République Algérienne Démocratique et Populaire

Ministère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

Centre Universitaire de Souk-Ahras

Faculté des Sciences et Technologies

Département Maths & Informatique

Ecole Doctorale de l'Est pôle Annaba

Mémoire En vue de l'obtention

Du diplôme de Magister

Spécialité Informatique option Intelligence Artificielle

Présenté Par

Siham AMROUCH

Encadré par

Dr. Mohamed Tarek KHADIR

Fusion d'ontologies de domaine pour la gestion de connaissances
Application aux Turbines à Vapeur

Promotion 2008-2009

À ma mère et mon père

À mes sœurs et mes frères

À ma nièce Ritège

Remerciements

Je tiens à remercier tout d'abord, mon directeur de recherche M. le Docteur Mohamed Tarek KHADIR d'avoir assuré une direction motivante tout au long de ce travail. Qu'il trouve ici l'expression de toute ma reconnaissance et ma gratitude.

Je remercie également …… d'avoir accepter de présider le jury. J'adresse mes remerciements à ……..et…… d'avoir accepté de juger ce mémoire.

Un remerciement spécial est aussi adressé à tout enseignant ayant contribué à ma formation

Mes vifs remerciements sont aussi adressés à ma mère qui n'a cessé de prier pour que j'atteigne mes objectifs dans la vie, à mon père la source et l'origine de tous mes succès, et à mes frères Halim et Nizar ainsi que mes sœurs Mounira et Afaf pour tous leurs soutiens et encouragements.

Je réserve mes derniers remerciements à tous ceux qui ont contribué de près ou de loin pour la réalisation de ce travail.

Liste des figures

Liste des tables

Partie 1 : Etat de l'art.

Chapitre I: Introduction

I. Introduction à la représentation et à la gestion de connaissances

Chapitre II: Le concept ontologique et la réutilisation d'ontologies.

Chapitre III: La fusion d'ontologies.

Chapitre IV: Les Turbines à vapeur et la maintenance industrielle.

Partie 2: Contribution.

Chapitre V: Conception, construction et fusion d'ontologies représentant la Turbine à Vapeur.

Chapitre VI: Le système général proposé, Conception et Implémentation.

I. Introduction à la représentation et à la gestion de connaissances

I.1. Introduction

Un formalisme de représentation des connaissances a pour but de permettre la modélisation de différents domaines où le choix de type de représentation n'est pas toujours évident, tel que dans des domaines complexes comme le commerce en ligne ou le contrôle d'un robot dans un environnement physique dynamique exigeant des représentations plus simples et plus souples. Dans cette introduction, nous montrons comment créer ces représentations en intégrant les concepts généraux tels que les actions, le temps, l'espace, les évènements mentaux, etc.

Nous commencerons cette section, en introduisant tout d'abord la notion d'ontologie générale qui organise les différents domaines en une hiérarchie de catégories de base des objets, de substances, et des mesures. Ensuite en examinant les représentations des actions capitales pour la construction d'agents fondés sur les connaissances et en expliquant aussi la notion des évènements (portions d'espaces temps) puis, les logiques des défauts ainsi que la notion des mondes clos, et en terminant par différents exemples de systèmes de représentation de connaissances dédiés à faire des raisonnements sur les catégories, tels que les réseaux sémantiques, les logiques de description et les ontologies.

Et à la fin, nous précisons notre objectif de ce travail et nous décrivons la structure de ce mémoire.

I.2. Ingénierie ontologique

C'est la représentation des concepts abstraits en faisant lien à l'ingénierie des connaissances et en nous basant sur l'idée de réserver des emplacements où les nouvelles connaissances de tout domaine pourront s'insérer. Nous définirons par exemple, ce que nous entendons par «objet physique» et les détails de différents types d'objets (robots, télévisions, livres, etc) pourront être spécifiés ultérieurement.

La représentation ontologique a utilisé au début la logique du premier ordre qui a présenté la limite de traiter les exceptions des généralisations. Par exemple, bien que

«les tomates sont rouges» soit une règle utile, certaines tomates sont vertes, jaunes ou oranges, d'où la nécessité d'apparition des ontologies générales à partir des ontologies spécialisés des domaines considérés qui doivent être pratiquement applicables à n'importe quel domaine particulier en les complétant par les axiomes spécifiques au domaine. Donc les différentes parties des connaissances de n'importe quel domaine doivent être unifiées, car le raisonnement et la résolution de problèmes peuvent porter simultanément sur plusieurs parties.

Pour mettre à l'épreuve cette ontologie, le domaine du e-commerce est très adéquat en considérant par exemple que l'agent d'achat sur Internet doit connaître une multitude de sujets et d'auteurs pour acheter des livres sur Amazon.com, tous les types de produits alimentaires pour faire ses courses sur Houra.fr, et ne rien ignorer des vide-greniers pour trouver de bonnes affaires sur Ebay.com.

I.3. Catégories et objets

L'organisation des objets en catégories constitue une part essentielle de la représentation des connaissances car la plus grande partie des raisonnement se déroule au niveau des catégories qui permettent d'effectuer des prédictions lorsque les objets sont classifiés, tel que on infère la présence d'objets à partir d'entrées perceptives, l'appartenance à une catégorie à partir des propriétés perçues, et l'on utilise ensuite les informations catégorielles pour effectuer des prédictions sur les objets. Par exemple, la couleur verte d'un objet, sa peau lisse et sa forme ronde permettent d'inférer qu'il s'agit d'une pastèque, puis qu'elle pourrait être utile dans une salade de fruits.

On constate donc que la logique du premier ordre offre deux possibilités pour représenter les catégories: les prédicats et les objets.

Les catégories servent donc à organiser et à simplifier la base de connaissance via l'héritage et les relations de sous classes entre les catégories, les organise en une taxonomie ou une hiérarchie taxonomique. Mais d'autres relations tel que «sous classe» sont parfois très utiles tel que les relations:

Disjoints({Animaux, Végétaux})

DécompositionExhaustives({Etasuniens, Canadien, Mexicains}, NordAmericains)

Partition({Mâles, Femelles}, Animaux}

I.3.1 Composition physique

Elle décrit le fait qu'un objet peut faire partie d'une autre partie, et donc que les objets peuvent être regroupés dans des hiérarchies *PartieDe*, tel que la relation *PartieDe* est transitive et réflexive.

Exemple: *PartieDe (*Bucarest, Roumanie*).*

On peut aussi définir une relation *PrtitionPartie* décrivant qu'un objet est composé de parties de sa *PartitionPartie* tel que la masse d'un objet composite est la somme des masses de ses parties.

I.3.2 Mesures

Généralement un objet est caractérisé par sa hauteur, sa masse, son coût…, et les valeurs qu'on les attribue sont donc des mesures, les mesures quantitatives ordinaires sont assez faciles à représenter tel que pour indiquer la longueur d'un segment S_1, on utilise: longueur (S_1) = pouce (1.5) = centimètres (3.81),

et on peut effectuer la conversion entre les unités à l'aide d'une équation entre les multiples d'une unité dans les termes de l'autre et on écrit: Centimètres (2.54*d) = pouce (d).

Mais d'autres mesures posent un problème plus important parce qu'il n' y a pas d'échelle de valeur incontestable. Par exemple les exercices sont plus ou moins difficiles, les poèmes sont plus ou moins beaux, d'où on conclut que l'aspect le plus important des mesures ne tient pas à leurs valeurs numériques particulières mais au fait qu'elles peuvent être ordonnées par exemple par une valeur (>). Par exemple en déclarant que les exercices de Norvig sont plus difficiles que ceux de Russel:

E_1 Є Exercices et E_2 Є Exercices et Auteur (Norvig, E_1) et Auteur (Russel, E_2) →Difficulté (E_1) > Difficulté (E_2).

I. 3.3 Substances et objets

Tous les objets physiques appartiennent aux deux catégories qui sont coextensives c'est à dire qu'elles renvoient aux mêmes entités, en l'occurrence les substances et les objets. Les substances constituant des propriétés qui sont intrinsèques (appartenant à la substance plutôt qu'à l'objet lui même), tel que lorsqu'on coupe une substance en deux, les deux morceaux conservent les mêmes propriétés intrinsèques (par exemple la densité et l'énergie), à l'inverse des objets caractérisés par des propriétés extrinsèques tel que la longueur, le poids,…

I.4. Actions, Situations et Évènements

Les raisonnements portant sur les résultats des actions constituent une part essentielle des opérations d'un agent fondé sur les connaissances.

I.4.1 Ontologie de calcul des situations

Les actions sont des prédicats tel que: *Avancer*, *Tourner* (à droite), et les situations sont des termes logiques, S_0 est la situation initiale, et chaque situation S_i se calcule en appliquant une action à une situation précédente S_{i-n}, la fonction *résultat (a,s)* dénote que le résultat de l'action a est la situation s.

Pour commencer, on peut dire que l'exécution d'une séquence d'actions vide laisse la situation inchangée: *Résultat ([], s)= s,* et l'exécution d'une séquence d'actions non vide revient au même que l'exécution de la première action puis des celles des suivantes à partir de la situation résultante:

Résultat ([a/seq],s) = *Résultat* (seq, *Résultat (*a,s*))*

I.4.2 Description des actions en calcul des situations

Dans la version du calcul des situations la plus simple, chaque action est décrite par deux axiomes: un axiome de possibilité qui indique quand il est possible d'exécuter l'action, et un axiome d'effet qui décrit ce que produit l'exécution d'une

action possible. Nous emploierons *poss (a,s)* pour signifier la possibilité d'exécuter une action *a* dans une situation *s*. Les axiomes ont la forme suivante:

AXIOME DE POSSIBILITE: *préconditions → poss(a,s).*

AXIOME D'EFFET: *poss(a,s) → Changement qui résulte de l'exécution de l'action.*

Exemples:

1/ *A(Agent,x,s) et Adjacent(x,y) → poss(Aller(x,y),s).* Axiome de possibilité.

2/ poss(Aller(x,y),s) → A(Agent,y,Résultat (x,y),s)). Axiome d'effet.

Si on suppose qu'un agent est en position [1.1], et qu'il doit aller à la situation où il y a de l'or (G1) qui est [1.2] pour le ramener, l'axiome d'effet pour aller nous permet de conclure que l'agent atteint l'emplacement [1.2]:

A(Agent,[1.2]Résultat(Aller[1.1],[1.2],s0)), si on considère maintenant l'action *ramasser(G1)* il faut montrer que celle-ci est possible dans la nouvelle situation à savoir: *A(G1,[1.2]Résultat(Aller[1.1],[1.2],s0)),*

Mais l'action aller de l'agent ne doit pas affecter l'emplacement de l'or qui est en [1.2], là où il était en *s0*, le problème est donc que les axiomes d'effets indique ce qui change mais pas ce qui demeure identique (problème de la persistance). Le problème d'inférence de la persistance qui lui est étroitement associé consiste à projeter le résultat d'une séquence d'actions de *t* étapes à un instant plutôt qu'à un autre, et le troisième problème est celui de s'assurer que les conditions de succès d'une action ont été spécifiés tel que par exemple aller échoue si l'agent meurt en route, (ces problèmes ne sont pas résolus à ce stade)

I.4.3 Temps et calcul des évènements

De manière informelle un évènement est plus étendu qu'une action. Il est fondé sur des points dans le temps et non sur des situations et son calcul est conçu pour permettre le raisonnement sur des intervalles de temps. L'axiome de calcule des évènements dit qu'un fluent est vrai à un point dans le temps s'il a été initié par un événement à un moment dans le passé, et qu'un événement survenu n' y a pas mis fin

initié (e,f,t): l'événement e à l'instant t fait que le fluent f devient vrai.

termine (w,f,t): *f* cesse d'être vrai.

survient (e,t): l'événement *e* survient à l'instant *t*.

clippé (f,t,t2): *f* se termine à un moment entre *t* et *t2*.

De manière formelle l'axiome de calcule d'évènement est:

T(f,t2)↔ ∃ e, t Survient(e,t) et Initie(e,f,t) et (t<t2) et ¬Clippé(f,t,t2).

Clippé(f,t,t2)↔ ∃ e, t1 Survient(e,t1) et Termine(e,f,t1) et (t<t1) et (t1<t2).

I.4.4 Evènements généralisés

Un évènement généralisé est composé à partir de portions d'espace temps. Par exemple, la seconde guère mondiale est un événement généralisé qui a eu lieu aux différents points de l'espace temps et on peut la subdiviser en sous événements tel que:

SousEvènement (BatailleDAngleterre, SecondeGuèrreMondiale).

I.4.5 Les intervalles

Le temps est très important pour tout agent qui entreprend des actions, et on le représente à travers deux types d'intervalles: les instants et les intervalles étendus qui ne se diffèrent que par le fait que les instants ont une durée nulle.

On peut aussi définir les situations entre deux intervalles (i,j) tels que: *Rencontre(i,j), Avant (i,j), Après (i,j),Pendant (i,j), et Chevauchement (i,j).*

I.4.6 Fluents et objets

Un fluent décrit un objet physique ou un événement généralisé qui se diffère à des moments différents, par exemple le fluent :

*à un moment de l'année 1999 la population de l'USA s'élevait à 271 millio*ns s'écrit comme suit : *E (population (USA, 271.000.000),1999).*

I.5 Evènements mentaux et objets mentaux

Dans les domaines multi-agents, il devient important pour un agent de raisonner sur les états mentaux des autres agents. Ceci revient à identifier un modèle des objets mentaux, ainsi que les processus mentaux qui manipulent ces objets mentaux..

I.5.1 Théorie formelle des croyances

En plus des règles d'inférence normales, il faut disposer de règles spécifiques aux croyances qui expriment les relations entre les agents et les objets mentaux, par exemple croire, vouloir, savoir,…tel que selon les axiomes un agent peut déduire infailliblement n'importe quel conséquence de ces croyances. C'est ce qu'on appelle l'omniscience logique.

I.5.2 Connaissances, croyances, Temps et Actions

Une connaissance est une croyance vraie et justifiée. Souvent, un agent a affaire à des croyances qui évoluent avec le temps, et réalise donc des plans tenant en compte des modifications de ces propres croyances. Par exemple pour dire que Lois croit aujourd'hui que superman peut voler, on écrit:

T(Croire(Lois, «T(Voler(Superman)»),Aujourd'hui)

Si on considère que l'objet de croyance est changeable dans le temps, par exemple pour dire que Lois croit aujourd'hui que Superman a pu voler hier:

T(Croire(Lois, «T(Voler(Superman),hier»,Aujourd'hui)

I.6. Raisonnement avec informations par défaut

Par exemple le nombre de pieds d'un être humain est pris par défaut égal à 2 :

I.6.1 Le monde ouvert et le monde clos

La logique du premièr ordre diffère des systèmes de Base De Données d'au moins de deux manières: l'hypothèse du monde clos et l'hypothèse d'unicité des noms.

L'hypothèse du monde clos est inspirée du fait que les BDDs (et les gens) supposent que l'information fournie est complète, et que les énoncés atomiques de base non énoncés comme étant vrais sont supposés faux, alors que l'hypothèse d'unicité des noms est basée sur la supposition que les noms distincts renvoient aux objets distincts.

I.6.2 Négation par l'échec

L'idée est qu'un littéral négatif P puisse être«prouvé» vrai seulement dans le cas où la preuve de P échoue. C'est une forme de raisonnement par défaut étroitement liée à l'hypothèse du monde clos. On suppose que quelque chose est fausse s'il est impossible de prouver que c'est vrai. Par exemple : *DisqueSCSI* ← *Disque et not DisqueIDE*

I.6.3 Logique des défauts

La logique des défauts est un formalisme où la règle par défaut a la forme: *P:J1,…,Jn/C, où P* est appelé le prérequis, *C* la conclusion et *Ji* sont les justifications. Si l'on peut prouver que l'une d'entre elles est fausse, alors il est impossible de tirer la conclusion.

I.7. **Systèmes de raisonnement pour les catégories**

Trois familles de systèmes de représentation de connaissances spécifiques à l'organisation des raisonnements sur les catégories, en l'occurrence les réseaux sémantiques, les logiques de description et les ontologies:

I.7.1 Les réseaux sémantiques:

Une notation graphique typique qui affiche les noms d'objets ou de catégories dans des boites ovales ou rectangulaires et connecte celles-ci par des arcs portant des étiquettes. Par exemple la figure 1 indique un lien *MembreDe* entre *Marie* et

PersonnesDeSexeFéminin qui correspond à l'assertion logique *Marie* $\in$ *PersonnesDeSexeFéminin*.

La relation *AuneMère* associée à la boite à bordure double est une notation spécialisée pour indiquer qu'il s'agit d'une relation entre un ou plusieurs membres de la catégorie *personnes et un et un seul membre de la catégorie PersonneDeSexeFéminin,* ce lien énonce que:

$\forall$ x, x $\in$ personnes$\rightarrow$ [$\forall$ *y AuneMère(x,y)*$\rightarrow$ *y* $\in$ *PersonnesDeSexeFéminin*]

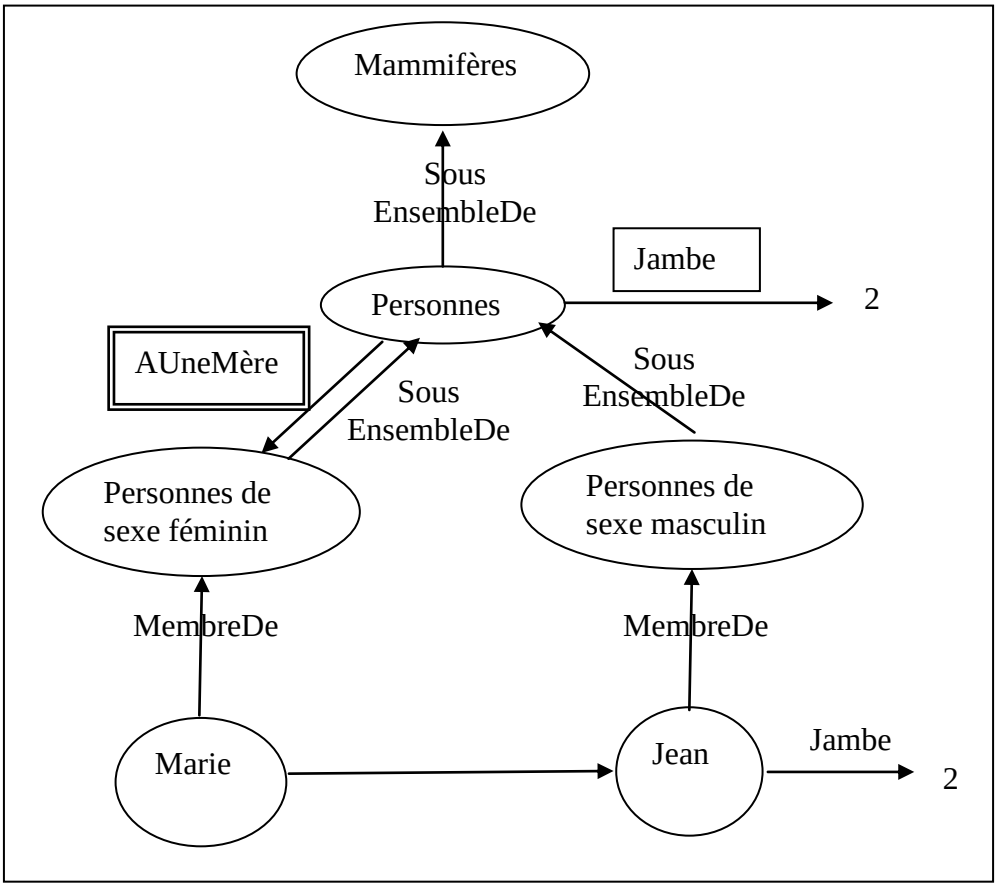


Figure I.1. Un réseau sémantique à 4 objets et 4 catégories

I.7.2 Logique de description

Les logiques de description facilitent la description des définitions et des propriétés des catégories. Leurs principales tâches d'inférence sont la *Subsomption* et la *Classification*.

Par exemple: Pour décrire l'ensemble des hommes ayant au moins trois fils dont tous

sont sans emploi et mariés à des médecins, et au maximum deux filles qui sont toutes professeurs dans un département de physique ou de mathématique, on écrira:

And(Homme,AtLeast(3,Fils), AuPlus(2,Fille),

All(Fils,And(SansEmploi,Marié,All(2pouse,Medcin))),

All(Fille,And(Professeur,Fills(département,Physique,Maths))))).

I.7.3 Les ontologies

Une ontologie aide à la description d'un domaine donné et peut servir de charpente à une base de connaissances. Il s'agit bien d'une notation formelle qui peut être graphique sous forme d'une hiérarchie de nœuds qui représentent les concepts ou les classes reliés par des arcs qui représentent les relations entre eux. Les ontologies visent à modéliser explicitement les connaissances d'un domaine particulier de manière à faciliter la recherche, l'extraction, l'intégration et le partage d'informations entre les différents systèmes ou individus d'une communauté au sein de ce domaine

La figure 2 présente un exemple d'ontologie modélisant le domaine d'une communauté de recherche au sein d'une université.

