diminuer la charge pour un utilisateur.

- Offrir un accès restreint aux objets : c'est-à-dire offrir la possibilité d'accès aux objets en ne s'intéressant qu'aux propriétés qui sont utiles pour un point de vue donné, et ce, à tout moment. Il devient alors très facile de manipuler les objets puisque l'on se limite toujours aux connaissances strictement nécessaires.

Les objectifs présentés ci-dessus ont motivé notre recherche pour proposer l'intégration de la notion de points de vue dans les ontologies en utilisant les patrons de conceptions d'ontologie. Ceux-ci permettent aussi la réutilisation et la modularité (*cf* chapitre 2).Dans ce qui suit, nous présentons notre méthodologie d'intégration des points de vue dans les ontologies en utilisant les patrons de conception d'ontologie.

## 3 Méthodologie

La méthodologie d'intégration de la notion de point de vue dans les ontologies en utilisant les patrons de conception d'ontologie repose sur deux critiques :

- Les ontologies qui modélisent un univers de discours ne tiennent pas en compte la diversité des visions qu'on peut avoir sur les objets du monde réel.

- La majorité des ontologies développées sont des ontologies de domaine. Elles sont créées pour une application et des exigences précises. Elles ne tiennent pas compte de leur réutilisation et évolution. Cependant, avec les nouvelles technologies et l'apparition du web des données ouvertes et liées, l'ontologie doit être facile à réutiliser, à maintenir et à évoluer.

Notre méthodologie intègre le paradigme de point de vue en utilisant les patrons de conception d'ontologie (patron d'architecture) dans la construction d'une ontologie multipoints de vue. Dans cette section, nous présentons tout d'abord les exigences qui nous

conduisent au développement du patron multipoints de vue puis nous décrivons le modèle d'ontologie multipoints de vue avant de présenter notre patron multipoints de vue.

### 3. 1 Exigences de développement

D'après Marino (Marino, 1993), « La notion de point de vue prend tout son sens à l'intérieur du concept ; la famille d'objets décrite par un concept particulier peut être regardée selon différents points de vue. Ce regard sélectif permet, d'une part de ne voir que les attributs du concept qui sont pertinents pour le point de vue en question et d'autre part, de structurer les instances du concept dans une hiérarchie de classes significative pour le point de vue ». Cette définition identifie deux caractéristiques à respecter pour supporter la modélisation de la notion de point vue : (1) Permettre un regard sélectif aux attributs des concepts et (2) créer une hiérarchie de classes significative pour le point de vue. Partant de ces deux caractéristiques et de la définition de Marcaillou (Marcaillou, 1995) concernant les vues (*cf* chapitre 1.Section 3), nous identifions les exigences suivantes guidant le développement de notre patron de conception multipoints de vue.

- **Exigence 1** : Le patron multipoints de vue doit être évolutif. Le contenu partageable (global) et le contenu partiel d'une vue spécifique sont décomposés de telle sorte qu'ils puissent varier indépendamment et sans redondance. Le découplage des contenus accroît les possibilités d'extension.
- **Exigence 2** : La solution doit permettre aux utilisateurs d'ontologie la construction des concepts multipoints de vue.
- **Exigence 3** : Le patron doit permettre aux utilisateurs de créer des points de vue qui les intéressent en cachant ou en combinant plusieurs vues (modélisation partielle d'un objet)
- **Exigence 4** : Les différentes vues de même objet réel doivent avoir le même identifiant.
- **Exigence 5** : Le patron doit supporter une recherche approfondie au niveau de chaque ensemble de concept de même vue.
- **Exigence 6** : Le patron doit permettre la préservation des liens sémantiques entre les différentes vues de même concept.
- **Exigence 7** : La création d'un concept multipoints de vue est régie par certaines contraintes afin de contrôler la coexistence entre vues.

### 3.2 Modèle d'ontologie multipoints de vue

En représentation des connaissances, la plupart des approches assimilent la notion de point de vue à celle de perspective proposée par Minsky (Minsky, 1975) en considérant un aspect plutôt cognitif que spatial du placement de l'observateur. Il s'agit d'une position conceptuelle par rapport à un objet servant à lui donner une description particulière.

L'étude des travaux antérieurs sur les points de vue (cf. chapitre 1), nous a permis de recenser des propriétés similaires entre les différents éléments des modèles proposés. Ces propriétés concernent principalement la modélisation des entités, des relations, des liens, et des instances. Le tableau 4.1 présentent ces éléments dans quatre systèmes.

| Propriétés principales | TROPES(Marino, 1993) | MVDB(Benchikha & Boufaida, 2007) | MVP(Bach, 2006) | MPV(Hemam, 201 |
|---|---|---|---|---|
| Données intrinsèques | Ensemble d'attributs clés qui sont visibles depuis tous les points de vue | Un schéma référenciel. Chacune de ses classes détient la description de base commune aux différents points de vue. | Toutes les descriptions et les définitions à propos des entités (sauf les individus) dans le modèle sont considérées valides et vraies selon le point de vue général (un consensus et une cohérence au niveau global dans le modèle) | Les Concepts globaux sont vus par l'ensemble des points de vue avec certaines propriétés communes. |
| Données spécifiques | Ensembles des attributs qui sont visibles depuis un ou plusieurs points de vue | Un schéma point de vue représente une extension descriptive des entités du référentiel selon un point de vue donné. | caractéristiques pertinentes par rapport à un point de vue. | Les concepts locaux sont vus et décrits localement selon un point de vue donné. |
| Relations | Subsomption+ relations classiques+ relation sort-de dans les points de vue et est-un pour le lien d'appartenance d'une instance aux différents points de vue | Subsomption+ relation classiques+ relation Vp-Extension pour lier un objet point de vue à son objet multipoints de vue | Subsomption+ relations classiques | Subsomption+ relations classiques +une relation globale (une relation lexicale, qui permet de relier les sous-concepts hiérarchisés différemment selon des points de vue différents) |
| Lien entre les points de vue | Passerelle exprime une relation ensembliste entre deux classes de deux points de vue différents d'un même concept. | Vp-Dependency est une relation qui définit des passerelles entre les classes des différents points de vue | passerelles sont des liens entre points de vue permettant de relier les classes décomposées différemment selon des points de vue différents | Une passerelle décrit un lien entre un concept source (ou un ensemble de concepts sources) et un concept cible de deux (ou plusieurs) points de vue différents |
| instance | la mono-instanciation au niveau d'un point de vue et la multi-instanciation au niveau du concept | les objets multipoints de vue et les objets points de vue | Un individu peut être décrit selon un ou plusieurs points de vue (la multi-représentation d'un objet réel) | Point de vue et multipoints de vue |

**Tableau 4.1** Les différents éléments des modèles de points de vue

D'après le tableau 4.1, les différents modèles partagent les mêmes principes suivants :

- Permettre la représentation multiple des objets.

- Introduire la notion de visibilité partielle aux attributs de concepts. les modèles distinguent entre les données des concepts spécifiques à chaque point de vue et les données communes, partageables par tous les points de vue.
- Permettre la construction des hiérarchies sur le même ensemble d'objets.
- Introduire la notion de passerelle qui permet d'exprimer des relations entre les concepts des différents points de vue.
- Permettre l'instanciation multiple des objets (instance multipoints de vue).

Cependant ces modèles se diffèrent par :

- Le modèle de connaissance qui supporte la notion de point de vue : Le modèle de TROPES (Marino, 1993) et MVDB (Benchikha & Boufaida, 2007) sont dédiés à l'intégration de la notion de point de vue dans la représentation de connaissance par objets. MVP (Bach, 2006) et MPV (Hemam, 2012) sont des modèles d'ontologies.
- Le mécanisme d'implémentation de ces principes : Par exemple TROPES (Marino, 1993) utilise la relation 'est-un' pour rattacher chaque instance multipoints de vue, dans chaque point de vue, à sa classe d'appartenance et MVDB (Benchikha & Boufaida, 2007) utilise la relation sémantique 'Vp-Extension' pour lier un objet point de vue à son objet multipoints. Dans les ontologies, le modèle MVP (Bach, 2006), par exemple, utilise le mécanisme de subsomption pour décrire une nouvelle classe dans un point de vue à partir d'une autre classe 'mère' de même point de vue ou d'un point de vue différent (point de vue général ou non). Le modèle MPV (Hemam, 2012) adapte le mécanisme d'estampillage pour permettre la multi représentation des concepts dans le formalisme de la logique de descriptions.

Les modèles points de vue présentés dans la littérature intègrent la notion de point de vue par un changement sémantique dans le modèle original cependant ils ne s'intéressent pas à 'comment le modèle devrait ressembler'. Pour cela, nous proposons un patron de conception d'architecture pour répondre à cette question. Le modèle d'ontologie sous-jacent partage les mêmes principes cités ci-dessous. La figure 4.1 présente une ontologie multipoints telle que nous la concevrons. En effet, les concepts ont une description globale (vue globale) et une ou plusieurs descriptions particulières (vues locales). Chacune est relative à une représentation selon un point de vue. Par exemple, c1 est décrit selon la vue locale 1 et c2 est décrit selon les deux vues locales 1 et 2. c1 et c2 partagent la même vue globale. Les vues locales peuvent être liées par des liens sémantiques pour enrichir ou assurer la cohérence des objets représentés. Un utilisateur peut à son tour définir un point de vue sur l'ontologie en ne spécifiant que les vues qui l'intéressent.

Dans la section suivante, nous présentons en détails notre patron de conception que nous complétons par un ensemble de patrons de correspondance d'ontologies modélisant des liens entre les vues.

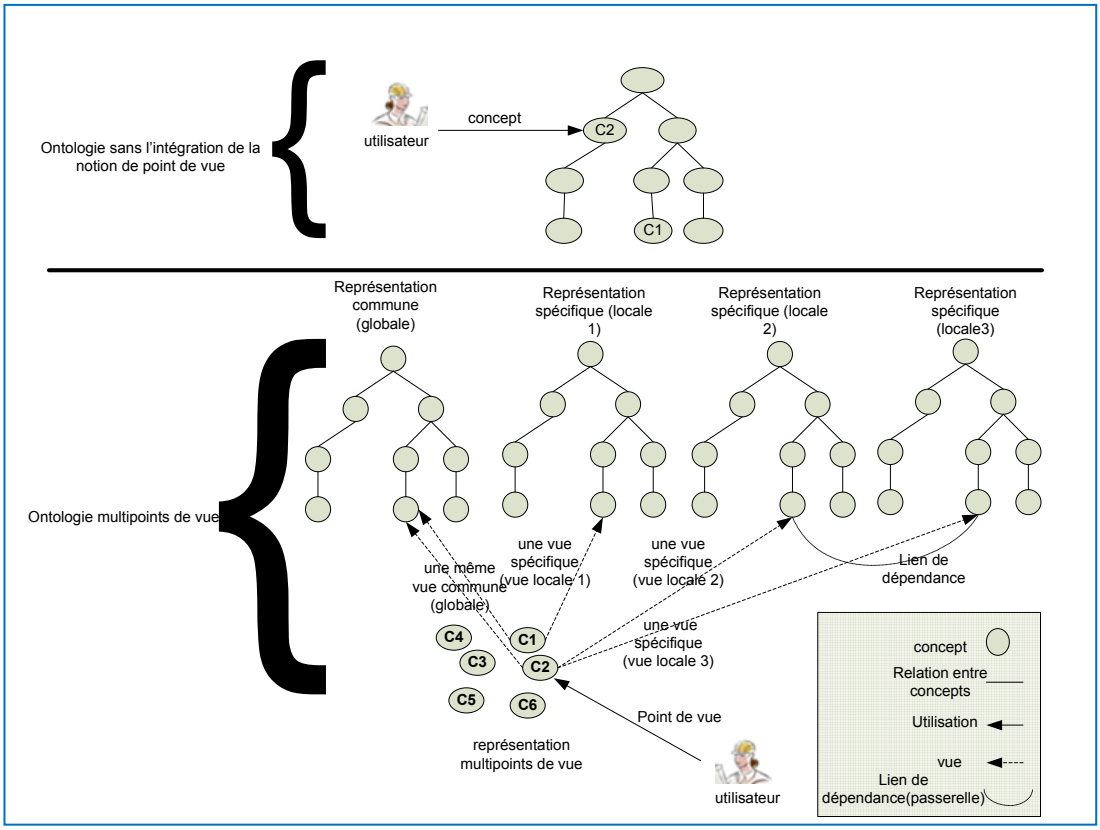



**Figure 4.1** Ontologie multipoints de vue

# 4 Patrons de conception d'ontologie multipoints de vue

Le principal avantage dans l'utilisation des patrons de conception est qu'ils diminuent l'incidence des choix pauvres de modélisation qui pourraient causer des problèmes à une date ultérieure. En outre, l'ontologie basée sur les patrons de conception est plus lisible, plus maintenable et plus réutilisable. Dans ce qui suit, nous commençons par présenter des définitions de base avant de décrire nos patrons de conception.

## 4.1 Définitions

Nous définissons l'ontologie multipoints de vue, la vue et le point de vue comme suit :

**Définition 4.1 (Ontologie multipoints de vue)** : On appelle une ontologie multipoints de vue, une ontologie qui supporte la description multiple d'un concept ontologique (concept à plusieurs vues) et permet la création d'un concept multipoints de vue en combinant plusieurs vues. Les vues de même concept sont liées par des liens nommés « passerelles » qui présentent les relations sémantiques entre les vues (contradiction, inclusion, équivalence, etc.). Dans cette ontologie la création des concepts multipoints de vue est contrôlée par des contraintes pour éviter la combinaison des vues sémantiquement contradictoires ou incompatibles.

Une ontologie multipoints de vue O est définie formellement par ($V^O$,$C_G^O$,$A_G^O$,$R_G^O$, $I_G^O$, $PV^O$,$PASS^O$, $CON^O$) où

- $V^O$ = *{V1, V2, V3…..Vn}* est un ensemble des vues locales,

- $PV^O$ est une hiérarchie des concepts d'un point de vue d'utilisateur,
- $C_G^O$ est un ensemble de concepts globaux. Un concept de cet ensemble est nommé « concept GlobalView »
- $A_G^O$ est un ensemble des attributs globaux
- $R_G^O$ est un ensemble de relations entre les concepts globaux
- $I_G^O$ est un ensemble des individus globaux,
- $PASS^O$ est l'ensemble de passerelles entre les $V^O$ et
- $CON^O$ représente l'ensemble des contraintes associées à la combinaison des vues.

**Définition 4. 2 (Vue)** : Une vue est une conceptualisation partielle du domaine. La vue d'un concept est une description partielle de ce concept dans cette vue.

Une vue V est définie formellement par ($C_L^O$,$A^O$, $R_L^O$,$I_L^O$,$NS$,≤) où

- $C_L^O$ est un ensemble de concepts locaux classifiés par un opérateur de subsomption≤. Un concept de cet ensemble est nommé « concept LocalView » ;
- $A^O$ est l'ensemble des attributs locaux
- $R_L^O$ est un ensemble de relations entre les concepts locaux;
- $I_L^O$ est un ensemble d'individus locaux et;
- $NS$ est un espace de noms des concepts de vue $C_L^O$.

**Définition 4.3 (Point de vue)** : Un point de vue est la vision d'un utilisateur et correspond à l'utilisation spécifique des concepts de vue. Cette utilisation peut engendrer une hiérarchie des concepts pouvant être rattachés à plusieurs vues.

Un point de vue $PV^O$ est définiformellement par ($C_{MVP}^O$,$A_{MVP}^O$,$R_{MVP}^O$, $I_{MVP}^O$,, ≤ ) où:

- $C_{MVP}^O$ est un ensemble de concepts multipoints de vue, un concept de cet ensemble est nommé « concept Multiviewpoints »
- $A_{MVP}^O$ est un ensemble des attributs de concepts multipoints de vue
- $R_{MVP}^O$ est un ensemble de relations entre concepts multipoints de vue.
- $I_{MVP}^O$ est un ensemble d'individus, instances des concepts « Multiviewpoints »,
- ≤ opérateur de subsomption afin de classifier les concepts $C_{MVP}^O$ dans une hiérarchie $PV^O$ d'un utilisateur aussi nommée hiérarchie des concepts multipoints de vue.

### 4.2 Description du patron de conception multipoints de vue

Notre patron de conception d'ontologie multipoints de vue (Kasri & Benchikha, 2014) (Kasri & Benchikha, 2014) est inspiré du patron de conception en génie logiciel « stratégie » (Gamma, Helm, Johnson, & Vlissides, 1995) présenté dans le chapitre 2. Stratégie est un patron de comportement qui définit une famille d'algorithmes, encapsule chacun d'eux et les

rend interchangeables. Stratégie permet aux algorithmes de varier indépendamment des clients qui les utilisent.

Le patron multipoints de vue (MV2P : MultiViewPoints Pattern) est un patron de conception d'ontologie d'architecture. L'application du patron d'architecture dans un domaine donné (tourisme, biologie) donne un patron de contenu qui modélise les problèmes concernant le contenu de l'ontologie de domaine. Ces problèmes sont récurrents et peuvent être généraux où spécifiques à un domaine particulier. Dans notre travail, nous nous intéressons à l'intégration de la notion de point de vue dans le contenu de l'ontologie. Les primitives de la recommandation du W3C pour le langage de l'ontologie ne peuvent pas prendre en compte des expressions du point de vue. Pour cela, nous proposons un patron de conception d'ontologie d'architecture en utilisant la représentation UML qui permet de rendre compte des différents points de vue dans un modèle unique. Nous avons choisi UML comme un langage de représentation pour être indépendant de tout langage d'implémentation.

Nous présentons ci-dessous la description de notre pattern multipoint de vue selon le template (gabarit) suivant (Kasri & Benchikha, 2014) (Kasri & Benchikha, 2016) : nom, problème à résoudre, solution et conséquences d'utilisation.

a) **Le nom du patron :** MV2P
b) **Le problème :** Décrire les concepts multipoints de vues dans une ontologie non canonique.
c) **La solution :** Définir informellement les concepts clés d'une ontologie multipoints de vues et les présenter dans le contexte des exigences qu'ils permettent de couvrir. Dans ce qui suit, nous décrivons le développement du patron MV2P par rapport aux exigences présentées ci-dessus. Pour être plus clair, nous présenterons aussi un extrait OWL correspondant aux quelques solutions proposées en utilisant le tableau de correspondance 4.2. À l'instant, ce tableau contient seulement les correspondances des éléments nécessaires pour décrire notre patron. Il peut être étendu pour supporter d'autres éléments.

| Elément de diagramme de classe UML | Elément d'une ontologie OWL |
|---|---|
| Classe UML | Classe OWL |
| héritage | Subsomption par rdfs:subClassOf |
| Relation entre deux classes | Relation de type wl:ObjectProperty |
| attribut | Relation de type owl:DatatypeProperty |
| Cardinalité 1..* | Restriction par owl:someValuesFrom |
| Autre cardinalité | Restriction de cardinalité sur les relations |

**Tableau 4.2** Correspondance entre UML et OWL

- **Exigence 1** : Le patron multipoints de vue doit être évolutif. Le contenu partageable (global) et le contenu partiel d'une vue spécifique sont décomposés de telle sorte qu'ils puissent varier indépendamment et sans redondance. Le découplage des contenus accroît les possibilités d'extension.

Pour séparer les propriétés partageables des concepts et les propriétés spécifiques aux vues, deux concepts sont définis : LocalView et GlobalView:

- Concept **LocalView**:Il représente une vue d'un objet réel. Il contient les connaissances pertinentes à cette vue.
- Concept **GlobalView**:Il contient les connaissances partageables entre les différentes vues de même objet réel.

En prenant l'exemple de chapitre 2, un logement est décrit par les propriétés suivantes : prix, charge, nbr_pièces, nbr_balcons, localisation, adresse, étage et surface (voir la figure 4.2).

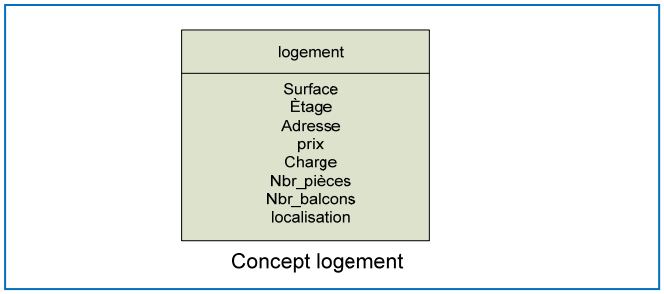



**Figure 4.2** Concept logement

La partie de l'ontologie suivante présente le concept logement de la figure 4.2 en OWL.

```
<owl:Classrdf:ID="Logement" />

<owl:DatatypePropertyrdf:ID="surface">
<rdfs:domainrdf:resource="#Logement" />
<rdfs:rangerdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
 ……
 ……
<owl:DatatypePropertyrdf:ID="Localisation">
<rdfs:domainrdf:resource="#Logement" />
<rdfs:rangerdf:resource="&xsd;String"/>
</owl:DatatypeProperty>
```

En utilisant les nouveaux concepts, deux types de propriétés seront distingués : des propriétés de la vue globale qui sont partageables et les propriétés spécifiques qui seront regroupées selon la vue à laquelle elles appartiennent (voir la figure 4.3).

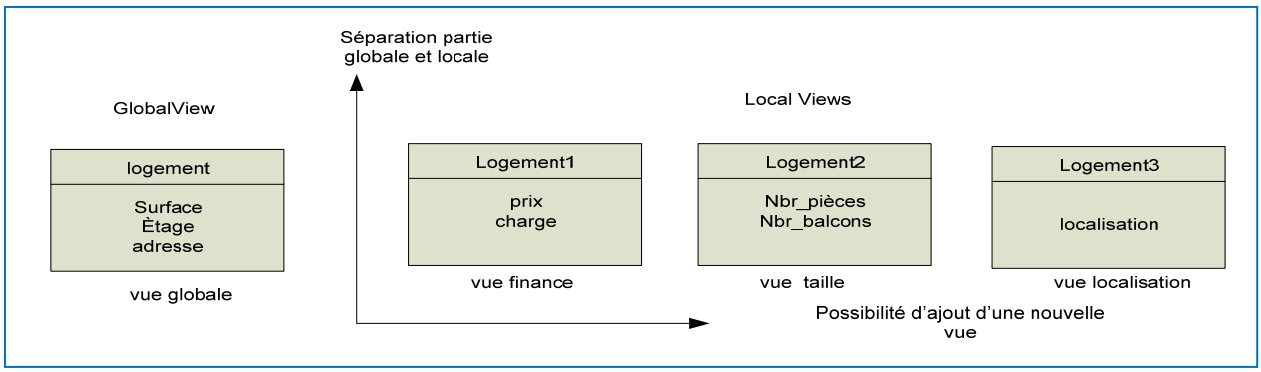



**Figure 4.3** Séparation de contenu (vue globale/vues locales)

- **Exigence 2** : La solution doit permettre aux utilisateurs d'ontologie la construction des concepts multipoints de vue.

Le patron permet la représentation multiple d'un même concept grâce aux concepts GlobalView et LocalView. Pour permettre cette construction on définit :

- Concept **Multiviewpoints** : C'est un concept doté de deux relations pour encapsuler le concept global (GlobalView) d'un objet et le rattacher à un ou plusieurs concepts locaux (LocalView).
- Relation **has_globalView** : C'est une relation entre le concept Multiviewpoints et le concept GlobalView qui encapsule son identification partageable.
- Relation **has_localViews** : C'est une relation rattachant le concept Multiviewpoints aux concepts de type LocalView.

La figure 4.4 présente le concept Multiviewpoints 'Logement' avec le GlobalView et les différents LocalViews.

Dans ce qui suit nous utilisons une convention de nommage pour différencier entre les concepts en déterminant le nom du concept, le type de concept et la vue du concept si le concept est un concept LocalView et en utilisant ( ::) pour séparer entre eux.

Nom du concept :: Type de concept :: [vue de concept ]

Où

- Nom de concept : c'est un concept de domaine comme personne, logement, voiture…etc.
- Type de concept : c'est l'un des trois types : GlobalView, LocalView, Multiviewpoints
- Vue de concept : c'est une vue de concept dans le domaine comme finance, taille, localisation…etc.

En utilisant cette notation, le concept logement de la vue finance et celui de la vue globale sont présentés respectivement comme suit :

- Logement ::LocalView ::Finance
- Logement ::GlobalView

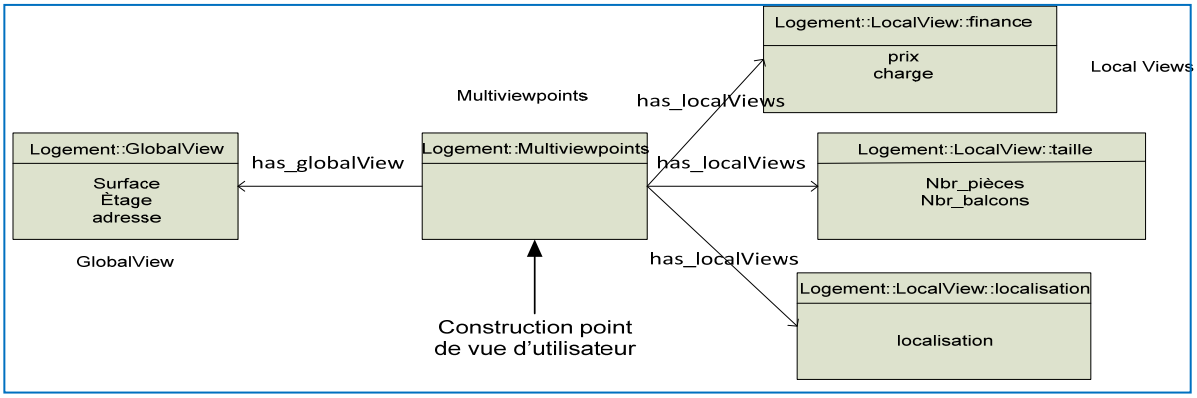



**Figure 4.4** Le concept « Multiviewpoints » et ses différents concepts « LocalView » et « GlobalView »

La partie de l'ontologie suivante représente le concept Multiviewpoints et ses vues locales et globale en OWL.

```
<owl:ObjectPropertyrdf:ID=" has_GlobalView">
<rdfs:domainrdf:resource="#Logement:: Multiviewpoints  " />
<rdfs:rangerdf:resource="#Logement::GlobalView" />
</owl:ObjectProperty>
<owl:ObjectPropertyrdf:ID=" has_localViews">
<rdfs:domainrdf:resource="#Logement:: Multiviewpoints  " />
<rdfs:range>
<owl:Class>
<owl:unionOfrdf:parseType="Collection">
<owl:Classrdf:about="# Logement::LocalView::Taille "/>
<owl:Classrdf:about="#Logement::LocalView::Finanace "/>
<owl:Classrdf:about="#Logement::LocalView::Localisation"/>
</owl:unionOf>
</owl:Class>
</rdfs:range>
</owl:ObjectProperty>
```

Dans le cas présent, le patron sépare entre les différentes vues et le contenu partageable par la création de nouveaux concepts (GlobalView, LocalView). Cependant, le concept Multiviewpoints crée ne présente qu'une restructuration ou un morcellement de l'objet initial en définissant dessus des relations has_localViews et has_globalView. Ces relations permettent une séparation du contenu en termes de vues sur l'objet mais l'objet reste manipuler comme un tout par l'utilisateur. En effet, nous rappelons que l'un des objectifs de l'intégration de la notion de point de vue est de permettre aux utilisateurs selon leurs besoins de sélectionner tout ce qui les intéressent et pas plus. Nous rappelons aussi que l'objectif des

patrons des conceptions d'ontologie est de permettre au langage d'implémentation d'ontologie comme OWL (ou autres) de supporter des solutions de conception complexes sans mettre en cause le langage lui-même c.-à-d. sans modification ou extension au niveau de méta-modèle du langage. En effet, dans ces conditions, cette solution semble insatisfaisante. Une nouvelle solution est proposée dans ce qui suit par l'ajout d'un nouveau concept 'AbstractView' pour permettre de couvrir la troisième exigence.

- **Exigence 3** : Le patron doit permettre aux utilisateurs de créer des points de vue qui les intéressent en cachant ou en combinant plusieurs vues (un accès restreint selon le point de vue de l'utilisateur)

Le patron doit restreindre l'accès aux vues selon le point de vue de l'utilisateur. A cet effet, le concept « AbstractView » est défini. « AbstractView »permet de cacher des vues d'un concept « Multiviewpoints ».

- Concept **AbstractView** : il représente la racine de toute hiérarchie de vue locale. Il s'agit du concept abstrait qui est utilisé pour rendre les vues visible ou invisibles à un utilisateur selon son intérêt (voir figure 4.5).
- Relation **has_localViews** : elle devient une relation entre le concept « Multiviewpoints » et le concept « AbstractView ». Cette relation peut apporter des contraintes sur les types des instances de vue qui participent à la relation.

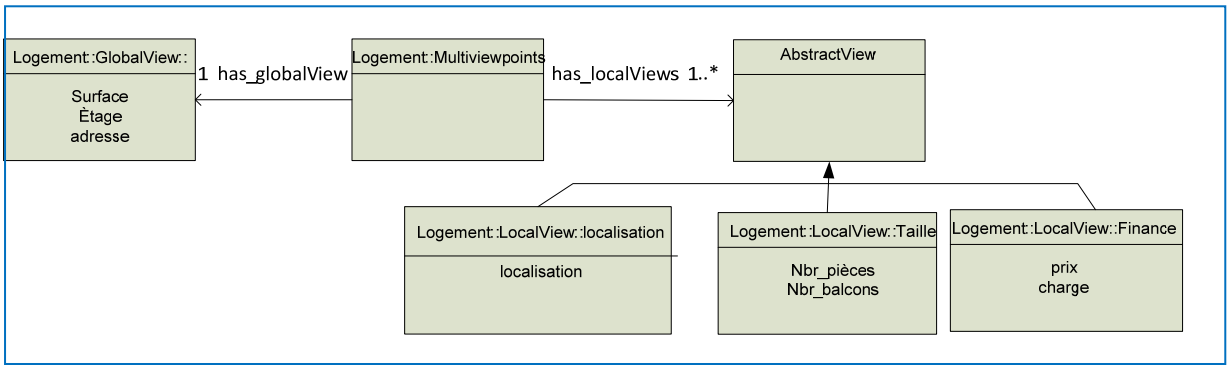



**Figure 4.5** La visibilité des concepts « LocalView » derrière le concept « AbstractView »

L'implémentation OWL suivante illustre l'utilité du concept « AbstractView » pour cacher les différentes vues du concept « Multiviewpoints ».

```
<owl:Classrdf:ID="logement::Multiviewpoints">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onPropertyrdf:resource="#has_localViews" />
<owl:someValuesFromrdf:resource="#AbstractView" />
</owl:Restriction>
</rdfs:subClassOf>
  ...
</owl:Class>
<owl:Classrdf:ID="Logement::LocalView::Taille">
<rdfs:subClassOfrdf:resource="#AbstarctView" />
</owl:Class>
<owl:Classrdf:ID="Logement::LocalView::Finance">
<rdfs:subClassOfrdf:resource="#AbstarctView" />
</owl:Class>
<owl:Classrdf:ID="Logement::LocalView::Localisation">
<rdfs:subClassOfrdf:resource="#AbstarctView" />
</owl:Class>
```

Cette solution est plus satisfaisante. L'utilisateur utilise le concept Multiviewpoints afin d'indiquer les différentes vues qui l'intéressent. Ce concept est attaché au concept AbstractView par la relation has_localViews qui prend comme domaine des instances de type 'AbstractView'. Les instances des concepts (Logement::LocalView::Taille, Logement::LocalView::Finance, Logement::LocalView::Loaclisation) sont de type AbstractView grâce à la relation de subsomption. (rdfs:subClassOf). En effet, lorsque l'utilisateur utilise le concept Multiviewpoints pour créer son point de vue, il n'a pas besoin de connaitre toutes les vues puisqu'il utilise seulement quelques vues grâce à la restriction sur le concept AbstractView contenant la contrainte (owl:someValuesFrom).

- **Exigence 4** : Les différentes vues de même objet réel doivent avoir le même identifiant.

Pour lier les différentes vues de même objet réel avec le même identifiant, nous définissons le concept « IDRef » :

- Le concept **IDRef** : c'est le concept qui a été utilisé pour identifier les différentes vues de même objet réel avec le même identifiant unique (*cf* figure 4.6).
- Relation**Has_IDRef**: relation entre le concept (co-domaine) « IDRef » et le concept« Multiviewpoints ».

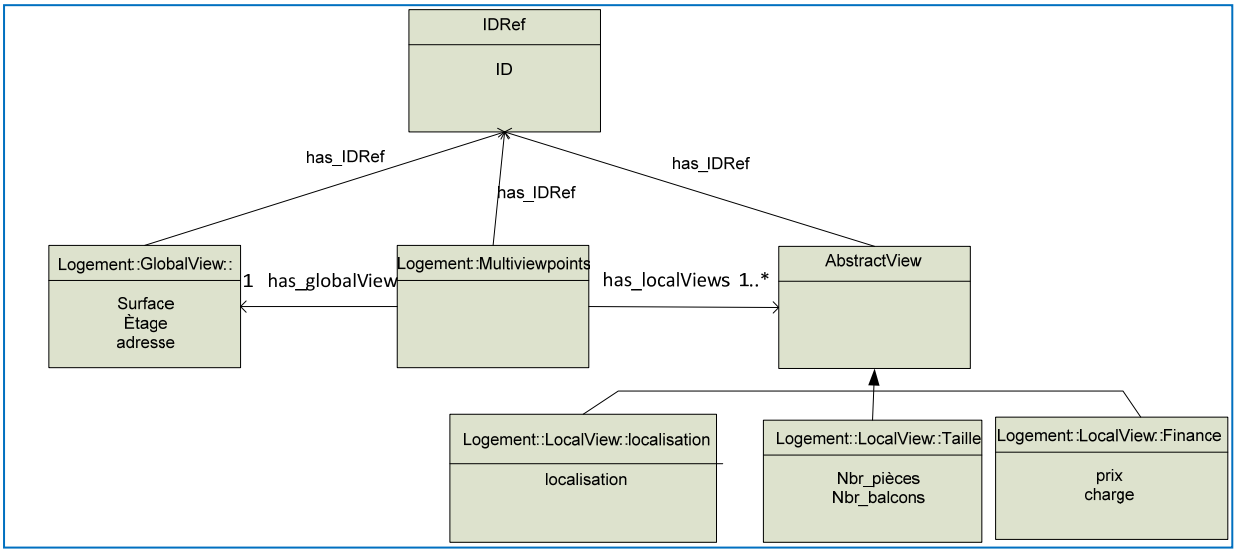



**Figure 4.6** Les différentes vues de même objet réel liées au même identifiant

- **Exigence 5** : Le patron doit supporter une recherche approfondie au niveau de chaque ensemble de concepts de même vue.

Pour faciliter l'accès aux vues, le patron permet une recherche approfondie à partir de la racine de chaque vue en ajoutant un 'espace de noms appelé « NameSpace » pour chaque hiérarchie des concepts LocalView.

