وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار - عنابة

Faculté des sciences de l'ingénieur **Année :** 2011

Département d'informatique

MEMOIRE

Présenté en vue de l'obtention du diplôme de **MAGISTER**

Appariement d'ontologies hétérogènes

Application aux turbines à vapeur

Option

TIC & Ingénierie du document

Par

Saida Gharbi

DIRECTEUR DE MEMOIRE : Med Tarek Khadir Professeur Université de Annaba

DEVANT LE JURY

PRESEDENT :	Farah Nadir	PR	Université de Annaba
EXAMINATEURS:	Seridi Hassina	MC	Université de Annaba
	Bahi Halima	MC	Université de Annaba

ملخص

يستند بناء الانتولوجيا على إعادة استخدام الانتولوجيا ت المتواجدة، وذلك لأن بناءها من نقطة الصفر (من الصفر) هو عملية طويلة ،مكلفة وشاقة ، كما أن الانتولوجيات المتواجدة غير متجانسة، ما دفعنا إلى محاذاة الانتولوجيا التي تعتبر مفتاح لتحقيق التكامل والتوفيق بين هذه الهياكل المختلفة.

الهدف الرئيسي لهذه المذكرة هو اقتراح طريقة لتحديد الروابط الموجودة بين الانتولوجيا ت. اقتراحنا يجمع بين الأساليب والطرائق التي تطابق المصطلحات، الهيكلية، واستخدام المعرفة التكميلية. وهو يتألف من ثلاث مراحل رئيسية : المرحلة الأولى من محاذاة مصدر ي الانتولوجيات المنبع والهدف ،أما الثانية فهي المراساة و الثالثة الاشتقاق. مرحلة المحاذاة تهدف إلى حساب القيم المتشابهة بين مفاهيم كل من الأنتولوجيا المصدر والهدف بإستخد ام الطرق المعتمدة على اللغة أو الهيكلية. مرحلة المرساة تحاذي المفاهيم التي لم يتم محاذاتها في المرحلة السابقة مع الأنتولوجيا المكملة. أما الاشتقاق فيعتمد على نقاط المرساة الناتجة عن المرحلة السابقة المتمثلة في م فاهيم الأنتولوجيا التكميلية و المفكر ليعطينا محاذاة إضافية من أجل تحسين أداء التقنيات التقليدية على المعتمدة على هيكل أو ثراء اللغة الممثلة للانتولوجيا.

وقد نتج عن إقتراحنا نظام جديد للمحاذاة تمت تج ربته على الانتولوجيا ت التي تصف التوربينات البخارية في المجال الصناعي ، واختباره أمام أنظمة أخرى مثل COMA++ ، يسمى هذا النظام BRMAP (**BACKGROUND _RAISONER MAPPING**).

كلمات مفاتيح: *بناء الأنتولوجيا، الأنتولوجيا ، OWL، محاذاة الأنتولوجيا، Taxomap، الأنتولوجيا التكميلية*

Résumé

La construction d'ontologies se fonde sur *la réutilisation* d'ontologies déjà existantes, car la construction d'ontologies à partir de zéro (from scratch) est un processus long, couteux et très laborieux, parallèlement, elle accentue le phénomène de l'hétérogénéité des ontologies. Dans ce contexte, l'alignement des ontologies est la solution pour réaliser l'intégration et le rapprochement de ces différentes structures.

Le principal objectif de ce mémoire est de proposer une approche d'alignement pour identifier les liens de correspondance entre des ontologies. Notre approche combine les techniques et les méthodes d'appariement terminologiques, structurelles et ceux qu'utilisent des connaissances complémentaires. Elle se compose de trois phases principales: la phase d'alignement de deux ontologies source et cible, la phase d'ancrage et la phase de dérivation. La phase d'alignement consiste à calculer les valeurs de similarité entre les concepts de l'ontologie source et ceux de l'ontologie cible en utilisant les mesures terminologiques et structurelles. La phase d'ancrage consiste à apparier chaque concept des deux ontologies source et cible non aligné à la phase précédente avec les concepts de l'ontologie complémentaire. La phase de dérivation s'appuie sur la structuration de l'ontologie complémentaire, les méthodes sémantiques intégrées dans le raisonneur utilisé, les points d'ancrage extraits de l'ancrage, et définit des appariements supplémentaire dans le but d'améliorer le rendement des techniques classiques qui se base sur la structure ou la richesse du langage de représentation des ontologies.

Notre approche a donné lieu à un nouveau système d'alignement appelé BRMAP (BACKGROUND _RAISONER MAPPING). Il a été expérimenté sur des ontologies décrivant la turbine à vapeur dans le domaine industriel, et ainsi testé par rapport aux autres systèmes d'alignement comme COMA++.

Mots clés : *Ingénierie des ontologies, Ontologie, OWL, Appariement des ontologies, Taxomap, Ressource complémentaire*

Abstract

Ontology construction may be based on the reutilization of existing ontologies, as constructing ontologies from scratch is a long, costly and laborious process. Moreover, it accentuates the phenomenon of ontology heterogeneity. In this context, the alignment of ontologies is the key to achieving integration and reconciliation of these different structures.

The main goal of the work is to introduce a method for finding semantic correspondences among ontologies. The approach brings together syntactic, linguistic, structural, and those using *Background* knowledge. The approach consists of three main phases: the alignment phase of source and target ontologies, the anchor phase and finally the derivation phase. The alignment phase computes the similarity values between the concepts of source ontology and concepts of target ontology using the terminological and structural measures. The anchor phase concept matches each concept of the two ontologies source and target, not yet aligned in the previous phase with the concepts of the background ontology. The derivation phase is based on the structuring of the background ontology and the methods integrated into the semantic reasoner, it uses the anchor points extracted from the anchor phase, and identifies additional matches in order to improve performance of conventional techniques based on the structure or the richness of language representation of ontologies.

Our approach has led to a new alignment system called BRMAP (BACKGROUND _RAISONER MAPPING). It was tested on ontologies describing a steam turbine in the industrial field, and thus tested against other systems such as COMA + +.

Keywords: *Ontology Engineering, ontology, OWL, ontology mapping, ontology alignment, Taxomap, Background Knowledge*

Remerciement

Au moment où s'achève la rédaction de ce mémoire, je tiens à remercier tous les gens qui ont contribué à le rendre possible.

Après Dieu, je tiens à exprimer toute ma reconnaissance et mon profond respect à mon encadreur Dr M. Khadir, d'avoir accepté de diriger mon travail, de m'avoir fait confiance et de m'avoir laissé m'exprimer.

Je remercie également madame Seridi Hassina pour m'avoir aidé et soutenu durant ce travail par ses conseils.

Je voudrais remercier monsieur Farah Nadir, maitre de conférence à l'université de Annaba de m'avoir fait l'honneur de présider mon jury.

Je veux également exprimer toute ma gratitude à madame Seridi Hassina et à madame Bahi Halima maitres de conférences à l'université de Annaba qui ont eu la grande gentillesse de mettre leur savoir au service de mon travail en acceptant d'en être les rapporteurs.

Je salue également ma maman, ma sœur et mes frères pour leur soutien et leur amour. Ainsi mes tantes, mes amis, et mes collègues pour leur soutien.

Merci à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail.

Liste des tableaux

Liste des figures

Liste des figures

Table des matières

Table des matières

Table des matières

Table des matières

Introduction générale

1. Contexte et problématique

De plus en plus d'applications industrielles accessibles sur le Web ou par d'autres réseaux, offrent des services fondés sur le partage et l'interopérabilité des structures de connaissances issues de diverses organisations, et représentant divers aspects d'un domaine modélisé. Plusieurs approches existent pour mettre en œuvre cette interopérabilité. Parmi ces approches, on distingue l'alignement d'ontologies, qui consiste à identifier des relations entre éléments de différentes ontologies en recherchant de mappings, appariements ou mises en correspondance. Cette approche est aussi importante dans les systèmes d'intégration puisqu'elle autorise la prise en compte conjointe de ressources décrites par des ontologies différentes.

De plus, ces dernières années la construction d'ontologies se fonde sur *la réutilisation* d'ontologies déjà existantes, car la construction d'ontologies à partir de zéro (from scratch) est un processus long, couteux et très laborieux, parallèlement, elle accentue le phénomène de l'hétérogénéité des ontologies, multipliant le nombre d'ontologies décrivant le même domaine (surtout lorsqu'on sait que l'objectif ultime du web sémantique est d'arriver à instaurer une ontologie de référence pour chaque domaine). Dans ce contexte, l'alignement des ontologies est une solution plausible afin de réaliser l'intégration et le rapprochement de ces différentes structures.

L'alignement se base sur des techniques d'alignement, *Euzenat et al.* (2004) identifient différents types de méthodes d'alignement d'ontologies : terminologiques, structurelles, extensionnelles et sémantiques, qui proviennent de disciplines variées, telles que la fouille de données, les sciences du langage, les statistiques ou la représentation des connaissances. Ces techniques d'alignement tirent parti des différents aspects des ontologies (leur structure, les noms des différents éléments, les objets, la sémantique du langage). Ils sont, pour la plupart, basés sur la recherche d'analogies dans les modèles comparés : concepts identiques ou similaires, structures identiques ou proches, propriétés identiques ou conciliables, etc, [Shvaiko et Euzenat 2005] [Kalfoglou et Schorlemmer 2003]. Ainsi, les techniques mises en œuvre dans ces travaux se trouvent limitées lorsque les ontologies à apparier sont faiblement structurées ou se limitent à de simples hiérarchies de classification.

Pour compléter les techniques classiques d'appariement qui expxloitent la structure ou la richesse du langage de représentation des ontologies, et qui ne s'appliquent plus quand les ontologies à apparier sont faiblement structurées ou se limitent à de simples hiérarchies de classification identifier des mappings entre les concepts de deux ontologies, O_{Src} et O_{Tar}. De nombreux travaux portent actuellement sur l'utilisation de connaissances complémentaires, dites de "background" ou de support, représentées le plus souvent sous la forme d'une 3ème ontologie, *OBK* (voir Aleksovski *et al.*, 2006a, 2006b, Sabou *et al.*, 2006, Kalfoglou & Hu 2005, Reynaud & Safar, 2006, 2007).

L'utilisation de connaissances complémentaires pour aligner deux ontologies passe par la mise en œuvre d'un processus d'alignement dont le schéma général est aujourd'hui bien identifié. Par contre, la façon dont le processus est mis en œuvre et les problèmes posés dépendent en grande partie de la nature des connaissances complémentaires utilisées, dans notre cas il s'agit de connaissances décrivant une turbine à vapeur dans le domaine de l'industrie.

2. Contribution

Nous nous intéressons dans ce mémoire de magister à l'alignement des ontologies décrites en OWL, cet alignement se base sur l'utilisation des techniques classiques de mise en correspondance et ceux exploitant des connaissances complémentaires. Notre proposition se décompose en deux phases, la phase de dérivation qui permet à l'aide d'une représentation formelle des concepts de l'ontologie complémentaire et d'un raisonneur, de découvrir automatiquement des correspondances sémantiques entre les points d'ancrage définis à la phase d'ancrage afin d'améliorer les performances de l'algorithme XMAP [Djeddi et khadir, 2010].

La phase d'ancrage se fonde sur l'utilisation des techniques classiques d'alignement. Ces techniques incluent les méthodes de calcul de similarités basées sur : la comparaison terminologique, la structure des concepts d'ontologies à aligner.

La phase de dérivation s'appuie sur la structuration de l'ontologie *complémentaire* O_{Bk} et *un raisonneur*. Ce dernier intègre des méthodes sémantiques permettent d'effectuer des déductions sur les connaissances d'O_{Bk} par l'application de techniques de raisonnement, lesquelles se base sur des formalismes de représentation des connaissances fondées sur des logiques formelles.

3. Plan du mémoire

Dans ce mémoire nous adoptons le plan qui suit, nous donnons une brève description des contenus respectifs des chapitres le composant.

Le Chapitre 1 : présente les connaissances utilisées dans ce mémoire tell que les principes et les techniques d'alignement des ontologies. En définissant l'ontologie, et le langage sélectionné pour représenter cette dernière. Ensuite nous montrons l'importance d'alignement, les techniques classiques et ceux qui utilisent une connaissance complémentaire pour calculer la similarité entre deux entités dans les différents domaines pour régler le problème de l'hétérogénéité. Nous présentons aussi les travaux traitant ce difficile problème. Enfin, nous présentons le besoin et l'utilité de ces techniques dans le domaine industriel.

Le Chapitre 2 : est consacré à décrire les principes de la logique de description et l'utilité du raisonnement parce que notre approche utilise un raisonneur à la phase de dérivation.

Le Chapitre 3 : illustre l'algorithme de notre proposition qui consiste à mettre en correspondance deux ontologies en utilisant une ontologie complémentaire et un moteur d'inférence. Cet algorithme se décompose en de deux phases : –phase d'ancrage et –phase d'inférence. La première exploite des caractéristiques du langage OWL-DL pour déduire la similarité entre deux entités de deux ontologies et la valeur de similarité est calculée en deux parties : la partie linguistique qui se compose du nom, et la partie structurelle qui se base sur des informations, la présence des propriétés et leurs contraintes de cardinalités. La deuxième utilise un raisonneur pour déduire les relations existantes entre les points d'ancrage, ensuite on déduit la relation entre les entités des deux ontologies.

Le Chapitre 4: décrit l'environnement de développement ainsi que l'implémentation, l'expérimentation et la comparaison de notre outil par rapport à quelques outils d'alignement.

Le Chapitre 5 : est consacré à la conclusion en proposant des perspectives pouvant améliorer notre outil.

Chapitre I

Dans ce chapitre nous présentons des connaissances de base utilisées pour notre thème de recherche. Il s'agit des principes et des technologies de l'alignement des ontologies. Nous commençons par définir l'ontologie, et le langage sélectionné pour représenter cette dernière. Ensuite nous montrons des techniques classiques et ceux qui utilisent une connaissance complémentaire pour calculer la similarité entre deux entités dans le monde du Web sémantique pour régler le problème de l'hétérogénéité. Nous présentons aussi des travaux dans la littérature qu'attaquent ce difficile problème. Enfin, nous présentons le besoin et l'utilité de ces techniques dans le domaine industriel.

I.1. L'ontologie

Au cours des dernières années, les ontologies sont de plus en plus utilisées en informatique, en industrie, en biologie et en médecine, même en gestion des connaissances. Cette évolution a engendré des définitions variées de terme ontologie.

I.1.1. Définitions d'ontologie

En philosophie, l'**ontologie** (du grec ὄν, ὄντος, participe présent du verbe être) est l'étude de l'être en tant qu'être, c'est-à-dire l'étude des propriétés générales de ce qui existe. Par analogie, le terme est repris en informatique et en science de l'information, où [Neches et al., 1991] présente la première définition de l'ontologie dans le domaine de l'informatique : « *Une ontologie définit les termes et les relations de base comportant le vocabulaire d'un domaine aussi bien que les règles pour combiner des termes et les relations afin de définir des extensions du vocabulaire* ».

La première définition pour les ontologies dans le domaine de l'intelligence artificielle est donc donnée par Gruber :

"An ontology is *an explicit specification of a conceptualisation."*

« *une ontologie est une spécification explicite d'une conceptualisation » [Gruber, 1993]*.

- Le terme conceptualisation : fait référence à un modèle abstrait d'une partie de monde réel permettant d'identifier les concepts pertinents de ce monde.
- Le terme explicite : signifie que l'identification de la structure des concepts ainsi que les contraintes sur leur utilisation sont définie d'une manière claire et précise.

Dans [Guarino et Giaretta, 1995], les auteurs essaient de donner une clarification terminologique du terme « ontologie » utilisé dans le domaine de l'intelligence artificielle. Les

sept interprétations possibles du terme « ontologie » sont analysées en se basant sur la notion de la « conceptualisation » qui est définie dans une manière sémantique rigoureuse.

Les auteurs de [Paolo et al., 2003] définissent les ontologies comme « *des modèles partagés d'un domaine encodant une vue qui est commune à un ensemble de différentes parties* »

I.1.2. Les composants d'une ontologie

Comme tout formalisme de représentation, avant d'utiliser les ontologies, il est nécessaire de connaître les composants de base de l'ontologie. Une ontologie peut être vue comme un ensemble structuré de concepts et de relations entre ces concepts destinés à représenter les objets du monde sous une forme compréhensible aussi bien par les hommes que par les machines. Les connaissances traduites par une ontologie sont véhiculées à l'aide d'un certain nombre de constituants ou briques de base qui sont principalement des : concepts, relations, fonctions, axiomes, instances [Gruber, 1993], [Gomez-Perez, 2000].

Les concepts : aussi appelés termes ou classes sont des notions (ou objets) permettant la description d'un concept, d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc. Ils peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs. Habituellement, les concepts sont organisés en taxonomie. Une taxonomie est une hiérarchie de concepts (ou objets) reliés entre eux en fonction de critères sémantiques particuliers.

Les relations : traduisent les liens existant entre les concepts de façon à représenter un type d'interaction entre les concepts d'un domaine. Elles sont formellement définies comme tout sous-ensemble d'un produit de n ensembles, c'est-à-dire R : C1xC2x…..Cn. Des exemples de relations binaires sont : sous-concept-de, sorte-de, etc. Ces relations nous permettent de capturer la structuration ainsi que l'interaction entre les concepts, ce qui permet de représenter une grande partie de la sémantique de l'ontologie.

Les propriétés (ou attributs) : sont des restrictions des concepts ou des relations.

Les fonctions : sont des cas particuliers de relations dans lesquelles le $n^{ième}$ élément de la relation est défini de manière unique à partir des n-1 éléments précédents. Formellement, les fonctions sont définies ainsi : F : c1 x c2 x c3 …….. x c_{n-1} -> c_n

Les axiomes de l'ontologie permettent de définir la sémantique des termes (classes, relations), leurs propriétés et toutes contraintes quant à leur interprétation. Ils sont définis à l'aide de formules bien formées de la logique du premier ordre en utilisant les prédicats de l'ontologie.

Les instances : sont utilisées pour représenter les occurrences.

I.1.3. La construction d'une ontologie

Il existe trois méthodes possibles de création d'une ontologie. Une ontologie peut être construite d'une façon manuelle, automatique ou mixte (semi-automatique).

La construction manuelle : Il existe plusieurs méthodologies offrant des techniques classiques de collecte et d'analyse des connaissances.

La création d'une ontologie d'une manière automatique se base sur des méthodes formelles et des techniques d'extraction des connaissances en employant des outils linguistiques et statiques des méthodes de classification automatique issues de la théorie de l'information ou encore les méthodes de regroupement conceptuel qui segmentent automatiquement les textes en unités thématiquement homogènes, et regroupant ces unités en fonction d'une mesure de similarité fondée sur la fréquence des mots [Ferret et al., 2001]

La construction mixte ou semi-automatique, les ontologies sont construites par des techniques automatiques tout en intégrant des méthodes permettant d'étendre des ontologies ayant été construites automatiquement.

Quel que soit le mode choisi, l'élaboration de toute l'ontologie doit s'appuyer sur un certain nombre de règles et une méthodologie de construction qu'il est nécessaire de se respecter.

I.1.3.1. Les constructeurs

- **Les constructeurs d'ontologie**

Le constructeur d'ontologie permet de regrouper la définition d'un ensemble de concepts qui sont des classes ou des propriétés. Une ontologie définit un domaine d'unicité des noms aussi appelé espace de nom permettant d'identifier les concepts qu'elle définit de manière

unique. Elle est souvent décrite par des informations sur le fournisseur de cette ontologie. Certains modèles d'ontologies tels que OWL fournie également des descripteurs permettant de gérer les versions des ontologies conçues et de les décomposer en modules.

- **Les constructeurs de classes**

Le constructeur de classes est un mécanisme d'abstraction permettant de regrouper un ensemble d'instances présentant des caractéristiques communes. Dans un univers particulier, une classe est associée à un ensemble d'instances appelé son extension. Une classe a une définition intention qui décrit le concept sous-jacent. Cette définition est généralement composée des éléments suivants :

- *Un identifiant* : il permet de référencer cette classe. Cet identifiant est universel et unique. Il est parfois complété par un numéro de version afin de gérer l'évolution des concepts d'une ontologie.
- *Une description textuelle* : elle permet de rattacher une classe à une connaissance préexistante de l'utilisateur. Une ontologie étant une conceptualisation acceptée par une vaste communauté d'utilisateurs, elles sont souvent utilisées dans un contexte international comme par exemple le Web. En conséquence, les définitions textuelles associées aux concepts qu'elle définit sont souvent définies dans plusieurs langues naturelles.
- *Les classes qu'elle généralise et spécialise* : Les classes sont organisées dans une hiérarchie où elles sont liées par une relation de subsomption. La plupart des modèles supportent l'héritage multiple ou permettent de le représenter.

Les éléments précédents permettent de rattacher une classe à un savoir préexistant partagé par le lecteur. Ils définissent également des conditions nécessaires d'appartenance d'une instance à une classe. Ils permettent ainsi de définir une classe primitive (concepts primitifs).

- **Les constructeurs de propriété**

Le constructeur de propriété permet de décrire les instances d'une classe. Les propriétés comme les classes, possèdent toujours un identifiant et une partie textuelle éventuellement définie dans plusieurs langues naturelles. Chacune des propriétés doit être définie sur une classe des instances qu'elle décrit. Cette classe est le domaine de définition de la propriété. Dans certains formalismes, comme par exemple RDF-Schéma, ce domaine peut être l'intersection de plusieurs classes. Il peut également être facultatif.

- **Les constructeurs de types de données**

Le constructeur de type de données permet de définir le codomaine des propriétés d'une ontologie. Les modèles d'ontologie permettent la définition de types simples, principalement

les types entiers, réel, chaine de caractères, booléen et date. Une classe peut également être utilisée comme type de données. Dans ce cas, la valeur d'une telle propriété est l'identifiant d'une instance de la classe formant son codomaine. Enfin, les modèles d'ontologies permettent la définition de type collection dont les éléments sont soit d'un type simple, soit des identifiants d'instances de classes.

- **Les constructeurs d'instances**

Le constructeur d'instance permet de définir l'extension d'une classe dans un certain univers. A l'inverse de ce qui se passe pour les classes, aucun modèle d'ontologie ne fait l'hypothèse d'existence d'un identifiant unique et universel pour les instances. En RDF-Schéma et OWL, l'identifiant est un URI, qui peut être utilisée pour référencer cette instance en dehors de ce système. Cependant, une instance n'a pas forcément un seul identifiant. Ainsi, un même objet du monde réel pourra être identifié différemment dans différents systèmes.

Une instance est définie en indiquant ses classes d'appartenance, elle est également caractérisée par un ensemble de valeurs de propriétés.

I.1.3.2. Les principes de construction d'une ontologie

Le processus de construction d'une ontologie doit respecter certains principes de bases qui permettent d'obtenir une ontologie susceptible de répondre aux objectifs de l'ontologie. Le constructeur de l'ontologie, se doit donc de garder à l'esprit ces principaux critères tout au long du cycle de développement de son ontologie [Gruber,1993] :

La clarté et objectivité : l'ontologie doit fournir le sens des termes définis en offrant des définitions objectives ainsi que de la documentation associée en langage naturel.

L'exhaustivité : une définition exprimée par une condition nécessaire et suffisante est préférable à une définition exprimée seulement par une condition nécessaire ou par une condition suffisante.

La cohérence : afin de pouvoir formuler des inférences cohérentes avec les définitions.

L'extensibilité monotone maximale : les nouveaux termes, qu'ils relèvent de la langue générale ou d'une langue de spécialité, devraient être inclus dans l'ontologie sans entraîner de modifications dans les définitions existantes.

L'intervention ontologique minimale : intervenir le moins possible sur le monde en phase de modélisation, L'ontologie devrait spécifier le moins possible le sens de ses termes, de façon à ce que les parties impliquées dans l'ontologie aient les mains libres pour spécialiser et instancier l'ontologie à leur guise.

I.2. Le langage de représentation des ontologies OWL

Un langage d'ontologie est un langage formel permettant de représenter une ontologie. Pour une représentation simple des ontologies nous pouvons utiliser RDF(S)[1]. Les ontologies en RDF(S) peuvent être sérialisées en des langages tels que XML[2] ou N3[3]. Cependant, une ontologie est employée pour représenter des connaissances dans un domaine, donc il est nécessaire de disposer d'un langage aussi expressif pour les représenter, et RDF(S) ne répond pas à ces besoins. Ainsi, le W3C[4] a recommandé un langage standardisé plus puissant au niveau expressivité, qui est spécialement conçu pour représenter des ontologies dans le Web sémantique. Cela permet avec facilité de créer, partager et échanger des connaissances dans le Web sémantique. Le langage d'ontologie recommandé est le langage OWL. Il est dérivé du langage DAML+OIL[5]. OWL couvre la plupart des caractéristiques du langage DAML+OIL et renomme la plupart de ses primitives.

I.2.1. Définition :

Le langage OWL se compose de trois sous-langages qui proposent une expressivité croissante, chacun conçu pour des communautés de développeurs et des utilisateurs spécifiques :OWL Lite, OWL DL, OWL Full. Chacun est une extension plus simple à son prédécesseur.

Le langage **OWL Lite** répond à des besoins de hiérarchie de classification et de fonctionnalités de contraintes simples, de cardinalité 0 ou 1. Une cardinalité 0 ou 1 correspond à des relations fonctionnelles, par exemple : une personne a une adresse. Toutefois, cette personne peut avoir un ou plusieurs prénoms, OWL Lite ne suffit donc pas pour cette situation.

Le langage **OWL DL** concerne les utilisateurs qui souhaitent une expressivité maximale couplée à la complétude du calcul (cela signifie que toutes les inférences seront assurées e) et la décidabilité du système de raisonnement (c'est- à -dire que tous les calculs seront terminés dans un intervalle de temps déterminé). Ce langage inclut toutes les structures OWL avec certaines restrictions, comme la séparation des types: une classe ne peut pas aussi être un individu ou une propriété. Il est nommé DL car il correspond à la logique descriptive.

1 *www.w3.org/TR/**rdf**-schema/*
2 *www.w3.org/**XML**/*
3 *www.w3.org/DesignIssues/Notation3*
4 http://www.w3.org/
5 http://www.w3.org/TR/daml+oil-reference

Le langage **OWL Full** se destine aux personnes souhaitant une expressivité maximale. Il a l'avantage de la compatibilité complète avec RDF/RDFS, mais l'inconvénient d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie.

OWL est construit sur RDF et RDFS et utilise la syntaxe RDF/XML (il existe d'autres syntaxes OWL).

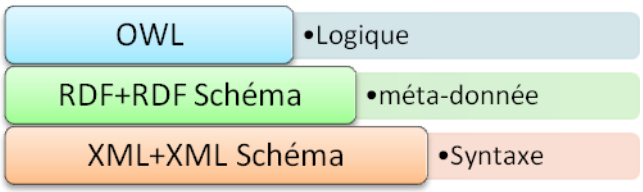

Figure I.1.: l'évolution des langages de représentation des Ontologies

I.2.2. Le choix de la formalisation de l'ontologie par l'OWL :

L'objectif de formalisation d'une ontologie est de faciliter l'implémentation de l'ontologie pour la rendre opérationnelle au sein d'un système informatique .La formalisation consiste en la réécriture de l'ontologie informelle avec le langage de haut niveau OWL, que nous avons choisi pour spécifier les connaissances ontologique.

OWL et spécialement OWL DL est le plus favorisé dans notre cas à cause de son mécanisme de raisonnement basé sur la logique de description et sa syntaxe supportée par XML. OWL a ses fondements dans la logique de description comparé à d'autres formalismes. La description logique a des avantages dont principaux sont :

1. La capacité d'exploiter le raisonnement automatique en se focalisant sur la sémantique nous aide dans notre proposition qu'utilise un raisonneur. Ce formalisme offre à l'utilisateur des définitions intentionnelles pour les concepts qui peuvent être exploités par les classificateurs.

2. La description logique ajoute automatiquement une nouvelle classe dans le bon endroit de la taxonomie tout en détectant l'héritage et en empêchant la contradiction dans les définitions des classes .Cette propriété qui devient importante dans le cas de l'enrichissement de l'ontologie joue un rôle pour faciliter la localisation des concepts qui sont mis à jours.

3. Permettre de réaliser un certain compromis entre l'expressivité et la calculabilité. L'expressivité d'un langage de formalisation de l'ontologie est de permettre de refléter au mieux le sens tourné vers l'utilisateur /calculabilité : Permettre des inférences tournées vers la machine.

4. Les autres formalismes (RDF, RDF (S)) ne permettent pas le raisonnement car ils sont limités.

I.2.3. Des éléments OWL

Le langage OWL fournit des mécanismes pour créer tous les composants d'une ontologie : classes, instances, propriétés et axiomes:

1. Classe définit un groupe d'individus mis ensemble parce qu'ils partagent certaines propriétés. Elle peut être définie par référence (URI vers une classe contenue dans une autre ontologie), par l'énumération de ses instances ou bien par ses propriétés. Par exemple, Amine et Yang appartiennent à la même classe Personne. Une classe peut être aussi définie comme l'union, l'intersection ou le complément d'autres classes. Les classes peuvent être organisées hiérarchiquement selon une taxonomie. Les classes définies par l'utilisateur sont d'ailleurs toutes des enfants de la « super-classe» OWL :Thing. ;

2. Une instance de classe. La définition d'un individu consiste à énoncer un « fait ». On peut distinguer deux types de faits :

i) *les faits concernant l'appartenance à une classe* : La plupart des faits concernent généralement la déclaration de l'appartenance à une classe d'un individu et les valeurs de propriétés de cet individu.

ii) *les faits concernant l'identité des individus* : Une difficulté qui peut éventuellement apparaître dans le nommage des individus concerne la non-unicité éventuelle des noms attribués aux individus. Par exemple, un même individu pourrait être désigné de plusieurs façons différentes. C'est la raison pour laquelle OWL propose un mécanisme permettant de lever cette ambiguïté, à l'aide des propriétés owl :sameAs, owl :diffrentFrom et owl :allDifferent[10].

3. ***Propriétés*** permet de définir des faits ou des relations entre ces classes. Il existe en OWL deux types de propriétés :

i) *propriété d'objet* (owl :ObjectProperty) qui définit une propriété entre deux individus d'une classe ou de plusieurs classes, c'est à dire une relation ;

ii) *une propriété de type de données* (owl :DataTypeProperty), c'est à dire une relation entre une valeur ou donnée et un individu d'une classe. Aussi, les propriétés peuvent être organisées hiérarchiquement.

Un document OWL se compose en général :

i)d'un espace de nommage ;

ii) d'une en-tête (owl :Ontology) pour décrire le contenu de l'ontologie ;

iii) de la définition de classes ;

iv) de la définition des propriétés et enfin d'assertion de fait.

I.2.4. Domaine d'application D'OWL

La flexibilité apportée par OWL représente un enjeu majeur sur bien des domaines. Notamment, pour la représentation des ontologies, et la capacité d'inférer sur ces derniers. En effet, les ontologies ainsi représentées peuvent valoriser le fonctionnement du Web de plusieurs façons. Notamment, pour améliorer la pertinence des moteurs de recherches ; en faisant référence à un concept précis au lieu d'utiliser des mots-clés ambigus [Elizabet et Pezet]. Aussi, la capacité des agents à comprendre les données manipulées offre de nouvelles perspectives dans le domaine de l'automatisation ; On citera, la composition automatique de services Web[6] et la composition automatique de workflow[7].

L'introduction d'une ontologie dans un système d'information vise à réduire, voire éliminer, la confusion conceptuelle et terminologique et à tendre vers une compréhension partagée pour améliorer la communication, le partage, l'interopérabilité et le degré de réutilisation possible. OWL offre un cadre unificateur et fournit des « primitives », des éléments de base pour améliorer la communication entre les personnes, entre les personnes et les systèmes, et entre les systèmes. Intégrer une ontologie écrite en OWL à un système d'information permet donc de déclarer formellement un certain nombre de connaissances utilisées pour caractériser les informations gérées par le système et de se baser sur ces caractérisations et la formalisation de leur signification pour automatiser des tâches de traitement de l'information [Fabien Gandon, 2006].

6 Un service Web est un ensemble de protocoles et de normes informatiques utilis´es pour ´echanger des donn´ees entre les applications.

7 On appelle « workflow » (traduisez littéralement « flux de travail ») la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier (aussi appelé processus opérationnel ou bien procédure d'entreprise).

En somme, les champs d'application éventuels sont vastes : raisonnement automatique, représentation de données structurées, traduction automatisée, résolution de problèmes par inférences.

I.3. Comment est né le besoin à l'alignement des ontologies ?

La notion d'ontologie est devenue un élément clé dans toute une gamme d'applications faisant appel à des connaissances. Une ontologie est définie comme la conceptualisation des objets reconnus comme existant dans un domaine, de leurs propriétés et des relations les reliant. La structure d'une ontologie permet de représenter les connaissances d'un domaine sous un format informatique en vue de les rendre utilisables pour différentes applications.

Cependant, comme n'importe qu'elle autre représentation explicite des connaissances, une ontologie dépend toujours d'assomptions implicites comme les objectifs des concepteurs, leurs capitaux connaissances, rendant l'objectivité de la représentation un but difficile à concrétiser. Ces connaissances implicites sont à l'origine de différentes formes d'hétérogénéité entre les ontologies, même entre les ontologies décrivant le même domaine. Le choix d'une ontologie particulière ou l'exploitation de plusieurs d'entre elles devient difficile, ce que nécessite une comparaison entre eux afin de passer de l'une à l'autre ou de les intégrer, en générant le plus automatiquement possible des relations ou appariements entre les concepts de deux ontologies.

La génération de ces relations rencontre un problème difficile qu'on appelle l'hétérogénéité. Il existe plusieurs classifications d'hétérogénéité entre les ontologies [Benerecetti et al., 2000]; [Klein, 2001]; [Euzenat, 2001]; [Corcho, 2004]; [Hameed et al., 2004]; [Ghidini et Giunchiglia., 2004]. Dans ce qui suit, Nous présentons une classification [Euzenat et al., 2007b] qui résume les principales formes d'hétérogénéité en quatre niveaux, à savoir :

1. Le niveau syntaxique : Il s'agit de toutes les formes d'hétérogénéité relatives au choix du format de représentation. En effet il existe différentes façons de représenter les ontologies (OWL, KIF …) et chaque langage est basé sur une syntaxe différente.

2. Le niveau terminologique : l'hétérogénéité, à ce niveau, intervient dans le fait d'affecter des noms aux entités (classes, propriétés, relations …) qui constituent une ontologie. Nommer une entité revient à lui associer un objet linguistique à partir d'un langage public. Des exemples de ce type d'hétérogénéité :

- Différents noms utilisés pour désigner une même entité (synonymie)
- Le même nom utilisé pour désigner deux entités distinctes (polysémie)
- Au contraire, le même terme peut représenter différent concepts ; l'homonymie est un problème qui nécessite bien souvent l'intervention humaine.
- Des mots provenant de différentes langues (Français, Anglais, Italien …) utilisés pour désigner une même entité.
- Des variations syntaxiques du même mot (différentes prononciations, abréviations, utilisation des préfixes et des suffixes …)
- Finalement, l'encodage des données au sein de l'ontologie diffère bien souvent, que ce soit pour les dates, les unités (monnaie, distances, ...).

3. Le niveau conceptuel: Les divergences à ce niveau peuvent être résumées en trois aspects :

- *La couverture* : la différence entre deux ontologies peut être au niveau de la portée de la couverture du domaine décrit. Elles peuvent couvrir des parties différentes (du monde réel ou d'un domaine) ou alors des parties qui se chevauchent, par exemple: une ontologie sur le sport couvre le sport de la course automobile qu'une autre ignorerait complètement.
- *La granularité* : deux ontologies peuvent décrire les mêmes entités avec des niveaux de détail différents, par exemple : une ontologie concernée par la comptabilité va considérer le concept générique du document alors qu'une ontologie décrivant le domaine des bibliothèques va distinguer entre les différents types de documents : romans, nouvelles, biographies, manuscrits …
- *La perspective* : deux ontologies peuvent décrire un domaine de deux points de vue différents. Par exemple : le concept de la chaleur chez un Norvégien sera forcément différent du même concept chez un Sénégalais.

La figure suivante est une représentation graphique de ces trois dimensions, à travers lesquelles une ontologie peut différer d'une autre ontologie au niveau conceptuel :

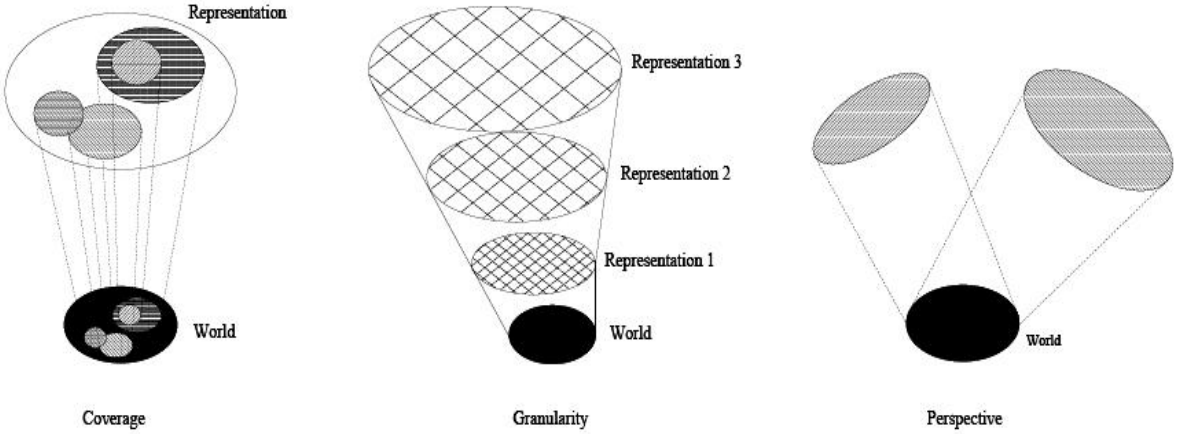

Figure I.2. Les trois dimensions de l'hétérogénéité au niveau conceptuel

4. Le niveau sémiotique ou pragmatique: ce type d'hétérogénéité intervient lorsqu'il y a une différence d'interprétation de la même ontologie par différentes personnes ou différentes communautés. Ces différences d'interprétations sont souvent liées au choix du formalisme de représentation des connaissances, par exemple, des clauses de la logique du premier ordre et une représentation hiérarchique de classes, sont-ils équivalents, est-ce qu'ils véhiculent la même connaissance ?

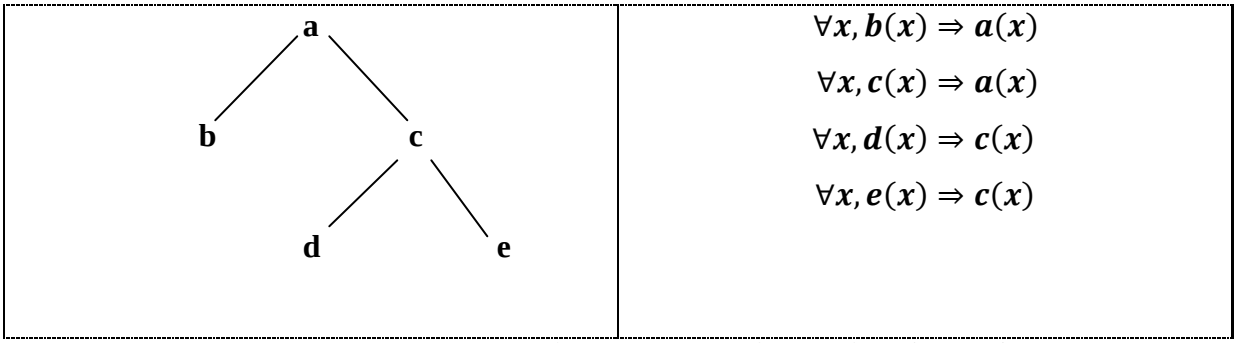

Figure I. 3. Est-ce que ces représentations traduisent la même information ?

En résumé, comprendre les différentes formes d'hétérogénéité des ontologies est primordial pour la réussite de l'alignement de ces dernières, car il est très risqué de réaliser un alignement entre des entités, en se basant seulement sur les liens sémantiques, les autres dimensions doivent être prises en compte, par exemple : [zanobini, 2003]

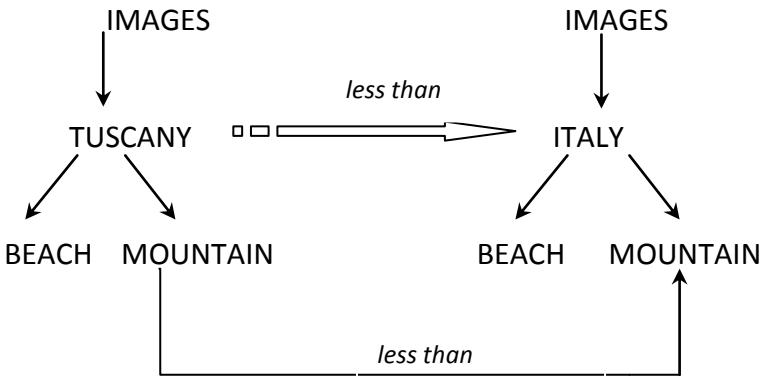

Figure I.4. Exemple de relations sémantiques

Le concept « Mountain » dans la première hiérarchie décrit les images de montagnes en toscane alors que le concept « Mountain » dans la seconde hiérarchie décrit les images de montagnes en Italie, cela ne veut pas dire qu'ils ne sont pas liés, ils ne sont pas équivalents certes, mais, en tenant compte de la structure, il apparaît que le premier concept est moins général que le second (la divergence, ici, est liée à la couverture et donc elle est d'ordre conceptuelle).

I. 4. Alignement des ontologies :

I.4.1.Définition de l'alignement :

« L'alignement de structures est le processus de mise en correspondance sémantique[8] *des entités qui les composent***. »** [Euzenat et al., 2007b]

Ces structures peuvent être des ontologies, des schémas XML ou des bases de données. Les liens sémantiques comprennent les relations : d'équivalence (=), de généralisation/spécialisation(,), de chevauchement (∏) ou encore d'incompatibilité ( ). L'évaluation de la véracité de ces liens peut être booléenne ou par le biais d'autres mesures telles que : les probabilités, les mesures symboliques, les mesures de similarité.