Parmi ces formalismes de représentation de connaissances et d'autres présentés dans la littérature, nous avons choisi d'exploiter la notion d'ontologies pour modéliser notre domaine d'application. Dans notre cas, le domaine de la maintenance industrielle à travers la modélisation de base de données représentant la Turbine à Vapeur. Notre choix s'est imposé vu qu'une ontologie supporte la voluminosité des BDDs qu'elle vise à refléter au temps où la Turbine à Vapeur qu'on veut modéliser est assez volumineuse (plus de 2000 composants), en plus de ses caractéristiques techniques facilitant la recherche, l'extraction, l'intégration, et le partage d'informations, lui permettant d'être intégré impeccablement dans un système expert ou un SBC.

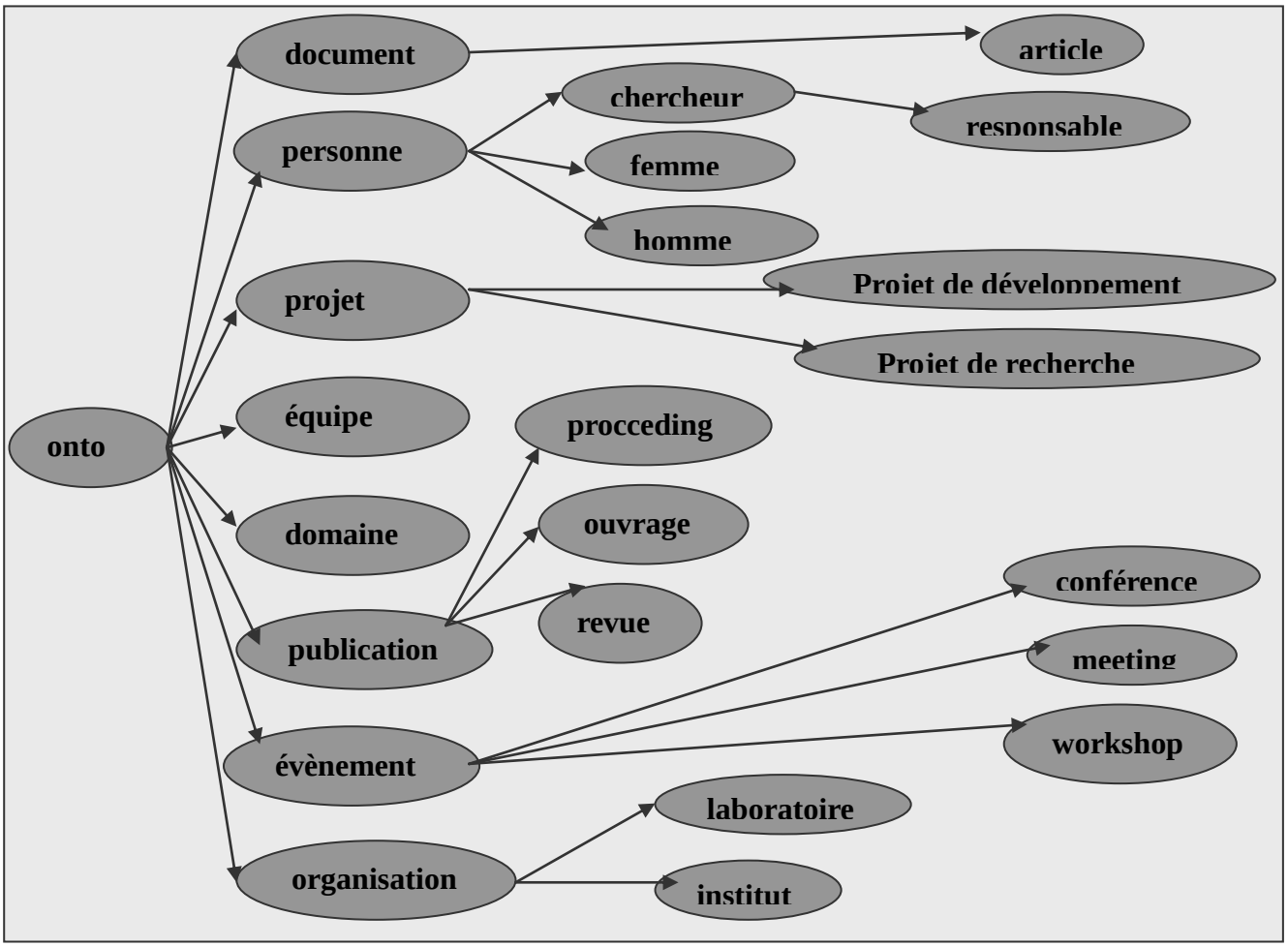


Figure I.2: une ontologie de domaine modélisant une communauté de recherche

I.8. Conclusion

Dans ce travail, une idée détaillée de la manière avec laquelle les bases de connaissances réelles sont construites est exposée, tel que une ontologie générale qui organise et lie ensemble plusieurs domaines de différentes spécialités et qui doit couvrir une grande diversité de connaissances permettant ainsi de gérer n'importe quel domaine. Une ontologie supérieur basée sur les catégories et le calcul des événements en abordant les notions des objets structurés, du temps et de l'espace, des changements des situations, des processus et des substances, a été aussi présentée, où la représentation des actions, des évènements et du temps est fait soit par le calcul des situations, soit par le calcul des évènements permettant ainsi à un agent de construire des plans grâce à l'inférence logique. Ensuite l'hypothèse du monde clos permettant l'intégrité des littéraux négatifs dans le raisonnement logique a été introduite. Pour

obtenir une organisation hiérarchique des catégories, trois types de systèmes de représentation ont été abordé, en l'occurrence les réseaux sémantiques, les logiques de description et les ontologies fournissant l'outil d'héritage comme une forme importante d'une règle d'inférence permettant de déduire les propriétés des objets à partir de leur appartenance à des catégories bien définies.

I.9. L'objectif de notre travail

L'objectif de notre travail est de modéliser une Turbine à Vapeur, et précisément celle exploitée au sein de l'entreprise de SONELGAZ pour la génération électrique, en utilisant la notion d'ontologie. Nous commençons alors, à partir de deux BDDs relationnelles décrivant la Turbine à Vapeur selon deux différents contextes, la première CentraleTopo décrit les caractéristiques topologiques de la Turbine à Vapeur tels que la description, la famille, la zone, la fonction de chaque composant, alors que la deuxième, Diagnostique, décrit les cas d'intervention sur chacun de ses composants, pour maintenance ou réparation, à travers la description du défaut, sa cause, son symptôme et son remède.

Alors, à partir de ces deux BDDs, nous construisons automatiquement deux ontologies Topo et Diagno, puis nous les utilisons à travers un processus de fusion d'ontologies, pour obtenir une ontologie plus large et plus complète couvrant un domaine d'application plus vaste, et offrant ainsi des réponses plus pertinentes aux questions des opérateurs et des techniciens qui utilisent cette turbine. Nous visons donc à fusionner ces deux ontologies à travers deux différentes manières: dans la première, nous allons utiliser le plugin sous protégé 2000, PROMPT, et dans la deuxième, nous allons proposer et implémenter un nouvel algorithme de fusion.

I.10. Structure du mémoire

Ce mémoire est structuré en deux parties couvrant six chapitres. La première partie décrivant l'état de l'art, est composée de quatre chapitres, alors que la deuxième partie, présentant notre contribution, est composée de deux chapitres, comme suit:

· Le premier chapitre est un chapitre d'introduction dans lequel nous avons introduit les notions de représentation de connaissances en général, en présentant à titre d'exemple quelques systèmes de gestion ou de représentation de connaissances, tels que les réseaux sémantiques, les logiques de descriptions ainsi que les ontologies, tout en précisant notre objectif du travail, ainsi que la structure de ce mémoire.

· Le deuxième chapitre décrit le concept ontologique, ses composants de base, ses apports, son processus de développement, ainsi que ses méthodologies de construction, ses langages de représentation et ses éditeurs de développement.

· Le troisième chapitre décrit le processus général de la fusion d'ontologies, les problèmes les plus rencontrés lors de la fusion ainsi que les techniques et les outils de fusion, et se termine par une description de quelques travaux réalisés dans ce sens.

· Le quatrième chapitre décrit la Turbine à Vapeur, ses principaux composants, ses principes de fonctionnement ainsi que ses avantages et ses inconvénients. Puis nous présentons le domaine de la maintenance industrielle tout en décrivant la réparation et la maintenance des Turbine à Vapeur, ainsi que son exploitation lors d'un processus de production d'électricité au sein de l'entreprise de SONELGAZ.

· Le cinquième chapitre présente brièvement les notions de base qu'on a choisies pour concevoir et implémenter notre travail, tout en justifiant à chaque fois nos choix, tels que méthodologies de construction, langages de représentation, modèles de conception et outils de développement.

· Le dernier chapitre présente notre approche proposée tout en montrant comment nous avons appliqué et exploité les notions de base décrites dans le chapitre précédent, puis il décrit notre implémentation, tout en présentant les interfaces commentées de notre système.

Le mémoire est clôturé par une conclusion et quelques perspectives, ainsi qu'un résumé de tout ce travail.

II.1. Introduction

Dans un domaine donné, les connaissances initiales se développent au fur et à mesure des acquis de son environnement à partir de différentes sources d'informations, constituant ainsi une masse importante de connaissances pour pouvoir bien refléter le segment de la réalité qu'elle tente de formaliser. Dans le but de pouvoir manipuler ou traiter ce gros volume de bases de connaissances, la gestion de connaissances constitue une problématique primordiale pour contourner le problème de la difficulté de capitalisation de connaissances pertinentes causé par la grande quantité d'informations échangées, d'où la nécessité d'une instrumentation de la gestion de connaissances.

Dans notre travail, la gestion de connaissances est assurée par l'utilisation d'ontologies. Les ontologies deviennent de plus en plus des modèles de représentation et de stockage d'informations très efficaces, facilitant le traitement et la gestion de connaissances à travers les techniques de l'IA, et en offrant le potentiel d'assemblage d'une grande quantité d'informations appartenant à différentes sources d'informations ou de données à travers ce qu'on appelle « La fusion d'ontologies ». Préalablement, chaque source de données (BDD) peut faire l'objet d'une construction ontologique.
« Les ontologies permettent une formalisation informatique des connaissances. Cette représentation peut être cognitivement sémantique (ontologies destinées à être exploitées par l'utilisateur), computationnellement sémantique (ontologies destinées à être exploitées par la machine) ou une combinaison des deux » [AUB 07].

L'exploitation simultanée de plusieurs ontologies différentes et indépendantes, construites par différents ontologistes et pendant différents intervalles de temps, mais modélisant le même domaine de connaissances, pose un problème de discordance entre ces ontologies et devient une voie de recherche de plus en plus abordée en IA.

Les solutions de Fusion d'Ontologies, qui vont être étudiées en détail dans le chapitre suivant, peuvent être envisagées pour résoudre le problème de l'hétérogénéité lors de l'évolution des ontologies.

Dans notre travail, nous considérons les ontologies comme un outil ou une instrumentation pour la gestion de connaissances. Le but de ce chapitre est de présenter un état de l'art sur le concept ontologique. Il s'articulera autour de quatre parties: D'abord, nous présentons la notion d'une ontologie, ses composants, ses

objectifs, et ses critères de classification, puis nous abordons le processus d'ingénierie ontologique, ses principes de base et son cycle de développement, ensuite nous entamons les langages de représentation, les méthodologies de développement ainsi que les éditeurs de construction d'ontologies. Après cela nous exposons le problème d'hétérogénéité lors de l'évolution et la réutilisation en plus d'autres opérations pouvant être appliquées sur les ontologies. Nous terminons par une conclusion.

II.2. Qu'est ce qu'une ontologie ?

Dans la dernière décennie nous constatons une inspiration très intelligente d'une notion philosophique très ancienne vers le domaine de l'intelligence Artificielle et de l'informatique en général, et qui a rapidement constitué une voie de recherche très importante dans plusieurs communautés et surtout celles basées sur l'ingénierie et la gestion de connaissances ainsi que la réutilisation et l'ingénierie des systèmes. Ce potentiel est justifié par sa facilité d'échange d'informations entre ces différents systèmes permettant ainsi la gestion des connaissances partagées et communes à un domaine particulier. Il s'agit bien de la notion d'ontologie.

De ce fait pour pouvoir différencier entre les concepts de ces deux domaines, la convention dit que la notion « **O**ntologie », (avec un O majuscule) soit attribuée au domaine issu de la philosophie, et « ontologie », (avec un o minuscule) aux autres.

II.2.1 L'Ontologie en philosophie

En philosophie, l'étymologie grecque ***« ontologia »*** est composé comme suit: ***onto*** veut dire l'***être*** et ***logia*** veut dire ***science***, et ainsi le terme ***«Ontologie»*** veut dire ***«science de l'être»***. Si on veut considérer sa signification linguistique, on peut retirer la définition du dictionnaire ROBERT: *« La partie de la métaphysique qui s'applique à l'être en tant qu'être indépendamment de ses déterminations particulières ».* Dans le domaine de la philosophie antique, Aristote attribuait la notion d'Ontologie au métaphysique (philosophie première), où il représentait l'être et les différentes situations à l'envers de ses diverses actions et réactions sous forme de plusieurs catégories tel que la substance, la quantité, la qualité, etc. Un schéma illustratif de cette ontologie tiré de [IZZ 06] est présenté en Figure 1:

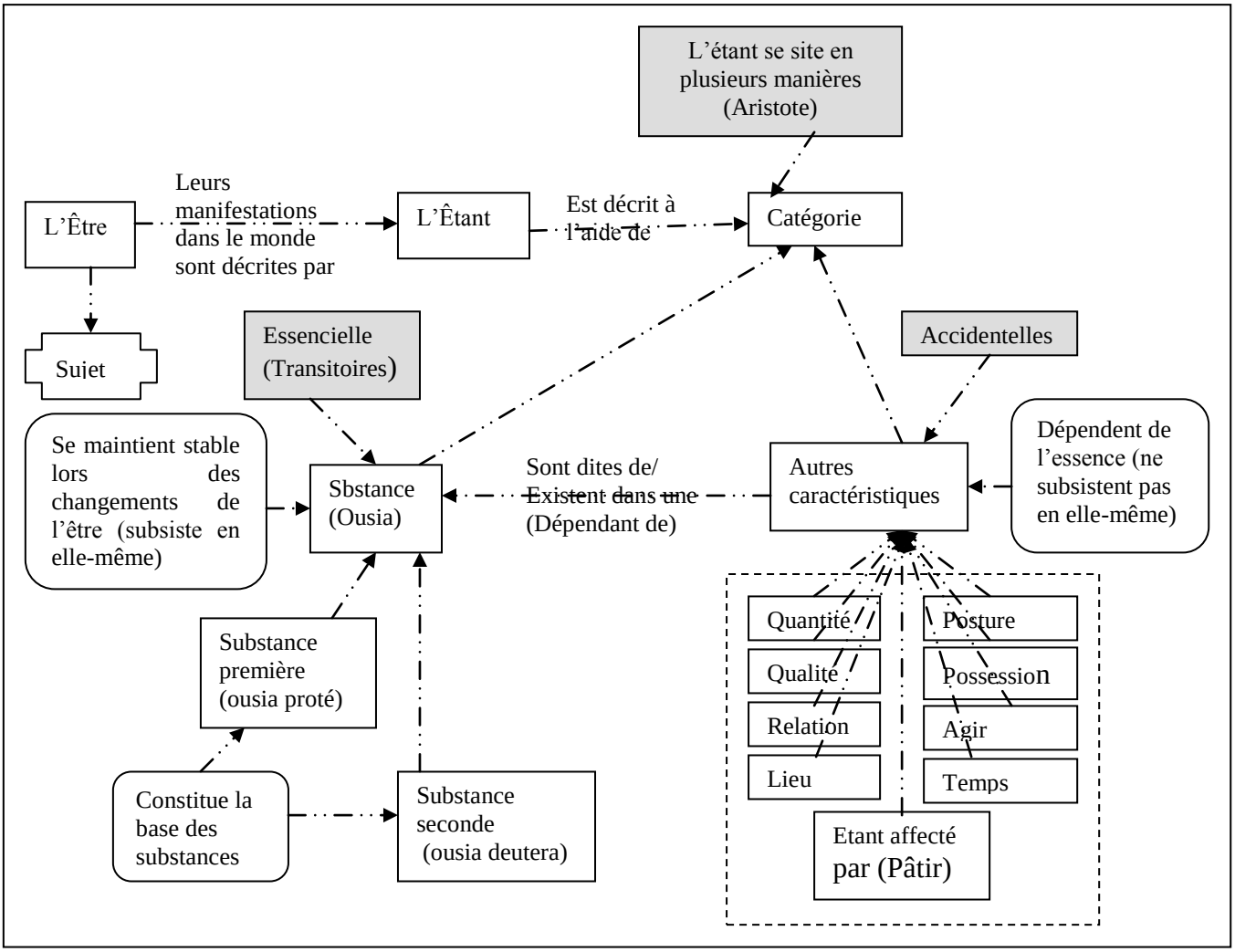


Figure II.1: L'Ontologie de l'être selon Aristote.

II.2.2 L'ontologie en IA

En informatique et en particulier en Intelligence Artificielle et Sciences cognitives, la notion d'ontologie est apparue pour la première fois vers les années 90, où John McCarthy a fait une correspondance entre les concepts de base de l'Ontologie (philosophique) et l'activité de construire des théories logiques de systèmes de l'IA [AUB 07], puis plusieurs acceptions et points de vue ont été proposés par les différents auteurs du domaine.Nous les présentons dans un ordre chronologique dans la section suivante:

D'abord, Neeches et al ont défini l'ontologie en 1991 par: *« An ontology defines the basic terms and relations to define extensions of the vocabulary »*.[NEE et al, 91]. En 1993, Tom Gruber a suggéré la définition la plus succincte dans la littérature: *«Une ontologie est une spécification explicite d'une conceptualisation », « An ontology is*

an explicit specification of a conceptualization ». [GRU 93]. Puis Guarino l'avait définie en 1995 par: «*Une ontologie est une théorie logique qui permet une spécification explicite et partielle d'une conceptualisation*», [GUA 95]. Ensuite, l'ontologie a été connue selon Borst en 1997 comme étant: *« les ontologies se définissent comme une spécification formelle d'une conceptualisation partagée », « An ontology is a shared specification of a shared conceptualization »,* [BOR 97], et lors de cette même période elle a été définie par Swartout comme étant: *"une ontologie est un ensemble de termes structurés de façon hiérarchique, conçu afin de décrire un domaine et qui peut servir de charpente à une base de connaissances »,* [SWA 97]. Alors qu'en 1998, elle a été décrie par Studer à travers la suggestion: "*Une ontologie peut prendre différentes formes, mais elle inclura nécessairement un vocabulaire de termes et une spécification de leur signification. Cette dernière inclut des définitions et une indication de la façon dont les concepts sont reliés entre eux* ». [STU 98]. Après cela, Gomez-Perez l'avait définie en 1999 par: *« une ontologie apporte les moyens pour décrire explicitement la conceptualisation sous-jacente aux connaissances représentées dans une base de connaissances ».* [GOM 99]. En plus, Brodeur à suggéré en 2004 qu' *« Elle peut prendre la forme d'un thésaurus, réseau sémantique, taxonomie, modèle conceptuel, répertoire de données, etc », [BRO 04]*. Enfin, une synthèse de ces définitions donnée par Amrouch et al en 2009 déclare qu': « Une ontologie est une structure formelle, qui peut être graphique, où les nœuds représentent les concepts ou les classes, et les arcs représentent les relations, modélisant ainsi explicitement les connaissances d'un domaine particulier de manière à faciliter la recherche, l'extraction, l'intégration et le partage d'informations entre les différents systèmes ou individus au sein de ce domaine » [AMR et al, 09].

Bien que ces définitions offrent des points de vue différents mais complémentaires sur le concept ontologique, elles focalisent sur des critères communs qui représentent les caractères de base d'une ontologie, et qui sont:

- ✓ Formelle: Signifie que l'ontologie est exprimable à l'aide d'une logique pouvant être traitée par une machine (machinable).
- ✓ Spécification explicite: Signifie que les concepts, relations, fonctions, instances et axiomes d'une ontologie, sont définies d'une manière déclarative.

- ✓ Partagée: Signifie que l'ontologie modélise les connaissances partagées entre les différents individus d'une communauté d'un domaine particulier.

II.3. Qu'est ce qu'une ontologie n'est pas ?

Dans [MIZ 03], Mizoguichi propose, une distinction entre la notion d'ontologie et d'autres concepts comme suit:

Une ontologie est plus qu'une simple liste de termes, et ceci pour deux raisons: D'une part, bien qu'une ontologie fournisse un vocabulaire commun pour une certaine activité, au même titre qu'une liste de termes, ce qui distingue ces deux concepts est la présence d'une structure (en particulier par des liens is-a) dans l'ontologie. D'autre part, il faut bien distinguer la notion de nom de concept dans une ontologie (un nom de variable informatique), du terme qui lui est associé (un mot du vocabulaire d'une langue).