- Le concept **NameSpace**: c'est un niveau supplémentaire des concepts noté (nameSpace :: vue de domaine) qui représente les racines des hiérarchies de vue. Ce niveau est utile pour regrouper des concepts de même vue dans la même hiérarchie. L'avantage principal d'un espace de noms est qu'une recherche approfondie à partir de la racine de l'espace de noms d'une vue se fera dans l'ensemble des concepts de même vue (*cf* figure 4.7).

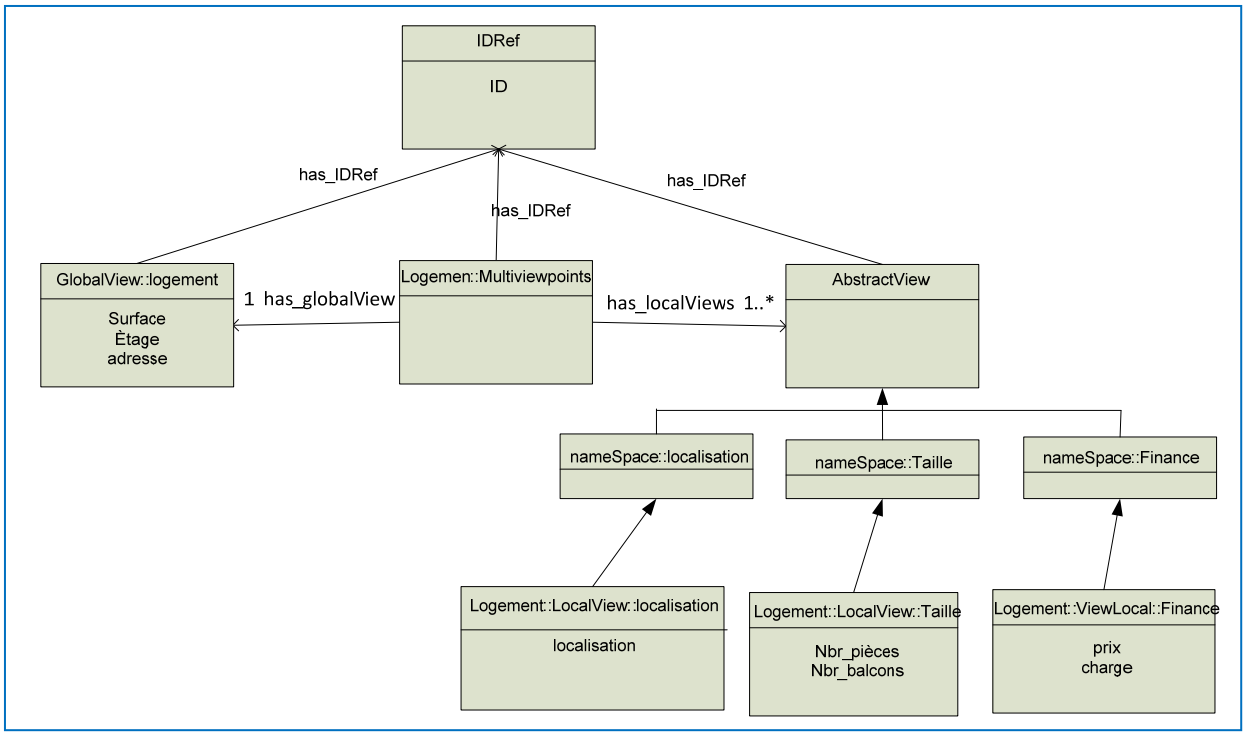



**Figure 4.7** Niveau des espaces de noms.

- **Exigence 6** : Le patron doit permettre la préservation des liens sémantiques entre les différentes vues de même concept.

Les liens sémantiques, appelés aussi passerelles, sont représentés par l'axiome « with_Bridge ».

- Axiome **with_Bridge**: c'est une dépendance qui exprime un lien entre les concepts de vue. Une passerelle symbolise, à la fois, la séparation entre deux concepts des vues distinctes et la liaison entre eux. Par exemple, nous exprimons qu'un logement de grande taille (vue Taille) est toujours un logement cher (vue Finance) par une passerelle qui lie leurs concepts. Dans le patron, les passerelles inter vues implémentées sont déterminées par un expert où avec l'utilisation des différentes techniques de mise de correspondance (alignement) (Euzenat, 2004) (Kasri & Benchikha, 2011) afin de préciser les liens entre les concepts.
  Pour mieux exprimer ces passerelles, nous introduisons des patrons de correspondance des vues. Les patrons de correspondance sont destinés à fournir des modèles de référence en aidant à modéliser les alignements d'ontologies, à l'instar des patrons de conception dans l'ingénierie logicielle qui aident à modéliser les conceptions de logiciels (Gangemi & Presutti, 2009).

- **Exigence 7** : La création d'un concept multipoints de vue est régie par certaines contraintes afin de contrôler la coexistence entre vues.

La combinaison entre les vues doit être contrôlée. Pour cela des contraintes sur le concept « Multiviewpoints » sont définies.

- Axiome **with_Constraints**: cet axiome comporte plusieurs règles qui agissent sur la cohérence d'un concept multipoints de vues lors de sa création à partir des points de vue liés par des passerelles.

Notre patron est décrit en UML et nous complétons sa description par l'intégration des contraintes en utilisant OCL[1](Warmer, & Kleppe, 1999).Une contrainte est une restriction sur une ou plusieurs valeurs dans le modèle orienté objet ou une partie de ce modèle. Elle est formulée en fonction d'un diagramme de classes et appliquée au niveau des objets. Dans l'approche orientée objet, une contrainte OCL peut porter sur une classe, appelée Contexte de la contrainte. Plusieurs stéréotypes sont applicables comme inv[2], pre[3], post[4], etc. Dans le cadre de nos travaux, nous nous intéressons aux invariants (inv) sur le concept « Multiviewpoints ». Les invariants sont des conditions qui doivent être vraies et respectées par chaque instance en permanence. Dans ce qui suit, les différentes contraintes sont présentées sur le Contexte Multiviewpoints pour restreindre la coexistence de différentes vues dans le même concept « Multiviewpoints » créé selon le point de vue d'un utilisateur (Kasri & Benchikha, 2016).

- **Contrainte No. 1**
  - **Concept concerné :** concept Multiviewpoints
  - **Description :** Le multipoints de vue créé par l'utilisateur (point de vue de l'utilisateur) doit contenir au moins une des vues locales.
  - **Expression OCL:**
  - Context Multiviewpoints inv:
  - Multiviewpoints.has_localViews->notEmpty()

- **Contrainte No. 2**
  - **Concept concerné:** concept Multiviewpoints
  - **Description:** Le multipoints de vue créé par l'utilisateur (point de vue de l'utilisateur) doit contenir une seule vue globale.
  - **Expression OCL:**
  - ContextMultiviewpoints inv:
  - Multiviewpoints.has_globalView->size()=1

- **Contrainte No. 3**
  - **Concept concerné:** concept Multiviewpoints
  - **Description:** Le multipoints de vue créé par l'utilisateur (point de vue de l'utilisateur) ne doit par contenir des vues redondantes.

[1]OCL est un langage de contrainte du modèle orienté objet
[2]inv : invariant de classe qui exprime une contrainte qui doit être respectée en permanence et vrai pour les instances de ce type à n'importe quel moment.
[3]pre : précondition qui permet de spécifier une contrainte qui doit être vérifiée avant l'appel d'une opération
[4]post : postcondition qui permet de spécifier une contrainte qui doit être vérifiée après l'appel d'une opération.

- **Expression OCL:**
- Context Multiviewpoints inv:
- Not(Multiviewpoints.has_localViews.allInstances->exist(v1,v2:LocalView|v1=v2) )

- **Contrainte No. 4**
  - **Concept concerné:** concept MultiViewpoints
  - **Description:** le multipoints de vue créé par l'utilisateur ne doit pas contenir des vues ayant des ponts d'exclusion.
  - **Expression OCL:**
  - Context Multiviewpoints inv:
  - Not(Multiviewpoints.has_localViews.allInstances->exist(v1,v2:LocalView|v1.bridgexclude->includes(v2)))

- **Contrainte No. 5**
  - **Concept concerné:** concept Multiviewpoints
  - **Description:** Le multipoints de vue créé par l'utilisateur (point de vue de l'utilisateur) ne doit pas contenir d'une vue abstraite.
  - **Expression OCL:**
  - Context Multiviewpoints inv:
  - Multiviewpoints.hasLocalViews.allInstances -> select(oclType = AbstractView) ->isEmpty()

- **Contrainte No. 6**
  - **Concept concerné:** concept Multiviewpoints
  - **Description:** les vues d'un multipoints de vue créé par un utilisateur (point de vue de l'utilisateur) ne doivent pas êtredes espaces de nom.
  - **Expression OCL:**
  - Context Multiviewpoints inv:
  - Multiviewpoints.has_LocalViews.allInstances -> select(oclType = spaceName) ->isEmpty()

- **Contrainte No. 7**
  - **Concept concerné:** concept Multiviewpoints
  - **Description**: Le multipoints de vue créé par l'utilisateur (point de vue de l'utilisateur) doit contenir des vues ayant le même identifiant.
  - **Expression OCL**:
  - Context Multiviewpoints inv:
  - Multiviewpoints.has_GlobalView.hasIDRef=any(Multiviewpoint.has_localViews->forAll(v1,v2:LocalView|v1.hasIDRef=v2.hasIDRef) )

### a) Conséquences :

– Généralement les ingénieurs d'ontologie utilisent l'héritage multiple pour exprimer le fait qu'une instance peut être rattachée à plusieurs concepts. Cependant cette solution d'utiliser une relation de spécialisation multiple oblige à créer de nombreux concepts pour exprimer

toutes les combinaisons de différentes vues. Pour cela nous utilisons la relation has_localViews. Cette solution permet à une instance de concept Multiviewpoints de prendre un ou plusieurs vues avec la même structure du patron. Cette cohabitation de plusieurs vues au même temps dans le même patron permet d'exprimer des contraintes inter vues.

- La spécialisation au niveau des hiérarchies des concepts GlobalView et LocalView répond bien à la demande d'affinage des vues. Par exemple, il est possible d'affiner la description d'une vue au niveau des concepts LocalView à l'aide de mécanisme d'héritage par (voir figure 4.8 (a)) :
    - Extension de la liste de propriétés des concepts LocalView :
    - Restriction sur les valeurs de propriétés des concepts LocalView.
- La présence de concept AbstractView comme racine des hiérarchies des concepts LocalView donne au patron une capacité d'évolution. Il est possible d'ajouter ou supprimer dynamiquement un concept LoacalView ou une vue complète (une hiérarchie complète à partir du concept nameSpace ::LocalView) sans modifier la structure de patron (voir figure 4.8).

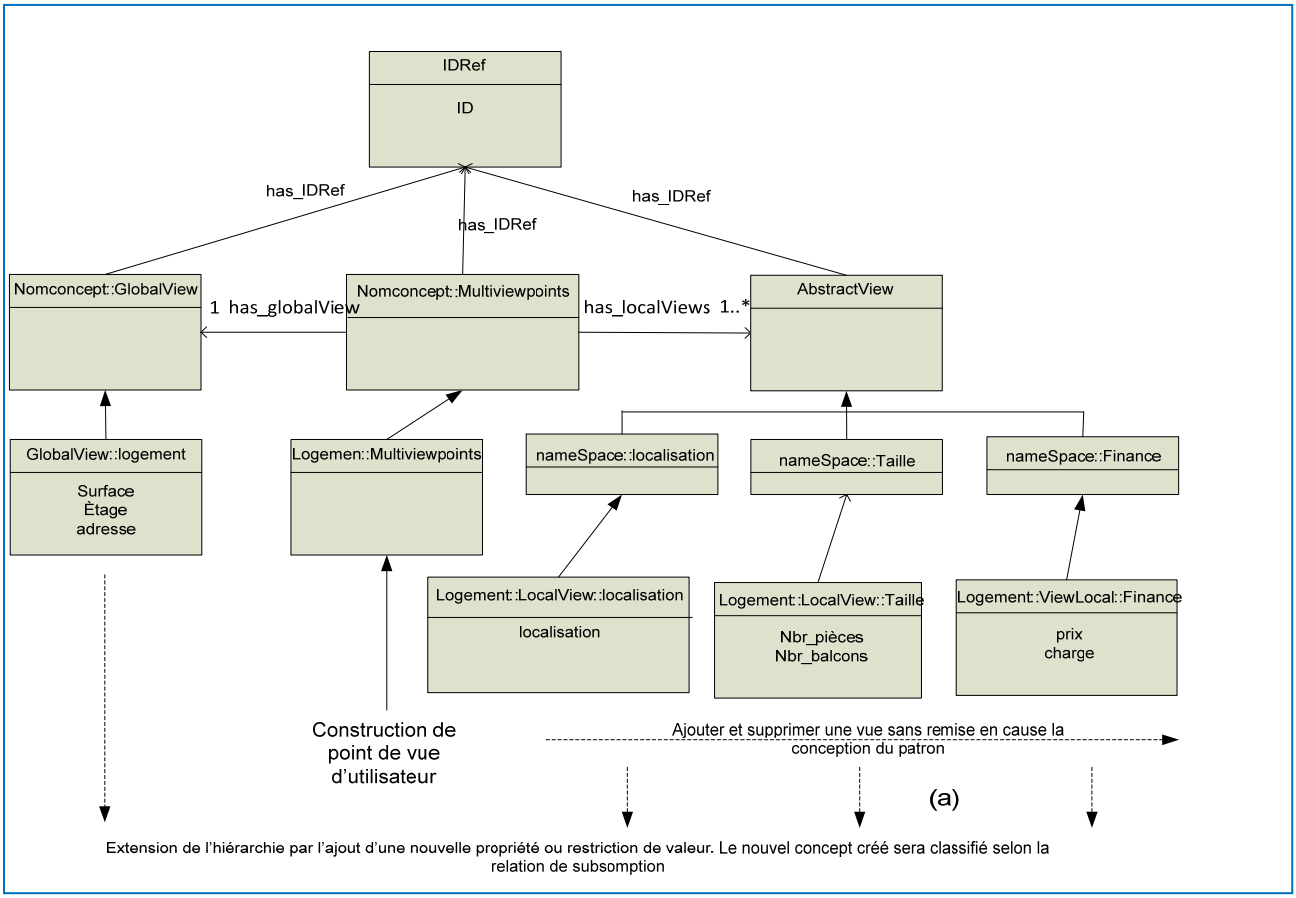



**Figure 4.8** L'addition/suppression et extension/restriction des vues

- Les contraintes sur la relation has_LocalViews permettent une représentation plus précise de celle-ci et interdisent des cohabitations incohérentes (utilisation des vue contradictoire) des vues.

### 4.3 Les patrons de correspondance des vues

Les passerelles inter-vues sont déterminées par un expert ou en utilisant des différentes techniques d'alignement pour préciser les liens sémantiques entre les concepts. Ces liens sont représentés par des règles qui affirment la direction, la possibilité ou l'impossibilité du passage des correspondances d'une vue à un autre. Pour rendre la mise en œuvre des passerelles plus facile et plus expressive, nous proposons dans ce qui suit un ensemble de patrons de correspondance des vues qui permettent de lier les différents concepts de vue.

- **Patron de correspondance d'ontologie N0. 1**
  1. **Nom :** Passerelle d'inclusion totale
  2. **Problème :** deux concepts de vues différentes où l'un est inclus totalement dans l'autre.
  3. **Contexte:** le patron est utilisé lorsque tout individu du premier concept est aussi un individu du deuxième concept.
  4. **Solution :** le patron établit une correspondance entre les individus où chaque individu de la première vue est un individu de la deuxième vue. Cette passerelle peut être décrite par la règle :
     - soit $V1\ et\ V2 \in V^O$ sont des vues, $S, K \in C_L^O$ sont des concepts de vue notés par $(S :: LocalView :: V1)\ (K :: LocalView :: V2)$ et $i \in I_L^O$ est un individu de vue.
       - $\forall\ i \in (S :: LocalView :: V1) \Rightarrow i \in (K :: LocalView :: V2)$

- **Patron de correspondance d'ontologie N0. 2**
  1. **Nom :** Passerelle d'inclusion partielle
  2. **Problème** : deux concepts de vues différentes et l'un est inclus partiellement dans l'autre.
  3. **Contexte** : le patron est utilisé lorsqu'il existe quelques individus du premier concept qui sont des individus du deuxième concept.
  4. **Solution :** le patron établit une correspondance entre quelques individus de la première vue et autres individus de la deuxième vue. Cette passerelle peut être décrire par la règle :
     - soit $V1\ et\ V2 \in V^O$ sont des vues, $S, K \in C_L^O$ sont des concepts de vue notés par $(S :: LocalView :: V1)\ et\ (K :: LocalView :: V2)$ et $i \in I_L^O$ est un individu de vue.
       - $\exists\ i \in (S :: LocalView :: V1) \Rightarrow i \in (K :: LocalView :: V2)$

- **Patron de correspondance d'ontologie N0. 3**
  1. **Nom**: Passerelle d'équivalence
  1. **Problème**: deux concepts de vues différentes sont équivalents l'un à l'autre.
  2. **Contexte**: le patron est utilisé lorsque l'ensemble des individus du premier concept est le même du deuxième concept. Cette passerelle peut décrire par la règle :
  3. **Solution**: le patron établit une correspondance d'équivalence entre les deux concepts correspondants. Cette passerelle peut être décrite par la règle :
     - soit $V1 et\ V2\ \in V^O$ sont des vues, $S, K\ \in C_L^O$sont des concepts de vue notés par $(S::LocalView::V1)\ et(K::LocalView::V2)$et $i \in I_L^O$est un individu de vue.
       - $i \in (S::LocalView::V1) \Leftrightarrow i \in (K::LocalView::V2)$

- **Patron de correspondance d'ontologie N0. 4**
  1. **Nom:** Passerelle d'exclusion
  2. **Problème :** deux concepts de vues différentes sont disjoints l'un à l'autre.
  3. **Contexte:** le patron est utilisé lorsque tous les individus du première concept sont jamais des individus de deuxième concept.
  4. **Solution :** le patron établie une correspondance entre les individus lorsque un individu appartient au premier concept n'appartient pas au deuxième concept. Cette passerelle peut être décrite par la règle
     - soit $V1 et\ V2\ \in V^O$ sont des vues, $S, K\ \in C_L^O$sont des concepts de vue notés par $(S::LocalView::V1)\ et(K::LocalView::V2)$et $i \in I_L^O$est un individu de vue.
       - $\forall\ i \in (S::LocalView::V1)\ S \Rightarrow i \notin (K::LocalView::V2)$

Après avoir couvert les différentes exigences citées dans la section 3, nous résumons l'architecture générale de notre patron multipoints de vue MV2P dans la figure 4.9.

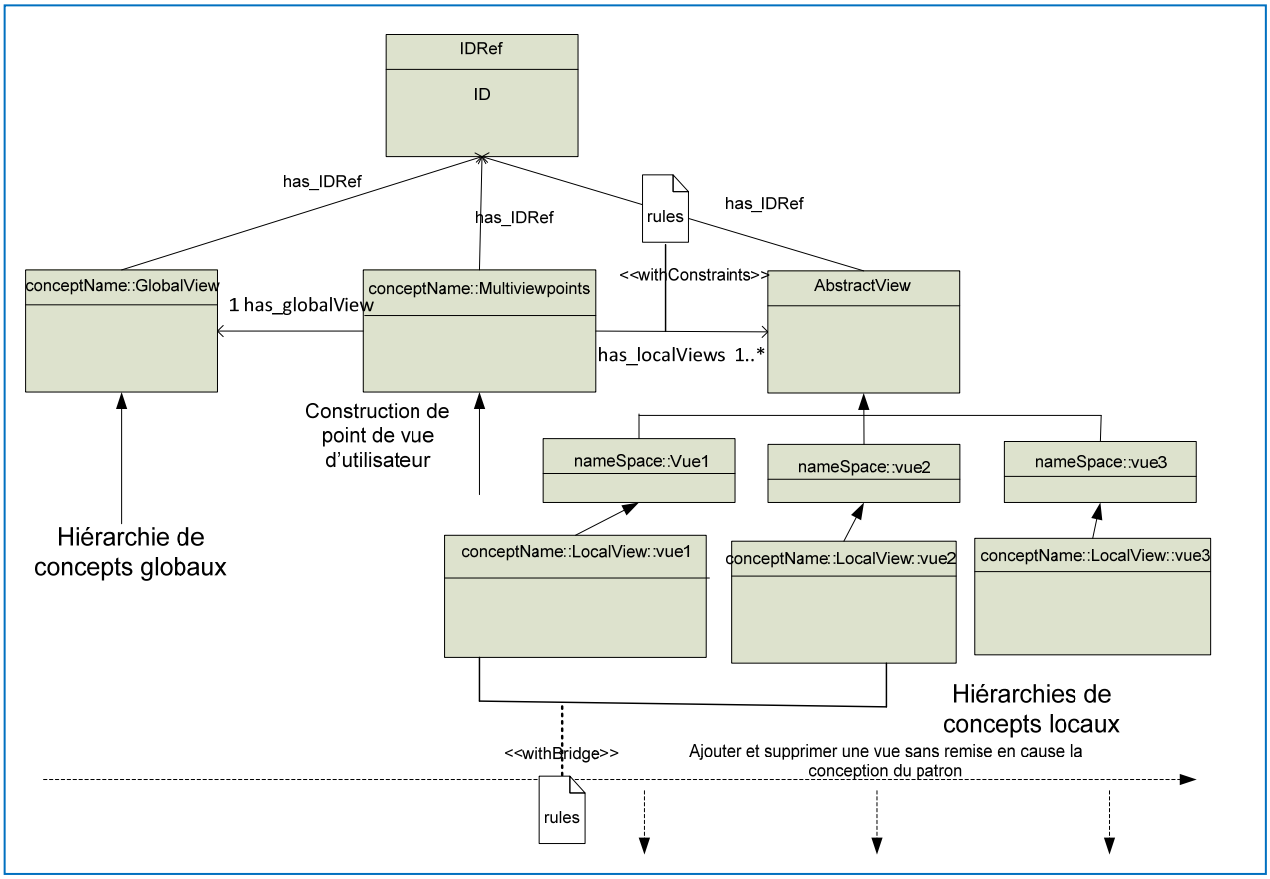



**Figure 4.9** Le patron MV2P en UML

## 4.4 Application

Nous illustrons notre patron par un exemple de système de location /vente des logements en ligne (*cf* figure 4.10). Dans cet exemple, pour le concept logement on suppose quatre vues :

a) la vue Taille contient les propriétés qui concernent la conception, l'architecture et la distribution du logement (nbr_pièces, nbr_balcons…etc),
b) la vue Localisation prend en compte l'environnement de logement (localisation : centreVille…etc.),
c) la vue Type qui décompose les logements suivant leur type; une maison ou un appartement (propriété :type…etc.),
d) la vue Finance s'occupe des dépenses de location du logement (charge, prix…etc.).

En plus, le concept logement a une vue globale qui regroupe toutes les propriétés communes entre les différentes vues.

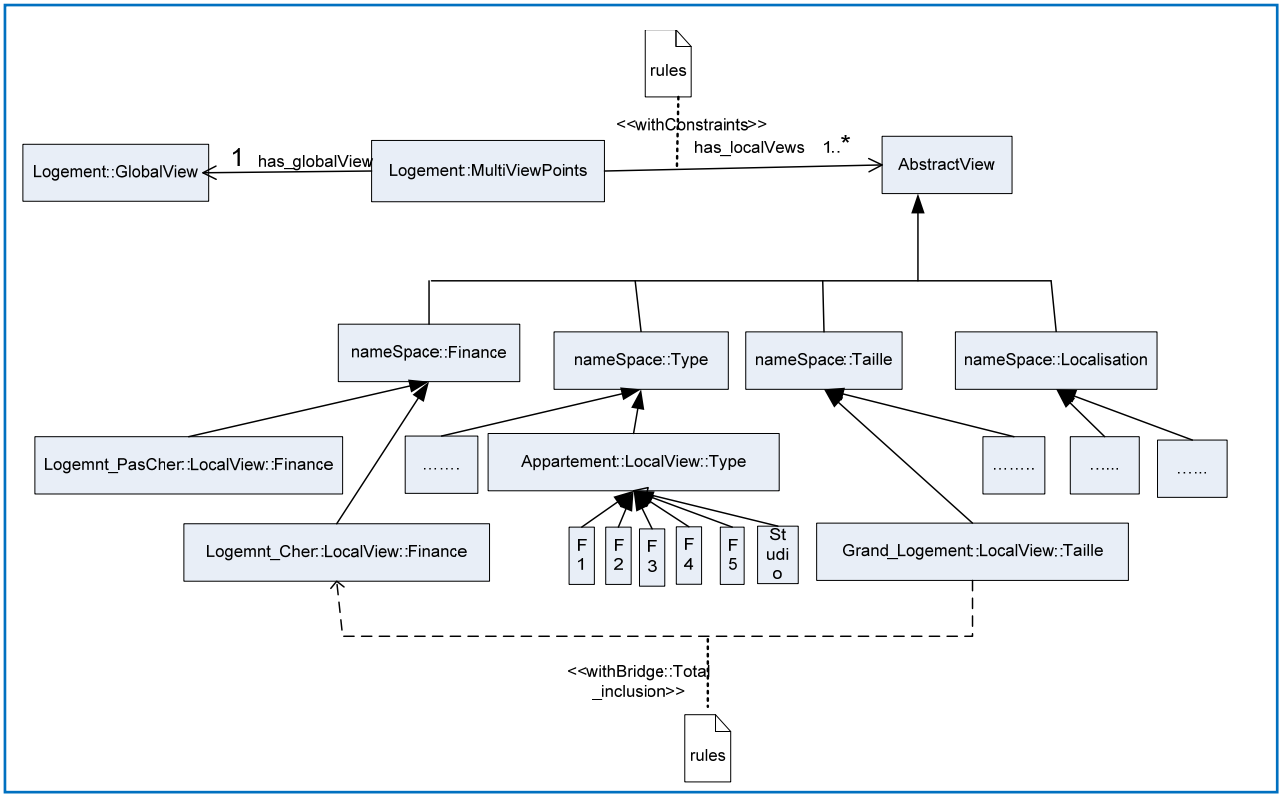



**Figure 4.10** Instanciation du patron MV2P dans le domaine de location /vente des logements.

Des axiomes peuvent être exprimés via les patrons de correspondance. Par exemple, si l'on sait que les grands logements sont chers, on peut exprimer cette connaissance inter-vue entre Taille et Finance pour le concept Logement à travers une passerelle d'inclusion totale du concept Grand_Logement vers le concept Logement_Cher comme suit :

- $\forall i \in$ *Grand_Logement ::LocalView::Taille* $\Longrightarrow$

  $i \in$ *Logement_Cher ::LocalView::Finance (inclusion totale)*

- $\forall i \in$ *Grand_Logement ::LocalView::Taille* $\Longrightarrow$

  $i \in$ *Logement_PasCher ::LocalView::Finance (inclusion partielle)*

# 5 Validation des patrons de conception d'ontologie

L'évaluation est une étape importante et centrale lorsqu'un nouveau modèle, approche ou méthode est proposé. En ce qui concerne les patrons de conception d'ontologie, il n'existe pas de méthodologies rigoureuses pour les évaluer. Cependant certains critères, tels que la réutilisation, peuvent être utiles pour évaluer le succès d'un patron [55]. Par conséquent, une évaluation complète du patron de conception d'ontologie proposé ne peut être donnée à ce stade. Toutefois, nous utilisons l'extension de l'analyse formelle des concepts (structure de patrons) pour compléter la spécification informelle du patron par une spécification formelle (Kasri & Benchikha, 2014) (Kasri & Benchikha, 2014). Celle-ci permet de vérifier si l'implémentation du patron permet de résoudre le problème de conception posé tout en

respectant le modèle de solution proposé. Le résultat de l'évaluation sur un exemple donné sera un treillis qui présente un point de vue de l'utilisateur courant.

L'AFC classique est valable pour construire un treillis de concepts ou les objets sont décrits par des attributs binaires mais pas pour les objets complexes comme les individus d'un point de vue d'un utilisateur (individu multipoints de vue) où leur description est un ensemble des individus de vue locale et globale. Cela signifie que pour construire un treillis de concepts des individus d'un point de vue, il faut définir un ordre partiel de leurs descriptions. C'est l'idée principale de la structure de patrons définie par Ganter and Kuznetsov (Ganter & Kuznetsov, 2001).Une telle structure se formalise par un triplet $(\mathrm{G}, (D,\sqcap), \delta)$ où $\mathrm{G}$ est un ensemble d'objets, $(D,\sqcap)$ est un infimum-demi-treillis d'éléments appelés patrons et $\delta$ est une fonction qui associe à tout objet $\mathrm{g} \in \mathrm{G}$ sa description $\delta(\mathrm{g})$ dans$D$. $(D,\sqcap)$est un ensemble partiellement ordonné de descriptions d'objets. L'infimum $\sqcap$ est une opération idempotente, commutative et associative qui induit une relation d'ordre ou relation de subsomption. Les patrons sont ordonnés par $\mathrm{c} \sqsubseteq \mathrm{d} \Leftrightarrow \mathrm{c} \sqcap \mathrm{d} = \mathrm{c}$ , $\forall \mathrm{c}, \mathrm{d} \in D$. Les deux opérateurs de dérivation suivants forment une connexion de Galois, avec $\mathrm{A} \subseteq \mathrm{G}$ et $\mathrm{d} \in (D,\sqcap)$,

$$A^{\square} = \prod_{\mathrm{g} \in \mathrm{G}} \delta(\mathrm{g}) \qquad (1)$$

$$d^{\square} = \{\mathrm{g} \in \mathrm{G} \mid \mathrm{d} \sqsubseteq \delta(\mathrm{g})\} \qquad (2)$$

Les concepts obtenus sont des couples $(\mathrm{A}, \mathrm{d})$ avec $\mathrm{A} \in \mathrm{G}$et $\mathrm{d} \in (\mathrm{D},\sqcap)$, tels que $\mathrm{A}^{\square} = \mathrm{d}$ et $\mathrm{A} = \mathrm{d}^{\square}$ et sont ordonnés par $(\mathrm{A}_1, \mathrm{d}_1) \leq (\mathrm{A}_2, \mathrm{d}_2) \Leftrightarrow \mathrm{A}_1 \subseteq \mathrm{A}_2$ $(\Leftrightarrow \mathrm{d}_2 \sqsubseteq \mathrm{d}_1)$ pour former un treillis de concepts.

Pour utiliser l'analyse des structures des patrons, il faut déterminer les éléments du triplet $(G, (D,\sqcap), \delta)$ où:

- $G$ est un ensemble d'objets ;
- $(D,\sqcap)$est un inf-demi-treillis d'éléments appelés patrons et ;
- et $\delta$ est une fonction qui associe à tout objet $g \in G$ sa description $\delta(\mathrm{g})$ dans $D$.

À cette fin, nous proposons une approche composée de quatre étapes pour la validation formelle du patron de conception d'ontologie multipoints de vue qui sont :

1) Transformation des objets au format applicable par l'analyse des structures de patrons
2) Construire l'espace $D$des patrons$\mathrm{g} \in \mathrm{G}$
3) Déterminer l'opérateur d'infimum $\sqcap$ pour classifier les descriptions des patrons $\mathrm{g} \in \mathrm{G}$
4) Déterminer la fonction d'affectation des descriptions$\delta$
5) Construction du treillis des concepts de point de vue de l'utilisateur (Treillis des concepts multipoints de vue)

Nous détaillons dans ce qui suit ces étapes pour construire un treillis de concepts d'un point de vue d'utilisateur par la structure de patron en exploitant le patron d'ontologie multipoints de vue. Ce treillis simule la création de l'hiérarchie des concepts d'un point de vue d'utilisateur (hiérarchie crée à partir de concept Multiviewpoints).

### 5.1 Transformation des objets au format applicable par l'analyse des structures de patrons

Pour rendre le patron exploitable par l'extension de l'analyse formelle 'les structures des patrons' les concepts essentiels à présenter sont : LocalView, GlobalView, Multiviewpoints et leurs instances (individus).

1) Concept LocalView ou GlobalView : pour rendre un concept de ce type applicable par l'analyse de structure de patron, on doit le représenter par un vecteur de couples (propriétés, domaine). La représentation du concept logement de la figure 4.10 est :

- **Exemple :**Logement ::GlobalView=<[numéro:String],[adresse:String],[étage:Integer],[surface: Integer]>Grand_Logement=<[nbr_Balcons : Integer],[nbr_Pièces>=3]>

2) une instance d'un concept LocalView ou GlobalView est représentée par le remplacement des domaines par des valeurs dans le vecteur (propriétés, domaine).

- **Exemple :**Log_global1=<[numéro=001],[adresse=Skikda],[étage=3],[surface=90]>

3) Concept Multiviewpoints : c'est le concept utilisé par l'utilisateur pour créer son point de vue. Il est représenté un ensemble des concepts GlobalView (un seul concept) et LocalView (un ou plusieurs) participés pour le construire en format des vecteurs :

- **Exemple :**Logement ::Multiviewpoints={<[numéro :String],[adresse :String],[étage :Integer],[surface :Integer]>,<[annexe :Park]>,<[nbr_Balcons : Integer],[nbr_Pièces>=3]>,<[loyer : Integer],[charges : Integer]>}

4) Une instance d'un concept Multiviewpoints est représentée par le remplacement des domaines par des valeurs dans les vecteurs (propriétés, domaine).

- **Exemple :**Log_PV1={<[numéro=001],[adresse=Skikda],[étage=3],[surface=90]>,<[annexe=Park00E]>,<[nbr_Balcons=3],[nbr_Pièces=4]>,<[loyer=60000],[charges=1000]>}

### 5.2 Construire l'espace $\boldsymbol{D}$ des patrons g $\in$ G(Espace d'Instanciation $\mathcal{D}$)

Du point de vue de design pattern, l'instanciation du patron MV2P consiste à remplacer les concepts du patron avec ceux d'un domaine d'application. L'instanciation d'un concept Multiviewpoints consiste à créer un individu ayant une propriété de type has_globalView et au moins une propriété de type has_localViews. Cet individu est crée suivant une opération d'instanciation dans un espace d'instanciation $\mathcal{D}$. Un espace d'instanciation $\mathcal{D}$ est composé des vecteurs qui représentent les concepts GlobalView et LocalView constituant le patron. Pour notre exemple de la figure 4.10, l'espace d'instanciation est :

- **Exemple :** $\mathcal{D}$ ={<[numéro:String],[adresse:String],[étage:Integer],[surface:Integer]>,<[annexe :Park]>,<[nbr_Balcons:Integer],[nbr_Pièces>=3]>,<[loyer:Integer],[charges:Integer]> ……..}

### 5.3 Déterminer l'opérateur d'infimum ⊓ pour classifier les descriptions des patrons $\mathbf{g} \in \mathbf{G}$

Les individus d'un point de vue d'utilisateur, créés à partir du concept « Multiviewpoints » notés $\boldsymbol{I^{O}_{MVP}}$, sont ordonnés au sein d'un inf.-demi-treillis (D, ⊓) où D présente l'espace d'instanciation $\boldsymbol{\mathcal{D}}$ de ces individus. L'intersection ensembliste ∩ possède les propriétés d'un infimum ⊓ dans un demi-treillis. Les individus $\boldsymbol{I^{O}_{MVP}}$ sont désignés par des ensembles. Nous utilisons l'intersection ensembliste comme un opérateur donnant l'infimum des patron $\mathbf{d_i}$ dans $\boldsymbol{\mathcal{D}}$.