I.4.2. Le processus de l'alignement

Ce processus peut être considéré comme une fonction f, qui à partir d'une paire d'ontologies O et O', un alignement en entrée A (optionnel), un ensemble de paramètres p (ex: paramètres de pondération, seuils …) et un ensemble de ressources externes r (ex : thésaurus,

8 Liens basés sur **la signification** des termes.

lexique…), détermine un alignement A' entre ces deux ontologies : **A' = f (O, O', A,** p, r**).** Ceci peut être représenté schématiquement de la manière suivante :

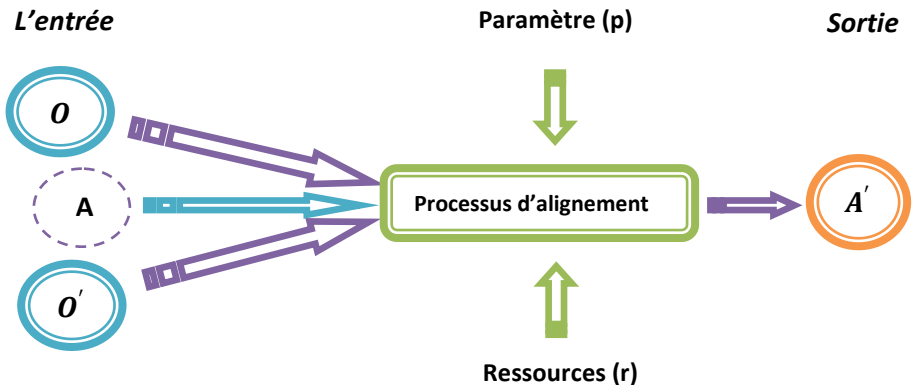

Figure I.5. Processus d'alignement

Le processus d'alignement est exécuté selon *une stratégie* ou une combinaison de techniques d'alignement de base décrites dans la section 5.

I.4.2.1. L'entrée : est constitué essentiellement des structures destinées à être alignées et qui peuvent être, comme énoncé précédemment, des schémas XML, des schémas relationnels, des ontologies. Notre présent travail, se situe dans le cadre de l'alignement des ontologies «ontology matching[9]» qui diffère à plusieurs égards, de l'alignement des schémas « schema matching », mais néanmoins reste assez proche, comme le montre le tableau ci-dessous :

Schema Matching(l'alignement des schémas)	Ontology Matching(l'alignement des ontologies)
<u>Différences</u>	
Souvent, Les données des schémas ne sont pas porteuses de sémantique explicite.	Les ontologies sont des systèmes logiques porteurs de définitions (à travers les axiomes).
Les schémas (relationnels, par exemple) ne fournissent pas de généralisation.	Les définitions des ontologies sont un ensemble d'axiomes logiques aptes à être généralisés par rapport à un contexte donné.
Les ontologies sont plus riches que les schémas, par exemple OWL permet la définition de nouvelles classes à travers les unions et les intersections.	
<u>Points communs</u>	
L'alignement de schémas ou d'ontologies vise à identifier les relations sémantiques (subsomption, équivalence, etc.) existantes entre les entités constituantes (concepts, relations) issues de deux structures.	
Les schémas et les ontologies comprennent des vocabulaires de termes qui décrivent le domaine d'intérêt.	

9 Sites web dédiés à ce domaine : http://www.ontologymatching.org, http://www.atl.lmco.com/projects/ontology

Les schémas et les ontologies sont porteurs de définitions des termes du vocabulaire utilisé.
Les ontologies peuvent être considérées comme des schémas de bases de connaissances.

Tableau I.1. «Schema matching» vs. «Ontology matching»

[Shvaiko et Euzenat., 2005a] concluent ce tableau comparatif en disant ceci: « *Techniques developed for both problems are of a mutual benefit* ».

Remarque : l'input peut être enrichi par un alignement en entrée (qui aurait besoin d'être complété par une nouvelle itération d'alignement).

I.4.2.2. La sortie : est un ensemble d'alignements reliant les entités qui composent les deux ontologies. Un alignement est décrit comme un ensemble de cinq éléments : < **id, e, e', r, n** > telle que :

- **id**: identifiant unique d'un mapping.
- **e** : une entité, à aligner, appartenant à O (classe, propriété, contrainte, instance).
- **e'** : une entité, à aligner, appartenant à O'.
- **r** : la relation qui relie **e** à **e'** (⊆, ∏,).
- **n** : la mesure de confiance de la relation **r**, généralement une valeur réelle comprise dans l'intervalle [0,1]. Plus le **n** est proche du 1, plus la relation est considérée comme étant forte.

L'output est caractérisé par :

a) ***La multiplicité*** (contraintes sur les relations entre les entités des deux ontologies). Si on considère, d'une part, les valeurs suivantes:

1 : une et une seule relation ? : de 0 à 1 relation + : de 1 à plusieurs relations * : de 0 à plusieurs relations	à partir d'une entité d'ontologie

D'autre part, les deux orientations possibles d'un alignement entre deux ontologies (**O** **O'** et **O'** **O**), la multiplicité peut prendre les valeurs suivantes :

1 :1, 1 :?, ?:1 , 1 :+, +:1 , 1 :*, *: 1, ?:?, ?:+, +:?, ?:*, * :?, + :*, * :+, + :+, * :*.

Voici quelques exemples sur les configurations de multiplicité entre deux ontologies, constituée chacune de trois entités [Euzenat et shvaiko., 2007].

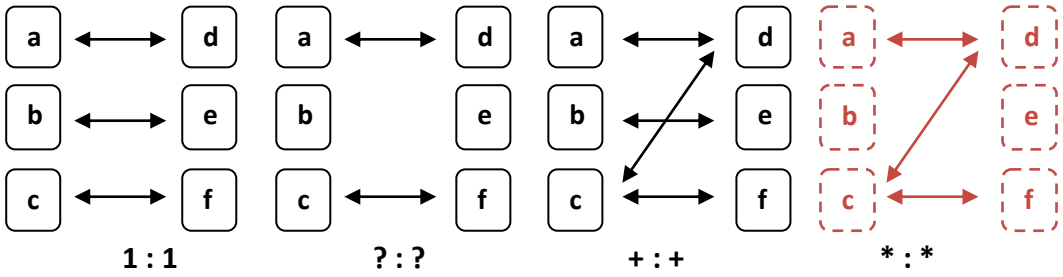

Figure I.6. Exemples de configurations de multiplicité entre 2 ontologies

b) Le niveau de l'alignement :

- ***Le niveau (0) :*** Les alignements concernent des entités identifiées par des URIs.
- ***Le niveau (1) :*** Les alignements de paires d'entités sont remplacés par des paires d'ensembles d'entités.
- ***Le niveau (2) :*** des correspondances plus générales, que les niveaux précédents, peuvent être utilisés.

c) le format de l'output : il existe plusieurs possibilités de représenter l'output, dont :

- *OWL* : grâce aux primitives suivantes : equivalentClass, equivalentProperty, subClassOf, subPropertyOf, exemple :

```
<owl : Property rdf : about= "&onto1;#author">
<owl:equivalentProperty rdf:resource="&onto2;#author"/>
</owl:Property>
<owl:Class rdf:about="&onto1;#Book">
<owl:equivalentClass rdf:resource="&onto2;#Volume"/>
</owl:Class>
<owl:Class rdf:about="&onto2;#title">
<owl:subclass ="&onto1;#name"/>
</owl:Class>
```

- *Contextualized OWL (C-OWL)* : est une extension de OWL pour exprimer les mappings entre ontologies hétérogènes [Bouquet et al., 2003, Bouquet et al., 2004], la nouveauté dans ce langage, par rapport à OWL, est le concept de « bridge rules » qui permettent l'expression de relations entre les classes, les relation et les instances interprétées dans des domaines hétérogènes. « bridge rules » sont des correspondances orientées d'une ontologie source vers une ontologie destination, ils utilisent les symboles

des relations sémantiques (précédemment citées). Les « bridge rules » sont interprétés à partir de l'ontologie destination, ils expliquent comment l'ontologie source est traduite selon l'ontologie destination. C-OWL est conçu aussi pour la représentation d'*ontologies contextuelles* ou *contextualisées* [Bouquet et al., 2004]. Les ontologies contextuelles sont des représentations locales, appelées *contextes*, qui sont en relation avec d'autres contextes par l'intermédiaire d'*appariements*.

I.5. Techniques d'alignement

Dans cette partie, nous allons examiner les techniques et les méthodes utilisées dans la littérature qu'attaquent au problème de recherche de la similarité, ou de la correspondance entre deux entités en général, qu'elles apparaissent dans des schémas, ou dans des ontologies représentées soit en RDF(S), soit en OWL.

I.5.1 Les méthodes de base pour mesurer la similarité

Avant de présenter ces méthodes, nous donnons la définition de la similarité.

I.5.1.1 La similarité

La notion de similarité dans notre contexte n'est pas celle que l'on peut trouver en psychologie ou en mathématiques. En psychologie sociale, la similarité se rapporte à comment les attitudes, les valeurs, les intérêts et la personnalité correspondent entre les personnes.

En mathématiques, plusieurs relations d'équivalence (qui sont des relations binaires réflexives, symétriques et transitives) sont appelées la similarité. Ces relations existent par exemple :

(i) En géométrie : Deux objets géométriques sont similaires si l'un est isométrique avec le résultat d'un agrandissement ou rétrécissement uniforme de l'autre.

L'un peut être obtenu à partir de l'autre uniformément élargi, rétréci, avec une rotation éventuelle (tous les deux ont la même forme), ou en plus, en appliquant un effet du miroir (l'un a la même forme que l'image de miroir de l'autre). Par exemple, tous les cercles sont similaires entre eux, tous les carrés sont similaires entre eux, et toutes les paraboles sont similaires entre elles. D'autre part, ni les ellipses, ni les hyperboles ne sont similaires entre elles. Deux triangles sont semblables si et seulement si ils ont les mêmes 3 angles.

(ii) En algèbre linéaire, deux matrices A et B de taille n × n sont dites similaires s'il existe une matrice inversible P de même taille n × n satisfaisant $P^{-1}A\ P = B$.

(iii) En topologie, la similarité est une fonction telle que sa valeur est plus grande quand deux points sont plus proches (contrairement à la distance, qui est une mesure de dissimilarité : plus les points sont proches, plus la distance est petite).

Dans notre contexte, la notion de similarité sémantique est vue comme celle de la similarité topologique en mathématiques, où on l'associe à une fonction, appelée fonction de la similarité. La définition de cette fonction de la similarité peut changer selon les approches, selon les propriétés souhaitées. La valeur de cette fonction est souvent comprise entre 0 et 1, ce qui permet des possibilités d'interprétation probabiliste de la similarité. Des propriétés ou des caractéristiques communes possibles de la fonction sont des caractéristiques positives, auto similaires ou maximales, symétriques ou réflexives. On peut aussi trouver d'autres caractéristiques telles que la finitude ou la transitivité.

Définition 1 *(Similarité). La similarité* $S: O \times O \rightarrow R$ *est une fonction d'une paire d'entités à un nombre réel exprimant la similarité entre ces deux entités telle que :*

- $\forall a, b \in O, S(a,b) \geq 0$ $(positivité)$
- $\forall a, b, c \in O,\ S(a,a) \geq S(b,c)\ et\ S(a,a) = S(a,b) \leftrightarrow a = b$ $(autosimilarité\ ou\ maximalité)$
- $\forall a, b \in O,\ S(a,b) = S(b,a)$ $(symétrie)$
- $\forall a, b, c \in \mathrm{O}, \mathrm{S(a,b)} = \mathrm{S(b,c)} \rightarrow \mathrm{S(a,b)} = \mathrm{S(a,c)}$ (transitivité)
- $\forall a, b \in O, S(a,b) \leq \infty$ $(finitude)$

La dissimilarité est parfois utilisée au lieu de la similarité. Elle est définie de manière analogue à la similarité, sauf qu'elle n'est pas transitive :

Définition 2 *(Dissimilarité). La dissimilarité* $DS: O \times O \rightarrow R$ *est une fonction d'une paire d'entités à un nombre réel exprimant la dissimilarité entre ces deux entités telle que :*

- $\forall a, b \in O, DS(a,b) \geq 0$ $(positivité)$
- $\forall a, b, c \in O, DS(a,a) \leq DS(b,c)\ et\ DS(a,a) = 0$ $(minimalité)$
- $\forall a, b \in O, DS(a,b) = DS(b,a)$ $(symétrie)$
- $\forall a, b \in O, DS(a,b) \leq \infty$ $(finitude)$

La distance est une mesure utilisée aussi souvent que les mesures de similarité. Elle mesure la dissimilarité de deux entités, elle est inverse de la similarité : si la valeur de la fonction de similarité de deux entités est élevée, la distance entre elle est petite et vice-versa. Elle est donc définie dans [Euzenat et al., 2004] comme suit :

Définition 3 *(Distance). La distance* $D: O \times O \to R$ *est une fonction de la dissimilarité satisfaisant la définitivité et l'inégalité triangulaire :*

- $\forall a, b \in O, D(a,b) = 0 \leftrightarrow a = b$ $(définitivité)$
- $\forall a, b, c \in O, D(a,b) + D(b,c) \geq D(a,c)$ $(inégalité\ triangulaire)$

Les valeurs de similarité sont souvent normalisées pour pouvoir être combinées dans des formules plus complexes. Si la valeur de similarité et la valeur de dissimilarité entre deux entités sont normalisées, notées $\bar{S}$ et $\overline{DS}$, alors on a $\bar{S} + \overline{DS} = 1$

Définition 4 *(Normalisation). Une mesure est une mesure normalisée si les valeurs calculées par cette mesure ne peuvent varier que dans un intervalle de 0 à 1. Ces valeurs calculées sont appelées valeurs normalisées. Les fonctions du calcul sont appelées fonctions normalisées et notées* $\overline{f}$.

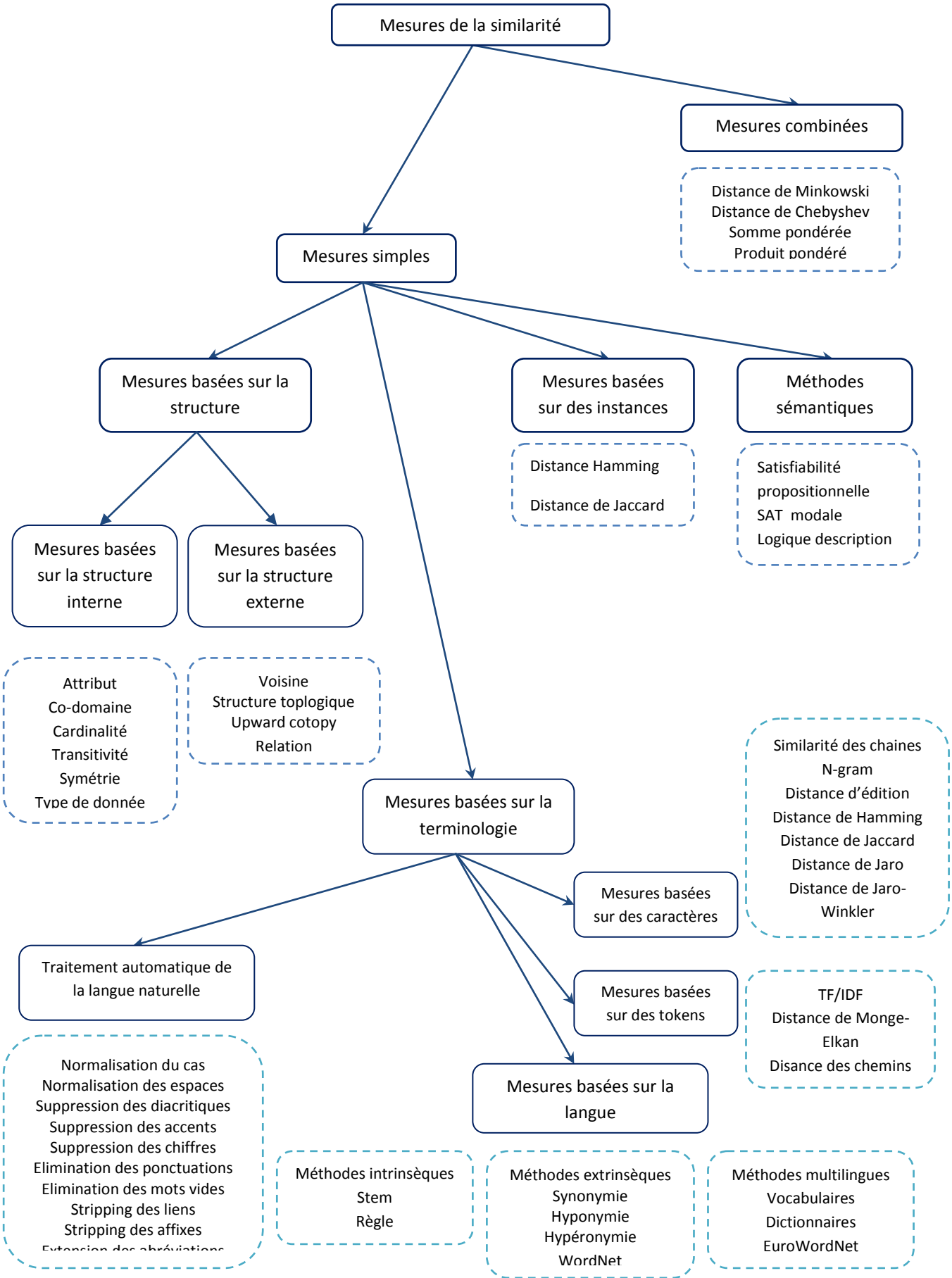

Figure I.7. Les catégories des mesures de similarité selon différentes techniques

Les mesures de la similarité, de la dissimilarité, de la distance peuvent être classées selon la nature des entités que l'on veut comparer : des termes, des chaînes de caractères, des structures, des instances (des individus des classes), des modèles théoriques.

La Figure I.7 résume différentes mesures des similarités, catégorisées selon les techniques utilisées. Ce résumé est une synthèse des travaux présentés dans [Rahm et Bernstein., 2001], [Euzenat et al., 2004] et [Shvaiko et Euzenat., 2005a].

I.5.1.2 Les méthodes terminologiques

Ces méthodes se basent sur la comparaison des termes ou des chaînes de caractères ou bien les textes. Elles sont employées pour calculer la valeur de la similarité des entités textuelles, telles que des noms, des étiquettes, des commentaires, des descriptions…etc. Ces méthodes peuvent encore être divisées en deux sous-catégories : l'une contient des méthodes qui comparent des termes en se basant sur les caractères contenus dans ces termes et l'autre utilise certaines connaissances linguistiques.

I.5.1.2.1 Les méthodes basées sur des chaînes de caractères

Ces méthodes analysent la structure des chaînes de caractères, l'ordre des caractères dans la chaîne, le nombre d'apparitions d'une lettre dans une chaîne pour concevoir des mesures de la similarité. Par contre, elles n'exploitent pas la signification des termes. Par exemple, les mesures dans cette catégorie retournent une grande valeur de similarité (jusqu'à 1) si elles comparent les termes « Voiture » et « voitures », mais une petite valeur, voire la valeur 0, si elles comparent les termes « voiture » et « bagnole ».

Les résultats de la comparaison des chaînes de caractères seront améliorés si ces chaînes sont « nettoyées » ou traitées avant de les fournir aux formules calculant la similarité. Cette phase est appelée la phase de normalisation ou de normalisation textuelle, qui diffère de la normalisation des valeurs de similarité dans un intervalle de [0,1] discutée dans la définition4. Les différents types de normalisation textuelle sont ceux empruntés au domaine de traitement automatique de la langue naturelle (TALN) :

- Normalisation des caractères : ce type de normalisation convertit toutes les majuscules dans une chaîne de caractères en leurs formes minuscules ou vice-versa. Par exemple, la chaîne de caractères « VoitureS » sera convertie à « voitures » et

ensuite, elle est considérée comme égale exactement à l'autre chaîne de caractères « voitures ».

- Normalisation des espaces : ce type de normalisation remplace toutes les séquences consécutives des espaces, des tabulations, des retours de chariot (les caractères CR) trouvées dans une chaîne de caractères par un seul caractère d'espace. Par exemple, l'expression « ma voiture » est normalisée à « ma voiture ».

- Suppression des signes diacritiques ou des accents (aigus, graves…) : ce type de traitement remplace des caractères avec des signes diacritiques par caractères correspondants sans signes diacritiques. Par exemple, le mot « Hanoï » est remplacé par le mot « Hanoi » sans changer la signification du mot, la capitale du Vietnam. Cependant, certaines suppressions changeront la signification du terme : « là » (adverbe de lieu) et « la » (article).
- Suppression des chiffres.
- Élimination des ponctuations.
- Élimination des mots vides (les mots contenant peu d'informations tels que « est », « un », « les »…).
- Suppression des affixes (préfixes, suffixes).
- Extension des abréviations.
- Tokenisation.
- Lemmatisation (passer au singulier, à l'infinitif pour les verbes, au masculin pour les adjectifs…).

Il existe plusieurs mesures calculant la valeur de similarité ou la distance entre deux chaînes de caractères dans la littérature telles que la similarité de Jaccard, la distance de Hamming, la distance de Levenshtein ,le n-gram. Certaines sont implémentées en Java[10], en C[11]. Nous présentons ici quelques mesures les plus utilisées dans les approches d'alignement d'ontologies dans le cadre du Web sémantique.

10 http://www.dcs.shef.ac.uk/~sam/stringmetrics.html
11 http://www.monkey.org/~jose/software/libdistance/distance_3.html

Si nous considérons une chaîne de caractères comme un ensemble de caractères S, la similarité de Jaccard entre deux chaînes est définie ainsi :

Définition 5 *(Similarité de Jaccard). Soit s et t deux chaines de caractère. Soit S et T les ensembles des caractères de s et t respectivement. La similarité de Jaccard est une fonction de la similarité* $S_{Jaccard} : S \times S \rightarrow [0,1]$ *telle que :*

$$\overline{S_{Jaccard}}(s,t) = \frac{|S \cap T|}{|S \cup T|} \quad (1)$$

Une classe importante des fonctions mesurant la distance entre deux chaînes de caractères s et t, est constituée des fonctions calculant la distance d'édition (edit distance), dans lesquelles la distance est le coût de la meilleure séquence des opérations d'édition qui convertissent s en t. Des opérations d'édition typiques sont celles de l'insertion, de la suppression, et de la substitution de caractère, et à chaque opération est affecté à un coût.

Définition 6 *(Distance d'édition). Soit un ensemble d'opérations d'édition* $OP, op \in OP, op: S \rightarrow S$ *et une fonction de coût d'édition* $w: OP \rightarrow R$*. Pour n'importe quelle paire des chaînes de caractères, il existe une séquence des opérations d'édition qui transforme la première en la seconde (et vice versa), la distance d'édition de deux chaînes de caractères s et t est une fonction de la dissimilarité* $DS_{de} : S \times S \rightarrow [0,1]$ *telle que* $DS_{de}(s,t)$ *est le coût de la séquence des opérations la moins coûteuse qui transforme s en t.*

$$\overline{DS_{de}}(s,t) = min_{(op_1)_1 : op_n(\ldots op_1(s))=t} \left(\sum_{k=1} w_{op_i}\right) \quad (2)$$

Les variantes de cette mesure sont différentes au niveau des coûts assignés aux opérations d'édition. On peut trouver :

- La distance de Levenstein où tous les coûts sont égaux à 1.
- La distance de Damerau qui est presque identique à la distance de Levenshtein mais qui peut tolérer des caractères adjacents qui ont été permutés, une erreur typographique habituelle.
- La distance de Smith-Waterman [Durban et al. 1998] qui affecte des coûts relativement inférieurs à la séquence des insertions ou des suppressions.
- La distance de Needleman-Wunsch avec des matrices des coûts pour chaque paire des caractères dans des opérations des insertions ou des suppressions.

La métrique Jaro [Jaro, 1995; 1989] produit la similarité entre deux chaînes de caractères en se basant sur le nombre et l'ordre des caractères communs entre elles.

Définition 7 *(Distance de Jaro). Soit s et t deux chaînes de caractères. Soit N_c le nombre des caractères communs apparaissant dans les deux chaînes dans une distance de moitié de la longueur de la chaîne la plus courte. Soit N_t le nombre des caractères transposés, qui sont des caractères communs apparaissant dans des positions différentes. La distance de Jaro est une fonction de la dissimilarité $DS_{Jaro} : S \times S \rightarrow [0,1]$ telle que :*

$$DS_{Jaro}\,(s,t) = 1 - \frac{1}{3}\left(\frac{N_c}{|S|} + \frac{N_c}{|t|} + \frac{N_c - N_{t/2}}{N_c}\right) \quad (3)$$

Il existe aussi des distances qui sont des variantes de la distance de Jaro, telles que la distance Jaro-Winkler [Winkler, 1999] :

Définition 8 *(Distance de Jaro-Winkler). Soit s et t deux chaînes de caractères. Soit P la longueur du préfixe commun le plus long de s et t. Soit n un nombre positif. La distance de Jaro-Winkler est une fonction de la dissimilarité $DS_{JaroWinkler} : S\ x\ S \rightarrow [0,1]$ telle que :*

$$\overline{\mathrm{DS}_{JaroWinkler}}\,(s,t) = \overline{DS_{Jaro}}\,(s,t) - \frac{\max(P,n)}{10}\,\overline{DS_{Jaro}}\,(s,t) \quad (4)$$

I.5.1.2.2 Les distances basées sur des tokens

Les mesures présentées ci-dessus s'adaptent bien lorsque l'on veut comparer deux termes ou deux courtes chaînes de caractères. Il existe aussi des cas où l'on a besoin de comparer des textes longs ou bien des documents textuels. Dans ces cas, ces entités sont découpées en plusieurs morceaux, appelés tokens. Elles deviennent des ensembles des tokens, et la similarité entre elles est produite grâce aux mesures de similarité basées sur des tokens.

Il existe plusieurs mesures de cette catégorie dans la littérature telles que la similarité de Dagan [Dagan et al., 1999], la distance de Jensen-Shannon, la distance de Fellegi-Sunter [Fellegi et Sunter., 1969]… Nous présentons ici quelques mesures les plus utilisées dans les approches d'alignement d'ontologies dans le cadre du Web sémantique.

La similarité de Jaccard (Définition 5) peut être étendue pour comparer des ensembles des tokens, en définissant la similarité comme le rapport entre la cardinalité de l'intersection des ensembles sur la cardinalité de leur union.

Une mesure qui est largement employée dans le domaine de la recherche d'information, semble convenable ici. Il s'agit du TF/IDF (Term frequency/Inverse Document Frequency). Dans sa conception originale, TF/IDF est employé pour mesurer la pertinence d'un terme dans l'ensemble de documents. La fréquence de terme, TF, dans un document donné montre l'importance de ce terme dans le document en question. La fréquence inverse de document, IDF, est une mesure de l'importance générale du terme dans l'ensemble de documents.

Définition 9 *(TF/IDF). Soit D un corpus des documents, |D| dénote le nombre des documents dans le corpus D. Soit t un terme à considérer, n(t) étant le nombre d'occurrences du terme t dans un document, et N étant le nombre des termes dans ce document, et d(t) étant le nombre des documents qui contiennent au moins une fois le terme t. Les mesures de TF et TF/IDF sont définies comme suivante :*

$$TF = \frac{n(t)}{N} \; et \; {}^{TF}/_{IDF} = TF \times \log(\frac{|D|}{d(t)}) \tag{5}$$

La similarité des entités textuelles peut être construite comme la valeur cosinus de deux vecteurs représentant ces entités, où chaque dimension correspond à un terme et sa valeur correspond à la valeur TF/IDF de ce terme.

On définira un n-gram de caractères par une suite de n caractères : bi-grams pour n=2, tri-grams pour n=3, quadri-grams pour n=4, etc. Il n'est plus question de chercher un délimiteur comme c'était le cas pour le mot. Un découpage en n-grams de caractères, quelque soit n, reste valable pour toutes les langues utilisant un alphabet et la concaténation comme opérateur de construction de texte.

.

Définition 10 *(n-gram)**. Soit*** $ngram(s,n)$ ***l'ensemble de sous chaines de caractères s de taille n. La mesure de similarité*** $ngram$ ***est une similarité*** $\sigma: \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}$ ***tel que:***

$$\sigma(s,t) = |ngram(s,n) \cap ngram(t,n)| \tag{6}$$

La version normalisée de cette fonction est comme suit.

$$\bar{\sigma}(s,t) = \frac{|ngram\ (s,n) \cap ngram\ (t,n)|}{\min(|s|,|t|) - n + 1} \tag{7}$$

[Monge et Elkan., 1996] propose une méthode hybride pour comparer des chaînes de caractères longues, qui découpe ces deux chaînes en plusieurs chaînes plus courtes.

Ensuite, ces dernières sont comparées par une mesure de distance quelconque citée ci dessus. Enfin, les résultats obtenus sont combinés.

Définition 11 *(Similarité hybride). Soit* $s = a_1 \dots a_K$ *et* $t = b_1 \dots b_L$ *deux chaînes de caractères, où* a_i *et* b_j *sont des sous-chaînes de s et t respectivement. Soit S une mesure de la similarité entre deux chaînes des caractères. La similarité hybride est une fonction de la similarité* $S_{hybride} : S \times S \rightarrow [0,1]$ *telle que :*

$$\overline{S_{hybride}}(s,t) = \frac{1}{K}\sum_{i=1}^{K} \max_{j=1}^{L} S(a_i, b_j) \quad (8)$$

[Cohen et al., 2003] fait une excellente comparaison de plusieurs mesures de similarité montrant le point fort de chaque technique pour une tâche particulière. Chaque mesure de distance ou de similarité s'adapte mieux dans certains domaines d'application.

Le Tableau I.2 (synthèse de [Cohen et al., 2003]) résume des domaines d'applications pour des mesures présentées dans cette partie.

Mesure de similarité	Domaine d'application
N-gram	Bigrams (n=2) est efficace avec des erreurs typographique mineures
Distance d'édition	Peut être appliquée aux entités ayant une longueur variable. Pour atteindre une exactitude raisonnable, les coûts des opérations de modification dépendent de chaque domaine
Distance de Hamming	Utilisée principalement pour les entités numérique ayant des tailles fixes, comme les codes postaux ou les numéros de sécurité sociale
Distance de Monge-Elkan	La meilleure performance au niveau des résultats dans plusieurs expériences. Peut être employée dans plusieurs domaines
Distance de Jaro/Jaro Winkler	Presque même performance au niveau des résultats que monge-Elkan mais beaucoup plus rapide
Distance basée sur TF/IDF	La meilleure pour la comparaison des textes longs(Basée sur les tokens)

Tableau I.2. Critères principaux d'utilisation des mesures de la similarité

I.5.1.2.3 Les méthodes linguistiques

La similarité entre deux entités représentées par des termes peut aussi être déduite en analysant ces termes à l'aide des méthodes linguistiques. Ces méthodes exploitent essentiellement des propriétés expressives et productives de la langue naturelle [Maynard et Ananiadou., 1999]. Les informations exploitées peuvent être celles intrinsèques (des propriétés linguistiques internes des termes telles que des propriétés morphologiques ou syntaxiques) ou celles extrinsèques (employant des ressources externes telles que des vocabulaires ou des dictionnaires).

- **Les méthodes intrinsèques**

Une même entité ou un même concept peut être référencé par plusieurs termes (synonymie) ou par plusieurs variantes d'un même terme. Le Tableau I.3 (extrait de [Euzenat et al., 2004]) montre des variantes possibles du termes enzyme activity.

Type	Sous-type	Exemlpe
Morphologique	Inflexion Dérivation Flexionnel-Dérivationnel	enzyme activities enzymatic activity enzymatic activities
Syntaxique	Insertion Permutation Coordination	Enzyme amidolytic Activity of enzyme Enzyme and bactericidal activity
Morpho-syntaxique	Dérivation-Coordination Inflexion-Permutation	Enzymatic and bactericidal activity Activity of enzymes
Sémantique		Fermentation
Multilingue	French Vietnamien	Activité d'enzyme Sur lèn men

Tableau I.3. Variantes du terme enzyme activity (extrait de [Euzenat et al., 2004])

Les méthodes intrinsèques fonctionnent avec le principe de chercher la forme canonique ou représentative d'un mot ou d'un terme (lemme) à partir de ses variantes linguistiques (lexème). La similarité entre deux termes est donc décidée en comparant leurs lemmes. Par exemple, le résultat de la mesure de similarité exacte de deux mots « ran » et « running » sera égal à 0 (c-a-d. ils sont différents), alors que le résultat de la même mesure pour les lemmes de ces mots sera égal à 1, ce qui indique que « ran » et « running » sont similaires.

La recherche du lemme d'un mot peut être effectuée dans un dictionnaire. Une autre approche qui est automatique et plus légère et plus efficace est d'utiliser des stemmers. Un stemmer est un programme ou un algorithme qui détermine la forme radicale à partir d'une forme infléchie ou dérivée d'un mot donné. Les radicaux (stems) trouvés par les stemmers n'ont pas besoin d'être identiques à la racine morphologique du mot. Il suffit que les mots similaires soient associés à un même radical, même si ce radical n'est pas une racine de mot valide. Un stemmer pour le français, par exemple, devrait identifier les chaînes de caractères « maintenaient », « maintenait », « maintenant », ou « maintenir » comme basées sur la racine "mainten".

Une approche plus complexe pour déterminer le radical exact d'un mot est la lemmatisation. Ce processus comprend la détermination de la partie du discours (catégorie lexicologique) d'un mot, et l'application des règles de normalisation différentes pour chaque

partie du discours. Cette approche exige la connaissance de la grammaire d'une langue, des règles différentes… Elle est donc lourde, compliquée et difficile à implémenter.

Le premier stemmer publié a été écrit par Julie Beth Lovins [Lovins, 1968]. Ensuite un autre stemmer a été développé par Martin Porter [Porter, 1980]. Ce dernier est très largement utilisé, et est devenu l'algorithme standard utilisé pour chercher des radicaux dans la langue anglaise.

L'algorithme est implémenté dans plusieurs langages de programmation. Snowball[12], un framework pour implémenter des algorithmes de stemmers, leurs variantes ou bien des algorithmes de stemmers pour des autres langues (par exemple le finnois, le russe, le danois l'allemand, le français…), est aussi créé par Porter.

- **Les méthodes extrinsèques**

Ces méthodes calculent la valeur de similarité entre deux termes en employant des ressources externes telles que des dictionnaires, des lexiques ou des vocabulaires. La similarité est décidée grâce aux liens sémantiques déjà existants dans ces ressources externes tels que des liens synonymes (pour l'équivalence), des liens hyponymes/ hypernymes (pour la subsomption). Par exemple, à l'aide des ressources des synonymes, « voiture » et « bagnole » sont dites similaires.

Typiquement, WordNet[13], un système lexicologique, est employé pour trouver des relations telles que la synonymie entre des termes, ou pour calculer la distance sémantique entre ces termes, en utilisant des liens sémantiques, afin de décider s'il existe une relation entre eux.

Les ressources externes utilisées dans les méthodes extrinsèques peuvent aussi être des vocabulaires ou des dictionnaires multilingues, ou d'autres systèmes tels que: EuroWordNet[14], Polylex[15].

12 http://snowball.tartarus.org
13 http://wordnet.princeton.edu/
14 http://www.illc.uva.nl/EuroWordNet/ - un système de réseaux sémantiques pour des langues européennes où chaque langue développe son propre WordNet et elles sont reliées entre elles par des liens inter-langues.
15 http://www.informatics.susx.ac.uk/research/nlp/polylex/ - un lexique multilingue pour le Néerlandais, l'Anglais et l'Allemand, construit à partir de différents lexiques monolingues contenues dans la base de données CELEX (http://www.ru.nl/celex/)

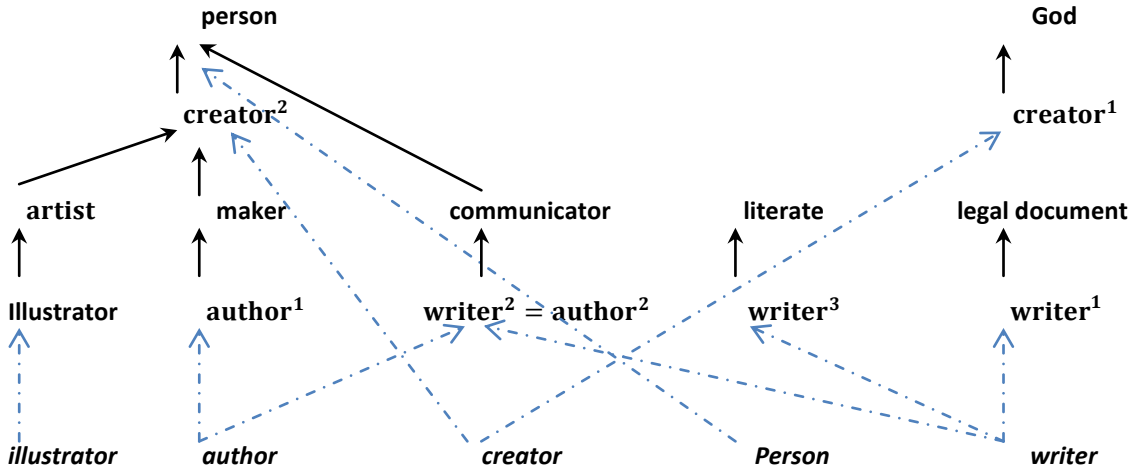

Figure I.8. Un fragment de la hiérarchie de WordNet pour mettre en évidence les liens entre les termes : « Author, writer, creator, illustrator & person» [Euzenat et shvaiko., 2007]

I.5.1.3 Les méthodes structurelles

Ce sont des méthodes qui déduisent la similarité de deux entités en exploitant des informations structurelles lorsque les entités en question sont reliées aux autres par des liens sémantiques ou syntaxiques, formant ainsi une hiérarchie ou un graphe des entités.

Nous appelons méthodes structurelles internes les méthodes qui n'exploitent que des informations concernant des attributs d'entité, et méthodes structurelles externes les autres qui considèrent des relations entre des entités.

I.5.1.3.1 Les méthodes structurelles internes

Ces méthodes calculent la similarité entre deux entités en exploitant des informations des structures internes de ces entités. Dans la plupart des cas, ce sont des informations concernant des attributs de l'entité, telles que des informations du codomaine des attributs, celles de la cardinalité des attributs, celles des caractéristiques des attributs (la transitivité, la symétrie), ou celles des autres types de restriction sur des attributs.

Par exemple, en considérant l'entité : le concept « Humain », nous pouvons exploiter des informations concernant des attributs de ce concept tels que l'intervalle des valeurs de donnée pour l'attribut « hasAge », à savoir [0, 150] ; la cardinalité de l'attribut « hasEpouse », à savoir 1 ; ou bien la caractéristique transitive de l'attribut « hasAncestor ».

I.5.1.3.2 Les méthodes structurelles externes

Contrairement aux méthodes décrites dans I.5.1.3.1, qui exploitent des informations des attributs d'entité, les méthodes structurelles externes exploitent des relations entre des entités elles-mêmes, qui sont souvent des relations de subsomption (is_a ou spécialisation) ou de méréologie (part-whole). Avec ces relations, les entités sont considérées dans des hiérarchies et la similarité entre elles est déduite de l'analyse de leurs positions dans ces hiérarchies. L'idée de base est que si deux entités sont similaires, leurs voisines pourraient également être d'une façon ou d'une autre similaires. Cette observation peut être exploitée de plusieurs manières différentes en regardant des relations avec d'autres entités dans des hiérarchies. Deux entités peuvent être considérées similaires si :

- Leurs super-entités directes (ou toutes leurs super-entités) sont similaires.
- Leurs sœurs (ou toutes leurs sœurs, qui sont les entités ayant la même super-entité directe avec les entités en question) sont déjà similaires.
- Leurs sous-entités directes (ou toutes leurs sous-entités) sont déjà similaires.
- Leurs descendants (entités dans le sous-arbre ayant pour racine l'entité en question) sont déjà similaires.
- Toutes (ou presque toutes) leurs feuilles (les entités de même type, qui n'ont aucune sous-entité, dans le sous-arbre ayant pour racine l'entité en question) sont déjà similaires.
- Toutes (ou presque toutes) les entités dans les chemins de la racine aux entités en question sont déjà similaires.

Des combinaisons des heuristiques ci-dessus sont aussi possibles.

Cependant, cette approche peut rencontrer quelques difficultés dans les cas, où les hiérarchies sont différentes au niveau de granularité. Par exemple, si dans une hiérarchie, l'entité « Personne » a deux sous-entités « Enfant » et « Adulte », et si dans une autre hiérarchie, la même entité « Personne » est divisée en deux autres sous-entités « Femme » et « Homme », la déduction que « Enfant » et « Femme » ou « Enfant » et « Homme » sont similaires, est incorrecte dans tous les cas.

I.5.1.4 Les méthodes extensionnelles

Ces méthodes déduisent la similarité entre deux entités qui sont notamment des concepts ou des classes en analysant leurs extensions, c.à.d. leurs ensembles d'instances.

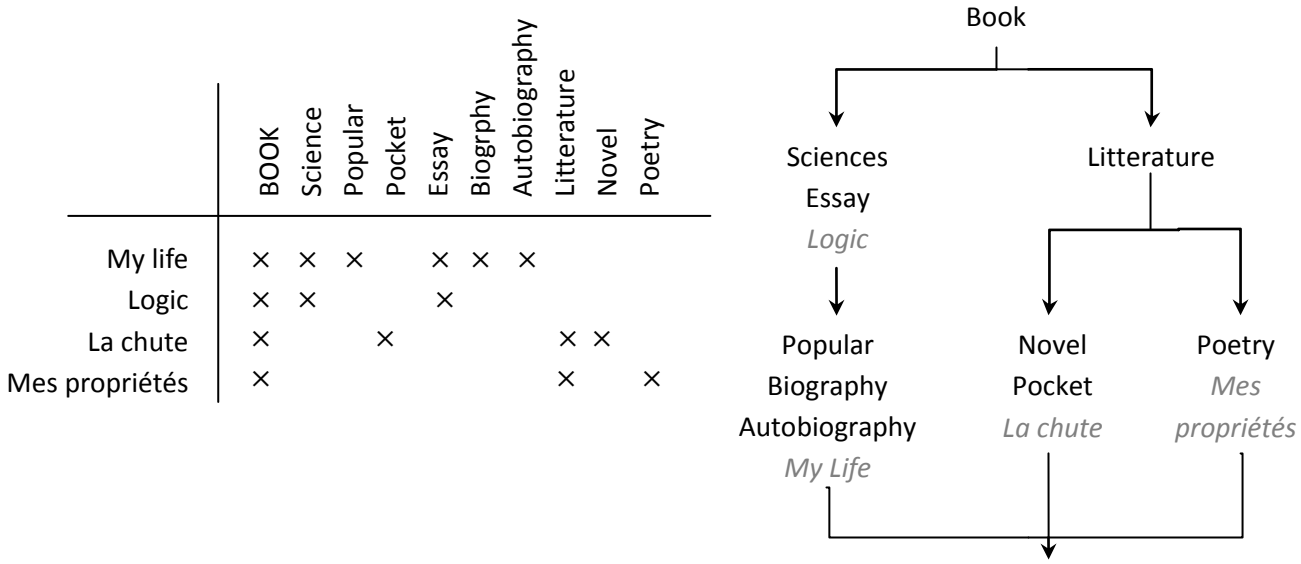

Figure I.9. Déduire les liens entre des concepts à partir d'un ensemble d'instances [Euzenat et shvaiko., 2007]

Dans le cas ou les ensembles d'instances partagent une partie commune, il est possible d'utiliser des mesures qui emploient des opérations sur les ensembles, telles que celles de Hamming ou de Jaccard. Fondamentalement, la mesure de Hamming compte un nombre d'éléments différents entre deux ensembles à comparer et la mesure de Jaccard est le rapport entre l'intersection des ensembles et leur union (voir Définition 5). Ces mesures peuvent être adaptées pour construire des mesures extensionnelles.