Une ontologie est plus qu'une hiérarchie de concepts : En effet, bien qu'une ontologie comporte une représentation arborescente des concepts (graphe is-a), toute hiérarchie de concepts n'est pas suffisante en tant qu'ontologie.

Une ontologie n'est pas un formalisme de représentation des connaissances: Contrairement, par exemple, à un réseau sémantique, une ontologie n'est pas un formalisme de représentation de connaissances, tel que la manière de représentation de ses liens « is-a » n'a pas d'importance.

II.4. Les composants d'une ontologie

A l'instar de tous les autres formalismes de représentation de connaissances, tels que les arbres de décision, les réseaux neuronaux, les réseaux sémantiques, etc, les ontologies sont développées en se basant sur un certain nombre d'unités constructives pour décrire et représenter les connaissances de manière formelle. Ces unités ontologiques constructives sont spécifiées par Gomez-Perez dans [GOM 99], basé sur la définition de Gruber en 1993, à savoir:

1. Les concepts: Ou encore les classes ou les termes de l'ontologie, ils peuvent représenter, généralement sous une forme hiérarchique, des objets physiques bien que d'autres logiques ou morales qui sont extrait après la conceptualisation du domaine de

problème en fonction des objectifs à atteindre et de l'application qui va utiliser l'ontologie.

«Un concept est n'importe quelle chose ou sujet de laquelle on peut dire quelque chose, et peut ainsi aussi bien être la description d'une tâche, d'une fonction, d'une action, d'une stratégie, d'un raisonnement, etc ». [GOM 99].

2. Les relations: Relient les différents concepts de l'ontologie, décrivant ainsi les divers interactions entre eux, ce qui reflète alors une grande partie de la sémantique de l'ontologie à travers les étiquetages (Nom de relation) attribués à ces associations tels que la subsomption (spécification ou généralisation) (*is-a*), l'instanciation (*instance-of*), la composition (*part-of*), l'affectation ou l'attribution (*associated-to*), etc.

3. Les fonctions: Sont des cas particuliers de relations, où le nième élément, l'extrant, est défini en fonction des *n-1* éléments précédents, intrants, et formellement une fonction est ainsi défini:

$$F : C_1 \times C_2 \times K\ K\ K \times C_n$$
$$E_n\ \alpha\ \ f(E_1, E_2, K\ K\ K\ E_{n-1})$$

4. Les axiomes: Représentent des propositions acceptées comme toujours vraies à propos des abstractions du domaine de problème, et qui jouent un rôle primordial dans le processus de déduction ou d'inférence vu qu'ils constituent des points de départ pour certains mécanismes d'inférences (le chaînage avant), ou des points d'arrêts pour d'autres (le chaînage arrière) motivant ainsi le raisonnement du moteur d'inférence basé sur l'ontologie.

5. Les instances: Appelées aussi les individus pour représenter les éléments singuliers des concepts dans le domaine du problème modélisé.

II.5. L'apport des ontologies (pourquoi utiliser les ontologies ?)

Dans cette section, nous allons discuter pourquoi les ontologies sont elles développées et utilisées, et quel est le rôle qu'elles peuvent jouer ? Une ontologie offre une aide permettant la mise en œuvre et l'exploitation de diverses techniques et mécanismes nécessaires pour la gestion de connaissances et l'échange d'informations dans les applications logicielles et systèmes d'informations. Dans [GRÜ et al, 96], Gruninger et al déclarent que les ontologies sont développées pour offrir une aide dans

au moins trois domaines d'application à savoir la communication entre les êtres humains, l'interopérabilité entre les Systèmes d'Informations hétérogènes, ainsi que la réutilisabilité et le partage d'informations, et nous pouvons donc résumer le rôle des ontologies dans le domaine des systèmes d'informations comme suit:

- La communication: Une ontologie est un modèle standard qui décrit et spécifie explicitement son domaine, constituant ainsi un espace partageable entre les (sous) systèmes et/ou individus, où ils partagent leurs points de vue, compréhension et perspectives, éliminant ainsi tout risques de confusion ou d'incompréhension, et favorisant tout effort de collaboration et de partenariat.
- L'interopérabilité entre les systèmes d'informations: Elle dépend principalement de la flexibilité de communication entre eux, car leurs actions et réactions résultent des messages envoyés et reçus. En jouant le rôle d'un format d'échange, où elle répertorie tous les concepts qui doivent être échangés par les applications. L'ontologie permet à des systèmes d'informations de coopérer.
- L'aide à la spécification et à la conceptualisation de systèmes: Vue son caractère de partageabilité, l'ontologie peut être un concept réutilisable et/ou partageable par plusieurs (sous) systèmes, aidant ainsi à l'acquisition d'informations, l'analyse de données et la spécification de besoins. En plus sa structuration plus lisible et compréhensible de la documentation d'un logiciel. Ensuite, elle peut structurer la documentation d'un logiciel, ce qui permet d'éviter tout risque d'ambiguïté ou de confusion lors de la spécification de besoins. Finalement elle soutient l'automatisation du processus de vérification de cohérence réduisant ainsi les coûts de maintenance.

II.6. Critères de classification des ontologies

Dans la littérature les ontologies peuvent être classifiées en fonction de plusieurs critères. Dans cette section nous allons présenter les critères les plus cités, en l'occurrence: l'objet de la conceptualisation, le niveau de granularité, le niveau de complétude, le niveau de formalisme de représentation, et le poids de l'ontologie, nous illustrons dans la figure ci-dessous une catégorisation des ontologies selon les cinq critères ainsi cités:

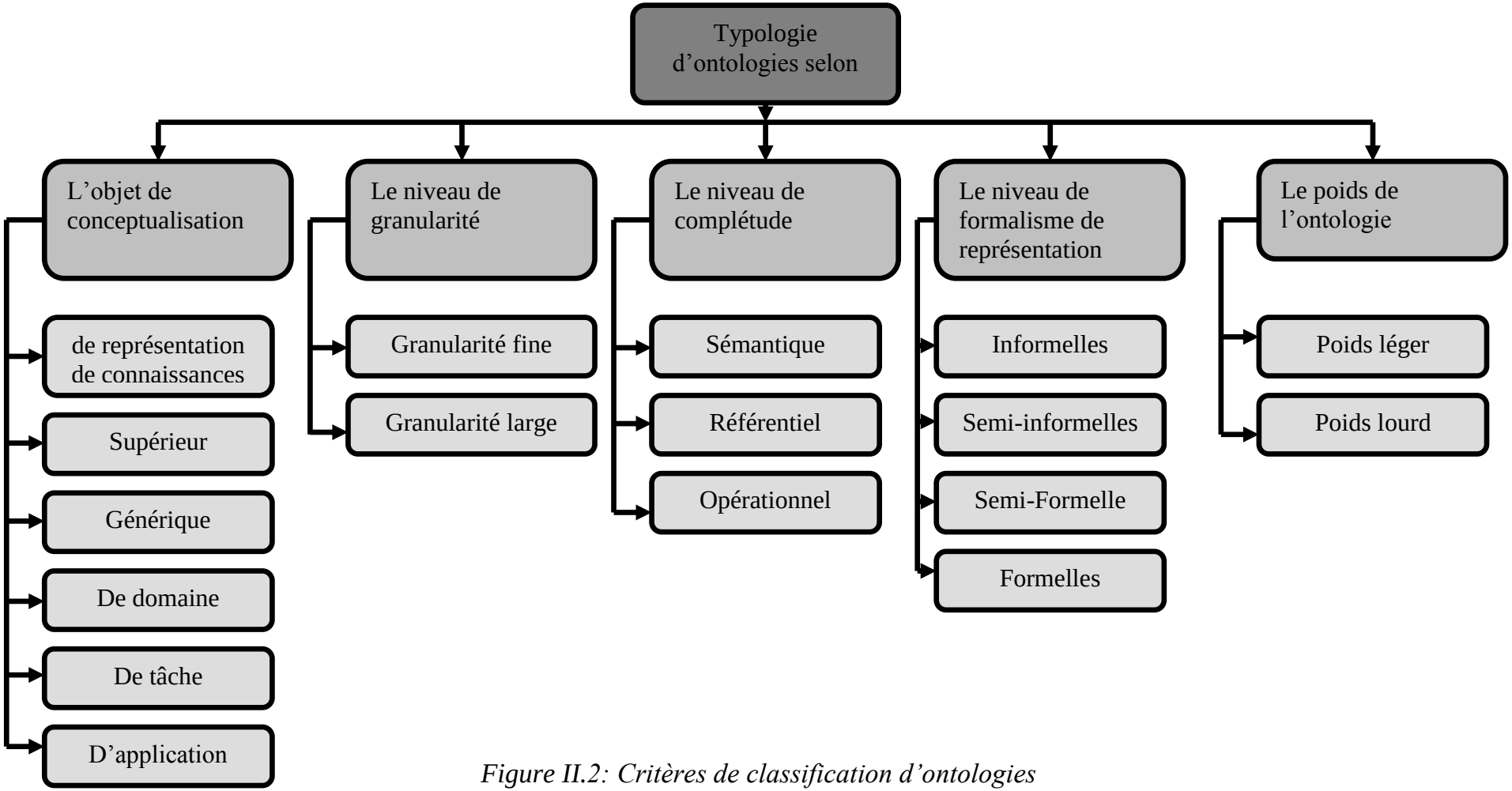


Figure II.2: Critères de classification d'ontologies

1. Classification selon l'objet de conceptualisation: Dans [KHA 07], certains spécialistes du domaine ont classifié les ontologies en fonction de leur objet de conceptualisation en six catégories, ces types d'ontologies sont ainsi décrits par les auteurs ci-indiqués:

a. Ontologie de représentation des connaissances: Ce type d'ontologies regroupe les concepts impliqués dans la formalisation des connaissances. Un exemple est l'ontologie de Frame qui intègre les primitives de représentation des langages à base de frames: classes, instances, facets, propriétés/slots, relations, restrictions, etc, [GOM 99].

b. Ontologie supérieure ou de Haut niveau: Cette ontologie est une ontologie générale. Son sujet est l'étude des catégories des choses qui existent dans le monde, soit les concepts de haute abstraction tels que: les entités, les événements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés. L'ontologie de haut niveau est fondée sur: la théorie de l'identité, la méréologie (theory of whole and parts role) et la théorie de la dépendance, [GUA 95], [SOW 95]. Un schéma illustratif de l'ontologie supérieur définie par Sowa en 95 est donné en figure 03.

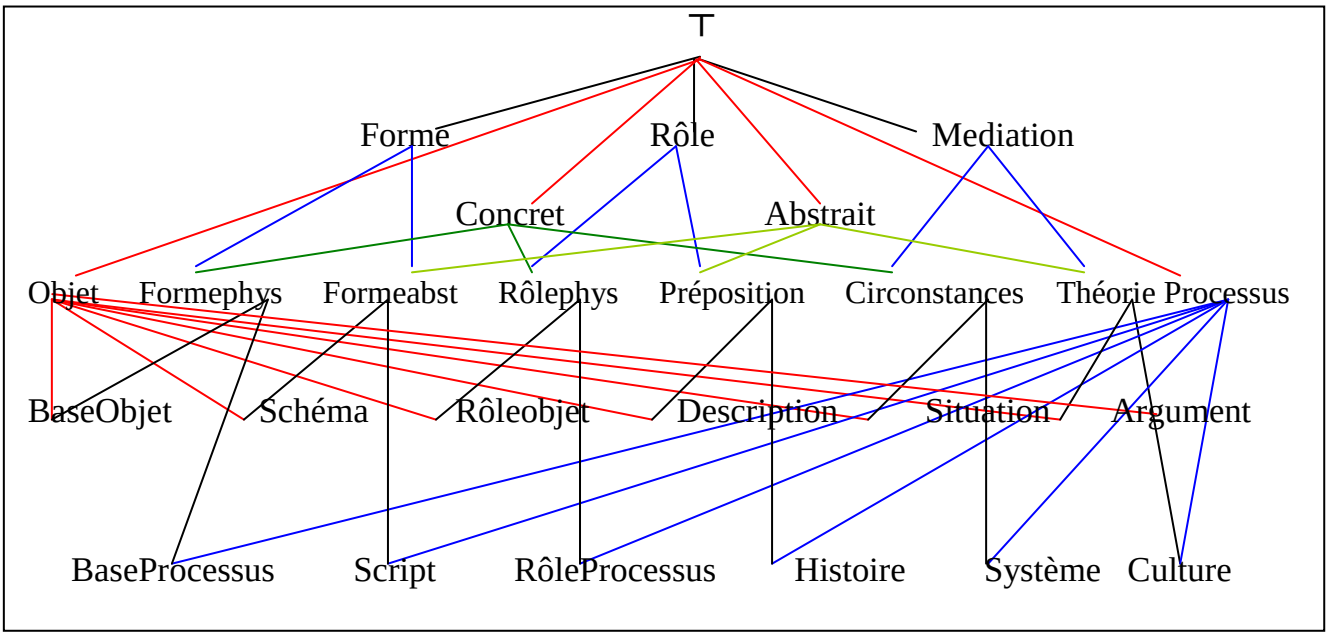


Figure II.3. L'ontologie (de haut niveau) des catégories des choses.

c. Ontologie Générique: Cette ontologie aussi appelée, méta-ontologies, véhicule des connaissances génériques moins abstraites que celles véhiculées par l'ontologie de haut niveau, mais assez générales néanmoins pour être réutilisées à

travers différents domaines. Elle peut adresser des connaissances factuelles (Generic domain ontoloy) ou encore des connaissances visant à résoudre des problèmes génériques (connaissances procédurales) appartenant et réutilisables à travers différents domaines (Generic task ontoloy). Deux exemples de ce type d'ontologies sont:

1) l'ontologie méréologique, [BOR 97], contenant des relations, *Partie-de* et

2) l'ontologie topologique contenant des relations, *Associé-à*. [GOM 99].

d. Ontologie du Domaine: (Définie par Mizoguchi en 2000), cette ontologie régit un ensemble de vocabulaires et de concepts qui décrit un domaine d'application ou monde cible. Elle permet de créer des modèles d'objets du monde cible. L'ontologie du domaine est une méta-description d'une représentation des connaissances, c'est-à-dire une sorte de méta-modèle de connaissance dont les concepts et propriétés sont de type déclaratif. La plupart des ontologies existantes sont des ontologies du domaine, selon Mizoguchi, elle caractérise la connaissance du domaine où la tâche est réalisée.

e. Ontologie de Tâches: Ce type d'ontologies est utilisé pour conceptualiser des tâches spécifiques dans les systèmes, telles que les tâches de diagnostic, de planification, de conception, de configuration, de tutorat, soit tout ce qui concerne la résolution de problèmes. Elle régit un ensemble de vocabulaires et de concepts qui décrivent une structure de résolution des problèmes inhérente aux tâches et indépendante du domaine, [MIZ 03].

f. Ontologie d'Application: Il s'agit de l'ontologie la plus spécifique où les concepts correspondent souvent aux rôles joués par les entités du domaine tout en exécutant une certaine activité, [MAE 02].

2. Classification selon le niveau de granularité: En fonction du niveau de granularité (niveau de détail), les ontologies peuvent être dites de fine granularité ou de large granularité:

a. Les ontologies de fine granularité: Ce sont des ontologies à un vocabulaire très détaillé assurant ainsi une description très fine du domaine de problème. Il est à noter que plus le niveau de granularité est fine plus les concepts modélisés vont correspondre à des notions spécifiques.

b. Les ontologies de large granularité: Ce sont des ontologies à un vocabulaire moins détaillé offrant une description un peu générale du domaine de problème, où des accords peuvent être préalablement existant concernant la conceptualisation du domaine. Un bon exemple d'ontologie de large granularité est l'ontologie de haut niveau (car ses concepts sont raffinés subséquemment dans d'autres ontologies de domaine ou d'application).

3. Classification selon le niveau de complétude: Dans [BAC 00], Bachimont propose une typologie en fonction de degré de complétude sur les trois niveaux suivant:

a. Niveau sémantique: Tous les concepts (caractérisés par un terme/libellé) doivent respecter les quatre principes différentiels: 1) Communauté avec l'ancêtre; 2) Différence (spécification) par rapport à l'ancêtre; 3) Communauté avec les concepts frères (situés au même niveau); 4) Différence par rapport aux concepts frères (sinon il n'aurait pas lieu de le définir). Ces principes correspondent à ***l'engagement sémantique*** qui assure que chaque concept aura un sens univoque et non contextuel associé. Deux concepts sémantiques sont identiques si l'interprétation du terme/libellé à travers les quatre principes différentiels aboutit à un sens équivalent.

b. Niveau référentiel: Outre les caractéristiques énoncées au niveau précédent, les concepts référentiels (ou formels) se caractérisent par un terme/libellé dont la sémantique est définie par une extension d'objets. L'engagement ontologique spécifie les objets du domaine qui peuvent être associés au concept, conformément à sa signification formelle. Deux concepts formels seront identiques s'ils possèdent la même extension (ex: les concepts d'étoile du matin et d'étoile du soir associés à Vénus).

c. Niveau Opérationnel: Outre les caractéristiques énoncées au niveau précédent, les concepts du niveau opérationnel ou computationnel sont caractérisés par les opérations qu'il est possible de leur appliquer pour générer des inférences (engagement computationnel). Deux concepts opérationnels sont identiques s'ils possèdent le même potentiel d'inférence.

4. Classification selon le niveau de formalisme de représentation: Selon le niveau de formalité du langage de représentation utilisé lors du développement de

l'ontologie, dans [GRÜ et al, 96], Uschold et Gruninger proposent, une classification en 04 types qui sont: les ontologies informelles, semi informelles, semi formelles et formelles, tels que:

a. Ontologies informelles: Qui sont des ontologies opérationnalisées dans un langage naturel, ce qui correspond à ce qui est communément appelé la sémantique ouverte;

b. Ontologies semi-informelles: Ce sont des ontologies qui sont décrites à l'aide d'un langage naturel structuré et limité;

c. Ontologies semi-formelles: Qui sont des ontologies qui utilisent un langage Artificiel défini formellement;

d. Ontologies formelles: Qui sont des ontologies qui se basent sur l'utilisation d'un langage artificiel contenant une sémantique formelle telle que les logiques de description.

5. Classification selon le poids de l'ontologie: Dans [MIZ 03], Mizoguichi propose une classification des ontologies selon leur poids en deux types:

a. Les ontologies de poids léger décrivent simplement l'hiérarchie de concepts et de relations entre ces concepts.

b. Les ontologies de poids lourd décrivent les propriétés avancées de ces concepts permettant de faire des inférences et des déductions.

II.7. Catégorie d'ontologies

Dans [JEA 07], Jean a mis en œuvre une taxinomie des ontologies selon trois catégories comme suit:

II.7.1 Les Ontologies Conceptuelles Canoniques (OCC)

Définir un modèle conceptuel ou un modèle d'échange consiste donc à définir un vocabulaire canonique dans lequel chaque information dans le domaine cible est capturée de manière unique, sans définir le moindre constructeur de synonymie. S. Jean appelle Ontologies Conceptuelles Canoniques (OCC), les ontologies dont les définitions ne contiennent aucune redondance. Dans les OCC, chaque concept du domaine est décrit d'une seule façon, en utilisant une description qui ne peut inclure que des conditions nécessaires. En conséquence, les OCC ne comportent que des

concepts primitifs. Nous verrons ultérieurement que les OCC sont particulièrement importantes dans le contexte de la gestion ou de l'échange de données.

II.7.2 Les Ontologies Conceptuelles Non Canoniques (OCNC)

Dans le but d'offrir une plus grande souplesse d'accès aux utilisateurs, un concepteur d'une base de données peut définir des vues. Ces concepts définis sont spécifiés en utilisant l'opérateur CREATE VIEW sur la base des concepts primitifs qui constituent le schéma de la base de données.

Quels que soient les constructeurs offerts pour exprimer des équivalences conceptuelles, nous appelons Ontologies Conceptuelles Non Canoniques (OCNC), les ontologies qui contiennent non seulement des concepts primitifs, mais aussi des concepts définis. Les OCNC sont particulièrement utiles lorsqu'elles sont utilisées comme schéma global de requête. En effet, la redondance qu'elles introduisent permet d'augmenter le nombre de concepts en termes desquels il est possible d'exprimer une requête donnée.

Les constructeurs OCNC sont également très utiles pour définir des mappings entre différentes ontologies.

II.7.3 Ontologies Linguistiques (OL)

La documentation d'un modèle conceptuel est largement, ou totalement, exprimée dans une langue naturelle.