### 5.4 Déterminer la fonction d'affectation des descriptions δ

La structure de patrons se formalisme par un triplet $(\boldsymbol{I^{O}_{MVP}}, (\boldsymbol{\mathcal{D}},\cap), \boldsymbol{\delta}(\boldsymbol{vpi}))$ où $\boldsymbol{I^{O}_{MVP}}$ est l'ensemble des individus Multiviewpoints (point de vue d'utilisateur), $(\boldsymbol{\mathcal{D}}, \cap)$.est un demi-treillis des individus Multiviewpoints et $\boldsymbol{\delta}$ est une fonction d'affectation qui associe a tout $\boldsymbol{vpi} \in \boldsymbol{I^{O}_{MVP}}$sa description $\boldsymbol{\delta}(\boldsymbol{vpi})$ dans $\boldsymbol{\mathcal{D}}$. Cette description est une représentation du concept Multiviewpoints crée par l'utilisateur dans l'espace d'instanciation $\boldsymbol{\mathcal{D}}$. on peut calculer $\boldsymbol{\delta}(\boldsymbol{vpi})$ par le regroupement de $\boldsymbol{\delta}$ de chaque individu de concept LocalView et GlobalView constituant $\mathbf{i}$. La fonction $\boldsymbol{\delta}$ d'un individu LocalView et GlobalView est donnée par la projection directe dans l'espace d'instanciation $\boldsymbol{\mathcal{D}}$.

### 5.5 Construction du treillis des concepts de point de vue d'utilisateur (Treillis des concepts multipoints de vue)

Nous pouvons classifier les individus Multiviewpoints (point de vue d'utilisateur) dans des concepts et les ordonner par les deux opérateurs de dérivation $\mathbf{A}$ et $\mathbf{d}$ [58] en remplaçant ⊑ par = $\mathbf{dans\ d}$ car nous cherchons à regrouper les individus qui sont dans la même catégorie (même concept).

Avant la construction de treillis de concepts, chaque concept multipoint de vue doit être examiné pour l'insérer dans le treillis de concepts. Pour cette raison, nous proposons un algorithme de validation basé sur les contraintes d'OCL présentées dans la section 4.

---

1:**Function** IsValidated(**In**: … **Out**: ***valid is Boolean***)

---

2://initiation of valid with true
*3:valid←false*
*4:……*
*5:if Constraint(……)//testing*
*6:valid←true*
*7:end if*
*8:……*

9:Returnvalid

---

Considérons par exemple la contrainte " il n'y a pas une passerelle d'exclusion entre les instances". L'algorithme de validation est comme suit:

---

*1:**Function** IsValidated(**In**: A'⊂ $I^{O}_{MVP}$, v1 ∉ A' **Out**: **valid is Boolean**)*

---

*2://initiation of valid with true*
*3:valid←false*
*4:for each v2 ∈ A'*
*5:ifNoExclusionBridge(v1,v2)//testing if there is a exclusion bridge*
*6:valid←true*
*7:end if*
*8:end for*
*9:Return valid*

---

En utilisant les opérateurs de dérivation, nous pouvons construire les concepts formels point de vue avec l'algorithme suivant :

---

*1.**Algorithm** ViewpointsConceptBuilding(**In**:$\mathcal{D}$ instanciation domain of pattern,$I^{O}_{MVP}$ set of Multiviewpoints individuals, **Out**: VPFC set of viewpoints formal concepts when a Concept is a pair(A,d)with A ⊆ $I^{O}_{MVP}$ and d ∈ ($\mathcal{D}$,∩))*

---

*2:while$I^{O}_{MVP}$ is notEmpty do*
*3: // initialize A*
*4:let A←{v1/v1 ∈ $I^{O}_{MVP}$ }*
*5://A' for testing*
*6:let A'←{v1/v1 ∈ $I^{O}_{MVP}$}*
*7://let vpfc∈ VPFC*
*8:vpfc← (A, OperatorA(A))*
*9:for each v2 ∈ $I^{O}_{MVP}$ do*
*10:if v2 ∉ A' and isValidated(v2, A')then*
*11:A←A ∪ {v2}*
*12:A'←A' ∪ {v2}*
*13:if A= OpeartorD(OperatorA(A))then*
*14:vpfc← (A, OperatorA(A))*
*15: else*
*16:Remove v2 from A*
*17:end if*
*18:end if*
*19:end for*
*20: VPFC ← VPFC ∪ {vpfc}*
*21:Remove A from $I^{O}_{MVP}$*
*22:endwhile*

---

Pour calculer les opérateurs de dérivation A□ et d□, les fonctions d'OperatorA et OperatorD sont définies comme suit:

*1:**Function** OperatorA(**In**: **A** ⊂ $I^{O}_{MVP}$, **Out**: **d** set obtained from intersection of A elements descriptions)*
*2://$vpi_i$ ∈ A,$\delta(vpi_i)$:direct projection of $vpi_i$ in D*
*3:d← $\delta(vpi_1)$∩ $\delta(vpi_2)$… ∩ $\delta(vpi_n)$//n is A dimension*
*4:Return d*

*1:**Function** OperatorD(**In**: **d** set obtained for OperatorA, **Out**: **A**⊂ $I^{O}_{MVP}$)*
*2://$vpi_i$ ∈ $I^{O}_{MVP}$ ,$\delta(vpi_i)$:direct projection of $vpi_i$ in D*
*3:A← {$vpi_i$ ∈ $I^{O}_{MVP}$/ d= $\delta(vpi_i)$}*
*4:Return A*

## 6 Application sur une étude de cas

Dans cette partie, nous utilisons notre patron MV2P dans le système de location /vente des logements en ligne. L'objectif ici est de monter comment construire les points de vue par un treillis de concepts qui seront par la suite transformés en une hiérarchie ontologique. Nous supposons dans un premier temps que nous avons déjà une ontologie multipoints de vues créée par le patron MV2P et nous étudions le problème de construire des points de vue par la structure de patrons en exploitant le patron MV2P. Notre objectif est de classifier des individus crée par l'utilisateur à partir du concept Multiviewpoints « Logement ::Multiviewpoints» (*cf* section 5*)* pour construire la hiérarchie de son point de vue.

Etant donné La structure de patrons ($I^{O}_{MVP}$, ($\mathcal{D}$,∩), δ(ipv)) dont l'espace d'instanciation D est :

D={<[numéro :String],[adresse :String],[étage :Integer],[surface :Integer]>,<[annexe=park 00E]>,<[nbr_Balcons :Integer],[nbr_Pièces>3]>,<[loyer :String],[charges : Integer]> ,<[localisation :String]>……}s

Les descriptions des individus crées par un utilisateur selon son point de vue dans le système sont :

- Log_PV1={<[numéro=001],[adresse=Skikda],[étage=3],[surface=140]>,<[annexe=park0 0E]>,<[ nbr_Balcons =3],[ nbr_Pièces =4]>,<[loyer=20000],[ charges =200]>}
- Log_PV2={<[numéro= 002 ],[adresse=Ager],[étage=2],[surface=120]>,<[annexe=park00 B1]>,<[ nbr_Balcons =2],[ nbr_Pièces =3]>,[ loyer =33476],[ charges =780]>}
- Log_PV3={<[numéro=003],[adresse=Skikda],[étage=3],[surface=140]>,<[annexe=park0 0H]>,<[ nbr_Balcons =3],[ nbr_Pièces =4]>,<[ loyer =25000],[ charges =200]>,<[localisation=centre-ville]>}

- Log_PV4={<[numéro=004],[adresse=Skikda],[étage=1],[surface=70]>,<[annexe=park00 OP]>,<[ nbr_Balcons =1],[ nbr_Pièces =2]>,<[ localisation=centre-ville]>}
- Log_PV5={<[numéro=005],[adresse=Annaba],[étage=4],[surface=65]>,<[annexe=Grage] >,<[ nbr_Balcons =1],[ nbr_Pièces =2]>,<[localisation=seaside]>}
- Log_PV6={<[numéro=006],[adresse=Skikda],[étage=1],[surface=80]>,<[annexe=park00 H2]>,<[ loyer =3000],[ charges =100]>,<[localisation=citycenter]>}
- Log_PV7={<[numéro=007],[adresse=Annaba],[étage=1],[surface=80]>,<[annexe=parkK H2]>,<[ loyer =3000],[ charges =100]>,<[localisation=citycenter]>}
- Log_PV8={<[numéro=008],[adresse=Skikda],[étage=2],[surface=65]>,<[annexe=park00 HT],[type=F2]>,<[nbr_Balcons=2],[nbr_Pièces=2]>,<[loyer=6000],[charges=200]>,<[loc alisation=citycenter]>}

Pour démonstration, nous appliquons les opérateurs de dérivation sur quelques individus de l'exemple pour calculer les concepts Multiviewpoints de ces individus et déterminer la relation de subsumption entre eux. Dans le treillis, un concept crée par un utilisateur sera représenté par un couple (A,d) où d est le patron (la description) qui représente le concept dans l'espace d'instanciation *D* et A est l'ensemble des individus des concept LocalView et GlobalView qui participent à sa construction. Notre démarche est alors la suivante :

- **Le premier opérateur : on prend $\boldsymbol{A} = \{\boldsymbol{Log\_PV1}, \boldsymbol{Log\_PV2}\}$ donc $\boldsymbol{A}^{\square} = \{\boldsymbol{Log\_PV1}, \boldsymbol{Log\_PV2}\}^{\square}$**

$\{\text{Log\_PV1}, \text{Log\_PV2}\}^{\square} = \delta\ (\text{Log\_PV1}) \cap \delta\ (\text{Log\_PV2})$

δ(Log_PV1 )={<[numéro :String],[adresse :String],[étage :Integer],[surface :Integer]>,<[an nexe=Park]>,<[nbr_Balcons :Integer],[nbr_Pièces>=3]>,<[loyer:String],[charges :Integer]>}

δ(Log_PV2) = {<[numéro :String],[adresse :String],[étage :Integer],[surface :Integer]>,<[a nnexe=Park]>,<[nbr_Balcons :Integer],[nbr_Pièces>=3]>,<[loyer :String],[charges :Integer]> }donc
$\{\text{Log\_PV1}, \text{Log\_PV2}\}^{\square}$={<[numéro:String],[adresse:String],[étage :Integer],[surface :Integer]> ,<[annexe=Park]>,<[nbr_Balcons:Integer],[nbr_Pièces>=3]>,<[loyer:String],[charges:Integer] >}

- **Le deuxième opérateur : on prend le résultat de $\{\boldsymbol{Log\_PV1}, \boldsymbol{Log\_PV2}\}^{\square}$ comme $\boldsymbol{d}$ et on calcule $\boldsymbol{d}^{\square}$**

$(\{< [\text{numéro: String}], [\text{adresse: String}], [\text{étage : Integer}], [\text{surface : Integer}] >, < [\text{annexe} = \text{Park}] >, < [\text{nbr\_Balcons: Integer}], [\text{nbr\_Pièces} >= 3] >, < [\text{loyer: String}], [\text{charges: Integer}] >\})^{\square} = \{\ \text{Log\_PV1}, \text{Log\_PV2}\ \}$

Puisque$\{\text{Log_PV1}, \text{Log_PV2}\}^{\square}$={<[numéro:String],[adresse:String],[étage :Integer],[surfac e :Integer]>,<[annexe=Park]>,<[nbr_Balcons:Integer],[nbr_Pièces>=3]>,<[loyer:String],[char ges:Integer]>}

Et ({< [*numéro*: String], [adresse: String], [étage : Integer], [surface : Integer] >,< [$annexe = Park$] >,< [$nbr\_Balcons$: $Integer$], [$nbr\_Pièces \geq 3$] >,< [$loyer$: $String$], [$charges$: $Integer$] >}) = { Log_PV1, Log_PV2 }

Alors({Log_PV1, Log_PV2 },{<[numéro:String],[adresse:String],[étage :Integer],[surface :Integer]>,<[annexe=Park]>,<[nbr_Balcons:Integer],[nbr_Pièces>=3]>,<[loyer:String],[charges:Integer]>} ) est un concept point de vue formel noté (A1,d1) car A = d et d =A. On a aussi :

(A2,d2)=({ Log_PV3 },},{<[numéro :String],[adresse:String],[étage :Integer],[surface :Integer]>,<[annexe :Park]>,<[nbr_Balcons :Integer],[nbr_Pièces>=3]>,<[loyer :String],[charges : Integer]>, <[localisation :String]>}) est un concept point de vue formel.

- **Classifier les concepts crées par l'utilisateur dans un treillis (point de vue d'utilisateur)**

Maintenant, les concepts d'utilisateur (concepts Multiviewpoints) sont ordonnés dans le treillis. Ces concepts sont représentés par les paires (A, d). Selon la représentation textuelle proposée précédemment A et d sont des ensembles des vecteurs. Dans notre approche, la classification est basée sur la description d. Donc $(A_1, d_1) \leq (A_2, d_2) \iff d_2 \sqsubseteq d_1$.Les deux concepts sont ordonnés par $(A_2, d_2) \leq (A_1, d_1)$ car $d_1 \sqsubseteq d_2$ pour former un treillis de concepts c.-à-d. le concept $(A_2, d_2)$ est subsumé par le concept$(A_1, d_1)$. Nous nommons les concepts LogementPV3 :: Multiviewpoints et LogementPV12 ::Multiviewpoints respectivement. La figure 4.11 présente le treillis résultant.

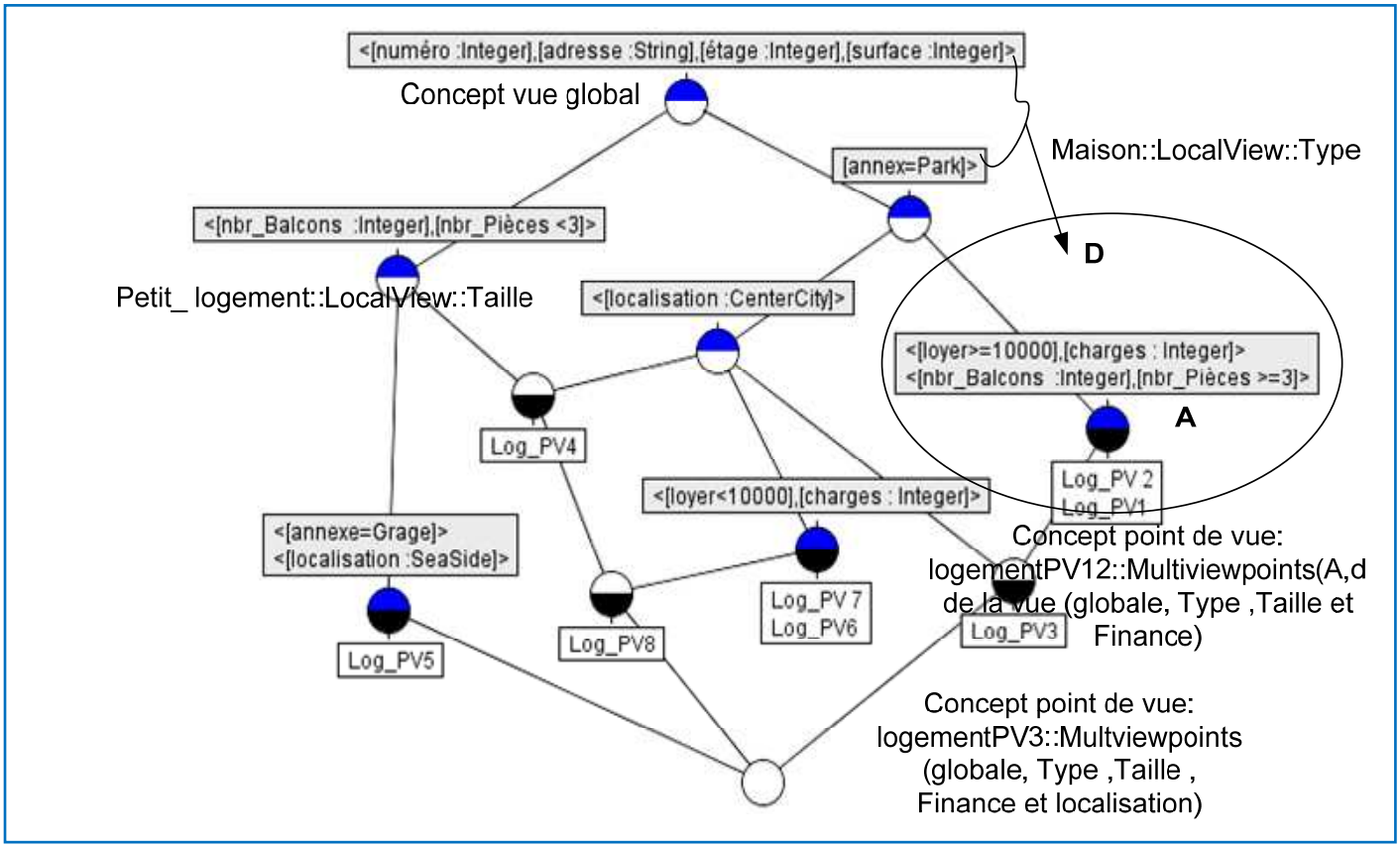



**Figure 4.11** Treillis des concepts point de vue

## 7 Cycle de vie du patron de conception d'ontologie

La conception d'une ontologie s'appuie sur les mêmes principes que ceux appliqués pour développer un composant logiciel. Cette tâche est difficile et nécessite la mise en place de procédés élaborés afin d'extraire la connaissance d'un domaine donné. Pour cela une ontologie doit posséder un cycle de vie qui nécessite d'être précisé à l'instar du cycle de vie d'un logiciel (Dieng et al., 2001) (Gandon, 2006). Dans le cadre de nos travaux, nous nous sommes plus particulièrement intéressées à l'utilisation des patterns dans la tache de conception pour construire une ontologie de qualité et réutilisable.

Dans ce qui suit nous proposons un cycle de vie d'un patron de conception d'ontologie. Il est défini par les différentes étapes nécessaires pour sa création et son utilisation. Deux acteurs manipulant les patrons peuvent être distingués:

1. Les concepteurs qui définissent les patterns et
2. Les ingénieurs d'ontologie qui utilisent ces patterns pour créer leurs ontologies.

Le cycle passe par quatre étapes comme indiqué dans la figure 4.12.

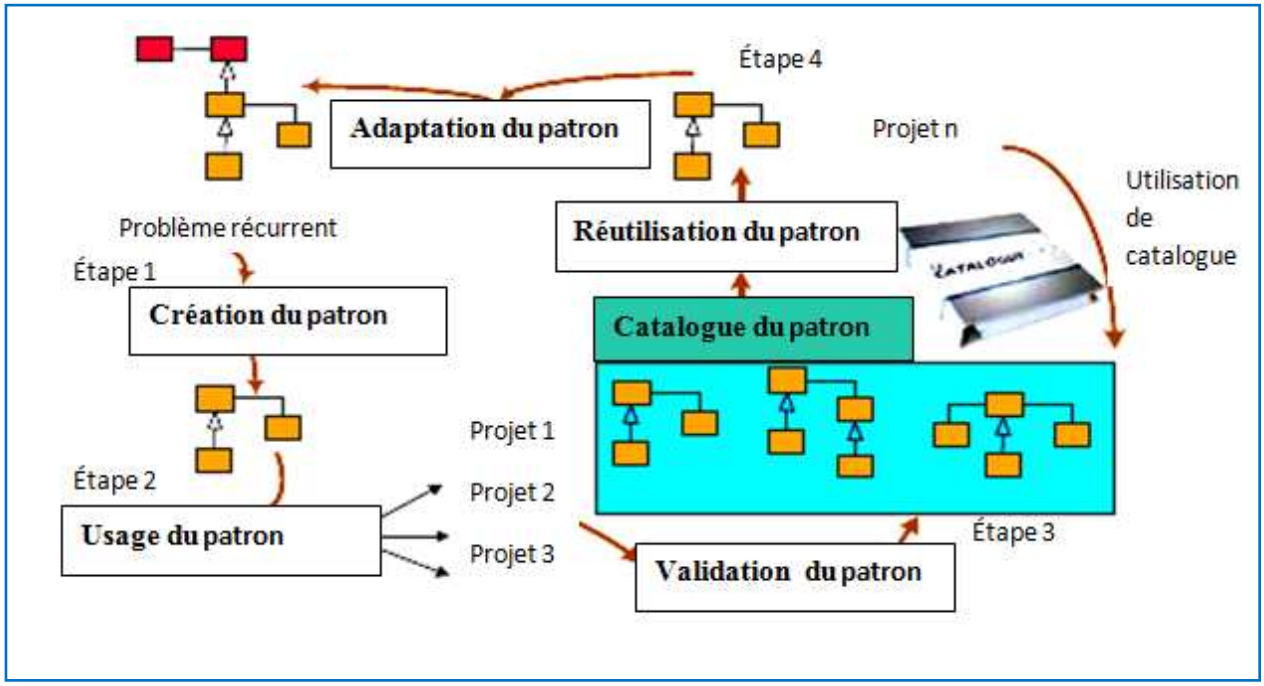



**Figure 4.12** Cycle de vie du patron de conception d'ontologie.

1. **Création** : durant cette phase, un modèle de solution est proposé pour le problème récurrent. La description d'un pattern particulier s'articule autour de quatre particularités essentielles :

– Le nom du pattern : le nom du pattern est bien entendu essentiel, et fait l'objet d'une standardisation dans le monde informatique. Cela facilite la communication entre les développeurs.

- L'énoncé du problème : où seront décrites la situation ou les situations dans lesquelles le pattern peut s'appliquer.
- La solution : qui décrit les éléments à mettre en jeux, leurs relations et leur collaboration. Cette solution est toujours générique et l'ingénieur d'ontologie aura pour tache de l'adapter à un contexte particulier.
- Conséquences : c'est l'impact du pattern c.-à-d. ce qui se passera s'il est implémenté.

2. **Usage** : Pour un ingénieur d'ontologie, qui est confronté à un problème récurrent, l'usage du pattern consiste à implémenter ce pattern dans un langage d'ontologie.
3. **Validation** : généralement un pattern est validé après trois utilisations (Shalloway, & Trott, 2002). Ensuite, il doit être Empaqueté dans le catalogue pour qu'il puisse être diffusable et réutilisable par d'autres projets.
4. **Réutilisation** : consiste à réutiliser le pattern dans les différents contextes. la démarche commence par la détection du pattern adéquat. Cette détection consiste à : (1) recenser les patterns à mettre en œuvre pour résoudre le problème ou bien les patterns qui ont été utilisés pour résoudre des problèmes similaires, (2) Choisir le pattern adéquat pour résoudre le problème. Cependant, le mauvais choix du pattern peut impliquer des conséquences indésirables. (3) imitation du patron dans un contexte donné. L'imitation est la plus simple opération utilisée dans l'approche orientée objet (Coad, 1995). L'imitation est la duplication et l'adaptation de solution pour un contexte donné. Elle est définit comme suit (Rieu, Giraudin, & Saint Marcel, 1999)

| |
|---|
| Imiter un patron = dupliquer + adapter la solution du patron<br>Adapter un duplicata de patron = (renommer \| redéfinir \| ajouter \| supprimer)* des propriétés statiques (attributs), dynamiques (méthodes) et relationnelles (associations) |

## 8 Conclusion

Les patrons de conception d'ontologie sont reconnus comme étant très utiles pour améliorer la qualité et la modularité et faciliter la réutilisation et l'évolution. Dans ce chapitre, nous avons proposé un patron de conception d'ontologie multipoints de vue(MV2P) dont la proposition assure :

- L'indépendance de tout langage d'implémentation d'ontologie
- La validation formelle du patron

Notre patron de conception d'ontologie d'architecture décrit en UML permet de rendre compte des différents points de vue dans un modèle ontologique unique. Nous avons aussi proposé un ensemble de patrons de conception de correspondance qui nous permettent de lier les différentes vues du patron de conception. Nous avons porté une attention particulière à la validation formelle de ce patron par l'extension de l'analyse formelle de concepts « structure de patrons » que nous avons présentée en détail.

La réutilisation du patron MV2P peut être appliquée dans le domaine de l'intégration des données. En effet, l'intégration des sources de données ontologiques est un des problèmes épineux très étudié ces dernières années. Dans le chapitre prochain, nous verrons l'impact de la réutilisation du patron de conception multipoints de vue dans un système d'intégration sémantique des données.

Chapitre 5

# Une Ontologie Multipoints de Vue pour Intégrer les Bases de Données à Base Ontologique

## Sommaire



## 1 Introduction

Une approche d'intégration hybride (*cf* Chapitre 1 section 2) à base d'ontologie se résume principalement en : (1) alignement des ontologies locales (2) construction du schéma global et (3) construction d'un ensemble de mapping pour accéder aux données locales. La plupart des recherches dans ce domaine sont centrées sur la découverte des correspondances (alignement) (Shvaiko & Euzenat, 2005) (Kasri & Benchikha, 2011) (Otero-Cerdeira, Rodriguez-Martinez, & Gomez-Rodriguez, 2015) (Wang, Bhagavatula, Neumann, Wilhelm, & Ammar, 2018). Généralement, les méthodes d'alignement s'intéressent seulement à la relation d'équivalence (=) entre les concepts. Les correspondances résultantes sont utilisées pour intégrer les ontologies locales dans une ontologie globale par la fusion des concepts équivalents. Cependant, la plupart de ces méthodes exécutent le processus de fusion de manière interactive à l'aide d'un expert. Un ensemble de mapping entre le schéma global (ontologie intégrée

globale) et les ontologies locales est construit manuellement pour offrir un accès aux données locales. Certains systèmes offrent un accès par vue en utilisant des requêtes comme les vues des bases de données (Krisnadh et al., 2015).

D'autres approches étudient l'intégration des ontologies en utilisant les méthodes formelles comme l'analyse formelle des concepts (AFC) pour automatiser complètement ou réduire l'intervention des êtres humains (Stumme & Maedche, 2001). D'autres s'intéressent à l'utilisation des nouvelles technologies comme les patrons de conception d'ontologie de contenu pour fournir une perspective unifiée sur les données tout en permettant encore un degré assez important de sémantique (Krisnadh et al., 2015) (*cf* Chapitre 1). En comparaison avec une ontologie globale monolithique, un patron de contenu peut être vu comme un extrait qui ne définit qu'une notion particulière sans complexité excessive que l'ontologie globale peut entraîner.

Dans ce cadre de recherche, nous enregistrons notre contribution qui consiste à utiliser le parton de conception multipoints de vue proposé dans le chapitre 4 et l'analyse relationnelle des concepts pour intégrer des sources de données à base ontologique. Le choix de l'analyse relationnelle de concepts pour intégrer les ontologies s'appuie sur le fait qu'elle présente un cadre formel largement utilisé dans de nombreux domaines, et les résultats témoignent de son apport dans la construction et la restructuration des ontologies (Bendaoud, Toussaint, & Napoli, 2010) (Rouane-Hacene, Fennouh, Nkambou, & Valtchev, 2010).

Le chapitre est organisé en six sections. La section 2 motive l'intégration de la notion de point de vue dans un schéma global d'intégration de données ainsi que l'utilisation des patrons de conception d'ontologie pour sa construction. Dans la section 3, nous présentons un aperçu général de l'architecture du système d'intégration et les différentes phases du processus d'intégration proposé. Ces dernières seront détaillées dans les sections 4 et 5. La section 6 conclut ce chapitre.

## 2 Motivation et objectif

Nous proposons dans cette section de motiver l'intégration de la notion de point de vue dans le schéma global d'un système d'intégration à base d'ontologie en utilisant les patrons de conception d'ontologie. Ensuite, nous présentons nos objectifs et le système d'intégration que nous proposons de réaliser.

### 2.1 Motivation

L'idée d'intégrer la notion de points de vue et d'utiliser les patrons dans un système d'intégration sémantique des données relève principalement de deux constatations que nous développons dans ce qui suit.

▪ **Intégration à base d'ontologie v.s Intégration à base d'ontologie multipoints de vue :** Actuellement, le problème d'hétérogénéité des sources de données dans un système d'intégration est traité par l'intégration à base ontologique. L'ontologie présente le cœur de ce type d'intégration. En effet, grâce à son caractère formel, explicite et consensuel les ingénieurs

peuvent décrire explicitement la sémantique des concepts de domaine ce qui permet de réduire l'hétérogénéité sémantique entre les différentes sources intégrées. Les ontologies permettent non seulement la description explicite de différentes sources mais elles peuvent être aussi utilisées comme modèle global pour exprimer les requêtes des utilisateurs dans un environnement distribué. En effet, elles sont utilisées par le médiateur pour enrichir les requêtes sémantiquement et pour déterminer de quelle façon ces dernières doivent être effectuées (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2017) (Kontchakov, Rezk, Rodriguez-Muro, Xiao, & Zakharyaschev, 2014). Cette utilisation couvre l'accès et l'exploitation des données. Cependant, les utilisateurs ont besoin non seulement d'accéder aux données mais en plus de n'accéder qu'aux parties qui les intéressent. En effet, il est souhaitable et même recommandé que le système puisse offrir un accès restreint et sélectif aux données selon les points de vue des utilisateurs. C'est le rôle de l'utilisation d'une ontologie multipoints de vue.

▪ **Intégration à base d'ontologie v.s Intégration sémantique par patron de conception d'ontologie :** Dans les systèmes d'intégration à base d'ontologie, les solutions apportées pour identifier l'ontologie du schéma global sont différentes. Certains systèmes utilisent une ontologie générique (universelle) existante. D'autres fusionnent les différentes sources dans une ontologie globale. Pour des besoins d'intégration comme l'exploitation facile, l'évolution des sources et la maintenance, cette ontologie doit être de qualité, modulaire et réutilisable. Aujourd'hui, il est indéniable que les patrons de conception d'ontologie soient largement acceptés comme des modèles de solution pour guider le développement des ontologies de qualité. Pour l'intégration des données les patrons de conception d'ontologie offrent un cadre rigide pour décrire le vocabulaire commun des sources des données hétérogènes. Ce vocabulaire aide, par la suite, à la communication entre les différents acteurs du système d'intégration (les fournisseurs de données, les ingénieurs d'ontologie, les utilisateurs du système). Les patrons rendent aussi les ontologies plus modulaires puisque chaque patron modélise une seule notion d'un domaine donné. Une ontologie peut combiner plusieurs patrons de conception pour couvrir un domaine. Cette modularisation (1) rend la réutilisation des parties d'ontologie possible, (2) diminue le niveau de complexité de l'ontologie.

Sur la base des deux constatations précédentes, nous proposons une nouvelle solution pour l'intégration des données à base ontologique palliant les limites des approches existantes et répondant à de nouvelles exigences. La figure 5.1 présente une vue générale de l'approche d'intégration hybride proposée où :

- Les ontologies locales sont des bases de données à base ontologique ;
- Le schéma global de l'intégration est l'ontologie multipoint de vue créée par le patron multipoints de vue et l'analyse relationnelle de concept.

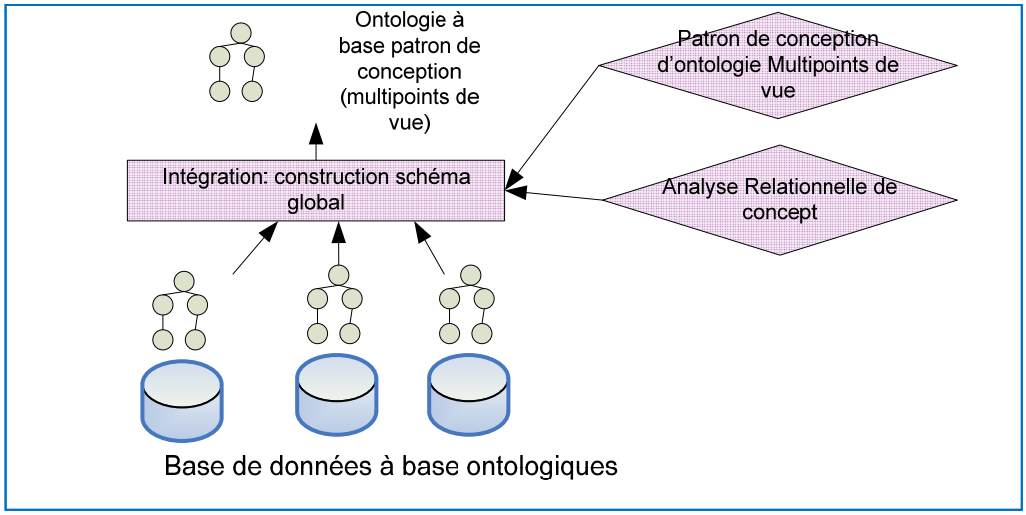



**Figure 5.1** Intégration à base de patron de conception (patron d'ontologie multipoints de vue)

Les principes fondamentaux de notre approche sont :

1. Le schéma global est une ontologie multipoints de vue qui joue le rôle d'une interface unique de communication
2. L'ontologie globale est une ontologie à base de patron de conception évolutif, modulaire et réutilisable.
3. La construction de la hiérarchie de l'ontologie multipoints est faite formellement par l'analyse relationnelle en réduisant les erreurs de conception au niveau de la structure et aussi en assurant la cohérence des données.
4. Les différentes ontologies locales sont restructurées selon le patron MV2P. Cette restructuration facilite leur intégration en offrant une harmonisation virtuelle entre elles.
5. L'accès aux données se fait par point de vue en réduisant la complexité de l'ontologie globale.

## 2.2 Système d'intégration hybride à base de patron de conception d'ontologie

La démarche pour intégrer les ontologies de base de données à base ontologique consiste, dans un premier temps, à chercher un moyeu pour fournir un accès aux données par points vue aux utilisateurs. Dans un deuxième temps, proposer une approche d'intégration semi-automatique basée sur des méthodes formelle. Notons que dans le cadre de notre travail, nous visons l'intégration des données à base d'ontologie en OWL. Celles-ci sont stockées dans un schéma relationnel comme (SDB de Jena (Wilkinson, Sayers, Kuno, & Reynolds, 2003), SOR (Lu et al., 2007)).

Notre approche passe par deux grandes étapes ; la pré-intégration et l'intégration (voir la figure 5.2).

1. L'étape de pré-intégration : La pré-intégration inclut toutes les étapes préliminaires qui ont pour objectif de préparer les données ontologiques à intégrer. Elle consiste à (1)

spécifier des vues par l'ingénieur d'ontologie, (2) imiter le patron pour être utilisé comme un modèle d'ontologie et (3) restructurer les ontologies en entrée (ontologies locales) pour les rendre homogènes sur le plan schématique selon le patron multipoints de vue (harmonisation virtuelle).

2. L'étape d'intégration : L'intégration inclut (1) l'alignement des ontologies locales pour identifier et fournir toutes les correspondances entre elles et résoudre les conflits entre les correspondances trouvées puis (2) fusionner les ontologies dans une ontologie globale et multipoints de vue.