Définition 12 *(Distance de Hamming, version adaptée pour les ensembles des instances). Soit S et T deux ensembles. La distance de Hamming (appelée aussi la différence symétrique) entre S et T est une fonction de la dissimilarité.*

$$\overline{DS_{Hamming}}(S,T) = \frac{|S \cup T - S \cap T|}{|S \cup T|} \qquad (9)$$

Définition 13 *(Distance de Jaccard, version adaptée pour les ensembles des instances). Soit S et T deux ensembles. Soit P(x) la probabilité d'une instance aléatoire être dans l'ensemble X. La distance de Jaccard est une fonction de la dissimilarité* $DS_{Jaccard} : 2^E \times 2^E \rightarrow [0,1]$ *telle que :*

$$\overline{DS_{Jaccard}}(s,t) = 1 - \frac{P(S \cap T)}{P(S \cup T)} \qquad (10)$$

Les mesures ci-dessus produisent la similarité de deux entités qui est en fait la similarité entre les deux ensembles de leurs instances en se basant sur la comparaison exacte des

éléments dans deux ensembles. Dans le cas où les ensembles des instances ne partagent aucune partie commune, ces mesures ne sont plus applicables (le résultat retourné sera toujours égal à 1, c.-à-d. les entités à comparer sont toujours différentes).

I.5.1.5. Les méthodes sémantiques

I.5.1.5.1. Les techniques basées sur les ontologies externes

Lorsque deux ontologies doivent être alignées, il est préférable que les comparaisons se fassent selon un capital de connaissances commun. Ce type de techniques s'intéresse à l'utilisation d'ontologie formelle intermédiaire pour répondre à ce besoin. Cette ontologie va définir un contexte commun [Giunchiglia et al., 2006] pour les deux ontologies à aligner, et sera utilisée dans notre étude afin de compléter les techniques classiques d'alignement cités précédemment qui se basent sur la structure ou la richesse du langage de représentation des ontologies, et ne donnent pas un bon résultat quand les ontologies à apparier sont faiblement structurées ou se limitent à de simples hiérarchies de classification.

L'idée est que cette ontologie, avec une couverture appréciable du domaine d'intérêt des ontologies (ou une ontologie encore plus générale comme une ontologie de haut niveau), va permettre de lever le voile sur les ambigüités concernant les différentes significations possibles des termes. Des exemples d'ontologies intermédiaires : FMA[16] "the Foundational Model of Anatomy" et SUMO[17] "the Suggested Upper Merged Ontology".

I.5.1.5.2. Les techniques déductives

Les méthodes sémantiques se basent sur des modèles de logique (tels que la satisfiabilité propositionnelle (SAT), la SAT modale ou les logiques de descriptions) et sur des méthodes de déduction pour déduire la similarité entre deux entités.

Les techniques des logiques de description (telles que le test de subsomption) peuvent être employées pour vérifier des relations sémantiques entre des entités telles que l'équivalence (la similarité est égale à 1), la subsomption (la similarité est de 0 à 1) ou l'exclusion (la similarité est égale à 0), et permettent donc de déduire la similarité de deux entités. Dans notre proposition cette technique est combinée avec la précédente

16 http://sig.biostr.washington.edu/projects/fm/
17 http://www.ontologyportal.com

I.5.2 Les méthodes de combinaison des similarités

Une entité peut être considérée sous plusieurs différents aspects, soit en s'appuyant sur son nom, soit sur ses attributs, ou soit sur ses relations avec d'autres entités. La similarité entre deux entités peut donc être calculée en se basant sur plusieurs aspects ; les caractéristiques d'une entité sont comparées avec les caractéristiques correspondantes d'une autre par une des mesures de similarité de base présentées dans la section I.5.1, cela retourne une valeur de la similarité (ou de la dissimilarité/distance).

Il faut donc un moyen pour combiner toutes les valeurs de similarité calculées de chaque aspect pour produire une seule valeur de similarité représentative pour deux entités à comparer. Cette partie analyse quelques approches existantes dans la littérature.

La distance de Minkowski entre deux entités est définie comme suivante :

Définition 14 *(Distance de Minkowski). Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. La distance de Minkowski entre x et y est une fonction de la dissimilarité $DS_{Minkowski}$: $O\ x\ O \rightarrow R$ telle que :*

$$DS_{Minkowski}\ (x, y) = \sqrt[p]{\sum_{i=1}^{n} DS(x_i, y_i)^p} \qquad (11)$$

Cette distance est une mesure généralisée avec différentes valeurs de p, $p \geq 1$.

Quand p est égale à 1, elle devient la distance de « city block » et quand p = 2 elle devient la distance euclidienne. La distance de Chebyshev (appelée aussi la distance de valeur maximum) est un cas spécial de la distance de Minkowski avec $p = \infty$:
$DS_{Chebys\,hev}(\mathrm{x}, \mathrm{y}) = \max_{\mathrm{i}} \mathrm{DS}(\mathrm{x_i}, y_i)$.

Cette mesure n'est une fonction linéaire que quand p = 1 ou $p = \infty$. Dans le cas où p = 1, une variante de cette mesure avec des poids est souvent utilisée. Le bon côté de cette variante est que nous pouvons contrôler l'influence (ou l'importance) de chaque dimension sur la valeur finale de la distance. Les dimensions plus importantes seront associées avec les poids plus élevés, donc les valeurs de ces dimensions influenceront mieux à la valeur agrégée finale.

Définition 15 $(Somme\ pondérée)$*. Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. Soit ω_i le poids de la dimension i. Soit $DS(x_i, y_i)$ la dissimilarité de la paire des objets à la dimension i. La somme pondérée entre x et y est une fonction de la dissimilarité DS_{sp}: $O\ x\ O \rightarrow R$ telle que :*

$$DS_{sp}(x,y) = \sum_{i=1}^{n} \omega_i * DS(x_i, y_i) \quad (12)$$

En général, la somme des poids est égale à 1 : $\sum_{i=1}^{n} \omega_i = 1$ *, dans ce cas, nous avons la version normalisée de* DS_{sp}.

Une autre mesure analogue à la somme pondérée est le produit pondéré.

Cependant, un inconvénient de cette mesure est que le résultat sera égal à 0 si une des dimensions est égale à 0.

Définition 16 *(Produit pondéré)*. Soit O l'ensemble d'objets qui peuvent être analysés dans n dimensions. Soit x et y deux objets dans O. Soit w_i le poids de la dimension i. Soit $DS(x_i, y_i)$ la dissimilarité de la paire des objets à la dimension i. Le produit pondéré entre x et y est une fonction de la dissimilarité $DS_{pp} : O \; x \; O \rightarrow R$ telle que :

$$DS_{sp}(x,y) = \prod_{i=1}^{n} DS(x_i, y_i)^{w_i} \quad (13)$$

Toutes les approches, qui combinent des valeurs de similarité calculées par différentes mesures, emploient la méthode de la somme pondérée (Définition 15). Cependant, certaines approches (par exemple Anchor-PROMPT) déduisent des alignements en examinant des critères heuristiques sans utiliser des méthodes de combinaison des similarités.

Résumé : Les techniques d'alignement présentées sont les outils dont dispose les concepteurs de systèmes d'alignement pour trouver les solutions aux problèmes de l'hétérogénéité des ontologies. La conception et le développement de ces systèmes nécessite un traitement plus global, en effet, ces systèmes doivent pouvoir : [Euzenat et Shvaiko., 2007]

- *Sélectionner et combiner* les techniques de base de l'alignement afin d'améliorer la qualité de l'alignement résultant.
- *Apprendre* à partir des données les meilleurs techniques et les meilleurs paramètres.
- Tenir compte des remarques des experts du domaine de l'ontologie et des utilisateurs d'une manière générale.

I .6.Les approches d'alignement d'ontologies

Dans cette partie, nous allons analyser des approches déjà existantes dans la littérature qui concernent le problème d'alignement d'ontologies. Comme nous l'avons souligné au début de

la section précédente, les schémas, les répertoires, les vocabulaires peuvent, pour certains aspects, être considérés comme des ontologies simples, ne disposant que des concepts (ou classes) organisés ou non dans une hiérarchie de subsomption, sans autres relations entre ces concepts.

Nous étudions donc quelques approches permettant d'établir des correspondances entre des schémas, des répertoires ou des vocabulaires dans des domaines tels que les bases de données, l'intégration des données…Nous présentons tout d'abord une distinction entre les termes souvent utilisés : fusion(merge), alignement (align) et intégration (integration) :

La fusion est l'action de construire une nouvelle ontologie en unifiant plusieurs ontologies dans une seule ontologie [Pinto et Martins, 2001] [Stumme et Maedche, 2001]. Le but final est de créer une seule ontologie cohérente qui inclut toutes les informations de toutes les sources [Noy et Musen, 2001] [Klein, 2001].

L'alignement est employé quand des sources doivent être rendues conformes et cohérentes entre elles mais elles sont toujours gardées séparément [Noy et Musen, 2001]. Cela implique de mettre deux (ou plus) ontologies en accord mutuel, et de les rendre conformes et cohérentes [Corcho et Gomez-Perez, 2001] [Klein, 2001]. Plusieurs alignements sont créés pendant ce processus, pour définir collectivement les relations entre les ontologies originales.

L'intégration entraîne de construire une nouvelle ontologie en composant des pièces d'autres ontologies disponibles [Pinto et Martins, 2001]. Comme la fusion, ce processus produit comme résultat une nouvelle ontologie. La différence entre l'intégration et la fusion est que seules certaines parties des ontologies originales seront intégrées, le but n'est pas d'accomplir une fusion complète [Hameed et al., 2003].

I .6.1 Les approches dans d'autres domaines que le Web sémantique

Rahm et Bernstein dans [Rahm et Bernstein, 2001] a fait une excellente revue des travaux attaquant le problème de mise en correspondance automatique des schémas dans le domaine de base de données. Cette revue a été citée dans beaucoup d'articles de recherche sur la mise en correspondance des schémas ou bien des ontologies. Elle inspire aussi d'autres revues

telles que [Euzenat et al., 2004], [Shvaiko et Euzenat, 2005]. Nous examinons un peu plus en détail certaines approches.

Similarity Flooding (SF) [Melnik et al., 2001] est un algorithme de mise en correspondance des graphes génériques. Son application à la mise en correspondance de schémas est présentée dans [Melnik et al., 2001]. SF convertit les schémas (SQL DDL, XML) en des graphes étiquetés orientés (directed labeled graphs) et puis il applique le calcul de point-fixe pour déterminer les nœuds similaires dans les graphes. Cette approche est basée sur une comparaison très simple des chaînes des caractères des noms des nœuds pour calculer des correspondances initiales, puis ces correspondances sont fournies au module de mise en correspondance structurel de SF. Bien que SF ait appliqué une nouvelle approche orientée structurelle basée sur l'intuition que les éléments de deux schémas distincts sont similaires quand leurs éléments adjacents sont déjà similaires pour propager la similitude de deux éléments à leurs voisins respectifs, il se fonde principalement sur des étiquettes des arcs dans les graphes. S'il n'y a aucune étiquette pour des arcs dans le graphe ou si ces étiquettes sont presque identiques, l'algorithme ne fonctionne pas bien. Sans utilisation d'un dictionnaire terminologique externe (tel que WordNet [Miller, 1995]), l'algorithme ne donne pas de bons résultats de mise en correspondance au niveau linguistique dans la première phase, qui seront fournis à la deuxième phase, influençant ainsi les résultats finaux.

Contrairement à SF, ***Cupid*** [Madhavan et al., 2001] a proposé une approche de recherche des correspondances combinant un module sophistiqué de mise en correspondance des noms et un algorithme de mise en correspondance au niveau structurel. L'algorithme se compose de trois étapes : (i) le niveau linguistique : les valeurs de similarité entre les noms des éléments (les étiquettes) sont calculées en employant des techniques et mesures linguistiques (I.5.1.2) telles que la normalisation des chaînes des caractères (I.5.1.2.1), la catégorisation, les mesures de similarité sur des préfixes, des suffixes, l'emploi le thésaurus des synonymes, des hypernymes ; (ii) le niveau structurel : la similarité structurelle de deux éléments est la similarité de deux arbres dont leurs racines sont les éléments à comparer. Cette dernière similarité est calculée en se basant principalement sur la similarité des feuilles de ces arbres. (iii) la valeur de similarité finale de deux éléments est la somme pondérée de deux valeurs calculées dans deux étapes précédentes. Si cette valeur finale est plus grande qu'un seuil prédéfini, deux éléments sont considérés similaires. Dans son approche, Cupid produit la valeur de similarité de deux éléments en se basant

principalement sur la similitude des éléments atomiques dans les graphes (c'est-à-dire des feuilles). Ainsi s'il y a des différences significatives en structure des graphes donnés, Cupid ne peut pas trouver des correspondances correctes. Par exemple, si un concept est situé à la place d'une feuille dans le premier graphe (schéma), mais dans le deuxième, il est à la place d'élément non- feuille qui est la racine d'un sous-graphe, Cupid ne détectera pas qu'il s'agit du même concept.

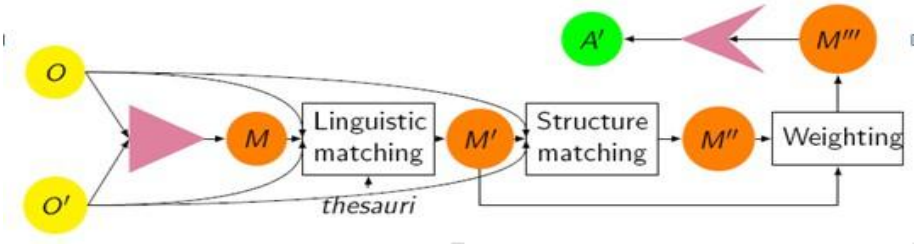

Figure I.10. Architecture de Cupid.

Do et Rahm ont développé ***COMA*** [Do et Rahm, 2002] comme un système permettant de mettre en correspondance des schémas (des bases des données, de XML) automatiquement ou bien manuellement. Le système fournit une bibliothèque des algorithmes de mise en correspondance de base (appelés matchers) et quelques mécanismes pour combiner des résultats des algorithmes de base afin d'obtenir une valeur de similarité finale de deux éléments dans deux schémas. La bibliothèque des matchers se compose de (i) 6 matchers simples qui emploient des techniques linguistiques (I.5.1.2) telles que la similarité des préfixes, des suffixes, n-gram, la distance d'édition , la similarité phonétique (soundex), la synonymie avec des dictionnaires externes, la similarité des types de données prédéfinie ; (ii) 5 matchers hybrides qui combinent des matchers simples précédents en exploitant quelques informations structurelles telles que des chemins entre des éléments, la similarité de leurs enfants. COMA présente quelques stratégies pour agréger des résultats de similarité pour chaque paire d'éléments, calculés de différents matchers tels que le choix de la valeur maximale, la valeur moyenne, la somme pondérée ou la valeur minimale. La sélection de bonnes correspondances parmi toutes les paires d'éléments repose sur une des stratégies suivantes: la paire ayant la valeur de similarité agrégée maximale, ou les paires ayant les valeurs de similarité agrégées dépassant un seuil. Le système COMA permet aussi d'exécuter plusieurs itérations de calcul et d'utiliser le résultat de l'étape précédente dans le calcul de similarité de l'étape actuelle, ou d'avoir des interactions des utilisateurs dans chaque itération.

I .6.2 Les approches dans le domaine du Web sémantique

Dans [Dieng et Hug, 1998], les auteurs considèrent un contexte où les experts peuvent employer leurs propres ontologies, appelées les ontologies personnelles. Une ontologie est alors représentée par un support dans le formalisme des graphes conceptuels : ce support comprend un treillis de types de concepts, une hiérarchie de types de relations et un ensemble de marqueurs permettant de désigner les instances. L'objectif est de construire un modèle de connaissances commun (ontologie commune) à partir de différents modèles de connaissances des experts (ontologie personnelle). Cela est réalisé, dans un système appelé ***MULTIKAT***, par la comparaison des ontologies personnelles en exploitant des techniques reposant sur des opérations du formalisme de graphe conceptuel ou sur la structure des graphes. La comparaison de deux supports est effectuée en trois étapes : la comparaison et la fusion des treillis de type de concept, la comparaison et la fusion des hiérarchies de type de relation et la comparaison et la fusion de deux ensembles de marqueurs. Les techniques utilisées pour obtenir la similarité entre deux entités sont alors : l'égalité des chaînes de caractères sur des noms de type, des synonymes, la comparaison des sous-types et des super-types directs dans la hiérarchie de types (I.5.1.3.2), la somme pondérée (*Définition 15*) pour combiner des valeurs de similarité.

Anchor-PROMPT [Noy et Musen, 2001] construit un graphe étiqueté orienté représentant l'ontologie à partir de la hiérarchie des concepts (appelés classes dans l'algorithme) et de la hiérarchie des relations (appelées slots dans l'algorithme), où les noeuds dans le graphe sont des concepts et les arcs dénotent des relations entre les concepts (les étiquettes des arcs sont les noms des relations). Une liste initiale des paires d'ancres (des paires de concepts similaires) définies par les utilisateurs ou automatiquement identifiées par la mise en correspondance lexicologique sert d'entrée à l'algorithme. Anchor-PROMPT analyse alors les chemins dans les sous-graphes limités par les ancres et il détermine quels concepts apparaissent fréquemment en positions similaires sur les chemins similaires. En s'appuyant sur ces fréquences, l'algorithme décide si ces concepts sont sémantiquement similaires. Cependant, Anchor-PROMPT ne cherche que des correspondances des concepts, pas des correspondances des relations. En outre, il emploie des noms de relation pour des étiquettes sur les arcs et la comparaison des chaînes de caractères de ces étiquettes n'est que la comparaison simple. Ainsi si les noms de relation sont différemment définis, l'algorithme ne fonctionnera pas bien. Les résultats retournés par l'algorithme seront également limités si les structures des ontologies sont différentes (par exemple l'une est profonde avec beaucoup

de concepts au milieu, et l'autre est peu profonde). L'algorithme rencontre des problèmes si une hiérarchie a seulement quelques niveaux et si la plupart des relations sont associées aux concepts au-dessus de la hiérarchie.

GLUE [Doan et al., 2002] est la version évoluée de LSD [Doan et al., 2000] dont le but est de trouver semi-automatiquement des correspondances entre des schémas pour l'intégration de données. Comme LSD, GLUE utilise la technique d'apprentissage (telle que Naive Bayes) pour trouver des correspondances entre deux ontologies. GLUE comprend plusieurs modules d'apprentissage (learners), qui sont entraînés par des instances des ontologies. Ces modules emploient la technique d'apprentissage Bayes naïf [Domingos et Pazzani, 1997] en exploitant différentes caractéristiques des instances telles que les valeurs textuelles des instances, les noms des instances, les formats des valeurs…. Les prévisions de ces modules de mise en correspondance sont combinées par un méta-module de mise en correspondance en employant la somme pondérée. Le résultat final des correspondances sera déduit à partir des valeurs de similarité agrégées en employant la technique d'optimisation de contraintes « relaxation labeling » (la technique permettant de résoudre le problème d'assignement des étiquettes aux nœuds d'un graphe en donnant un ensemble de contraintes). Un inconvénient de cette approche est qu'elle se fonde principalement sur les instances des ontologies, qui ne sont pas toujours abondamment disponibles pour plusieurs ontologies. Un autre inconvénient est que l'ontologie est modélisée comme une taxonomie des concepts et que chaque concept a quelques attributs. Avec cette organisation, GLUE n'emploie pas des informations contenues dans la taxonomie (hiérarchie) des relations. GLUE fait également l'utilisation modeste des informations sur la taxonomie des concepts.

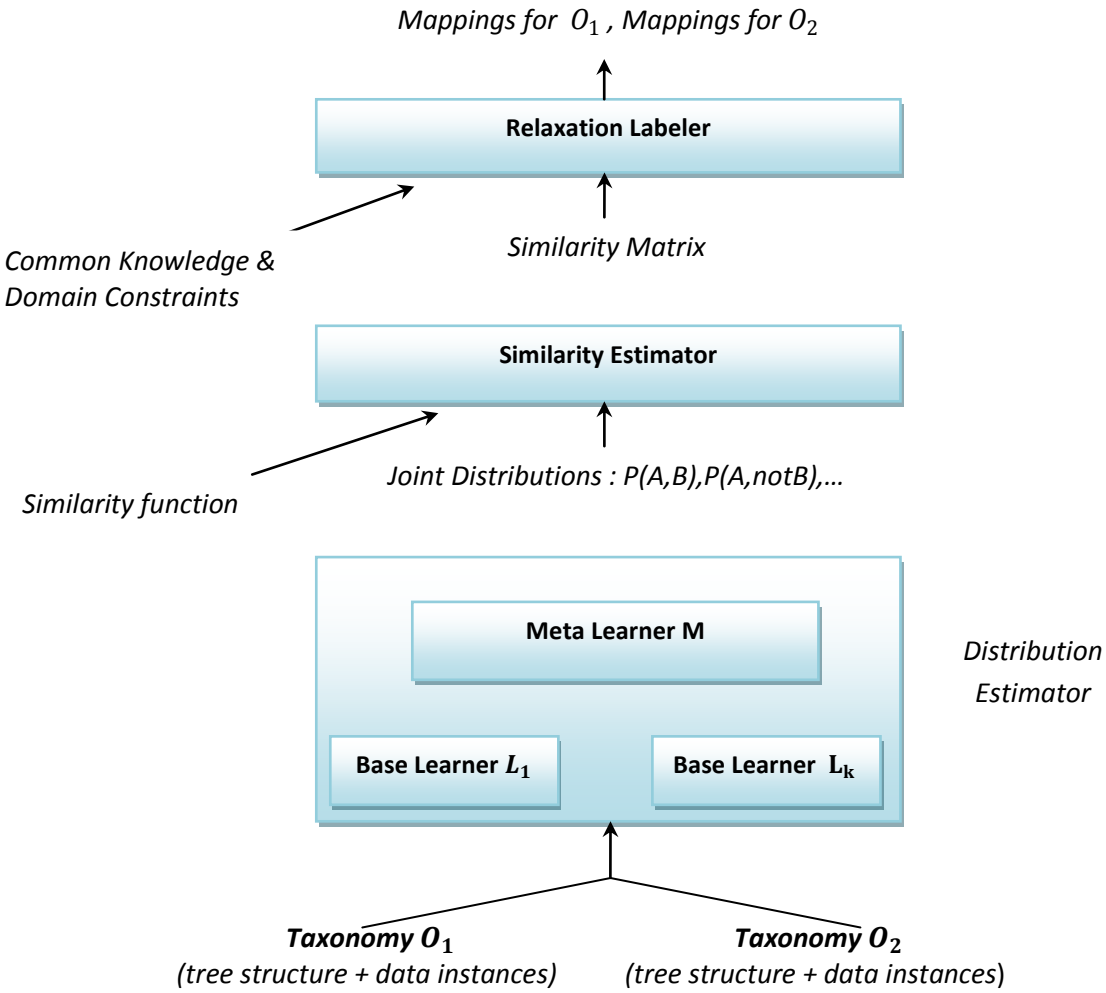

Figure I.11. Architecture de GLUE [AnHai Doan et al., 2003]

S-Match [Giunchiglia et al., 2004] est un algorithme et un système pour chercher sémantiquement des correspondances, basé sur l'idée d'employer le moteur de la satisfiabilité propositionnelle (SAT) [Giunchiglia et Shvaiko, 2003] pour le problème de mise en correspondance des schémas. Il prend comme entrée deux graphes des concepts (schémas), et produit en sortie des rapports entre les concepts tels que l'équivalence, overlapping, différence (mismatch), plus général ou plus spécifique. L'idée principale de cette approche est d'utiliser la logique pour coder le concept d'un nœud dans le graphe et d'appliquer SAT pour trouver des rapports. Le concept à un nœud, qui est alors transformé en formule propositionnelle, est la conjonction de tous les concepts des étiquettes des nœuds sur le chemin de la racine du graphe jusqu'au nœud en question. Le concept d'étiquette d'un nœud est construit en deux étapes : (i) la normalisation de l'étiquette : telle que la tokenization, la lemmatisation, et (ii) l'extraction du sens de l'étiquette normalisée (des lemmes) à partir de WordNet [Miller, 1995]. Ensuite, les relations sémantiques

(l'équivalence, plus général, plus spécifique) entre deux étiquettes de deux schémas sont (i) calculées grâce aux « matchers », les modules qui calculent la similarité entre deux étiquettes en employant des mesures de similarité de base (I.5.1.2) telles que la similarité des préfixes, des suffixes, la distance d'édition, la similarité de n-gram (définition 10) ; ou bien (ii) déduites en employant des matchers qui exploitent la sémantique dans WordNet, la similarité entre des hiérarchies, la similarité entre des commentaires [Giunchiglia et Yatskevich, 2004]. Ces relations sémantiques sont aussi encodées en logique. Enfin, le rapport entre deux concepts qui doit être prouvé est également converti en formule propositionnelle. Le moteur SAT calcule sur l'ensemble de formules propositionnelles pour vérifier si le rapport supposé est vrai ou faux. Cela permet donc de déduire des correspondances entre deux ontologies.

L'approche ***NOM*** [Ehrig et Sure, 2004] mette en correspondance des ontologies, est inspiré de COMA [Do et Rahm, 2002] dans le domaine de base de données. La représentation interne des ontologies est en format RDF(S). Cette approche se base sur un ensemble de règles. Les 17 règles listées sont catégorisées en différentes couches, dans lesquelles les règles exploitent des caractéristiques des entités à comparer. Ces couches correspondent aux couches du Web sémantique proposées par Tim Berners-Lee pour l'architecture du Web sémantique [Berners-Lee, 2003], telles que la couche des entités (instances), la couche des réseaux sémantiques, la couche des logiques de description… Les règles sont ensuite transformées en des formules de calcul de la similarité. Des mesures de base employées pour calculer la similarité sont l'égalité des chaînes de caractères, la similarité des chaînes (*Définition 6*) sur des noms de concept, de relation, d'instance ; la similarité des ensembles pour des ensembles des entités formés des sous-concepts/relations, des super-concepts/ relations, des concepts/relations voisins, des instances, des instances de propriétés, etc. Les valeurs de similarité calculées sont ensuite agrégées par des méthodes de combinaison telles que la somme pondérée (Définition 15), la fonction de Sigmoïde (une variante de la somme pondérée), des techniques d'apprentissage. Enfin, les meilleures valeurs agrégées de similarité sont choisies par une des méthodes « cut-off » (telles que la méthode du seuil, du delta ou du pourcentage [Do et Rahm., 2002]) pour déterminer des entités correspondantes.

L'algorithme ***QOM*** (Quick OntologyMapping) [Ehrig et Staab, 2004] est un échange entre l'efficacité (la qualité d'alignement) et l'efficacité (la vitesse de trouver des alignements), il s'agit d'une variante optimisée de NOM [Ehrig et Sure, 2004]. Il est employé

pour mettre en correspondance des ontologies « light-weight » (plutôt des thesaurus) telles que la hiérarchie des sujets de l'ACM, la structure des répertoires dans des ordinateurs personnels, le WordNet, ou l'UMLS. Ce sont des taxonomies ayant un nombre énorme de concepts (>104 concepts). Ehrig et Staab ont montré que leur QOM met en correspondance ces ontologies dans un délai acceptable sans sacrifier beaucoup la qualité du résultat final. Comme NOM, QOM représente des ontologies en OWL et utilise aussi des mesures de similarité telles que l'égalité des chaînes de caractères, la similarité des chaînes (Définition 6) sur des noms de concept, de relation, d'instance ; la similarité des ensembles… Cependant, QOM a quelques modifications en comparaison avec NOM pour réduire la complexité de calcul, donc le temps d'exécution, telles que des stratégies de sélection des candidats à comparer (par hasard, des candidats ayant des étiquettes à proximité dans la liste triée…) ; des ensembles à comparer sont aussi limités (par exemple, QOM ne compare que deux ensembles de concepts parentaux directs de deux instances, au lieu de tous les concepts ancestraux comme dans NOM).

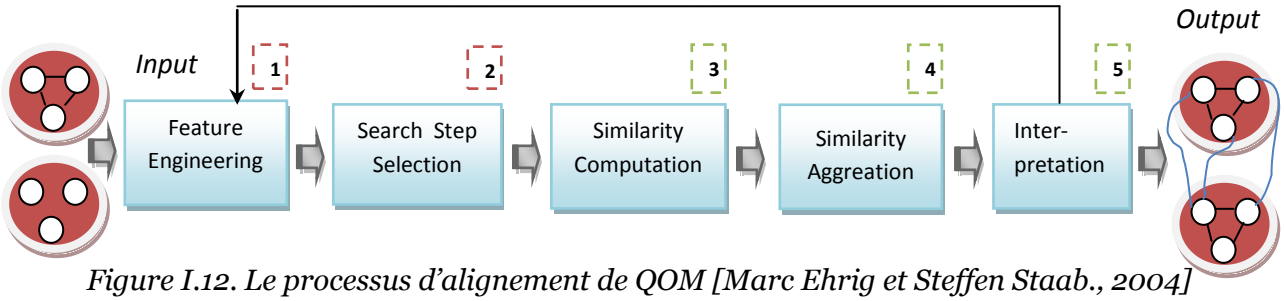

Figure I.12. Le processus d'alignement de QOM [Marc Ehrig et Steffen Staab., 2004]

OLA [Euzenat et Valtchev., 2004] est un algorithme pour aligner des ontologies représentées en OWL. Il essaie de calculer la similarité de deux entités dans deux ontologies en se basant sur leurs caractéristiques (leurs types : classe, relation ou instance, leurs rapports avec d'autres entités : sous-classe, domaine, co-domaine…) et de combiner les valeurs de similarités calculées pour chaque paire d'entités de manière homogène. La combinaison est la somme pondérée des valeurs de similarité de chaque caractéristique. Les poids sont associés suivant le type d'entité à comparer et ses caractéristiques. Ils sont mis dans une matrice des poids et sont prédéfinis avant l'exécution de l'algorithme. Les mesures de similarité de base employées dans l'algorithme sont l'égalité des chaînes des caractères pour des URIs des entités, des mesures de similarité des suffixes ou des chaînes des caractères (I.5.1.2.1) pour des étiquettes des entités, la similarité (l'égalité) des types des données. Pour la similarité

entre deux ensembles, le cas très souvent rencontré dans OWL (par exemple, en comparant deux entités, l'algorithme exploite la similarité de deux ensembles d'entités qui sont sous-entités des entités en question). À partir des valeurs de similarité calculées par des mesures de base, l'algorithme applique un calcul du point fixe, avec des itérations pour améliorer la similarité de deux entités. Quand il n'y a plus d'améliorations, des alignements entre deux ontologies sont générés.

I.6.3. Approche générale de l'utilisation de connaissances complémentaire

La différence entre cette approche et ceux citées précédemment est qu'on va utiliser une ontologie complémentaire afin d'aligner les concepts d'une ontologie, dite ontologie source O_{Src}, avec les concepts d'une autre ontologie, dite ontologie cible (target) O_{Tar}. Pour simplifier la présentation générale des différents travaux référencés, nous considèrerons que chaque ontologie O ne comprend qu'un ensemble de concepts C et un ensemble de relations R entre ces concepts.

Pour identifier l'existence d'un mapping de la forme (X_{Src} *relation* Y_{Tar}) où $X_{Src} \in C_{Src}$, $Y_{Tar} \in C_{Tar}$, et *relation* $\in R$, l'ensemble des relations exprimables entre deux concepts appartenant respectivement à l'une et à l'autre des ontologies considérées, l'approche générale suivie se décompose en 2 phases : l'ancrage et la dérivation.

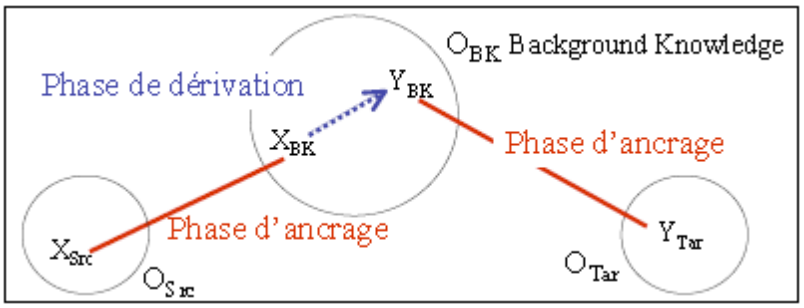

Figure I.13. *Schéma général de dérivation d'un mapping (X_{Src} relation Y_{Tar})*

L'ancrage consiste tout d'abord à apparier chacun des 2 concepts X_{Src} et Y_{Tar}, pris indépendamment l'un de l'autre, avec un ou des concepts de la 3ème ontologie (O_{BK}), c'est-à-dire, à identifier des mappings de la forme (X_{Src} *relation* X_{BK}) et (Y_{Tar} *relation* Y_{BK}) où X_{BK} et $Y_{BK} \in C_{BK}$ et sont appelés des *ancres* ou *points d'ancrage.*

La dérivation consiste ensuite à s'appuyer sur la structuration de O_{BK} pour :

- soit rechercher s'il existe des relations entre les différents points d'ancrage X_{BK}, Y_{BK} identifiés, afin d'essayer d'en dériver des relations (des mappings « sémantiques ») entre les éléments des ontologies à apparier,

- soit utiliser une mesure de similarité entre noeuds d'un même graphe, pour identifier pour chaque ancre X_{BK} d'un concept de l'ontologie source, l'ancre Y_{BK} du concept de l'ontologie cible qui lui est le plus similaire. Remarquons que la relation identifiée par cette mesure de similarité est une relation de proximité qui ne contient pas d'information sur la sémantique du lien unissant les deux concepts.

Si l'on considère que l'appariement d'ontologies est une fonction sur 2 ontologies qui retourne un ensemble de relations entre leurs concepts, f : ($O1$,$O2$) $\rightarrow$ {(X *relation* Y) | $X \in C1$, *relation* $\in R$, $Y \in C2$}, l'approche générale suivie par ces différents travaux revient donc à faire globalement trois appariements d'ontologies. En effet, la phase d'ancrage comporte deux appariements d'ontologies f(O_{Src}, O_{BK}) et f(O_{Tar}, O_{BK}) et la phase de dérivation, un appariement d'une ontologie sur elle-même f(O_{BK}, O_{BK}).

Pour effectuer la phase d'ancrage vers les éléments de O_{BK}, les auteurs s'appuient sur des heuristiques terminologiques simples qui portent sur les labels et les synonymes des termes désignant les concepts. Une première heuristique utilise une mesure de type *edit-distance* et considère que si les labels de deux concepts ne se différencient pas par plus de deux caractères, les concepts considérés peuvent être reliés par une relation d'équivalence. Une deuxième heuristique s'appuie sur l'inclusion de labels et consiste à dire que si tous les mots du label ou du synonyme d'un concept A se trouvent dans le label ou le synonyme d'un concept B, alors B sera considéré comme plus spécialisé que A ($B \leq A$).

I .6.2.1. Recherche de relations entre les ancres dans O_{BK}

Dans la recherche de mappings de la forme (X_{Src} *relation* Y_{Tar}), l'ensemble R des relations utilisées est l'ensemble $\{\leq, \geq, \equiv\}$ où $X \leq Y$ peut se lire, suivant les cas, *«X isA Y»*, *«X part-of Y»* ou plus généralement *«X narrower-than Y»*. Les mappings cherchés sont dérivés en exploitant des règles de la forme :

- Si ($X_{Src} \leq X_{BK}$) et ($X_{BK} \leq Y_{BK}$) et ($Y_{BK} \leq Y_{Tar}$) alors ($X_{Src} \leq Y_{Tar}$)
- Si ($X_{Src} \geq X_{BK}$) et ($X_{BK} \geq Y_{BK}$) et ($Y_{BK} \geq Y_{Tar}$) alors ($X_{Src} \geq Y_{Tar}$).

Ces règles utilisent aussi la relation d'équivalence, $\equiv$, en considérant que l'existence d'une relation de type $A \equiv B$ permet de rajouter les deux relations $A \leq B$ et $A \geq B$ et qu'inversement, le fait d'avoir pu dériver les deux relations $X_{Src} \leq Y_{Tar}$ et $X_{Src} \geq Y_{Tar}$ permet de dériver la relation $X_{Src} \equiv Y_{Tar}$.

Ces différentes règles permettent ainsi de dériver des mappings « sémantiques », i.e. des mappings reliant deux concepts par un lien de type *isA* ou *isEq* dont la sémantique est bien définie et qui peuvent être justifiés et prouvés par des mécanismes d'inférences (Sabou *et al.,* 2006).

Les travaux basés sur des règles de dérivation se différencient sur le type d'ontologie de support employée. Aleksovski dans [Aleksovski *et al.,* 2006a et 2006b], suppose que la recherche de dérivation peut s'effectuer sur une ontologie de support unique, préalablement identifiée et qui couvre a priori tous les concepts des ontologies à apparier. A l'inverse, les travaux décrits dans [Sabou *et al.,* 2006] font l'hypothèse opposée : la recherche de dérivation ne peut s'effectuer qu'au sein de multiples ontologies de support, sélectionnées dynamiquement. La section suivante décrit plus précisément cette approche.

I .6.2.2.Recherche de dérivation dans une ontologie de support unique et complète.

L'idée de base qui sous-tend ces travaux est que l'ontologie de support O_{BK} est plus complète et plus détaillée que les deux ontologies à rapprocher, et qu'elle contient une description en compréhension du domaine des 2 autres. Les deux d'abord à essayer d'apparier chacun des concepts des 2 ontologies initiales (O_{Src} et O_{Tar}) avec les concepts de la 3ème (O_{BK}). La dérivation consiste ensuite à rechercher au sein de O_{BK} les relations qui existent entre les différents points d'ancrage identifiés, puis d'en dériver des relations entre les éléments de O_{Src} et O_{Tar}.

Dans [Aleksovski *et al.,* 2006a], les concepts à rapprocher sont des éléments issus de 2 listes de vocabulaires plats, non structurés. L'ontologie O_{BK} utilisée pour rechercher les dérivations est une ontologie représentant des points de vue multiples (ou aspects), ce qui permet d'identifier plusieurs dérivations entre 2 points d'ancrage, suivant les différents aspects. Un ensemble de mappings (Gold Standard) a été élaboré avec le concours manuel d'un expert. Puis, les auteurs ont réalisé 2 expérimentations : l'une en recherchant directement des appariements entre les termes de O_{Src} et O_{Tar}, l'autre en recherchant d'abord les ancrages dans O_{BK}, puis les dérivations entre les paires d'ancres trouvées. Les auteurs observent une amélioration de la précision des mappings obtenus, dans la 2ème expérimentation. Ceci peut s'expliquer par l'existence de multiples dérivations obtenues dans O_{BK}, qui permet d'identifier des proximités sémantiques non identifiables par de simples rapprochements terminologiques.

Dans [Aleksovski *et al.*, 2006b], les concepts à rapprocher appartiennent à 2 ontologies structurées par des relations du type « *X narrower-than Y* » et « *X Broader-than Y* » ($\{\leq, \geq\}$) et O_{BK} contient des relations de type *is-a* et *part-of*. Ces 2 relations permettent d'inférer des relations de type *narrower-than*, dans la recherche de dérivation entre 2 ancres en s'appuyant sur les règles suivantes:

Si (X_{BK} *isA* Y_{BK}) alors ($X_{BK} \leq Y_{BK}$) et Si (X_{BK} *part-of* Y_{BK}) alors ($X_{BK} \leq Y_{BK}$).

Ils utilisent la fermeture transitive de relations : Si (X^1_{BK} *isA* X^2_{BK}) et (X^2_{BK} *isA* X^3_{BK}) et .. et (X^{n-1}_{BK} *isA* X^n_{BK}) alors dériver ($X^1_{BK} \leq X^n_{BK}$), qui s'applique aussi aux relations *part-of* et peut mêler les relations *isA* et *part-of* ou au contraire imposer de n'employer les relations *isA* qu'après avoir exploité tous les *part-of*.

De nouvelles expérimentations sont effectuées dans ce contexte, la 1ere en recherchant directement des appariements entre les termes de O_{Src} et O_{Tar}, et les suivantes par dérivation, en utilisant ou pas la fermeture transitive de relation, et sans imposer ou en imposant des contraintes sur l'ordre d'utilisation des relations *isA* et *part-of* lors de la fermeture. Pour pallier l'absence de mappings de référence, les évaluations de ces expérimentations ont été faites en choisissant au hasard 30 concepts de O_{Src} et en évaluant manuellement la correction des relations trouvées. La dernière technique de dérivation est celle qui donne les meilleurs résultats, toutes les relations identifiées ayant été jugées correctes.

On remarque qu'utiliser une ontologie de support permet de supposer que toutes les ancres des concepts à mettre en relation peuvent être identifiées en une passe avant d'effectuer la recherche de dérivation proprement dite, ce que améliore le coté performance de l'application.

Un comparatif des travaux présentés dans cette section est donné par le Tableau I.4.