S. Jean appelle Ontologies Linguistiques (OL), les ontologies qui définissent l'ensemble des termes qui apparaissent dans la description langagière d'un domaine. Dans cette catégorie d'ontologies, outre les relations entre concepts représentés par des termes (par exemple, subsumée_par pour la relation de subsomption), des relations entre les termes (par exemple, la synonymie ou l'homonymie) doivent être également définies. Les relations entre les termes étant souvent fortement contextuelles, les inférences automatiques basées sur ces ontologies nécessitent, en général, la supervision d'un expert. Les OL aident à reconnaître des similarités conceptuelles entre des phrases même si différents termes sont utilisés. Néanmoins, puisque la signification des termes est contextuelle et que les relations entre termes sont

approximatives, de fausses similarités peuvent être produites et les résultats ne peuvent jamais être considérés comme fiables.

II.8 L'ingénierie ontologique:

II.8.1 Les principes de base de construction d'ontologies

Il existe un ensemble de critères et de principes pour mieux guider le développement d'ontologies. Nous résumons ici certains de ces critères conceptuels qui sont proposés par plusieurs auteurs, notamment par [GRU 93], et [BAC 01].

Cinq critères génériques guidant l'ontologisation ont été proposés par [GRU 93], et on peut les résumer ainsi:

1. La clarté et l'objectivité: Les termes doivent être définis de façon claire et objective et disposant d'une documentation en langage naturel.

2. La cohérence: Les axiomes doivent être consistant à fin de permettre à l'ontologie de réaliser des inférences cohérentes aux définitions.

3. Extensibilité monotone maximale: C'est-à-dire, la possibilité d'ajouter de nouveaux termes et d'étendre l'ontologie sans avoir à réviser ou à modifier les anciennes.

4. Minimalité des postulats d'encodage assurant une bonne portabilité.

5. Engagement ontologique minimale: Il s'agit de donner aux utilisateurs d'ontologie une marge de liberté plus grande pour spécifier et instancier l'ontologie selon leurs besoins indépendamment des suppositions concernant le monde modélisé.

Et quatre critères proposés par [BAC 01], et nous les résumons ainsi:

1. Le principe de communauté avec le père, ou principe de similarité: C'est-à-dire que le concept hérite l'intention de son concept père.

2. Le principe de différence avec le père ou le principe de différence: C'est-à-dire que l'intention d'un concept doit être différente de celle de son père.

3. Le principe de communauté avec la fratrie ou principe de sémantique unique: C'est-à-dire qu'une propriété est commune entre les concepts frères ayant le même père mais où elle est exprimée différemment pour chaque frère.

4. Le principe de différence avec la fratrie ou principe d'opposition: C'est-à-dire que les frères doivent tous être incompatible, sinon il n'y aurait pas un seul sens de toutes les spécifications.

II.8.2 Processus de construction

A l'instar des grandes applications développées en GL, les ontologies sont considérées comme des objets techniques évolutifs ayant un cycle de vie, où le processus de développement passe en général par les mêmes étapes de développement d'un logiciel qui à partir de données brutes du domaine de problème exprimées généralement en langages naturel (informel), développe le modèle conceptuel dans une étape de conceptualisation, puis génère l'ontologie formel dans une étape d'ontologisation qui va l'exprimer dans un langage de représentation formel et opérationnel lors de l'étape de l'opérationnalisation.

Il est à noter que le processus de construction ontologique n'est pas purement linéaire car la réalisation d'une étape peut mettre en cause les résultats des étapes précédentes, d'où la nécessité de retours en arrière pour supprimer, ajouter ou modifier les modules antérieurs. La Figure 4 représente un schéma général du processus de construction d'ontologie.

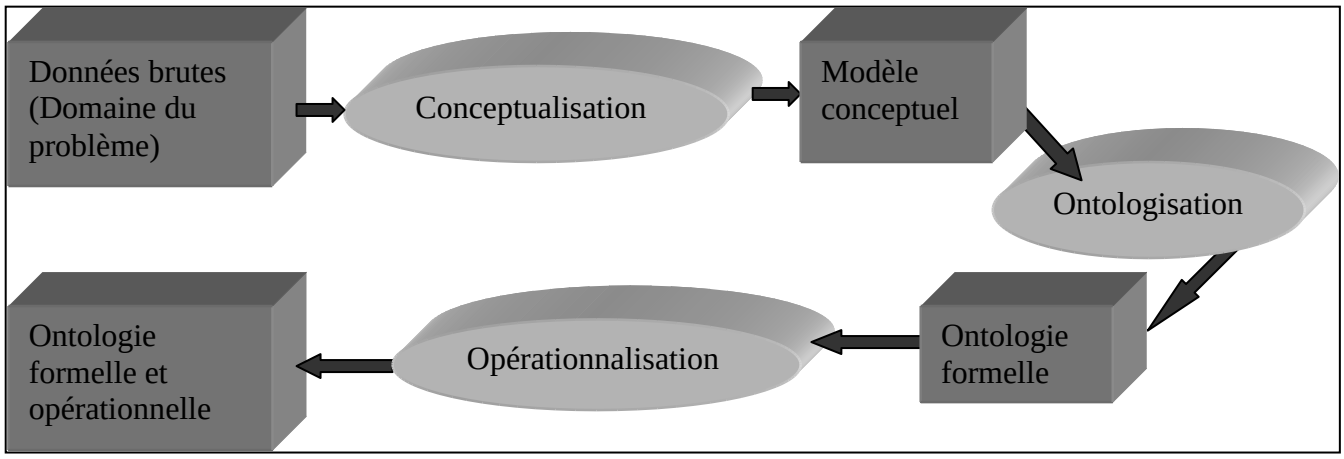


Figure II.4:Processus de construction d'ontologie

II.8.2.1. La conceptualisation:

Elle consiste à partir d'un corpus (données brutes) à identifier les connaissances du domaine en dégageant les concepts et les relations entre ces concepts décrivant informellement ses entités cognitives. Le processus de la conceptualisation doit faire la distinction entre les connaissances spécifiques au domaine et celle ne servant qu'à l'expression des connaissances de domaine. En plus, si le développement ontologique prévoit l'intégration d'autres ontologies, il ne faut pas prendre en compte les connaissances déjà présentées dans ces ontologies dans la conceptualisation de l'ontologie en question. Puis la nature conceptuelle des termes désignant les entités du domaine de connaissances (tels que concepts, relations, axiomes, propriétés, etc) doit être précisée. L'acquisition des connaissances du domaine demande l'intervention d'un expert (ou un groupe d'expert) du domaine assisté par l'ingénieur de connaissances qui applique son expertise des techniques de représentation de connaissance pour les rendre machinables, favorisant ainsi la structuration de connaissances pour aboutir à un modèle conceptuel consistant des concepts, relations, axiomes, propriétés, slots, facets, etc.

Une des tâches les plus délicates dans le processus de conceptualisation est d'identifier les connaissances qui sont implicitement utilisées dans le domaine, mais qui n'ont pas été exprimé ni dans le corpus analysé, ni par les experts du domaine car, par exemple, elles semblent complètement évidentes, d'où vient l'importance des rôles des utilisateurs finaux dans le processus de la construction ontologique, et spécialement en phase de la conceptualisation pour la mise en évidence de ces connaissances implicites à travers l'utilisation de l'ontologie, lors d'une phase de test opérationnel et/ou de test de questions de compétence, encore une preuve de ce qui a été cité précédemment de ce que le processus de construction n'est pas purement linéaire et là un retours en arrière aura lieu lors de la découverte de connaissances non spécifiées par l'utilisateur final de l'ontologie.

II.8.2.2. L'ontologisation:

Cette étape consiste à structurer et à formaliser autant que possible (partiellement) le modèle conceptuel résultant de la phase précédente pour construire

une ontologie spécifiant la terminologie et la sémantique du domaine à travers un modèle doté d'une sémantique formelle mais pas opérationnelle. Cette formalisation partielle, menée par l'ingénieure de connaissances assisté par l'expert de domaine, facilite sa représentation ultérieure dans un langage complètement formel et opérationnel. En d'autres termes, l'ontologisation est vue comme une traduction dans un certain formalisme de représentation de connaissances exprimée à priori en langage naturel [FUR 04]. A ce stade les connaissances ne sont pas toutes totalement formalisées vue l'impossibilité de lever certaines ambiguïtés et/ou la difficulté ou l'impossibilité de formaliser certaines expressions en langage naturel donnant ainsi le caractère semi formel à cette ontologie, ce qui empêche par exemple son intégration dans un système à base de connaissances.

II.8.2.3. L'opérationnalisation:

Elle consiste à transcrire l'ontologie (semi) formelle, résultante de l'étape précédente, dans un langage formel et opérationnel de représentation d'ontologies, où les termes ontologiques (concepts et relations) sont enrichies par les opérations qu'on peut leur appliquer pour inférer ou déduire de nouvelles connaissances permettant ainsi, en plus des interprétations possibles menées par la représentation formelle de spécifier la façon par laquelle ces connaissances sont exploitées pour inférer ou déduire de nouvelles connaissances. En d'autres termes permettant de spécifier comment ces connaissances opèrent sur d'autres pour faire un raisonnement ce qui rend l'ontologie intégrable et utilisable par un SBC. Il est à noter que le processus d'opérationnalisation est mené par l'ingénieur de connaissances.

Nous pouvons donc résumer la nature structurelle des représentations de connaissances résultantes des trois phases du processus de construction ontologique comme suit:

La conceptualisation (informelle ou semi formelle): En langage naturel ou semi structuré, ne disposant pas de sémantique claire ou tout au moins d'une sémantique fixée à priori.

L'ontologisation (formelle): Spécifie la syntaxe et la sémantique.

L'opérationnalisation (formelle et opérationnelle): Dotée de services inférentiels permettant de mettre en œuvre des raisonnements.

II.8.3 Cycle de vie d'une ontologie

Comme tout composant logiciel, une ontologie est destinée à être réutilisée et parfois partagée entre plusieurs applications (et/ou individus d'une communauté) satisfaisant des besoins et des fonctions opérationnels différents. Alors pour construire une ontologie, on peut adopter pour les mêmes techniques et principes que ceux appliqués en GL lors du développement des Systèmes d'Information.

Selon [IZZ 06], les activités liées aux ontologies sont d'une part des activités de développement (Spécification, Formalisation et Implémentation), et d'autre part des activités de gestion de projet (Organisation, Planification, Estimation, Contrôle et Assurance de qualité), s'y ajoute un certain nombre d'activités transversales de support (Evaluation, Documentation, gestion de la configuration). Plusieurs spécifications différentes de cycles de vie d'ontologies inspirées du GL ont été proposées dans la littérature, mais qui se ressemblent toutes dans les phases principales de cycle de vie.

Un cycle de vie qui constitue les phases les plus communes entre les différentes propositions est illustré dans la figure ci-dessous:

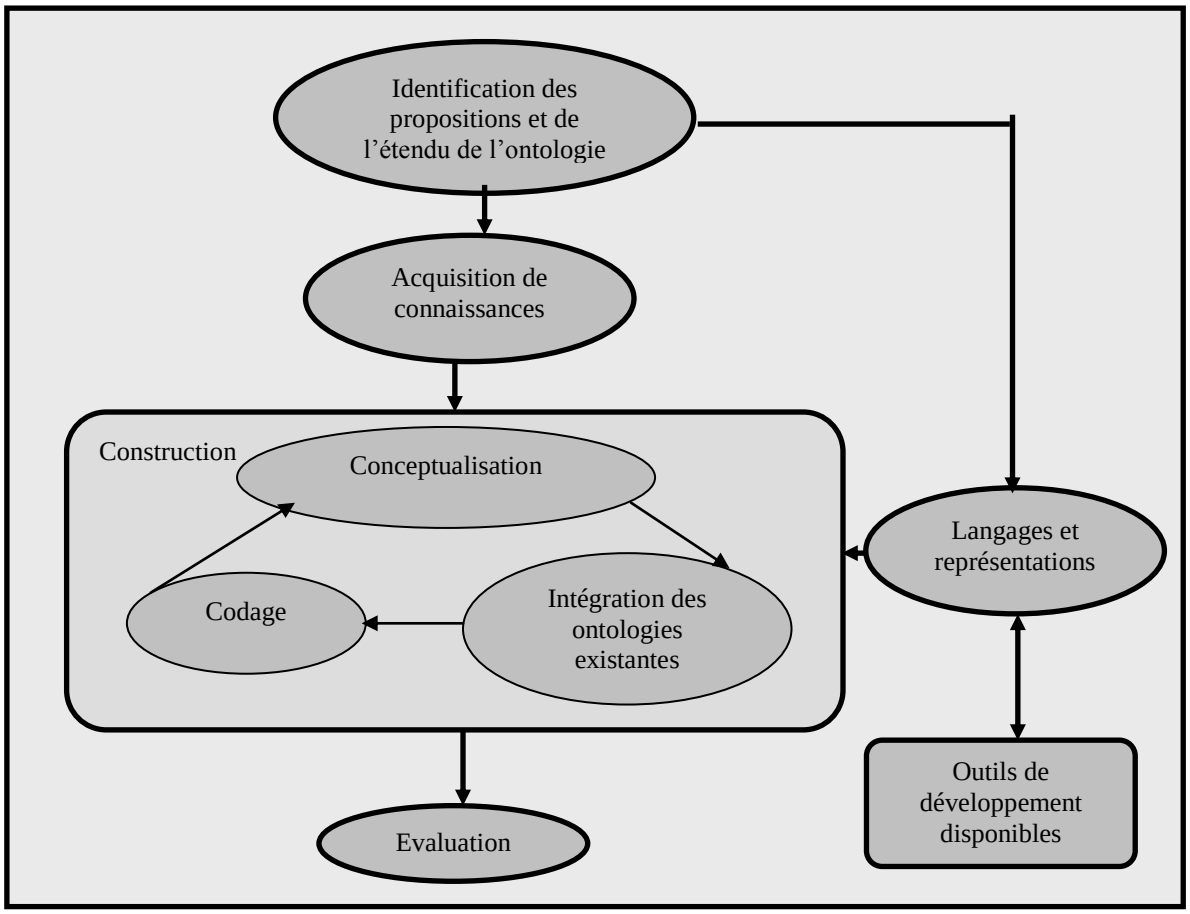


Figure II.5: Cycle de développement d'ontologie.

Nous pouvons décrire les étapes illustrés en Figure 5 comme suit:

- **Spécification:** C'est une étape primordiale pour la conception, l'évaluation et la réutilisation de l'ontologie. Elle consiste à identifier les propositions, les niveaux de granularité et de généralisation concernant les concepts.
- **Acquisition de connaissances:** Elle consiste en l'absorption et la collection de connaissances de domaine à travers l'analyse des scénarios et documents tel que les BDDs, les textes standardisés, les papiers scientifiques, en plus d'autres ontologies, ou en organisant des interviews avec les experts de domaine qui doivent être accompagnés par un ingénieur de connaissances.
- **La conceptualisation:** Elle consiste en l'identification des concepts clés reflétant le domaine de connaissances, leur propriétés, ainsi que les relations qui les relient puis en la mise en évidence des descriptions de ces concepts, relations et attributs en langage naturel et enfin en la modélisation des connaissances de domaines à travers un modèle conceptuel explicite. Cette

étape doit être menée par l'expert de domaine assisté par l'ingénieur de connaissances.

- **Intégration:** Elle consiste en la combinaison des données (et/ou connaissances) valables à partir de BDDs, et ontologies précédemment construites pour obtenir une ontologies consistantes et plus complète.
- **Codage:** Elle consiste en la représentation du modèle conceptuel dans un langage formel de représentation ontologique tels que la logique de description, les frames, les réseaux sémantiques, etc.
- **Evaluation:** Elle consiste en l'appréciation de compétence et capacité de l'ontologie à satisfaire les besoins de son application. Il s'agit donc d'évaluer l'ontologie du point de vue de la complétude, la consistance, et la minimalité de redondances.
- **Documentation:** Des descriptions et des définitions complètes, formelles ou informelles des assumptions et des exemples sont inévitables pour faciliter et aider à l'utilisation ou la réutilisation des ontologies, car une ontologie qui ne peut être comprise ne peut être utilisée.

Un autre cycle de vie proposé par [FUR 02] pour l'ingénierie d'ontologie dans le domaine du web sémantique qui est itératif et incrémental et comprend: La conception initiale, le raffinement conceptuel, l'évaluation et l'évolution, et est illustré dans la figure ci-dessous:

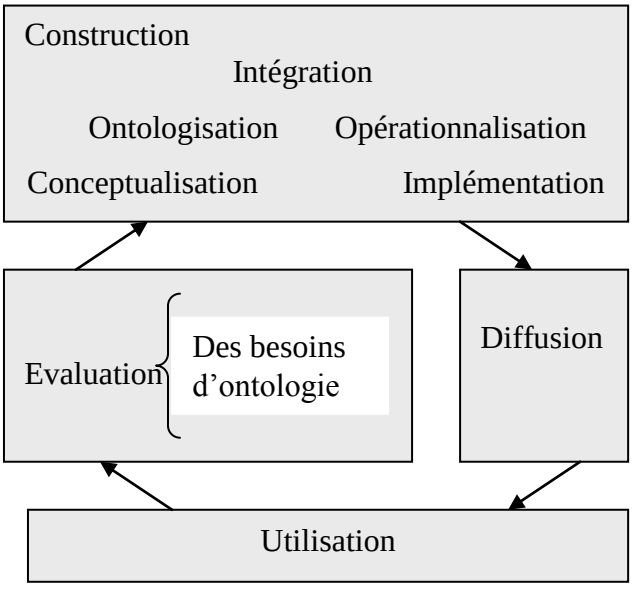


Figure II.6: Cycle de vie d'une ontologie [FUR 02]

Un autre cycle de vie proposé par [DIE et al, 01], comprend: Appréciation initiale des besoins, construction, diffusion, utilisation, puis le cycle peut recommencer, après chaque utilisation significative. L'ontologie et les besoins sont réévalués et l'ontologie peut être étendue et si nécessaire en partie reconstruite, cette proposition est illustrée en figure 07.

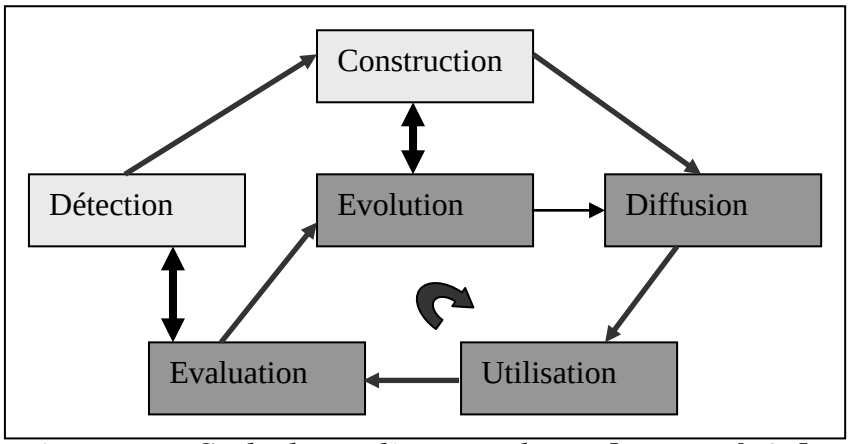


Figure II.7: Cycle de vie d'une ontologie, [DIE et al, 01].

Un autre cycle de vie proposé par [FER et al, 97], comprend principalement la spécification, la conceptualisation, la formalisation, l'intégration, l'implémentation, et la maintenance, et qui est fortement inspiré du modèle en spirale utilisé en GL et permet de développer des ontologies d'une manière prototypique.

Un dernier cycle de vie proposé par [GAN 02], qui résulte de la fusion des deux derniers cycles de vie, dans le cadre du projet CoMMA mené par l'équipe ACACIA de l'INRIA et qui propose une meilleure articulation entre les différentes étapes de construction ontologique.

De tout ce qui précède, on peut conclure qu'il ne s'agit pas d'un cycle de la cascade, mais plutôt du cycle de la spirale, qui est itératif et incrémental, où de nombreux allers-retours sur les étapes précédentes (pas nécessaire le just-avant) sont envisagés.

II.8.4 L'évaluation d'une ontologie

A l'instar des applications en GL, l'évaluation d'une ontologie se fait à travers deux processus différents mais complémentaires à savoir la vérification qui consiste à vérifier qu'on a bien construit l'ontologie relativement à un modèle formel, et la

validation qui consiste à s'assurer qu'on a construit la bonne ontologie relativement au domaine du problème:

II.8.4.1 La vérification:

Elle consiste à vérifier si l'ontologie construite est correcte, c'est à dire qu'elle correspond bien au modèle conceptuel représentant le domaine. Selon [FUR 04], la vérification d'une ontologie repose sur le test de trois grands types de propriétés, en l'occurrence: la conformité, la cohérence, et la minimalité. Dans ce qui suit nous résumons les tâches attendues par chaque type de test:

1. La conformité d'une ontologie à un modèle conceptuel: Comme son nom l'indique, elle consiste à vérifier que la forme de l'ontologie est conforme à la syntaxe imposée par le modèle conceptuel et indépendamment du domaine de connaissances.