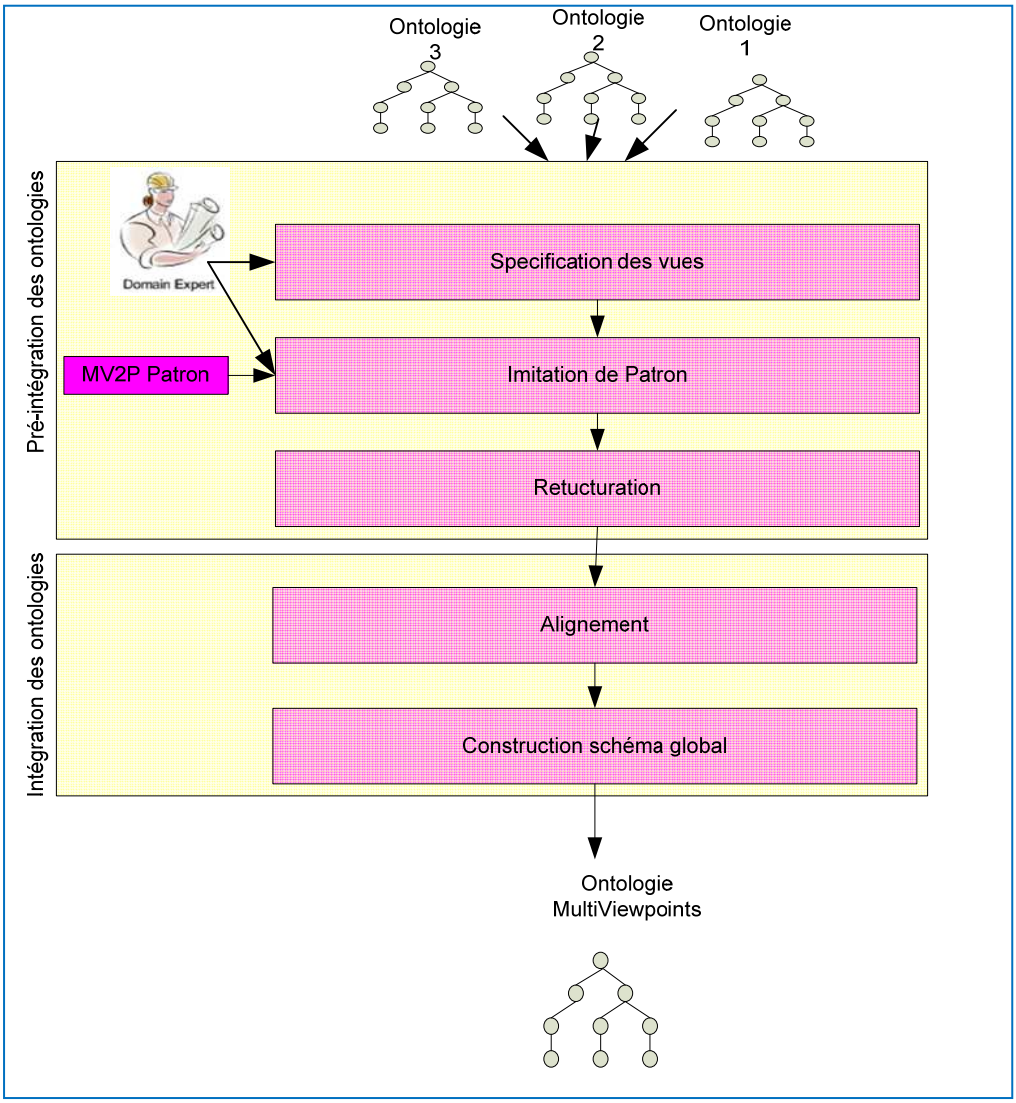



**Figure 5.2** Processus d'intégration des ontologies des sources locales

L'étape de spécification des vues consiste à préciser les différentes vues des concepts ontologiques. Dans l'étape d'imitation, le patron multipoints de vue d'architecture MV2P sera transformé au patron de contenu selon le domaine des ontologies à intégrer (tourisme, biologie, etc.). L'étape de restructuration consiste à restructurer les ontologies locales selon le patron de contenu multipoints de vue où les concepts de chaque ontologie sont éclatés et classifiés dans les différentes hiérarchies du patron multipoints de vue. Cette étape permet la construction d'une couche d'harmonisation virtuelle pour le processus d'intégration. L'étape d'alignement

consiste à faire une confrontation entre les concepts des différentes ontologies restructurées pour l'identification des similarités (les concepts correspondants). La dernière phase est la phase de construction du schéma global (ontologie globale multipoints de vue). Elle consiste à: (1) résoudre les conflits et fusionner les concepts similaires, puis (2) réorganiser les concepts dans les différentes hiérarchies en utilisant l'analyse relationnelle de concepts.

Nous illustrons à travers l'exemple ci- dessous, l'application de l'étape de pré-intégration afin de restructurer les ontologies en entrée en ontologies multipoints de vue.

- **Exemple :** Considérons une des ontologies locales à intégrer soit l'ontologie OnTourism (Lisi, & Esposito, 2014). Nous nous concentrons sur le concept Accommodation et ses sous-concepts pour illustrer les différentes phases d'intégration. La figure 5.3 présente un extrait de cette ontologie. Dans l'ontologie OnTourism, le concept 'Accommodation' est définie comme montré dans le tableau 5.1 :

| Propriété | Type | Domaine | Co-daomaine |
|---|---|---|---|
| **Name** | owl:DatatypeProperty | Accommodation | String |
| **Email** | owl:DatatypeProperty | Accommodation | String |
| **Fax** | owl:DatatypeProperty | Accommodation | String |
| **Phone** | owl:DatatypeProperty | Accommodation | String |
| **numberOfRooms** | owl:DatatypeProperty | Accommodation | int |
| **hasPrice** | owl:DatatypeProperty | Accommodation | integer |
| **hasRate** | owl:ObjectProperty | Accommodation | Rate |
| **hasLandscape** | owl:ObjectProperty | Accommodation | Landscape |
| **hasAmenety** | owl:ObjectProperty | Accommodation | Aamenety |
| **Provides** | owl:ObjectProperty | Accommodation | Service |

**Tableau 5.1** Le concept 'Accommodation'

Dans l'ontologie OnTourism, les sous concepts du concept Accommodation ont les mêmes propriétés. Dans ce qui suit, nous illustrerons l'étape de pré-intégration sur l'ontologie OnTourism.

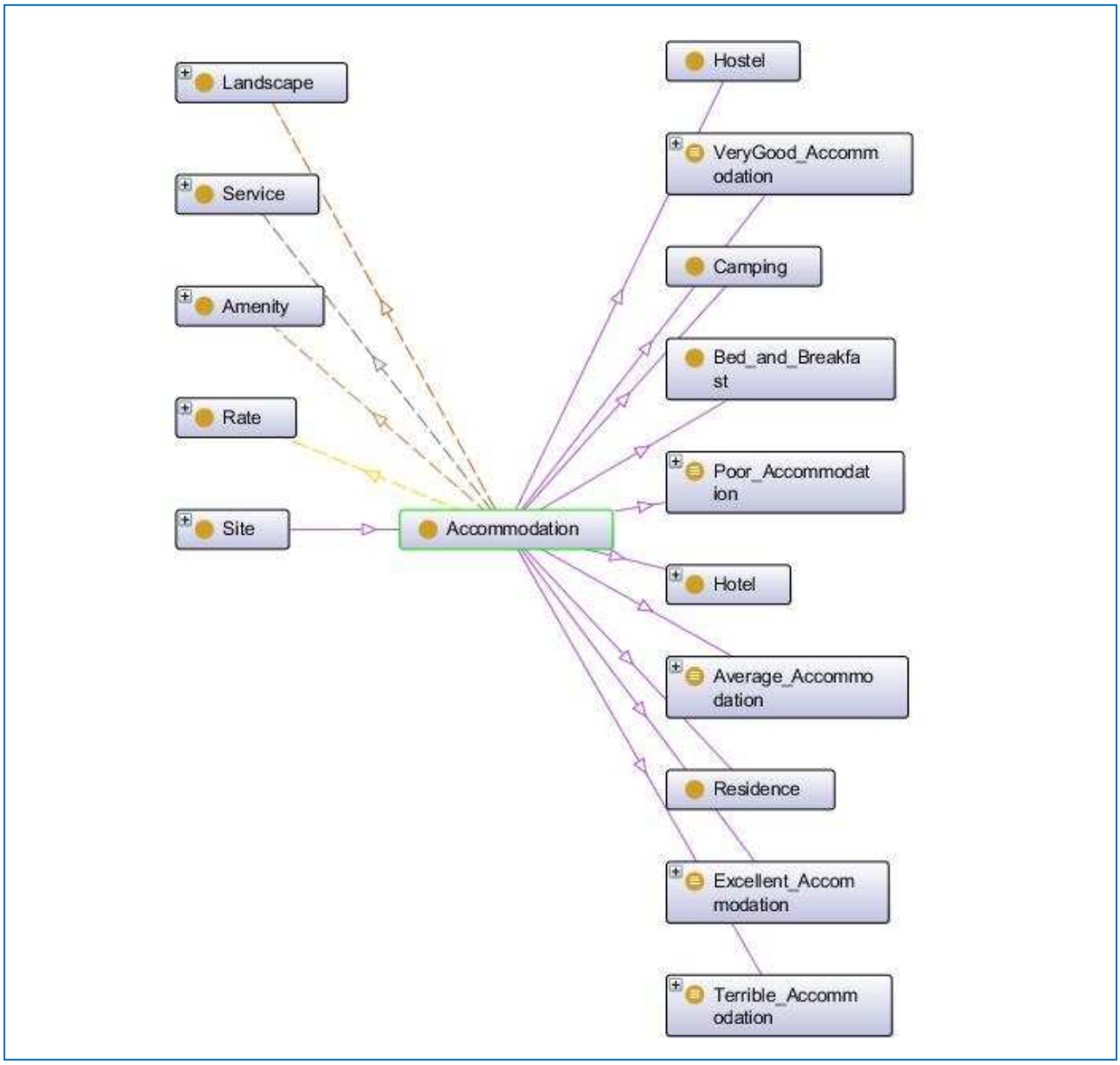



**Figure 5.3** Un extrait de l'ontologie OnTourism par Protégé

## 3 Pré-intégration des ontologies locales

Cette étape consiste à établir des schémas locaux virtuels (hyperschémas) faciles à comparer les uns aux autres. Elle homogénéise la représentation des différentes ontologies (schémas sémantiques locaux des bases de données) à intégrer en les exprimant sous la forme du patron MV2P. Celui-ci joue le rôle d'un modèle d'harmonisation qui permet de diminuer les conflits entre les différents schémas et aussi l'espace de recherche d'alignement. Dans ce qui suit nous présenterons les différentes étapes pour restructurer les ontologies locales selon le patron multipoints de vue.

### 3.1 Phase de spécification des vues de concepts

Réellement, dans le développement d'une ontologie, tous les aspects d'un domaine d'application sont couverts. Les concepts contiennent les propriétés qui caractérisent les objets réels du domaine selon différentes vues. Dans cette étape d'intégration, l'intervention humaine

est obligatoire. L'ingénieur d'ontologie, en fonction des exigences de son système, doit préciser les différentes vues ainsi que les concepts qui doivent être éclatés selon ces vues.

- **Exemple** : Prenons l'exemple de l'ontologie OnTourism. Si l'ingénieur d'ontologie choisit de permettre un accès par point de vue aux accommodations du système alors il choisit le concept Accommodation pour être un concept multipoints de vue et considère par exemple les vues suivantes: « Size_view » contient les propriétés de la vue taille comme "numberOfRooms". « Localization_view » prend en compte l'environnement du concept Accommodation et contient les propriétés : 'hasDistance', 'isLocatedAt', 'hasAmenty' et 'hasLandscape'. « Financial_view » représente les frais de location d'un logement et contient les propriétés haPrice et hasRate. En outre, le concept Accommodation a une vue globale « Global_view » qui peut regrouper toutes les propriétés communes entre les différentes vues et elles sont: 'email', 'fax', 'homepage', 'name' et 'phone.

### 3.2 Phase d'imitation du patron Multipoints de vue

L'imitation du patron de conception dans l'approche orientée objet est la duplication et l'adaptation de solution pour un contexte donné. Elle est définie comme suit (Rieu, Giraudin, & Saint Marcel, 1999).

> ***Imiter un patron*** *= dupliquer + adapter la solution du patron*
> ***Adapter un duplicata de patron*** *= (renommer | redéfinir | ajouter | supprimer)* des propriétés statiques (attributs), dynamiques (méthodes) et relationnelles (associations)*

Dans le contexte de notre travail, l'imitation du patron de conception d'ontologie est définie comme suit (Kasri & Benchikha, 2016):

> Imiter ***un patron*** *= dupliquer* + adapter la solution proposée par le patron au domaine d'application
> Adapter ***un duplicata de patron*** *= (renommer | redéfinir | repositionner| ajouter | supprimer)* des propriétés (attribut/relation) ou des concepts du patron*

- **Exemple** : L'imitation du patron MV2P présenté dans le chapitre 4 donne un patron multipoints de vue de contenu dans le domaine du tourisme (selon l'ontologie OnTourism). Dans l'exemple précédent, l'imitation du patron multipoints de vue fournit un squelette de l'ontologie OnTourism à restructurer. Ce qui consiste en :

1. Adaptation du nom de concept « AbstractView » : le nom de ce concept sera changé en "AccommodationView ». Il permet la variation des vues dans le concept Accommodation.
2. Adaptation du nom des concepts « nameSpace ::vue de domaine » : les noms des espaces de nom seront renommés selon les vues proposées dans l'étape de spécification de vue (Financial, Size, Localization) comme suit :
    - nameSpace:: Vue1 est changé en Accommodation:: Financial

- nameSpace:: Vue2 est changé en Accommodation :: Size
- nameSpace:: Vue3 est changé en Accommodation :: Localization

3. Adaptation du nom de concept « Nameconcept : :GlobalView » : le nom de ce concept sera changé en « Accommodation :: Global »
4. Adaptation du nom de concept « Nameconcpet : :Multiviewpoints » : le nom de ce concept sera changé en « Accommodation :: Multiviewpoints ».
5. Repositionnement du concept Excelent_Accommodation, Poor_Accommodation, Terrible_Accommodation et VeryGood_Accommodation de l'ontologie OnTourism: ces concepts sont migrés vers la hiérarchie de vue locale « Financial » comme des sous concepts de « Accommodation::Financial ». Ces concepts présentent un sous-niveau du concept espace de nom « Accommodation::Financial »
6. Ajout de nouveaux concepts pour former des sous niveaux pour les espaces de nom « Accommodation :: Size » et « Accommodation :: Localization » : nous ajoutons :

   - Big_Accommodation, Small_Accommodation et Averrage_Accommodation comme des sous concepts de Accommodation::Size.
   - Accommodation_in_City, Accommodation_with_Sea, Accommodation_with_Montain et Accommodation_with_River comme des sous concepts de Accommodation::Localization.

Après l'imitation, le squelette de conception obtenu ou le patron multipoints de vue imité (patron de contenu du domaine de tourisme) est présenté en UML comme montré à la figure 5.4.

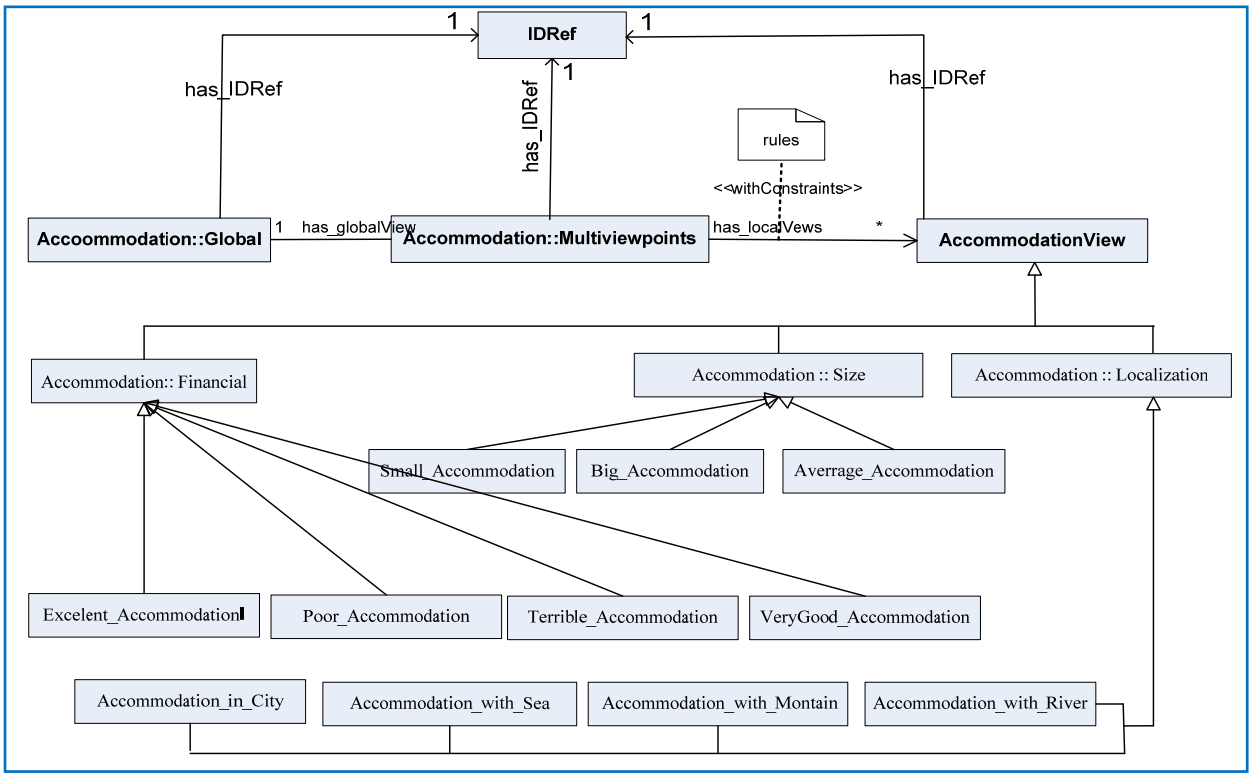



**Figure 5.4** Imitation du patron MV2P

### 3.3 Phase de restructuration

La restructuration de l'ontologie, comme elle a été définie dans (Rouane-Hacene, Fennouh, Nkambou, & Valtchev, 2010), est un changement de la structure actuelle d'une ontologie en une autre structure : nouvelle, meilleure, plus organisée et plus complète. Il est nécessaire pour notre travail de pouvoir organiser les concepts de l'ontologie locale suivant les caractéristiques du patron multipoints de vue (les éléments du patron MV2P)en particulier les relations incluant les passerelles (liens) entre les concepts vue (Kasri & Benchikha, 2016). Pour cela, nous avons choisi d'utiliser l'Analyse Relationnelle de Concepts (voir le chapitre 3).

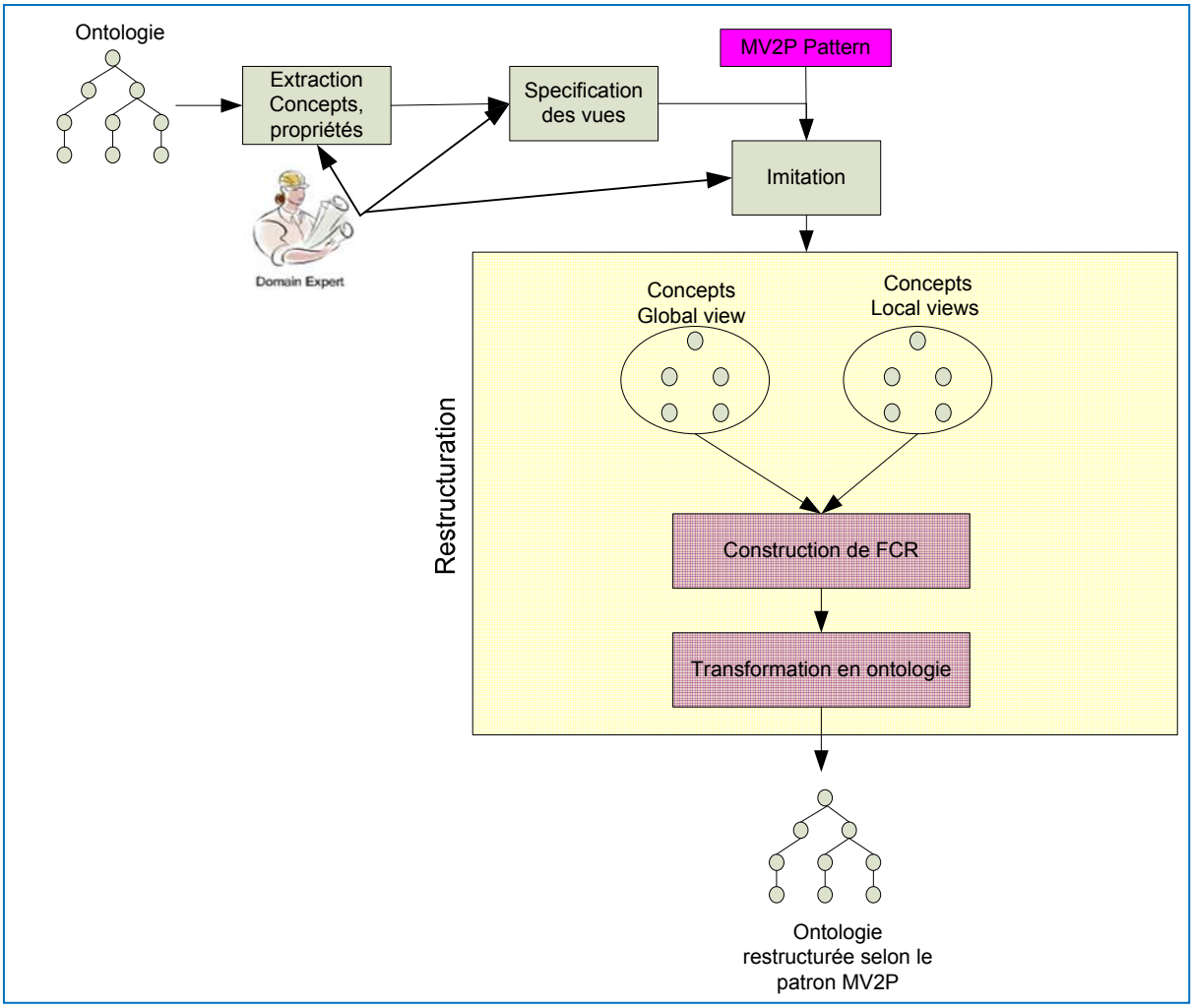



**Figure 5.5** Le processus de restructuration

Pour implémenter l'ARC, la transformation des concepts ontologiques en contextes relationnels est nécessaire. L'approche de restructuration comprend deux étapes: (1) la construction des familles des contextes relationnels à partir des concepts de l'ontologie et le patron MV2P imité, et (2) la transformation des familles des contextes relationnels (FCR) en une l'ontologie.

### 3.3.1 La construction des familles des contextes relationnels (FCR)

Durant cette phase, l'ontologie sera transformée aux FCR. L'analyse relationnelle de concepts ARC permet de décomposer les données en entrée en plusieurs ensembles d'objets qui sont classifiés dans le treillis de concepts et connectés par des relations. Dans cette approche, les objets de contextes sont les éléments du parton de conception d'ontologie multipoints de vue et l'ontologie locale.

Le processus de restructuration commence par (Kasri & Benchikha, 2016): le contexte initial GlobalView, le contexte initial LocalView et le contexte initial MultiViewpoints. L'ontologie à restructurer comporte à la fois les concepts du patron multipoints de vue imités et les concepts de l'ontologie initiale. Dans le patron multipoints de vue, il y a trois types d'hiérarchie de concepts de type : GlobalView, LocalView, et Multiviewpoints. Les concepts restructurés vont être classifiés au niveau de ces hiérarchies.

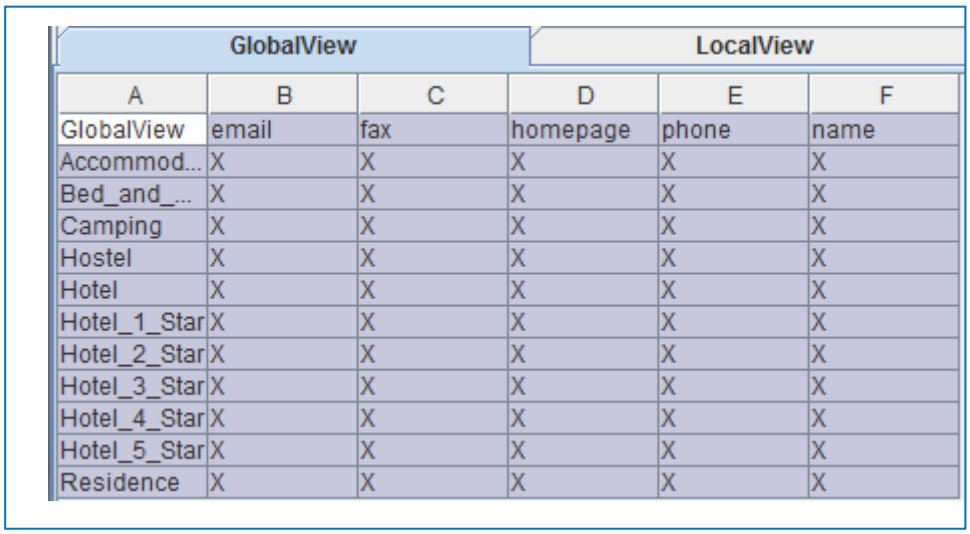
Dans l'ontologie restructurée, les hiérarchies locales sont reliées par des bridges de différents types : inclusion totale, partiale, exclusion…etc. selon les patrons de correspondances proposés dans le chapitre 4. Pour appliquer l'ARC, nous convertissons les concepts du patron et de l'ontologie initiale en une famille de contextes relationnels. Cette conversion se fait de la manière suivante :

1. Chaque hiérarchie dans le patron multipoints de vue sera représentée par un contexte formel.
2. Les concepts ontologiques et les concepts du patron multipoints de vue seront représentés par les objets des contextes.
3. Les attributs des concepts ontologiques seront représentés par les attributs binaires des contextes.
4. Les relations et les passerelles dans l'ontologie seront représentées par une relation binaire entre les contextes.

Dans cette démarche, quel que soit l'ontologie initiale à restructurer, il y'aura toujours les trois contextes formels qui sont définis comme suit :

**Définition 5.1 (Contexte initial GlobalView)** : un contexte initial GlobalView est un contexte formel $\mathcal{K}_g$=($O_g, A_g, I_g$) où $O_g$ est l'ensemble des objets qui modélisent les concepts de la partie globale du patron multipoints de vue et l'ontologie initiale. $A_g$ est l'ensemble des attributs globaux. $I_g$ est la relation d'incidence entre les objets (les concepts globaux) et l'ensemble des attributs globaux.

- **Exemple** : Prenons l'exemple de l'ontologie OnTourism. Nous réduisons le nombre de concepts afin de mieux visualiser le treillis. Le contexte initial GlobalView est représenté dans la figure 5.6 (le contexte sélectionné). Les objets de ce contexte sont: le concept « Accommodation :: Global » du patron MV2P imité et les concepts de l'ontologie initiale. Les attributs formels de ce contexte sont les attributs globaux : name, fax, email, phone, homepage (voir figure 5.7)



GlobalView | LocalView

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| GlobalView | email | fax | homepage | phone | name |
| Accommod... | X | X | X | X | X |
| Bed_and_... | X | X | X | X | X |
| Camping | X | X | X | X | X |
| Hostel | X | X | X | X | X |
| Hotel | X | X | X | X | X |
| Hotel_1_Star | X | X | X | X | X |
| Hotel_2_Star | X | X | X | X | X |
| Hotel_3_Star | X | X | X | X | X |
| Hotel_4_Star | X | X | X | X | X |
| Hotel_5_Star | X | X | X | X | X |
| Residence | X | X | X | X | X |

**Figure 5.6** Le contexte initial GlobalView[1]

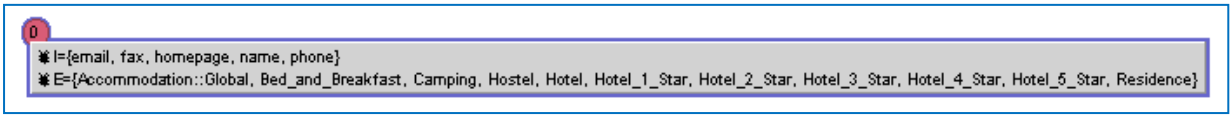
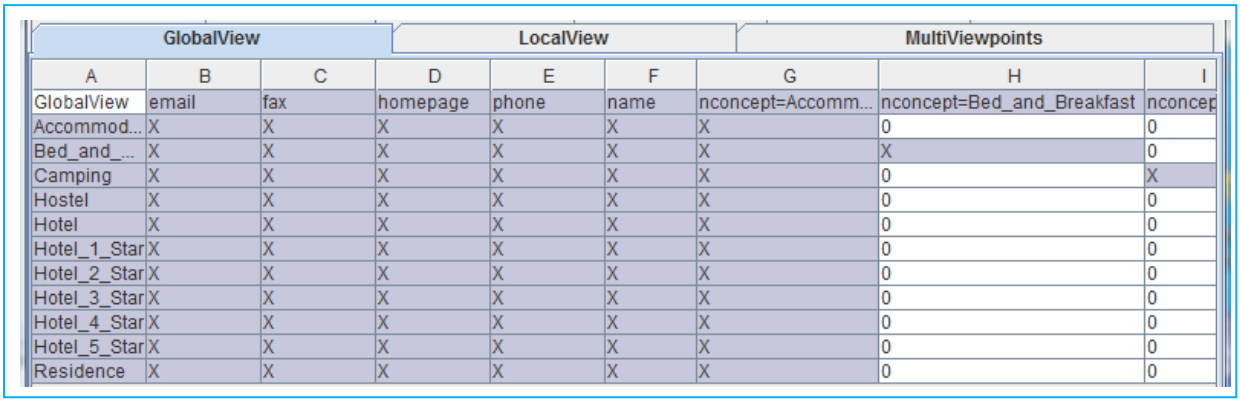



**Figure 5.7** Le treillis de contexte initial GlobalView

Dans la hiérarchie GlobalView tous les concepts ont les mêmes attributs (email, fax, homepage, phone, name). Après l'application de l'ARC, ces concepts seront regroupés dans le même concept formel (voir la figure 5.7). Afin de forcer la préservation des hiérarchies des concepts initiaux dans l'ontologie résultante, nous incluons le nom du concept comme attributs dans les contextes formels (nconcept=Hotel, nconcept=Camping…etc.) comme montré dans la figure 5.8 (le contexte sélectionné).



GlobalView | LocalView | MultiViewpoints

| A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| GlobalView | email | fax | homepage | phone | name | nconcept=Accomm... | nconcept=Bed_and_Breakfast | nconcep |
| Accommod... | X | X | X | X | X | X | 0 | 0 |
| Bed_and_... | X | X | X | X | X | X | X | 0 |
| Camping | X | X | X | X | X | X | 0 | X |
| Hostel | X | X | X | X | X | X | 0 | 0 |
| Hotel | X | X | X | X | X | X | 0 | 0 |
| Hotel_1_Star | X | X | X | X | X | X | 0 | 0 |
| Hotel_2_Star | X | X | X | X | X | X | 0 | 0 |
| Hotel_3_Star | X | X | X | X | X | X | 0 | 0 |
| Hotel_4_Star | X | X | X | X | X | X | 0 | 0 |
| Hotel_5_Star | X | X | X | X | X | X | 0 | 0 |
| Residence | X | X | X | X | X | X | 0 | 0 |

**Figure 5.8** Le contexte GlobalView

Le treillis construit (voir figure 5.9) à partir de ce contexte reste invariable car il n'y a pas une nouvelle relation ajoutée.

[1] Nous utilisons GALICIA pour visualiser les treillis. GALICIA est un outil multiplateforme pour créer, visualizer et sauvgarder le treillis de concepts. il estdisponible en :http://www.iro.umontreal.ca/~galicia/

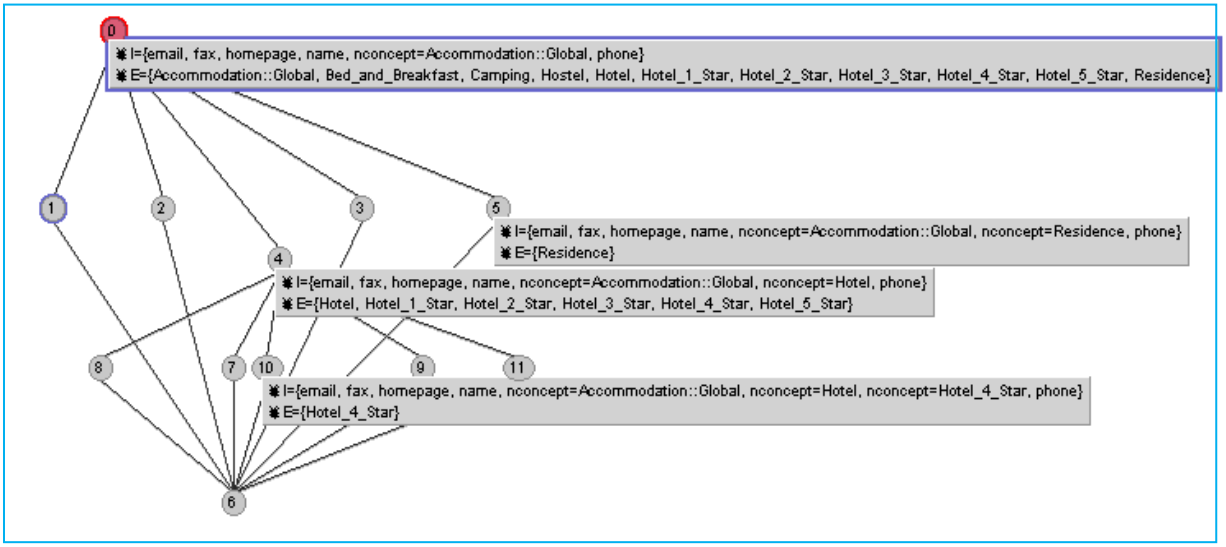



**Figure 5.9** Le treillis de contexte GlobalView

**Définition 5.2 (Contexte initial localView)** : un contexte initial LocalView est un contexte formel $\mathcal{K}_l=(O_l, A_l, I_l)$où $O_l$ est l'ensemble des objets qui modélisent les concepts de la partie locale du patron multipoints de vue et l'ontologie initiale. $A_l$est l'ensemble des attributs locaux. $I_l$est la relation d'incidence entre les objets (les concepts locaux) et l'ensemble des attributs locaux.

- **Exemple** : Le contexte initial LocalView de l'exemple de l'ontologie OnTourism est représenté dans la figure 5.10. Les objets formels de ce contexte sont tous les concepts de la hiérarchie vue locale du patron imité. Les attributs formels sont les attributs des concepts ontologiques inhérents aux vues (numberOfRooms pour la vue size, par exemple) et l'ensemble des attributs ajoutés « nconcept ».