Approche	Entrée	Représentation interne	Mesures terminologiques	Mesures structurelles	Mesures extensionnel	Mesures sémantiques	Mesures combinées	Observation*
MULTIKAT [Dieng et Hug., 1998]	Ontologies personnelles	Graphe conceptuel	égalité des chaînes de caractères synonymes	Super-types Sous-types	-		Somme pondérée	
Anchor-PROMPT [Noy et Musen., 2001]	Ontologies	RDF(S), graphe RDF	égalité des chaînes de caractères pour les étiquettes des concepts et des relations					Comparaison des chemins (information structurelle) entre des paires d'ancres
Cupid [Madhavan et al., 2001]	Schémas génériques	Graphes	Normalisation, catégorisation pour les noms d'élément de schéma Similarité des préfixes, des suffixes Thésaurus des synonymes, des Hypernymes	Similarité des arbres, des feuilles			Somme pondérée	La valeur de similarité agrégée dépasse un seuil prédéfini
Similarity Flooding (SF) [Melnik et al., 2001] [Melnik et al., 2002]	Graphes génériques Schémas de SQL, XML	Graphes étiquetés orientés	Similarité des préfixes, des suffixes pour les noms des nœuds					Calcul du point fixe : la similarité se répand sur le graphe (information structurelle)
GLUE [Doan et al., 2002]	Ontologies + instances	Non précisée	Techniques terminologiques : tokenisation, stem égalité des chaînes de caractères pour des tokens				Méta-learner : somme pondérée	Apprentissage automatique (Bayes naïf) à partir des valeurs textuelles, des noms des instances
COMA [Do et Rahm., 2002]	Schémas : bases des données relationnelles, XML	Graphes acycliques orientés	Similarité des préfixes, des suffixe, de n-gram, des synonymes, des types de donnée, distance d'édition, soundex,	Similarité agrégée des enfants, des feuilles ou des chemins entre des éléments			La valeur maximale ou minimale, la somme pondérée ou la valeur moyenne	MaxN, MaxDelta, ou dépasse un seuil
S-Match [Giunchiglia et al., 2004]	Schémas des bases des données, de XML, hiérarchies	Graphes	Similarité des préfixes, des suffixe, de n-gram, la distance d'édition pour les étiquettes	Distance des hiérarchies		Encode le concept d'un noeud en logique propositionnelle. Formule un		

[Giunchiglia et al., 2006]	des concepts, ontologies					ensemble d'équations logiques. Vérifie par SAT		
NOM [Ehrig et Sure., 2004]	Ontologies	RDF(S)	égalité des chaînes de caractères pour les URIs des concepts, des relations ou des instances similarité des chaînes pour des étiquettes des concepts, des relations ou des instances	Similarité des ensembles pour des super-concepts/relations, des sous-concepts/relations, des voisins, des concepts parentaux des instances	Similarité des ensembles pour des instances des concepts, des instances des relations		Somme pondérée Fonction de Sigmoid Technique d'apprentissage (trois niveaux) avec des réseaux neurologiques	
QOM [Ehrig et Staab., 2004]	Ontologies « lightweight» : hiérarchie des sujets de l'ACM, structure des répertoire dans des ordinateurs personnels, le WordNet, l'UMLS	RDF(S)	Comme NOM	Comme NOM avec des modifications : limitations sur des voisines directes	Comme NOM		Comme NOM	
OLA [Euzenat et Valtchev., 2004]	Ontologies en OWL	OL-graphes	égalité des chaînes de caractères pour les URIs des entités Similarité des chaînes des caractères pour des étiquettes (non précise) Similarité des types des données	Oui			Somme pondérée	Calcul du pointfixe (information structurelle)
[Aleksovski *et al.*, 2006a]	Ontologies en OWL	OWL	Phase d'ancrage: mesure de type *edit-distance ou une heuristique d'inclusion de labels*			Phase dérivation :cherche la relation entre les différents points d'ancrage identifiés		Utilise une ontologie complémentaire

Tableau I.4. Résumé des approches d'alignement de schéma et d'ontologie

I.7. Quelques domaines d'application de l'alignement des ontologies

I.7.1. L'ingénierie ontologique

Est un contexte où les concepteurs d'ontologies sont confrontés à l'hétérogénéité de ces dernières, plus précisément, par rapport aux applications suivantes :

La construction d'ontologies : ces dernières années, le maître mot dans la démarche de construction des ontologies est *la réutilisation* d'ontologies déjà existantes, car la construction d'ontologies à partir de zéro (from scratch) est un processus long, couteux et très laborieux, parallèlement, elle accentue le phénomène de l'hétérogénéité des ontologies, multipliant le nombre d'ontologies décrivant le même domaine (surtout lorsqu'on sait que l'objectif ultime du web sémantique est d'arriver à instaurer une ontologie de référence pour chaque domaine). Dans ce contexte, l'alignement des ontologies est la solution pour réaliser l'intégration et le rapprochement de ces différentes structures.

L'évolution des ontologies : Beaucoup d'ontologies sont en continuelle évolution comme la Geneontology[18], et de ce fait, plusieurs versions de la même ontologie sont disponibles, mettant les développeurs et les ingénieurs de la connaissance dans la confusion, ne sachant pas ce qui a changé, l'alignement va permettre d'identifier les différences entre deux versions: les entités qui ont été ajoutés, supprimés ou renommés (voir figure I.14).

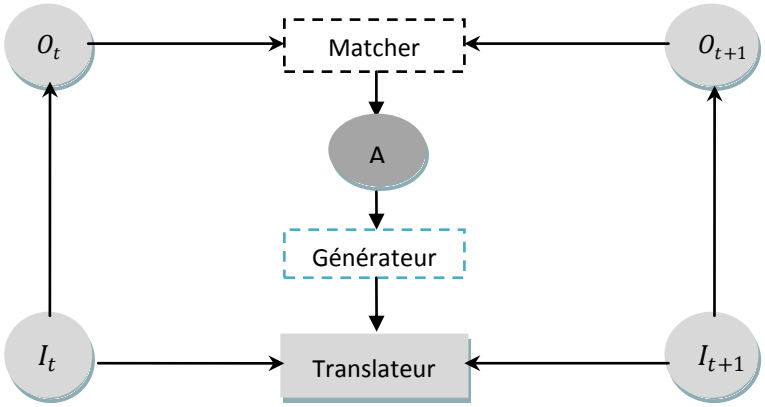

Figure I.14. Scénario centralisé d'évolution des ontologies [Euzenat et shvaiko., 2007]

Dans ce scenario il est utile de : (1) aligné l'ancienne version O_t et la nouvelle version O_{t+1} dont il résulte un ensemble de correspondance (A) entre deux versions, (2) générée une transformation en utilisant ces correspondances et (3) transformé les instances des données I_t au I_{t+n}.

18 http://www.geneontology.org

I.7.2. L'intégration d'information

C'est une application classique de l'alignement d'ontologies, elle comprend l'intégration des schémas, les entrepôts de données, l'intégration des données et l'intégration des catalogues. Les ontologies jouent un rôle clé en intégration de sources d'information multiples et hétérogènes.

Son rôle est double. D'une part, elle précise le sens des concepts d'un domaine en étant le reflet d'un certain consensus au sein d'une communauté. D'autre part, elle fournit une sémantique formelle. Dans le contexte de l'intégration, les ontologies peuvent aider à comprendre et interpréter des descriptions hétérogènes de contenus relatifs à un même domaine pour ensuite pouvoir plus facilement les mettre en relation.

La figure I.15 présente un scénario général de l'intégration d'information, étant donné, un ensemble de sources locales d'information (ontologies locales LO1, …LOn) qui stockent leurs informations dans des formats différents (SQL, XML, RDF), fournit aux utilisateurs une interface de requêtes uniforme à travers une ontologie globale ou commune (CO) à toutes les sources d'information. Ce qui permet aux utilisateurs d'adresser des requêtes directement à l'ontologie globale. Les sources de données sont transformés en ontologies locales qui sont alignées par rapport à une ontologie globale, *les alignements obtenus aident à générer les médiateurs* qui, à leurs tours, transforment les requêtes adressées à l'ontologie globale en requêtes pour les sources d'information locales et traduisent les réponses dans l'autre sens.

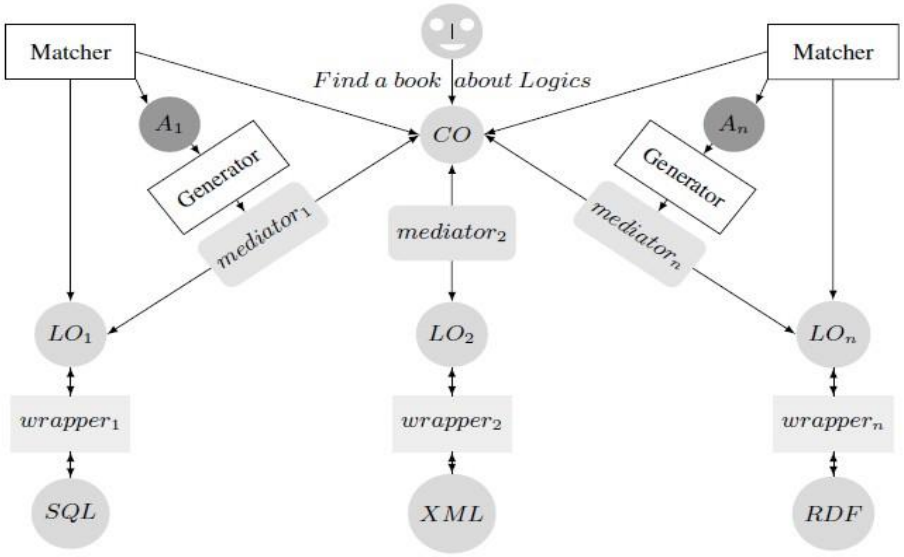

*Figure I.15. Scénario centralisé d'*intégration d'information *[Euzenat et shvaiko., 2007]*

I.7.3. Le Partage des informations dans les systèmes pair à pair (P2P)

Les architectures pair à pair se présentent actuellement comme une solution viable pour permettre le partage de ressources à l'échelle de l'Internet. En effet, aussi bien d'un point de vue commercial que scientifique, les architectures pair à pair suscitent un véritable engouement. Le paradigme du P2P garantit un fonctionnement à large échelle [Oram, 2001]. Un très grand nombre de pairs peut interagir dans le réseau, de manière à permettre le partage d'une grande quantité de ressources. Aussi appelé d'égal à égal, chaque participant à un système P2P peut être à la fois client et serveur. Le fonctionnement du système ne repose sur aucune coordination centralisée. Ainsi, le comportement global du réseau résulte uniquement des interactions locales entre les pairs qui se connectent et se déconnectent.

On distingue de nombreuses applications P2P comme (a) des systèmes de partage de fichiers comme Gnutella[19], (b) des moteurs de recherche P2P, (c) des messageries instantanées comme ICQ[20], (d) des plateformes de calcul distribué intensif tel Seti@home[21], (e) des plateformes collaboratives comme Oceanstore [Kubiatowicz, 2000] ou Ivy [Muthitacharoen, 2001] et (f) des systèmes de partage de données définies selon un schéma ou une ontologie Edutella [Nejdl, 2002].

P2P est un modèle de communication distribué dans lequel les pairs ont des capacités fonctionnelles équivalentes dans les échanges de données et de services [Zaihrayeu, 2006]. Les pairs doivent s'échanger des informations alors qu'ils utilisent des terminologies différentes (le problème de traduction des requêtes et de leurs réponses). Une des étapes pour résoudre ce problème serait d'identifier les relations qui existent entre leurs ontologies respectives, autrement dit, réaliser un alignement d'ontologies (voir figure I.16).

19 Gnutella : http://www.gnutella.org
20 ICQ : http://www.icq.com
21 Seti@home : http://setiathome.ssl.berkeley.edu/

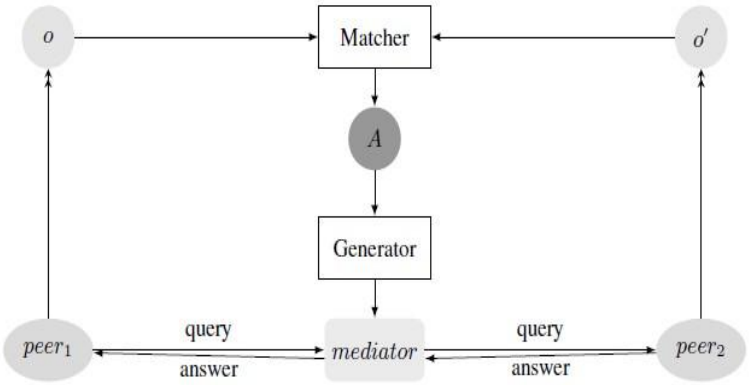

Figure I.16. Réponse à une requête dans un système P2P [Euzenat et shvaiko., 2007]

I.7.4. La Composition des services web

Les services web sont des processus qui exposent leurs interfaces aux utilisateurs du web qui les invoquent. Les services web sémantiques fournissent un moyen plus riche et plus précis de décrire les services à travers les langages de représentation des connaissances et des ontologies [Fensel et al., 2007]. Par exemple, un service web fournit la description de son output à l'aide d'une ontologie et un autre service web utilise une seconde ontologie pour décrire son input, aligner ces deux ontologies permettrait de vérifier si ce qui a été délivré par le premier service correspond à ce qui était attendu par le second service et cela grâce à un médiateur entre ces deux services, généré à partir de l'alignement des deux ontologies précédemment citées.

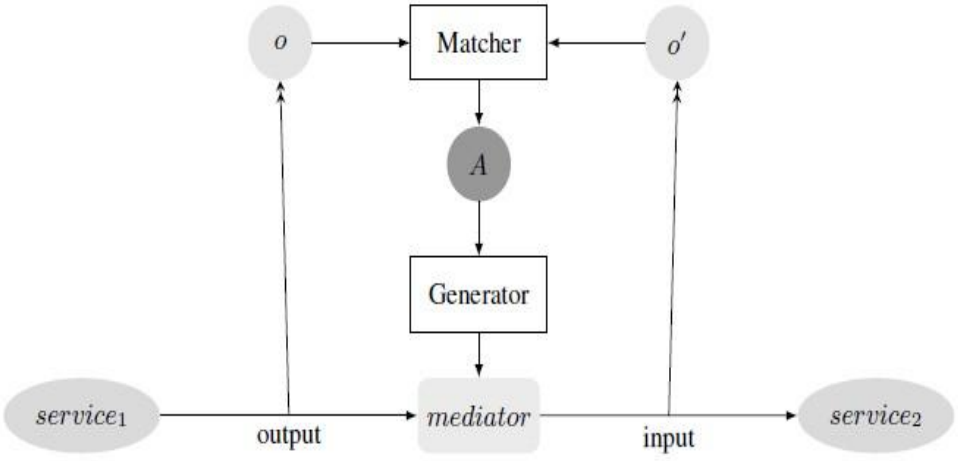

Figure I.17. Composition d'un service web [Euzenat et shvaiko., 2007]

I.7.5. La communication entre agents

Lorsque deux agents autonomes et conçus indépendamment se rencontrent, ils ont la possibilité de s'échanger des messages mais peu de chances pour se comprendre s'ils ne partagent pas le même langage et la même ontologie. L'alignement de leurs ontologies respectives intervient à ce niveau pour traduire les messages ou bien intégrer des passerelles entre leurs axiomes dans le modèle propre à chaque agent (pour pouvoir interpréter les messages). La flexibilité apportée par OWL représente un enjeu majeur sur bien des domaines. Notamment, pour la représentation des ontologies, et la capacité d'inférer sur ces derniers. En effet, les ontologies ainsi représentées peuvent valoriser le fonctionnement du Web de plusieurs façons. Notamment, pour améliorer la pertinence des moteurs de recherches; en faisant référence à un concept précis au lieu d'utiliser des mots-clés ambigus [Lacombe, 2006]. Aussi, la capacité des agents à comprendre les données manipulées offre de nouvelles perspectives dans le domaine de l'automatisation ; On citera, la composition automatique de services Web et la composition automatique deworkflow .

I.8. Conclusion

Dans cette étude nous utilisons une ontologie complémentaire O_{BK} qui est une représentation détaillée de la turbine à vapeur donnée par un expert. En commençant par apparier chacun des concepts des deux ontologies initiales (O_{Src} et O_{Tar}) avec les concepts de la 3ème (O_{BK}) en exploitant les techniques d'alignement classiques , puis inférer les relations existantes entre les différents ancres identifiés à la phase d'ancrage. Elle se différencie des travaux cités précédemment dans la section I.6.3 par le fait qu'à la phase d'ancrage nous adoptons l'algorithme de XMAP et à la phase de dérivation nous utilisons une raisonneur afin d'inférer les relations existantes entre les différents points d'ancrage identifiés au sein de O_{BK}, puis d'en dériver des relations entre les éléments de O_{Src} et O_{Tar}. L'utilisation d'un raisonneur avec l'ontologie sera le sujet du chapitre suivant.

Chapitre II

L'une des caractéristiques principales des ontologies décrites en OWL DL est qu'elles puissent être traitées par un raisonneur. L'un des services principaux qu'offrent un raisonneur est de tester si oui ou non une classe est une sous classe d'une autre classe. En procédant de la sorte sur toutes les classes d'une ontologie, il est possible pour un raisonneur d'inférer sur toute la hiérarchie de classes dans une ontologie. Un autre service standard qu'offre un raisonneur est de tester la consistance d'une ontologie. En se basant sur les conditions d'une classe, le reasoner peut vérifier si oui ou non il est possible pour une classe d'avoir une instance. Lors de la création d'ontologies assez consistantes (plus de 1000 classes), l'utilisation d'un raisonneur pour calculer les relations classe/sous classes devient vitale. Ceci permet une classification simple, on parle alors de classification *assertée* et de classification *inférée*. Afin de mieux comprendre, l'utilité de raisonnement et d'inférence dans le domaine des ontologies nous allons présenter dans ce chapitre les principes de la logique de description. Ensuite nous allons analyser et évaluer les différents moteurs d'inférence.

II.1. La logique de description

OWL bénéficie des résultats de plus de 15 ans de recherche dans les Logiques de Description (DL) [Baader et al 2003b, Horrocks et al 2004]. En effet, pour OWL, une sémantique est définie de telle sorte que les gros fragments (expressions) du langage peuvent directement être exprimés en Description Logics [Baader et al 2003c]. Les logiques de descriptions sont une famille qui permet la représentation de connaissance et sont des descendants de la théorie des réseaux sémantique et de KL-ONE [Brachman et al 1985]. Les logiques de description décrivent les domaines en termes de concepts (classes), rôles (propriétés et relations) et individus (instances).

II.1.1. Les deux niveaux de description

Dans la terminologie des logiques de description, un tuple de la T-box et de la A-box définit une base de connaissance. Un individuel est un objet spécifique nommé. Avec quelques restrictions, on peut voir que les bases logiques d'OWL peuvent être caractérisées par les logiques de descriptions de type SHIQ [Baader et al 2005]. Cela veut dire qu'avec quelques restrictions, les documents OWL peuvent être automatiquement transformés en T-box SHIQ. La partie RDF des documents OWL peuvent être transformés en A-box SHIQ [Haarslev et al 2003, MEI et al 2004]. Le tableau II.1 donne les principaux constructeurs des langages OWL Lite, CARIN-*ALN*, DAML+OIL (sensiblement similaire à OWL DL). Ce dernier est équivalent à la logique de description *SHIQ* d'un point de vue formel.

Constructeur du langage	DAML+OIL syntaxe	OWL Lite	CARIN-*ALN* syntaxe
1. Conjonction	C1 ^ C2	C1^C2 où C1 et C2 nommés ou restrictions	C1^C2
2. Alternative	C1 ∧ C2	Non	Non
3. Universalité	∀ r, C	∀ r, C	∀ r, C
4. Existentialité	∃ r, C	∃ r, C	Non
5. OneOf	{x1…xn}	{x1…xn}	Non
6. Négation	$\neg C$	Non	$\neg C$ sur C de base
7. Cardinalité	$\leq n\,r\;C\;\geq n\,r\;C = n\,r\;C$	$\leq n\,r\;C\;\geq n\,r\;C = n\,r\;C$ pour n=0 ou 1	$\leq n\,r\;\geq n\,r$

Tableau II.1 : Constructeurs de différents langages pour les ontologies

Le premier, le niveau terminologique ou TBox, décrit les connaissances générales d'un domaine alors que le second, le niveau factuel ou ABox, représente une configuration précise. Une TBox comprend la définition des concepts et des rôles, alors qu'une ABox décrit les individus en les nommant et en spécifiant en termes de concepts et de rôles, des assertions qui portent sur ces individus nommés. Plusieurs ABox peuvent être associés à une même TBox ; chacune représente une configuration constituée d'individus, et utilise les concepts et rôles de la TBox pour l'exprimer.

Avant de commencer nous présentons les termes qui vont être utilisé dans la suite.

Les entités atomiques : Les concepts atomiques et rôles atomiques constituent les entités élémentaires d'une TBox. Les noms débutant par une lettre majuscule désignent les concepts, alors que ceux débutant par une lettre minuscule dénomment les rôles (par exemple : les concepts *Femelle* , *Mâle*, *Homme* et *Femme,* et le rôle *relationParentEnfant*).

Les concepts et rôles atomiques prédénis : Les LD prédénissent minimalement quatre concepts atomiques : le concept $\top$ et le rôle $\top_R$, les plus généraux de leur catégorie respective, et le concept $\bot$ ainsi que le rôle $\bot_R$ les plus spécifiques (c'est-à-dire l'ensemble vide).

La notation : La suite de ce chapitre adopte la notation suivante : R dénote un rôle, C,D des concepts composés et A,B des concepts atomiques.

La notion d'interprétation : Une interprétation se compose d'un domaine d'interprétation $\triangle^{\mathcal{I}}$ et d'une fonction d'interprétation $^{\mathcal{I}}$. Le domaine d'interprétation consiste en un ensemble d'individus. La fonction d'interprétation assigne à chaque concept atomique A un ensemble tel que $A^{\mathcal{I}} \in \triangle^{\mathcal{I}}$, et à chaque rôle atomique R une relation binaire $R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \text{x} \triangle^{\mathcal{I}}$.

La logique minimale $\mathcal{AL}$: La logique minimale nommée $\mathcal{AL}$ a été introduite par [Schaub et Smolka] et revêt d'une grande importance dans le domaine. Cette logique est

minimale, dans le sens où une logique moins expressive représente peu d'intérêt [Baader et al 203a].

II.1.1.1. Le niveau terminologique (TBox)

Une TBox contient des axiomes terminologiques de la forme C ≡D ou C ⊑D(Le coté gauche de l'exemple suivant présente un TBox). La première sert à énoncer des relations d'équivalence entre concepts, alors que la seconde permet d'exprimer des relations d'inclusion. Une interprétation I satisfait un axiome C ⊑D si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Une interprétation I satisfait un axiome C ≡D si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$. Une interprétation satisfait une TBox (est un modèle de TBox) si et seulement si l'interprétation satisfait tous les axiomes de la TBox.

TBox	*ABox*
$Femelle \sqsubseteq \top \sqcap \neg Mâle$	$Humain(Anne)$
$Mâle \sqsubseteq \top \sqcap \neg Femelle$	$Femelle(Anne)$
$Animal \equiv Mâle \sqcup Femelle$	$Femme(Sophie)$
$Humain \sqsubseteq Animal$	$Humain(Robert)$
$Femme \equiv Humain \sqcap Femelle$	$\neg Femelle(Robert)$
$Homme \equiv Humain \sqcap \neg Femelle$	$Homme(David)$
$Mère \equiv Femme \sqcap \exists relationParentEnfant$	$relationParentEnfant(Sophie, Anne)$
$Père \equiv Homme \sqcap \exists relationParentEnfant$	$relationParentEnfant(Robert, David)$
$MèreSansFille \equiv Mère \sqcap$ $\forall relationParentEnfant.\neg Femme$	
$relationParentEnfant \sqsubseteq \top_R$	

Exemple : Une base de connaissances composée d'une TBox et d'une ABox

II.1.1.2. Le niveau factuel (ABox)

Une ABox contient un ensemble d'assertions sur les individus : (1) des assertions d'appartenance et (2) des assertions de rôle. Chaque ABox doit être associé à une TBox, car les assertions s'expriment en termes des concepts et des rôles de la TBox. La partie droite de l'exemple de dessus présente une ABox.

Une ABox désigne des individus dans ses assertions par des noms qu'elle leur donne. Ce texte utilise le terme individu nommé (nominal ou individual name, en anglais) pour référer à ces noms. L'exemple du tableau II.1 comprend les individus nommés suivants : Anne, David, Robert et Sophie. La suite de ce texte représente par les lettres a, b les individus nommés. Une fonction d'interprétation assigne à chacun de ces noms a, un individu $a^{\mathcal{I}}$ tel que $a^{\mathcal{I}} \in \triangle^{\mathcal{I}}$. Les moteurs d'inférence pour LD adoptent généralement l'hypothèse de noms uniques (HNU), c'est-à-dire que pour tout individu nommé a et b, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ [Baader et al 2003a].

Chaque assertion d'appartenance d'une ABox (notée C(a)), déclare que pour cette ABox, il existe un individu nommé a, membre du concept C de la TBox associée. Une interprétation satisfait une assertion d'appartenance C(a) si et seulement si $a^{\mathcal{I}} \in C^{\mathcal{I}}$.

II.2. L'inférence

L'inférence s'effectue au niveau terminologique ou factuel. Les sections II.2.1 et II.2.2 abordent le raisonnement au niveau terminologique et factuel, respectivement. Pour terminer, la section II.2.3 présente un tableau comparatif des différents moteurs d'inférence.

II.2.1. L'inférence au niveau terminologique

Quatre principaux problèmes d'inférence se présentent au niveau terminologique [Baader et al 2003a] **:**

- **Satisfiabilité** : Un concept C d'une terminologie $\mathcal{T}$ est satisfiable si et seulement existe un modèle $\mathcal{I}$ de $\mathcal{T}$ tel que $C^{\mathcal{I}} \neq \emptyset$.
- **Subsomption** : Un concept C est subsumé par un concept D pour une terminologie si et seulement si $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$ pour tout modèle $\mathcal{I}$ de $\mathcal{T}$.
- **Équivalence** : Un concept C est équivalent à un concept D pour une terminologie et seulement si $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$ pour chaque modèle $\mathcal{I}$ de $\mathcal{T}$.
- **Disjonction (disjointness)** : Des concepts C et D sont disjoints par rapport terminologie $\mathcal{T}$ si et seulement si $C^{\mathcal{I}} \setminus D^{\mathcal{I}} =$; pour chaque modèle $\mathcal{I}$ de $\mathcal{T}$.

Les moteurs d'inférence actuels tirent généralement profit du fait que les quatre types de problèmes d'inférence peuvent être réduits à des problèmes de subsomption ou à des problèmes de satisfiabilité. Les figures II.1 et II.2 illustrent cette propriété qui implique, que les moteurs d'inférence des LD ne nécessitent souvent qu'un seul algorithme d'inférence pour raisonner au niveau terminologique [Baader et al 2003a]. D'ailleurs, les deux grandes classes d'algorithmes de raisonnement pour les LD (algorithmes de subsomption de type normalisation/comparaison et algorithmes de vérification de satisfiabilité à base de tableaux) correspondent aux façons de réduire respectivement des problèmes d'inférence à des problèmes de subsomption et de satisfiabilité [Baader et al., 2003 a,b,c].

C est insatisfiable $\iff$ C est subsumé par $\bot$
C et D sont équivalents $\iff$ C est subsumé par D, et D par C
C et D sont disjoints $\iff$ $C \sqcap D$ est subsumé par $\bot$

Figure II.1. Réduction des problèmes d'inférence d'une TBox à des problèmes de subsomption

C est subsumé par D $\iff$ $C \sqcap \neg D$ est insatisfiable
C et D sont équivalents $\iff$ $C \sqcap \neg D$ et $\neg C \sqcap D$ sont insatisfiables
C et D sont disjoints $\iff$ $C \sqcap D$ est insatisfiable

Figure II.2 Réduction des problèmes d'inférence d'une TBox à des problèmes de satisfiabilité

Complexité	Logiques de description
P	$\mathcal{AL}$, $\mathcal{ALN}$
NP	$\mathcal{ALE}$
PSpace	$\mathcal{ALC}$, $\mathcal{ALEN}$
ExpTime	$\mathcal{SHIQ}$, $\mathcal{SHOQ}$
NExpTime	...

Tableau II.2 Complexité de la vérification de la subsomption et de la satisfiabilité en fonction de l'expressivité des LD

Le tableau II.2 présente un aperçu non exhaustif de la complexité du raisonnement au niveau terminologique en fonction de l'expressivité [Donini et al 2003; Baader et al., 2003; Horrocks et Sattler, 2005]. Ce tableau met en évidence la connaissance pointue des LD que la communauté scientifique détient. Les classes de complexité énumérées dans le tableau proviennent de la théorie de la complexité informatique. Voici une définition de ces classes [Padadimitriou, 1994] :

- **P**: la classe des problèmes de décision (un problème de décision prend en entrée un énoncé de problème et produit en sortie une réponse positive ou négative: oui ou non) qui requièrent un temps polynomial par rapport à la taille du problème pour obtenir une solution avec une machine de Turing déterministe.
- **NP**: la classe des problèmes qui nécessitent un temps polynomial pour trouver une solution avec une machine de Turing non déterministe.
- **PSpace** : la classe des problèmes de décision qui requièrent une quantité de mémoire polynomiale pour résoudre un problème avec une machine de Turing déterministe ou non déterministe.
- **ExpTime** : la classe des problèmes de décision solvables par une machine de Turing déterministe en un temps $\Theta(2^{p(n)})$ où $p(n)$ est une fonction polynomiale de n, la taille du problème.

- **NExpTime** : la classe des problèmes de décision solvables par une machine de Turing non-déterministe en un temps $\Theta(2^{p(n)})$ où $p(n)$ est une fonction polynomiale de n, la taille du problème.

  Il est connu que P $\subseteq$ NP $\subseteq$ PSpace $\subseteq$ ExpTime $\subseteq$ NExpTime (Padadimitriou, 1994).

II.2.2. L'inférence au niveau factuel

Le niveau factuel comprend quatre principaux problèmes d'inférence [Baader et Nutt, 2003] :

- **Cohérence** : Une ABox A est cohérente par rapport à une TBox $\mathcal{T}$ si et seulement s'il existe un modèle $\mathcal{I}$ de A et $\mathcal{T}$.
- **Vérification d'instance** : Vérifier par inférence si une assertion C(a) est vraie pour tout modèle $\mathcal{I}$ d'une ABox A et d'une TBox $\mathcal{T}$.
- **Vérification de rôle** : Vérifier par inférence si une assertion R(a, b) est vraie pour tout modèle $\mathcal{I}$ d'une ABox A et d'une TBox $\mathcal{T}$.
- **Problème de récupération** : Pour une ABox A, un concept C d'une terminologie $\mathcal{T}$, inférer les individus $a^{\mathcal{I}}1 \ldots a^{\mathcal{I}}n \in$ CI pour tout modèle $\mathcal{I}$ de $\mathcal{T}$.

II.3. Les Ontologies et le raisonnement

La représentation des connaissances par les ontologies peut s'accompagner des mécanismes de raisonnement. Le raisonnement concerne la manipulation des connaissances déjà acquises pour produire de nouvelles connaissances. Il utilise des mécanismes d'inférence qui permettent la résolution des problèmes pour lesquels il n'existe pas de procédures explicites dans le programme. L'opération d'inférence consiste à admettre une proposition préalable qui a la valeur Vrai. Le raisonnement, déduction, induction, etc., sont des cas spéciaux de l'inférence. Différents mécanismes de raisonnement sont utilisés selon les objectifs du système à mettre en place : raisonnement logique, raisonnement par classification, le filtrage, l'héritage et le raisonnement à base de règles.

- ***Le raisonnement logique***

Le raisonnement logique se base sur un mécanisme de déduction qui utilise un ensemble de règles d'inférence pour déduire des nouveaux faits à partir des faits connus. Ces règles, le modus ponens, le modus tollens et la spécialisation universelle sont combinées avec des manipulations syntaxiques des formules (filtrage et unification) pour élaborer la déduction

[Masini et al., 1989]. La logique fournit un formalisme clair et non ambigu. Cette clarté vient d'une part du fait que la signification d'une formule ne dépend que de sa structure et de la signification donnée à ses composants atomiques et d'autres part du fait que le langage d'expression logique est proche du langage naturel [Halton et al.,1991]. De plus les connecteurs logiques et les quantificateurs permettent une riche description du monde. Les inférences faites avec la logique du premier ordre sont correctes, complètes et fondées [Stillings et al., 1989]. Plusieurs langages de formalisation d'ontologies dotés de mécanismes de raisonnement logique sont proposés tel que OWL [OWL, 2004].

- ***Le raisonnement par classification***

Le raisonnement par classification consiste à confronter une nouvelle connaissance à un ensemble de connaissances connues pour déduire des informations liées à cette nouvelle connaissance. Dans des représentations à base de frames, la classification consiste à positionner un nouveau schéma dans une hiérarchie de frames connue. En général les systèmes à base de frames distinguent la classification de classes de la classification d'instances. La classification des classes modifie les liens taxinomiques des classes et constitue un mécanisme de gestion et de maintien de la base ; la classification d'instances est un mécanisme de raisonnement qui permet de compléter la connaissance d'une nouvelle instance en la plaçant correctement dans la base et en récupérant l'information déduite de ce classement.

- ***Le filtrage***

Utilisé par la plupart des réseaux sémantiques. Il consiste à parcourir le graphe et à chercher tous les sous-graphes ayant des propriétés ou une structure commune avec un graphe cible.

- ***L'héritage***

L'héritage est un mécanisme de raisonnement qui consiste à récupérer des informations des classes représentant des concepts plus généraux, pour les utiliser dans des classes plus spécialisées ; cette récupération se fait en suivant les liens de spécialisation « est-un ». La relation de spécialisation représente l'inclusion ensembliste ; toutes les instances d'une classe le sont aussi pour ses superclasses. Une classe doit donc pouvoir « récupérer » l'information de ses superclasses. Le mécanisme d'héritage de propriétés permet la récupération de cette information à travers les liens de spécialisation et évite ainsi d'avoir à recopier les attributs des superclasses dans la sous-classe. L'héritage est dynamique, et veut dire que l'information

héritée d'une classe n'est pas stockée dans la sous classe mais récupérée chaque fois que le système accède à la sous-classe. Cela garantit que toute modification faite à une classe est prise en compte par ses sous-classes. Le mécanisme d'héritage est un raccourci d'écriture (car il évite à recopier l'information) plus qu'un réel mécanisme d'inférence de nouvelles connaissances.

- ***Le raisonnement à base de règles***

Le raisonnement à base de règles est également un mécanisme de raisonnement sur les connaissances. L'élément de base des systèmes à base de règles est la règle d'inférence ; une règle a la forme suivante :

Si <condition> Alors <Action>

La partie condition est exprimée par un prédicat logique correspondant à une affirmation sur la base de connaissance qui doit être vraie au moment de valider la règle pour que l'action soit déclenchée ; la partie action, qui est la partie exécutable de la règle indique des ajouts ou modification à faire à la base. Un système à base de règles comporte trois parties : une base de règles, une base de faits et un moteur d'inférence. Les systèmes à base de règles permettent en général de bien résoudre les problèmes de diagnostic. Ils offrent un cadre déclaratif pour exprimer des connaissances procédurales, de « savoir-faire », ce qui permet de voir clairement les conditions dans lesquelles une règle est applicable.

II.4.Les moteurs d'inférence

Inférence: "opération mentale qui consiste à tirer une conclusion d'une série de propositions reconnues pour vraies". Appliqué à OWL, inféré une ontologie consiste à en tirer une série de nouveau objet qui découle de la définition des objets dans l'ontologie (par exemple, ajout d'attribut sur un Individus ou d'individus dans une classe). On utilise pour ceci un moteur d'inférence, aussi appelé raisonneur (reasonner). Parmi les outils de raisonnement exploités nous citons quelques moteurs d'inférence.

Actuellement, plusieurs moteurs d'inférences gratuits ou commerciaux tels que Racer, Pellet, Fact, Fact++, Surnia, F-OWL et Howlet existent. La majorité de ces moteurs sont conçus pour raisonner sur les logiques de description, mais acceptent en entrée des fichiers OWL. Certains moteurs d'inférence ne peuvent raisonner qu'au niveau terminologique (c'est-à-dire au niveau des concepts et des propriétés).

Nous présenterons plus en détail dans la suite Pellet et Racer qui sont à l'heure actuelle les deux seuls moteurs d'inférence, permettant le raisonnement sur la ABox et la TBox et

exploitent des ontologies possédant un niveau d'expressivité en logique de description et en OWL satisfaisant.

Le tableau suivant dresse une comparaison des principaux moteurs d'inférence pour les LD43: FaCT [Horrocks, 1998], Racer [Haarslev et Möller, 2001], Pellet [Sirin et Parsia, 2004], FaCT++ [Tsarkov et Horrocks, 2004], F-OWL [Zou et al., 2004], Surnia44, et Hoolet45. Le critère "Mise-à-échelle" mesure la capacité à demeurer efficace proportionnellement à la complexification des ontologies. Le tableau reprend les données de [Zou et al., 2004] pour ce critère, celui de décidabilité et pour les caractéristiques de Hoolet, Surnia et F-OWL.

43 http://fr.wikipedia.org/wiki/Logique_de_description
44 http://owl.man.ac.uk/hoolet/, 2004.
45 http://dev.w3.org/cvsweb/2000/10/swap/surnia/,2000.

Moteur	Racer	FaCT	Pellet	FaCT++	Surnia	Hoolet	F-OWL
LD	SHIQ(D)	SHIQ, SHF	SHIN(D), SHON(D)	SHIF(D)	Logique prédicats	Logique prédicats	SHIQ(D) et RDF
Implantation	C++	Common Lisp	Java	C++	Python	Java	Java
Inférence	TBox/ABox	TBox	TBox/ABox	TBox	TBox/ABox	TBox/ABox	TBox/ABox
API Java	Oui	Oui	Natif	oui	?	Oui	oui
Mise-à-échelle	Bonne	Bonne	Bonne	bonne	Médiocre	médiocre	Médiocre
OWL	OWL-DL~ +	OWL-DL~ +	OWL-DL~ +	OWL-DL~ +	OWL-FULL~ +	OWL-DL~ +	OWL-FULL~ +
Décidabilité	oui(OWL-LITE)	oui	oui (OWL-LITE)	oui	non	Non	non
DIG46	Oui	Oui	Non	?	Non	Non	Non

Tableau II.3 Tableau comparatif des principaux moteurs d'inférence pour LD

46 L'interface de communication DIG : un protocole standard pour interroger un moteur d'inférence par des requêtes http (http ://www.w3.org/Protocols/)

Le raisonnement sur TBox ou ABox : Tous ces moteurs raisonnent autant sur des ABox que sur des TBox, mis à part FaCT et FaCT++ qui se spécialisent en raisonnant sur des TBox seulement.

L'expressivité et l'efficacité des moteurs d'inférence : Les moteurs F-OWL, Hoolet et Surnia raisonnent sur des logiques de description très expressives. Hoolet et Surnia raisonnent sur l'expressivité totale de la logique des prédicats, alors que F-OWL infère sur la logique *SHIQ(D)* et sur l'expressivité totale de RDF (Ressource Description Framework, un modèle RDF représente un domaine par un ensemble de triplets sujet, prédicat, valeur qui lient par des propriétés (prédicat) des ressources entre elles (sujet, valeur)).Ces trois moteurs se basent sur des méthodes expérimentales de raisonnement qui exhibent des performances intéressantes pour des problèmes simples, mais insuffisantes pour une utilisation industrielle, en raison de leur faible efficacité et de la non-décidabilité de leurs algorithmes. Ce texte les mentionne à titre informatif.

Comme l'indique le critère "Mise-à-échelle" dans le tableau, les moteurs les plus performants actuellement sont Racer, FaCT, FaCT++ et Pellet. FaCT, FaCT++ et Racer disposent probablement de la plus grande notoriété actuellement, chacun d'eux est utilisé dans de nombreux projets.

Les interfaces de programmation Java : La plupart des moteurs mentionnés dans cette section procurent une interface de programmation (API) (autre qu'une interface DIG) pour faciliter l'accessibilité par un programme Java (voir le tableau II.2).

II.4.1 Racer

Racer [Haarslev et Möller, 2001] est le moteur d'inférence sans doute le plus connu et l'un des plus utilisés dans le domaine à cause de performances et sa stabilité. Il est commercialisé par Racer Systems GmbH & Co. KG, fondé en 2004 par Volker Haarslev, Kay Hidde, Ralf Möller et Michael Wessel qui travaillaient à l'université de Hambourg. Racer travaille sur les ontologies modélisées par son langage, mais il accepte des ontologies décrites en RDF ou OWL, ces dernières étant traduites vers le langage utilisé par Racer. Ce moteur d'inférence possède également son propre langage de requête nRQL (new Racerpro query Language) pour interroger les ontologies sur la ABox et la TBox .

Les avantages :

– La documentation sur Racer est importante, provenant des concepteurs et des utilisateurs.
– Racer permet l'utilisation d'un mécanisme d'abonnement à un concept qui permet d'être informé de la création de nouvelles instances de ce concept.
– Il permet de créer avec ce mécanisme d'abonnement un monde fermé local. Dans le monde ouvert, il n'est pas possible de s'abonner au concept de Livre sans Auteur, car un livre peut avoir un auteur décrit dans le futur. Racer permet de le faire. En effectuant préalablement 2 souscriptions la première (A) sur le concept de Livre, la seconde (B) sur le concept de Livre avec au moins un Auteur. L'abonnement sur le concept définit par A − B est donc un abonnement sur les livres sans auteur.
– Racer permet l'ajout d'assertions et d'individus dans les ABox après le chargement de l'ontologie.
– Racer permet l'utilisation de règles SWRL.

Les inconvénients :

– Racer suppose que tous les propriétés sur les datatypes sont fonctionnelles (pas de valeurs multiples pour un datatype property)
– Racer ne permet pas l'utilisation de type de données utilisateur (type défini par l'utilisateur), car il possède ces propres types de données et il effectue une conversion avec les types de base.
– Racer est un produit commercial, il n'existe pas de version libre d'utilisation. Cependant il est possible d'obtenir une licence gratuite dans le cadre de la recherche scientifique.

II.4.2. Pellet

Le moteur Pellet [Evren et al,2006] est beaucoup plus récent. Pellet est l'un des projets du MINDSWAP Group, un groupe de recherche sur le web sémantique de l'université du Maryland. Il est disponible en OpenSource et offre des évolutions fréquentes. Pellet travaille sur des ontologies décrites en RDF ou OWL et permet les requêtes avec RDQL et SPARQL sur la ABox et la TBox. Comme pour Racer, nous allons présenter quelques avantages et points négatifs de Pellet.

Les inconvénients :

– Pellet possède une documentation pauvre en comparaison de celle de Racer. En effet racer est le plus utilisé et donc le plus documenté par des particuliers. De plus la documentation officielle de Pellet reste assez pauvre en comparaison de celle de Racer.

– Actuellement Pellet ne permet pas l'utilisation de règles SWRL26 (la prochaine version l'inclura).
– Pellet n'offre pas de système de souscription à un concept.

Les avantages :

– Pellet est open-source et développé en Java.
– Pellet est un raisonneur OWL DL complet.
– Pellet propose en cas d'incohérence dans l'ontologie des réparations possibles, ainsi qu'une heuristique permettant d'obtenir les informations à ajouter dans l'ontologie pour passer au sous-langage OWL inférieur (OWL Full > OWL DL > OWL Lite).