2. La cohérence: Elle vise à s'assurer que les connaissances représentés dans l'ontologie ne présentent pas de contradictions logiques (et/ou sémantiques).

Au temps où le test de conformité ne vérifie que la syntaxe, on peut trouver des axiomes qui sont syntaxiquement correctes, mais qui peuvent être logiquement contradictoires. Il est à noter que le test de cohérence est aussi indépendant du domaine de connaissances.

3. La minimalité: Elle consiste à vérifier que l'ontologie a la taille la plus petite que possible dans le sens où elle ne contient pas des connaissances superflues.

En résumé, le processus de la vérification vise à contrôler qu'il n'y a pas de cycles dans les hiérarchies constituant l'ontologie, qu'il n'y a pas de redondances de concepts ou de relations et que ces hiérarchies sont bien connexes (c'est à dire qu'elles ne présentent pas des parties isolées des autres, et donc sans aucun sens, vu que la sémantique est transmise à travers les liens).

II.8.4.2 La validation:

Elle consiste à s'assurer qu'on a construit la bonne ontologie qui doit être conforme au domaine de connaissances dans le sens où elle reflète toute la sémantique qu'il renferme, et complète c'est à dire qu'elle présente toutes les connaissances du domaine qu'elle modélise.

En général, le processus de la validation consiste à poser des questions de compétence dans un système à base de connaissances (SBC) opérationnel. L'impossibilité de poser de telles questions ou d'en fournir une bonne réponse ou la possibilité de fournir une réponse mais qui n'est pas correcte ou cohérente, met en cause la modélisation des connaissances et nécessite un retour en arrière pour une éventuelle correction ou maintenance de l'ontologie.

II.8.5 Langages de représentation, méthodologies de construction et éditeurs de développement ontologiques

II.8.5.1 Les langages de représentation d'ontologies

Pour qu'une ontologie puisse être exploitée par une application, elle doit être spécifiée et codée, c'est à dire exprimée sous une forme machinable (formelle) en utilisant un langage de représentation d'ontologies. A l'aube des années 90 plusieurs langages de représentation d'ontologies basés sur l'IA et s'inspirant essentiellement de la logique de premier ordre, la logique de description ou les frames de Minsky ont été spécifiés. Au milieu des années 90, et avec l'émergence d'Internet d'autres langages basés web ont été créés permettant d'exploiter la richesse du web. En plus de ces langages de représentation d'ontologies, d'autres formalismes de représentation de connaissances qui sont aussi proposés par l'IA et dont certains d'entres eux ont été utilisés vers la fin des années 90, peuvent aussi être considérés comme des langages de représentation tel que les réseaux sémantiques, les langages de Frames, les graphes conceptuels ainsi que les logiques de description.

Dans ce qui suit, nous allons présenter une synthèse des langages de représentation des ontologies en s'inspirant de différenets références bibliographiques, en l'occurrence: [IZZ 06], [SUX 04], [SHN 05], [AUB 07], [COR et al 02], [OZD 04], [CAH 05], [CEC 01]:

II.8.5.1.1. Les langages de représentation traditionnels:

. Les réseaux sémantiques: inventés en 1968 par Quinlan, permettant de modéliser un domaine de connaissances à travers un modèle graphique, où les nœuds représentent les concepts, et les arcs représentent les relations supportant ainsi la

notion d'héritage et permettant d'appliquer des mécanismes de raisonnement fondés sur le parcours du réseau.

. Les frames de Minsky: Comme son nom l'indique, ils sont inventés par minsky en 1975, permettent de représenter le domaine de connaissances dans un schéma constituant des frames, où chaque frame représente une entité (physique ou logique), et est décrite par un ensemble de slots et sont reliés entre eux par des relations d'héritage permettant d'hiérarchiser ces frames.

. Les graphes conceptuels (GC): Introduits par Sowa en 1974, inspirés à la fois des réseaux sémantiques et des frames de minsky, décrivant l'organisation des connaissances d'un domaine à travers des graphes biparties, finis, connexes décrivant les concepts (CG) et les relations (RG), ou chaque nœud relation relie deux concepts à travers des arcs.

. Les logiques de description: Introduits par Napolien en 1990, ils permettent de représenter les connaissances à travers deux types de langages: Les langages assertionnels (A-Box) décrivant les entités, et les langages terminologiques (T-Box) décrivant les concepts et les rôles en manipulant trois types d'entités: Les individus, les concepts et les rôles.

. KIF (Knowledge Interchange Format): Introduit en 1992, il se base sur la logique de premier ordre pour représenter les concepts, les relations, les fonctions, les axiomes, les instances ainsi que les procédures. Il est considéré comme le langage le plus expressif par rapport aux autres langages de représentation d'ontologies.
http://logic.standford.edu/kif/kif.html

. Le langage ONTOLOINGUA: Créé en 1992, à l'université de Standford en ajoutant une syntaxe au langage KIF permettant de capturer intuitivement l'assemblage des axiomes en combinant des frames de la logique du premier ordre, supportant ainsi la conception et la spécification d'ontologies à travers un logique sémantique (basé KIF) en modélisant les relations, les classes, les fonctions, les individus et les axiomes.
http://ontolingua.standford.edu/

. LOOM: Créé en 1992 à l'université de Southcalifornia, il est basé sur la logique de description et les règles de production. Il peut fournir une classification

automatique des concepts à travers un moteur déductif (classifieur) raisonnant en chaînage avant et fournissant une unification sémantique.

http://www.isi.edu/isd/loom/loom-home.html

. **OCML** (Operational Conceptual Modelling Language): Créé en 1993 par KMI de l'Open University, en ajoutant des définitions opérationnels pour chaque fonction du langage ONTOLINGUA, et en prenant en charge les règles de production et de déduction facilitant ainsi la construction de mécanismes de raisonnement, ce qui fait d'OCML un langage de représentation et de développement d'ontologies exécutables adaptées aux méthodes de résolution de problèmes.

. **FLOGIC** (Frame-Logic): Créé en 1995 à l'université de Karlsruhe, il combine la logique de premier ordre et les frames de Minsky pour représenter les concepts, les taxonomies de concepts, les relations binaires, les fonctions, les instances, les axiomes et les règles de déduction, ces derniers sont utilisés par le moteur d'inférence (OntoBroker) pour dériver de nouvelles connaissances.

. **OKBC** (Open Knowledge Base Activity): Créé en 1997, il s'agit d'un protocole plutôt qu'un langage de représentation. Il permet l'accès aux ontologies et base de connaissances à travers un API (Application Program Interface) intégré. OKBC offre un modèle uniforme de système de représentation de connaissances basé sur une conceptualisation commune. Il a été choisi par FIPA comme un standard d'échange d'ontologies.

http://www.ai.sri.com/~okbc/

. **Cycl:** Il s'agit d'une extension de la logique des prédicats en lui ajoutant la prise en charge d'égalité, de raisonnement par défaut et de skolimisation. Son objectif est de construire un modèle formel représentant une large base de connaissance sur machine (ontologie).

http://www.cyc.com

. **Telos:** Inventé au sein de l'université de Toronto pour supporter le développement des systèmes d'informations. Il permet de représenter les connaissances d'un domaine particulier, ainsi que des bases de données déductives. Il permet aussi de représenter explicitement le temps, les primitives de spécification des contraintes d'intégrité et des règles de déduction.

II.8.5.1.2. Les langages de représentation d'ontologies basés web

. XML (eXtensible Murkup Language): il s'agit d'un standard ouvert proposé par le W3C pour décrire les données et structurer les documents sur le web. XML tout seul ne peut être considéré comme un langage de représentation d'ontologie, alors que XML Schéma qui définit la structure, les contraintes, et la sémantique des documents XML peut en certain sens être utilisé pour spécifier des ontologies.

. XOL (XML Ontology Language): Il combine les frames et la syntaxe de XML pour représenter les concepts, les taxonomies et les règles binaires, son apport est de fournir un format d'échange des définitions d'ontologies à travers un ensemble d'utilisateurs. Le développement de XOL est inspiré du language Ontolingua que lui diffère par la syntaxe basée XML au temps où le language Ontolingua est basé Lisp.

. SHOE (Simple HTML Ontology Extensions): Créé en 1996. Il s'agit d'une extension simple du langage HTML pour incorporer de la sémantique aux documents web traditionnels à travers l'annotation des pages HTML par des ontologies. SHOE combine les frames et les règles de production pour représenter les concepts, les taxonomies, les relations, en plus des règles sous la forme de clauses de Horn qui vont être utilisés par un moteur d'inférence pour dériver de nouvelles connaissances et il ne possède aucune ontologies, catégories, relations, ou inférences prédéfinies.

. RDF (Resource Description Framework): C'est un modèle de données décrivant les ressources web à travers un ensemble de nœuds représentant les objets ou les ressources, et un ensemble d'arcs représentant les propriétés formant ainsi un graphe de relations entre les différentes ressources, où chaque ressource se voit associée un identifiant unique, URI (Unique Resource Identifier), ces ressources sont donc liées entre elles ou à un littéral par le biais d'un triplet, « sujet, prédicat, objet », mais comme RDF seul ne définit aucune primitive pour créer des ontologies, il donne une base pour plusieurs langages de définition d'ontologie tel que RDFS.

. RDFS: Il ajoute une dimension sémantique aux graphes RDF notamment en permettant de définir des classes qui pourront être hiérarchisées offrant ainsi un ensemble de primitives tel que: rdfs: Class, rdf: Property, rdfs: SubClass, il permet aussi de définir des classes de classes, des classes de propriétés, des classes de littéraux

(strings, entiers, booléens, etc), et des classes de faits. Il offre en plus la possibilité de définir des instances de relations entre ressources et classes, des relations de subsomptions entre classes et relations de subsomptions entre propriétés en utilisant respectivement les propriétés de rdfs:

rdf: type, rdfs: SubClassOf, et rdfs: SubPropertyOf, et en utilisant les propriétés: rdfs: domain et rdfs: range, rdfs permet de restreindre ou de limiter la porté ou le domaine d'application de certaines relations. RDFS n'est pas considéré comme un vrai langage de représentation d'ontologie grâce à sa faiblesse d'expressivité tel qu'il ne permet pas de représenter plusieurs choses tel que par exemple l'intersection, l'union, le complément de classe, les cardinalités, les propriétés de relations tel que la transitivité, la symétrie, l'inverse, etc.

. OIL (Ontology Inference Layer): Compatibles avec RDF(S) permettant ainsi de décrire les ontologies en combinant les primitives de modélisation utilisées dans les langages de frames et le raisonnement formel des logiques de description pour exprimer les ontologies sur le web. Il est de faible expressivité vu qu'il ne supporte pas tout les types tel que les types concrets d'où la volonté de pouvoir conserver un test de subsomption décidable.

. DAML+OIL: Il s'agit d'une extension RDF(S), en ajoutant des contraintes aux valeurs attribués aux propriétés et en spécifiant les propriétés qu'une classe peut avoir. Il supporte les types primitifs existant dans XML ainsi que la définition d'un certain nombre d'axiomes tel que l'équivalence de classes ou de propriétés. Il est équivalent à une logique descriptive très expressive, mais demeure assez limité dans le contexte d'ontologies d'application sur le web.

. OWL (Ontology Web Language): Comme son nom l'indique OWL est un langage de représentation d'ontologies destinées à être publiées et partagées sur le web. Il s'agit de la dernière recommandation du fameux W3C dans ce sens et constitue une surcouche de RDF(S) qui est lui-même une surcouche de XML. Voir la figure ci-dessous.

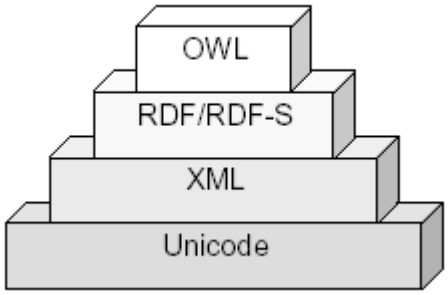


Figure II.7: Relation entre Unicode, XML, RDF, et OWL [AUB 07]

Il s'agit donc d'un langage qui permet d'hiérarchiser les classes représentant les concepts d'un domaine particulier à travers des relations de types « is-a », tout en spécifiant leurs propriétés qui peuvent être héritées à partir d'autres classes parents dans l'arborescence, ainsi que des contraintes de construction du graphe ou de la structure hiérarchique tel que les cardinalités minimales et maximales des relations qui peuvent être, autres que les relations de type « is-a », des relations logiques tel que l'(in)égalité, l'union, l'intersection ou la disjonction entre classes. Une ontologie OWL est caractérisée par une entête d'axiomes et de faits représentant les méta données décrivant l'ontologie, où les axiomes décrivent les concepts ou les relations, tel que les labels fournissant les noms des concepts ou des relations. En plus des commentaires pouvant être exploités pour générer une documentation associée à l'ontologie en cours de développement. Pour répondre aux besoins d'expressivité ontologique croissante OWL s'est évolué en mettant en œuvre trois sous langages en l'occurrence OWL-Lite, OWL-DL, et OWL-Full:

- **OWL-Lite**: La forme la plus simplifiée (Classification hierarchique) et la plus facile à implémenter (contraintes simples) d'OWL. Il s'agit d'une extension de RDF(S), et pour pouvoir exprimer les notions citées ci-dessus, OWL-Lite utilise en plus des constructeurs de RDF(S): Class, Property, subClass, SubProperty, domain, range, etc. De nouveaux constructeurs tel que IntersectionOf, equivalentClass, equivalentProperty, Cardinality, MinCardinality, MaxCardinality, InverseOf, DataTypeProperty, ObjectTypeProperty (Type de données), AllValuesFrom, SomeValuesFrom (restriction sur les rôles), Ontology, Imports (informations d'entête),etc.

- OWL-DL: Il s'agit d'une extension de OWL-Lite (en ajoutant des axiomes et des combinaisons binaires des expressions de classes et de relations), principalement basée sur la logique de description offrant ainsi une expressivité plus importante que celle d'OWL-Lite tout en assurant la complétude (tous les objectifs et buts démontrables peuvent être déduits), et de la décidabilité (tout raisonnement doit se terminer à un temps fini), mais il présente certains restrictions à l'utilisateur par exemple une classe ne peut être une instance d'une autre classe.

- OWL-Full: il s'agit d'une extension de OWL-DL. Il comble les lacunes des deux premiers sous langages OWL, en accroissant ainsi l'expressivité ontologique et en n'imposant aucune restriction à l'utilisation, mais aucune garantie de calculabilité de l'ontologie ainsi créé n'est fournie.

Donc en fonction de leurs besoins, les développeurs d'ontologies choisissent le sous langage qui leur convient le plus, qu'il s'agisse de la simplicité et de la facilité d'implémentation (OWL-Lite), de la richesse d'expression (OWL-DL), ou d'éviter les restrictions pouvant leur y imposées (OWL-Full).

Il est à noter qu'une ontologie OWL-Lite valide est une ontologie OWL-DL et OWL-Full valide.

Une étude comparative entre les langages de représentation d'ontologies les plus utilisées proposées par [COR 03], où Corcho a comparé tous ces langages dans un seul cadre évaluatif et par rapport à un sous ensemble d'attributs, où le symbole « + » désigne que l'attribut est supporté par le langage, et le symbole «-» désigne que l'attribut n'est pas supporté par le langage, et le symbole « ± » désigne que l'attribut n'est pas directement supporté par le langage. Mais il peut être représenté en intégrant d'autres fonctionnalités. Une conclusion de cette étude est résumée dans la *Table 1*. Il est à noter que les résultats obtenus avec le langage OWL, sont similaires à ceux obtenus par DAML et OIL (mais dans le contexte de spécification de ce travail).

- Critères de sélection d'un langage de représentation d'ontologies: Selon [IZZ 06], les critères de sélection de langage de représentation d'ontologie les plus souvent retenues sont: L'expressivité reflétant la quantité de connaissances ou le nombre de concepts pouvant être utilisés pour décrire les composants d'une ontologie, et la performance qui définit la capacité et la facilité de

représentation de connaissances du domaine. Il est à noter que plus le langage est expressif moins il sera performant. Il y a aussi un autre dilemme entre l'expressivité et le raisonnement, tel que parfois l'expressivité du langage doit être limitée pour assurer un bon service de raisonnement.

Concepts généralité	Onto-lingua	OCML	LOOM	FLOGIC	XOL	SHOE	RDF/S	OIL	DAML
Métaclasses	+	+	+	+	+	-	+	-	-
Partitions	+	±	+	±	-	-	-	+	+
Documentation	+	+	+	±	+	+	+	+	+
Attributs									
Attributs-instance	+	+	+	+	+	+	+	+	+
Attributs-classe	+	+	+	+	+	-	-	+	+
Etendu local	+	+	+	+	+	+	+	+	+
Etendu global	±	±	+	-	+	-	+	+	+
Facets									
Valeurs de slots par défaut	-	+	+	+	+	-	-	-	-
Contraintes de type	+	+	+	+	+	+	+	+	+
Contraintes de cardinalité	+	+	+	±	+	-	-	+	+
Documentation de slots	+	+	+	-	+	+	+	+	+
Taxonomies									
Sous classes	+	+	+	+	+	+	+	+	+
Partitions de sous classe exhaustifs	+	±	+	±	-	-	-	+	+
Décompositions disjoints	+	±	+	±	-	-	-	+	+
Non sous classe	±	-	±	-	-	-	-	+	+
Relations et fonctions									
Relations et fonctions n-aire	+	+	+	±	±	+	v	±	±
Contraintes-type	+	+	+	+	+	+	+	+	+
Contraintes d'intégrité	+	+	+	+	-	-	-	-	-
Définitions opérationnelles	-	+	+	+	-	-	-	-	-
Axiomes									
Logique 1er ordre	+	+	+	+	-	±	-	±	±
Logique 2ème ordre	+	-	-	-	-	-	-	-	-
Axiomes nommés	+	+	-	-	-	-	-	-	-
Axiomes fixes	+	+	+	-	-	-	-	-	-
Instances									
Instances de concepts	+	+	+	+	+	+	+	+	+
faits	+	+	+	+	+	+	+	+	+
assertions	-	-	-	-	-	+	±	±	±

Table II.1: Comparaison des langages de représentation d'ontologies.

II.8.5.2 Méthodologies de construction d'ontologies:

Dans ce qui suit nous décrivons les méthodologies les plus citées dans la littérature, où le concepteur suit l'une d'entre elles. Selon le type de construction qu'il va adopter, construire une ontologie à partir du zéro, ou construire une ontologie par intégration ou utilisation des ontologies préalablement existantes (fusion, alignement, mapping, etc):

II.8.5.2.1. La méthodologie d'Uschold et King:

Connue aussi par la méthodologie **Entreprise,** selon [IZZ 06], et [CAH 05], elle est inspirée des constructions modélisant une entreprise, et elle comprend:
L'identification de l'objectif de l'ontologie, sa construction, son évaluation, et finalement sa documentation. Lors du processus de construction, les auteurs proposent de capturer les connaissances, les coder, et finalement intégrer si nécessaire des ontologies préexistantes dans l'ontologie courante. Les trois stratégies suivantes sont proposées par les auteurs pour identifier les concepts: une stratégie descendante (top down strategy) où les concepts les plus généraux sont identifiés puis sont spécialisés, une stratégie ascendante (down top strategy) où les concepts les plus spécifiques sont identifiés puis sont généralisés, et une stratégie mixte (midle out strategy), où les concepts les plus importants sont identifiés puis sont généralisés et/ou spécialisés. Cette méthodologie est indépendante du système qui l'exploite et son processus de construction d'ontologie est indépendant de l'objectif de l'ontologie. Un schéma illustratif de la méthodologie Entreprise est présenté dans la figure 8:

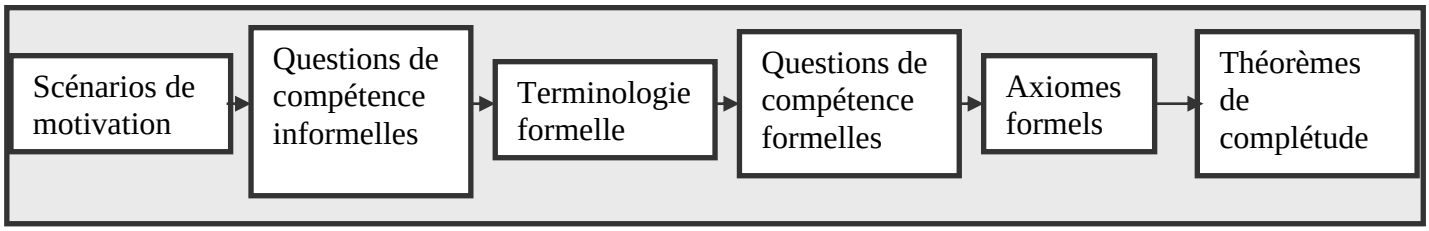


Figure II.8: La méthodologie de d'Uschold et King, *(Entreprise).*

II.8.5.2.2. La méthodologie de Gruninger et Fox:

Appelée aussi la méthodologie **TOVE**, Selon [IZZ 06], et [CAH 05], elle est inspirée du développement des SBC utilisant la logique du premier ordre. Elle comprend: L'identification des principaux scénarios (les application de l'ontologie),

l'identification des questions de compétence (que le système est censé pouvoir répondre) et alors identifier les concepts et les axiomes de l'ontologie en utilisant une stratégie mixte (midle out). Dans cette méthodologie qui est semi dépendante du système qui l'exploite, une description informelle des spécifications de l'ontologie sont formalisés (logique de premier ordre), donc elle peut être utilisée comme un guide de transformation de scénarios informels en modèles formels.