GlobalView | LocalView | MultiViewpoints

| A | B | C | D | E | |
|---|---|---|---|---|---|
| LocalView | numberOfRooms=3 | numberOfRooms<3 | numberOfRooms>3 | nconcept="Big_Accommod... | nco |
| Big_Accommodation | 0 | 0 | X | X | 0 |
| Small_Accommodation | 0 | X | 0 | 0 | X |
| Avrage_Accommodation | X | 0 | 0 | 0 | 0 |
| Accommodation::Size | 0 | 0 | 0 | 0 | 0 |
| Excellent_Accommodation | 0 | 0 | 0 | 0 | 0 |
| Poor_Accommodation | 0 | 0 | 0 | 0 | 0 |
| Terrible_Accommodation | 0 | 0 | 0 | 0 | 0 |
| VeryGood_Accommodation | 0 | 0 | 0 | 0 | 0 |
| Accommodation::Financial | 0 | 0 | 0 | 0 | 0 |
| Accommodation_in_City | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_Sea | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_River | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_Montain | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_Lake | 0 | 0 | 0 | 0 | 0 |
| Accommodation::Localization | 0 | 0 | 0 | 0 | 0 |
| AbstractView | 0 | 0 | 0 | 0 | 0 |

**Figure 5.10** Le contexte initial LocalView

Le treillis initial construit à partir de ce contexte est montré dans la figure 5.11.

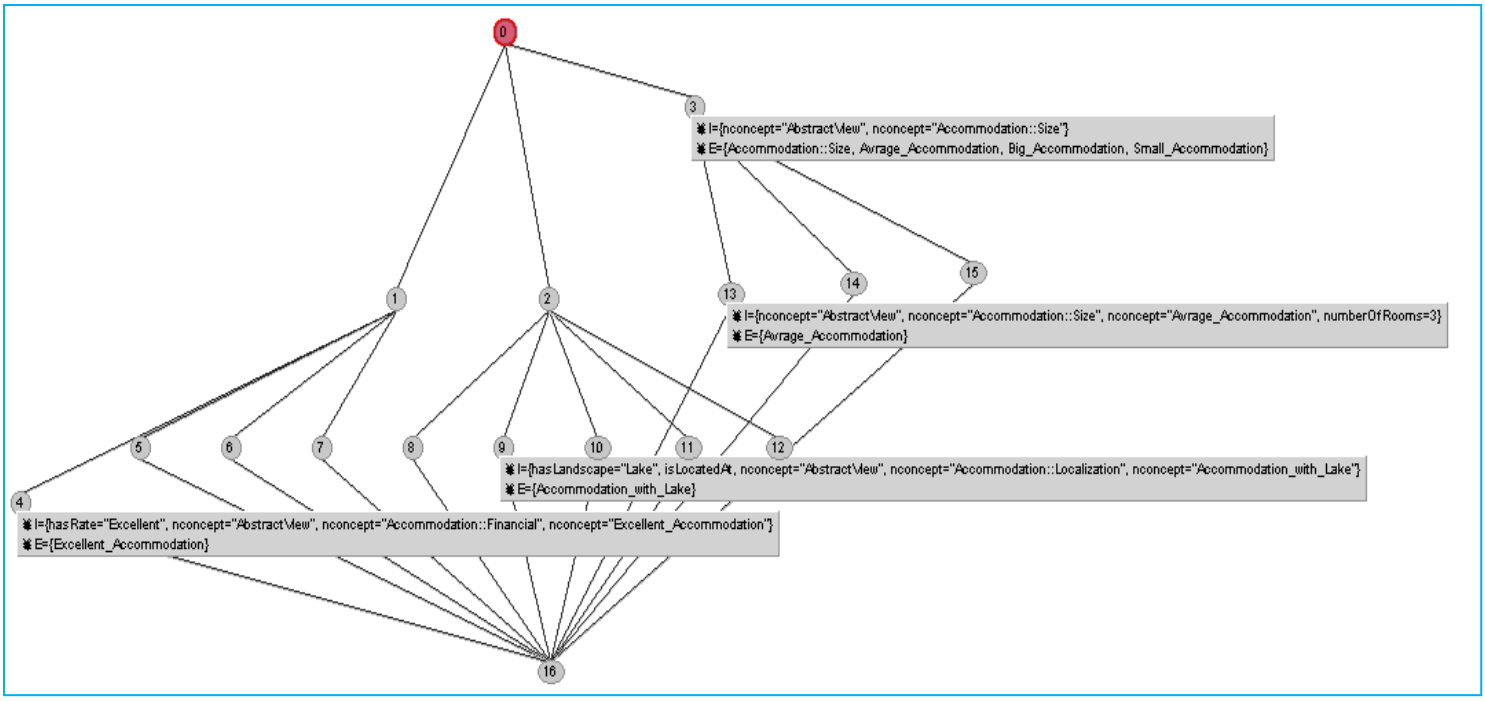



**Figure 5.11** Le treillis de contexte initial LocalView

**Définition 5.3 (Contexte initial MultiViewPoints)** : un contexte initial MultiViewPoints est un contexte formel $\mathcal{K}_m$=($O_m, A_m, I_m$)où $O_m$ est l'ensemble des objets qui modélisent les concepts de la partie multipoints de vue du patron multipoints de vue. Au départ, il contient seulement le concept multipoints de vue. $A_m$est l'ensemble des attributs qui contient seulement le nom du conceptselon la description du concept Multiviewpoints dans le patron MV2P.$I_m$ est une relation d'incidence entre les objets de $O_m$et $A_m$.

- **Exemple** : le contexte initial MultiViewpoints de l'exemple de l'ontologie OnTourism est représenté dans la figure 5.12. Au début, l'objet du contexte initial est le concept « Accommodation::Multiviewpoints ». Le seul attribut de ce concept formel est « nconcept ».

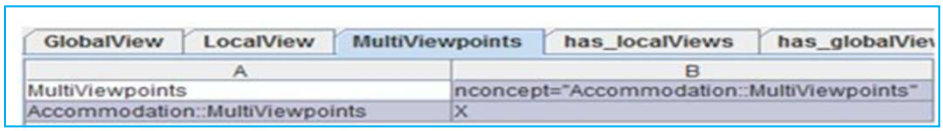



**Figure 5.12** Le contexte MultiViewpoints

Le treillis initial contient un seul concept (voir figure 5.13).


**Figure 5.13** Le treillis de contexte initial MultiViewpoints

En plus des contextes formels initiaux, l'ARC nécessite des contextes relationnels qui forment une famille de contextes relationnels (présenté dans le chapitre 3, Définition 3.6). Dans notre cas, pour chaque relation de l'ontologie à restructurer, un contexte relationnel est créé. Il existe aussi les contextes relationnels pour les relations du patron multipoints de vue (has_globalView, has_localViews). Concernant la relation (has_IDRef), nous ne pouvons pas la représenter comme un contexte relationnel car son domaine est construit à partir de plusieurs concepts qui appartiennent à différents contextes (GlobalView, LocalView, MultiViewPoints). Pour cette raison, la relation doit être ajoutée directement. Les passerelles entre les concepts sont aussi converties aux contextes relationnels. Nous présentons ci-dessous seulement trois passerelles :

- with_IncludeBridge $\subseteq O_l X O_l$
- with_PartialBridge $\subseteq O_l X O_l$
- with_ExcludeBridge $\subseteq O_l X O_l$

  Et les deux relations :
- has_globalView $\subseteq O_{\mathrm{m}} X O_g$
- has_localViews $\subseteq O_{\mathrm{m}} X\, O_l$

- **Exemple** : Dans ce qui suit, nous illustrons comment représenter les passerelles entre les concepts de vue comme des relations inter-contextes. Selon le patron MV2P, la hiérarchie des concepts « LocalView » a plusieurs passerelles qui présentent les lient inter-vues. Nous modélisons ces passerelles dans l'analyse relationnelle de concept par des relations inter-contextes. Chaque relation est transformée en un contexte relationnel (voir figure 5.14):
  - With_IncludeBridge ⊆ objets du contexte initial LocalView $X$ objets du contexte initial LocalView
  - With_PartialeBridge ⊆ objets du contexte initial LocalView $X$ objets du contexte initial LocalView
  - With_ExcludeBridge ⊆ objets du contexte initial LocalView $X$ objets du contexte initial LocalView

has_localViews | has_globalView | with_IncludeBridge | with_PartialBridge | with_ExcludeBridge

GlobalView | LocalView | MultiViewpoints

| A | B | D | C | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|
| with_IncludeBridge | Big... | Avr... | Sm... | Acc... | Excellent_Accommodation | Poor_Accommodation | Terrible_Ac... | VeryGood_ |
| Big_Accommodation | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 |
| Small_Accommodation | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 |
| Avrage_Accommodation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation::Size | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Excellent_Accommodation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Poor_Accommodation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Terrible_Accommodation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| VeryGood_Accommodation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation::Financial | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation_in_City | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_Sea | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 |
| Accommodation_with_River | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_Montain | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation_with_Lake | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Accommodation::Localization | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AbstractView | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 5.14** Relation with_IncludeBridge

La figure 5.14 montre les passerelles d'inclusion du concept « Big_Accommodation » dans le concept « Excellent_Accommodation », « small_Accommodation » dans « Poor_Accommodation » et « Accommodation_with_Sea » dans « Excellent_Accommodation »

Comme présenté dans le chapitre 3, l'ARC injecte les relations comme des nouveaux attributs aux objets grâce à un processus de graduation relationnelle (enrichissement des contextes formels par des relations). Le processus d'analyse relationnel est itératif où les contextes sont modifiés et par conséquent les treillis correspondant par la graduation relationnelle. Il commence par la construction des contextes initiaux. Ensuite, les treillis initiaux sont modifiés en augmentant tous les contextes d'au moins une nouvelle relation notée (*R* :*CI*)où '*R*' est la relation et '*CI*' son co-domaine (enrichissement). Ce processus itératif s'arrête quand un point fixe est atteint (aucun changement dans les contextes). Les nouvelles relations injectées par le mécanisme de graduation comme des nouveaux attributs ont des fortes chances de générer des nouveaux concepts.

**Application :** Le processus d'analyse relationnelle de concepts sera répété (pour ajouter des relations/des passerelles : with_IncludeBridge…etc) sur les familles des contextes de l'exemple précédent jusqu'à ce qu'il atteint un point fixe (aucun changement). La figure 5.15 montre le treillis de « LocalView » obtenu.

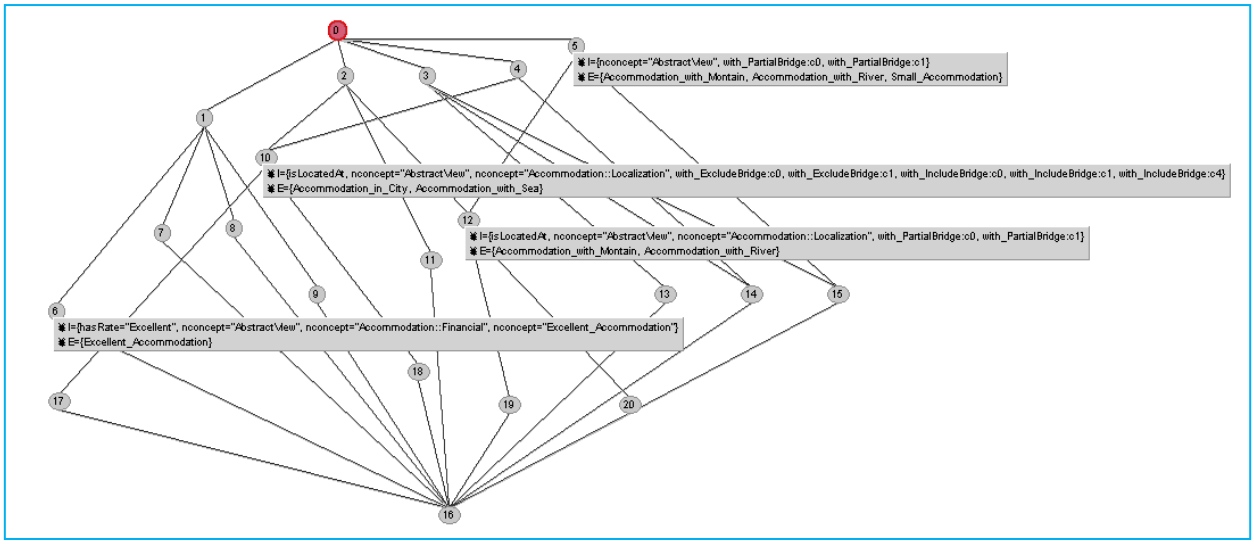



**Figure 5.15** Le treillis de contexte final LocalView

Le tableau 5.2 présente les nouveaux concepts générés par le processus de l'ARC. Nous observons que ces concepts factorisent des concepts déjà existants de même vue comme le concept formel « 10 » (vue Localization) et des concepts de vues différentes comme le concept formel « 4 » (vue Localization et Size)

| Nouveau Concept | Parent | Les concepts factorisés |
|---|---|---|
| Concept formel 4 | Concept formel 0 | Accommodation_in_City \| Accommodation_with_Sea\| Big_Accommodation |
| Concept formel 5 | Concept formel 0 | Accommodation_with_Montain\| Accommodation_with_River \| Small_Accommodation |
| Concept formel 10 | Concept formel 2<br>Concept formel 4 | Accommodation_ with_Montain \| Accommodation__with_River |
| Concept formel 12 | Concept formel 2<br>Concept formel 5<br>Concept formel 0 | Accommodation_in_City \| Accommodation_with_Sea |

**Tableau 5.2** Concepts générés par ARC.

Dans le patron MV2P imité, le concept Accommodation::Multiviewpoints est relié avec le concept AccommodationView par la relation has_localViews et avec Accommodation::Global par la relation has_globalView. Chaque relation sera transformée au contexte relationnel :

- has_localViews⊆ objets de contexte initial MultiViewpoints *X* objets de contexte initial LocalView.
- has_globalView ⊆ objets de contexte initial MultiViewpoints *X* objets de contexte initial GlobalView.

Le processus ARC sera répété pour ajouter les relations précédentes jusqu'à ce qu'&il atteint le point fixe. La figure 5.16 montre le treillis résultant.

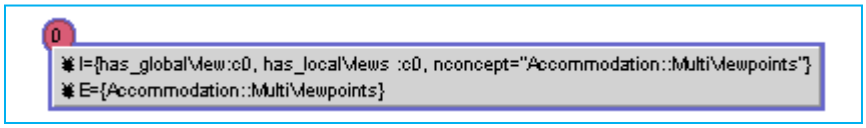



**Figure 5.16** Le treillis de contexte final MultiViewpoints

Ce contexte sera étendu par autres concepts lorsque l'utilisateur utilise l'ontologie pour créer des concepts multipoints de vue.

### 3.3.2 La transformation des familles des contextes relationnels (FCR) en une ontologie

Les Familles des contextes relationnels générées dans l'étape précédente seront transformées en une ontologie multipoints de vue en prenant en compte le patron MV2P imité comme squelette de l'ontologie résultante. Cette transformation consiste à explorer des treilles de concepts afin de choisir ceux qui sont pertinents pour la transformation. Ce processus contient les étapes suivantes :

1. l'ajout des attributs « nconcept='' » (*cf.* Section 4.3.1 : résultent des concepts formels dont les intentions contiennent ces attributs « nconcept='' » et les attributs « nconcept='' » de leurs parents (dans le treillis). Pour cela, nous explorons le treillis, et pour chaque concept formel :
   - nous supprimons les attributs formel « nconcept= '' » de leurs parents,

- nous renommons ce concept avec la valeur de son attribut « nconcept=’’ ».

Par exemple dans le treillis « GlobalView », l’intension « I » du concept formel « 4 » qui représente le concept « Hotel» contient deux attributs « nconcept=’’ » (Accommodation ::Global, Hotel). Pour cela, nous supprimons tous les attributs formels « nconcept=’’ » de leurs parents ; (« nconcept= Accommodation ::Global » de son parent Accommodation ::Global » et nous renommons ce concept formel 4 par « Hotel ».

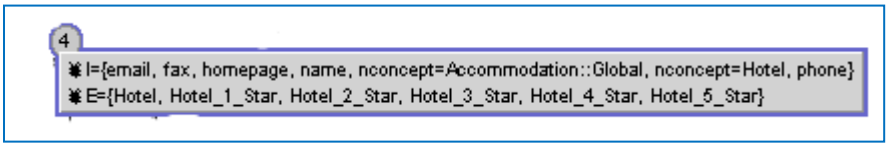



**Figure 5.17** Exemple de concept formel

2. Chaque concept formel du treillis sera transformé en une classe OWL « owl :Class ». par exemple le concept formel « 4 » est transformé en :

```
<owl:Classrdf:ID="Hotel"/>
```

3. Chaque relation inter-contexte sera transformée en propriété de type objet en OWL « owl:ObjectProperty » en explorant le treillis pour déterminer les domaines et co-domaine de la relation. Cette propriété porte le même nom que la relation inter-contexte. Par exemple, dans le treillis « MultiViewPoints », la relations « has_globalView » est ajoutée comme un attribut au concept formel « Accommodation ::Multiviewpoints » avec le co-domaine « *C0* »comme suit :

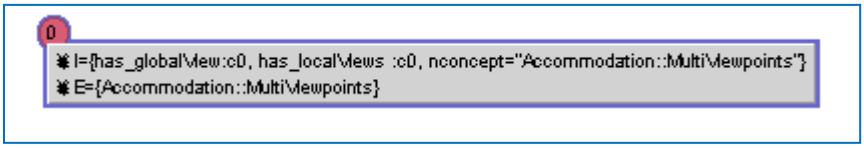



**Figure 5.18** Concept formel multipoints de vue « Accommodation ::Multiviewpoints ».

Le processus prend le patron imité comme un squelette. Celui-ci relie le concept « Accommodation::Multiviewpoints » et le concept « Accommodation:: Global » avec la relation « has_globalView ». C0 présente le concept formel « GlobalView » dans le treillis de concept GlobalView. Ce dernier sera transformé dans l’ontologie comme un co-domaine (rdfs :range ) du concept « Accommodation:: MultiViewpoints ».

```
<owl:ObjectPropertyrdf:ID=" has_globalView">
<rdfs:domainrdf:resource="#Accommodation:: MultiviewPoints  " />
<rdfs:rangerdf:resource="#Accommodation::Global" />
</owl:ObjectProperty>
```

4. L'ordre partiel entre les concepts formels sera transformé en une relation de subsomption en OWL « owl :subClassOf ». Prenons l'exemple du concept formel « 4 » du treillis « GlobalView ». la relation de subsomption est exprimée comme suit :

```
<owl:Classrdf:ID="Hotel">
<rdfs:subClassOfrdf:resource="# Accommodation:: MultiviewPoints"/>
</owl:Class>
```