II.5. Conclusion

Comme nous avons cité précédemment notre approche d'alignement se base sur l'utilisation d'une ontologie complémentaire et un raisonneur. Le raisonneur qu'on exploite est un API open source développé en java et s'appelle Pellet, il est intégré à la version 3.4 Béta du Protégé. Cette plate-forme et son utilisation seront décrites dans la suite de ce mémoire.

26 (Semantic Web Rule Language) est un langage de règles pour le web sémantique, combinant le langage OWL-DL et le langage RuleML (Rule Markup Language (Unary/Binary Datalog))
http://fr.wikipedia.org/wiki/Semantic_Web_Rule_Language

Chapitre III

Dans ce chapitre, nous présentons la conception de notre approche qui consiste à mettre en correspondance deux ontologies en utilisant une ontologie complémentaire et un moteur d'inférence. Cette algorithme se décompose en trois phases : -phase d'alignement –phase d'ancrage et –phase de dérivation. La phase d'alignement et d'ancrage exploitent des caractéristiques du langage OWL-DL pour déduire la similarité entre deux entités de deux ontologies. La valeur de similarité est calculée en deux parties : la partie linguistique qui se compose du nom, et la partie structurelle qui se base sur des informations, la présence des propriétés et leurs contraintes de cardinalités. A la phase de dérivation, on utilise un raisonneur pour déduire les relations existantes entre les points d'ancrage, ensuite on déduit la relation entre les entités des deux ontologies.

III. Conception de l'algorithme d'alignement :

Pour illustrer notre algorithme BRMAP de découverte de correspondances, nous prenons l'exemple d'une ontologie dite source O_{Src} souhaitant connaître ses correspondances avec une autre ontologie dite cible (target) O_{Tar} du même domaine. On suppose qu'il existe une ontologie O_{BK}, qui est plus détaillée que O_{Src} et O_{Tar} dite ontologie complémentaire (Background). On utilise aussi le moteur d'inférence pellet décrit dans le chapitre précédant.

Pour simplifier la présentation générale de notre algorithme, nous considèrerons que chaque ontologie *O* ne comprend qu'un ensemble de concepts *C* et un ensemble de relations *R* entre ces concepts. L'alignement entre ontologies dans notre approche illustrée dans *la figure III.1,* se décompose en trois phases. La première consiste à mettre en correspondance les concepts de O_{Src} avec les concepts de O_{Tar}, en identifiant un mapping de la forme (X_{Src} *relation* Y_{Tar}) où $X_{Src} \in C_{Src}$, $Y_{Tar} \in C_{Tar}$, et *relation* $\in R$, où R est l'ensemble des relations exprimables entre deux concepts appartenant respectivement à l'une et à l'autre des ontologies considérées. Les concepts non alignées dans la phase précédente vont être l'entrée de la deuxième phase appelée phase d'ancrage, et le résultat de cette dernière sera l'entrée de la phase de dérivation.

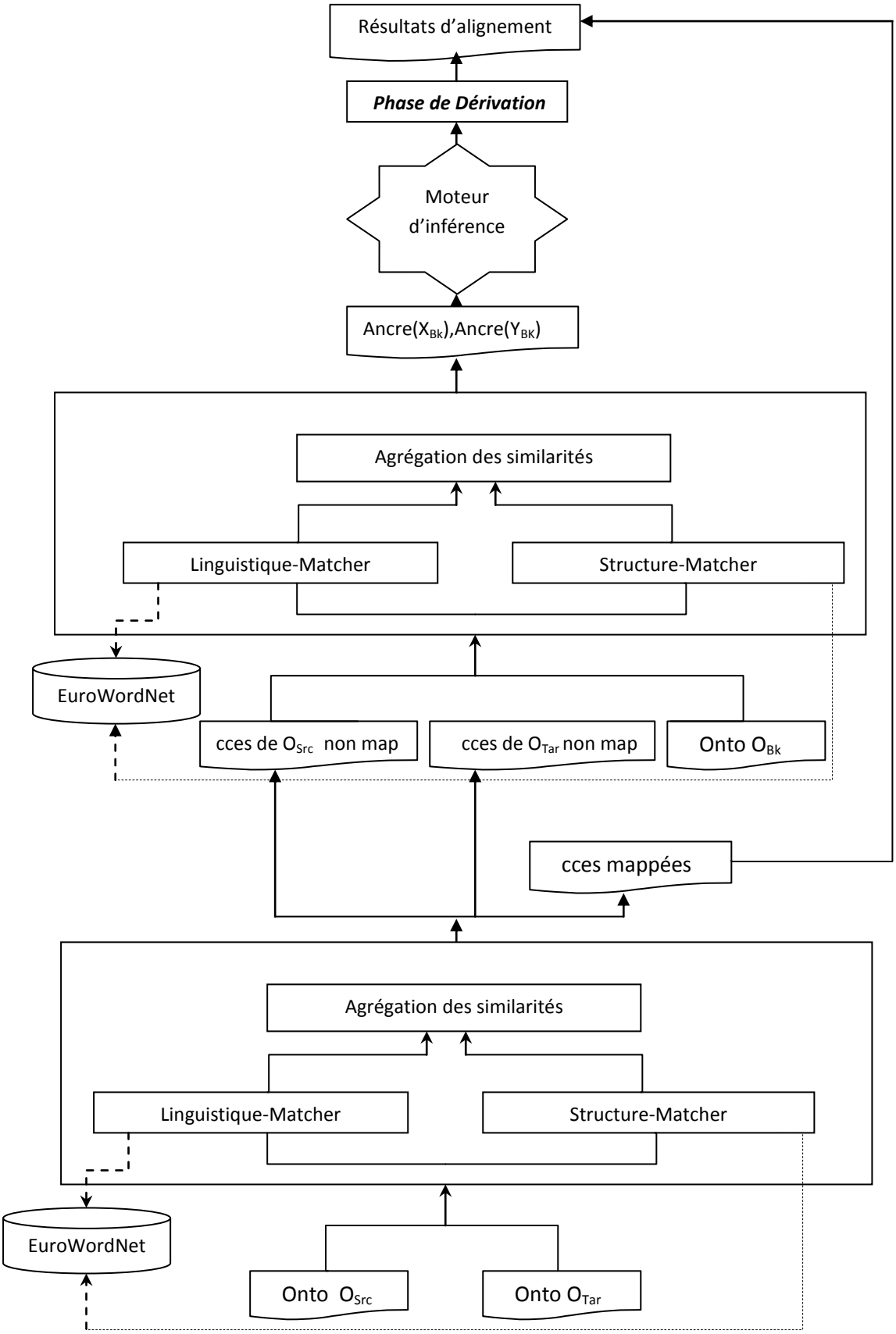

Figure III.1. Architecture de BRMAP

III.1.La phase d'alignement

Cette phase se base sur l'utilisation de l'algorithme XMAP [Djeddi et khadir, 2010], qui calcule les valeurs de similarité entre (O_{Src} et O_{Tar}). Ce calcul se fonde sur des mesures linguistiques ou structurelles qui sont appelées les valeurs de similarité partielles entre deux entités, et elles sont stockées dans une base des similarités(Vecteurs).

III.1.1. Description de l'algorithme XMAP

Cet algorithme utilise des connaissances représentées en OWL-DL, dont les descriptions des classes et des relations dans l'ontologie ont beaucoup des points communs au niveau syntaxique. Elles sont décrites en utilisant des primitives telles que rdf:id, rdfs:label pour leurs noms, leurs étiquettes, respectivement. Les classes et les relations sont également organisées dans des hiérarchies (la hiérarchie de classe et la hiérarchie de relation) grâce aux deux primitives rdfs:subClassOf et rdfs:subPropertyOf respectivement. Quand à la différence entre une classe et une relation dans leurs descriptions en OWL au niveau syntaxique, les deux primitives rdfs:domain et rdfs:range peuvent être utilisées pour spécifier le domaine et le co-domaine d'une relation.

Cet algorithme exploite les éléments communs à partir des descriptions des classes et des propriétés pour mesurer la similarité entre deux classes et deux propriétés, respectivement. Donc, dans cette partie, nous présentons des mesures de similarité pour les classes et la similarité entre les propriétés est calculée par des mesures construites de la même manière.

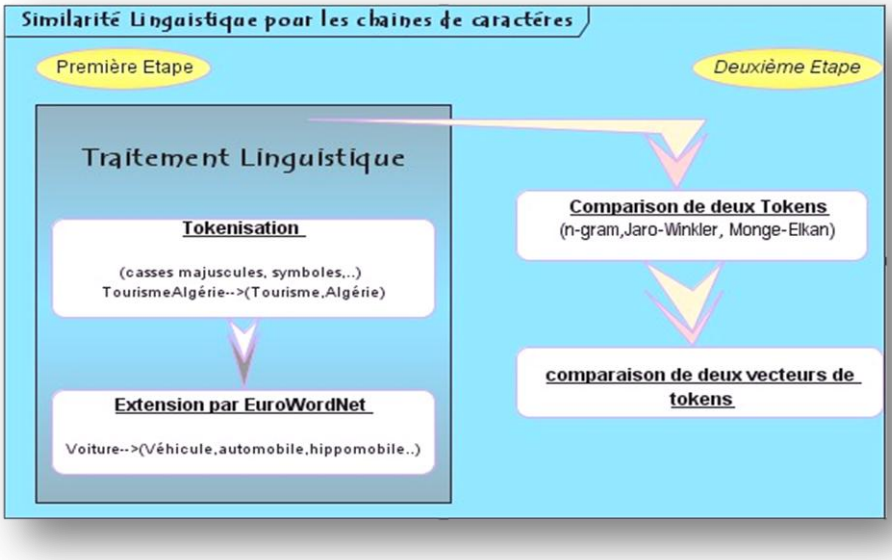

Figure III.2. Les calculs des similarités linguistiques pour les chaines de caractères

III.1.1.1 La similarité au niveau linguistique

La similarité linguistique de deux classes est calculée à partir des composantes linguistiques de la classe. En OWL, les composantes linguistiques dans une description d'une classe sont le nom de classe (spécifié par la primitive rdf:id), les étiquettes de la classe (spécifiées par une ou plusieurs primitives rdfs:label) et les commentaires de la classe (spécifiés par une ou plusieurs primitives rdfs:comment). Pour notre algorithme on s'intéresse seulement à la mesure de similarité construite à partir du nom de la classe car elle reflète son identification unique dans toute l'ontologie.

En OWL, les ressources, donc les classes ou les relations, sont identifiées par des URIs. Par exemple, la classe « Livre » est référencée en RDF(S) par l'URI « http://www.univ-annaba.fr/documents/exemple#Livre».<*rdfs:Class rdf:about=" http://www.univ-annaba.fr/documents/exemple#Livre " />*. L'URI se compose de deux parties : l'espace de noms (namespace) et le nom local.

Normalement, le nom d'une classe est une chaîne des caractères, sans espaces. Un nom de classe peut être un mot, un terme, ou une expression (une combinaison des mots). Ce nom est unique dans une ontologie pour identifier la classe. Le calcul de la valeur de similarité de deux noms est effectué dans deux étapes: la *normalisation* et la *comparaison.*

Fonction-1 **Normalisation(nom)**

résultat$_{normalisation}$ < – créer un ensemble vide
tokens < – Tokenisation(nom)
n < – nombre de tokens dans tokens
Pour i de **1** à ***n***
Tr < – Expansion(tokens[i])
Tr $< -Minusculation(Tr)$
résultat$_{normalisation}$ $< -Ajouter(Tr, résultat_{normalisation})$
Fin Pour

Retourner résultat$_{normalisation}$
Fin

L'étape de normalisation (*Fonction-1*) convertit un nom de classe en un ensemble d'unités lexicales des tokens. Un nom est découpé en plusieurs tokens grâce à la ponctuation, à la casse (majuscule), aux symboles spéciaux, aux chiffres. Par exemple, le nom de classe «AirCanada » est découpé en deux tokens «Air » et «Canada» ; le nom « Maison_de_Karim » est convertit à l'ensemble {« Maison», « de », «Karim»}. La normalisation du nom inclut une expansion de token : les abréviations, les acronymes sont élargis, par exemple le token «WS »

est élargi à {« Web », « Sémantique »}. Cette expansion est effectuée grâce à un dictionnaire externe, dont chaque entrée est une paire composée d'un token (abréviation ou acronyme) et d'un ensemble de mots qui correspondent au token. Le dictionnaire est soit construit spécialement pour le domaine où les ontologies à aligner se trouvent soit il s'agit d'un dictionnaire général contenant des termes communs. Les tokens dans l'ensemble de tokens sont enfin rendus minuscules pour être comparés après.

Fonction-2 $\mathbf{Sim_{max}(token, tokens)}$

$valeur_{max} < -0$
$n < -nombre\ de\ tokens\ dans\ tokens$
Pour i de **1** à $\boldsymbol{n}$
    $valeur_{sim} < -Mesure\_similarite(token, tokens[i])$
    Si $valeur_{sim} > valeur_{\max}$
        $valeur_{max} < - valeur_{sim}$
    Fin Si
Fin Pour

Retourner $valeur_{max}$
Fin

La troisième fonction est basée sur une stratégie pessimiste pour le calcule de mesure linguistique des deux concepts.

Fonction-3 $\mathbf{Sim_{moyenne}(token, tokens)}$

$valeur_{moyenne} < -0$
$Somme < -0$
$n < -nombre\ de\ tokens\ dans\ tokens$
Pour i de **1** à $\boldsymbol{n}$
    $Somme < -Somme + Mesure\_similarite(token, tokens[i])$
Fin Pour
$valeur_{moyenne} < -Somme/n$
Retourner $valeur_{moyenne}$
Fin

La comparaison de la similarité de deux noms de deux classes (*Fonction-2*) est la comparaison entre deux ensembles de tokens correspondant à ces noms. La similarité de deux tokens, qui sont actuellement des chaînes de caractères courtes, est calculée en employant la métrique n-gram (Définition 10), qui est basée sur le nombre et l'ordre de caractères communs entre deux chaînes des caractères.

Bien qu'ayant été proposée depuis les années quatre vingt et utilisée principalement en reconnaissance de la parole, la notion de **n-grams de caractères** prit davantage d'importance avec les travaux de [Greffenstette, 1995] sur l'identification de la langue et de [Damashek,

1995] sur le traitement de l'écrit. Le choix des n-grams apporte un autre avantage très important : il permet de contrôler la taille du lexique et de la maintenir à un seuil raisonnable pour de très larges noms. La taille du lexique était jusqu'à présent l'aspect le plus controversé et considéré comme une limite des techniques. Dans une approche avec découpage en n-grams de caractères, contrairement aux approches avec découpage en mots, il n'est pas question d'utiliser la lemmatisation pour réduire le lexique. La lemmatisation (qui consiste à remplacer une forme fléchie par son lemme) est, d'une part, relativement lourde à mettre en œuvre sur le plan informatique (elle utilise dans la plupart des cas un dictionnaire très volumineux), mais en plus, impose un traitement spécifique à chaque langue. La valeur de similarité de nom (S_{Nom}) entre deux noms est alors la moyenne des valeurs de similarité entre chaque token d'un ensemble et le token le plus similaire dans l'autre ensemble.

Définition 17 $(Similarité\ des\ noms)$. *Soit* n_1 *et* n_2 *deux noms de deux classes. Soit* N_1*et* N_2 *ensembles des tokens obtenus après la normalisation de* n_1 *et de* n_2*, respectivement. La similarité des noms est une fonction de la similarité* $S_{Nom} : S\ x\ S \rightarrow [0,1]$ *telle que*

$$\boldsymbol{S_{Nom}}(n_1, n_2) = \frac{\sum_{m_1 \in N_1} \boldsymbol{MJW}(m_1, N_2) + \sum_{m_2 \in N_2} \boldsymbol{MJW}(m_2, N_1)}{|N_1| + |N_2|} \qquad (14)$$

$$\text{où } \mathbf{MJW}(\mathrm{m_i}, \mathrm{N}) = \mathbf{max}_{\mathrm{m_j \in N}}\ \mathrm{S_{n-gram}}\ (\mathrm{m_i}, \mathrm{m_j})$$

$$|N_i|,\ i = 1,2,\ est\ la\ cardinalité\ de\ l'ensemble\ N_i$$

Fonction-4 $\boldsymbol{Similarité_{Nom}(nom1, nom2)}$

$tokens1 < -\boldsymbol{Normalisation}(nom_1)$
$tokens2 < -\boldsymbol{Normalisation}(nom_2)$
$n_1 < -nombre\ de\ tokens\ dans\ tokens\ 1$
$n_2 < -nombre\ de\ tokens\ dans\ tokens\ 2$
$somme1 < -0$
Pour i de 1 à n_1
  $sim < -\boldsymbol{Sim_{max}}(tokens[i], tokens2)$
  $somme1 < -somme1 + sim$
Fin Pour
$somme2 < -0$
Pour i de 1 à n_2
  $sim < -\boldsymbol{Sim_{max}}(tokens2[j], tokens1)$
  $somme2 < -somme2 + sim$
Fin Pour
si $n_1 = n_2 = 0$
$résultat < -1$
sinon
$résultat < -(somme1 + somme2)/(n_1 + n_2)$
Fin Si
Retourner $résultat$
Fin

Suivant la nature des ontologies à aligner : si ces ontologies partagent plusieurs classes ayant les mêmes noms, ω_{Nom} est mis à une valeur plus élevée par l'utilisateur, et dans le cas ou les concepts des deux ontologies à aligner sont largement différent, il faut affecter une valeur petite à ω_{Nom}, laissant une grande partie pour le calcule de similarité structurelle.

Définition 18 $(Similarité\ linguistique\ de\ deux\ classes)$***.*** *Soit A et B deux classes de deux ontologies. Soit Nom(x) qui retournent le nom. La similarité linguistique de deux classes A et B est une fonction de la similarité* $S_{Linguistique} : OxO \rightarrow [0,1]$ *telle que :*

$$S_{Linguistique}\ (A,B) = \boldsymbol{Similarité_{Nom}}\big(Nom(A), Nom(B)\big) * \omega_{Nom} \qquad (15)$$

Si cette valeur de similarité linguistique $S_{Linguistique}\ (A,B)$ excède un seuil prédéfini $T_{\text{Linguistique}} > 0$, nous considérons que la classe A de l'ontologie O_1 est linguistiquement similaire avec la classe B de l'ontologie O_2, et noté $\text{l} - \text{similar}(\text{A}, \text{B})$.

Définition 19 $(l - similar)$*. Soit A et B deux classes de deux ontologies. Soit* $T_{Linguistique}$ *un nombre réel non négatif. Deux classes A et B sont linguistiquement similaires et noté* $l - similar(A,B)$ *si et seulement si* $S_{Linguistique}\ (A,B) \geq T_{Linguistique}$.

III.1.1.2 La similarité au niveau structurelle

OWL fait la distinction entre deux types de propriétés : les propriétés d'objet permettent de relier des instances à d'autres instances et les propriétés de type de donnée permettent de relier des individus à des valeurs de données.

Une propriété d'objet est une instance de la classe owl:ObjectProperty, une propriété de type de donnée étant une instance de la classe owl:DatatypeProperty. Ces deux classes sont elles-mêmes sous-classes de la classe RDF rdf:Property.

La description par restriction de propriété permet de définir une classe anonyme composée de toutes les instances de owl:Thing qui satisfont une ou plusieurs propriétés. Ces contraintes peuvent être de deux types : contrainte de valeur ou contrainte de cardinalité. Une contrainte de valeur s'exerce sur la valeur d'une certaine propriété de l'individu (par exemple, pour un individu de la classe Humain, sexe = Homme), tandis qu'une contrainte de cardinalité porte sur le nombre de valeurs que peut prendre une propriété (par exemple, pour un individu de la classe Humain, aPourFrere est une propriété qui peut ne pas avoir de valeur, ou avoir plusieurs valeurs, suivant le nombre de frères de l'individu. La contrainte de cardinalité

portant sur aPourFrere restreindra donc la classe décrite aux individus pour lesquels la propriété aPourFrere apparaît un certain nombre de fois).

L'algorithme XMAP évalue la relation sémantique entre deux concepts en prenant en considération les types de contrainte de cardinalités et de valeurs entre leurs propriétés. Nous notons par C l'ensemble de concepts pour une ontologie donné. Pour un concept donné $c \in C$ nous notons par P(c) = {pi | pi(c)} l'ensemble de propriétés de c. À ce but, nous évaluons la similarité linguistique de propriétés aussi bien que le lien relationnelle entre le type de restriction pour chaque propriété.

La fonction $\boldsymbol{Restriction_{Type}}(r, r')$ *:* Le but de cette relation est de calculer une valeur dans l'intervalle [0,1] exprimant un lien relationnelle entre r et r′ basé sur leurs poids associés (voir la table III.1).

Restriction	*Poids*
OWLAllValuesFrom	*1.0*
OWLSomeValuesFrom	*1.0*
OWLMinCardinality>=1	*1.0*
OWLCardinality>=1	*1.0*
OWLMaxCardinality>=1	*1.0*
Autre restriction…	*0.5*

Tableau III.1. Les poids sont associés avec le type de restriction

La relation $\boldsymbol{Restriction_{Type}}(r, r')$ est rapporté au dessous, et la plus haute valeur (c.-à-d., 1.0) est obtenue quand r et r' ont le même poids.

Fonction-5 $\boldsymbol{Restriction_{Type}}(\mathbf{r}, \mathbf{r}')$

Entrée type de restriction r et r′
Sortie valeur dans l'intervalle [0,1] exprimant un lien relationnelle entre r et r′
Début fonction
Def ω_r et $\omega_{r'}$ comme les poids associées avec r et r′,respectivement.
Def $\kappa = 0$;
$\kappa = 1 - |\omega_r - \omega_{r'}|$;
Retourner κ;
Fin

La fonction $Similarité_{structurelle}$: Le calcul de la similarité structurelle est achevé en exploitant une fonction de deux concepts $c \; et \; c'$.

$$Similarité_{structurelle} \; (Description_{contexte} \; (c), Description_{contexte} \; (c')) \qquad (16)$$

Dans cette fonction, la description contextuelle d'un concept c est représentée à travers un vecteur$(c) = (dsp_1, \dots, dsp_n)$ ou $\forall i \in (1, \dots, n)$, $dsp_i = (p_i, r_i)$, dont p_i dénote la iéme propriétés et r_i dénote la restriction de p_i. Un facteur de contrôle F_k a été introduit pour raffiner les résultats de l'évaluation. En particulier, dans la présence de très basse valeur, F_k les augmente proportionnellement pour mieux équilibrer toute les résultats et évite de trop grands intervalles entre eux. Pour chaque propriété $p_i \in P(c)$ est alignée contre tous les propriétés $p_j \in P(c')$ en utilisant les deux fonctions $\boldsymbol{Similarité_{Nom}}(p, p')$ et $\boldsymbol{Restriction_{Type}}(r, r')$.

Fonction-5 $\boldsymbol{Similarité_{structurelle}}(\boldsymbol{Description_{contexte}}(c), \boldsymbol{Description_{contexte}}(c'))$

Entrée les vecteur $Description_{contexte}\ (c)$ et $Description_{contexte}\ (c')$ qui représentent la description structurelle d'un concept c et c', respectivement.
Sortie une mesure de similarité structurelle
Début fonction
Def *x=0, y=0, z=0 ;*
Pour chaque $dsp \in Description_{contexte}\ (c) | dsp = (p, r)$;
  Pour chaque $dsp' \in Description_{contexte}\ (c') | dsp'(p', r')$;
    Si $r = \emptyset \ ||\ r' = \emptyset$
      $y = \boldsymbol{Similarité_{Nom}}(p, p')$;
    Sinon
      $y = \boldsymbol{Similarité_{Nom}}(p, p') \cdot \boldsymbol{Restriction_{Type}}(r, r')$;
  $z = z + y$;

$z = z \div (length(Description_{contexte}\ (c)) \cdot length(Description_{contexte}\ (c')))$;
$F_k = 1 + (1 - z)$ est un facteur de contrôle
$\boldsymbol{x = z \cdot F_k}$;
Retourner $\boldsymbol{x}$;
Fin

Dans XMAP la précision dépend du choix du concept nommé dans la définition de l'ontologie. Les noms significatifs et précis garantiront des résultats plus appropriés.

L'algorithme XMAP : L'entrée de l'algorithme XMAP est constituée de deux concepts c et c' ; la valeur du poids ω_{LA} , et 0.5 est la valeur par défaut. $\omega_{LA} = 0.5$ assure que la mesure linguistique et la mesure basée sur la nature de description de propriétés ont le même impact pendant le calcul de la mesure finale, et si $\omega_{LA} = 1.0$ alors seulement la mesure linguistique est considérée. La sortie de XMAP est une mesure exprimant le degré de similarité entre deux concepts.

Algorithme $\boldsymbol{XMAP(c, c', \omega_{LA})}$

Entrée les deux concepts c *et* c', *et le poids* ω_{LA}
Sortie similarité finale entre c *et* c'
Début Algorithme
Def n et n′ sont les noms de c et c′, respectivement ;
Def $Description_{contexte}\ (c) = []$, $Description_{contexte}\ (c') = []$ *sont les vecteurs de description de propriété de c et* c'*,respectivement ;*
Def $portion - dsp = []$ *comme un paire de la forme* (np, tr)*, dont np est le nom associé a une propriété et* $tr \in$ `{OWLSomeValuesFrom, OWLMinCardinality>=1, OWLCardinality>=1, Autre restriction}`
Def $x = 0, y = 0$ et $Similarite_{finale} = 0$;
$P(c) = \{p_i | (p_i, c, tr)\}$;

$x = \boldsymbol{Similarit}é_{Nom}(n, n')$;
pour chaque *propriété* $p(c) \in P(c)$
$/* tr(p_i, c)$ est une relation qui exprime le type de restriction de p_i qui $\in c$
$portion - dsp = [p(c), tr(p_i, c)]$;
Ajouter $portion - dsp$ dans $Description_{contexte}\ (c)$;
pour chaque *propriété* $p(c') \in P(c')$
$/* tr(p_i, c')$ est une relation qui exprime le type de restriction de p_i qui $\in c'$
$portion - dsp = [p(c'), tr(p_i, c')]$;
Ajouter $portion - dsp$ dans $Description_{contexte}\ (c')$;
$y = Similarité_{structurelle}\ (Description_{contexte}\ (c), Description_{contexte}\ (c'))$;
$Similarite_{finale}\ (c, c') = \omega_{LA} \cdot x + (1 - \omega_{LA}) \cdot y$;
retourner $Similarite_{finale}\ (c, c')$;
Fin

III.2. La phase d'ancrage

Cette phase consiste tout d'abord à apparier chacun des 2 concepts X_{Src} et Y_{Tar}, pris respectivement de connaissances des deux ontologies source et cible non alignées dans la première phase, avec un ou des concepts de la 3ème ontologie (O_{BK}), c'est-à-dire, à identifier des mappings de la forme (X_{Src} *relation* X_{BK}) et (Y_{Tar} *relation* Y_{BK}) en utilisant l'algorithme XMAP [Djeddi et khadir, 2010], où X_{BK} et $Y_{BK} \in C_{BK}$ et sont appelés des *ancres* ou *points d'ancrage*.

III.2.1. Description de l'algorithme d'ancrage

L'application de XMAP sur (X_{src} et X_{BK}) et (X_{Tar} et X_{BK}) nous donne deux vecteurs. Le premier vecteur contient les ancres X_{Bk} et le concept X_{Src} correspondant, le deuxième vecteur est l'ensemble des ancres Y_{Bk} et le concept X_{Src} correspondant. On extrait de ses deux vecteurs les ancres qui vont être l'entrée de l'algorithme de la phase suivante.

Algorithme ***Ancrage(X_{Src},X_{Bk})***

Entrée les deux concepts $x_{Src} \in X_{Src}$ et $x_{Bk} \in X_{Bk}$, .
Sortie *ancre*
Début Algorithme
Def XMAP(x_{Src}, x_{Bk}, w_{LA}) est un algorithme qui évalue la relation sémantique entre deux concepts et retourne comme résultat x_{Src}, x_{Bk}, relation $r \in R=\{\leq, \geq, \equiv\}$;
Def Ancre(x_{Bk})=[] est un vecteur contenant l'ensemble des ancres ;
Def VMap(x_{Src}, x_{Bk},r)=[] est un vecteur contenant x_{Src}, x_{Bk} et la relation entre eux ;

pour chaque concept $x_{Src} \in X_{Src}$
VMap(x_{Src}, x_{Bk}, r)=XMAP(x_{Src}, x_{Bk}, w_{LA}) ;
pour chaque valeur de VMap(x_{Src}, x_{Bk}, r)
Ancre(x_{Bk}, x_{Src})=[x_{Bk}, x_{Src}] ;
retourner Ancre(x_{Bk}, x_{Src}) ;
Fin.

NB : idem pour Ancrage (Y_{Tar},Y_{Bk}).

III.3. Phase de dérivation :

Cette phase s'appuie sur la structuration de l'ontologie O_{BK} et le raisonneur Pellet. Ce raisonneur des méthodes sémantiques permettant d'effectuer des déductions sur les connaissances d'O_{Bk} par l'application de techniques de raisonnement, ces méthodes s'appuient sur des formalismes de représentation des connaissances fondées sur des logiques formelles, et les techniques citées au dessous appliquent les algorithmes de subsomption pour déduire les correspondances entre les ancres ;

1) subsomption : ($X_{BK} \leq Y_{BK}$) ou ($X_{BK} \geq Y_{BK}$)

2) équivalence : ($X_{BK} \leq Y_{BK}$ et $Y_{BK} \leq Y_{BK}$) signifie que ($X_{BK} \equiv Y_{BK}$)

Afin d'avoir des mappings de la forme (X_{Src} *relation* Y_{Tar}), l'ensemble R des relations utilisées est l'ensemble $\{\leq, \geq, \equiv\}$ où $X \leq Y$ peut se lire, suivant les cas, *«X isA Y», «X part-of Y»* ou plus généralement *«X narrower-than Y»*. Les mappings cherchés sont dérivés en exploitant des règles de la forme :

- Si ($X_{Src} \leq X_{BK}$) et ($X_{BK} \leq Y_{BK}$) et ($Y_{BK} \leq Y_{Tar}$) alors ($X_{Src} \leq Y_{Tar}$)
- Si ($X_{Src} \geq X_{BK}$) et ($X_{BK} \geq Y_{BK}$) et ($Y_{BK} \geq Y_{Tar}$) alors ($X_{Src} \geq Y_{Tar}$).

Ces règles utilisent aussi la relation d'équivalence, ≡, en considérant que l'existence d'une relation de type $A \equiv B$ permet de rajouter les deux relations $A \leq B$ et $A \geq B$ et qu'inversement, le fait d'avoir pu dériver les deux relations $X_{Src} \leq Y_{Tar}$ et $X_{Src} \geq Y_{Tar}$ permet de dériver la relation $X_{Src} \equiv Y_{Tar}$.

Ces différentes règles permettent ainsi de dériver des mappings « sémantiques », i.e. des mappings reliant deux concepts par un lien de type *isA* ou *isEq* dont la sémantique est bien définie et qui peuvent être justifiés et prouvés par des mécanismes d'inférences (Sabou *et al.*, 2006). Cette phase est assurée par l'algorithme décrit au dessous.

Algorithme ***Dérivation (X_{Src},X_{Tar})***

Entrée Les deux Vecteurs VMap(x_{Src}, x_{Bk},r) et VMap(y_{Tar},y_{Bk},r) ;
Sortie Relation entre x_{Src}, x_{Bk}
Début Algorithme
Def Reasoner(x_{Bk},y_{Bk}) fonction utilise le raisonneur Pellet et retourne r ∈ R={≤, ≥, ≡ };
Def DérivBack(x_{Bk},y_{Bk},r) =[] vecteur contenant des ancres et la relation r entre eux;

pour chaque concept $x_{Src} \in X_{Src}$
 pour chaque concept $x_{Tar} \in X_{Tar}$
  r=Reasoner(x_{Bk},y_{Bk}) ;
  DérivBack(x_{Bk},y_{Bk},r)=[x_{Bk},y_{Bk},r];
pour chaque élément de DérivBack(x_{Bk},y_{Bk},r)
 si ($x_{src} \leq x_{bk}$) et ($x_{bk} \leq y_{bk}$) et ($y_{bk} \leq y_{tar}$) **alors** ($x_{src} \leq y_{tar}$)
  Retourner x_{src} *isA* y_{tar} ;
 si ($x_{src} \geq x_{bk}$) et ($x_{bk} \geq y_{bk}$) et ($y_{bk} \geq y_{tar}$) **alors** ($x_{src} \geq y_{tar}$).
  Retourner y_{tar} *isA* x_{src} ;
 si ($x_{src} \leq y_{tar}$) et ($x_{src} \geq y_{tar}$) **alors**
  Retourner x_{src} *isEq* y_{tar} ;
  Fin .

Chapitre IV

Nous décrivons ici le cadre dans lequel notre outil d'alignement BkMap a été conçu. Nous présentons tout d'abord les outils utilisés, le contexte d'utilisation et les différentes répercussions de ce contexte sur le mécanisme d'alignement mis en œuvre, puis l'architecture générale de l'outil.

Le développement de notre application est fait en Java car ce langage offre plusieurs API permettant de manipuler les ontologies : OWL-API, Jena. De plus, il existe également plusieurs API de calcul de mesures de similarité réalisées en Java : n-gramme,… Nous les réutilisons dans notre application avec d'autres librairies comme Pellet, en plus d'autres algorithmes que nous avons développés.

IV.1 les outils utilisés

IV.1.1 Eclipse Ganymede

Eclipse est un environnement de développement intégré spécialement conçu pour le langage de programmation Java. Le logiciel est entièrement gratuit, open-source, mais est également extensible. Ainsi, la partie servant à développer en Java n'est qu'une partie des plug-ins qu'utilise Eclipse. En effet, d'autres plug-ins peuvent être utilisés afin de développer tous les langages et tous les formats de fichiers supportés. Il est ainsi possible de programmer en Java, en PHP, en XML, en HTML, en C#, ou encore en C++.

Grâce à son interface complète et accessible, Eclipse va permettre de développer des sites Web et logiciels en toute simplicité. En plus d'être doté de nombreuses performances, Eclipse est gratuit et open-source ce qui fait de lui un concurrent de taille face aux autres environnements de développement qui eux, sont généralement payant.

La base de cet environnement de développement intégré est l'Eclipse paltform qui est composée de : Platform Runtime (qui démarre la plateforme et gère les plug-ins), SWT (la bibliothèque graphique de base de l'EDI), JFace (bibliothèque graphique de plus haut niveau), Eclipse Workbench (dernière couche graphique permettant de manipuler des composants).

Nous avons utilisé Eclipse SDK version 3.4.1 pour programmer notre outil d'alignement d'ontologie.

IV.1.2. Protege

De nombreux outils permettent aujourd'hui d'éditer des ontologies. Dans tous les cas, force est de constater qu'aucun des plusieurs outils n'a réussi à s'imposer a l'exception de *Protege.* PROTEGE-200053 [NOY N., et al 2000] est un environnement graphique de développement d'ontologies développé par le SMI de Stanford. Dans le modèle des connaissances de PROTEGE, les ontologies consistent en une hiérarchie de classes qui ont des attributs (slots), qui peuvent eux-mêmes avoir certaines propriétés (facets). L'édition des listes de ces trois types d'objets se fait par l'intermédiaire de l'interface graphique, sans avoir besoin d'exprimer ce que l'on a à spécifier dans un langage formel : il suffit juste de remplir les différents formulaires correspondant à ce que l'on veut spécifier. Ce modèle autorise d'ailleurs une liberté de conception assez importante puisque le contenu des formulaires à remplir peut être modifié suivant les besoins via un système de métaclasses, qui constituent des sortes de « patrons » de connaissance. L'interface, très bien conçue, et l'architecture logicielle permettant l'insertion de plugins pouvant apporter de nouvelles fonctionnalités ont participé au succès de PROTÉGÉ-2000. Aujourd'hui, il regroupe une large communauté d'utilisateurs et bénéficie des toutes dernières avancées en matière de recherche ontologique : compatibilité OWL de référence, services inférentiels, gestion de bases de connaissances, visualisation d'ontologies, alignement et fusion, etc.

On peut le considérer non pas comme un simple environnement de construction d'ontologies basé sur les *frames*, mais comme un *framework* logiciel ouvert capable d'intégrer de multiples *plugins* implémentant les nombreuses recherches effectuées autour des ontologies.

L'intégration des plugins se fait à l'aide d'une interface très bien conçue, et une architecture logicielle permettant l'insertion de plugins pouvant apporter de nouvelles fonctionnalités qui ont participé au succès de PROTÉGÉ-2000. Aujourd'hui, il regroupe une large communauté d'utilisateurs et bénéficie des toutes dernières avancées en matière de recherche ontologique : compatibilité OWL de référence, services inférentiels, gestion de bases de connaissances, visualisation d'ontologies, alignement et fusion, etc.

53 http://protege.stanford.edu/index.shtml

Protégé est également une librairie Java open source qui peut être étendue pour créer de véritables applications à bases de connaissances.

Nous avons utilisé Protege_3.4.4 parce qu'il intègre le raisonneur pellet qu'on exploite autant que librairie dans le développement de notre application et le plugin DataMaster. En plus permet l'intégration de BRMAP54.

IV.1.2.1. Le Plugin DataMaster

DataMaster est Spécialisé dans l'import des structures et des données de bases de données relationnelles [Nyulas, 2007], il propose une méthode d'import des tables de la base de données relationnelle comme des concepts OWL.
Avec DataMaster, le paradigme relationnel est conservé puisque les données extraites de la base sont simplement inscrites dans l'ontologie générique *Relational.OWL* [de Laborda et al.,2005].

Cet outil n'est pas dimensionné pour effectuer une restructuration profonde du modèle et des données sources, il permet néanmoins de migrer les données vers une description dans une ontologie. En utilisant DataMaster avec notre base de données, chaque table a été convertie en un concept et chaque ligne de la table a été convertie en une instance du concept correspondant. Les valeurs des attributs ont été instanciées avec les valeurs des champs correspondants de la table.

IV.1.2.2. La librairie Pellet

Pellet [E. Sirin et al, 2007], [Clark and Parsia, 2009] est une librairie open-source Java qui implémente des raisonnements applicables aux ontologies OWL. Elle est utilisable en conjonction avec Jena et les API OWL. Parmi les fonctionnalités offertes par Pellet, on peut citer:

– la vérification de la consistance d'une ontologie ;

– la classification pour les concepts des taxonomies ;

– les tests de subsomption ;

–l'analyse et réparation des ontologies qui implique une vérification des documents OWL dans la syntaxe RDF, avec le but d'identifier et de corriger les fautes de syntaxe.

– le raisonnement sur des types de données.

54 BRMAP: "BACKGROUND_RAISONER MAPPING", Voir ANNEXE B

Pellet est un raisonneur puissant basé sur l'algorithme des tableaux, le seul qui prenne en charge pour les classes énumérées et les types de données définis par l'utilisateur [Mindswap, 2003], [E. Sirin et al, 2007]. Pour résoudre le problème des URI qui désignent les nouveaux types, Pellet adopte la solution proposée par DAML+OIL : la construction d'un nouvel URI à partir de l'URI du document contenant le schéma XML et le nom local du type simple [Mindswap, 2003]. Le document identifié par l'URI et qui contient le schéma XML sera analysé avec le but de localiser le type dont l'attribut name contient le nom de la définition. En conséquence, OWL pourra utiliser des types de données simples définis au premier niveau d'un document XML Schéma, en les référençant par l'URI du document et le nom local du type [Mindswap, 2003].

La nouvelle version, Pellet 2.0 est compatible avec le langage OWL 2 [E. Sirin et al, 2007] et intègre diverses techniques d'optimisation, y compris pour les nominaux, les requêtes conjonctives, le raisonnement progressif, etc. [E. Sirin et al, 2007].

IV.2. Contexte d'utilisation

Le contexte dans lequel a été conçu l'outil BRMap est celui de l'ingénierie ontologique où les ontologies sont hétérogène et en continuelle évolution, ce qui engendre différentes versions, cette hétérogénéité met les développeurs et les ingénieurs de la connaissance dans la confusion, ne sachant pas ce qui a changé, l'alignement va permettre d'identifier les différences entre deux versions: les entités qui ont été ajoutés, supprimés ou renommés. Dans la suite de ce chapitre nous allons prendre comme exemple l'alignement des deux ontologies décrivant la topologie et le diagnostic de la turbine à vapeur55 dans le domaine industriel afin faciliter la compréhension de cette installation aux ingénieurs.

IV.2.1. La construction des ontologies

Il existe plusieurs outils permettant l'édition d'ontologies, et nous avons choisi Protege 2000 décrit précédemment parce qu'il permet de décrire les ontologies en OWL, et il intègre le plugin DataMaster qu'on utilise pour construire les trois ontologies exploitées dans le test de notre application ;

55 Voir ANNEXE A

-L'ontologie source $O_{Src}56$ décrivant les cas d'interventions où chaque cas est défini par la cause de panne, défaut, symptôme et remède, extraite de la base de données « diagnostic.mdb » ;

-L'ontologie target $O_{Tar}30$ décrivant les caractéristiques (code, description, zone, fonction,….) des composants, extraite de la base de données « centraletopo.mdb »;

-L'ontologie complémentaire $O_{Bk}30$ décrit les cas d'interventions et les caractéristiques de la turbine à vapeur, extraite des deux bases de données précédentes ;

La construction des deux premières ontologies se fait à partir des bases de données de type Access à l'aide du Plug-in DataMaster. Pour ce faire, on doit tout d'abord créer une connexion ODBC (Open DataBase Connectivity) entre Protégé 2000 et la base de données en suivant les étapes ci-dessous :

- Démarrer/panneau de configuration ;
- Sélectionner outils d'administration ;
- L'option source de données (ODBC) ;
- Sélectionner « ms Access data base » (pour une base de données sous Access) ;
- Cliquer sur le bouton « configurer » et donner un nom à la base de données ;
- Cliquer sur le bouton « sélectionner » pour indiquer le chemin de la base de données ;
- Cliquer sur « OK » jusqu'à ce que toutes les fenêtres soient fermées.