II.8.5.2.3. La méthodologie Methontology:

Selon [IZZ 06], et [CAH 05], cette méthodologie est créée au sein du laboratoire de l'IA de l'université polytechniques de Madrid en 1998. Elle est utilisée pour construire une ontologie que ce soit à partir de zéro, ou en réutilisant d'autres ontologies préexistantes et elle comprend: L'identification du processus de développement d'ontologie,

c'est à dire qu'elle couvre tout le cycle de vie de l'ontologie qui est basé sur un prototype évolutif et sur des techniques particulières pour la réalisation des activités liées aux ontologies qui sont: l'ordonnancement de contrôle, (scheduling control), assurance de qualité, spécification, acquisition de connaissances, conceptualisation, en plus des activités de gestion de projets, et de support, cette méthodologie est indépendante de l'application qui l'exploite. Elle utilise une stratégie mixte pour l'identification des concepts de l'ontologie. Un schéma illustratif de la méthodologie Methontology est présenté dans la figure 9:

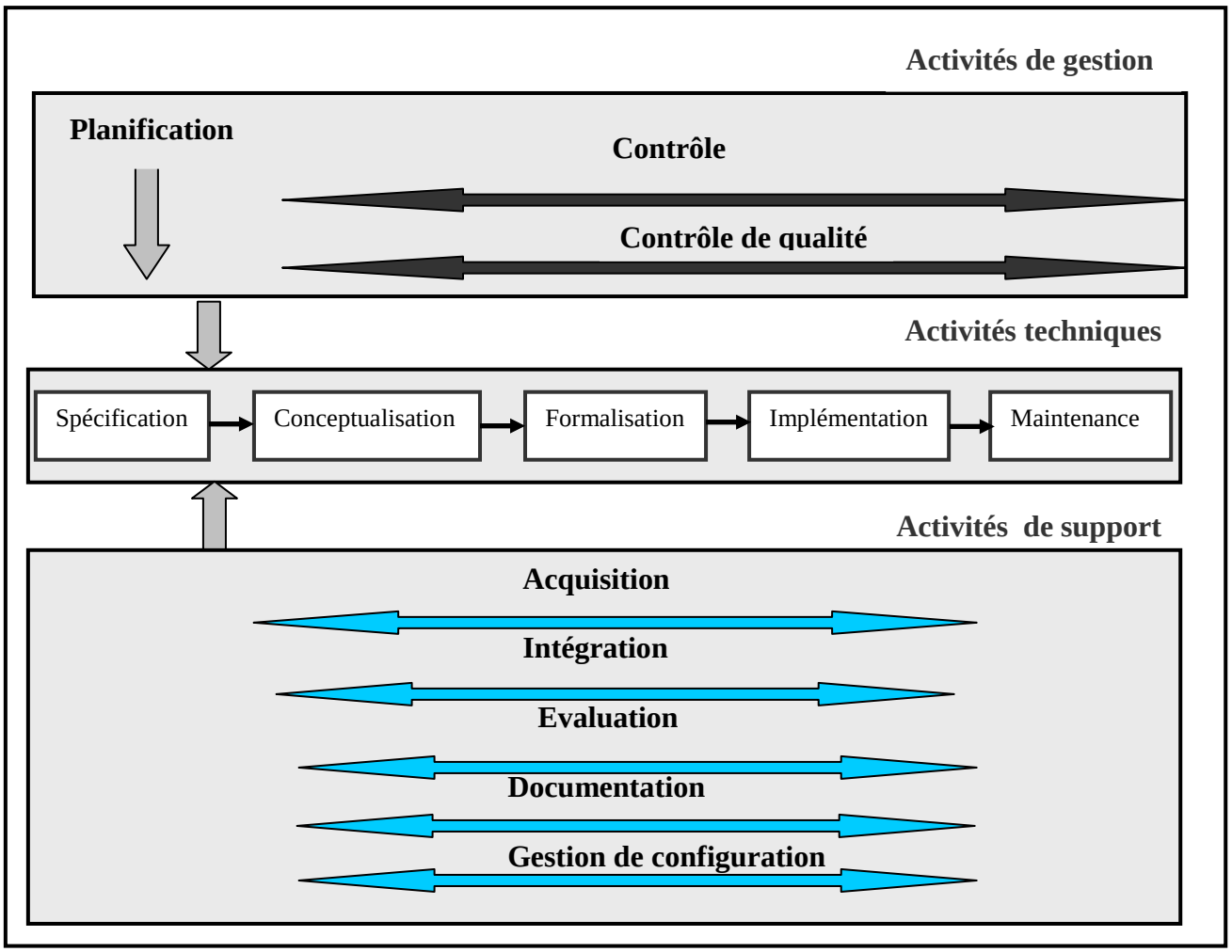


Figure II.9: Les composantes de Methontology.

II.8.5.2.4. La méthodologie de Amaya Berneras et al:

Selon [CAH 05], cette méthodologie utilise une stratégie descendante pour l'identification de concepts de l'ontologie. Elle est dépendante de l'application qui l'exploite, c'est à dire que à chaque fois une application est créée, l'ontologie modélisant les connaissances requis par cette application est construite en utilisant d'autres ontologies préexistante, et à son tour cette ontologie peut être utilisée pour construire des ontologies d'autres applications.

II.8.5.2.5. La méthodologie basée SENSUS (SENSUS-BASED METHODOLOGY):

Selon [CAH 05], cette méthodologie qui est développée au sein de l'IST (Information Sciences Institut) pour fournir une structure conceptuelle pour développer des translateurs, et elle utilise une stratégie descendante pour

l'identification des concepts de l'ontologie (à partir des ontologies plus larges préexistantes) et elle est semi dépendante de l'application qui l'exploite.

II.8.5.2.6. La méthodologie On-To-Knowledge:

Selon [CAH 05], elle inclut l'identification des objectifs attendus des outils de gestion de connaissances, et elle est basée sur l'analyse de scénarios utilisés. Elle comprend: L'identification des questions de compétences spécifiées, l'étude d'une version réutilisable et construction d'une première tentative de l'ontologie puis si nécessaire un raffinement, évaluation et maintenance, cette méthodologie est dépendante de l'application qu'elle l'exploite.

Une comparaison générale entre les méthodologies ci-dessus est résumée dans le tableau suivant:

Table2:

Critères	Entreprise	Tove	Meth-ontology	Amaya-Bernares et al	SENSUS	OnTo-Knowledge
Détaille de la methodologie	très petite	petite	grande	très petite	très petite	très petite
Recommandation de formalisation	néant	logique	néant	néant	néant	néant
Stratégie des applications de construction	Application indépendante	Semi-dépendante	indépendante	dépendante	Semi-dépendante	dépendante
Stratégie d'identification de concepts	Middle-out	Middle-out	Middle-out	Top-down	Top-down	Top-down
Cycle de vie recommandé	néant	néant	oui	néant	néant	oui

Table II.2: La différence entre les cinq méthodologies.

II.8.5.3 Les éditeurs d'ingénierie ontologique:

Les éditeurs jouent un rôle primordial en assistant l'ontologiste à construire et à développer les ontologies en suivant une méthodologie de conception plus ou moins complète et en respectant les principes du cycle de vie, et en exploitant des mécanismes de visualisation et de vérification du modèle conceptuel résultant des vérificateurs logiques pour vérifier la satisfiabilité du modèle spécifié. Dans ce qui suit nous allons présenter un certain nombre d'éditeurs et d'outils environnementaux de

création et de gestion d'ontologie en s'inspirant de quelques références bibliographiques tels que: [IZZ 06], [FUR 04], [CHA et al 04], nous avons donc choisi les outils les plus utilisés à savoir: DOE, OntoEdit, Protégé2000, ODE, webODE, WebOnto, OilEd, Ontolingua, Ontosaurus, Terminae.

1. DOE (Differential Ontology Editor): Fournit un début d'implémentation à la méthodologie de construction en utilisant les principes différentiels proposés par Bachimont, puis en ajoutant les concepts référentiels permettant ainsi d'interagir avec les hiérarchies. Le modèle conceptuel ainsi obtenu est finalement proche de celui du langage RDFS (des exports de l'ontologie dans les langages: RDFS, DAML+OIL, et OWL sont possibles), et formellement cet outil fournit des inférences en vérifiant la consistance de l'ontologie mais il ne permet pas d'éditer les propriétés conceptuels des concepts.

2. OntoEdit: (Ontology Editor): Développé à l'université de Karlsruhe, il est indépendent de tout langage de représentation, et plusieurs plugins peuvent être intégrés permettant la visualisation graphique de l'ontolgie, le test de la cohérence de l'ontologie, etc. Tels que ONTOKICK pour générer les spécifications de l'ontologie et OntoBroker pour inférer de nouvelles connaissances, et aussi pour importer et exporter l'ontologie vers de multiple langages tel que RDFS, DAML+OIL, et FLogic.

3. Protégé 2000: C'est un environnement graphique modulaire d'ingénierie ontologique développé au sein de l'SMI de Standford composé d'un éditeur et d'une librairie de plugins permettant l'édition, la visualisation, le contrôle ainsi que l'extraction d'ontologies à partir de sources textuelles. Une ontologie sous protégé 2000 est constituée de plusieurs hiérarchies de classes, où chaque classe est décrite par un ensemble d'attributs (slots) pouvant être spécifié en remplissant les formulaires fournis par protégé, le succès de protégé 2000 est justifié par la conception ainsi que l'architecture logicielle très bien spécifiée de son interface en plus des nouvelles fonctionnalités apportées par les plugins pouvant être intégrés dedans, tel que le plugin permettant d'utiliser le langage PAL (Protege Axiom Language), un sous ensemble du langage KIF permettant la spécification de nouvelles contraintes au cas où le formalisme des Frames ne suffit pas, ou Prompt pour la fusion de plusieurs ontologies. La figure ci-dessous montre une ontologie de voyage éditée avec Protégé 2000.

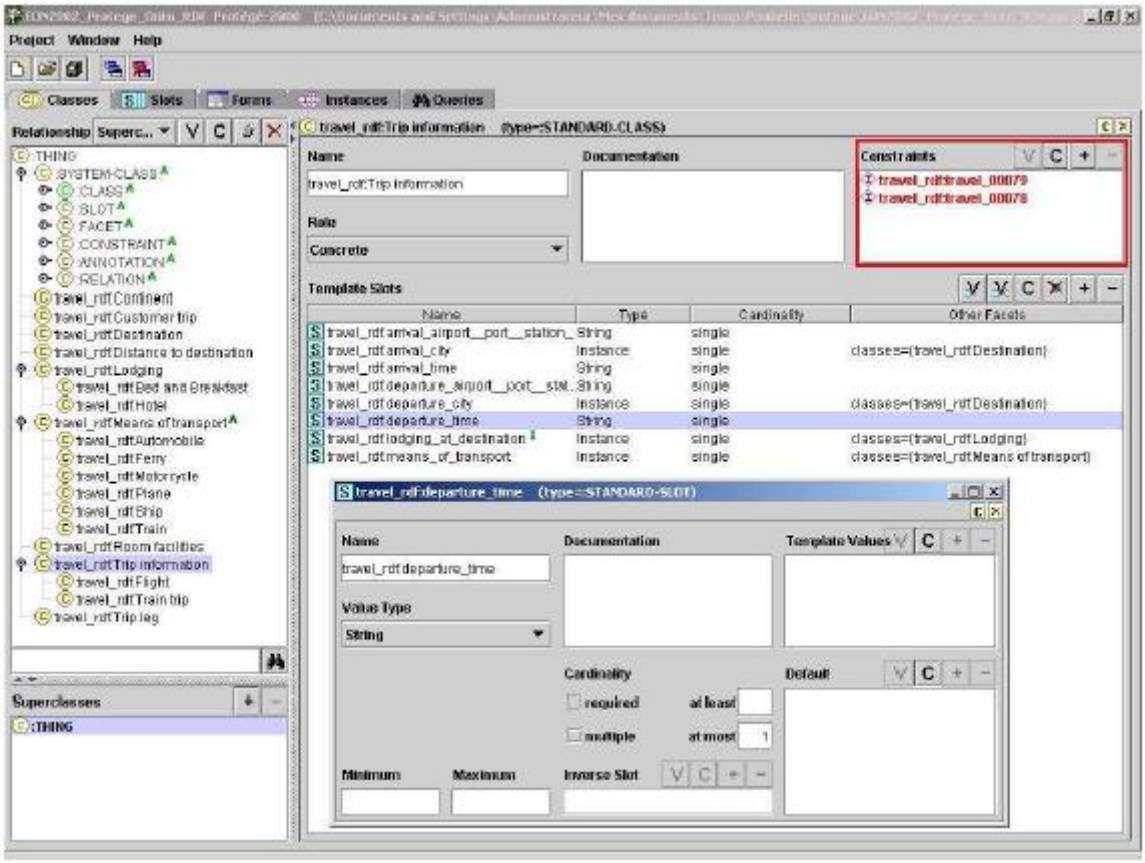


Figure II.10: L'éditeur d'ontologie Protégé2000, [FUR 04].

4. DOE (Ontology Design Environment): Développé à l'université de Madrid, permet de construire des ontologies au niveau connaissance, et supporte les langages de représentation Ontolingua et FLogic. Un moteur d'inférence spécifié en prolog peut être intégré pour inférer de nouvelles connaissances.

5. Web ODE: Développé au sein du même laboratoire où ODE a été développé. Il s'agit d'une plate forme de conception d'ontologies fonctionnant en ligne favorisant ainsi le travail collaboratif, il s'agit donc d'une adaptation de ODE pour le web.

6. WebOnto (web ontology): Développé à l'Open University, et supporte OCML comme un langage de représentation.

7. OilEd (OIL Editor): Développé à l'université de Manchister, il favorise la construction d'ontologies de petites et de moyennes tailles formalisés en utilisant les standard DAML+OIL ainsi que la logique de description. Il offre, grâce à son moteur d'inférence des services de raisonnement permettant de tester la satisfiabilité et de découvrir des subsomptions restées implicites dans l'ontologie. Il permet l'export d'ontologie sous les formats RDF, DAML+OIL, et OWL.

II.9 La réutilisation et l'évolution de l'ontologie en face du problème d'hétérogénéité

Lorsqu'une ontologie ne répond plus aux exigences des individus d'une communauté, un processus d'évolution doit avoir lieu pour l'adapter aux nouveaux besoins et fonctionnalités exigés par ces utilisateurs. Le processus d'évolution consiste en la modification de l'ontologie de façon à pouvoir s'adapter, par exemple, à l'évolution et aux changements portant sur le domaine du problème, en éliminant ou en ajoutant de nouveaux concepts et relations tout en intégrant les nouveaux slots et facets portant sur les concepts ainsi que les nouvelles contraintes portant sur les relations, puis en la propagation de ces modifications en mettant à jour tous les artefacts dépendant de l'ontologie ainsi modifiée et notamment l'application qui utilise cette ontologie.

Certains auteurs tels que [NOY et al, 03] définirent l'évolution de l'ontologie comme un processus adaptatif, c'est à dire qu'elle consiste en la capacité à gérer les changements de l'ontologie et leurs impacts en créant et en maintenant plusieurs versions de l'ontologie, bien que d'autres [MAE et al, 02] proposent la méthodologie AIFB pour gérer l'évolution de l'ontologie à travers cinq phases:

- **La première phase:** Consiste à représenter les changements générés par l'évolution qui s'effectue en fonction du type de changement, élémentaire ou complexe. Les changements élémentaires comme son nom l'indique correspondent à des opérations unitaires non décomposables tel que l'ajout, la suppression, et la modification des entités ontologiques, alors que les changements complexes correspondent à des opérations décomposables en plusieurs opérations élémentaires tels que le déplacement, la fusion, et la séparation des entités ontologiques.
- **La deuxième phase:** Consiste à s'assurer que la consistance des ontologies est conservée, c'est à dire que les entités ontologiques et les valeurs respectent toujours les contraintes imposées par le modèle conceptuel, et ce en ajustant ou en ajoutant les contraintes nécessaires.

- **La troisième phase:** C'est la phase d'implémentation. Elle consiste à exécuter les changements une fois qu'ils ont été validés par l'utilisateur.
- **La quatrième phase:** Consiste en la propagation des changements de l'ontologie en cours, aux ontologies dépendantes de celle-ci.
- **La cinquième phase:** Consiste à évaluer les changements et à les corriger si cela s'avère nécessaire.

Il est à noter que les liens entre les différentes ontologies ne sont pas automatiquement maintenus lors de l'évolution générant ainsi des effets négatifs sur la préservation de l'interopérabilité entre elles (les ontologies). L'évolution d'ontologie, bien que efficace pour l'adaptation des ontologies aux nouvelles situations et contraintes imposées par le changement du domaine ou par l'évolution des besoins et exigences des utilisateurs, présente l'inconvénient de ne pouvoir conserver l'historique des versions évoluées, alors pour contourner ce problème, une nouvelle approche est proposée par [NOY et al, 00] , il s'agit donc d'une méthodologie qui se base sur le principe du versionning où la gestion de l'évolution de l'ontologie impose la possibilité de conserver et d'accéder aux différentes versions créées le long du processus d'évolution d'ontologie, ainsi que de modéliser les relations notamment sémantiques (entre ces versions) permettant de détecter les changements produits lors du processus d'évolution. Bien que cette méthode permet d'éviter le problème de la méthode précédente, elle ne suggère pas comment ces versions peuvent être créées. A cet effet, une approche globale qui combine les deux approches a été proposée. Il s'agit d'une proposition qui intègre un formalisme graphique de représentation de l'évolution des ontologies en se basant sur les concepts et les techniques développés pour les BDDs temporelles, l'évolution de schéma et le versionning des bases de données.

II.10 Quelques opérations de réutilisation d'ontologies

En plus des opérations de réutilisation d'ontologies présentées dans la section précédente, une multitude d'autres opérations de réutilisation d'ontologies est collectée par Gomez-Perez, et dans ce qui suit nous en présentons un résumé:

- **Alignement d'ontologies** (Ontology Aligning): Appelé aussi, ontology mapping. Il s'agit de trouver les correspondances entre deux (ou plusieurs) ontologies puis les sauvegarder et/ou les exploiter dans un certain contexte.
- **Annotation d'ontologies**: il s'agit d'enrichir l'ontologie avec des informations supplémentaires sous forme de méta données ou de commentaires.
- **Vérification d'ontologie**: il s'agit de contrôler ou de réviser l'ontologie vis-à-vis les exigences de l'utilisateur tels que l'utilisabilité, l'utilité, l'abstraction, la qualité, etc.
- **Comparaison d'ontologies**: Il s'agit de trouver les différences entre deux (ou plusieurs) ontologies ou entre deux (ou plusieurs) modules d'ontologies.
- **Adaptation d'ontologie**: Comme son nom l'indique, il s'agit de faire adapter une ontologie aux besoins d'un utilisateur spécifique.
- **Enrichissement d'ontologie**: Il s'agit de faire étendre ou prolonger une ontologie en ajoutant de nouvelles structures conceptuelles (tels que les concepts, les relations, les rôles, etc)
- **Evaluation d'ontologie**: Il s'agit de vérifier la qualité technique d'une ontologie vis-à-vis un modèle de référence.
- **Evolution d'ontologie**: Il s'agit de faciliter la modification d'une ontologie tout en préservant sa consistance, l'évolution peut être vue comme une séquence de différentes activités pendant le développement de l'ontologie.
- **Extension de l'ontologie**: Il s'agit d'une activité d'enrichissement afin d'étendre la largeur d'une ontologie.
- **Intégration d'ontologie:** Comme son nom l'indique, il s'agit d'ancrer une ontologie dans une autre.
- **Localisation d'ontologie**: Il s'agit d'adapter une ontologie à une langue ou culture locale particulière.
- **Mapping d'ontologies**: Appelé aussi alignement d'ontologie. Il s'agit de trouver les correspondances entre deux (ou plusieurs) ontologies puis les sauvegarder et/ou les exploiter dans un certain contexte.