5. Les attributs des contextes formels seront transformés en propriétés de type de données (ils sont fournis dans les intensions des concepts formels). Prenons l'exemple du concept formel « 4 » du treillis « GlobalView ». l'attribut formel « homepage » du concept formel « Hotel » est exprimé comme suit :

```
<owl:DatatypePropertyrdf:about="#homepage">
<rdfs:domainrdf:resource="#Hotel"/>
<rdf:rangerdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

6. l'ingénieur d'ontologie/l'expert doit intervenir pour ajouter les axiomes de l'ontologie initiale ou pour nommer les nouveaux concepts générés.

Les nouveaux concepts générés et qui factorisent d'autres concepts seront transformés en classe OWL s'ils appartiennent à la même vue. Par exemple, L'ARC génère des nouveaux concepts dans le treillis « LocalView » comme montré dans le tableau 5.2 : Les concepts formels 4 et 5 seront éliminés pare ce qu'ils factorisent des concepts appartenant à des vues différentes (vue size et localization)

| Nouveau Concept | Parent | Les concepts factorisés |
|---|---|---|
| Concept formel 4 | Concept formel 0 | Accommodation_in_City \| Accommodation_with_Sea\| Big_Accommodation |
| Concept formel 5 | Concept formel 0 | Accommodation_with_Montain\| Accommodation_with_River \| Small_Accommodation |

**Tableau 5.3** Concepts formels éliminés

Seuls Les concepts formels 10 et 12 du tableau 5.2 seront transformés en deux classes OWL car ils appartiennent à la même vue (vue Localization).

## 4 Intégration des ontologies locales

Cette étape prend en entrée les ontologies restructurées selon le parton MV2P et applique un processus d'alignement pour trouver les correspondances, résolve les conflits et finalement fusionne les différentes ontologies en une ontologie globale multipoints de vue.

### 4.1 Phase d'alignement

Tout processus d'intégration est précédé par un processus d'alignement des ontologies à intégrer. Dans la littérature, plusieurs définitions de l'alignement peuvent se rencontrer selon la nature de la structure alignée (schémas de base de données, ontologies…). Dans ce travail,

nous adoptons la définition d'alignement (Scharffe, Zamazal, & Fensel, 2013) comme une opération de matching qui prend en entrée O et O' (schéma/ontologie) et retourne un ensemble de mapping A' où chaque élément de A' est défini comme un 5-uplet : <id, e, e', n, R> où :

- id : est l'identifiant du mapping
- e et e' : sont les entités mises en relation (propriété, classes…etc.)
- n : est une mesure de confiance (valeur entre 0 et 1)
- R : est une relation : équivalence (=), plus général (⊇), disjonction (⊥)…..

Le résultat de ce processus est un ensemble de correspondances (mapping). La relation R entre les éléments ontologiques correspondants est utilisée pour préciser les opérations nécessaires dans l'étape d'intégration ainsi que le chemin d'accès aux données. Cependant, la majorité des méthodes disponibles prennent en entrés deux ontologies et déterminent seulement la relation d'équivalence entre leurs concepts.

Dans notre démarche d'alignement, nous utilisons un outil d'alignement AgreementMakerLight (AML) [2] pour assurer l'alignement des éléments des ontologies restructurées (des ontologies multipoints de vue). Cette démarche repose sur le patron MV2P imité comme montré dans la figure 5.19. Le patron multipoints de vue facilite et améliore le processus d'alignement. Il a l'avantage de fournir des paires des hiérarchies des concepts de vue à aligner (ontologie1. Les concepts de vue Finance/ontologie 2. Les concepts de vue Finance).

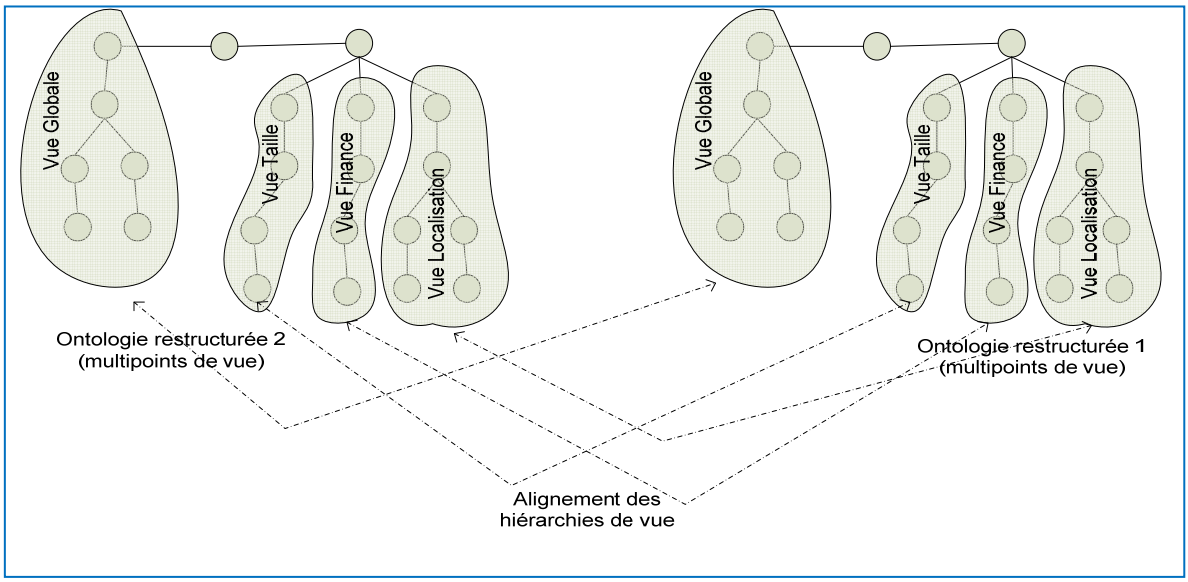



**Figure 5.19** Alignement des hiérarchies de vue

Cette démarche contient trois étapes :

1. Aligner les hiérarchies des concepts de vue globales (concept de type GobalView). Le résultat de cette étape est un ensemble de correspondances en format :<élément GlobalView de l'ontologie1, élément GlobalView de l'ontologie2 de correspondance, mesure de similarité, relation >

[2]https://github.com/AgreementMakerLight

2. Prendre les concepts des correspondances de l'étape précédente et aligner seulement leurs concepts de type LocalView. L'alignement se fait entre les même vues. La sortie de cette étape est un ensemble des correspondances des concepts de type LocalView.
3. L'ingénieur d'ontologie utilise les patrons de correspondance proposés dans le chapitre précédent pour établir des liens entre les correspondances selon le type de passerelle adéquat.

**Application** : Considérons, en plus de l'ontologie OnTourism, deux autres ontologies du domaine de tourisme :Accommodation[3] et QALL-ME (Ou, Pekar, Orasan, Spurk, & Negri, 2008) qui seront présentées de manière plus détaillée dans le chapitre suivant. Le concept Accommodation dans les deux ontologies est représenté dans les tableaux5.4 et 5.5 respectivement.

| Attribut | Domain | Range |
|---|---|---|
| **gr:name** | acco:Accommodation | string |
| **gr:description** | acco:Accommodation | string |
| **schema:geo** | acco:Accommodation | Schama:GeoCoordinates |
| **acco:size** | acco:Accommodation | gr:QuantitativeValue |
| **acco:occupancy** | acco:Accommodation | gr:QuantitativeValue |
| **acco:occupancyAdults** | acco:Accommodation | gr:QuantitativeValue |
| **acco:occupancyMinors** | acco:Accommodation | gr:QuantitativeValue |
| **acco:petsAllowed** | acco:Accommodation | xsd :boolean |
| **acco:partOf** | acco:Accommodation | acco:Accommodation |
| **acco:feature** | acco:Accommodation | acco:AccommodationFeature |
| **acco:includeFeature** | acco:Accommodation | acco:AccommodationFeature |
| **acco:optionalFeature** | acco:Accommodation | acco:AccommodationFeature |

**Tableau 5.4** Concept Accommodation dans l'Ontologie Accommodation

| Attribut | Domain | Range |
|---|---|---|
| **checkinTime** | Accommodation | time |
| **checkoutTime** | Accommodation | time |
| **priceRange** | Accommodation | string |
| **starRating** | Accommodation | int |
| **Comment** | Accommodation | string |
| **Description** | Accommodation | string |
| **specialRequirement** | Accommodation | string |
| **hasContact** | Accommodation | Contact |
| **hasCreditCard** | Accommodation | CreditCard |
| **hasDirectionLocation** | Accommodation | DirectionLocation |
| **hasDistance** | Accommodation | Distance |
| **hasEvent** | Accommodation | Event |
| **hasGPSCoordinate** | Accommodation | GPSCoordinate |
| **hasNearby** | Accommodation | Accommodation |
| **hasPostalAddress** | Accommodation | PostalAddress |
| **hasServiceProvider** | Accommodation | ServiceProvider |
| **hasSiteFacility** | Accommodation | SiteFacility |

**Tableau 5.5** Concept Accommodation dans l'ontologie QALL-ME Tourism

[3] http://ontologies.sti-innsbruck.at/acco/ns.html

Le tableau 5.6 présente le concept accommodation dans les trois ontologies selon différentes vues. L'alignement de ces concepts est effectué par type de hiérarchie de concepts (global/global, size/size…etc.). Dans chaque hiérarchie vue, différentes correspondances résultent de l'alignement du concept Accommodation.

| | **GlobalView** | **LocalView (vuelocalization)** | **LocalView (vue financial)** | **LocalView (vue size)** |
|---|---|---|---|---|
| **Ontologie Accommodation** | gr :name ;gr :description | schema :geo | | acco:size |
| **Ontologie OnTrousim** | email ;fax ;homepage ;phone ;name | isDistanceFor ;hasDistance ;isLocatedAt ;hasAmenty ;hasLandscape | hasRate | numberOfRooms |
| **Ontologie QALL-ME** | comment ;description ;hascontact ; | hasDirectionLocation ;hasDistance ;hasGPSCordinate ;hasNearby | priceRating | |

**Tableau 5.6** Concept Accommodation dans les différentes vues.

### 4.2 Phase de construction de l'ontologie globale (multipoints de vue) par intégration

La construction de l'ontologie globale est la phase finale du processus d'intégration. Elle consiste à créer une ontologie globale multipoints (OntoMuPoV) de vue et l'ensemble de maipping. Elle prend comme entrée l'ensemble des alignements résultants de la phase précédente. Le résultat du processus d'intégration est une ontologie qui contient tous les concepts des ontologies sources sans duplication les informations. Cette ontologie doit permettre un accès par point de vue aux utilisateurs. Nous définissons ici la sémantique du processus d'intégration en prenant compte le patron multipoints de vue. Ce dernier permet d'ajouter la notion de point de vue comme une nouvelle connaissance à l'ontologie résultante. Cette ontologie doit satisfaire les conditions requises suivantes :

- Préserver les concepts : tous les concepts d'entrés sont présents dans l'ontologie OntoMuPoV.
- Préserver la hiérarchie des concepts «est un » : toutes les relations de type « est un » des sources sont présentes dans l'ontologie OntoMuPoV.
- Préserver les propriétés : toutes les propriétés sont présentes dans l'ontologie OntoMuPoV. elles sont dans les concepts locaux ou dans les concepts globaux si elles sont partagées.
- Préserver les relations: toutes les relations sont présentes dans l'ontologie OntoMuPoV. elles sont entre les concepts locaux ou entre les concepts globaux si elles sont partagées.
- Permettre la création des concepts multipoints de vue : les concepts multipoints de vue combinent à la fois des concepts LocalView et GlobalView. Un utilisateur d'ontologie peut construire des concepts avec des propriétés pertinentes seulement à son point de vue.
- Créer la hiérarchie des concepts multipoints de vue : les concepts multipoints de vue peuvent être classifiés dans une hiérarchie avec la relation « est un ».

Le processus d'intégration pour la construction de l'ontologie multipoints de vue se déroule en trois étapes comme montré la figure 5.20 :

1- Résolution de conflits
2- Construction des FCR
3- Transformation de FCR en ontologie

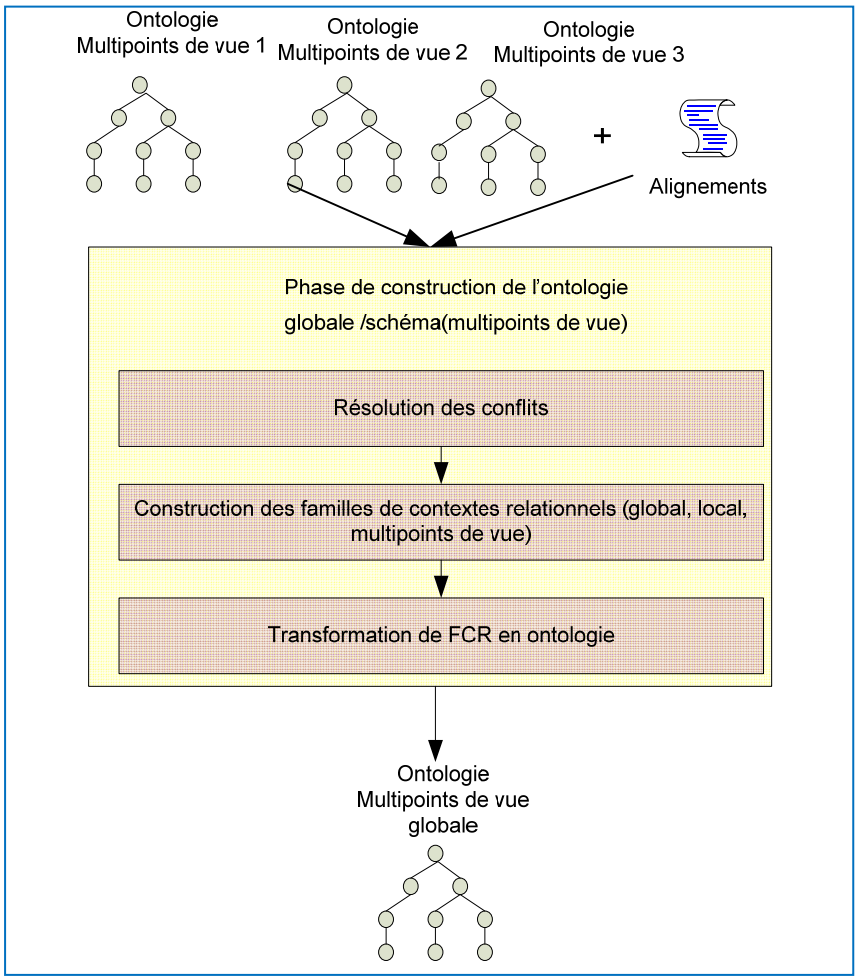



**Figure 5.20** Processus de construction d'ontologie globale multipoints de vue par intégration

Dans ce qui suit nous présenterons ces étapes en détaille.

### 4.2.1 La résolution des conflits

L'hétérogénéité entre les entités ontologiques a été largement étudiée. L'alignement des ontologies vise à établir des correspondances entre deux ontologies et permet de pallier ce problème d'hétérogénéité. Les ingénieurs d'ontologies utilisent leur propre terminologie pour nommer les entités ontologique et donnent des différentes interprétations aux objets du monde réel ce qui peuvent générer des conflits syntaxiques et sémantiques. Nous allons présenter dans ce qui suit quelques conflits :

- Conflit1 : Deux concepts (respectivement attributs, relations) qui représentent la même réalité ont deux noms différents.
- Conflit2 : Un concept correspond à un attribut d'un concept.
- Conflit3 : Un attribut de concept C1 correspond à une relation entre deux concepts, le domaine de la relation correspond au concept C1.
- Conflit4 : Un attribut d'un concept C1 correspond à un attribut d'un concept C2, C1 correspond à C2 mais leurs co-domaines sont différents.

Dans le processus d'intégration, l'étape de résolution des conflits consiste à utiliser les alignements pour générer un ensemble d'opérations sur les entités ontologiques pour préparer les contextes de l'analyse relationnelle des concepts. Ces opérations identifient les concepts et les attributs formels après la résolution des conflits. Supposons que l'ensemble des correspondances suivant (voir tableau 5.7) est obtenu après le déroulement du processus d'alignement. Les opérations suivantes sont effectuées pour intégrer les ontologies avec l'analyse relationnelle des concepts.

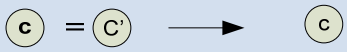
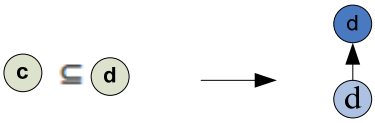
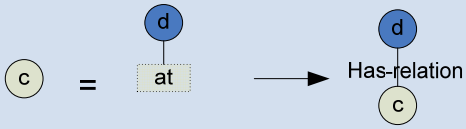
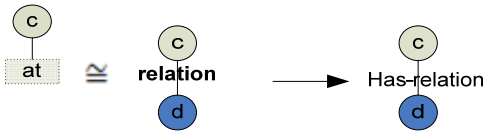
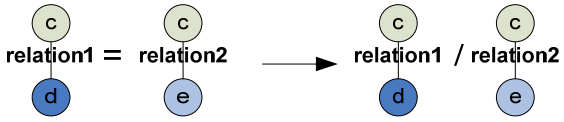
| **Correspondance** | **Description** | **Opérations pour les concepts et les attributs formels** |
|---|---|---|
| **résolution (1)**<br><id, c, c', >=0.7, =><br>  | Les correspondances sont équivalentes | Prendre l'un des concepts comme un concept formel dans le contexte de vue correspondant (contexte GlobalView/LocalView) |
| **résolution (2)**<br><id, c, d, >=0.7, ⊆><br>  | Un concept plus général que l'autre (plus spécifique) et ils sont de la même vue (globale/locale) | Prendre les deux concepts comme des concepts formels en utilisant l'attribut 'nconcept' pour préserver la relation d'héritage. |
| **résolution (3)**<br><id, concept, attribut, >=0.7, =><br>  | Un concept correspond à un attribut de la même vue (globale/locale) | Prendre le domaine de l'attribut comme un concept formel et créer une relation inter-contexte (porte le nom de l'attribut) avec le premier concept. Les deux concepts sont des concepts formels. |
| **résolution (4)**<br><id, attribut, relation, >=0.7, =><br>  | un attribut du premier concept correspond à une relation du deuxième concept et les deux concepts sont équivalents | Prendre l'un de deux concepts comme un concept formel et la relation de deuxième concept comme une relation inter-contexte |
| **résolution (5)**<br><id, attribut1, attribut2, >=0.7, =><br>  | Deux concepts équivalents ont deux attributs équivalents | Choisir l'un de deux attributs comme un attribut de concept formel choisi. |
| **résolution (6)**<br><id, relation1, relation2, >=0.7, =><br>  | Deux concepts équivalenst ont deux relations équivalentes mais avec deux co-domaines différents | Choisir l'une de deux relations avec son domaine et co-domaine |

**Tableau 5.7** Résolution de conflits

**Application :** Le tableau 5.8 montre les correspondances entre les trois ontologies et les différentes résolutions proposées d'après le tableau 5.7.

| Type de résolution | Ontologie Accommodation | Ontologie OnTrousim | Ontologie QALL-ME |
|---|---|---|---|
| **résolution (1)** | Acco :Accommodation | Accommodation | Accommodation |
| **résolution (5)** | gr :name | name | |
| **résolution (5)** | gr:description | | description |
| **résolution (4)** | | Email/fax/phone | hasContact |
| **résolution (6)** | schema :geo | isDistanceFor ,hasDistance , isLocatedAt ,hasAmenty , hasLandscape | hasDirectionLocation ,hasDistance ,hasGPSCordinate ,hasNearby |
| **résolution (4)** | | hasRate | priceRating |
| **résolution (6)** | acco :feature | provides | hasServiceProvider |
| **résolution (1)** | | Bed_and_Breakfast | BedandBreakfast |
| **résolution (1)** | acco :campingSite, acco :campingPitch | Camping | Campsite |
| **résolution (1)** | acco :Hotel, acco :HotelRoom | Hotel | Hotel,Hotel_1_Star, Hotel_2_Stars, Hotel_3_Stars, Hotel_4_Stars, Hotel_5_Stars |
| **résolution (1)** | acco :Resort | | Resort |

**Tableau 5.8** Résolution de conflits des ontologies OnTourism, Accommodation et QALL-ME

Le tableau 5.9 présente un exemple de schéma préféré après l'alignement après la résolution des conflits. Par exemple, le concept Accommodation est présenté par le concept « Acco :Accommodation » dans l'ontologie Accommodation, par le concept «Accommodation » dans l'ontologie OnTourism et par le concept « Accommodation » dans l'ontologie « QALL-ME » . Ce conflit est résolu par « la résolution 1 » du tableau 5.7. Un concept Accommodation est choisi comme concept formel et renommé « Accommodation ::Global » (schéma préféré).

| Ontologie Accomodation | Ontologie OnTrousim | Ontologie QALL-METourism | Schéma préféré |
|---|---|---|---|
| Acco :Accommodation | Accommodation | Accommodation | Accommodation ::Global |
| gr :name | name | | gr :name |
| gr:description | | description | gr :description |
| | email,fax,phone | hasContact | hasContact |
| schema :geo | isDistanceFor ,hasDistance , isLocatedAt ,hasAmenty , hasLandscape | hasDirectionLocation ,hasDistance ,hasGPSCordinate ,hasNearby | hasGPSCordinate ,isLocatedAt, hasDistance |
| | hasRate | priceRating | priceRating |
| acco :feature | provides | hasServiceProvider | hasServiceProvider |
| | Bed_and_Breakfast | BedandBreakfast | BedandBreakfast |
| acco :campingSite, acco :campingPitch | Camping | Campsite | Camping |
| acco :Hotel, acco :HotelRoom | Hotel | Hotel,Hotel_1_Star, Hotel_2_Stars, Hotel_3_Stars, Hotel_4_Stars, Hotel_5_Stars | Hotel |
| acco :Resort | | Resort | Resort |

**Tableau 5.9** Schéma préféré des ontologies OnTourism, Accommodation et QALL-ME

### 4.2.2 La construction des familles de contextes relationnels FCR

Après le traitement des conflits, le processus de restructuration se répète sur l'ensemble des concepts des ontologies à intégrer. Ici, l'entrée du processus est la même que le précédent en modifiant :

- le contexte initial GlobalView est un contexte formel $\mathcal{K}_g = (O_g, A_g, I_g)$ où $O_g$ est l'ensemble des objets qui modélisent les concepts de la partie globale des ontologies obtenues après la résolution de conflit sans redondance. $A_g$ est l'ensemble des attributs globaux. $I_g$ est la relation d'incidence entre les objets (les concepts globaux) et l'ensemble des attributs globaux.
- le contexte initial LocalView est un contexte formel $\mathcal{K}_l = (O_l, A_l, I_l)$où $O_l$ est l'ensemble des objets qui modélisent les concepts de la partie locale des ontologies obtenues après la résolution de conflit sans redondance. $A_l$ est l'ensemble des attributs locaux. $I_l$ est la relation d'incidence entre les objets (les concepts locaux) et l'ensemble des attributs locaux.

### 4.2.3 La transformation de FCR en ontologie

Dans cette étape, les étapes de transformation de FCR en ontologie OWL, présentées dans la section (2.3.2), seront utilisées.

# 5 Conclusion

L'intégration de sources de données hétérogènes dans le contexte du Web est un problème d'actualité. L'apport des ontologies comme schéma sémantique de représentation de ces données est indéniable. En effet, l'ontologie permet l'enrichissement sémantique des données et offrir un vocabulaire commun de partage. Aussi, la majorité des systèmes d'intégration actuels sont à base ontologie. Cependant ces systèmes peuvent être améliorés en utilisant les patrons de conception (Krisnadh et al., 2015).

Dans ce contexte, nous avons proposé une approche d'intégration sémantique à base de patron de conception. Cette approche d'intégration est basée sur une restructuration préalable des ontologies aux ontologies multipoints de vue afin de construire un accès unifié par point de vue aux données. Nous avons présenté une architecture d'intégration hybride où l'ontologie globale est une « ontologie multipoints de vue ».

Notre approche a l'avantage d'utilisée les patrons de conception d'ontologie pour intégrer la notion de point de vue comme connaissance dans l'ontologie/schéma global. Les patrons de conception d'ontologie sont des nouvelles technologies qui permettent la construction des ontologies de qualité et réutilisables. L'étape intermédiaire de restructuration permet la prise en compte conjointe des deux aspects suivants. Tout d'abord, elle permet d'offrir une harmonisation virtuelle des ontologies à intégrer qui facilite l'intégration. Ensuite, elle améliore et facilite l'alignement des concepts ontologiques selon les différentes vues. Notre démarche d'alignement a l'avantage que l'espace de recherche des correspondances est diminué dans l'étape (1) à la taille d'une hiérarchie des concepts de type GlobalView et dans l'étape (2) au nombre de correspondances résultant dans l'étape (1). De plus, plusieurs processus d'alignement sur les différentes hiérarchies des concepts peuvent être lancés en même temps.

Les principes que nous avons développés dans ce chapitre sont appliqués concrètement dans une application d'intégration. Sa présentation fait l'objet du prochain chapitre.

Chapitre 6

# Expérimentation et Implémentation

## Sommaire



## 1 Introduction

Le système OntoMuPoV (**Onto**logie **Mu**lti**Po**ints de **V**ue) est l'application que nous avons développée pendant cette thèse afin de répondre à la nécessité, pour les ingénieurs d'ontologie, de disposer d'outil informatique mettant en œuvre la restructuration et l'intégration des ontologies en introduisant la notion de point de vue. Le système OntoMuPoV est développé dans le langage Java selon le paradigme orienté-objet. Ce chapitre montre différents aspects technologiques de ce système et présente deux scénarios d'utilisation.

Ce chapitre est organisé en six sections. La section 2 présente des métriques d'évaluation de l'apprentissage des ontologies. Nous présentons, dans la section 3, l'architecture générale de notre application OntoMuPoV[1]. Dans la section 4, nous présentons une expérimentation de notre approche de restructuration dans le domaine touristique en utilisant les différentes métriques présentées dans la section 3 pour l'évaluer. Pour mettre en œuvre l'ontologie

[1]On nomme l'application d'intégration par le nom de l'ontologie multipoints de vue

multipoints de vue, nous proposons deux scénarios d'implémentation dans la section 5. La section 6 conclut ce chapitre.

## 2 Métriques d'évaluation

Au cours des dernières années, plusieurs mesures pour l'évaluation de l'apprentissage des ontologies ont été proposées. Dellschaft et Staab (Dellschaft & Staab, 2006) distinguent ces mesures par rapport à la couche d'ontologie évaluée (par exemple, la couche des termes lexicales, et la hiérarchie des concepts) en proposant des mesures taxonomiques : le rappel/ la précision taxonomiques et F-Measure/ F'-Measure. Dans suit, ces mesures seront présentées et appliquées sur une ontologie à évaluer$O_C$ et une ontologie de référence $O_R$ (Figure 6.1).

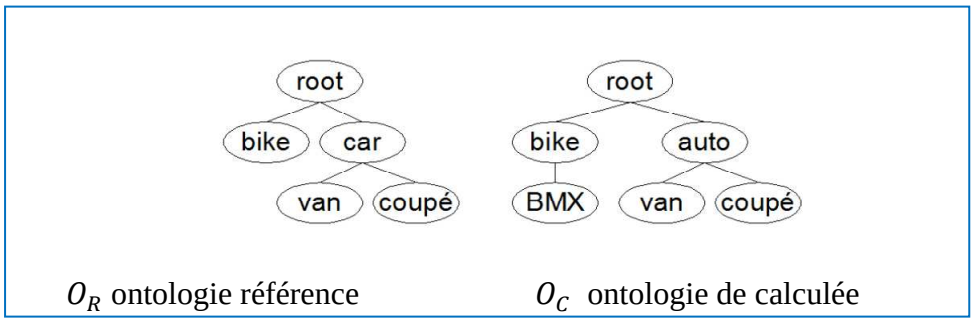



**Figure 6.1** Exemple d'ontologie calculée et ontologie de référence (Dellschaft & Staab, 2006)

### 2.1 Précision et rappel lexicaux

La précision lexicale (LP) et le rappel lexical (LR) sont définis par :

$$LP(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_C|} \qquad LR(O_C, O_R) = \frac{|C_C \cap C_R|}{|C_R|} \quad (1)$$

Ces mesures reflètent à quel point les termes lexicaux appris couvrent le domaine cible. Si on compare $O_{C1}$ et $O_R$ de la figure 6.1 l'un avec l'autre, on obtient :

$$LP\ (O_C, O_R) = \frac{|\{root,bike,van,coupé\}|}{|\{root,bike,BMX,auto,van,coupé\}|} = 4/6 = 0{,}67 \text{ et}$$

$$LR\ (O_C, O_R) = \frac{|\{root,bike,van,coupé\}|}{|\{root,bike,car,van,coupé\}|} = 4/5 = 0{,}8$$

### 2.2 Précision taxonomique locale

La similarité entre deux concepts est calculée en se basant sur la position d'un concept dans la hiérarchie. La précision taxonomique locale $tp_{ce}$ de deux concepts $c1 \in O_C$ et $c2 \in O_{R1}$ est définie par :

$$tp_{ce}(c_1, c_2, O_C, O_R) := \frac{|ce(c_1, O_C) \cap ce(c_2, O_R)|}{|ce(c_1, O_C)|} \quad (2)$$

*« ce »* présente la caractéristique de l'extraction d'un concept de son hiérarchie. Il est un élément très important dans la mesure de la taxonomique locale. Pour cela, plusieurs instanciations alternatives existent comme la cotopie sémantique (*sc*) et la cotopie sémentique commune (*csc*) (Dellschaft & Staab, 2006) présentées dans ce qui suit.

### 2.3 Précision et rappel taxonomique communs

Un concept d'une ontologie peut être caractérisé par sa cotopie sémantique. Cette dernière est définie par :

$$sc(c, \mathcal{O}) := \{c_i | c_i \in \mathcal{C} \wedge (c_i \leq c \vee c \leq c_i)\} \quad (3)$$

Cette définition signifie que la cotopie sémantique d'un concept es t tous ces super ou sous-concepts dans l'ontologie ($c_i \leq c \vee c \leq c_i$). Le tableau suivant présente la *sc* de l'exemple de la figure 6.1.

| $c$ | $sc(c, O_R)$ | $sc(c, O_C)$ |
|---|---|---|
| **root** | {root, bike, car, van, coupé} | {root, bike, BMX, auto, van, coupé} |
| **car** | {root, car, van, coupé} | - |
| **auto** | - | {root, van, coupé} |
| **van** | {root, car, van} | {root, auto, van} |
| **coupé** | {root, car, coupé} | {root, auto, coupé} |
| **bike** | {root, bike} | {root, bike,BMX} |
| **BMX** | - | {root, bike, BMX} |

**Tableau 6.1**La cotopie sémantique de l'exemple de la figure 6.1 [45]

Cependant l'utilisation de la cotopie sémantique pour définir la mesure de la précision taxonomique locale $tpsc$ donne un résultat influencé par la précision lexicale d'$O_C$. Pour surmonter ce problème, Dellschaft et Staab (Dellschaft & Staab, 2006) définissent une cotopie sémantique commune *csc* par :

$$csc(c, \mathcal{O}_1, \mathcal{O}_2) := \{c_i | c_i \in \mathcal{C}_1 \cap \mathcal{C}_2 \wedge (c_i < c \vee c < c_i)\} \quad (4)$$

La cotopie sémantique commune exclut tous les concepts qui ne sont pas inclus dans les deux ontologies en même temps. Le tableau suivant présente la c*sc* de l'exemple de la figure 6.1.

| $C$ | $csc(c, O_R, O_C)$ | $csc(c, O_C, O_R)$ |
|---|---|---|
| **Root** | {bike, van, coupé} | {bike, van, coupé} |
| **Car** | {root, van, coupé} | - |

| | | |
|---|---|---|
| **Auto** | - | {root, van, coupé} |
| **Van** | {root} | {root} |
| **coupé** | {root} | {root} |
| **Bike** | {root} | {root} |
| **BMX** | - | {root, bike} |

**Tableau 6.2** La cotopie sémantique commune de l'exemple de la figure 6.1 (Dellschaft & Staab, 2006)

Donc au lieu de calculer les mesures $TP_{sc}$ et $TR_{sc}$ qui n'offrent pas une évaluation séparée du lexique et l'hiérarchie de concepts. Les mesures $TP_{csc}$ et $TR_{csc}$ sont calculés.

$$TP_{csc}(\mathcal{O}_C, \mathcal{O}_R) := \frac{1}{|\mathcal{C}_C \cap \mathcal{C}_R|} \sum_{c \in \mathcal{C}_C \cap \mathcal{C}_R} tp_{csc}(c, c, \mathcal{O}_C, \mathcal{O}_R) \quad (4)$$

$$TR_{csc}(\mathcal{O}_C, \mathcal{O}_R) := TP_{csc}(\mathcal{O}_R, \mathcal{O}_C) \quad (5)$$

**2.4 F-mesure/F'-mesure**

La précision et le rappel taxonomique peuvent être combinés dans une même mesure.

$$TF(\mathcal{O}_C, \mathcal{O}_R) := \frac{2 \cdot TP(\mathcal{O}_C, \mathcal{O}_R) \cdot TR(\mathcal{O}_C, \mathcal{O}_R)}{TP(\mathcal{O}_C, \mathcal{O}_R) + TR(\mathcal{O}_C, \mathcal{O}_R)} \quad (6)$$

Si la valeur F-mesure est très élevée cela correspond à une meilleure qualité de la hiérarchie conceptuelle. On peut aussi ajouter une autre mesure $F' - mesure$ qui calcule la moyenne harmonique de $LR$ et $TF$:

$$TF'(\mathcal{O}_C, \mathcal{O}_R) := \frac{2 \cdot LR(\mathcal{O}_C, \mathcal{O}_R) \cdot TF(\mathcal{O}_C, \mathcal{O}_R)}{LR(\mathcal{O}_C, \mathcal{O}_R) + TF(\mathcal{O}_C, \mathcal{O}_R)} \quad (7)$$

# 3 Architecture d'OntoMuPoV

La figure 6.2 présente l'architecture générale de l'outil OntoMuPoV qui comprend cinq modules correspondant aux différentes étapes de notre approche de construction de l'ontologie multipoints de vue. L'implémentation de l'architecture proposée s'est faite en utilisant le langage Java. Java dispose de nombreuses librairies qui nous ont facilités le développement de l'application.

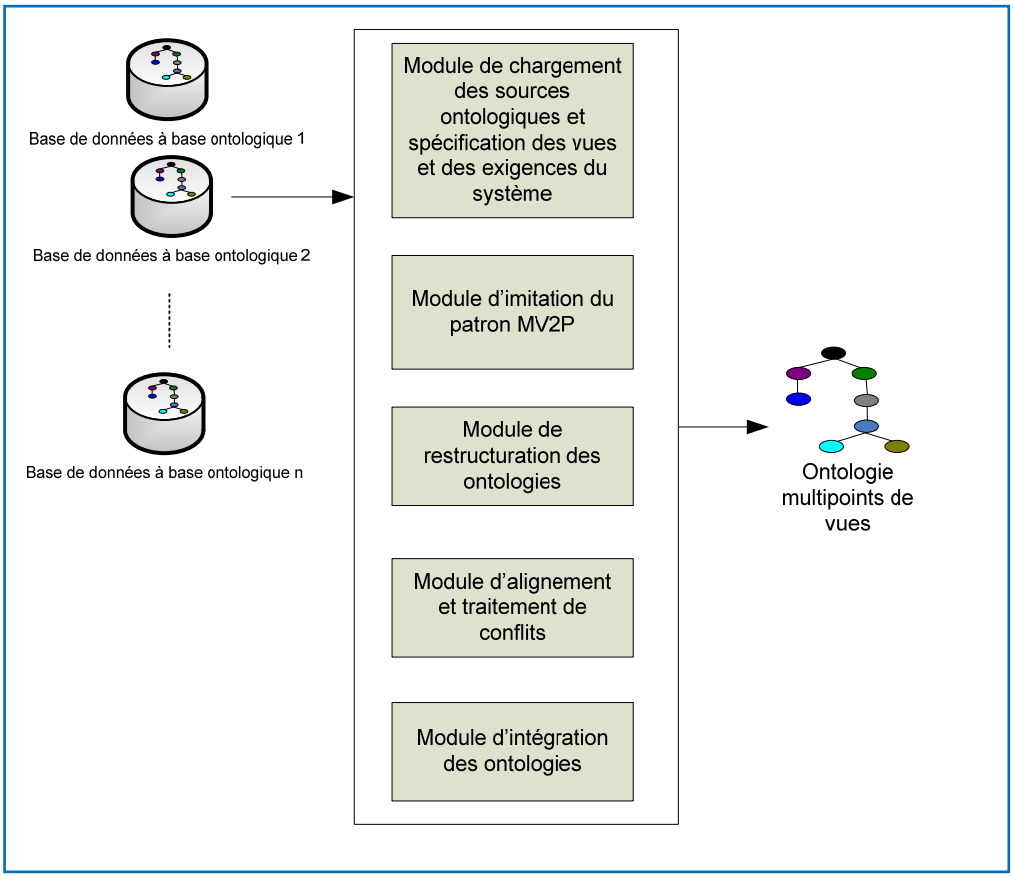



**Figure 6.2** Architecture du système OntoMuPoV

### 3.1 Chargement des sources ontologiques et Spécification des vues

Le module de chargement et spécification permet à l'ingénieur d'ontologie de sélectionner et charger les ontologies à intégrer dans l'ontologie multipoints de vue. Nous avons utilisé l'environnement Jenaproposé par les laboratoires HP (Carroll et al., 2004). C'est une architecture pour stocker et gérer les données ontologiques. Celle-ci fournit un environnement de programmation pour les formalismes d'ontologies RDF, RDFS et OWL en utilisant comme langage de requêtes SPARQL. Jena propose TDB et SDB pour persister les données ontologiques. TDB (tuple Data Base) permet de stocker les données ontologiques en se basant sur un système de fichiers ou la mémoire centrale. SDB (SPARQL Data Base) permet de stocker les données ontologiques en utilisant une base de données comme SGBDR PostgreSQL ou MySQL. Dans OntoMuPoV, nous utilisons la version SDB couplée avec SGBDR MySQL pour stocker les ontologies. Au même temps, ce module sert à identifier et spécifier les vues et les exigences du futur système. La figure 6.3 présente l'implémentation de l'interface de chargement d'OntoMuPoV.

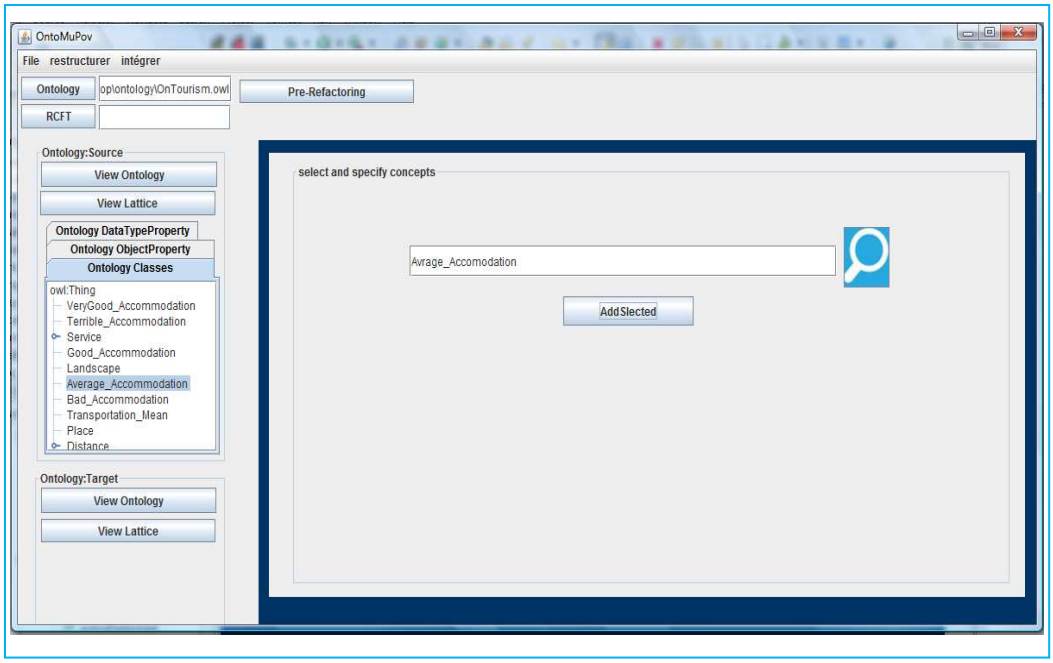



**Figure 6.3** Chargement d'ontologie par OntoMuPoV

### 3.2 Imitation du patron MV2P

Le module d'imitation permet à l'ingénieur d'ontologie de préparer les ontologies à l'étape de restructuration conformément à notre patron multipoints de vue. La figure 6.4 représente l'implémentation de l'interface d'imitation.

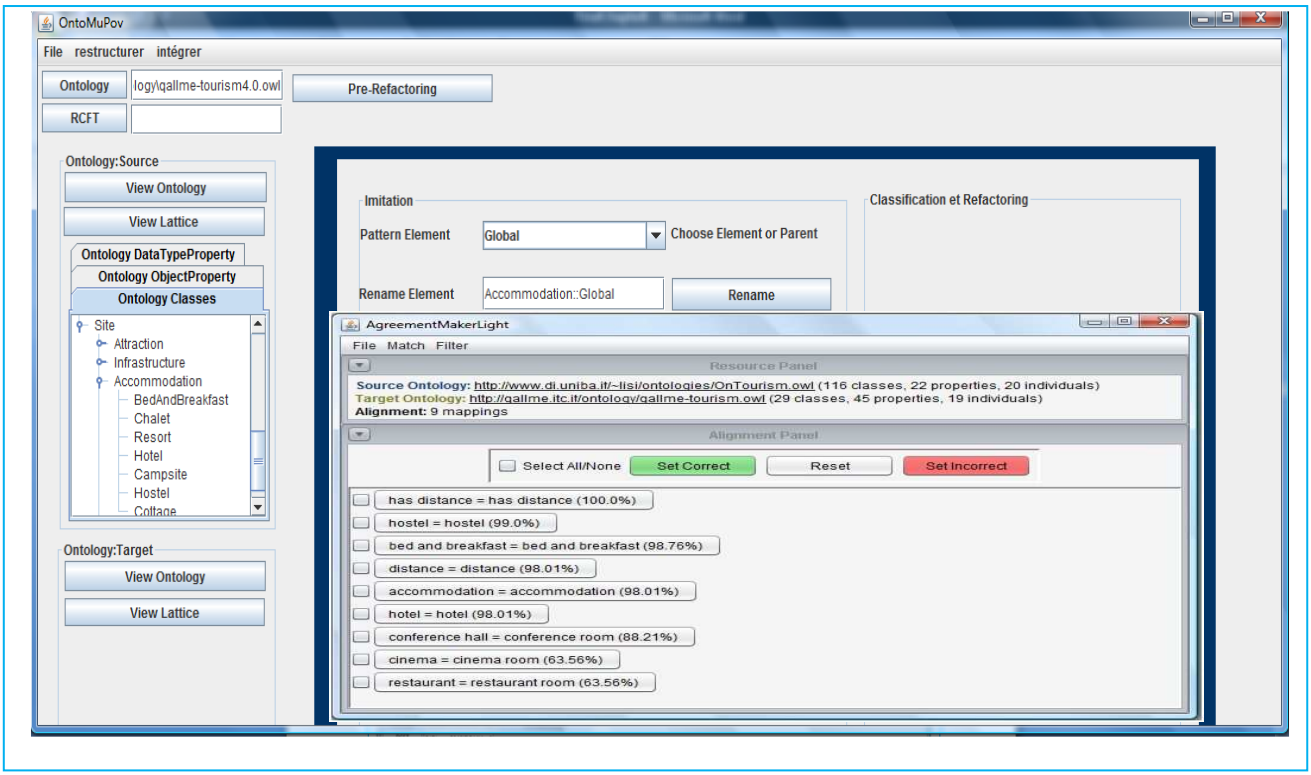



**Figure 6.4** Imitation du patron MV2P

### 3.3 Restructuration des ontologies

Le module de restructuration permet de changer la structure des ontologies basées sur le patron de conception d'ontologie MV2P. Ce module est implémenté en intégrant l'outil RCAexplore[2] pour construire les treillis comme le montre la figure 6.5. RCAExplore offre un éditeur de famille de contextes relationnels, un générateur interactif et un navigateur de treillis de concepts. Il permet l'analyse des concepts relationnels en autorisant le Scaling des contextes exploitant les différents opérateurs.

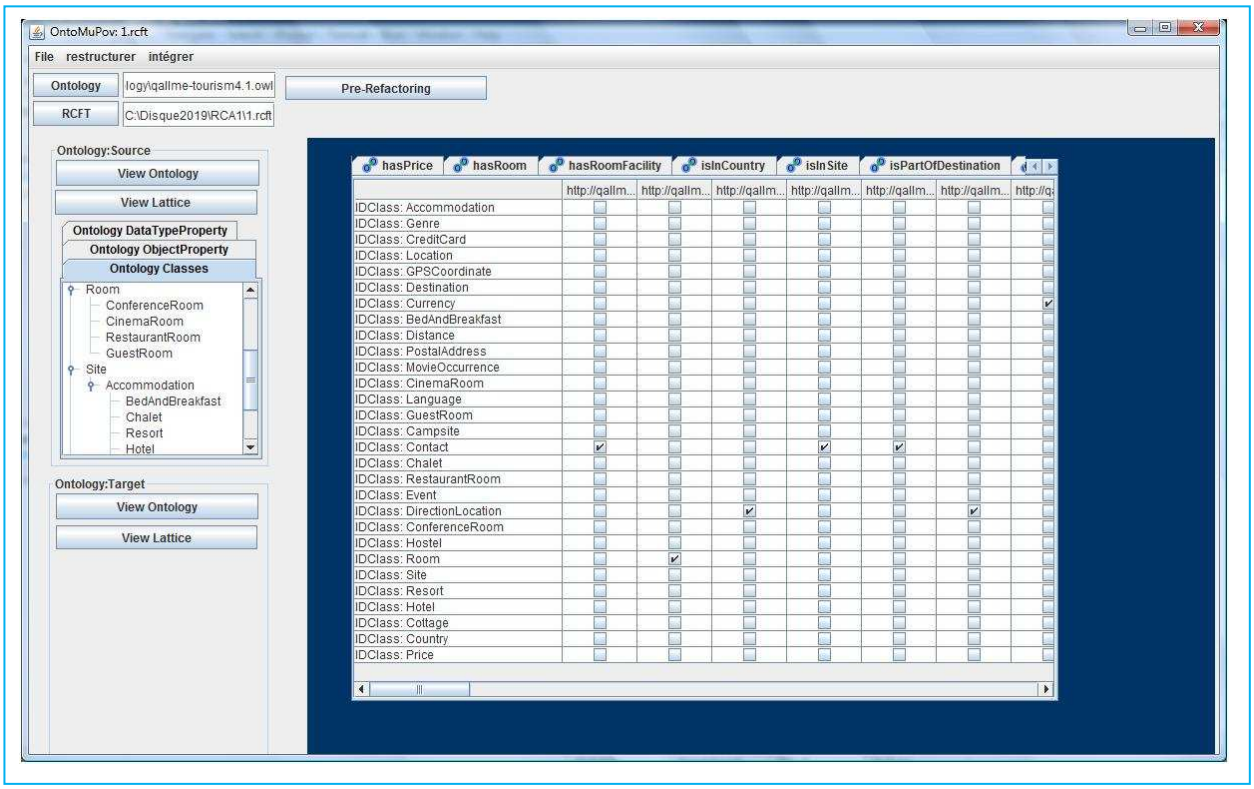



**Figure 6.5** Restructuration des ontologies par RFC

### 3.4 Alignement et traitement des conflits

Le module d'alignement et résolution permet d'aligner les hiérarchies de même vue en intégrant l'outil AgreementMakerLight (AML)[3] (*cf* Figure 6.6). Il permet aussi à l'ingénieur d'ontologie de traiter les conflits entre les correspondants. Le résultat de ce module est un ensemble des correspondants. Selon ces correspondances, des opérations sont appliquées pour résoudre les différents conflits.

[2]http://dolques.free.fr/rcaexplore/
[3]https://github.com/AgreementMakerLight

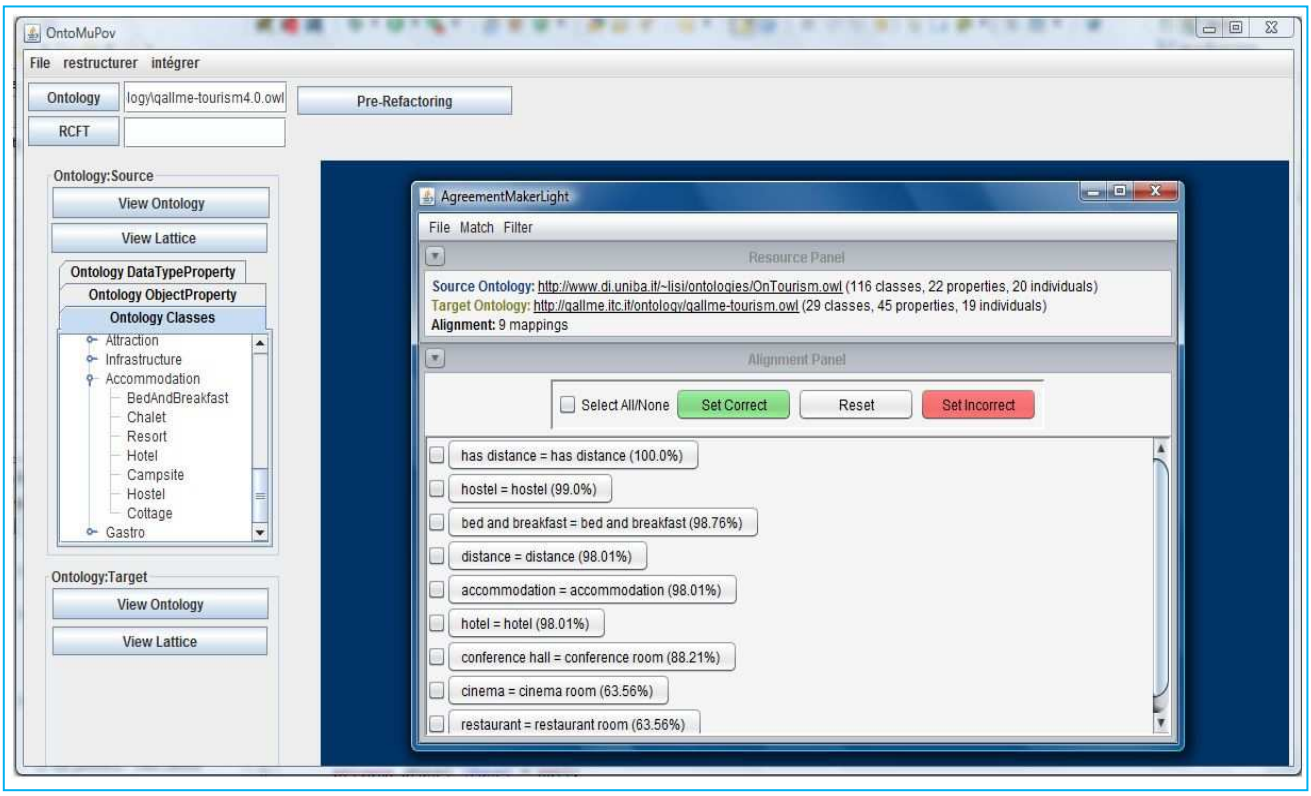



**Figure 6.6** Alignement des ontologies restructurées

### 3.5 Intégration des ontologies restructurées

En se basant sur les correspondances obtenues et résolutions proposées, ce module exécute les opérations de fusionnement pour créer l'ontologie multipoints de vue. La figure 6.7 montre ce module.

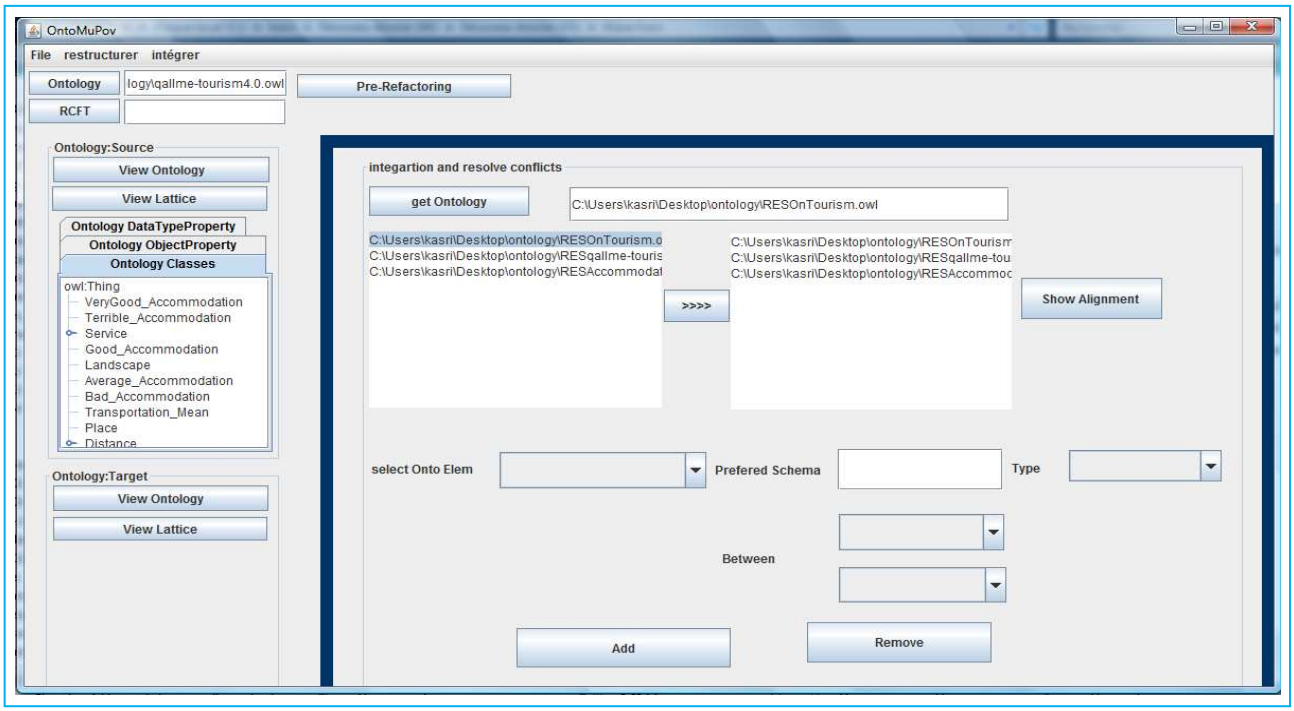



**Figure 6.7** Fusionnement des ontologies

## 4 Expérimentation de l'approche de restructuration

Nous avons choisi d'évaluer notre approche d'intégration sur le domaine touristique (Kasri & Benchikha, 2016) disposant de plusieurs ontologies accessibles librement. La plupart de ces ontologies sont disponibles en OWL. Pour valider une approche de restructuration des ontologies, deux ontologies devraient être utilisées : l'ontologie initiale (c'est l'ontologie à restructurer) et l'ontologie de référence (c'est l'ontologie qui a été désignée comme une ontologie ayant une bonne structuration). Le processus de restructuration sera appliqué sur l'ontologie initiale. L'ontologie restructurée sera par la suite comparée avec l'ontologie de référence en termes de précision et rappel linguistique et taxonomique (*cf* section 2). Cependant, dans notre cas, les ontologies de référence multipoints de vues n'existent pas pour fournir une évaluation précise. Cependant, bien que la nature de l'ontologie initiale et de l'ontologie multipoints de vue soient différentes, l'ontologie initiale sera utilisée comme ontologie initiale et de référence à la fois. Ainsi, en réalité, l'évaluation montrera comment l'ontologie restructurée sera modifiée par rapport à sa structure initiale. Elle montrera également dans quelle mesure ce changement influence le résultat de l'évaluation. Afin de prouver la qualité du processus de restructuration, nous l'avons appliqué sur les trois ontologies du domaine touristique : l'ontologie OnTourism (Lisi, & Esposito, 2014), l'ontologie Accommodation[4] et l'ontologie QALL-ME (Ou, Pekar, Orasan, Spurk, & Negri, 2008).

- **Ontologie OnTourism** (Lisi, & Esposito, 2014) : Elle est utilisée dans Puglia@Service. Ce dernier vise à créer une infrastructure des services innovants pour la région du Puglia, en Italie. Il se base sur des ontologies OWL du domaine de tourisme et voyage/ comme OnTourism. L'ontologie OnTourism contient 120 classes, 9 objectproperties et 13 dataproperties *(cf* figure 6.8 et tableau 6.3).

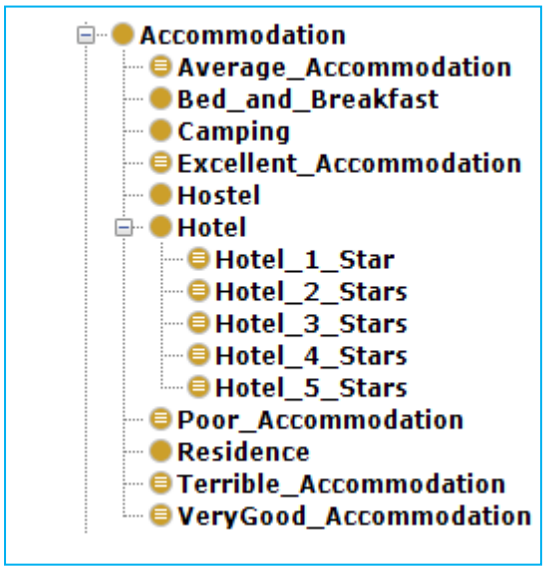



**Figure 6.8** Extrait de l'ontologie OnTourism

[4]http://ontologies.sti-innsbruck.at/acco/ns.html

| Attribut | Domain | Range |
|---|---|---|
| **email** | Accommodation | string |
| **fax** | Accommodation | string |
| **hasAmenity** | Accommodation | Amenity |
| **hasLandscape** | Accommodation | Landscape |
| **hasPrice** | Accommodation | integer |
| **hasRate** | Accommodation | Rate |
| **homepage** | Accommodation | string |
| **numberOfRooms** | Accommodation | int |
| **phone** | Accommodation | string |
| **provides** | Accommodation | Service |
| **hasDistance** | Accommodation | Distance |
| **isDistanceFor** | Accommodation | Site |
| **isLocatedAt** | Accommodation | Place |
| **name** | Accommodation | string |

**Tableau 6.3** Concept Accommodation dans l'ontologie OnTourism

- **Ontologie Accommodation :** C'est une extension de GoodRelations et fournit un vocabulaire supplémentaire pour décrire les chambres d'hôtel, les hôtels, les terrains de camping et autres formes. Elle contient 20 classes, 15 objectproperties et 11 dataproperties (*cf* figure 6.9 et tableau 6.4).

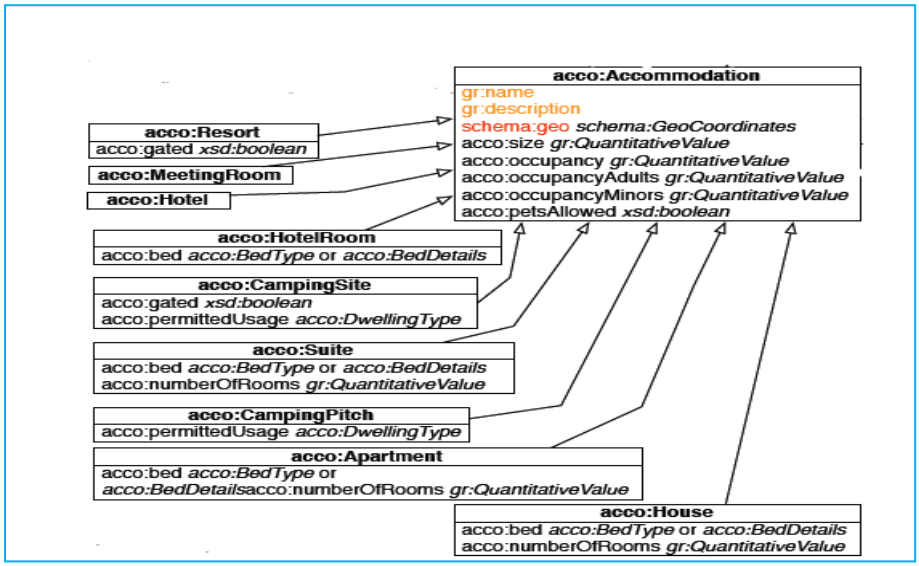



**Figure 6.9** Extrait de l'ontologie Accommodation

| Attribut | Domain | Range |
|---|---|---|
| **gr:name** | acco:Accommodation | string |
| **gr:description** | acco:Accommodation | string |
| **schema:geo** | acco:Accommodation | Schama:GeoCoordinates |
| **acco:size** | acco:Accommodation | gr:QuantitativeValue |
| **acco:occupancy** | acco:Accommodation | gr:QuantitativeValue |
| **acco:occupancyAdults** | acco:Accommodation | gr:QuantitativeValue |
| **acco:occupancyMinors** | acco:Accommodation | gr:QuantitativeValue |
| **acco:petsAllowed** | acco:Accommodation | xsd :boolean |
| **acco:partOf** | acco:Accommodation | acco:Accommodation |
| **acco:feature** | acco:Accommodation | acco:AccommodationFeature |
| **acco:includeFeature** | acco:Accommodation | acco:AccommodationFeature |
| **acco:optionalFeature** | acco:Accommodation | acco:AccommodationFeature |

**Tableau 6.4** Concept Accommodation dans l'ontologie Accommodation

- **Ontologie QALLE-ME (Ou, Pekar, Orasan, Spurk, & Negri, 2008)**: Elle a été développée et appliquée pour répondre aux questions dans le domaine du tourisme. Cette ontologie est codée en OWL DL et couvre plusieurs aspects importants liés au domaine touristique, y compris les destinations touristiques (villes et villages), les sites touristiques (hébergement, attraction et infrastructure), les événements touristiques (cinéma et spectacle) et le transport. Elle comporte 86 classes, 55 objectproperties et 55 dataproperties (*cf* figure 6.10 et tableau 6.5).

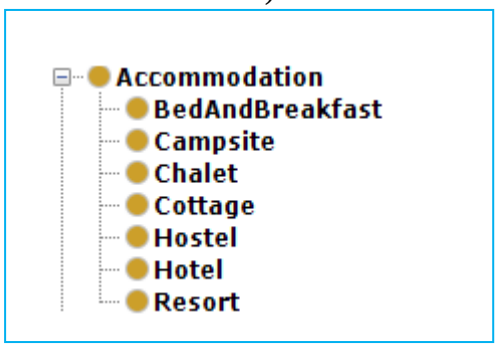



**Figure 6.10** Extrait de l'ontologie QALL-ME

| Attribut | Domain | Range |
|---|---|---|
| **checkinTime** | Accommodation | time |
| **checkoutTime** | Accommodation | time |
| **priceRange** | Accommodation | string |
| **starRating** | Accommodation | int |
| **comment** | Accommodation | string |
| **description** | Accommodation | string |
| **specialRequirement** | Accommodation | string |
| **hasContact** | Accommodation | Contact |
| **hasCreditCard** | Accommodation | CreditCard |
| **hasDirectionLocation** | Accommodation | DirectionLocation |
| **hasDistance** | Accommodation | Distance |
| **hasEvent** | Accommodation | Event |
| **hasGPSCoordinate** | Accommodation | GPSCoordinate |
| **hasNearby** | Accommodation | Accommodation |
| **hasPostalAddress** | Accommodation | PostalAddress |
| **hasServiceProvider** | Accommodation | ServiceProvider |
| **hasSiteFacility** | Accommodation | SiteFacility |

**Tableau 6.5** Concept Accommodation dans l'ontologie QALL-ME

Dans le cadre de cette expérimentation, nous montrons l'application des métriques en calculant la cotopie sémantique (*csc*) et la précision et le rappel taxonomique communs **(** $tp_{csc}$ et $tr_{csc}$ ) (*cf* section 2) pour les concepts de l'ontologie OnTourism (Kasri & Benchikha, 2016). Le tableau 6.6 montre le résultat obtenu pour quelques concepts.

| concept | $Csc(c, O_{OnT}, R_{OnT})$ | $Csc(c, R_{OnT}, O_{OnT},)$ | $tp_{csc}$ | $tr_{csc}$ |
|---|---|---|---|---|
| **Terribl_Accommodation** | {Thing} | { Thing,AbstractView,Accommodation ::Financial } | **1** | **0.33** |
| **veryGood_Accommodation** | { Thing } | {Thing AbstractView,Accommodation ::Financial } | **1** | **0.33** |
| **Excellent_Accommodation** | { Thing } | {Thing AbstractView,Accommodation ::Financial } | **1** | **0.33** |
| **Poor_Accommodation** | { Thing } | {Thing AbstractView,Accommodation ::Financial } | **1** | **0.33** |

| | | | | |
|---|---|---|---|---|
| **Average_Accommodation** | { Thing } | { Thing AbstractView,Accommodation ::Financial } | **1** | **0.33** |
| **Bed_And_Breakfast** | { Thing,Site,Accommodation } | { Thing,Site,Accommodation ::Global} | **0.66** | **0.66** |
| **Camping** | { Thing Site,Accommodation } | { Thing Site,Accommodation ::Global } | **0.66** | **0.66** |
| **Hostel** | { Thing Site,Accommodation } | { Thing Site,Accommodation ::Global } | **0.66** | **0.66** |
| **Hotel** | { Thing Site,Accommodation } | { Thing Site,Accommodation ::Global } | **0.66** | **0.66** |
| **Hotel_1_Star** | { Thing Site,Accommodation ,Hotel} | { Thing Site,Accommodation ::Global,Hotel } | **0.75** | **0.75** |
| **Hotel_2_Star** | { Thing Thing Site,Accommodation ,Hotel } | { Thing,Site,Accommodation ::Global,Hotel } | **0.75** | **0.75** |
| **Hotel_3_Star** | { Thing Thing Site,Accommodation ,Hotel } | { Thing ,Site,Accommodation ::Global,Hotel } | **0.75** | **0.75** |
| **Hotel_4_Star** | { Thing Thing Site,Accommodation ,Hotel } | { Thing ,Site,Accommodation ::Global,Hotel } | **0.75** | **0.75** |
| **Hotel_5_Star** | { Thing Thing Site,Accommodation ,Hotel } | { Thing ,Site,Accommodation ::Global,Hotel } | **0.75** | **0.75** |
| **Residance** | {Thing Site,Accommodation } | { Thing,Site,Accommodation ::Global } | **0.66** | **0.66** |

**Tableau 6.6** La précision et le rappel taxonomique communs de quelques concepts de l'ontologie OnTourism ($O_{OnT}$) et l'ontologie OnTourism restructurée ($R_{OnT}$)

Après l'application de plusieurs mesures sur les ontologies restructurées, les résultats sont fournis dans le tableau 6.7. Les métriques incluent le nombre de classes ($NC$) de l'ontologie restructurée, nombre de classes communes avec l'ontologie de référence ($NCC$), et quelques autres métriques qui représentent les différences lexicales et taxonomiques entre l'ontologie de référence et les ontologies restructurées en termes de rappel et précision, y compris la précision lexicale ( $LP$ ), rappel lexical ( $LR$ ), précision taxonomique ( $TP$ ), rappel taxonomique($TR$) et le score ($TF$ / $TF'$).

Un $TF$ supérieur correspond à une hiérarchie conceptuelle de meilleure qualité. Si le $TF$ n'est pas influencé par le niveau lexique de l'ontologie, $F - mesure$ taxonomique ($TF'$) est calculé comme la moyenne harmonique de $LR$ et $TF$.

| **Ontologie restructurée comparée par** | $NC$ | $NCC$ | $LP$ | $LR$ | $TP$ | $TR$ | $TF$ | $TF'$ |
|---|---|---|---|---|---|---|---|---|
| **OnTourism** | 139 | 120 | 0.86 | 0.99 | 0.84 | 0.81 | 0.82 | 0.89 |
| **Accommodation** | 33 | 20 | 0.57 | 0.95 | 0.51 | 1 | 0.63 | 0.76 |
| **QALL-ME** | 136 | 86 | 0.88 | 0.96 | 0.77 | 0.84 | 0.80 | 0.87 |

**Tableau 6.7** Evaluation des ontologies restructurées OnTourism, Accommodation et QALL-ME

Les performances de ce processus varient en fonction du nombre de classes, le nombre de noms changés et leurs positions dans la hiérarchie. Dans l'ensemble, la mesure taxonomique ($TF$ / $TF'$) est plus importante 0,60 pour toutes les ontologies de référence.

Par exemple, l'ontologie OnTourism a 120 concepts, et l'ontologie restructurée a 139 concepts. Les concepts communs sont 120 concepts. Le processus de restructuration a apporté 19 nouveaux concepts. Ce nombre est faible par rapport au nombre de concepts dans l'ontologie de référence (120). Nous notons également qu'il n'y a pas un grand changement dans les noms et les positions des classes seulement dans certaines classes comme le concept Accommodation dans l'ontologie OnTourism change son nom à Accommodation :: Global dans OnTourismMPV, et cinq concepts changent leurs positions en hiérarchie des concepts vue LocalView de MV2P qui sont Excellent_Accommodation, Average_Hébergement, VeryGood_Accommodation, Poor_Accommodation, Terrible_Accommodation. Avec cette ontologie, les valeurs des mesures sont supérieures à 0,80. Dans le cas de l'ontologie d'Accommodation, les valeurs des mesures diminuent parce que le nombre de concepts ajoutés est supérieur à la moitié du nombre de concepts dans l'ontologie du logement. Dans l'ontologie de QALL-ME, quatre concepts ont changé de nom et de position Pour les noms, on a : Accommodation en Accommodation :: Global, Attraction en NearbyAttraction, Gastro en NearbyGastro et Infrastructure en NearbyInfrastructure. 34 concepts ont changé leur position comme ZoomAndAquarium, Restaurant, Banque, Gym et Bibliothèque. Nous notons que l'existence d'une ontologie de référence multipoints de vue améliore grandement les résultats de restructuration.

# 5 Mise en œuvre de l'ontologie multipoints de vue

Pour sérialiser l'ontologie multipoints de vue, nous avons choisi l'utilisation des graphes nommés [5] pour la modéliser efficacement. Plus spécifiquement, nous modélisons chaque hiérarchie en tant que graphe nommé. Les graphes nommés permettent de regrouper les triplets RDF sous le même URI où chaque groupe dans notre cas présente une vue. Ensuite, nous pouvons utiliser un langage de requête pour interroger l'ontologie.

Pour mettre en œuvre l'ontologie multipoints de vue, nous avons deux propositions de scénario d'implémentation (Kasri & Benchikha, 2016).

a. la première consiste à diviser physiquement l'ontologie en documents Turtle(.ttl). Chaque document est un graphe nommé RDF qui représente une vue. Comment diviser l'ontologie en parties est une question qui a été traitée dans différents travaux du domaine de la modularisation d'ontologie (Conesa (Conesa & Olivé, 2006), Borgo (Borgo, 2011), Del Vescovo (Del Vescovo, 2011)). Pour notre cas, l'ontologie est divisée en différentes vues selon le patron multipoints de vue. Ainsi, nous construisons pour chaque vue un document Turtle qui contient des triplets appartenant à cette vue. De cette manière, l'application client peut utiliser des requêtes SPARQL pour extraire un point de vue qui peut être aussi matérialisé que l'ontologie OntoMuPoV en utilisant respectivement les clauses SELECT ou CONSTRUCT.

b. Dans la deuxième proposition, nous sérialisons l'ensemble des graphes nommés de différentes vues (l'ontologie OntoMuPoV) dans un seule document en utilisant TriG[6] qui est un format compact de texte brut basé sur Turtle. TriQL[7] est un langage de requête pour le modèle de données de graphes nommés. Il est basé sur RDQL. Nous utilisons la clause SELECT pour récupérer les informations du sous-ensemble en tant que point de vue de OntoMuPoV.

Dans la section suivante, nous présentons des exemples pour chaque proposition avec des exemples à tour de rôle.

## 5.1 Scenario 1 : Plusieurs documents Trutle pour sérialiser plusieurs vues

Dans ce scénario, nous divisons l'ontologie OntoMuPoV physiquement en plusieurs documents Turtle qui présentent les concepts du patron multipoints de vue, la vue globale, la vue de taille, la vue financière et la vue localisation.

Le premier document MVPPattern.ttl contient les concepts et les propriétés du patron multi-points de vue qui sont: Accommodation ::Multiviewpoints, IDref, AccommodationView, has_GlobalView, has_LocalViews, has_IDref, with_ExcludeBridge, with_IncludeBridge, with_PartialBridge. Tous ces concepts et propriétés sont définis dans <http: // localhost: 433 / mvp / MVPPattern.ttl #> qui utilise l'abréviation mvpp.

[5] https://www.w3.org/2009/07/NamedGraph.html
[6] http://www.wiwiss.fu-berlin.de/suhl/bizer/TriG
[7] http://www.wiwiss.fu-berlin.de/suhl/bizer/TriQL

Accommodation :: Global, Accommodation:: Taille, Accommodation:: Localization, et Accommodation :: Financial, chacun d'eux est défini dans l'URI de sa vue. La figure suivante présente le document MVPPattern.ttl.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix mvpp: <http://localhost:433/mvp/MVPPattern.ttl#> .
@prefix fv: <http://localhost:433/mvp/FinancialView.ttl#> .
@prefix lv: <http://localhost:433/mvp/LocalizationView.ttl#> .
@prefix sv: <http://localhost:433/mvp/SizeView.ttl#> .
@prefix gv: <http://localhost:433/mvp/GlobalView.ttl#> .

mvpp:AccommodationMulti-Viewpoints rdf:type owl:Class .
mvpp:IDRef rdf:type owl:Class .
mvpp:AccommodationView rdf:type owl:Class .
mvpp:hasGlobalView rdf:type owl:ObjectProperty ;
        rdfs:range gv:Accommodatin::Global ;
        rdfs:domain mvpp:AccommodationMulti-Viewpoints .
mvpp:hasLocalViews rdf:type owl:ObjectProperty ;
        rdfs:domain mvpp:AccommodationMulti-Viewpoints ;
        rdfs:range mvpp:AccommodationView .
mvpp:hasIDRef rdf:type owl:ObjectProperty ;
        rdfs:domain gv:Accommodatin::Global ,
        mvpp:AccommodationMulti-Viewpoints ,
        mvpp:AccommodationView ;
        rdfs:range mvpp:IDRef .
mvpp:with_ExcludeBridge rdf:type owl:ObjectProperty ;
        rdfs:range mvpp:AccommodationView ;
        rdfs:domain mvpp:AccommodationView .
mvpp:with_IncludeBridge rdf:type owl:ObjectProperty ;
        rdfs:domain mvpp:AccommodationView ;
        rdfs:range mvpp:AccommodationView .
mvpp:with_PartialBridge rdf:type owl:ObjectProperty ;
        rdfs:domain mvpp:AccommodationView ;
        rdfs:range mvpp:AccommodationView .
```