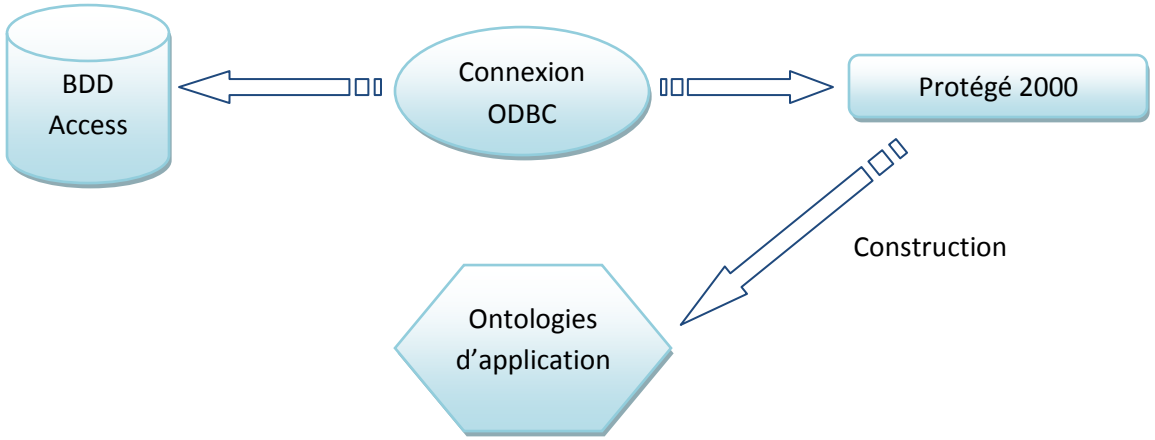

Figure IV.1. Etape de construction de deux ontologies

56 Voir ANNEXE C

Une fois la connexion établie, et après configuration du Protégé avec le plug-in DataMaster, on peut importer par exemple la base de données du diagnostic des équipements, qui regroupe plusieurs tables représentant les différents systèmes appartenant à la centrale. Chaque système faisant partie d'une tranche spécifique (Figure IV.2, page suivante).

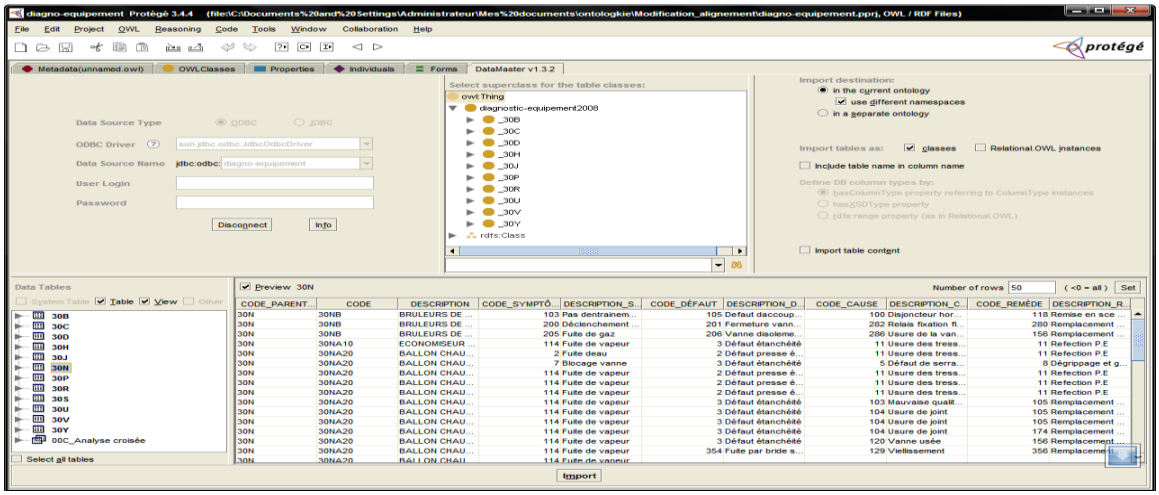

Figure IV.2. Ouverture de la base de données « diagnostic-equipement » dans DataMaster.

Donc, l'importation de la base de données va générer un ensemble de classes et de sous classe (les systèmes de la centrale). Après réorganisation de ces classes, on obtiendra l'ontologie suivante (Figure IV.3) :

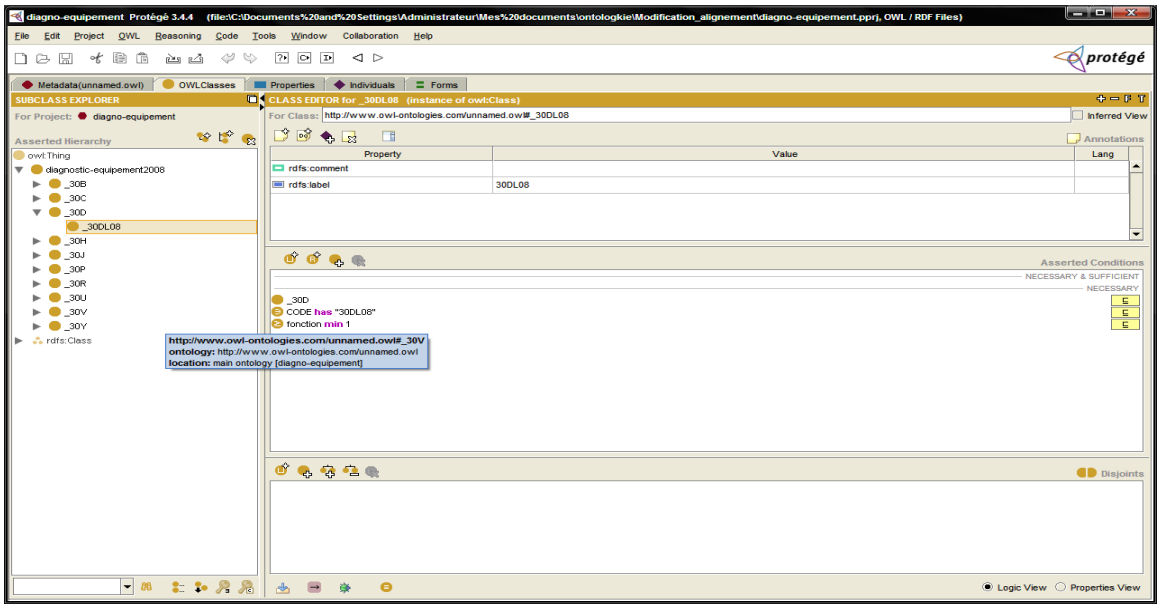

Figure IV.3. Présentation d'une partie hiérarchique de l'ontologie « diagno-equipement »

Par contre la construction de la troisième ontologie se fait d'une manière semi-automatique. En commençant par importer les deux bases de données citées précédemment à l'aide du DataMaster. Puis on ajoute manuellement les relations existantes entre les différents concepts de l'ontologie engendrée (voir figure IV.4).

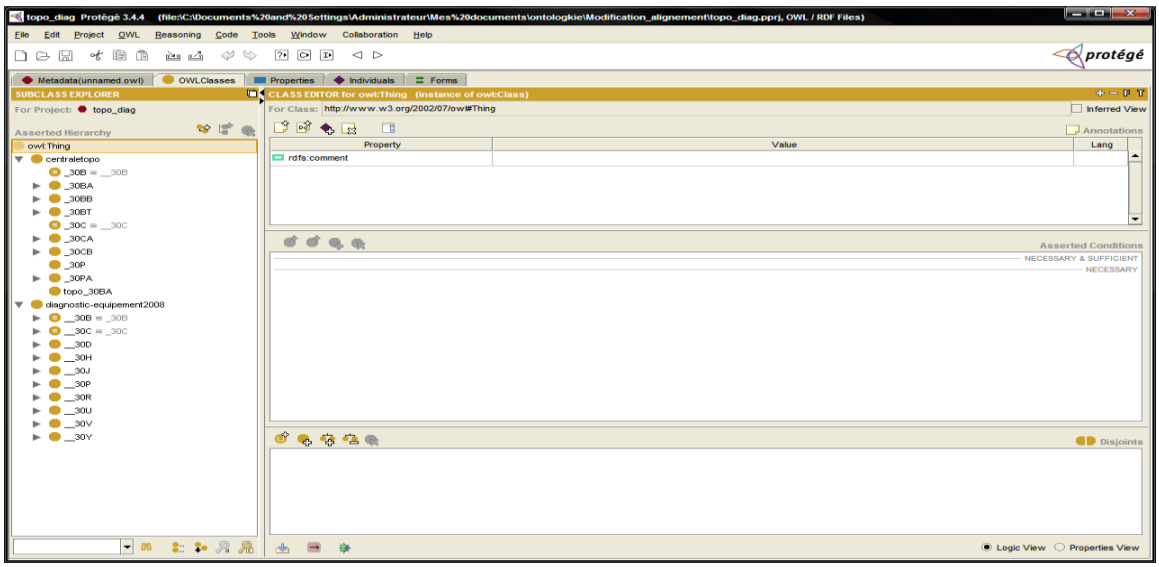

Figure IV.4. Présentation d'une partie hiérarchique de l'ontologie complémentaire «topo_ diag »

IV.3. Tests, résultats et discussions

Cette section montre les résultats du teste de notre système d'alignement sur les ontologies décrites dans la section précédente en comparant ces résultats par rapport à ceux obtenus quand par l'alignement de référence, COMA++ et XMAP.

IV.3.1. L'alignement de référence :

L'alignement manuel (de référence) doit se faire sur la base [Djeddi et khadir, 2010]

- des connaissances de l'expert du domaine des ontologies ;
- des définitions fournies par les thésaurus lexicaux tels que WordNet ;
- L'intuition de la personne qui réalise l'alignement, Mr J.Euzenat dit que « *si le problème de l'alignement est soumis à deux personnes différentes vous obtiendrez deux alignements différents qui ne sont pas forcément exempts d'erreurs* ».

Pour pallier à *la subjectivité* de cet alignement manuel, l'idéal serait de disposer : d'une ontologie de référence du domaine de connaissances des ontologies à aligner, afin de lever le voile sur les ambiguïtés concernant les différentes significations possibles d'un terme. Un exemple de ce type d'ontologie pour le domaine de l'anatomie humaine «The FMA ontology» qui est considérée comme telle car elle répond à un certain nombre de critères, comme le fait de couvrir une partie considérable de ce vaste domaine de connaissances mais aussi d'autres caractéristiques décrites en détail dans [Zhang et Bodenreider., 2005]. Ou encore d'alignements de référence entre les ontologies d'un même domaine fournis, par exemple, par les campagnes d'évaluation des systèmes automatiques d'alignement.

Remarque : un alignement manuel est toujours discutable, néanmoins, il doit être harmonieux et répondre au même raisonnement pour tous les alignements produits.

Démarche suivie

Nous disposons de deux ontologies : O1 et O2 à aligner. Les systèmes d'alignement sélectionnés ainsi que la plupart des systèmes automatiques existants alignent les ontologies par paire (deux par deux), donc nous réaliserons ce multi-alignement d'ontologies comme suit:

Figure IV .5. Ordre des alignements

IV.3.2. Métriques d'évaluation d'alignement

Les mesures de *Précision*, *Rappel* et *Fallout* (Do *et al.*, 2002) ont été des métriques largement exploitées pour estimer la qualité des alignements obtenus. Le EON3 "Evaluation of Ontology-based Tools" (EON, 2004, EON, 2006, Euzenat *et al.*, 2006) retient ces mesures pour l'évaluation de la qualité de l'alignement. L'objectif principal de ces mesures est l'automatisation du processus de comparaison des méthodes d'alignement ainsi que l'évaluation de la qualité des alignements produits. La première phase dans le processus d'évaluation de la qualité de l'alignement consiste à résoudre le problème manuellement. Le résultat obtenu manuellement est considéré comme l'alignement de référence. La comparaison du résultat de l'alignement de référence avec celui de l'appariement obtenu par la méthode d'alignement produit trois ensembles : *Nfound*, *Nexpected* et *Ncorrect*. L'ensemble *Nfound* représente les paires alignées

avec la méthode d'alignement. L'ensemble *Nexpected* désigne l'ensemble des couples appariés dans l'alignement de référence. L'ensemble *Ncorrect* est l'intersection des deux ensembles *Nfound* et *Nexpected*. Il représente l'ensemble des paires appartenant à la fois à l'alignement obtenu et l'alignement de référence. La *précision* est le rapport du nombre de paires pertinentes trouvées, *i.e.*, "*Ncorrect*", rapporté au nombre total de paires, *i.e.*, "*Nfound*". Il renvoie ainsi, la partie des vraies correspondances parmi celles trouvées. Ainsi, la fonction *précision* est définie par :

$$précision = \frac{|Ncorrect\ |}{|Nfound\ |}. \qquad (14)$$

Le *rappel* est le rapport du nombre de paires pertinentes trouvées, "*Ncorrect*", rapporté au nombre total de paires pertinentes, "*Nexpected*". Il spécifie ainsi, la part des vraies correspondances trouvées. La fonction *rappel* est définit par :

$$rappel\ = \frac{|Ncorrect\ |}{|Nexpected\ |} \qquad (15)$$

La mesure *Fallout* permet d'estimer le pourcentage d'erreurs obtenu au cours du processus d'alignement. Elle est définie par le rapport des paires erronées, "(*Nfound* − *Ncorrect*)", rapporté au nombre total des paires trouvées,"*Nfound*",

$$Fallout = \frac{N_{found}\ - |Ncorrect\ |}{|N_{found}\ |} \qquad (16)$$

IV.3.3. Résultats d'alignement des ontologies

IV.3.3.1. Alignement Manuel[57] : $O_1 \rightarrow O_2$

Concept d'ontologie Source « diagnostic »	Relation	Concept d'ontologie Target « centraletopo »
30C	Equivalence	30C
30B	Equivalence	30B
30CO42	Sous classe	30C
30CJ22	Sous classe	30C
30BA10	Sous classe	topo_30BA
diagnostic-equipement2008	Super classe	30PA21S161
30P	Super classe	30PA21S161

Tableau IV.1. Un exemple d'une partie d'alignement de référence

57 Voir ANNEXE D

IV.3.3.2. Alignement COMA++[58] : $O_1 \rightarrow O_2$

Etant donné que COMA++ est un système combinatoire, il est nécessaire avant de lancer l'exécution de ce programme, de stabiliser une combinaison des différents paramètres de ce système pour donner les meilleurs résultats possibles pour chaque alignement. Voici un tableau qui résume les combinaisons de paramètres (choix des matchers, l'agrégation, la direction, la sélection et la similarité combinée) testés par les concepteurs de COMA dans le cadre de la réalisation de séries de tests pour les besoins de l'évaluation de système d'alignement automatique.

Matchers		Aggregat	Direction	Selection	CombSim
No reuse	**5 single**		**-LargeSmall**	**-MaxN(1-4)**	**-Average**
	11 combinaisons	**-Max**	**-SmallLarge**	**-Delta(0.01-0.1)**	**-Dice**
		-Average	**-Both**	**-Thr(0.3-1.0)**	
		-Min		**-Thr(0.5)+MaxN(1-4)**	
Reuse	**2 single**			**-Thr(0.5)+Delta(0.01-0.1)**	
	12 combinaisons	**-Max**			**-Average**
		-Average			
		-Min			
∑= 16+14		**3**	**3**	**36**	**2**

Tableau IV.2. Les différentes combinaisons de paramètres testées par les concepteurs de COMA [Do et Rahm, 2002]

La configuration de COMA++ retenue pour l'alignement est donnée selon le tableau suivant :

Choix des Matchers	Il s'agit de la configuration par défaut de COMA, a savoir : les matchers hybrides suivants : Nom, Nom-chemin, Feuilles et Descendants, recommandés par les deux articles suivants : [Do et al., 2002] et [Do et Rahm., 2007], comme étant la meilleure combinaison de matchers dans la plupart des tests réalisés ;

58 Voir ANNEXE D

Agrégation	Average (paramètre par défaut): Considérer la moyenne des valeurs de similarité obtenues par les matchers (Tous les matchers ont la même importance) ;
Direction	BOTH d'O1 vers O2, pour les raisons suivantes : ▪ La direction par défaut dans COMA est BOTH (dans les deux directions), cette direction signifie que les alignements produits sont le résultat de **l'intersection** des alignements dans les deux directions. ▪ BRMAP, XMAP, Prompt et Falcon permettent de réaliser l'alignement dans les deux directions, donc, pour tenter de maximiser le nombre de critères communs entre ces approches à comparer, nous optons pour le choix de deux directions ;
Sélection	Nous allons effectuer des tests sur les paramètres de la sélection (Seuil, MaxN et MaxDelta), en prenant en considération, dans un premier temps, un paramètre à la fois, puis nous allons combiner deux paramètres ensuite trois paramètres « *a single approach may return imprecise match candidates. While Threshold may return too many match candidates, MaxN and MaxDelta may return match candidates with too little similarity. Thus, we support considering several criteria at the same time, in particular MaxN or MaxDelta in combination with a low threshold, e.g. 0.5* " [Do et Rahm., 2002a];
Similarité Combinée	Average, Nous optons pour la configuration par défaut qui est la moyenne des valeurs de similarité obtenues dans l'étape précédente.

Tableau IV.3. La configuration de COMA++ retenue pour l'alignement

Matchers	**Agrégation**	**Direction**	**Sélection**	**Similarité Combinée**
Nom, Nom-chemin, Feuilles et Descendants	Average	Both	Seuil, MaxN et MaxDelta	Average

Tableau IV.4. Récapitulatif de la sélection des paramètres de COMA++

Dans cet alignement nous retenons la configuration du tableau IV.4 en variant les paramètres de sélection (Seuil, MaxN et MaxDelta), les résultats de ce test ont illustré dans le tableau suivant :

Paramètres de Sélection	$N_{expected}$	N_{found}	$N_{Correct}$	Précision	Rappel	Fallout
Seuil(S) -> (0.1 - 0.5)	60	130	18	0.1384	0.3	0.5384
MaxDelta(D) -> (0.01 - 0.1)		130	18	0.1384	0.3	0.5384
MaxN (N)-> (1 – 5)		130	18	0.1384	0.3	0.5384
S=0 .5 *D-> (0.01 - 0.1)* *N-> (1 – 5)*	60	130	18	0.1384	0.3	0.5384
S-> (0.1 - 0.5) *D = 0.1* *N-> (1 – 5)*	60	130	18	0.1384	0.3	0.5384
S-> (0.1 - 0.5) *D -> (0.01 - 0.1)* *N= 5)*	60	130	18	0.1384	0.3	0.5384

Tableau IV.5. Résultats du test d'alignement COMA++: $O_1 \rightarrow O_2$

IV.3.3.3. Alignement XMAP[59] : $O_1 \rightarrow O_2$

Avant de commencer le test d'alignement nous devons retenir la configuration de XMAP illustrée dans le tableau suivant :

Choix des Matchers	NameMatcher et PropertyConstraintMatcher sont les deux matchers utilisés par l'algorithme ;
Agrégation	Average (paramètre par défaut): Considérer la moyenne des valeurs de similarité obtenues par les matchers (Tous les matchers ont la même importance) ;
Direction	BOTH d'O1 vers O2;
Sélection	Nous allons utiliser le seuil (*Threshold*) en le variant dans chaque test pour trouver des bons résultats ;
Similarité Combinée	Standard (Average), Nous optons pour la configuration par défaut (somme pondérée).

Tableau IV.6. Configuration de XMAP retenue pour l'alignement de $O_1 \rightarrow O_2$[Djeddi et khadir, 2010]

Une fois la configuration est faite nous initialisons les paramètres suivants :

59 Voir l'ANNEXE D

- ω_{la} ($weight\ of\ linguistic\ affinity$)**:** ce paramètre s'il est initialisé à 0.5 alors les deux matchers (NameMatcher et PropertyConstraintMatcher) ont la même importance.
- ***S*(seuil) :** indique le niveau minimum pour que deux entités soient similaires.

Paramètres de Sélection		$N_{expected}$	N_{found}	$N_{Correct}$	Précision	Rappel	Fallout
ω_{la}	S						
0.5	0.1	60	8058	18	0.0022	0.3	0.9977
0.5	0.2		7956	18	0.0023	0.3	0.9973
0.5	0.3		18	18	1	0.3	0.7
0.5	0.4		18	18	1	0.3	0.7
0.5	0.5		18	18	1	0.3	0.7
0.5	0.6		18	18	1	0.3	0.7
0.5	0.7		18	18	1	0.3	0.7
0.5	0.8		18	18	1	0.3	0.7
0.5	0.9		18	18	1	0.3	0.7
0.5	1		18	18	1	0.3	0.7

Tableau IV.7. Résultats du test d'alignement XMAP : $O_1 \rightarrow O_2$

A partir du tableau au dessus, nous remarquons que XMAP se stabilise à partir de $\omega_{la} = 0.5$ et S=0.4 avec un temps d'exécution moins long, et comme la phase d'ancrage de notre application se base sur l'algorithme de XMAP nous allons retenir ce paramétrage et la configuration du tableau IV à la phase d'ancrage dans l'implémentation du BRMAP.

IV.3.3.4. Alignement BRMAP[60] : $O_1 \rightarrow O_2$

Paramètres de Sélection		$N_{expected}$	N_{found}	$N_{Correct}$	Précision P_{BRMAP}	Rappel R_{BRMAP}	Fallout F_{BRMAP}
ω_{la}	S						
>=0.5	>=0.4	60	60	60	1	1	0

Tableau IV.8. Résultats du test d'alignement BRMAP : $O_1 \rightarrow O_2$

IV.3.3.5.Discussion des résultats

A partir de tableau IV.5, tableau IV.7, tableau IV.8, nous remarquons que :

$-P_{BRMAP} >= P_{XMAP} > P_{COMA++}$ (17)

$-R_{BRMAP} > R_{XMAP} >= R_{COMA++}$ (18)

$-F_{BRMAP} < F_{XMAP} < F_{COMA++}$ (19)

60 Voir ANNEXE D

Les résultats expérimentaux (17,18 ,19) illustrés dans la figure IV.6, montrent que BRMAP est plus précis, et donne des résultats plus satisfaisants par rapport aux autres outils de test exploités dans ce chapitre, mais avec un temps d'exécution plus lents.

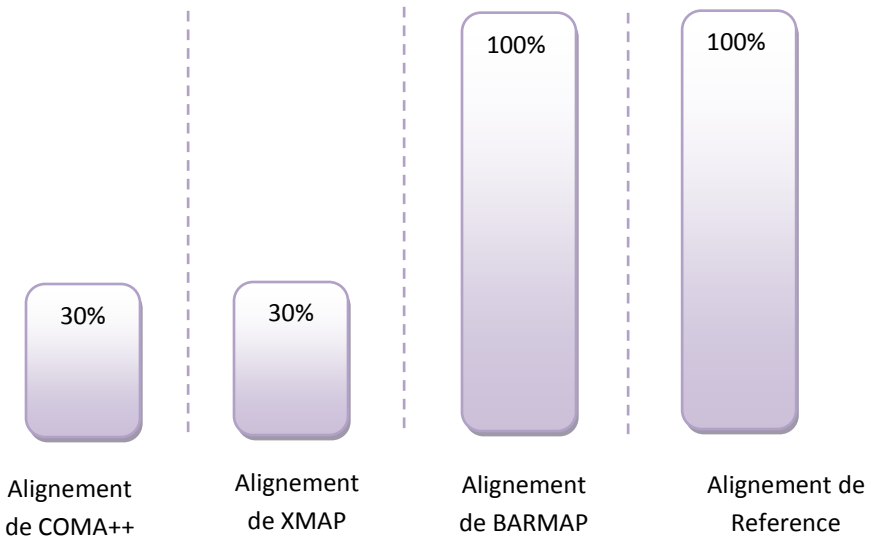

Figure IV.6. Pourcentage des correspondances correctes trouvées par COMA++, XMAP, BRMAP

Comme l'alignement manuel se base sur l'utilisation d'une ontologie de référence du domaine de connaissances des ontologies à aligner, et notre approche consiste à utiliser une ontologie complémentaire et un raisonneur pour automatiser l'alignement des concepts, donc BRMAP est un outil automatisant l'alignement manuel.

Chapitre V

V.1. Conclusion

Dans ce mémoire, nous avons présenté l'importance de l'alignement des ontologies décrites en OWL dans le domaine industriel en étudiant les différentes méthodes, approches et techniques utilisées. Ensuite nous avons implémenté l'idée d'utiliser une ontologie plus détaillée que les deux ontologies à aligner, dites de background ou de support, en proposant l'utilisation d'un raisonneur à la phase de dérivation afin d'automatiser la découverte de correspondance sémantique entre les différents ancres trouvés lors de la phase d'ancrage qui se base sur l'algorithme de XMAP. Ce dernier se base sur l'utilisation des techniques d'alignement classiques et ne donne pas de bons résultats quand les deux ontologies à apparier ont des structures différentes. Cette implémentation a engendré notre plugin BRMAP qui a donné des résultats satisfaisants.

Ce plugin a donné des résultats supérieurs aux autres outils exploités pendant l'étape de test, toutefois avec un temps d'exécution plus long étant donné la capacité volumineuse de l'ontologie complémentaire utilisée.

La disponibilité de cette dernière ontologie est le gros problème de notre proposition, parce que la construction d'une ontologie englobant les différents points de vue des différents constructeurs, est un travail qui s'avère très pénible. Ce travail nécessite la présence de plusieurs experts connaisseurs du domaine et l'ingénieur de connaissances.

V.2. Perspectives

- Les algorithmes proposés ont été implémentés en Java pour évaluer la qualité des correspondances entre deux ontologies. Cependant, ils ne sont pas encore optimisés au niveau de la performance. Il est nécessaire d'étudier des heuristiques particulières dans la structure de l'ontologie en OWL afin de réduire le temps d'alignement.

- Toutes les ontologies traitées sont représentées en OWL et respectant le même sens hiérarchique (sens ascendante ou descendante), une perspective est de tester notre outil sur des ontologies de différentes représentations hiérarchiques en traitant les défaillances trouvées.

- Faire des tests sur des ontologies décrivant d'autres domaines, comme le web sémantique.

Références bibliographiques

[Aleksovski *et al.*, 2006a] Aleksovski, Z., Klein, M., Ten Kate, W., Van Harmelen, F. (2006a). Matching Unstructured Vocabularies using a Background Ontology, Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW'06),Springer-Verlag.

[Aleksovski *et al.*, 2006b] Aleksovski, Z., Klein, M., Ten Kate, W., Van Harmelen, F. (2006b). Exploiting the Structure of Background Knowledge used in Ontology Matching. ISWC'06 Workshop on Ontology Matching (OM-2006), Athens, Georgia, USA.

[AnHai Doan et al., 2003] AnHai Doan, JayantMadhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. ***Learning to match ontologies on the semantic web.*** *The VLDB Journal*, 12(4):303–319, 2003.

[Baader et al 2005] BAADER F., HORROCKS I., SATTLER U., « *Description Logics as Ontology Languagesfor the Semantic Web.* », *Mechanizing Mathematical Reasoning*, 2005, p. 228-248.

[Haarslev et al 2003] HAARSLEV V., MOLLER R., « *Racer : An owl reasoning agent for the semantic web* », 2003.

[Baader et al 2003a], Baader, F. et Nutt, W., 2003. Basic description logics. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, pp. 47100.

[Baader et al 2003b], Baader, F., Horrocks, I. et Sattler, U., 2003. Description logics as ontology languages for the semantic web. Dans Hutter, D. et Stephan, W. (éditeurs), Festschrift in honor of Jörg Siekmann. Lecture Notes in Articial Intelligence. Springer-Verlag.

[Baader et al 2003c], Baader, F., 2003. Appendix 1 : Description logic terminology. Dans Baader, F., Calvanese, D., McGuinness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, pp. 495505.

[Brachman et al 1985] BRACHMAN R. J., SCHMOLZE J. G., « *An Overview of the KL-ONE Knowledge Representation System.* », *Cognitive Science*, vol. 9, no 2, 1985, p. 171-216.

[Benerecetti et al., 2000] Benerecetti M., Bouquet P. and Ghidini C., "*Contextual reasoning distilled*", Journal of Theoretical and Experimental artificial Intelligence, 12(3):279–305, July 2000.

[Berners-Lee, 2003] Berners-Lee, T. WWW past & future. Available at http://www.w3.org/2003/Talks/0922-rsoc-tbl. Slide 30. 2003.

[Bouquet et al., 2003] Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L., et Stuckenschmidt, H. C-OWL: Contextualizing Ontologies. International Semantic Web Conference 2003: 164-179

[Bouquet et al., 2004] Bouquet P., Giunchiglia F., van Harmelen F., Serafini L., Stuckenschmidt H., *"Contextualizing Ontologies", Journal of Web Semantics*, vol. 1, n° 4, p. 1-19, 2004.

[Clark and Parsia, 2009], Clark and Parsia. Pellet : The open source owl dl reasoner. Pellet home page, 2009.

[Corcho,2002] Corcho O, Fernández-López M, Gómez-Pérez A, Vicente O (2002) WebODE: an Integrated Workbench for Ontology Representation, Reasoning and Exchange. In: Gómez-Pérez A, Benjamins VR (eds) 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02). Sigüenza, Spain. (Lecture Notes in Artificial Intelligence LNAI [Domingue,1998] 73) Springer-Verlag, Berlin, Germany, pp. 138–153.

[Cohen et al., 2003] Cohen, W., Ravikumar, P., et Fienberg, S. A comparison of string metrics for matching names and records. Dans Proc. KDD-2003 Workshop on Data Cleaning and Object Consolidation, 2003.

[Corcho, 2004] Corcho O., "*A declarative approach to ontology translation with knowledge preservation***"**, PhD thesis, Universidad Politecnica de Madrid, Madrid (ES), 2004.

[Corcho et Gomez-Perez, 2001] Corcho, O. et Gomez-Perez, A. Solving Integration Problems of E-Commerce Standards and Initiatives through Ontological Mappings. IJCAI-01 Workshop on Ontologies and Information Sharing, pages 131–140, 2001.

[Dagan et al., 1999] Dagan, I., Lee, L., et Pereira, F. Similarity-based models of word cooccurrence probabilities. Machine Learning 34(1-3).

[Damashek, 1995] *Damashek M. Gauging Similarity with n-Grams : Language-Independent Categorization Of Text. Science 1995 ; 267 : 843-848.*

[Dieng et Hug, 1998] Dieng, R. et Hug, S. Comparison of "personal ontologies" represented through conceptual graphs. Dans Proc. 13th ECAI, Brighton (UK), pages 341–345, 1998.

[Djeddi et khadir, 2010] Conception et Réalisation d'un algorithme d'Alignement d'ontologies en OWL, mémoire Présentée Au Département d'informatique Université Badji Mokhtar Annaba, Pour l'obtention du diplôme de MASTER2 en Informatique Option : Sciences et Technologies de l'information et de la Communication 2009.

[Do et al., 2002] Do H., Melnik S., Rahm E., *« Comparison of schema matching evaluations »*, *Proceedings of the 2nd Int. Workshop on Web Databasess*, German Informatics Society, Erfurt, Germany, p. 221-237, 2002.

[Do et Rahm, 2002] Do, H.H. et Rahm, E. Coma – a system for flexible combination of schema matching approaches. Dans Proc. VLDB, pages 610–621, 2002.

[Do et Rahm., 2007] Do H.-H, and Rahm E., *"Matching large schemas: Approaches and evaluation"*, Information Systems, Volume32, Issue 6, p.857-885, September 2007.

[Doan et al., 2000] Doan, A.H., Domingos, P.et Halevy, A. Learning source descriptions for data integration. In: ProcWebDBWorkshop, pp. 81–92, 2000.

[Doan et al., 2002] Doan, A., Madhavan, J., Domingos, P., et Halevy, A. Learning to Map between Ontologies on the Semantic Web. The 11th International World Wide Web Conference (WWW'2002), Hawaii, USA.

[Domingos et Pazzani, 1997] Domingos, P. et Pazzani, M. On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. Machine Learning, 29:103, 1997.

[Donini et al 2003],Donini, F. M., 2003. Complexity of reasoning. Dans Baader, F., Calvanese, D., McGuin ness, D., Nardi, D. et Patel-Schneider, P. (éditeurs), The Description Logic Handbook : Theory, Implementation and Applications. Cambridge University Press, pp. 101141.

[Durban et al., 1998] Durban, R., Eddy, S. R., Krogh, A., et Mitchison, G. Biological sequence analysis - ***Probabilistic models of proteins and nucleic acids***. Cambridge: Cambridge University Press, 1998.

[E. Sirin et al, 2007], E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Journal of Web Semantics*, 5(2), 2007.

[Ehrig et Staab, 2004] Ehrig, M. et Staab, S. QOM - quick ontology mapping. Dans Proc. 3rd ISWC, Hiroshima (JP), November 2004.

[Ehrig et Sure, 2004] Ehrig, M. et Sure, Y. Ontology mapping – an integrated approach. Dans Christoph Bussler, John Davis, Dieter Fensel, and Rudi Studer, editors, Proc. 1st ESWS, Hersounisous (GR), volume 3053 of Lecture Notes in Computer Science, pages 76–91. Springer Verlag, MAY 2004.

[Elizabet et Pezet] Elisabeth Lacombe et Jo Link-Pezet. Le web sémantique.

[Euzenat, 2001] Euzenat J., ***"Towards a principled approach to semantic interoperability"*** ,in Proc. IJCAI Workshop on Ontologies and Information Sharing, pages 19–25, Seattle (WA US), 2001.

[Euzenat et al., 2004] Euzenat, J., Bach, T.L., Barrasa, J., Bouquet, P., Bo, J.D., Dieng-Kuntz, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., Maynard, D., Napoli, A., Stamou, G., Stuckenschmidt, H., Shvaiko, P., Tessaris, S., Acker; S.V. et Zaihrayeu, I. State of the art on ontology alignment, deliverable 2.2.3, IST Knowledge web NoE, Knowledge web NoE, 80p., June 2004.

[Euzenat et Valtchev, 2004] Euzenat, J. et Valtchev, P. Similarity-based ontology alignment in OWL-lite. Dans Proc. 15th ECAI, Valencia (ES), 2004.

[Euzenat et al., 2007] Euzenat J. and al., **"***Heterogeneity in the semantic web, deliverable 2.2"*, Knowledge Web, December 2007.

[Euzenat et shvaiko., 2007] Euzenat, J. and Shvaiko P., **"*Ontology matching*"**, Springer, Heidelberg (DE), 2007.

[Evren et al,2006],Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet : A practical owl-dl reasoner. Submitted for publication to Journal of Web Semantics, 2006.

[Fabien Gandon 2006] Fabien Gandon. Ontologies informatiques. Mai 2006.

[Fellegi et Sunter, 1969] Fellegi, I. P. et Sunter, A. B. A theory for record linkage. Journal of the American Statistical Society 64:1183–1210.

[Fensel et al., 2007] Fensel D. , Lausen H., Polleres A., De Bruijn J., Stollberg M., Roman D., and Domingue J.. *Enabling semantic web services: the web service modeling ontology*, Springer, Heidelberg (DE), 2007.

[Ferret, et al., 2001] Ferret O., Grau B et Jardino M., (2001), A cross-comparaison of two clustering methods, In Proceedings of the ACL Workshop on Evaluation for Langage and Dialogue Systems, Toulouse, France, July : 6-7.

[Ghidini et Giunchiglia., 2004] Ghidini C. and Giunchiglia F., "***A semantics for abstraction*"**, in Proceedings ECAI, 2004.

[Gomez-Perez 2000] Gomez-Perez A., Développement récents en matière de conception, de maintenance et d'utilisation d'ontologies, 3èmes rencontre Terminologie et intelligence artificielle TIA, Vol 19, PP : 9-20.

[Greffenstette, 1995] *Greffenstette G. Comparing Two Language Identification Schemes.* Communications JADT 1995 ; 85-96.

[Gruber, 1993] Gruber, T. R. (1993), A translation approach to portable ontology specifications, Knowledge Acquisition, 5((2): 199 – 220.

[Guarino et Giaretta, 1995] Guarino, N. et Giaretta, P. Ontologies and knowledge bases- towards a terminalogical clarification. Dans N.J. Mars, editor, Towards Very Large Knowledge Bases - Knowledge Building and Knowledge Sharing 1995, pages 25--32. IOS Press, Amsterdam, 1995.

[Giunchiglia et Shvaiko, 2003] Giunchiglia, F. et Shvaiko, P. Semantic matching. Dans Proc. IJCAI 2003 Workshop on ontologies and distributed systems, Acapulco (MX), pages 139–146, 2003.

[Giunchiglia et al., 2004] Giunchiglia, F., Shvaiko, P. et Yatskevich, M. S-Match: an algorithm and an implementation of semantic matching. Dans Proceedings of ESWS 2004, Heraklion (GR), pages 61–75, 2004.

[Giunchiglia et al., 2006] Giunchiglia F., Shvaiko P. and Yatskevich M., ***"Discovering missing background knowledge in ontology matching"***, in Proc. 16th European Conference on Artificial Intelligence (ECAI), pages 382–386, Riva del Garda (IT), 2006.

[Giunchiglia et Yatskevich, 2004] Giunchiglia F. et Yatskevich M. Element Level Semantic Matching. dans Meaning Coordination and Negotiation workshop at ISWC'04. Hiroshima, Japan, 2004.

[Haarslev et Möller, 2001] V. Haarslev and R. Moller. Racer user's guide and reference manual version 1.6. Technical report, University of Hamburg, Computer Science Department,2001.

[Halton et al., 1991] Halton J.P., Bouzid N., Charpillet F., Haton M-C., Laasri H., Marquis P., Mondot T., Napolia A., (1991), Le raisonnement en Intelligence Artificielle, Paris: InterEditions, 1991-480.

[Hameed et al., 2003] Hameed, A., Preece, A., et Sleeman, D. Ontology Reconciliation, Dans Staab, S and Studer, R, Eds. Handbook on Ontologies in Information Systems, pages pp. 231-250, 2003.

[Hameed et al., 2004] Hameed A., Preece A. and Sleeman D., "*Ontology reconciliation*", in Steffen Staab and Rudi Studer, editors, Handbook on ontologies, chapter 12, pages 231–250.Springer Verlag, Berlin (DE), 2004.

[Horrocks, 1998],Horrocks, I., 1998. The FaCT system. Dans de Swart, H. (éditeur), Automated Reasoning with Analytic Tableaux and Related Methods : International Conference Tableaux'98, number 1397 in Lecture Notes in Articial Intelligence. Springer-Verlag, pp. 307312.

[Horrocks et al 2004] HORROCKS I., PATEL-SCHNEIDER P. F., BOLEY H., TABET S., GROSOF B., DEAN M., « *SWRL : A Semantic Web Rule Language Combining OWL and RuleML* », *W3C Member Submission*, , 2004.

[**Horrocks et Sattler 2005**],Horrocks, I. et Sattler, U., 2005. A tableaux decision procedure for SHOIQ. Dans Proc. of the 19th Int. Joint Conf. on Articial Intelligence (IJCAI 2005).

[Jaro, 1989] Jaro, M. A. ***Advances in record-linkage methodology as applied to matching the 1985census of Tampa***, Florida. Journal of the American Statistical Association 84:414–420.

[Jaro, 1995] Jaro, M. A. ***Probabilistic linkage of large public health data files*** (disc: P687-689). Statistics in Medicine 14:491–498.

[Kalfoglou et Schorlemmer 2003] Kalfoglou, Y. et M. Schorlemmer (2003). Ontology mapping : the state of the art. In *Knowl.Eng. Rev., 18(1) :1–31*Reynaud, C., Safar, B. (2006a) When usual structural alignment techniques don't apply.ISWC '06 Workshop on Ontology Matching (OM-2006), Poster, Athens, Georgia, USA.

[Klein, 2001] Klein, M. Combining and relating ontologies: an analysis of problems and solutions. Proc. of the IJCAI-01 Workshop on Ontologies and Information Sharing , Seattle, USA, August 4-5, 2001, p. 53-62.

[Lacombe, 2006] Elisabeth Lacombe et Jo Link-Pezet. ***Le web sémantique.***

[Lovin, 1968] Lovins, J.B. Development of a stemming algorithm. Mechanical Translation and Computational Linguistics, 11: 22-31, 1968.

[Madhavan et al., 2001] Madhavan, J., Bernstein, P. et Rahm, E. Generic schema matching with cupid. Dans Proceedings of the 27th International Conference on Very Large Data Bases, pages 49–58. Morgan Kaufmann Publishers Inc., 2001.

[Masini et al., 1989] Masini G., Napoli, A., Colnet D., Leonard D., Tombre K., (1989), Les langages à objets, Paris : Intereditions : 1989-584.

[MEI et al 2004] MEI J., BONTAS E. P., « *Reasoning Paradigms for OWL Ontologies* », rapport, November 2004, Freie Universitat Berlin.

[Maynard et Ananiadou, 1999] Maynard, D.G. et Ananiadou, S. Term extraction using a similarity-based approach. In Recent Advances in Computational Terminology. John Benjamins, 1999.

[Melnik et al., 2001] Melnik, S., Garcia-Molina, H., et Rahm, E. Similarity Flooding: A Versatile Graph Matching Algorithm. Extended Technical Report, http://dbpubs.stanford.edu/pub/2001- 25.

[Miller, 1995] Miller, A.G. ***Wordnet: A lexical database for english.*** Communications of the ACM, 38(11):39–41, 1995.

[Mindswap, 2003], mindswap. Pallet, 2003.

[Monge et Elkan, 1996] Monge, A., et Elkan, C. The field-matching problem: algorithm and applications. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996.

[Noy et Musen, 2001] Noy, N. et Musen, M. Anchor-PROMPT: Using non-local context for semantic matching. Dans Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US), pages 63–70, 2001.

[NOY N., et al 2000], NOY N., FERGERSON R. & MUSEN M. (2000). The knowledge model of Protégé2000: Combining interoperability and flexibility. In R. D IENG & O.CORBY, Eds., *12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00)*, volume (1937) of *Lecture Notes in Computer Science*, p. 17–32, Juan-les-Pins, France: Springer Verlag.

[Oram, 2001] Oram A. (2001). ***Peer-to-Peer: Harnessing the Power of Disruptive Technologies.*** O'Reilly. Schmidt-Schaub, M., G. Smolka (1991). *Attributive concept descriptions with complements*.
Artificial Intelligence, Vol. 48(1) 1-26.

[OWL, 2004], OWL Web Ontology Langage Overview, 3WC Recommendation 10 February 2004, Disponible en ligne: http://www.w3.org/TR/owl-features/(consulté le 7/22/2006).

[Padadimitriou 1994],Padadimitriou, C. H., 1994. Computational complexity. Addison-Wesley Publishing Company, Massachusetts, É.-U.

[Pinto et Martins, 2001] Pinto, H.S. et Martins, J.P. A Methodology for Ontology Integration. In Proceedings of the First International Conference on Knowledge Capture , ACM Press, 2001.

[Porter, 1980] Porter, M.F. An Algorithm for Suffix Stripping, Program, 14(3): 130-137, 1980.

[Rahm et Bernstein, 2001] Rahm, E. et Bernstein, P. A survey of approaches to automatic schema matching. VLDB Journal, 10(4):334–350, 2001.