- **Matching d'ontologies**: Il s'agit de trouver ou de découvrir les relations ou les liens de correspondances entre les entités ou les modules de différentes ontologies.
- **Fusion d'ontologies:** Il s'agit de créer ou de construire une nouvelle ontologie ou un nouveau module d'ontologie à partir de deux ou plusieurs ontologies ou modules d'ontologies sources.
- **Modification d'ontologie**: Il s'agit de changer ou modifier une ontologie sans prendre en charge la préservation de la consistance.
- **Modularisation d'ontologie**: Il s'agit d'identifier un ou plusieurs modules dans une ontologie dans le but de supporter ou de faciliter la réutilisation ou la maintenance.
- **Extraction de module d'ontologie**: Il s'agit d'obtenir des modules concrets à partir d'ontologie pour être utilisé dans une proposition particulière (pour obtenir par exemple un sous vocabulaire particulier de l'ontologie originale).
- **Partionnement d'ontologie**: Il s'agit de diviser ou de partitionner une ontologie en un ensemble (pas nécessairement disjoint) de modules qui peuvent être traités séparément et dont l'union forme l'ontologie originale.
- **Réingenierie d'ontologie**: Il s'agit d'un processus de rétablissement et de transformation d'un modèle conceptuel modélisant une ontologie déjà fini et implémenté vers un modèle conceptuel plus correcte et plus complet qui va être ré-implémenté.
- **Restructuration d'ontologie**: Il s'agit de corriger et de réorganiser les connaissances contenues dans un modèle conceptuel initial, et de détecter les connaissances du domaine qui manque dans le modèle conceptuel.
- **Réparation d'ontologie**: Il s'agit d'un processus lancé par une ontologie spécifique qui s'appelle « diagnosis ontology », et qui consiste à corriger et à résoudre les erreurs tel que l'incomplétude ou l'incohérence.
- **Réutilisation d'ontologie**: Il s'agit d'utiliser une ontologie ou un module d'ontologie pour résoudre un certain problème, en exploitant une ontologie ou un module d'ontologie, par exemple pour développer de nouvelles ontologies,

le développement de différentes applications basé-ontologie, l'alignement d'ontologies, etc.

- **Recherche d'ontologie**: Il s'agit de trouver les ontologies ou les modules d'ontologie candidats pour être réutilisée.
- **Résumé d'ontologie**: Il s'agit d'obtenir une abstraction ou un résumé du contenu de l'ontologie.
- **Translation d'ontologie**: Il s'agit de changer le formalisme ou le langage de représentation de l'ontologie vers un autre. La translation d'ontologie peut faire partie du processus de ré-ingénierie d'ontologie.
- **La mise à jour de l'ontologie** (ontology upgrade): Il s'agit du moindre changement effectué sur une ontologie sans faire évoluer la version de l'ontologie.
- **Révolution de l'ontologie** (ontology upgrade): Il s'agit de remplacer une ontologie (existante) par une nouvelle version.
- **Validation d'ontologie**: Il s'agit de l'évaluation de l'ontologie tout en comparant le sens des définitions de l'ontologie avec le modèle visant à conceptualiser le domaine du problème (en répondant à la question: est ce qu'on a développé la bonne ontologie ?)
- **Ontology versionning**: Il s'agit de porter les changements d'ontologies tout en créant diverses versions de l'ontologie.

II.11 Conclusion

La modélisation d'un domaine d'application par une ontologie permet une prise en charge efficace facilitant le stockage, le traitement et la gestion de connaissances permettant ainsi d'acquérir facilement et d'une manière efficace les réponses des techniciens et des opérateurs à des requêtes de différents niveaux et dans différents buts de maintenance, d'exploitation, d'adaptation, etc.

Les ontologies constituent, de ce fait, un instrument très efficace pour modéliser un segment de la réalité vu leur nature générique en plus de la richesse d'expression qu'elles fournissent.

Le niveau de formalisme de représentati

Nous avons présenté dans ce chapitre ce qui est une ontologie, ses composants, et ses critères de classification, en plus de ses principes de bases de construction et d'utilisation, guidant de façon primordiale le processus d'ingénierie ontologique permettant aux développeurs de projet et au système de partager un référentiel commun, servant de base à la collaboration et à l'interopérabilité entre plusieurs systèmes ou entre plusieurs individus d'une communauté au sein d'un domaine particulier. Dans notre travail, l'ontologie servira de base à un référentiel aidant à l'accès aux informations pertinentes et répondant à des requêtes posées par l'utilisateur à des fins de maintenance, d'exploitation, ou d'adaptation, et particulièrement appliquées à une ontologie résultante de la fusion de plusieurs ontologies permettant d'offrir le maximum d'informations et de connaissances pour bien refléter le domaine d'application.

III.1. Introduction

La fusion d'ontologies constitue l'un des problèmes les plus adressés dans la littérature d'ontologies qui focalise sur des spécifications explicites des conceptualisations. La fusion d'ontologies peut donc être comprise comme étant l'effort de construction d'une seule ontologie à partir d'un ensemble d'ontologies sources couvrant ainsi un domaine d'application plus large. La notion d'ontologie est ultimement liée à d'autres notions qui sont le mapping (l'appareillement) et l'alignement, les trois notions ensemble constituent ce qu'on appelle « la médiation d'ontologies ». Selon [GRA et al, 06] la fusion d'ontologies est vue comme étant un processus complexe composé de trois sous processus à savoir le mapping et l'alignement d'ontologies, la réconciliation entre les différents choix des modèles conceptuels et des langages de représentation des ontologies à fusionner et une fois ces ontologies sont suffisamment homogènes, le dernier sous processus consiste à construire l'union des ontologies contrôlées qui peut être constamment complexe vu la complexité potentielle des interactions entre les axiomes des ontologies en entrée.

Le processus de fusion des ontologies s'exécute donc sur deux (ou plusieurs) ontologies en entrée et donne en sortie une seule ontologie qui regroupe toutes les connaissances des ontologies sources. Pour combler les insuffisances de la fusion manuelle des ontologies basée sur des outils conventionnels d'édition sans support technique, rendant ainsi l'opération longue, difficile et sujette à l'erreur, plusieurs systèmes et cadres de travail et de support pour la fusion ont été proposés, ces derniers sont basés sur l'heuristique de l'appariement (le matching) syntaxique et sémantique.

L'objectif de ce chapitre est donc d'explorer la technique de la fusion d'ontologies dont nous avons introduit tout d'abord à travers la notion de la réutilisation d'ontologies qui peut être mise en œuvre à travers la médiation et/ou l'intégration d'ontologies. Puis nous présentons le concept de la fusion d'ontologies, son processus, ses techniques, ses problèmes. Ensuite nous résumons le travail intitulé « Un langage et un algorithme pour la fusion d'ontologies » [RAS et al, 08], suivi par d'autres outils de fusion d'ontologies et nous terminons par quelques applications de la fusion d'ontologies et une conclusion.

III.2. La réutilisation d'ontologies

La réutilisation des ontologies déjà construites peut être faite à travers trois exploitations majeures:

- Intégration d'ontologies lors de la construction d'une nouvelle ontologie en réutilisant d'autres ontologies disponibles.
- Intégration d'ontologies en fusionnant différentes ontologies autour du même sujet ou du même domaine en une seule ontologie qui les unifie.
- Intégration d'une ontologie (ou plus) dans une application (ou plus).

Dans les deux sections suivantes nous présentons le concept de la réutilisation d'ontologies à travers deux fameux processus, à savoir la médiation et l'intégration d'ontologies:

III.2.1. La médiation d'ontologies

Elle constitue une nouvelle voie de recherche qui reste toujours très riche et qui étudie les techniques permettant la réutilisation d'ontologies tout en identifiant les ressemblances ainsi que les différences entre eux. Ces différences constituent un vrai obstacle en face de l'interopérabilité entre ces ontologies mais elles peuvent heureusement être réconciliés à travers la dite « médiation d'ontologies ». Selon [BRU et al, 06], la médiation d'ontologie peut être faite de différentes manières à savoir: la fusion d'ontologies visant à créer une nouvelle et unique ontologie à partir de plusieurs ontologies, l'alignement d'ontologies visant à découvrir (semi) automatiquement les correspondances entre les ontologies et le mapping d'ontologies visant à représenter les correspondances entre ces ontologies. Dans ce qui suit nous détaillons ces trois types de techniques de médiation d'ontologies:

III.2.1.1 Le mapping d'ontologies

C'est une spécification déclarative du sens commun entre deux ontologies à travers la représentation des correspondances entre eux. Ces correspondances sont enregistrées isolément des ontologies (voir Figure 1), constituant ainsi une partie à part des ontologies sources et représentant les liens entre les entités correspondantes à travers des axiomes spécifiés dans un langage de mapping spécifique. Ces

correspondances peuvent être utilisées par exemple pour interroger des bases de connaissances hétérogènes à travers une interface utilisateur commune, ou pour transformer ou interpréter les données d'un formalisme de représentation vers un autre.

Tout processus de mapping passe par les trois phases suivantes: découvrir les correspondances de mapping, les représenter puis les exploiter (ou les exécuter)

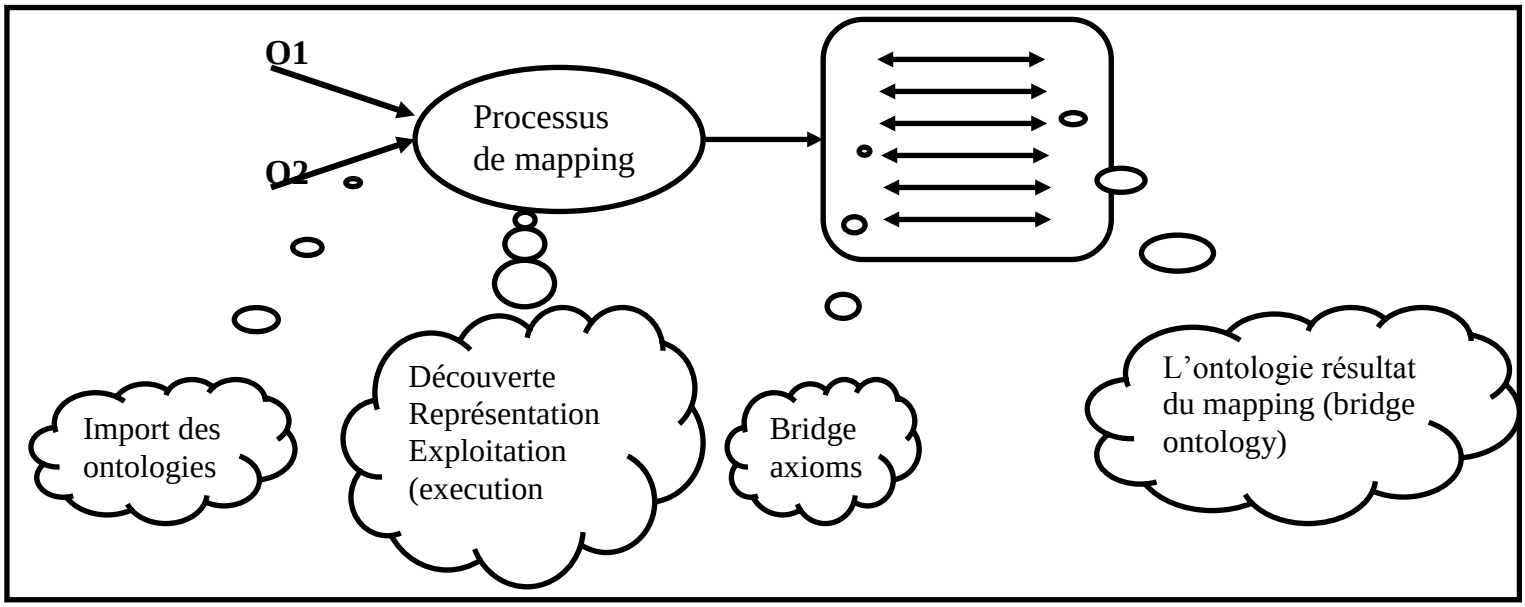


Figure III.1: Le mapping d'ontologies.

III.2.1.2 L'alignement d'ontologies

Il s'agit de la spécification des ressemblances entre deux ou plusieurs ontologies sources à travers un processus (semi) automatique de découverte de similarités. En d'autres termes, il s'agit d'un processus ayant pour entrées un ensemble d'ontologies sources et pour sortie une spécification des correspondances entre ces ontologies. Les algorithmes d'alignement sont classifiés selon deux dimensions:

- Selon la première dimension on distingue l'alignement basée schéma et l'alignement basée instance:
    - Dans l'alignement basé schéma, différents aspects de concepts et de relations des ontologies sont pris en compte pour identifier les similarités en se basant sur des mesures particulières.
    - Dans l'alignement basé instance, les instances de différents concepts de différentes ontologies sont comparés entre eux pour dégager les similarités entre les concepts.

- Selon la deuxième dimension on distingue: L'alignement au niveau élément et l'alignement au niveau structure:
    - L'alignement au niveau élément découvre les similarités en comparant les propriétés de certains concepts et relations (tel que nom).
    - L'alignement au niveau structure découvre les similarités en comparant les structures des ontologies.

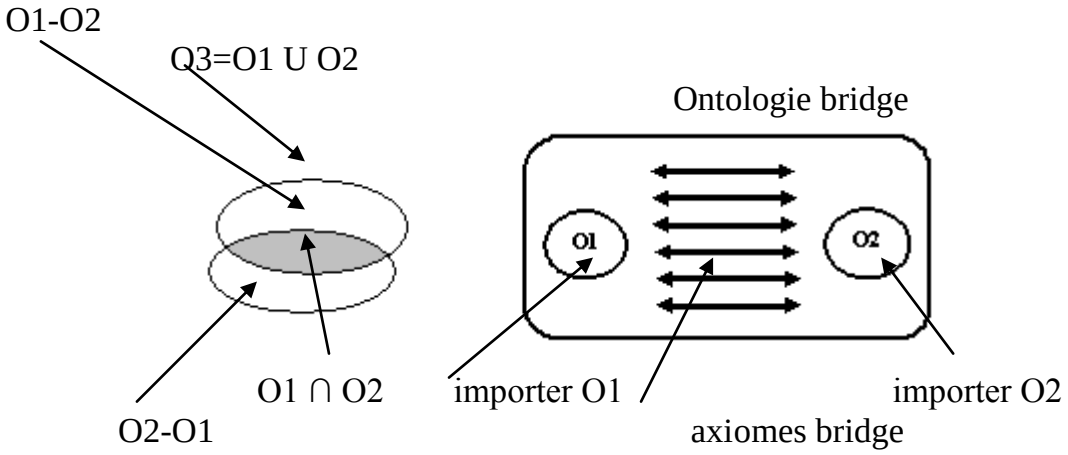
Il est à noter que les derniers types d'alignement peuvent être combinés pour aboutir à de meilleurs résultats.

III.2.1.3 La fusion d'ontologies

Il s'agit de créer une nouvelle et unique ontologie représentant l'union des deux ontologies sources de façon à regrouper toutes les connaissances contenues dans les deux ontologies, en d'autres termes toutes les ressemblances et les dissemblances présentées par les deux ontologies doivent être reflétées par l'ontologie résultat de ce processus.

Deux différentes approches de fusion d'ontologies sont proposées dans la littérature:

- Dans la première approche le processus de fusion a pour entrées deux ontologies sources et produit en sortie une nouvelle et unique ontologie de fusion regroupant toutes les connaissances des ontologies sources (par exemple PROMPT).
- Dans la deuxième approche les deux ontologies sources ne sont pas remplacées par une nouvelle ontologie de fusion mais sont annotées ou enrichies par des correspondances en utilisant des axiomes médiateurs appelés « bridge axioms » donnant ainsi une ontologie médiateur appelée « bridge ontology » (par exemple OntoMerge). Selon [DOU et al, 02], *bridging axioms* utilise des items (symboles) de vocabulaires des deux ontologies sources et cible, mais qui peuvent utiliser en plus de nouveaux symboles, l'ensemble de tous ces symboles constitue alors « l'ontologie de la fusion » (ou encore the bridging ontology).



(a) fusion complet (b) ontologie bridge

Figure III.2: La sortie du processus de la fusion.

III.2.2. L'intégration d'ontologies

III.2.2.1 Description

Afin de développer ou concevoir une ontologie qui couvre un domaine de connaissances plus large, l'intégration et particulièrement la fusion d'ontologies est une opération primordiale, où différentes ontologies à propos du même sujet sont fusionnées en une seule ontologie cohérente qui les unifie toutes. Selon Pinto et al dans [PIN 00], l'ontologie résultat d'une intégration doit répondre à une série de besoins pour lesquelles elle a été conçue et développée, en plus de:

- L'évaluation habituelle: Vérification et validation.
- Les critères d'estimation: Complétude, concision, consistance, extensibilité et robustesse.
- Les traits habituels: La clarté, la cohérence, l'extensibilité, partie prise de codage minimum, engagement ontologique minimal.

Cette ontologie doit aussi être non ambiguë, établie sur des distinctions fondamentales appropriées, consistante et cohérente par tout (bien que composée de connaissances de différentes ontologies intégrées), et devraient avoir un niveau suffisant et approprié de détail dans l'ontologie tout entière (c'est à dire qu'il ne faudra pas trouver d'îlots de niveau de détails exagérés et d'autres de niveau adéquat), et enfin la granularité se devra être homogène.

III.2.2.2 Les prérequis à l'intégration

Avant d'exécuter le processus d'intégration, [GAË 02] a mis au point un certain nombre d'activités qui doivent précéder l'opération d'intégration et permettant de choisir l'ontologie la meilleure (la plus étroitement) et la plus facilement (avec le moindre d'opérations) adaptable pour devenir l'ontologie nécessaire, et qui sont:

- La recherche et le choix des ontologies candidates à l'intégration.
- Les évaluer par des experts du domaine en fonction des critères spécialisés orientés vers l'intégration.
- Les estimer par des ontologistes en fonction de critères spécialisés orientés vers l'intégration.
- Le choix de l'ontologie la plus adéquate à réutiliser parmi les ontologies candidates analysées.

Une fois cette ontologie trouvée, ses connaissances sont intégrées dans l'ontologie source en appliquant des opérations d'intégration qui peuvent être des opérations de composition, de combinaison, de modification ou d'assemblage. Ces connaissances peuvent donc être, utilisées tel qu'elles sont, adaptées ou modifiées, spécialisées (menant à une ontologie plus spécifique), généralisées (augmentée par une connaissance plus générale ou par une connaissance de même niveau).

L'ontologie résultante de l'intégration doit donc être analysée, évaluée, et estimée pour corriger les erreurs, compléter les insuffisances, et élaguer les sur-connaissances, en répondant aux questions suivantes:

- Quelle connaissance est manquante ?
- Quelle connaissance devrait être retirée ?
- Quelle connaissance devrait être replacée ?
- Quelles sources de la connaissance devraient être changées ?
- Quelle documentation devrait être changée ?
- Quelle terminologie devrait être changée ?
- Quelles définitions devraient être changées ?
- Quelles pratiques devraient être changées ?

III.3. La fusion d'ontologies

Selon [PRE et al, 05], lors de la fusion d'ontologies deux cas de figure peuvent être représentés, dans le cas général les ontologies sources doivent disparaître et seulement l'ontologie de la fusion doit apparaître, et là l'ensemble complet des instances des ontologies sources doivent être fusionnés, alors que dans le cas spécial, les ontologies sources persistent mais qui seront enrichies par des correspondances appelées « les axiomes bridge », représentant les correspondances entre les deux ontologies sources, et là chaque ontologie source maintient son ensemble d'instances.

III.3.1. Définition du processus de fusion d'ontologies

Dans [SOW 97], l'auteur a défini le processus de la fusion d'ontologies comme étant le processus de trouver des points communs entre deux ontologies différentes A et B et de dériver une nouvelle ontologie C facilitant l'interopérabilité entre des systèmes (sur une machine) qui sont basés sur les deux ontologies A et B. L'ontologie C peut donc remplacer les deux ontologies A et B ou jouer le rôle d'un intermédiaire entre un système basé sur l'ontologie A et un autre basé sur l'ontologie B.

III.3.2. Le processus de la fusion d'ontologies

Pour des raisons de simplicité, nous supposons que le processus fusionne à la fois deux ontologies O1 et O2 en entrée donnant ainsi une seule et unique ontologie de fusion en sortie.