**Figure 6.11**Document MVPPattern.ttl

De la même manière, nous créons les autres documents. Dans notre ontologie, chaque vue contient les propriétés qui la représentent. Nous définissons donc ces propriétés et leurs co-domaines dans le même URI de vue. La figure suivante présente la vue « taille ».

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix mvpp: <http://localhost:433/mvp/MVPPattern.ttl#>.
@prefix sv: <http://localhost:433/mvp/SizeView.ttl#> .

sv:numberOfRooms rdf:type owl:DatatypeProperty ;
        rdfs:domain sv:Accommodation::Size ;
        rdfs:range xsd:int .
mvpp:AccommodationView rdf:type owl:Class .
sv:Accommodation::Size rdf:type owl:Class ;
        rdfs:subClassOf mvpp:AccommodationView .
sv:Average-Accommodation rdf:type owl:Class ;
        owl:equivalentClass [ rdf:type owl:Class ;
        owl:intersectionOf(
sv:Accommodation::Size
        [ rdf:type owl:Restriction ;
        owl:onProperty sv:numberOfRooms ;
        owl:hasValue 3 ] ) ] ;
        rdfs:subClassOf sv:Accommodation::Size .
sv:Big-Accommodation rdf:type owl:Class ;
        owl:equivalentClass [ rdf:type owl:Class ;
        owl:intersectionOf ( sv:Accommodation::Size
        [ rdf:type owl:Restriction ;
        owl:onProperty sv:numberOfRooms ;
        owl:someValuesFrom [ rdf:type rdfs:Datatype ;
        owl:onDatatype xsd:int ;
        owl:withRestrictions ( [ xsd:minExclusive 3
        ] ) ] ] ) ] ;
        rdfs:subClassOf sv:Accommodation::Size .
sv:Small-Accommodation rdf:type owl:Class ;
        owl:equivalentClass [ rdf:type owl:Class ;
        owl:intersectionOf ( sv:Accommodation::Size
        [ rdf:type owl:Restriction ;
        owl:onProperty sv:numberOfRooms ;
        owl:someValuesFrom [ rdf:type rdfs:Datatype ;
        owl:onDatatype xsd:int ;
        owl:withRestrictions ( [ xsd:maxExclusive 3
        ] ) ] ] ) ] ;
        rdfs:subClassOf sv:Accommodation::Size .
```

**Figure 6.12**Document pour la vue taille (size)

Nous pouvons imaginer des applications qui bénéficieront de l'ontologie OntoMuPoV en permettant un accès sélectif selon le point de vue de l'utilisateur. Considérons deux profils d'utilisateurs :

- Un étudiant défini comme un utilisateur avec un petit budget, alors il choisira l'accommodation la moins chère. Dans ce cas, l'étudiant s'intéresse par la vue financière.

- Un homme d'affaire défini comme une personne avec un budget élevé. Il veut être proche de son entreprise, donc il s'intéresse par la localisation.

Dans ces deux exemples, le système peut construire une requête SPARQL sur des graphes nommés en utilisant des clauses SELECT ou CONSTRUCT.

La requête ci-dessous utilise les graphes nommés pour trouver une accommodation appropriée pour le profil « étudiant ». La première clause GRAPH associe les patrons aux graphes nommés http: // localhost: 433 / mvp / GlobalView.ttl pour trouver le détail et les informations globales sur les accommodations. Le patron dans la deuxième clause GRAPH trouve la bonne accommodation dans la vue financière où le prix est inférieur à 500, et avec le même identifiant IDRef que celui déjà trouvé dans la première clause GRAPH.

```
PREFIX  mvpp:  <http://localhost:433/mvp/MVPPattern.ttl#>
PREFIX  gv:  <http://localhost:433/mvp/GlobalView.ttl#>
PREFIX  fv:  <http://localhost:433/mvp/FinancialView.ttl#>
PREFIX  g:  <http://localhost:433/mvp/>
SELECT ?name ?phone ?fax ?mail
FROM NAMED <http://localhost:433/mvp/GlobalView.ttl>
FROM NAMED <http://localhost:433/mvp/FinancialView.ttl>
WHERE
{
 GRAPH g:GlobalView.ttl {
?accogv rdf:type gv:Accommodatin::Global;
gv:name ?name;
 gv:phone ?phone;
gv:fax ?fax;
 mvpp:IDRef ?id.  } .
 GRAPH g: FinancialView.ttl  {
?accofv rdf:type fv:Good_Accommodation
    mvpp:IDRef ?id;
                fv:hasPrice ?price.
     FILTER( ?price <500)  }
}
```

**Figure 6.13**Requête SPARQL avec SELECT

En utilisant CONSTRUCT, il est possible d'extraire des parties ou l'ensemble des graphes de l'ontologie OntoMuPoV . L'exemple suivant retourne le graphe qui représente le profil ou le point de vue d'un homme d'affaires qui s'intéresse à la vue localisation. Son but est de trouver un logement proche de son entreprise à Londres.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX mvpp: <http://localhost:433/mvp/MVPPattern.ttl#>
PREFIX lv: <http://localhost:433/mvp/GlobalView.ttl#>
PREFIX gv: <http://localhost:433/mvp/LocalizationView.ttl#>
CONSTRUCT
{ ?acco rdfs:subClassOf mvpp:AccommodationMulti-Viewpoints;
                        rdf:type owl:Class;
          mvpp:hasLocalView ?accoview;
          mvvp:hasGlobalView ?accoglobal;
          mvpp:IDRef ?id. }
where
{
?accoglobal mvpp:IDRef ?id.
?accoview mvpp:IDRef ?id;
      rdfs:subClassOf lv: Accommodation-in-City;
      lv:isLocatedAt ?place.
?place lv:address "London".
}
```

**Figure 6.14**Requête SPARQL avec CONSTRUCT

### 5.2 Scenario 2 : Un seul document TriG pour sérialiser plusieurs vues

Dans ce scénario, nous utilisons TriG pour sérialiser plusieurs vues dans le même document. Le document TriG permet d'écrire des graphes nommés sous une forme textuelle compacte. Le document TriG présenté dans la figure 6.15 présente un extrait de l'ontologie OntoMuPoV. Le premier graphe contient des informations sur le patron multipoints de vue nommé vp: pattern. Chaque vue est  représentée par un graphe qui se désigne comme le nom du graphe URI vp: viewname.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix mvpp: <http://localhost:433/mvp/MVPPattern.ttl#> .
@prefix fv: <http://localhost:433/mvp/FinancialView.ttl#> .
@prefix lv: <http://localhost:433/mvp/LocalizationView.ttl#> .
@prefix sv: <http://localhost:433/mvp/SizeView.ttl#> .
@prefix gv: <http://localhost:433/mvp/GlobalView.ttl#> .
@prefix vp: <http://localhost:433/mvp/>

 vp:pattern {
mvpp:AccommodationMulti-Viewpoints rdf:type owl:Class .
mvpp:IDref rdf:type owl:Class .
mvpp:AccommodationView rdf:type owl:Class .
lv:Accommodation::Localization rdfs:subClassOf
mvpp:AccommodationView .
.....
}
```

```
vp :size {
sv:numberOfRooms rdf:type owl:DatatypeProperty ;
        rdfs:domain sv:Accommodation::Size ;
        rdfs:range xsd:int .
sv:Accommodation::Size rdf:type owl:Class ;
        rdfs:subClassOf mvpp:AccommodationView .
sv:Average-Accommodation rdf:type owl:Class ;

            owl:equivalentClass [ rdf:type owl:Class ;
.....
.....
.....
}
```

**Figure 6.15** Extrait à partir d'un document TriG

Nous utilisons la requête TriQL pour sélectionner l'accommodation avec son nom, son prix et numberOfRoom comme indiqué dans l'exemple suivant.