[Reynaud et Safar, 2006, 2007] Reynaud, C., Safar, B. (2006b). Structural Techniques for Alignment of Taxonomies: experiments and evaluation, In TR 1453, LRI, Université Paris-Sud, Juin 2006.

[Sabou *et al.*, 2006] Sabou, M., D'Aquin, M., Motta, E. (2006). Using the Semantic Web as Background Knowledge for Ontology Mapping, ISWC'06 Workshop on Ontology Matching (OM-2006), Athens, Georgia, USA.

[Schaub et Smolka] Schmidt-Schaub, M. et Smolka, G., 1991. Attributive concept descriptions with complements. Articial Intelligence 48 (1), 126.

[Shvaiko et Euzenat, 2005] Shvaiko, P., Euzenat, J. A Survey of Schema-based Matching Approaches. Journal on Data Semantics, 2005.

[Sirin et Parsia, 2004], Sirin, E. et Parsia, B., 2004. Pellet : An owl dl reasoner. Dans Haarslev, V. et Möller, R. (éditeurs), Proceedings of the International Workshop on Description Logics (DL2004).

[Stillings et al., 1989] Stillings N.A., Feinstein M.H., Garfield J.L. et al., (1989), Cogntive Science: Cambridge, MA: MIT Press, 533p.

[Stumme et Mädche, 2001] Stumme, G. et Mädche, A. FCA-merge: bottom-up merging of ontologies. Dans Proc. 17th IJCAI, Seattle (WA US), pages 225–230, 2001.

[Tsarkov et Horrocks, 2004], Tsarkov, D. et Horrocks, I., 2004. Ecient reasoning with range and domain constraints. Dans Proc. of the 2004 Description Logic Workshop (DL 2004). pp. 4150.

[Zaihrayeu, 2006] Zaihrayeu I., ***"Towards Peer-to-Peer Information Management Systems"***,PhD thesis, International Doctorate School in Information and Communication Technology, University of Trento, Italy, March 2006.

[Zou et al., 2004], Zou, Y., Finin, T. et Chen, H., 2004. F-owl : an inference engine for semantic web. Dans Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems. Greenbelt, Maryland.

Annexe A

I. Historique

Figure A.1: L'Éolipyle d'Héron d'Alexandrie

Les premiers travaux sur la vapeur d'eau et son utilisation remontent à **l'Antiquité : Héron d'Alexandrie** conçut et construisit au **Modèle: Ier siècle ap. J.-C. l'éolipyle**, qui bien que considéré comme un jouet du fait de sa faible puissance, n'en était pas moins un moteur à vapeur, à **réaction.**

Il fallut attendre le XVII^{e} siècle, pour que l'idée d'utiliser la puissance de la vapeur d'eau réapparaisse. En **1601, Giovanni Battista della Porta**, puis en **1615 Salomon de Caus** décrivent une **pompe** capable de chasser l'eau d'un récipient. Puis en **1629, Giovanni Branea**, suggère l'idée de **moulins,** mus par la vapeur et l'année d'après **David Ramseye** obtient un brevet pour une pompe, mue par un moteur à feu. En **1663, Edward Somerset** améliore le projet de **Caus,** en équipant la chambre à vapeur d'un refroidisseur, il construit un modèle de grande taille, mais il meurt avant d'avoir pu appliquer pratiquement sa création.

En **1698, Thomas Savery** dépose un brevet sur une pompe destinée à **l'exploitation minière**, fonctionnant à la vapeur, directement inspirée des travaux de **Edward Somerset.**

Par la suite, il la perfectionne en collaboration avec **Thomas Newcomen**, grâce entre autres, aux travaux du Français **Denis Papin.** Ce dernier après avoir inventé un prototype **d'autocuiseur**, avait eu l'idée du piston, donnant ainsi accès à des puissances insoupçonnées jusqu'alors. Un premier modèle commercial fut utilisé dès **1712** dans les mines de charbon, près de **Dudley**, dans le centre de **l'Angleterre**. Ces pompes fonctionnaient en produisant un vide dans une chambre fermée où l'on fait se condenser de la vapeur, grâce à un jet d'eau. Les vannes d'admission et d'échappement, d'abord à commande manuelle, sont automatisées par **Henry Beighton**, en **1718**. Ces pompes deviennent rapidement courantes dans toutes les mines humides de **l'Europe**. Elles restent cependant très coûteuses à l'emploi car le cylindre doit être réchauffé avant chaque admission de vapeur.

L'Écossais **James Watt,** après avoir réparé un moteur Newcomen, en **1763**, cherche alors des idées d'amélioration pour en augmenter l'efficacité. Ses réflexions débouchent, en **1765** sur l'idée d'une chambre de condensation pour la vapeur séparée par une valve, idée sur laquelle il dépose un brevet en **1769**. Il commence alors à produire des moteurs améliorés avec le financement de **Matthew Boulton.**

Il continue en parallèle à chercher des idées sur et autour de son invention. En **1781** il met au point le système mécanique permettant de créer un mouvement de rotation à partir du mouvement rectiligne du piston, ce qui lui permet ensuite de concevoir le cylindre à **double action** où la vapeur entraîne le **piston**, lors de sa montée et de sa descente. La puissance de la machine en est fortement augmentée.

Il formalise aussi une utilisation possible en **1784**, en déposant un brevet sur une **locomotive à vapeur,** invente un indicateur de pression de la vapeur dans le **cylindre,** et en **1788,** une valve de puissance sur laquelle il utilise ensuite l'idée de Boulton d'employer un **régulateur centrifuge** pour rendre la puissance produite constante en dépit des variations dans la production de vapeur et les sollicitations de puisance en sortie. Il introduit aussi une nouvelle unité de mesure de la puissance, le **cheval vapeur**.

Certains lui reprochent d'avoir freiné le développement des systèmes à haute pression, fonctionnant par l'expansion de la vapeur, auxquels il ne croyait pas, mais prônés par d'autres inventeurs comme **Jonathan Hornblower,** qui durent attendre l'expiration des **brevets** en **1800,** après leur prolongation en **1782**. Ce dernier a mis au point en **1781**, un double cylindre

combiné où la vapeur passe d'abord dans un cylindre dans lequel elle pousse le piston avant de passer dans un cylindre fonctionnant selon le principe de la condensation qui équivaut à un système à double action. Mais son invention reste expérimentale sans application possible du fait des brevets de Watt, et il faut alors attendre **1803** et **Arthur Woolf** pour la voir émerger enfin. Combiné à un nouveau type de condenseur conçu par **Edmund Cartwright** qui enveloppe le cylindre et l'apparition des **chaudières** produisant de la vapeur à haute pression, cela va permettre la fabrication de machines compactes et puissantes, nécessaires à une utilisation mobile.

La turbine à vapeur est le fruit du travail de nombreux chercheurs et ingénieurs à la fin du XIX^{e} siècle. Parmi les contributions notoires au développement de ce type de turbine, on peut mentionner celles du Britannique Charles Algernon Parsons et celles du Suédois Carl Gustaf Patrik de Laval. Parsons fut à l'origine du principe de la séparation des étages, selon lequel la vapeur se dilate dans un certain nombre d'étages, produisant à chaque fois de l'énergie. De Laval fut le premier à concevoir des jets et des augets adaptés à une utilisation efficace de la vapeur en expansion.

I.1.Principe et fonctionnement

Par l'intermédiaire d'un système de tiroir de distribution, ouvrant et fermant des lumières, la **vapeur** d'eau sous pression est envoyée à une extrémité d'un **cylindre**, où elle pousse un **piston.** Ce dernier entraîne la **bielle** qui est articulée dessus, elle est fixée aussi sur le **volant d'inertie** en un point excentré de son axe de rotation, son mouvement provoque donc une rotation du volant.

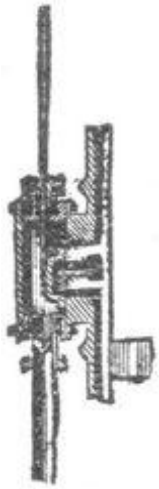

Figure A.2 : Tiroir à vapeur

Du volant repart une biellette commandant le **tiroir d'admission** et **d'échappement**. Quand le piston arrive au bout du cylindre, la biellette repousse le tiroir :

- Dans le cas du cylindre simple effet, le tiroir referme la **lumière** d'entrée de la vapeur et du même côté ouvre une autre lumière pour laisser s'échapper la vapeur contenue dans le cylindre. Le volant, par l'énergie cinétique accumulée, continue de tourner, repoussant ainsi le piston au point de départ.
- Dans un cylindre à double effet, le tiroir ouvre, en plus, une lumière d'admission pour la vapeur de l'autre côté, elle repousse le piston qui continue sa poussée sur le volant.

Sur ce volant on place une **courroie** établissant une liaison élastique avec la poulie d'entrée d'une machine transformant ce mouvement en un **travail** spécifique. Pour être utilisable industriellement, cette énergie doit le plus souvent être régulée afin que la vitesse de rotation ne dépende ni des aléas de la chauffe, ni surtout de la sollicitation de puissance en sortie. C'est là qu'intervient le **régulateur centrifuge** mis au point par **Watt**, qui agit directement sur la **vanne** par laquelle la vapeur arrive de la **chaudière**[35].

II. Architecture

II. 1.Description

II.1.1 Architecture matérielle

35 http://www.inti.be/ecotopie/hydro.html#histori

L'installation comprend donc une chaudière qui produit de la vapeur d'eau sous pression. La taille de l'installation ne permettant pas le fonctionnement d'une turbine, cette vapeur est détendue par deux vannes modulantes. Le condenseur transforme cette vapeur restante grâce à son circuit d'eau froide. Les condensats sont acheminés vers la bâche alimentaire ou elle est pompée par les pompes qui alimentent la chaudière.

Un certains nombre de capteurs et d'actionneurs appareillent l'ensemble pour le commander et le surveiller. Ils sont reliés à un ou plusieurs calculateurs[36].

II.1.2.Combustion externe

Les systèmes à combustion externe permettent d'atteindre de grandes puissances (70 MW) avec des carburants de basse qualité, mais avec une grande consommation (380 kg/MWh) et une faible efficacité thermique. Ils sont aussi très longs à mettre en route (4h) et demandent beaucoup d'espace. Ils sont encore typiquement utilisés pour des applications spécialisées : sur les bâtiments militaires en utilisant l'énergie **nucléaire** pour une meilleure autonomie, et sur les **transporteurs de gaz liquéfié** où il est possible de réutiliser le gaz des soutes.

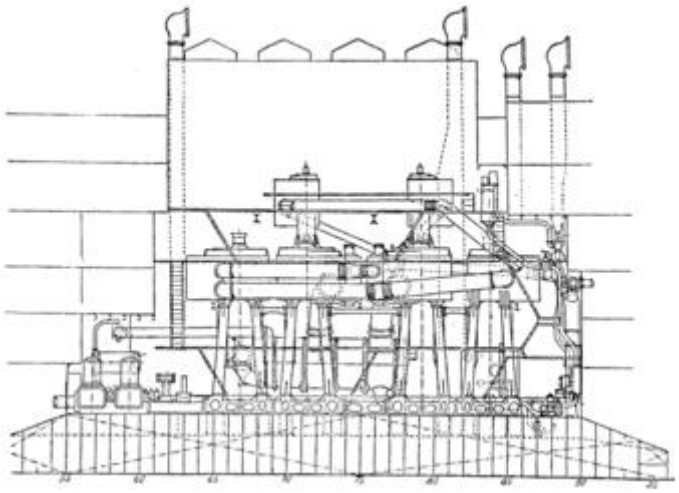

Figure A.3 : Combustion externe de la turbine à vapeur

Schéma de la machine à vapeur du ***Deutschland*** ; les traits horizontaux représentent les ponts et donnent une idée de la taille de l'ensemble.

36 http://www.univ-lille1.fr/lail/gene_vap/fr/liedg.htm

Ce système fut le premier historiquement. Il se caractérise par La chaudière qui Produit de l'énergie en chauffant de l'eau en **vapeur** grâce au combustible ; le **moteur** transforme cette énergie calorique en travail mécanique.

Les chaudières sont constituées d'un corps étanche, parcouru par des tubes de chauffe où circule le liquide à chauffer, qui entoure un **foyer** où a lieu la combustion. Les premiers modèles créés sont assez simples du fait des limitations de la **métallurgie** de l'époque, les tubes de chauffe sont en **cuivre** et les pressions internes très faibles. L'utilisation de l'**acier** permettra par la suite d'augmenter très fortement le rendement en augmentant les pressions en jeu et récupérant la plus grande partie de l'énergie crée, par des techniques très raffinées. À la fin du **XIX^{e} siècle,** le **charbon** commence à céder la place au **mazout** en tant que combustible, ce qui permet de diminuer la main d'œuvre nécessaire au fonctionnemen, en supprimant les hordes de **chauffeurs** qui alimentaient les chaudières à la pelle et de simplifier le stockage du

combustible, maintenant liquide. L'apogée de cette technologie intervient dans les années 1920. Après la Seconde Guerre mondiale, elles furent définitivement supplantées du fait de leur rendement inférieur, mais leur fonction de production de vapeur se retrouva alors dans les réacteurs nucléaires qui présentent l'avantage d'utiliser un carburant quasiment inépuisable ;

Le moteur peut être une machine à vapeur ou une turbine à vapeur. Avec une **machine à vapeur**, la détente de la vapeur est utilisée dans des cylindres, en poussant un piston ; ce mouvement rectiligne était alors transformé en une rotation par l'intermédiare d'une bielle , qui agissait sur un arbre. La métallurgie progressant et la pression des vapeurs produites augmentant, on fit passer alors la vapeur par plusieurs détentes, dans plusieurs cylindres successifs, donnant naissance aux machines à double, puis triple expansion.

À l'aube du XX^{e} siècle émergea la **turbine** qui permit une utilisation plus souple et efficace de la vapeur, en ayant l'avantage de créer directement un mouvement circulaire, supprimant beaucoup de pertes par frottement mécanique et permettant d'atteindre plus facilement des vitesse de rotation supérieures. Les premiers travaux furent réalisés par le britannique Charles Algernon Parsons et déboucha sur le navire expérimental *Turbinia*, qui atteignit 34 nœuds en 1897. Comme la machine à vapeur, on perfectionna de façon très poussée l'utilisation maximale de la vapeur par l'utilisation de groupe de turbines, travaillant

dans des gammes de pressions décroissantes et entraînant le même arbre. L'usage des turbines finit par remplacer totalement les machines à vapeur à piston, entre les deux guerres mondiales[37].

II.2. Les principaux composants des turbines à vapeur

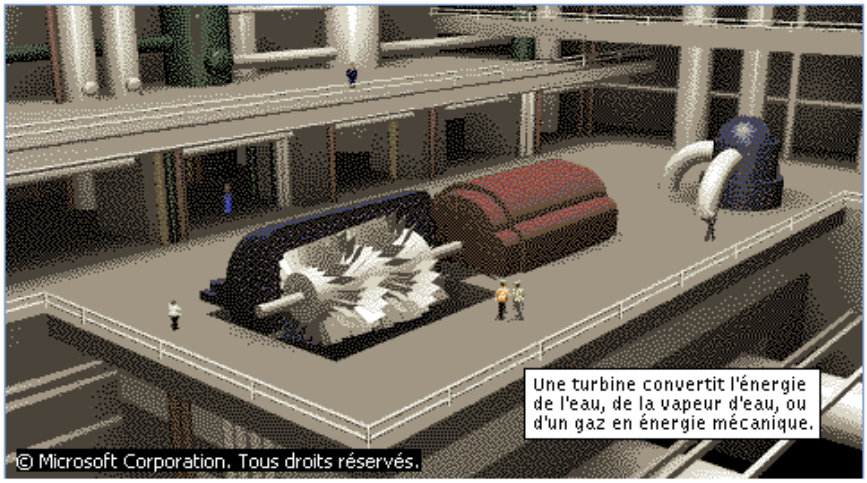

Figure A.4 : la turbine à vapeur[38]

II.2.1.Alternateur

L'alternateur est une machine électrique du type génératrice à courant alternatif qui transforme l'énergie mécanique en énergie électrique. Il est entraîné par la turbine[39].

II.2.2.Transformateurs

- **Transformateur principal (TP) :**

L'évacuation de l'énergie produite par l'alternateur est évacuée sur le réseau haute tension à travers un transformateur principal élévateur : 13800V/63000V, un disjoncteur 63 KV (disjoncteur 52), trois câbles souterrains à pression d'huile et une ligne triphasée aérienne.

- **Transformateur de soutirage (TS) :**

37 http://www.univ-lille1.fr/lail/gene_vap/fr/liedg.htm

38 http://www.metalock.co.uk/gallery.asp

39 http://www.supinfo-projects.com/fr/2004/nuclear_energy_production2electricity/4/

Les auxiliaires du groupe sont alimentés à travers un transformateur de soutirage (TS) abaisseur : 13800V/6300V en service normal et un transformateur de démarrage (TD) abaisseur : 63000V/ 6300V en secours[39]**.**

II.2.3.Chaudière

Le rôle du générateur de vapeur est d'extraire l'énergie calorifique du combustible pour la céder à l'eau et produire de la vapeur à des paramètres fixés. Il constitue la source chaude du cycle thermodynamique. Cette vapeur sera utilisée par la turbine pour fournir de l'énergie mécanique[39].

II.2.4. Condenseur

Afin de maximiser le rendement de la turbine à vapeur, la pression et la température de la sortie de vapeur doivent être aussi basses que possible. Pour cela, la vapeur qui sort de la turbine est dirigée vers le condenseur où elle est refroidie et condensée. Le condenseur est un échangeur de chaleur avec des milliers de tubes dans lesquels l'eau du circuit de refroidissement circule. La vapeur circule sur les tubes et se condense au contact de ceux-ci. L'eau du circuit de refroidissement extrait alors la chaleur de la vapeur[40].

II.2.4.Pompe alimentaire

La pompe KSB à très haute pression est une pompe a centrifuge multicellulaire. Elle comprend un corps d'aspiration, un corps de refoulement et un certain nombre d'étages ou de cellules assemblées par des tirants.

L'eau, provenant de la bâche alimentaire à la pompe, possède une énergie de pression et une énergie cinétique qui seront augmentées dans les turbines en mouvement pour alimenter le générateur de vapeur (chaudière) en quantité nécessaires d'eau pour maintenir le niveau normal[40].

III. Fonctionnement

Bien que les turbines à vapeur soient construites selon deux principes différents (à action ou à réaction, leurs éléments essentiels sont similaires. Elles se composent de tuyères ou de jets, et d'ailettes. La vapeur s'écoule dans les tuyères, dans les quelles elle se dilate. Ainsi, sa

40 http://www.metalock.co.uk/gallery.asp

température diminue et son énergie cinétique augmente. La vapeur en mouvement exerce une pression contre les ailettes, entraînant leur rotation. La disposition des jets et des ailettes, fixes ou stationnaires, dépend du type de turbine. À la sortie du dernier condenseur (échangeur thermique), l'eau peut être de nouveau vaporisée et surchauffée. L'eau ou la vapeur en sortie est alors ramenée vers la chaudière et la pompe « alimentaire », qui compresse de l'eau à l'état liquide. Il s'agit d'une turbine auxiliaire intégrée au cycle thermodynamique de la turbine principale utilisant de la vapeur soutirée dans celle-ci. Les turbines à vapeur possèdent toutefois un équipement annexe, nécessaire à leur fonctionnement. Parmi celui-ci, un palier de tourillon supporte l'arbre et un palier de butée le positionne de manière axiale. Un système d'huile assure le graissage des paliers ; des joints réduisent les pertes de vapeur tout au long de son trajet. Enfin, un système d'étanchéité empêche la vapeur de s'échapper à l'extérieur de la turbine et l'air d'y entrer. La vitesse de rotation est commandée par des soupapes situées aux entrées d'admission de la machine et pilotées par des systèmes de régulation électroniques ou mécaniques. Les turbines à réaction développent une poussée axiale considérable, du fait de la chute de pression sur les ailettes mobiles. Cette poussée est généralement compensée par l'utilisation d'un piston d'équilibrage.

La turbine à vapeur utilise des principes thermodynamiques. Lorsque la vapeur se dilate, sa température et donc son énergie interne diminuent. Cette réduction de l'énergie interne s'accompagne d'une augmentation de l'énergie cinétique sous la forme d'une accélération des particules de vapeur. Cette transformation rend une grande partie de l'énergie disponible. Ainsi, une réduction de 100 kJ de l'énergie interne, du fait de la dilatation, peut provoquer un accroissement de la vitesse des particules de vapeur de l'ordre de 2 800 km/h. À de telles vitesses, l'énergie disponible est importante. Lorsque la pression de la vapeur d'eau en sortie de la turbine est égale à la pression atmosphérique, la turbine est dite à condensation.

Aujourd'hui, les turbines à vapeur sont généralement limitées à une température maximale de 580 °C dans le premier étage, et à une pression maximale d'admission de 170 à 180 bars[41].

III.1. Types de turbines à vapeur

41 http://www.metalock.co.uk/gallery.asp

III.1.1. Turbine à action

La forme la plus simple de turbine à vapeur est la turbine à action, dans laquelle les jets sont fixés sur la partie intérieure de l'enveloppe de la turbine, et les ailettes placées sur le bord des roues tournantes montées sur un arbre central. La vapeur se déplaçant dans une tuyère fixe passe sur les ailettes incurvées, qui absorbent une partie de l'énergie cinétique de la vapeur dilatée, faisant ainsi tourner la roue et l'arbre sur lesquels elles sont montées. Cette turbine estconçue de manière à ce que la vapeur entrant par une extrémité de la turbine se dilate à travers une succession de tuyères jusqu'à ce qu'elle ait perdu la majeure partie de son énergie interne.

III.1.2. Turbine à réaction

Dans la turbine à réaction, une partie de l'énergie mécanique est obtenue par l'impact de la vapeur sur les ailettes. La partie la plus importante est obtenue par l'accélération de la vapeur lors de son passage dans la roue de la turbine, où elle se dilate. Une turbine de ce type se compose de deux jeux d'ailettes, l'un fixe, l'autre mobile. Ces ailettes sont disposées de telle façon que chaque paire joue le rôle de tuyère, à travers laquelle la vapeur se dilate lors de son passage. Dans chaque étage, une faible quantité d'énergie thermique est convertie en énergie cinétique. La vapeur se détend dans les aubes fixes, puis entraîne les aubes mobiles disposées sur la roue ou le tambour de la turbine. Les ailettes d'une turbine à réaction sont en général montées sur un tambour, qui fait alors office d'arbre.

Les turbines à réaction nécessitent en général davantage d'étages que les turbines à action. Il a pu être démontré que, pour le même diamètre et la même gamme énergétique, une turbine à réaction a besoin de deux fois plus d'étages pour obtenir un rendement maximal. Les grosses turbines, qui sont généralement à action, utilisent une certaine réaction à la base du trajet de vapeur pour assurer un débit efficace à travers les auges. Nombre de turbines, qui sont normalement à réaction, disposent d'un premier étage de commande d'impulsion, qui permet d'envisager la réduction du nombre total d'étages nécessaires. Les arbres des turbines de chaque étage sont reliés entre eux au moyen d'accouplements[42].

42 http://fr.encarta.msn.com/encyclopedia_761563866_3/turbine.html

III.2. La turbine à vapeur et ses variantes

La turbine à vapeur reste la machine la plus utilisée pour la cogénération à biomasse. Une installation de production d'électricité classique comporte, outre l'alternateur, au moins quatre éléments : une chaudière qui fournit de la vapeur à haute pression (quelques dizaines de bars) ; une turbine qui détend la vapeur et dont l'arbre produit le travail moteur ; un condenseur qui permet, grâce à une source froide (l'eau de retour d'un réseau de chauffage urbain, par exemple), de condenser totalement la vapeur ; une pompe qui redonne au fluide la pression qu'il avait à l'amont de la turbine. Le rendement électrique d'une telle installation est le produit de cinq rendements successifs fonction de la qualité de la combustion (rendement chaudière), des niveaux enthalpiques de la vapeur à l'entrée et à la sortie de la turbine (rendement électrique du cycle théorique ou encore rendement thermique théorique), de l'importance de la détente en zone diphasique par rapport à la détente en zone surchauffée (rendement isentropique), de la qualité de la turbine (rendement mécanique) et de la qualité de l'alternateur (rendement alternateur)[43].

III.2.1. Turbines à contrepression

Avec une turbine à contrepression, la vapeur reste strictement en phase gazeuse, ce qui signifie que sa pression de sortie est au minimum de quelques bars (3 à 5 bars), mais qu'elle peut être supérieure selon les besoins aval. L'intérêt de ce type de turbine est en effet de délivrer de la vapeur à un niveau enthalpique suffisant pour qu'elle soit utilisable.
L'inconvénient réside dans la faiblesse du rendement électrique du cycle théorique, si bien qu'avec une pression de sortie de 3 bars, par exemple, il est difficile d'obtenir un rendement électrique global supérieur à 15-18% selon la pression d'admission. En revanche, le niveau d'investissement est inférieur à celui d'une turbine à condensation. L'utilisation type des chaudières à contrepression est une centrale qui doit de toute façon produire de la chaleur (pour un process industriel ou un réseau de chaleur par exemple) et pour laquelle on peut justifier la production d'électricité, même à faible rendement, par le seul surcoût d'une surchauffe de la vapeur[10].

III.2.2. Turbines à condensation

43 http://www.energie-plus.com/news/fullstory.php/aid/1454.html

Dans une turbine à condensation, la pression de sortie de la vapeur peut descendre jusqu'à 40mbars environ, ce qui fait apparaître des condensats dans la turbine, qu'il faut évacuer par des purgeurs. Même si le rendement isentropique est moins bon, le rendement électrique global de la centrale est nettement amélioré par rapport au cas précédent, puisqu'il peut atteindre 25 à 30 %. Le problème, c'est que la température de sortie est basse : 30°C à 40mbars, 46°C à 100mbars, 80°C à 500mbars...Pour pouvoir récupérer l'énergie thermique résiduelle, qui représente environ 60% de l'énergie primaire fournie à la chaudière, il faut donc disposer d'une source froide en quantité suffisante (chauffage basse température, préchauffage d'ECS, préchauffage d'eau pour vapeur de process...) Sinon, le rendement énergétique global de l'instalation est très médiocre. L'utilisation type d'une telle turbine simple est une centrale de production d'électricité sans valorisation de chaleur, ou avec une valorisation marginale[43].

III.2.3. Combinaison et soutirage

Deux solutions sont envisageables pour tirer un meilleur parti des turbines. La 1ère consiste à combiner les 2 types : une turbine à contrepression suivie d'une turbine à condensation. Ce montage est particulièrement approprié dans le cas d'un process industriel à fonctionnement intermittent. A la sortie de la turbine à contrepression, la vapeur peut être envoyée, selon les besoins, vers un condenseur pour assurer des besoins de chauffabe, vers le process quand il fonctionne, ou encore vers la turbine à condensation. Cette combinaison permet de gagner quelques points de rendement électrique, tout en utilisant de la vapeur. Il est possible d'aller plus loin en intercalant entre les deux turbines une surchauffeur qui va remonter la température de la vapeur à sa valeur initiale à la pression intermédiaire. L'intérêt de ce cycle avec resurchauffe est que l'énergie thermique fournie lors de la resurchaufe se retrouve, au rendement de la seconde turbine près, entièrement sous forme d'électricité. Le rendement électrique global peut ainsi être amélioré de 5 à 8 % et peut atteindre, voire dépasser 30 %, mais les surcoûts sont importants et cette solution est réservée à des installations de forte puissance.

Dans la 2e solution, on extrait une partie de la vapeur entre 2 étages de la turbine, quel que soit son type, avant la détente complète. Dans le cas le plus simple, le soutirage est effectué pour permettre un usage process ou haute température de la vapeur et le mode de fonctionnement se rapproche de la combinaison précédente, à la différence près que les

parties contrepression et condensation sont réunies dans une même turbine et que plusieurs pressions d'utilisation sont possibles.

On peut également soutirer de la vapeur pour préchauffer l'eau d'alimentation de la chaudière. Cette opération conduit à réduire la production d'électricité mais à en augmenter le rendement. Plusieurs soutirages et réchauffages successifs sont réalisables, mais avec des gains de rendement électrique décroissants. Il est ainsi possible d'augmenter le rendement électrique de 5%, cependant le procédé est ici aussi très coûteux (pompes et échangeurs mélangeurs supplémentaires) ce qui le destine plutôt aux installations de grosse puissance (plusieurs dizaines, voire centaines de MW)[44].

IV. Caractéristiques des turbines à vapeur

IV.1. Taille des composants

Étant donné l'augmentation de volume liée à la dilatation de la vapeur dans les différents étages d'une turbine, la taille des ouvertures à travers lesquelles passe la vapeur doit s'accroître d'un étage à l'autre. Dans la conception pratique des turbines, cet accroissement est réalisé en allongeant les ailettes d'un étage à l'autre, en augmentant le diamètre du tambour ou de la roue sur lesquels sont montées les ailettes, et en ajoutant deux ou plusieurs sections de turbine en parallèle. Par conséquent, une petite turbine industrielle peut avoir une forme plus ou moins conique, avec son plus petit diamètre côté haute pression, ou admission, et son diamètre le plus large côté basse pression, ou échappement. Une grosse turbine destinée à une centrale nucléaire peut avoir quatre rotors se composant d'une section à haute pression à double flux, suivie de trois sections à basse pression à double flux[45].

IV.2 Étages spécifiques

Les turbines à action utilisent généralement un étage de pression appelé turbine Rateau (du nom de l'ingénieur français Auguste Rateau), dans lequel le taux de compression à chaque étage est pratiquement uniforme. Les anciennes turbines à action utilisaient un étage

44 http://www.energie-plus.com/news/fullstory.php/aid/1454.html

45 http://fr.encarta.msn.com/encyclopedia_761563866_4/turbine.html

de vitesse de Curtis, mis au point par l'Américain Charles Gordon Curtis. Cet étage comporte deux jeux d'auges mobiles, avec un jeu intermédiaire d'ailettes fixes à la suite des tuyères.

La séparation d'étages d'une turbine à réaction est parfois appelée séparation de Parsons, du nom de son inventeur, le Britannique Charles Parsons.

Une turbine à réaction comporte souvent un premier étage à action qui permet le réglage du système ; une turbine à action possède en général dans ses derniers étages un degré de réaction voisin de 50 %[46]. **[**

IV.3. Rendement

L'efficacité de l'expansion dans une turbine à vapeur moderne est élevée en raison de l'état de développement des composants du trajet de la vapeur, et de la capacité à récupérer les pertes d'un étage dans les étages en aval, par réchauffement. Le rendement avec lequel une section de la turbine convertit l'énergie thermodynamique disponible en travail mécanique dépasse généralement 90 %. Le rendement thermodynamique d'une installation thermique est en fait bien inférieur, en raison de l'énergie perdue dans la vapeur d'échappement de la turbine[47].

IV.4 Applications

Les turbines à vapeur sont notamment utilisées dans la production d'électricité à partir d'énergie thermique ou pour la propulsion des bateaux. Dans les systèmes de cogénération c'est-à-dire utilisant à la fois la chaleur de traitement (celle utilisée lors d'un processus industriel) et l'électricité, la vapeur est portée à haute pression dans une chaudière, puis extraite de la turbine à la pression et à la température exigées par ce procédé. Dans ce cas, la turbine est dite à contre pression. Les turbines à vapeur peuvent être utilisées en cycles combinés avec un générateur de vapeur qui récupère la chaleur. Les unités industrielles sont utilisées pour entraîner des machines, des pompes, des compresseurs et des générateurs. Leur puissance nominale va de quelques centaines de Watts à plus de 1 300 MW.

46 http://fr.encarta.msn.com/encyclopedia_761563866_4/turbine.html

47 http://fr.encarta.msn.com/encyclopedia_761563866_4/turbine.html

La turbine à vapeur est parfois associée à une turbine à gaz. Le rendement de la turbine à gaz étant faible, elle est généralement utilisée pour la production d'énergie de pointe, les calories des gaz d'échappement de la turbine à gaz servant à faire fonctionner la chaudière de la
turbine à vapeur[48]. **[8]**

V. La maintenance

Bien que le mot **Maintenance** ne soit apparu dans le vocabulaire industriel que dans les années 1950, les concepts de maintenance, tels que nous les connaissons aujourd'hui, remontent en fait à la plus haute antiquité. En effet, la maintenance, qui date de l'époque où on a commencé à fabriquer des objets, est certainement l'un des « plus vieux métiers du monde ». Citons simplement l'ouvrage **D'acquae ductu urbis Romae** écrit par Sextus Julius Frontinus en 99 après Jésus-Christ. Ce rapport au Sénat romain peut être considéré à juste titre comme le livre fondateur de la maintenance moderne. L'auteur, curateur des eaux de la ville de Rome sous le règne de l'Empereur Nerva Auguste et chargé, à ce titre, de l'exploitation et de l'entretien du réseau d'aqueducs et de « machines » qui alimentaient en eau la ville de Rome, découvrait les concepts de la maintenance moderne : entretien préventif, programmation des grands arrêts, planification des tâches, documentation technique et historique, suivi des coûts d'entretien, standardisation des pièces de rechange, implication des exploitants dans la maintenance de leurs installations, relations contractuelles etc. Il a fallu cependant attendre la fin du XX^{e} siècle, l'apparition de l'outil informatique, l'émergence de méthodes et outils tels que l'AMDEC, la fiabilité expérimentale, la MBF, le coût global de cycle de vie, la sûreté de fonctionnement etc. pour voir réellement du nouveau, mais ce « nouveau » va très vite et la maintenance bouge de plus en plus rapidement.

Le nouvel ordre industriel qui se met progressivement en place depuis une dizaine d'années au niveau mondial bouleverse l'environnement de la production et modifie les comportements et les mentalités des hommes qui y travaillent. Nous sommes d'ores et déjà entrés dans une nouvelle ère industrielle qui se caractérise par des mutations technologiques

48 http://fr.encarta.msn.com/encyclopedia_761563866_4/turbine.html

certes, mais également par des mutations organisationnelles et sociales dont les enjeux sont très forts et qui remettent fondamentalement en cause les certitudes les mieux établies.

Aujourd'hui, la problématique de l'entreprise est d'être capable de fournir à son « marché », dans les meilleures conditions de coût et les délais les plus courts, les produits de qualité que recherchent des clientèles de plus en plus exigeantes. C'est pourquoi, composante majeure de l'utilisation optimale des moyens de production, la maintenance est passée progressivement de l'état de service improductif, cher et subalterne, au statut de fonction essentielle voire stratégique du système productif. Elle implique alors une parfaite maîtrise du processus de production. Elle fait appel à une **approche systémique** qui lui est propre et à des **méthodes et outils de plus en plus performants**. Élément stratégique du système productif, elle concerne tout un chacun dans l'entreprise et n'est plus la chasse gardée de quelques spécialistes.

V.1Réparation et maintenance des turbines à vapeur

Le fonctionnement efficace des turbines à vapeur est important pour les industries mondiales, mais comme toutes les machines, ces équipements doivent être continuellement inspectés et entretenus pour produire les meilleures performances. Nous offrons une maintenance sur place pour les garder en bon état de marche et réduire au minimum les risques de panne[49].

V.2.Maintenance des turbines à vapeur de centrales électriques

Les installations de turbine à vapeur produisent la plus grande partie de l'électricité requise par les industries mondiales. Par conséquent, les pannes ne sont pas simplement coûteuses, elles peuvent causer d'énormes perturbations. Nous aidons les ingénieurs de centrale et les constructeurs à réparer et modifier leurs turbines à vapeur pendant les arrêts planifiés ou en cas d'urgence. Nous pouvons effectuer la réparation des turbines à vapeur sur place et les travaux suivants sont typiques de ce que nous offrons :

- L'usinage orbital des paliers de rotor de turbine
- L'alésage et le fraisage de corps de turbines, de pompes et de réducteurs

49 http://www.metalock.co.uk/turbines_a_vapeur.html

- Perçage, taraudage et brochage des composants de turbine[15].

V.3.Programme d'entretien des turbines à vapeur des industries pétrolières

Un programme d'entretien des turbines à vapeur est essentiel pour l'on veut maintenir la profitabilité et la sécurité tout en protégeant l'environnement. Une inspection continue et un entretien régulier sont deux facteurs essentiels à l'obtention de procédés de production qui ne nuisent pas à l'environnement[49].

VI .Les avantages et inconvénients de la turbine à vapeur

VI.1. Avantages

- La production d'énergie est relativement indépendante des conditions météorologiques, la source d'énergie peut être (dans une certaine mesure) facilement stockée et la densité de puissance est très élevée.
- Elles permettent de faire de la **cogénération** : lorsque l'on a besoin à un endroit déterminé (agglomération, industries chimiques, serres, ...) de chaleur en grande quantité, il est intéressant de créer une centrale thermique qui produit de l'électricité et dont le circuit de refroidissement sert de source de chaleur pour l'application désirée. (les centrales solaires, hydraulique et l'éolien le permettent aussi, mais quand le soleil, l'eau ou le vent sont présents)
- C'est une manière de rentabiliser les inévitables pertes de ce type de centrales. La Co- ou tri-génération ne sont cependant pas encore systématiques

VI.2.Inconvénients

- Les sources d'énergie fossiles ont le défaut d'être épuisables et polluantes, induisant de plus une dépendance à l'égard des producteurs de ressources
- Le caractère très centralisé des centrales, et la dépendance au réseau électrique THT les rendent vulnérable
- Les centrales thermiques à flamme produisent du dioxyde et monoxyde de carbone, des oxydes de l'azote et de la vapeur d'eau.

Annexe B

I. Intégration de BRMAP dans Protege2000

❖ ***Exigences***

Les instructions dans le Tableau 22, doivent exister pour utiliser BRMAP - Protégé Plugin.

Système d'exploitation	Le logiciel à été testée sur Microsoft Windows 2000/XP/Vista, Linux.
Java	Plateforme Java 1.5/1.6 /1.7
Protégé	A partir de la version 3.4.4 du Protégé (disponible à http://protege.stanford.edu).

Tableau B.1. Des exigences pour l'intégration de BRMAP dans Protege2000

❖ ***Installation***

Pour pouvoir installer BRMAP dans protégé 2000, on doit suivre les étapes ci-dessous :

1. Télécharger le plugin BRMAP ;
2. Extrayez les dossiers dans un répertoire temporaire de votre choix ;
3. Déplacez le répertoire BRMAP dans le répertoire des plugins de votre logiciel Protégé (e.g C:\Program Files\Protege_3.4\plugins).

❖ ***Configuration du classpath (utilisateurs Windows)***

Modifier le classpath de protégé en suivant les étapes suivantes :

1. Localisez et ouvrez le fichier Protege.lax en suivant le chemin d'installation de votre logiciel protégé (e.g C:\Program Files\Protege_3.4\ Protege.lax) ;
2. Trouvez la section LAX.CLASS.PATH ;
3. Ajoutez, à la fin de la chaine de caractères, la ligne suivante:

```
plugins/BRMAP/lib/jaxrpc-api.jar;plugins/ BRMAP /lib/jaxrpc-impl.jar;
plugins/ BRMAP /lib/activation.jar;plugins/ BRMAP /lib/FastInfoset.jar;
plugins/ BRMAP /lib/jaxp-api.jar;plugins/ BRMAP /lib/jax-qname.jar;
plugins/ BRMAP /lib/jaxrpc-spi.jar;plugins/ BRMAP /lib/jsr173.jar;
plugins/ BRMAP /lib/mail.jar;plugins/ BRMAP /lib/relaxngDatatype.jar;
plugins/ BRMAP /lib/saaj-api.jar;plugins/ BRMAP /lib/saaj-impl.jar;
plugins/ BRMAP /lib/xsdlib.jar;plugins/ BRMAP /lib/log4j-1.2.13.jar;
plugins/ BRMAP /lib/resolver.jar;plugins/ BRMAP /lib/xercesImpl.jar;
plugins/ BRMAP /lib/xercesSamples.jar;plugins/ BRMAP /lib/xml-apis.jar
```

II. Utiliser BRMAP - Protégé Plugin.

Une fois l'outil du Protégé est démarrée vous avez besoin de:

1) Ouvrir un Project OWL ;
2) Activer le Plugin BRMAP en ouvrirons le menu Project → Configure dans Protégé.

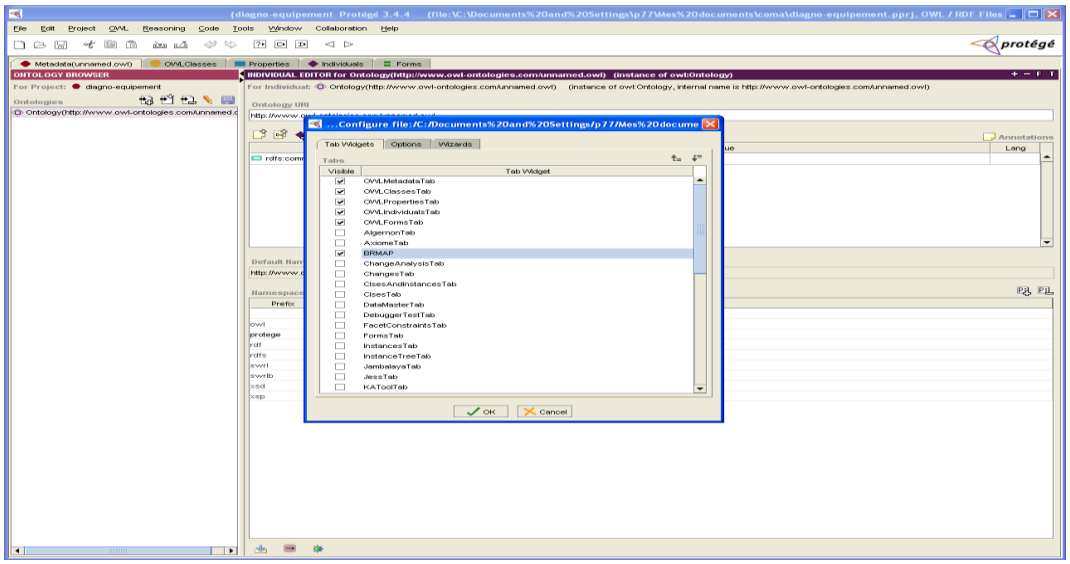

Figure B.1. Activation de BRMAP

Une fois ces étapes achevées, on peut visualiser la console BRMAP dans protégé2000 illustré dans la figure B.2 .

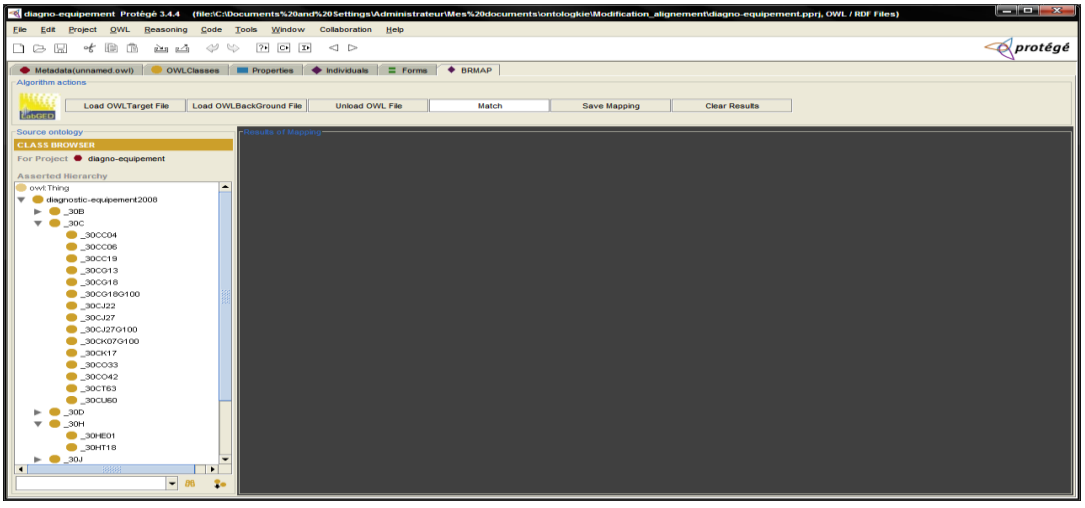

Figure B.2. La console BRMAP dans Protégé

3) En fait, pour exécuter l'application, il suffisait de charger les fichiers OWL des deux ontologies cible et complémentaire en appuyant sur les deux boutons « Load OWL Target File » et « Load OWLBackground File ».

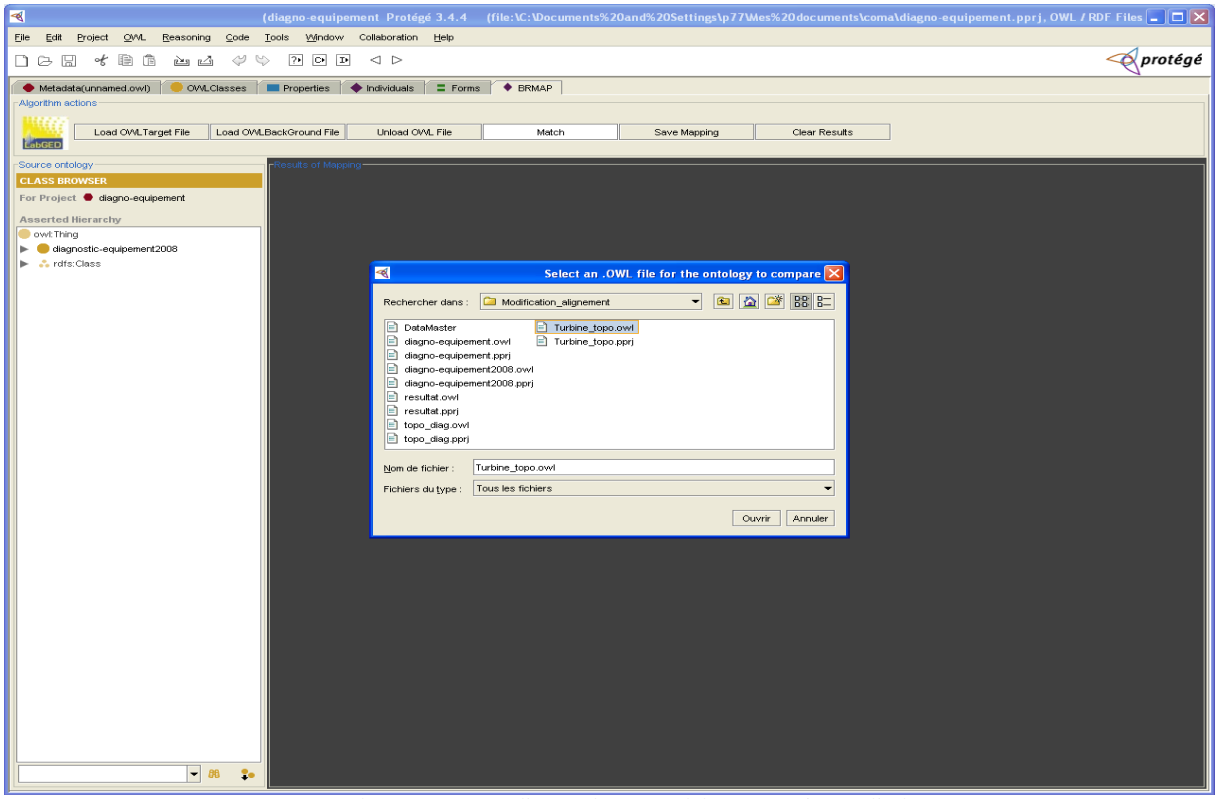

Figure B.3. Chargement d'ontologie cible pour faire l'alignement

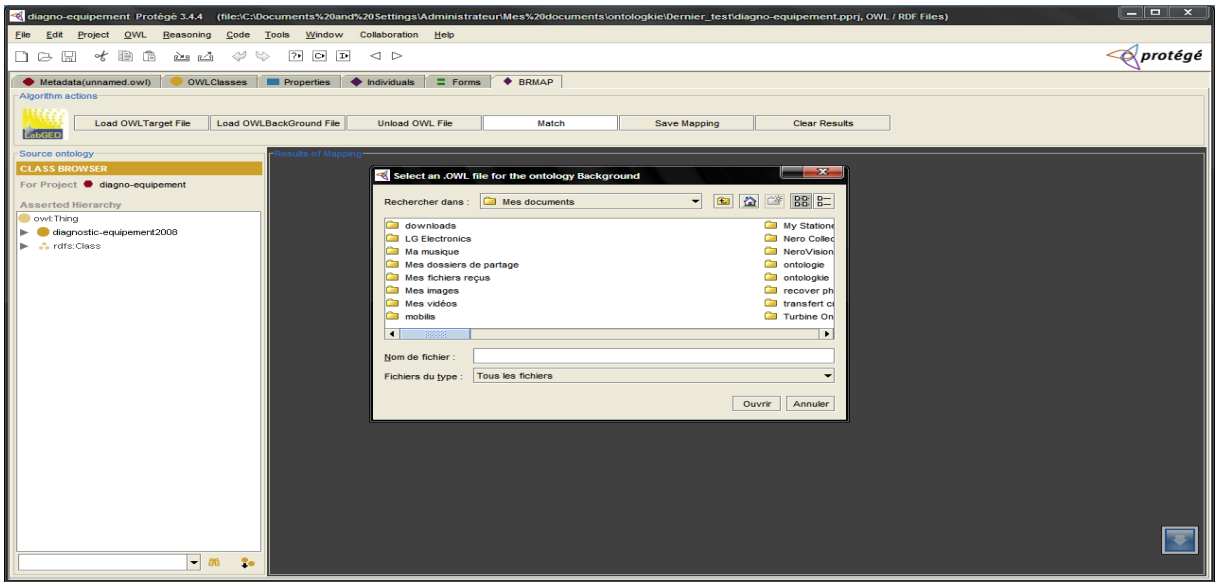

Figure B.4. Chargement d'ontologie complémentaire pour faire l'alignement

4) Puis en appuyant sur le bouton « Match » une boite dialogue s'affiche, contenant les différents paramètres à choisir tel que le Threshold (seuil) et la stratégie d'alignement.

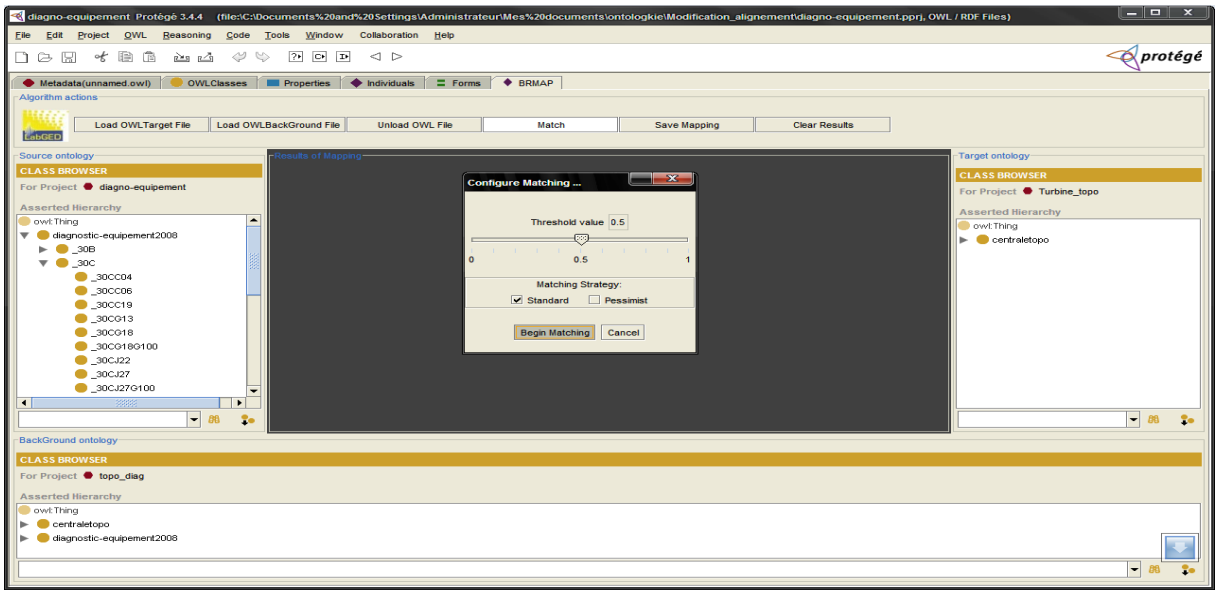

Figure B.5. Configuration des paramètres d'alignement

5) Enfin en appuyant sur le bouton « Begin Matching » le processus d'alignement se déclenche et s'affiche sous forme de tableau, dont chaque ligne représente une paire d'entité de deux ontologies avec le coefficient de correspondance estimé, et un autre contenant la liste des concepts source et cible découvert par l'alignement complémentaire.

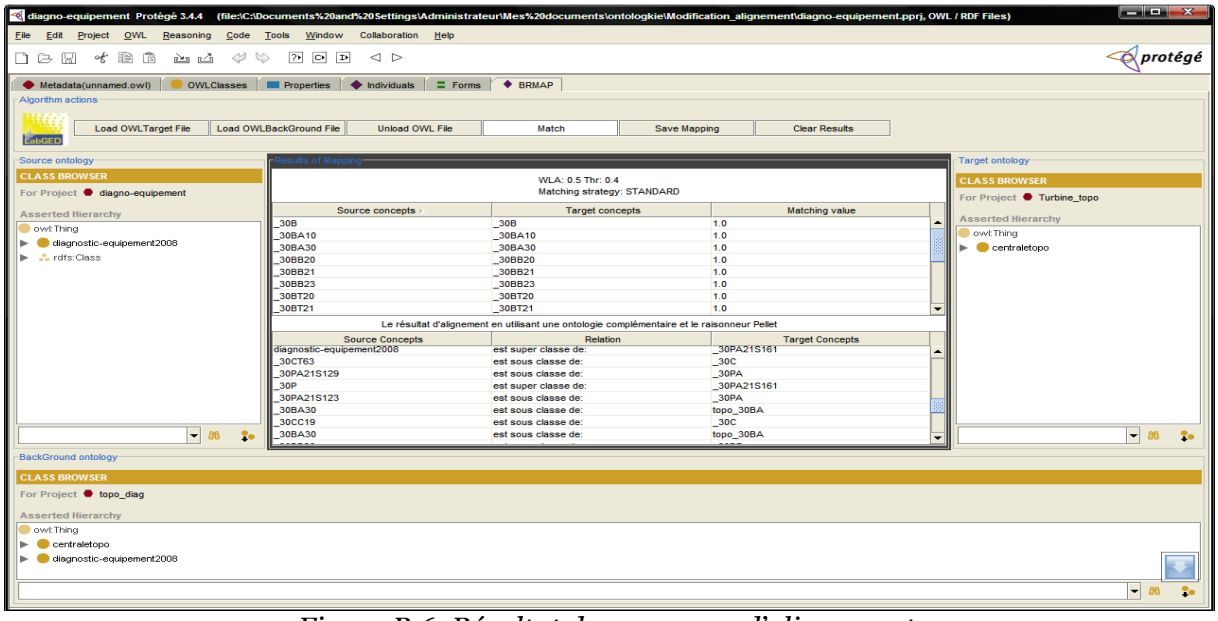

Figure B.6. Résultat du processus d'alignement

Annexe C

C.1. Portion de l'ontologie source

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
  xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://www.owl-ontologies.com/unnamed.owl#"
 xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="_30UA12">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >30UA12</rdfs:label>
  <rdfs:subClassOf>
   <Table_Metaclass rdf:ID="_30U">
    <owl:disjointWith>
     <Table_Metaclass rdf:ID="_30D">
      <rdfs:subClassOf>
       <owl:Class rdf:ID="diagnostic-equipement2008"/>
      </rdfs:subClassOf>
      <owl:disjointWith>
       <Table_Metaclass rdf:ID="_30V">
        <owl:disjointWith>
         <Table_Metaclass rdf:ID="_30P">
          <owl:disjointWith rdf:resource="#_30D"/>
          <owl:disjointWith rdf:resource="#_30U"/>
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >30P</rdfs:label>
          <owl:disjointWith>
           <Table_Metaclass rdf:ID="_30R">
            <owl:disjointWith rdf:resource="#_30D"/>
            <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
            <owl:disjointWith>
             <Table_Metaclass rdf:ID="_30B">
              <owl:disjointWith rdf:resource="#_30R"/>
              <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >30B</rdfs:label>
              <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
              <owl:disjointWith>
               <Table_Metaclass rdf:ID="_30H">
                <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >30H</rdfs:label>
                <owl:disjointWith>
                 <Table_Metaclass rdf:ID="_30C">
                  <owl:disjointWith rdf:resource="#_30R"/>
                  <owl:disjointWith rdf:resource="#_30U"/>
                  <owl:disjointWith rdf:resource="#_30D"/>
                  <rdfs:label rdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#string"
                  >30C</rdfs:label>
                  <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
                  <owl:disjointWith rdf:resource="#_30V"/>
                  <owl:disjointWith>
                   <Table_Metaclass rdf:ID="_30J">
                    <owl:disjointWith rdf:resource="#_30H"/>
                    <Is_Bridge_Table rdf:datatype=
                    "http://www.w3.org/2001/XMLSchema#boolean"
                    >false</Is_Bridge_Table>
                    <owl:disjointWith rdf:resource="#_30U"/>
                    <owl:disjointWith rdf:resource="#_30V"/>
                    <owl:disjointWith rdf:resource="#_30P"/>
                    <owl:disjointWith rdf:resource="#_30C"/>
                    <owl:disjointWith rdf:resource="#_30D"/>
                    <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
                    <owl:disjointWith rdf:resource="#_30R"/>
                    <owl:disjointWith>
                     <Table_Metaclass rdf:ID="_30Y">
                      <owl:disjointWith rdf:resource="#_30C"/>
```

```
                    <owl:disjointWith rdf:resource="#_30U"/>
                    <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
                    <owl:disjointWith rdf:resource="#_30B"/>
                    <owl:disjointWith rdf:resource="#_30D"/>
                    <owl:disjointWith rdf:resource="#_30J"/>
                    <owl:disjointWith rdf:resource="#_30V"/>
                    <owl:disjointWith rdf:resource="#_30H"/>
                    <rdfs:label rdf:datatype=
                    "http://www.w3.org/2001/XMLSchema#string"
                    >30Y</rdfs:label>
                    <owl:disjointWith rdf:resource="#_30R"/>
                    <owl:disjointWith rdf:resource="#_30P"/>
                    <Is_Bridge_Table rdf:datatype=
                    "http://www.w3.org/2001/XMLSchema#boolean"
                    >false</Is_Bridge_Table>
                   </Table_Metaclass>
                  </owl:disjointWith>
                  <rdfs:label rdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#string"
                  >30J</rdfs:label>
                  <owl:disjointWith rdf:resource="#_30B"/>
                 </Table_Metaclass>
                </owl:disjointWith>
                <owl:disjointWith rdf:resource="#_30P"/>
                <Is_Bridge_Table rdf:datatype=
                "http://www.w3.org/2001/XMLSchema#boolean"
                >false</Is_Bridge_Table>
                <owl:disjointWith rdf:resource="#_30B"/>
                <owl:disjointWith rdf:resource="#_30Y"/>
                <owl:disjointWith rdf:resource="#_30H"/>
               </Table_Metaclass>
              </owl:disjointWith>
              <owl:disjointWith rdf:resource="#_30P"/>
              <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
              <owl:disjointWith rdf:resource="#_30V"/>
              <owl:disjointWith rdf:resource="#_30D"/>
              <Is_Bridge_Table rdf:datatype=
              "http://www.w3.org/2001/XMLSchema#boolean"
              >false</Is_Bridge_Table>
              <owl:disjointWith rdf:resource="#_30B"/>
              <owl:disjointWith rdf:resource="#_30J"/>
              <owl:disjointWith rdf:resource="#_30Y"/>
              <owl:disjointWith rdf:resource="#_30R"/>
              <owl:disjointWith rdf:resource="#_30U"/>
             </Table_Metaclass>
            </owl:disjointWith>
            <Is_Bridge_Table rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
            >false</Is_Bridge_Table>
            <owl:disjointWith rdf:resource="#_30P"/>
            <owl:disjointWith rdf:resource="#_30U"/>
            <owl:disjointWith rdf:resource="#_30C"/>
            <owl:disjointWith rdf:resource="#_30Y"/>
            <owl:disjointWith rdf:resource="#_30D"/>
            <owl:disjointWith rdf:resource="#_30J"/>
            <owl:disjointWith rdf:resource="#_30V"/>
           </Table_Metaclass>
          </owl:disjointWith>
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >30R</rdfs:label>
          <owl:disjointWith rdf:resource="#_30V"/>
          <owl:disjointWith rdf:resource="#_30Y"/>
          <owl:disjointWith rdf:resource="#_30J"/>
          <owl:disjointWith rdf:resource="#_30H"/>
          <Is_Bridge_Table rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
          >false</Is_Bridge_Table>
          <owl:disjointWith rdf:resource="#_30P"/>
          <owl:disjointWith rdf:resource="#_30C"/>
          <owl:disjointWith rdf:resource="#_30U"/>
         </Table_Metaclass>
        </owl:disjointWith>
        <Is_Bridge_Table rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</Is_Bridge_Table>
        <owl:disjointWith rdf:resource="#_30H"/>
```

```
          <owl:disjointWith rdf:resource="#_30Y"/>
          <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
          <owl:disjointWith rdf:resource="#_30C"/>
          <owl:disjointWith rdf:resource="#_30B"/>
          <owl:disjointWith rdf:resource="#_30V"/>
          <owl:disjointWith rdf:resource="#_30J"/>
         </Table_Metaclass>
        </owl:disjointWith>
        <owl:disjointWith rdf:resource="#_30J"/>
        <owl:disjointWith rdf:resource="#_30C"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >30V</rdfs:label>
        <owl:disjointWith rdf:resource="#_30U"/>
        <owl:disjointWith rdf:resource="#_30B"/>
        <owl:disjointWith rdf:resource="#_30H"/>
        <Is_Bridge_Table rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</Is_Bridge_Table>
        <owl:disjointWith rdf:resource="#_30Y"/>
        <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
        <owl:disjointWith rdf:resource="#_30R"/>
        <owl:disjointWith rdf:resource="#_30D"/>
       </Table_Metaclass>
      </owl:disjointWith>
      <owl:disjointWith rdf:resource="#_30B"/>
      <owl:disjointWith rdf:resource="#_30J"/>
      <Is_Bridge_Table rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</Is_Bridge_Table>
      <owl:disjointWith rdf:resource="#_30C"/>
      <owl:disjointWith rdf:resource="#_30R"/>
      <owl:disjointWith rdf:resource="#_30Y"/>
      <owl:disjointWith rdf:resource="#_30H"/>
      <owl:disjointWith rdf:resource="#_30P"/>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >30D</rdfs:label>
      <owl:disjointWith rdf:resource="#_30U"/>
     </Table_Metaclass>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#_30J"/>
    <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
    <owl:disjointWith rdf:resource="#_30H"/>
    <owl:disjointWith rdf:resource="#_30C"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >30U</rdfs:label>
    <owl:disjointWith rdf:resource="#_30V"/>
    <owl:disjointWith rdf:resource="#_30R"/>
    <owl:disjointWith rdf:resource="#_30P"/>
    <owl:disjointWith rdf:resource="#_30Y"/>
    <Is_Bridge_Table rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</Is_Bridge_Table>
    <owl:disjointWith rdf:resource="#_30B"/>
```

C.2. Portion de l'ontologie cible

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
   xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns="http://www.owl-ontologies.com/unnamed.owl#"
 xml:base="http://www.owl-ontologies.com/unnamed.owl">
 <owl:Ontology rdf:about=""/>
 <owl:Class rdf:ID="_30CA10">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30CA"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BA20">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30BA"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BB21">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30BB"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BT20">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30BT"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30PA21S158">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30PA"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#_30PA">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="centraletopo"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BT23K100">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BT"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BA30">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BA"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30CB">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
 </owl:Class>
 <owl:Class rdf:ID="_30BT24">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BT"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BT23X300">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BT"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30CB08">
  <rdfs:subClassOf rdf:resource="#_30CB"/>
 </owl:Class>
```

```
<owl:Class rdf:ID="_30BT22H100">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BT"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30BA10">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BA"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30PA21S161">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
</owl:Class>
<owl:Class rdf:ID="_30P">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
</owl:Class>
<owl:Class rdf:about="#_30BB">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
</owl:Class>
<owl:Class rdf:about="#_30BT">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
</owl:Class>
<owl:Class rdf:ID="_30BB23">
  <rdfs:subClassOf rdf:resource="#_30BB"/>
</owl:Class>
<owl:Class rdf:ID="_30C">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
</owl:Class>
<owl:Class rdf:ID="_30BB32">
  <rdfs:subClassOf rdf:resource="#_30BB"/>
</owl:Class>
<owl:Class rdf:ID="_30PA21S123">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
</owl:Class>
<owl:Class rdf:ID="_30BT23C500">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
</owl:Class>
<owl:Class rdf:ID="_30PA21S157">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
</owl:Class>
<owl:Class rdf:ID="_30B">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
</owl:Class>
<owl:Class rdf:ID="_30CB07">
  <rdfs:subClassOf rdf:resource="#_30CB"/>
</owl:Class>
<owl:Class rdf:ID="_30BA21">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BA"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30BT22H200">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
</owl:Class>
<owl:Class rdf:ID="_30CB03">
  <rdfs:subClassOf rdf:resource="#_30CB"/>
</owl:Class>
<owl:Class rdf:ID="_30BT23">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
</owl:Class>
<owl:Class rdf:ID="_30PA21S129">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
</owl:Class>
```

C.3. Portion de l'ontologie complémentaire

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
   xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:swrl="http://www.w3.org/2003/11/swrl#"
```

```
  xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://www.owl-ontologies.com/unnamed.owl#"
xml:base="http://www.owl-ontologies.com/unnamed.owl">
<owl:Ontology rdf:about=""/>
<owl:Class rdf:ID="__30JK04">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="__30J"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30CA06">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30CA"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="__30P">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="diagnostic-equipement2008"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="__30BB21">
  <owl:equivalentClass>
    <owl:Class rdf:ID="_30BB21"/>
  </owl:equivalentClass>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="__30B"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:about="#_30BB21">
  <owl:equivalentClass rdf:resource="#__30BB21"/>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30BB"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30PA21S158">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="_30PA"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30CB">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="centraletopo"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="__30YC54">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="__30Y"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="_30P">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
</owl:Class>
<owl:Class rdf:ID="_30CA12">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30CA"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="__30BT20">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#__30B"/>
  </rdfs:subClassOf>
  <owl:equivalentClass>
    <owl:Class rdf:ID="_30BT20"/>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="_30BB35">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BB"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="__30JK03">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#__30J"/>
```

```
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30B">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
  <owl:equivalentClass>
   <owl:Class rdf:about="#__30B"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="__30UA12">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="__30U"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30BB20">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30B"/>
  </rdfs:subClassOf>
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30BB20"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="_30BT11D400">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="_30BT"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30DL08">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="__30D"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#__30J">
  <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
 </owl:Class>
 <owl:Class rdf:about="#_30BB20">
  <owl:equivalentClass rdf:resource="#__30BB20"/>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#_30BB"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30BT21">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30B"/>
  </rdfs:subClassOf>
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30BT21"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="__30UB70">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30U"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30PA21S129">
  <rdfs:subClassOf rdf:resource="#__30P"/>
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30PA21S129"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="__30R">
  <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
 </owl:Class>
<owl:Class rdf:ID="__30VE10">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="__30V"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#_30PA">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
 </owl:Class>
 <owl:Class rdf:ID="_30CA08">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#_30CA"/>
```

```
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30UG10">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30U"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30PA21S157">
  <rdfs:subClassOf rdf:resource="#__30P"/>
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30PA21S157"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="_30CB08">
  <rdfs:subClassOf rdf:resource="#_30CB"/>
 </owl:Class>
<owl:Class rdf:ID="_30PA21S146">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
 </owl:Class>
 <owl:Class rdf:ID="__30UB">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30U"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30YM50">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30Y"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#_30BT">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
 </owl:Class>
 <owl:Class rdf:ID="__30PA21S147">
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30PA21S147"/>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#__30P"/>
 </owl:Class>
 <owl:Class rdf:ID="_30C">
  <rdfs:subClassOf rdf:resource="#centraletopo"/>
  <owl:equivalentClass>
   <owl:Class rdf:ID="__30C"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="_30PA21S155">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
 </owl:Class>
 <owl:Class rdf:about="#_30PA21S157">
  <owl:equivalentClass rdf:resource="#__30PA21S157"/>
  <rdfs:subClassOf rdf:resource="#_30PA"/>
 </owl:Class>
 <owl:Class rdf:ID="__30CU60">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30C"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30BA29">
  <rdfs:subClassOf>
   <owl:Class rdf:ID="_30BA"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30PA21S161">
  <rdfs:subClassOf rdf:resource="#__30P"/>
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30PA21S161"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="__30VE56Z0300">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30V"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30VB10">
```

```
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30V"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#_30BT21">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
  <owl:equivalentClass rdf:resource="#__30BT21"/>
 </owl:Class>
 <owl:Class rdf:ID="__30YC41S0202">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30Y"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#_30PA21S147">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
  <owl:equivalentClass rdf:resource="#__30PA21S147"/>
 </owl:Class>
 <owl:Class rdf:about="#__30Y">
  <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
 </owl:Class>
 <owl:Class rdf:ID="__30UC10">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30U"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30H">
  <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
 </owl:Class>
 <owl:Class rdf:ID="_30CB01">
  <rdfs:subClassOf rdf:resource="#_30CB"/>
 </owl:Class>
 <owl:Class rdf:ID="__30YC21Z0100">
  <rdfs:subClassOf rdf:resource="#__30Y"/>
 </owl:Class>
 <owl:Class rdf:ID="_30BB26">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#_30BB"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="__30VB30">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#__30V"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="_30PA21S148">
  <rdfs:subClassOf rdf:resource="#_30PA"/>
 </owl:Class>
 <owl:Class rdf:ID="_30BA20">
  <owl:equivalentClass>
   <owl:Class rdf:ID="__30BA20"/>
  </owl:equivalentClass>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#_30BA"/>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:about="#_30BT20">
  <owl:equivalentClass rdf:resource="#__30BT20"/>
  <rdfs:subClassOf rdf:resource="#_30BT"/>
 </owl:Class>
 <owl:Class rdf:ID="__30YM34Z0303">
  <rdfs:subClassOf rdf:resource="#__30Y"/>
 </owl:Class>
<owl:Class rdf:ID="_30BA30">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#_30BA"/>
  </rdfs:subClassOf>
  <owl:equivalentClass>
   <owl:Class rdf:ID="__30BA30"/>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="__30PA21S159">
  <owl:equivalentClass>
   <owl:Class rdf:ID="_30PA21S154"/>
```

```
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#__30P"/>
</owl:Class>
<owl:Class rdf:ID="_30BT21B400">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
</owl:Class>
<owl:Class rdf:ID="_30BT10">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
</owl:Class>
<owl:Class rdf:ID="_30BA10">
  <owl:equivalentClass>
    <owl:Class rdf:ID="__30BA10"/>
  </owl:equivalentClass>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#_30BA"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="__30YC51">
  <rdfs:subClassOf rdf:resource="#__30Y"/>
</owl:Class>
<owl:Class rdf:about="#__30D">
  <rdfs:subClassOf rdf:resource="#diagnostic-equipement2008"/>
</owl:Class>
<owl:Class rdf:ID="_30BT15">
  <rdfs:subClassOf rdf:resource="#_30BT"/>
</owl:Class>
```

Annexe D

1. Le résultat d'alignement Manuel

Concept d'ontologie Source « diagnostic »	Relation	Concept d'ontologie Target « centraletopo »
30P	Equivalence	30P
30C	Equivalence	30C
30BT21	Equivalence	30BT21
30BT20	Equivalence	30BT20
30BB23	Equivalence	30BB23
30BB21	Equivalence	30BB21
30BB20	Equivalence	30BB20
30BA30	Equivalence	30BA30
30BA10	Equivalence	30BA10
30B	Equivalence	30B
30PA10	Equivalence	30PA10
30PA21S123	Equivalence	30PA21S123
30PA21S129	Equivalence	30PA21S129
30PA21S147	Equivalence	30PA21S147
30PA21S157	Equivalence	30PA21S157
30PA21S159	Equivalence	30PA21S159
30PA21S161	Equivalence	30PA21S161
30PA23	Equivalence	30PA23
30CO42	Sous classe	30C
30CJ22	Sous classe	30C
30BA10	Sous classe	topo_30BA
30PA21S159	Sous classe	30PA
30CJ27G100	Sous classe	30C
30PA23	Sous classe	30PA
30CG18	Sous classe	30C
30CG13	Sous classe	30C
30CK07G100	Sous classe	30C
30CK17	Sous classe	30C
30CJ27	Sous classe	30C
30PA10	Sous classe	30PA
30PA10	Sous classe	30PA
30BT21	Sous classe	30B
30BT20	Sous classe	30B
30P	Sous classe	centraletopo
30CG18G100	Sous classe	30C
30PA23	Sous classe	30PA
30CU60	Sous classe	30C
30BT21	Sous classe	30B
30BT20	Sous classe	30B
30CC06	Sous classe	30C
30CT63	Sous classe	30C
30PA21S129	Sous classe	30PA
30PA21S123	Sous classe	30PA
30BA30	Sous classe	topo_30BA
30CC19	Sous classe	30C
30BA30	Sous classe	topo_30BA
30BB20	Sous classe	30BB
30PA21S123	Sous classe	30PA
30PA21S129	Sous classe	30PA

30BB21	Sous classe	30BB
30BB23	Sous classe	30BB
30BA22	Sous classe	topo_30BA
30CO33	Sous classe	30C
30BA10	Sous classe	topo_30BA
30BB23	Sous classe	30BB
30BB21	Sous classe	30BB
30BB20	Sous classe	30BB
diagnostic-equipement2008	Super classe	30PA21S161
30P	Super classe	30PA21S161
Total des correspondances=60		

Tableau D.1. Alignement de référence de deux ontologies de Turbine à vapeur

2. Alignement en utilisant COMA++ : $O_1 \rightarrow O_2$

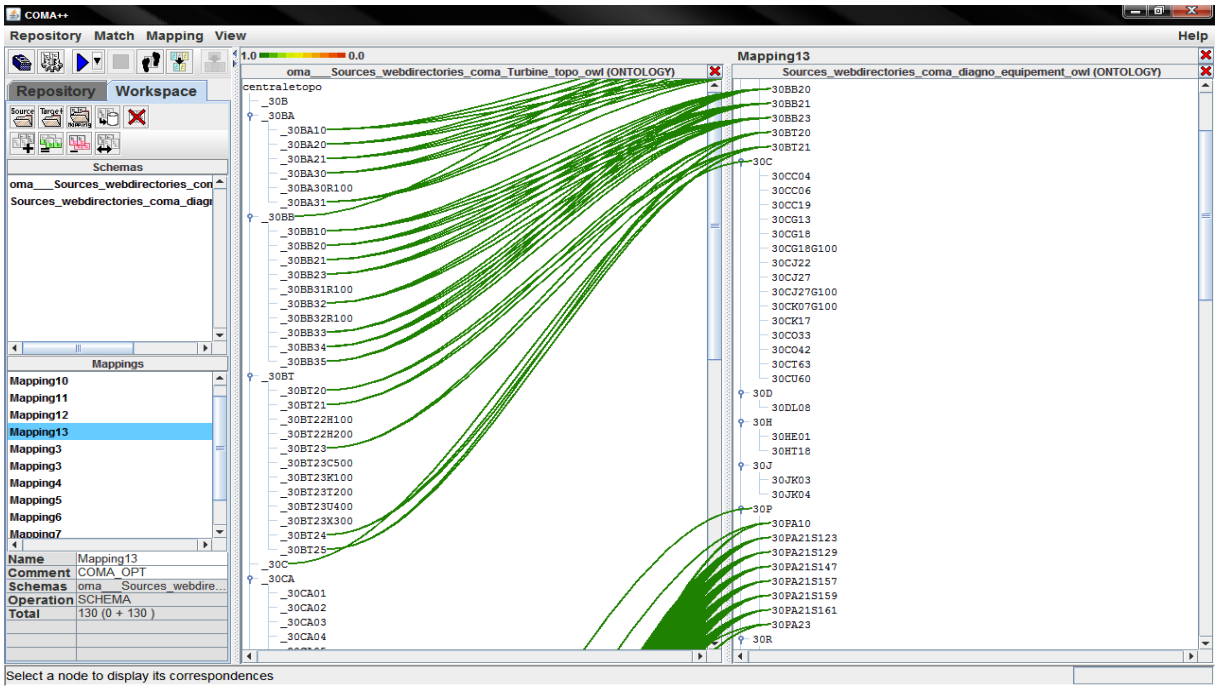

Figure D.1. Alignement de deux ontologies de Turbine à vapeur en utilisant COMA++

3. Alignement en utilisant XMAP : $O_1 \rightarrow O_2$

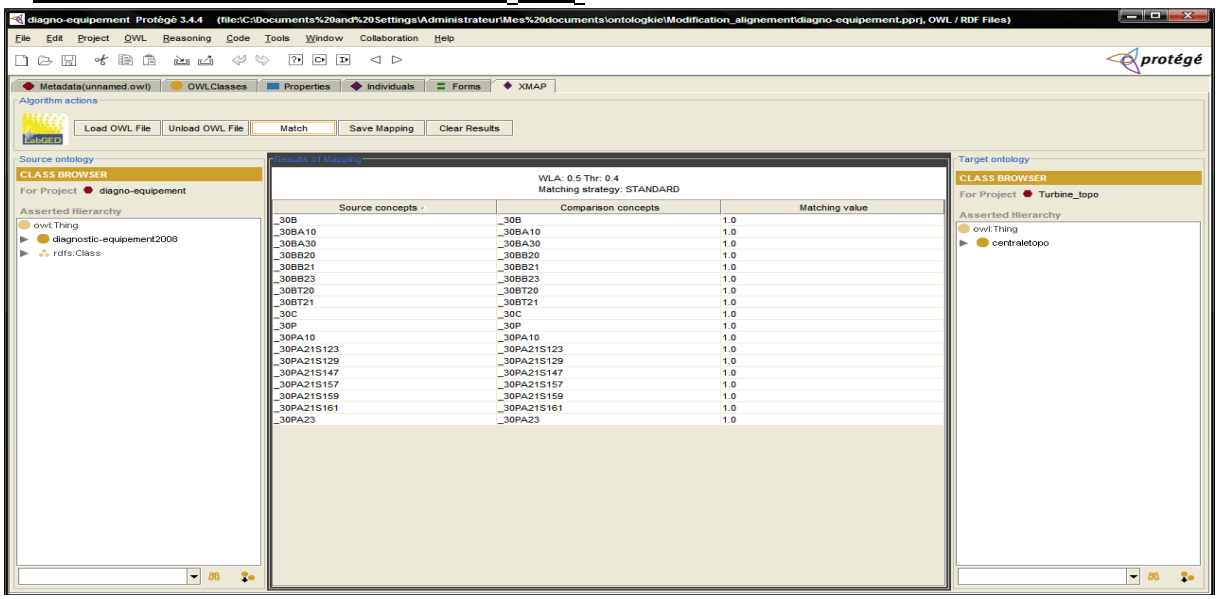

Figure D.2. Alignement de deux ontologies de Turbine à vapeur en utilisant XMAP

4.1. Alignement en utilisant BRMAP : $O_1 \rightarrow O_2$

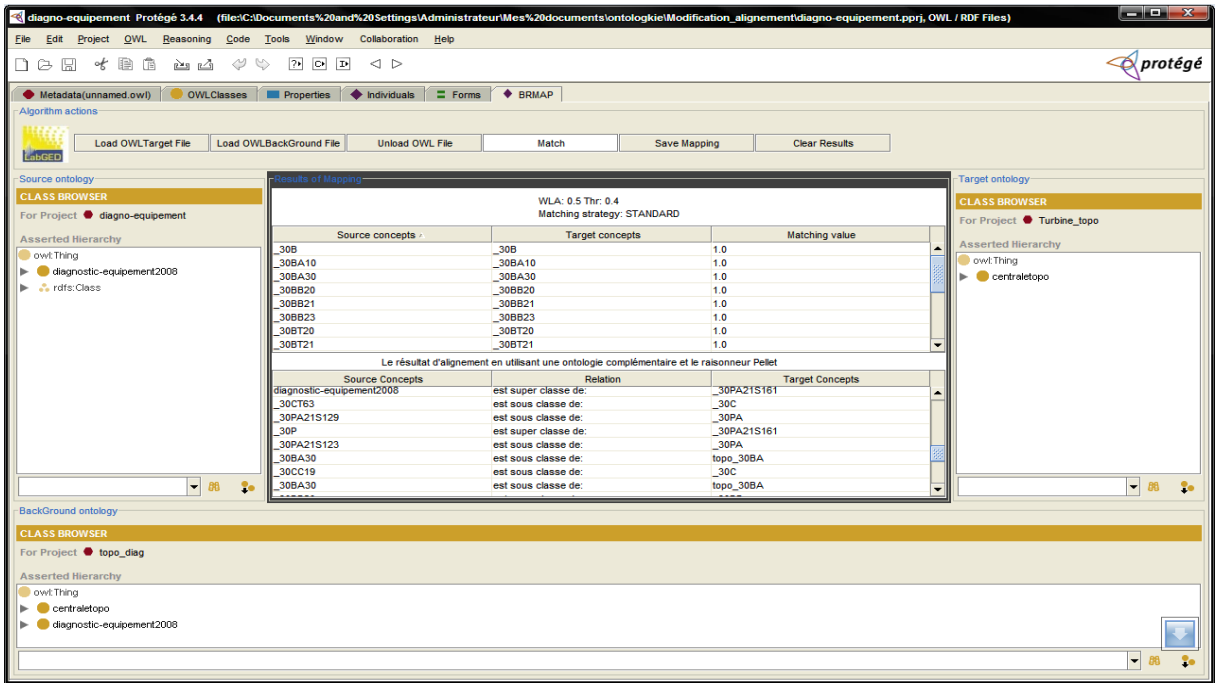

Figure D.3. Alignement de deux ontologies de Turbine à vapeur en utilisant BRMAP

4.2. Le résultat d'alignement en utilisant BRMAP :

Concept d'ontologie Source « diagnostic »	**Relation**	**Concept d'ontologie Target « centraletopo »**
30P	Equivalence	30P
30C	Equivalence	30C
30BT21	Equivalence	30BT21
30BT20	Equivalence	30BT20
30BB23	Equivalence	30BB23
30BB21	Equivalence	30BB21
30BB20	Equivalence	30BB20
30BA30	Equivalence	30BA30
30BA10	Equivalence	30BA10
30B	Equivalence	30B
30PA10	Equivalence	30PA10
30PA21S123	Equivalence	30PA21S123
30PA21S129	Equivalence	30PA21S129
30PA21S147	Equivalence	30PA21S147
30PA21S157	Equivalence	30PA21S157
30PA21S159	Equivalence	30PA21S159
30PA21S161	Equivalence	30PA21S161
30PA23	Equivalence	30PA23
Les relations découvertes quand on utilise une ontologie diag_topo et Pellet		
30C	Equivalence	30C
30PA21S161	Equivalence	30PA21S161
30PA21S147	Equivalence	30PA21S147
30PA21S157	Equivalence	30PA21S157
30PA21S159	Equivalence	30PA21S159
30PA21S157	Equivalence	30PA21S157
30PA21S161	Equivalence	30PA21S161
30PA21S147	Equivalence	30PA21S147
30B	Equivalence	30B
30C	Equivalence	30C
30B	Equivalence	30B
30CO42	Sous classe	30C
30CJ22	Sous classe	30C
30BA10	Sous classe	topo_30BA
30PA21S159	Sous classe	30PA
30CJ27G100	Sous classe	30C
30PA23	Sous classe	30PA
30CG18	Sous classe	30C
30CG13	Sous classe	30C
30CK07G100	Sous classe	30C
30CK17	Sous classe	30C
30CJ27	Sous classe	30C
30PA10	Sous classe	30PA
30PA10	Sous classe	30PA
30BT21	Sous classe	30B
30BT20	Sous classe	30B
30P	Sous classe	centraletopo
30CG18G100	Sous classe	30C

30PA23	Sous classe	30PA
30CU60	Sous classe	30C
30BT21	Sous classe	30B
30BT20	Sous classe	30B
30CC06	Sous classe	30C
30CT63	Sous classe	30C
30PA21S129	Sous classe	30PA
30PA21S123	Sous classe	30PA
30BA30	Sous classe	topo_30BA
30CC19	Sous classe	30C
30BA30	Sous classe	topo_30BA
30BB20	Sous classe	30BB
30PA21S123	Sous classe	30PA
30PA21S129	Sous classe	30PA
30BB21	Sous classe	30BB
30BB23	Sous classe	30BB
30BA22	Sous classe	topo_30BA
30CO33	Sous classe	30C
30BA10	Sous classe	topo_30BA
30BB23	Sous classe	30BB
30BB21	Sous classe	30BB
30BB20	Sous classe	30BB
diagnostic-equipement2008	Super classe	30PA21S161
30P	Super classe	30PA21S161

Tableau D.2. Alignement de deux ontologies de Turbine à vapeur en utilisant BRMAP