```
SELECT ?accogv ?name ?price ?numberOfRoom
WHERE ?graph
(?accogv rdf:type gv:Accommodatin::Global.
 ?accogv  gv:name ?name.
?accogv mvpp:IDRef ?id.
?accosv rdf:type sv:Accommodation::Size.
?accosv mvpp:IDRef ?id.
?accosv sv:numberOfRoom ? ?numberOfRoom.
?accofv rdf:type fv:Good_Accommodation.
fv:hasPrice ?price.
AND ?price < 1000 && ?numberOfRoom >1
)
USING rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ,
    mvpp FOR <http://localhost:433/mvp/MVPPattern.ttl#> ,
    Fv FOR <http://localhost:433/mvp/FinancialView.ttl#> ,
    sv FOR <http://localhost:433/mvp/SizeView.ttl#> ,
    gv FOR <http://localhost:433/mvp/GlobalView.ttl#> .
```

**Figure 6.16**Requête TriQL

## 6 Conclusion

Nous avons décrit dans ce chapitre l'implémentation de notre architecture d'intégration en utilisant trois ontologies (l'ontologie OnTourism, l'ontologie Accommodation et l'ontologie QALL-ME) comme exemple applicatif. L'évaluation de notre système est effectuée en mesurant la précision lexicale (LP), rappel lexical (LR), précision taxonomique (TP), rappel taxonomique (TR) et le score (TF / TF') pour évaluer la restructuration des ontologies locales par le patron multipoints de vue. Généralement, Le but de la restructuration d'ontologies est de corriger des anomalies (Rouane-Hacene, Fennouh, Nkambou, & Valtchev, 2010) et le processus d'évaluation utilisé est basé sur une ontologie de référence créée par un expert. Cependant notre objectif est de restructurer des ontologies locales selon le patron MV2P. Notre résultat d'évaluation est influencé par le processus de restructuration (processus d'analyse relationnelle) ainsi que le patron MV2P et l'ontologie de référence utilisée. Le patron MV2P injecte, dans l'ontologie restructurée, des nouveaux concepts comme (Accommodation ::GlobalView, Accommodation:: Financial…etc.) et des nouvelles hiérarchies (vue Financial, Size…etc) qui influencent directement les métriques lexicales et taxonomiques. En outre, l'absence de l'ontologie de référence multipoints de vue créée par un expert de domaine pour chaque ontologie à restructurer présente une limitation pour une véritable évaluation. Nous sommes bien conscients que la qualité des résultats d'évaluation peut être amoindrie par cette absence. Ceci nous conduit à la conclusion que l'usage des ontologies de référence créée par un expert selon le patron MV2P dans le processus d'évaluation est plus avantageux que d'utiliser les ontologies locales.

Pour conclure ce chapitre, nous proposons aussi deux scénarios pour mettre en œuvre l’ontologie multipoints de vue. Le premier consiste à diviser physiquement l'ontologie en documents Turtle et le deuxième procède par sérialisation de l'ensemble des graphes nommés des différentes vues (l’ontologie OntoMuPoV) dans un seul document en utilisant TriG.

# CONCLUSION GÉNÉRALE

Pour rendre une ontologie accessible et réutilisable à partir de points de vue différents, l'approche multipoint de vue a été utilisée pour modéliser une telle ontologie. L'intégration de la notion de point de vue dans la représentation des connaissances est très importante afin de modéliser le monde d'une façon plus naturelle et plus réelle et avoir une meilleure visibilité des éléments ontologiques (concepts, rôles, individus). Nous récapitulons ici notre contribution et présentons quelques perspectives.

Dans cette thèse, nous avons travaillé activement à la réalisation de principaux objectifs suivants :

1) proposition d'un patron de conception d'ontologie multipoints de vue « MV2P » : Il y a une poussée de recherche vers l'utilisation des patrons de conception dans l'ingénierie ontologique. Ceux-ci aident à rendre les ontologies plus lisibles, plus maintenables et plus réutilisables. Dans notre cas, nous avons proposé un patron de conception d'ontologie d'architecture pour intégrer la notion de point de vue dans la structure de l'ontologie. Ce patron permet une représentation multiple des concepts ontologique à parti de laquelle chaque utilisateur de l'ontologie va pouvoir voir son propre point de vue sur un concept donné.
2) proposition des patrons de correspondance des vues : les vues du patron multipoints de vue sont liées par des ponts qui représentent la direction, la possibilité ou l'impossibilité du passage des correspondances d'une vue à une autre. Ces liens de correspondances sont représentés d'une manière plus expressive et indépendante de tout langage ou formalisme d'ontologie en utilisant un ensemble de patrons de correspondances qui sont: Passerelle d'inclusion totale, Passerelle d'inclusion partielle, Passerelle d'équivalence et Passerelle d'exclusion. Ces patrons de correspondance sont destinés à fournir des modèles de référence en aidant à modéliser les alignements entre les vues.
3) proposition d'un Framework de validation du patron de conception d'ontologie MV2P : pour évaluer un patron de conception d'ontologie, Il n'y a pas des méthodologies rigoureuses mais seulement certains critères, telle que la réutilisation. Pour cela nous avons proposé un Framework de validation de notre patron de conception d'ontologie. Ce Framework est basé sur l'extension de l'analyse formelle des concepts (les structures de patrons) pour spécifier le patron formellement. Celle-ci permet de vérifier si l'implémentation du patron résout le problème de conception posé tout en respectant le modèle de solution proposé.
4) proposition une approche d'intégration hybride : en se basant sur le patron multipoints de vue MV2P, nous avons proposé une architecture d'intégration sémantique des bases de données à base ontologique. Ce système d'intégration prend le patron multipoints de vue comme un modèle d'harmonisation virtuelle pour les ontologies des bases de données à base ontologique et aussi un modèle d'accès par point de vue. Cette approche contient deux phases principales : pré-intégration et intégration. La pré-intégration inclut toutes les étapes préliminaires qui consistent à préparer les données ontologiques à intégrer. Ces étapes sont la spécification des vues, l'imitation

de patron MV2P et la restructuration des ontologies. Pour cette dernière étape, nous avons proposé une approche de restructuration d'ontologie basée sur le patron de conception d'ontologie MV2P et l'analyse relationnelle de concepts. L'approche de restructuration comprend deux étapes: (1) la construction des familles des contextes relationnels à partir des concepts d'ontologie et le patron imité, et (2) la transformation des familles des contextes relationnels (FCR) en une l'ontologie.

5) développement d'un outil nommé OntoMuPoV (**Onto**logie **Mu**lti**Po**ints de **V**ue). Cet outil assiste l'ingénieur d'ontologie dans la création d'une ontologie multipoints de vues pour l'intégration des ontologies des bases de données à base ontologique.

De nombreuses perspectives peuvent être envisagées. Dans ce qui suit nous présentons celles qui nous paraissent être possibles.

1. Nous envisageons de valider le patron MV2P par TLA+ [1] en spécifiant son comportement. TLA +, la logique temporelle des actions, un formalisme introduit au début des années 1990. Il est essentiellement utilisé pour spécifier les propriétés comportementales d'un système, c'est-à-dire ce que le système normalement va faire (une simulation du comportement). Dan notre cas, on va simuler la création, la coexistence des objets de même vue ou des vue différentes.
2. Nous envisageons d'évaluer le système d'intégration proposé en matière de l'importance d'intégration de la notion de point de vue dans son schéma global. Pour cela, nous allons intégrer un module d'interrogation et évaluer le système avec et sans un schéma global créé selon le patron MV2P.
3. Dans nos travaux précédents, nous avons proposé une méthode d'alignement des ontologies larges (Kasri & Benchikha, 2011). Nous envisageons de développer une nouvelle méthode d'alignement des ontologies multipoints de vue en utilisant l'extension de l'analyse formelle de concepts (AFC) « structures de patrons ». La particularité qui nous intéresse est la production des regroupements d'objets complexes (vues correspondances) sur la base des caractéristiques communes (vue globale partagée), c'est l'objectif de l'alignement orienté vue.

[1] https://lamport.azurewebsites.net/tla/tla.html

# Bibliographie


Aarnio, P., Seilonen, I., & Friman, M. (2014). Semantic repository for case-based reasoning in CBM services. *in Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pp. 1–8.

Abbes S. B., Scheuerman A., Meilender T. & D'aquin M. (2012). Characterizing modular ontologies. *In 6th International Workshop on Modular Ontologies*-WoMO 2012, pp. 13–25.

Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D. & Tolle, K. (2001) The ics-FORTH RDFSuite: Managing voluminous RDF description bases, *in Proceedings of the 2nd International Workshop on the Semantic Web*, pp. 1–13.

Alexander, C., Ishikawa, S., & Silverstein, M., (1977, August) A Pattern Language: Towns, Buildings, Construction Center *for Environmental Structure Series. Oxford University Press, New York.*

Alirezaie, M., Hammar, K. & Blomqvist, E. (2018). SmartEnv as a network of ontology patterns. *Semantic Web* 9(6): 903-918.

Artale, A., Calvanese, D., Kontchakov, R. & Zakharyaschev, M. (2009) The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR), 36:1–69.*

Azmeh, Z., Huchard, M., Tibermacine, C., Urtado, C. & Vauttier, S. (2008). Wspab : A tool for automatic classification and selection of web services using formal concept analysis. *ECOWS'08 IEEE Sixth European Conference on Web Services*, pp. 31–40.

Baader, F. (2004) The description logic handbook: Theory, implemention, and applications. 2. repr. *Cambridge. Cambridge Univ. Press,.isbn: 9780521781763.*

Bach, T.L. (2006) Construction d'un Web Sémantique Multi-Points de Vue, *PhD thesis, Department of Computer Science, Ecole des Mines de Paris a Sophia Antipolis.*

Bachimont B., Isaac., A. & Troncy., R. (2002). Semantic commitment for designing ontologies: A proposal. *In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW*), volume LNAI 2473 of Lecture Notes in Artificial Intelligence, pp. 114–121.

Bailey, J., Bry, F., Furche, T. & Schaffert, S. (2005) Web and semantic web query languages: A survey. *in Proceedings of the First international conference on Reasoning Web*, Springer-Verlag, pp. 35–133,.

Bakhtouchi, A., Bellatreche, L. Jean, S. & Ait-Ameur, Y. (2012) Ontologies as a solution for simultaneously integrating and reconciliating data sources. *In Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on, IEEE. pp. 1–12*

Barbut, M. & Monjardet, B. (1970) Ordre et Classification, Algèbre et Combinatoire. *Hachette, Paris*.

Batini, C., Lenzerini, M., & Navathe, S. (1986) A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys, 18(4):323-364.*

Bechhofer, S., Van, H. F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P. & Stein, L. (2004) Owl web ontology language reference. *W3C, http://www.w3.org/TR/owlref/.*

Benchikha, F. & Boufaida, M. (2007) The Viewpoint Mechanism for Objectoriented Databases Modelling, Distribution and Evolution, *In Journal of Computing and Information Technology. Vol 15, pp. 95-110.*

Bendaoud, R., Toussaint, Y. & Napoli, A. (2010) L'Analyse Formelle de Concepts au service de la construction et l'enrichissement d'une ontologie. *Revue des Nouvelles Technologies de l'Information*, RNTI-E-18, pp. 133-164.

Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M., & Vincini, M. (2000) Information Integration: The MOMIS Project Demonstration. *In Proceedings of 26th International Conference on Very Large Data Bases (VLDB'00), Morgan Kaufman, pp. 611–614.*
Benslimane, D., Arara, A., Falquet, G., Maamar, Z., Thiran, P. & Gargouri, F. (2006) Contextual ontologies: motivations, challenges, and solutions, *Fourth Biennial International Conference on Advances in Information Systems, Springer, Izmir,Turkey,* pp.168–176.
Berners-Lee, T., Hendler, J. & Lassila, O. (2001) The semantic web. *Scientific American, 284 : 34-43*.
Berthet, D., Bonjour, M., De Bessé, B., Falquet, G., Léonard, M., & Sindayamaze, J.(1994). ConcepTerm : Construction de dictionnaires encyclopédiques multilingues et informatisés. *CUI technical report*, University of Geneva.
Birkhoff G. (1940). Lattice Theory. *American Mathematical Society*, Providence, RI, 1st édition.
Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems,* 5(3):1–22.
Bordat, J.-P. (1986) Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences Humaines, 96 :31–47.*
Borgo, S. (2011) Goals of modularity: a voice from the foundational viewpoint, *The Fifth International Workshop on Modular Ontologies (WOMO'2011), Vol. 230 of Frontiers in Artificial Intelligence and Applications, IOS Press, Ljubljana, Slovenia, pp.1–6.*
Borst, W.N. (1997) Construction of engineering ontologies. *Construction of Engineering Ontologies*.
Brickley, D. & Guha, R. (2002). Rdf vocabulary description language 1.0: Rdf schema , W3C, http://www.w3.org/TR/rdfschema/.
Brigitte, B., Sylvie, S. & Clément, Av.J.B.(1999) Terminae: A linguistics-based tool for the building of a domain ontology. *In Knowledge Acquisition, Modeling and Management*, volume 1621, pages 49–66. Springer Berlin Heidelberg.
Broekstra, J., Kampman, A. & Van, .H.F. (2002) Sesame: A generic architecture for storing and querying rdf and rdf schema. *In Proceedings of the 1st International Semantic Web Conference (ISWC'02), pp. 54–68.*
Buchheit, M., Donini Francesco M., & Schaerf, A. (1993). Decidable Reasoning in Terminological Knowledge Representation Systems, *Journal of Artificial Intelligence Research*.
Calì, A., Frosini, R., Poulovassilis, A. & Wood, P.T. (2014) In: Flexible Querying for SPARQL. *Springer Berlin Heidelberg, Berlin, Heidelberg (2014) 473–490*.
Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., & Savo, D. F. (2011). The MASTRO system for ontology-based data access. *Semantic Web*, 2( 1): 43–53.
Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2017). Ontology-Based Data Access and Integration. *In L. Liu & M. T. Özsu (Eds.),* Encyclopedia of Database Systems (pp. 1–7). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4899-7993-3_80667-1.
Carpineto, C. & Romano, G. (2005). Using concept lattices for text retrieval and mining. *In Formal Concept Analysis*, pp. 161–179.
Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A. & Wilkinson, K. (2004) Jena: Implementing the Semantic Web Recommendations. *In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters (WWW'04),* pages 74–83, New York, NY, USA. ACM Press.

Chein, M. (1969) Algorithme de recherche des sous-matrices premières d'une matrice. *Bull. Math. Soc. Sci. Math. R.S. Roumanie, 13 :21–25.*
Cimiano, P. & Völker, J. (2005) Text2onto. *In Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems, (NLDB), Springer, pp. 227–238.*
Coad, P. (1995) Object Models: Strategies, Patterns and Applications, *Yourdon Press Computing Series, Austin, TX.*
Conesa, J. & Olivé, A. (2006) A method for pruning ontologies in the development of conceptual schemas of information systems, *Journal on Data Semantics, Vol. 5, pp.64–90.*
Corcho., O., Fernández., M., Gómez-Pérez., A., & López-Cima., A. (2005). "Building Legal Ontologies with METHONTOLOGY and WebODE". In : Law and the Semantic Web Heidelberg, DE: Springer (2005), *pp. 142-157.*
Cure, O. (2010) Merging expressive spatial ontologies using Formal Concept analysis with uncertainty considerations. *Methods for Handling Imperfect Spatial Information. Spinger-Verlag; 256:*188–209.
Daga, E., Blomqvist, E., Gangemi, A., Montiel, E., Nikitina, N., Presutti, V. & Villazón-Terrazas, B. (2010) NeOn D2.5.2 Pattern based ontology design: methodology and software support. https://pdfs.semanticscholar.org/
Dehainsala, H., Pierra, G. & Bellatreche, L. (2007) OntoDB: An Ontology-Based Database for Data Intensive Applications , *in Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07), pp. 497–508.*
Del Vescovo, C. (2011) The modular structure of an ontology: atomic decomposition towards applications, *Proceedings of the 24th International Workshop on Description Logics (DL'11), Vol. 745 of CEUR-WS, 13–16 July, Barcelona, Spain.*
Dellschaft, K. & Staab, S. (2006) On how to perform a gold standard based evaluation of ontology learning , *5th International Semantic Web Conference, ISWC 2006, LNCS, Vol. 4273, Springer, Athens, GA.*
Dictionnaire de l'Académie française, neuvième édition, Version informatisée.
http://atilf.atilf.fr/academie9.htm
Dieng, R., Cobyo, O., Gandonf, F., Giboin, A., Golebiowska, J., Matta, N. & Ribière, M., (2001) Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du knowledge management . *Dunod, 2 edition.*
Ekaputra, F.,J., Sabou, M., Serral, E., Kiesling, E., & Biffl, S.(2017). Ontology-Based Data Integration in Multi-Disciplinary Engineering Environments: *A Review*. OJIS 4(1): 1-26.
Euzenat, J., (2004). An API for ontology alignment. *In The Semantic Web–ISWC* 2004.
Falquet, G. & Mottaz, C.L. (2002) A model for the collaborative design of multi-point-of-view terminological knowledge bases , *in Dieng, R. and Matta, N. (Eds): Knowledge Management and Organizational Memories, Kluwer Academic Publishers, Stockholm, pp.193–202.*
Fankam, C., Bellatreche, L., Dehainsala, H., Ait-Ameur, Y. & Pierra, G. (2009) Sisro : conception de bases de données à partir d'ontologies de domaine. *Technique et science informatiques (TSI).*
Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995) Design Patterns -Elements of Reusable Object-Oriented Software , *Addison-Wesley.*
Gandon, F. (2006) Ontologies informatiques . *Interstices, Journal en ligne de l'INRIA. http://interstices.info/display.jsp?id=c_17672.*
Gangemi, A. (2005). Ontology Design Patterns for Semantic Web Content. *In Yolanda Gil*. Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, The Semantic Web – ISWC 2005, 4th *International Semantic Web Conference*, ISWC 2005, Galway,

Ireland, November 6-10, 2005, Proceedings, volume 3729 of *Lecture Notes in Computer Science*. Springer, *pp. 262–276.*
Gangemi, A. & Presutti, V. (2009) Ontology Design Patterns . *In S. Staab, R.Studer (Ed.), Handbook of Ontologies (2nd edition). Springer: Berlin.*
Ganter, B. & Wille, R. (1999) Formal Concept Analysis. *Springer, mathematical foundations edition.*
Ganter, B. (1984) Two basic algorithms in concept analysis. *FB4-Preprint 831, Technische Hochschule Darmstadt.*
Ganter, B. & Kuznetsov, S. O. (2001) Pattern Structures and Their Projections. *9th International Conference on Conceptual Structures. 2120, pp. 129-142.*
García-Castro, R. & Gómez-Pérez, A. (2013) The Object with States Ontology Design Pattern, *WOP. http://ceurws. org/Vol-1188/paper_5.pdf*
Germán, D. (2004) Introduction to Database Systems. *TuringMachine. http://turingmachine.org/courses/2004/csc370K04/lectures/01_intro_4up.pdf cours C SC 370 université de Victoria.*
Glöckner, M. & Ludwig, A. (2016) LoSe ODP-An Ontology Design Pattern for Logistics Services. *In Workshop on Ontology and Semantic Web Patterns (7th edition), Pascal Hitzler, Karl Hammer, Monika Solanki, Agnieszka Lawrynowicz, Andrea Nuzzolese, and Adila Krisnadhi (Eds.) http://ontologydesignpatterns.org/wiki/images/f/fb/WOP2016_paper_14.pdf*
Godin, R., Missaoui, R., & April, A. (1993) Experimental comparison of navigation in a galois lattice with conventional information retrieval methods. *International Journal of Man-machine Studies, 38 :747–767.*
Godin, R. (1991) Leaming Algorithms Using a Galois Lattice Structure, *Proceedings of the Third International Conference on Tools for Artificial Intelligence, pp. 22-29.*
Godin, R., & Valtchev, P. (2005). Formal concept analysis-based class hierarchy design in object-oriented software development. *In Formal Concept. Analysis* Springer Berlin Heidelberg. pp. 304-323.
Gomez, G-L.G. (2005) Construction automatisée de l'ontologie de systèmes médiateurs. Application à des systèmes intégrant des services standards accessible via le Web. *PhD thesis, Université Paris XI Orsay*
Graube, M., Urbas, L., & Hladik, J. (2016). Integrating industrial middleware in Linked Data collaboration networks", *in Proceedings of 21st International Conference on Emerging Technologies and Factory Automation (ETFA*), pp. 1–8.
Gruber, T.R. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition, 5(2):199–220.*
Guan-Yu, L., Shu-Peng, L. & Yan, Z. (2011) Formal Concept Analysis based Ontology Merging Method. *International Conference on Computer Science and Information Technology. 2010 Jul 9-11 Chengdu*; *8: 279–82.*
Guarino, N. (1998) Formal ontologies and information systems. *In N. Guarino, editor, Proceedings of FOIS'98, IOS Press, Amsterdam.*
Haase, P., Brockmans, S., Palma, R., Euzenat, J. & dÕAquin, M (2007). Updated Version of the Networked Ontology Model. *Deliverable D1.1.2, NeOn project.*
Haav, H. (2004) A Semi-automatic Method to Ontology Design by Using FCA. *Proceedings of the 2nd International Workshop on Concept Lattices and their Applications, pp. 13-25.*
Halevy, A. Y. (2001) Answering queries using views: A survey. *The VLDB Journal, 10(4): 270-294.*

Hamaz, K. & Benchikha, F. (2017). A novel method for providing relational databases with rich semantics and natural language processing. *J. Enterprise Inf. Management* 30(3): 503-525.
Hemam, M., (2012). Développement des ontologies multi-points de vue : Une approche basée sur la logique de description. *Thèse doctorat*, Université de Mentouri, Constantine, Algérie.
Huchard, M. Rouane-Hacene, M., Roume, C. & Valtchev, P. (2007) Relational concept discovery in structured datasets. *Annals of Mathematics and Artificial Intelligence*, 49(1-4).
Hustadt, U., Motik, B. & Sattler, U. (2007) Reasoning in description logics by a reduction to disjunctive datalog. *Journal of Automated Reasoning*, 39(3):351–384.96
Industrial automation systems and integration - parts library - part 42 (1998): Description methodology: Methodology for structuring parts families. Technical report.
InterOp, I. (2006). State of the art and state of the practice including initial possible research orientations, d8.1. Rapport technique, InterOp, Interoperability Research for Networked Enterprises Applications and Software.
Jean, S., Pierra, G., & Ait-Ameur, Y. (2007) Domain Ontologies : A Database-Oriented Analysis. *In Web Information Systems and Technologies, International Conferences, WEBIST 2005 and WEBIST 2006. Revised Selected Papers, Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg. *pp 238–254*.
Jiang, G., Ogasawara, K., Endoh, A. & Sakurai, T. (2003) Context-based Ontology building support in clinical domains using formal concept analysis. *International Journal of Medical Informatics.*; 71(1):71–81.
Jiang, G., Ogasawara, K., Nishimoto, N., Endoh A. & Sakurai, T. (2005) FCA View Tab: A concept-oriented view generation tool for clinical data using formal concept analysis. *Proceedings of the 8th International Protege Conference*. *pp 91-93.*
Jorge, P., Marcelo, A.b& Claudio, G. (2009). Semantics and complexity of SPARQL. ACM Trans. Database Syst. 34(3).
Kasri, S., and Benchikha, F. (2011) Large-scale ontologies: pattern and partition-based alignment. *IJWS 1(1/2)*: 36-53.
Kasri, S., and Benchikha, F. (2014) Multi-viewpoints Ontology Design Pattern and its Evaluation with Pattern Structures.*7th IADIS International Conference Information Systems*, Spain.
Kasri, S., and Benchikha F. (2014) Integrating Multi-viewpoints Paradigm in Ontology Using Ontology Design Patterns. ADBIS (2), *Ohrid, Macedonia. Advances in Intelligent Systems and Computing 312*, Springer. *pp. 257-268.*
Kasri, S. and Benchikha, F. (2016) Refactoring ontologies using design patterns and relational concepts analysis to integrate views: the case of tourism. *Int. J. Metadata, Semantics and Ontologies, inderscience*. 11( 4): 243–263.
Kaytoue, M. (2011) Traitement de données numériques par analyse formelle de concepts et structures de patrons. *PhD thesis, Universite henry Poincare, Nancy*. 111
Kendall, E., Dutra, M., Jacobs, J. & Schwab, S. (2005) Ontology Definition Metamodel. *Revised submission to OMG*.
Kontchakov, R., Rezk, M., Rodriguez-Muro, M., Xiao, G. & Zakharyaschev, M. (2014) Answering SPARQL queries over databases under OWL 2 QL entailment regime. *In: Proc. of International SemanticWeb Conference (ISWC 2014). Lecture Notes in Computer Science, Springer.*
Krisnadh, A.A., Arko, R.A., Carbotte, S., Chandler, C., Cheatham, M., Finin, T.W., Hitzler, P., Janowicz, K., Narock, T.W. , Raymond, L., Shepherd, A. & Wiebe, P. (2015). Ontology Pattern Modeling for Cross-Repository Data Integration in the Ocean

Sciences: The Oceanographic Cruise Example. *Semantic Web Enabled Software Engineering 2015. pp. 185-203.*
Kuznetsov, S.O. and Obiedkov, S. A. (2002) Comparing Performance of Algorithms for Generating Concept Lattices. *Journal of Experimental & Theoretical Artificial Intelligence, 14 :189–216.*
Lakhal, L. and Stumme, G. (2005). Efficient mining of association rules based on formal concept analysis. *In Formal Concept Analysis. pp. 180–195.*
Lisi, F.A. & Esposito, F. (2014) Supporting integrated tourism services with semantic technologies and machine learning, *International Semantic Web Conference, Riva del Garda, Italy, pp.361–364.*
Looser, D., Ma, H., Schewe, K.-D. (2013):Using Formal Concept Analysis for Ontology Maintenance in Human Resource Recruitment. *APCCM 2013*. p.p. 61-68.
Lu, J., Ma, L., Zhang, L. Brunner, J.-S., Wang, C. Pan, Y. & Yu, Y. (2007) Sor: a practical system for ontology storage, reasoning and search, *in Proceedings of the 33rd international conference on Very large data bases (VLDB'07). pp. 1402–1405.*
Lutz, C. (2008) The complexity of conjunctive query answering in expressive description logics. *In Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR 2008 ), number 5195 in LNAI*, Springer. *pp. 179–193.*
Lutz, C., Toman, D. & Wolter, F. (2009) Conjunctive query answering in the description logic EL using a relational database system. *In Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI 2009) . pp. 2070–2075.*
Ma, L., Su, Z., Pan, Y., Zhang, L. & Liu, T. (2004) RStar : an RDF Storage and Query System for Enterprise Resource Management. *In Proceedings of the 30th International Conference on Information and Knowledge Management* (CIKM'04). *pp. 484–491*.
Marcaillou E.,S. (1995). Intégration de la notion de points de vue dans la modélisation par objets : le langage VBOOL. *PhD thesis*, University of Toulouse, France.
Marino, O. (1993) Raisonnement classificatoire dans une représentation à objets multi-points de vue . *thèse de l'Université Joseph Fourier-Grenoble 1.*
Maynard, D, Funk, A. & Peters, W. (2009). Using lexico-syntactic ontology design patterns for ontology creation and population. *In Proceedings of WOP2009 collocated with ISWC2009*, volume 516. CEUR-WS.org.
McBrien, P., & Poulovassilis, A. (2003, March). Data integration by bi-directional schema transformation rules. *In Data Engineering, 2003. Proceedings. 19th International Conference on). IEEE. pp. 227-238.*
McGuinnes, D. & van Harmelen, F. (2012) OWL 2 Web Ontology Language Document Overview (Second Edition): *W3C Recommendation 11 December 2012. https://www.w3.org/TR/owl2- overview/.*
Miller, G. A. (1995) Wordnet : a lexical database for english, *Communications of the ACM*, vol. 38. pp. 39–41.
Minsky, M. (1975) A Framework for representing knowledge, *Chapitre 6, McGrawHill. pp.156-189*
Mortensen, J., Horridge, M., Musen, M.A. & Noy, N.F. (2012) Modest use of ontology design patterns in a repository of biomedical ontologies , *WOP'12 Proceedings of the 3rd International Conference on Ontology Patterns, Vol. 929, pp.37–48.*
Neches, R., Fikes, R., Finin, T., Gruber, T., Paqtil, R., Senator, T., & Swartout, W.R. (1999) Enabling technology for knowledge sharing. *AI Magazine, 12(3):36–56.137*
Norris, E. M. (1978) An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathématiques Pures et Appliquées, 23(2) :243–250.*
Nourine, L. & Raynaud, O. (1999) A fast algorithm for building lattices. *Information Processing Letters, 71(5-6) :199–204.*

Ortiz, M., Rudolph, S. & Simkus, M. (2011) Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. *In Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI-2011 ), pp 1039–1044. AAAI Press.*
Ortiz, M., Calvanese, D. & Eiter, T. (2008) Data complexity of query answering in expressive description logics via tableaux. *Journal of Automated Reasoning, 41(1):61–98.*
Otero-Cerdeira, L., Rodriguez-Martinez, F.J. & Gomez-Rodriguez, A. (2015). Ontology matching: A literature review. *Expert Systems with Applications* 42(2):949–971.https://doi.org/10.1016/j.eswa.2014.08.032.
Ou, S., Pekar, V., Orasan, C., Spurk, C. & Negri, M. (2008) Development and alignment of a domain-specific ontology for question answering, *Proceedings of the Sixth International Language Resources and Evaluation, ELRA, Marrakech, Morocco.*
Pan, Z. & Heflin, J. (2003) Dldb: Extending databases to support semantic web queries , *in Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03), p. 109–113.*
Park, M. J., Lee, J. H., Lee, C. H., Lin, J., Serres, O. & Chung, C.W., (2007) An Efficient and Scalable Management of Ontology, *in Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA'07), pp. 975–980.*
Peffers, K., Rothenberger, M.A., Tuunanen, T. & Vaezi, R. (2012). Design Science Research Evaluation. *DESRIST 2012, pp. 398-410.*
Pierra, G. (2008) Context representation in domain ontologies and its use for semantic integration of data, *Journal Of Data Semantics (JODS), vol. 4900, pp. 173–210.*
Pierra, G. (2003). Context-explication in conceptual ontologies: The PLIB approach. *in Proc. of Concurrent Engineering* (CE'2003), *pp. 243–254.*
Pierre, G.(2005) Merise: Modélisation de systèmes d'information.
Pietro-Diaz R. & Jones G.A. (1987). Breathing New Life, *Journal of Services and Technology, pp.23-31*.
Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M. & Rosati, R. (2008) Linking data to ontologies. *Journal on Data Semantics, 10:133–173.*
Presutti, V., Daga, E., Gangemi, A. & Blomqvist, E. (2009).eXtreme Design with Content Ontology Design Patterns. *Workshop on Ontology Patterns*, 516 CEUR, Washington D.C., USA.
Priss, U. (2006) Formal concept analysis in information science. *Annual Review of Information Science and Technology, 40 :521–543.*
Psyche V., Mendes O. & Bourdeau J. (2003) Apport de l'ingénierie ontologique aux environnements de formation à distance, revue STICEF, Numéro spécial : Technologies et Formation Ã distance, Vol 10.
Rahm, E. & A-Bernstein, P. (2001) A survey of approaches to automatic schema matching. VLDB J., 10(4) :334–350.
Rector, A. (2005) : Representing Specified Values in OWL: value partitions and value sets. *W3C Woring Group Note 17 May 2005. http://www.w3.org/TR/swbp-specified-values/*
Ribière, M. & Dieng, R. (1997) Introduction of viewpoints in conceptual graph formalism, *Proceeding of the 5th International Conference on Conceptual Structures, Springer-Verlag, Seattle, Washington, pp.168–182.*
Rieu, D., Giraudin, J.P. & Saint Marcel, C.A. (1999) Front- Conte, Des opérations et des relations pour les patrons de conception, *INFORSID'99, La Garde, France, pp.259–277.*
Rodriguez-Castro, B. (2012). Towards ontology design patterns to model multiple classification criteria of domain concepts in the Semantic Web. *Doctorate thesis* of University of Southampton, UK 2012.

Rosati, R. (2011) On conjunctive query answering in EL. *In Proc. of the 2007 International Workshop on Description Logics (DL 2007), volume 250 of CEUR Workshop Proceedings. CEUR-WS.org.*
Rouane-Hacene, M. (2006) Étude de l'analyse formelle dans les données relationnelles : Application à la restructuration des modèles structuraux UML. *Thèse en informatique, Université de Montréal, Faculté des arts et des sciences, Département d'informatique et de recherche opérationnelle.*
Rouane-Hacene, M., Huchard, M., Napoli, A. & Valtchev P. (2013) Relational concept analysis: mining concept lattices from multi-relational data. *In: Ann. Math. Artif. Intell. 67.1, pp. 81–108.*
Rouane-Hacene, M., Fennouh, S., Nkambou, R. & Valtchev, P. (2010) Refactoring of ontologies: improving the design of ontological models with concept analysis, *ICTAI, Vol. 2, pp.167–172.*
Scharffe, F. (2009) Correspondence Patterns Representation. *Doctorate Thesis*. University of Innsbruck, Austria.
Scharffe F., Zamazal O., & Fensel D. (2013). Ontology Alignment Design Patterns. *Knowledge and Information Systems*, Published online 26 April.
Schenk, D. & Wilson, P. (1994) *Information Modelling The EXPRESS Way*. *Oxford University Press.*
Shalloway, A. & Trott, J.R. (2002) Design patterns par la pratique, Eyrolles. 169
Shimizu, C., Hitzler, P., Paul, C.(2018) Ontology Design Patterns for Winston's Taxonomy Of Part-Whole Relations. *ISWC (Best Workshop Papers) 2018*: 119-129.
Shvaiko, P., Euzenat J. (2005). A Survey of Schema-Based Matching Approaches. *Journal on Data Semantics*, 4:146–171.
Sowa, J. F. (1997) Knowledge Representation: Logical, Philosophical and Computational Foundations. *Brooks Cole Publising Co., Pacific Grove, California,* http://www.jfsowa.com/krbook/ .
Stojanovic L. (2004). Methods and Tools for Ontology Evolution, *Ph.D. Thesis*, Karlsruhe University. Germany.
Studer, R., Benjamins, V.R. & Fensel, D. (1998) Knowledge Engineering: Principles and methods. *Data and Knowledge Engineering, 25: 161-197.*
Stumme, G. & Maedche, A. (2001) FCA-MERGE: Bottom-up merging of ontologies. *In International Joint Conference On AI, August 4-10,Seattle, Washington, USA , pp. 225–234.*
Stumme, G., Ehrig, M., Handschuh, S., Hotho, A., Maedche, A., Motik, B., Oberle, D., Schmitz, C., Staab, S., Stojanovic, L., Stojanovic, N., Studer, R., Sure, Y., Volz, R. & Zacharias, V. (2003) The Karlsruhe view on ontologies, *Rapport technique, University of Karlsruhe, Institute AIFB , Germany.*
Sure., Y., Tempich., C & Vrandecic., D. (2006). "Ontology Engineering Methodologies". In Semantic Web Technologies: Trends and Research in Ontology-based Systems, pp. 171-187.
Šváb, O., Svátek, V., Meilicke, C. & Stuckenschmidt, H. (2008) Testing the impact of pattern-based ontology refactoring on ontology matching results, *Proceedings of the 3rd International Conference on Ontology Matching, ACM, Karlsruhe, Germany, pp.229–233.*
Tang, S. & Cai, Z. (2010, Aug) Using the Format Concept Analysis to construct the tourism information ontology. *Seventh International Conference on Fuzzy Systems and Knowledge Discovery; 2010 Aug Yantai Shandong; 6: 2941–4.*
Teguiak, H. V. (2012) Construction d'ontologies à partir de textes : une approche basée sur les transformations de modéles. *PhD thesis*.

Tilley, T., Cole, R., Becker, P., & Eklund, P. (2005). A survey of formal concept analysis support for software engineering activities. *In Formal concept analysis* (pp. 250-271). Springer Berlin Heidelberg.
Valtchev, P. & Missaoui, R. (2001) Building concept (galois) lattices from parts : Generalizing the incremental methods, *pp. 290–303.*
Valtchev, P., Grosser, D., Roume, C. & Rouane-Hacene, M. (2003) Galicia : an open platform for lattices. In Using Conceptual Structures : *Contributions to 11th Intl. Conference on Conceptual Structures (ICCS ?03), pp. 241–254.*
Venable, J.R., Pries-Heje, J. & Baskerville, R. (2016): FEDS: a Framework for Evaluation in Design Science Research Open. *EJIS 25(1)*: 77-89.
Villerd, J. (2008) Représentations visuelles adaptatives de connaissances associant projection multidimensionnelle (MDS) et analyse de concepts formels (FCA). *Thèse de doctorat, École doctorale ICMS de l'École des Mines de Paris.*
Visser, P. R. S., Beer, M. D., Bench-Capon, T. J.M., Diaz, B.M. & Shave, M. J. R.,(1999) Resolving ontological heterogeneity in the kraft project , *in Proceedings of the 10th International Conference on Database and Expert Systems Applications, DEXA '99, (London, UK), Springer-Verlag. pp. 668–677.*
Visser, P.R.S., Jones, D.M., Bench-Caponand T.J.M. & Shave M.J.R. (1997) An Analysis of OntologyMismatches; Heterogeneityversus Interoperability, *Workingnotes of the AAAI 1997 Spring Symposium on Ontological Engineering, Stanford University, California, USA, pp.164-172.*
Volz, J., Bizer, C., Gaedke, M., & Kobilarov, G. (2009). Silk - A Link Discovery Framework for the Web of Data. *in 2nd Workshop of Linked Data on the Web*.
Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. & Hübner, S. (2001, August) Ontology-based integration of information-a survey of existing approaches. *In IJCAI-01 workshop: ontologies and information sharing Vol. 2001, pp. 108-117.*
Wang, L.L., Bhagavatula, C., Neumann, M., Lo, K., Wilhelm, C. & Ammar, W. (2018). Ontology alignment in the biomedical domain using entity definitions and context. *BioNLP 2018, pp. 47-55.*
Warmer, J. & Kleppe, A. (1999) The Object Constraint Language: Precise Modelling with UML, Addison-Wesley, Boston, MA.
Weibel, S. (1997) The Dublin core: a simple content description model for electronic resources. *Bulletin of the American Society for Information Science and Technology, 24: 9–11*.
Wilkinson, K., Sayers, C., Kuno, H. & Reynolds, D. (2003) Efficient rdf storage and retrieval in jena2, *HP Laboratories Technical Report HPL-2003-266, pp. 131–150.*
Winston, M.E., Chaffin, R. & Herrmann, D. (1987). A taxonomy of part-whole relations. *Cognitive Science* 11(4), 417-444.
Xu, L. & Embley, W. (2004) *Combining the Best of Global-as-View and Local-as-View for Data Inte*gration, *In Information Systems Technologies and Its Applications – ISTA, pp.123–136.*