Dans ce qui suit nous décrivons les différentes étapes du processus de la fusion d'ontologies qui est illustré dans la figure ci-dessous:

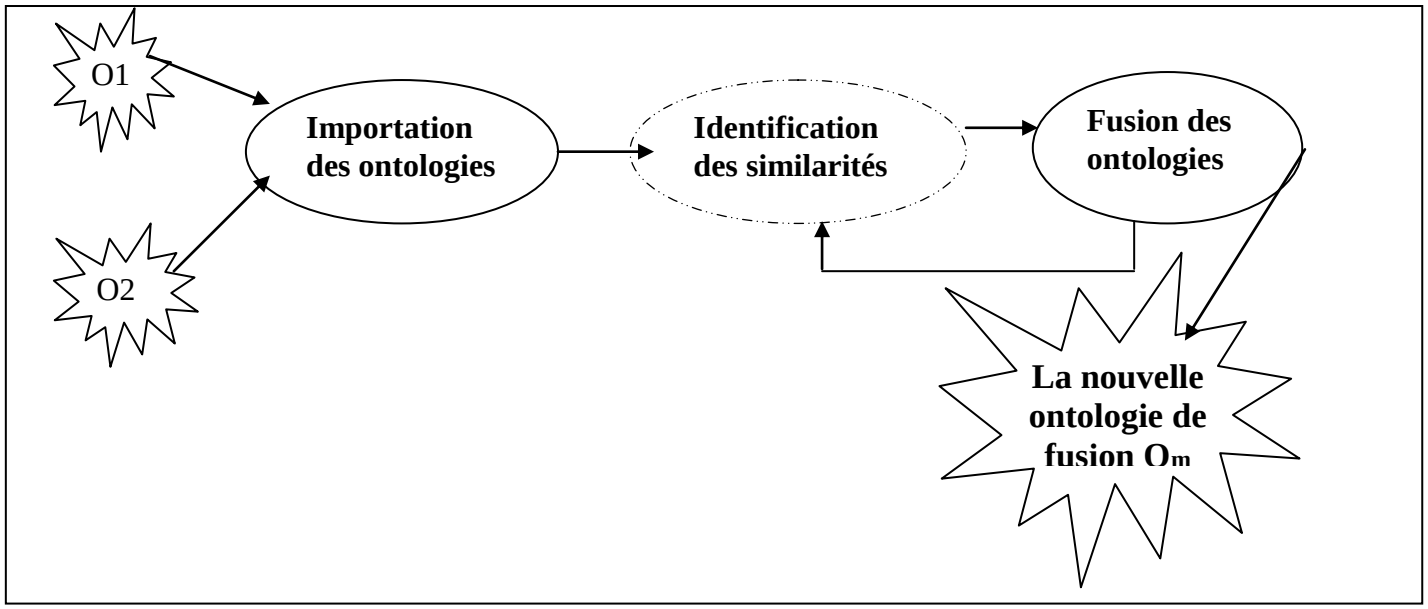


Figure III.3: Le processus de la fusion des ontologies.

III.3.2.1. L'importation des ontologies:

Les ontologies en entrée O1 et O2 peuvent être spécifiées dans différents langages de représentation, et pour pouvoir être fusionnées, elles doivent être converties en un format de représentation commun. Puis elles sont importées dans un outil de fusion (par exemple PROMPT [NOY et al, 00]), qui va effectuer la fusion proprement dite.

III.3.2.2. Identification de similarités:

Cette étape n'est obligatoire que si le processus de la fusion est automatique. Les similarités sont identifiées automatiquement à travers un opérateur d'appariement (opérateur de matching). (si le processus est complètement manuel cette étape peut être ignorée).

III.3.2.3. Fusion d'ontologies:

La spécification de la fusion est toujours manuelle mais elle peut être aidée par un outil de fusion, tel PROMPT qui donne des propositions concrètes des opérations de fusion où l'utilisateur n'intervient que pour dire « exécuter », au lieu de prendre en charge complètement l'opération de la fusion. Dans la plupart des cas, il existe un

cycle de retour en arrière sur l'étape précédente où l'outil de fusion peut donner plus de mesures de similarité quand une partie de la fusion est achevée.

III.3.3. Les techniques de fusion d'ontologies

Trois fameuses techniques de fusion (et/ou d'intégration) d'ontologies sont proposées dans la littérature. Elles servent de base dans la conception d'outils de fusion automatique ou semi-automatique, à savoir: la technique « Bottom-Up », la technique « Midle-Out » et la technique « Top-Down ».

III.3.3.1 La technique « Bottom-Up »

Cette technique est proposée par G. Stumme et A. Maedche à l'université de Karlsruhe en 2001, [STU et al, 01]. Comme son nom l'indique il s'agit d'une approche dans laquelle on remonte des concepts les plus spécifiques (les concepts subsumés) vers les concepts les plus généraux (les concepts subsumant) représentant les catégories de haut niveau, lors de l'identification des concepts qui se correspondent dans diverses ontologies par exemple en comparant les noms et les définitions en langage naturel de deux concepts et en contrôlant la proximité de deux concepts dans l'hiérarchie de concepts.

III.3.3.2 La technique « Midle-Out »

Il s'agit d'une approche où les concepts les plus spécifiques ou les plus importants sont définis en premier puis sont affinés ou spécialisés, tout en gardant le contrôle du niveau de détail qui ne peut être contrôlé avec l'approche « Bottom-Up ».

Contrairement à l'approche « Top-Down », dans cette approche, les concepts génériques sont définis en fonction des concepts majeurs supposés les plus stables augmentant ainsi la stabilité des hiérarchies

III.3.3.3 La technique « Top-Down »

Dans cette approche sont les concepts les plus génériques qui sont définis en premier et puis sont affinés ou spécialisés.

En fonction de la nature de l'application et des besoins spécifiés de la fusion une de ces trois techniques est choisie.

III.3.4 Les problèmes de la fusion d'ontologies

Selon [KLE 01] et [PRE et al, 05], la fusion d'ontologies peut être gênée ou perturbée par plusieurs types de problèmes dont le mismatch (qui est lui-même de différents types) constitue le problème le plus commun. La Figure ci-dessous résume les problèmes en face d'un processus de fusion d'ontologies:

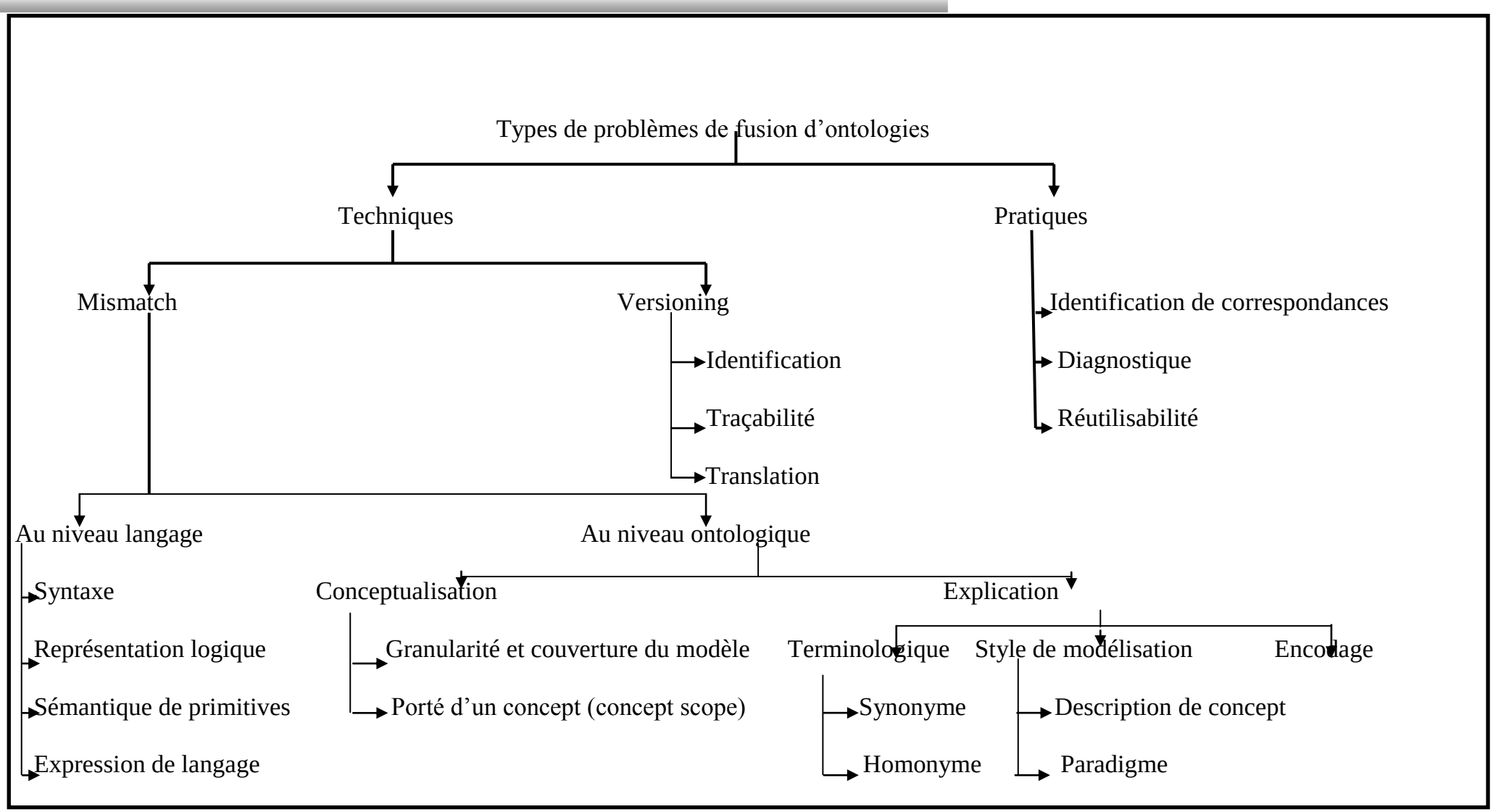


Figure III.4: Les différents types de problèmes de fusion d'ontologies

III.3.4.1 Les problèmes techniques

III.3.4.1.1 Mismatches entre ontologies:

Plusieurs types de mismatches entre ontologies sont constatés dans la littérature des ontologies et sont classifiés comme suit:

a- Mismatches au niveau langage (méta model level): Peuvent apparaître lorsque les ontologies sources sont écrites sous différents langages de représentation, et décrivent les mismatches entre les mécanismes décrivant les classes et les relations et reflètent les différences non sémantiques entre les ontologies sources, et eux mêmes peuvent être de différents niveaux:

- **La syntaxe:** Il est évident que différents langages de représentation d'ontologies utilisent différents syntaxes. Par exemple, pour définir le concept *« chaise »* en RDF: <*rdfs:Class ID = « chaise »*>, et dans LOOM: *(defconcept chaise)* définit la même classe, et une simple translation des ontologies dans un langage de représentation commun peut résoudre le problème.
- **Représentation logique:** apparaît lorsque des représentations logiques qui sont syntaxiquement différentes mais logiquement équivalentes sont utilisées pour représenter le même objet, tel que l'expression de la disjonction peut être faite par *(disjoint A B)* mais aussi par *(A subclass-of (not B), B subclass of (not A)). C*e type de mismatch est relativement facile à résoudre. En effectuant par exemple une translation des règles d'une logique de représentation vers une autre.
- **Sémantique de primitives:** apparaît lorsque des expressions sont syntaxiquement équivalentes mais sémantiquement différentes (de différents langages de représentation). En d'autres termes, même les deux ontologies utilisent la même syntaxe, leurs sémantiques peuvent être différentes, telle la syntaxe OIL-RDFSchéma interprète: <*rdfs:domain*> comme l'intersection des arguments, alors que RDFSchéma l'interprète comme l'union des sémantiques des arguments.

Ce problème peut aussi être résolu lors de la translation dans une représentation commune.

- **Expressivité du langage:** il s'agit de la différence d'expressivité entre deux langages qui implique qu'un objet est exprimable dans un langage et non pas dans un autre, tels certains langages détiennent des constructeurs pour exprimer la négation, supporter la notion d'ensemble, de valeurs par défauts, etc alors que d'autres non. Ce problème peut être résolu par translation des ontologies sources à un langage de représentation commun, mais si ce dernier ne fait pas partie des langages de représentation des ontologies sources certaines sémantiques peuvent être perdues.

b- Mismatches au niveau ontologique (model level): peuvent apparaître lorsque les ontologies sont écrites sous le même langage de représentation, mais en plus lorsque ces derniers sont différents, décrivant ainsi la différence de la façon par laquelle le domaine est modélisé, et reflétant donc les différences sémantiques entre les ontologies sources. Ce niveau de mismatch est aussi de plusieurs types ou classifications:

- **au niveau de la conceptualisation:** décrivant la différence des façons de conceptualisation ou d'interprétation du domaine donnant ainsi différents concepts ontologiques et/ou différentes relations entre eux:
  - **Différence de l'extension du concept (différence dans l'ensemble des instances « scope concept »):** Lorsque deux classes semblent représentant le même concept, mais n'ayant pas exactement les mêmes instances (class mismatch).
  - **Granularité et couverture du modèle:** décrivant la différence dans la partie du domaine couvert par l'ontologie et/ou le niveau de détail par lequel le domaine est modélisé. (par exemple dans un domaine de véhicule, une ontologie modélisant les voitures mais non pas les camions, et une autre ontologie modélisant les camions dans peu de catégories, et une troisième ontologie modélisant ces camions par plusieurs catégories).

Le problème de différence de conceptualisation ne peut être résolu automatiquement mais nécessite la décision et l'expertise d'un expert du domaine.

- **Au niveau explicatif:** décrivant la différence de la façon par laquelle la conceptualisation est spécifiée engendrant ainsi des différences de définitions, de termes, et de combinaisons des deux. Une classification des types de différences au niveau explication est comme suit:

  - **Selon le style de la modélisation:**

    a- Paradigme de représentation: Différents paradigmes peuvent être utilisés pour représenter les concepts, tel que temps, action, plans, causalité, attitudes propositionnels, etc. Par exemple un modèle représentant le temps par des intervalles de temps, alors qu'un autre le représente par des points, un autre exemple est l'utilisation de différents niveaux (top-level) d'ontologies.

    b- Description de concepts (convention de modélisation): plusieurs choix peuvent être pris pour modéliser les concepts dans une ontologie, telle la distinction entre deux classes peut être fait soit par l'utilisation des attributs de qualification soit par l'utilisation des séparateurs de classes. Un autre choix de description de concepts est sur comment les hiérarchies « is-a » sont construites, où la distinction des concepts (leurs attributs) peut être faite plus haut ou plus bas dans l'hiérarchie. Par exemple, dans l'assertion suivante, pour différencier entre publication scientifique et non scientifique, une dissertation peut être modélisée comme suit:

    Dissertation: <*book*<*scientific publication*<*publication*, ou encore

    Dissertation: <*scientific book*<*book*<*publication* (ou encore comme une sous classe des deux: book et scientific publication)

  - **Mismatch terminologique:**

    a- Synonymes (term mismatchs): Deux termes sont synonymes s'ils sont sémantiquement équivalents mais représentés par des noms différents. Pour résoudre ce problème, il est recommandé d'utiliser des thésaurus, ou des dictionnaires, mais il faut faire attention aux différences d'extensions possibles.

b- Homonymes (concept mismatchs):

Ce problème apparaît lorsque des concepts sémantiquement différents portent le même nom.

- **Encodage:** Des valeurs dans différentes ontologies peuvent être codées par différents formats, tel que la date peut être représentée par « dd/mm/yyyy » ou par « dd-mm-yyyy », une distance peut être représentée en kilomètres ou en miles.

III.1.4.1.2 Versioning d'ontologies:

Lors d'un processus de versioning, les exigences suivantes sont imposées sur un schéma de versioning (avec difficultés croissantes):

a- Identification: Pour chaque utilisation d'un concept ou d'une relation, le cadre du versioning doit fournir une référence non ambiguë pour étendre une définition.

b- Traçabilité: Le cadre du versioning doit établir une relation entre une version d'un concept ou d'une relation et une autre version de ce concept ou de cette relation.

c- Translation: Le cadre du versioning doit établir automatiquement des translations (conversions) d'une version à une autre pour permettre un accès transparent à la dernière version.

III.3.4.2 Les problèmes pratiques

En plus des problèmes techniques, il existe des problèmes pratiques qui gênent la facilité d'utilisation des ontologies fusionnées tel que:

III.3.4.2.1 La découverte de correspondances:

Il est donc difficile d'identifier les termes qui vont être appareillés ensemble.

III.3.4.2.2 Diagnostique:

Les conséquences ou les résultats d'un appareillement spécifique sont difficiles à diagnostiquer (ou à se voir).

III.3.4.2.3 La réutilisabilité:

Les ontologies sources utilisées pour la fusion continuent à évoluer, les correspondances crées pour la fusion doivent être le plus possible réutilisables pour fusionner les ontologies révisées.

III.3.5 La fusion d'ontologies et la fédération de BDDs

Le processus de la fusion d'ontologies comprend une comparaison entre au moins deux ontologies de sujets ou de domaines similaires. La fusion peut être faite à travers l'identification de ressemblances et/ou de dissemblances des niveaux structurels (les concepts et leurs relations) et syntaxiques (format de nommage), et le challenge principal est donc de consulter les résultats sémantiques lors du processus de la fusion.

Il est à noter que la littérature des ontologies focalise sur la détection des relations entre les spécifications sans prendre en considération les conceptualisations.

Le problème de la fusion d'ontologies est donc très similaire au problème de la fédération des BDDs qui a émergé quand les systèmes d'informations deviennent *ubiquitaires (pervasifs). E*n industrie, et en même temps quand les réseaux de communication des données devenaient très vastes et reliés entre eux, la fédération des BDDs rend donc le modèle de l'interface « client-serveur » très pratique aux utilisateurs pour leurs accès à plusieurs systèmes d'informations, rendant ainsi la vie dans le domaine industriel plus facile aux utilisateurs à travers leur représentation par une seule BDD fédérée ayant une méthode d'accès et un langage de requêtes unique, vaut mieux qu'une collection de BDDs avec différentes méthodes d'accès et langages de requêtes. Le problème de la fédération de BDDs est posé comme un problème de conception des versions intrinsèques des schémas de BDDs ayant comme un problème majeur (comme dans le cas de la fusion d'ontologies) l'hétérogénéité sémantique.

III.3.6 Un langage et un algorithme de fusion d'ontologies

Pour approfondir la compréhension de la notion de fusion d'ontologies, nous adoptons pour un apprentissage par l'exemple en s'inspirant du travail de A. D. C. Razgado et A. Guzman-Arenas [RAS et al, 08] de l'institut national polytechnique du

Mexique dans un travail intitulé « Un langage et un algorithme de fusion automatique d'ontologies », (« A language and an algorithm for automatic merging of ontologies »), où il proposent une nouvelle méthode qui permet de fusionner automatiquement deux ontologies pour obtenir une troisième plus large et plus complète. Cette méthode a donné de bon résultats malgré les inconsistances, les redondances, et les différents niveaux de granularité d'informations tout en comblant les lacunes rencontrées par les langages de représentation les plus connus (DAML+OIL, RDF et OWL) à travers un nouveau langage de représentation d'ontologies appelé OM, (pour Ontology Merging notation) qui vont être fusionnées automatiquement en exploitant un algorithme de fusion automatique appelé aussi OM (pour Ontology Merging algorithm).

Parmi les lacunes des langages de représentation courant (DAML+OIL, RDF et OWL) résolu par OM, on trouve:

- Une relation ne peut être un concept: Par exemple si « forme » est un concept et « couleur » est une relation, il serait difficile de relier « couleur » et « forme » par une autre relation.
- Les partitions (des sous ensembles avec des propriétés particulières) ne sont pas présentables.

Donc dans ce qui suit, nous synthétisons ce travail à travers la représentation du support de connaissances pour l'algorithme OM, le langage de représentation OM, ainsi que ses apports par rapport aux langages de représentation courant (DAML+OIL, RDF et OWL). Puis nous décrivons l'algorithme OM qui se base sur la fonction COM (COMparaison function), pour dégager les concepts similaires entre les deux ontologies tout en présentant la notion de confusion qu'il peut confronter, ainsi que ses apports par rapport aux autres outils présentés dans le chapitre précédent. Nous terminons par un ensemble d'exemples de fusion d'ontologies en utilisant l'algorithme OM ainsi que ses domaines d'application les plus favorisés.

III.3.6.1. Le support de connaissances pour OM

L'algorithme OM applique certains principes permettant de détecter les contradictions, les synonymes, les homonymes, etc, tel que:

- Les prépositions, les suffixes, les préfixes (stop words), tel: dans, le, la, pour, ce, cette, et, ou, etc, sont ignorés des mots de phrases.
- Les mots modifiant le sens des relations, tel que sans, à part, sont pris en compte.
- Les hiérarchies représentant des taxonomies de concepts reliés sont utilisées pour mesurer la confusion et détecter les synonymes.
- L'extraction des racines de mots est basée sur des ressources linguistiques tel que WordNet, l'utilisation de dictionnaires pour dégager les synonymes, les homonymes, etc.

III.3.6.2. Le langage OM

Le langage de représentation OM représente une ontologie à travers une structure arborescente spécifiée en XML, où les concepts et les relations sont identifiés par des labels qui vont entre deux crochets identifiant le début et la fin de la désignation du concept ou de la relation, tels que:

<concept> thing </concept>, <language> english </language>,

<relation> eats </relation>. Ces relations peuvent être implicites (non déclarées dans la structure), ou explicite (explicitement déclarées dans l'hierarchie).

III.3.6.3. Les apports du langage de représentation OM par rapport aux autres langages:

1/ La possibilité de représentation des partitions.

2/ Une relation peut être un concept.

3/ La possibilité de représentation des relations n-aire.

III.3.6.4. L'algorithme OM pour la fusion automatique d'ontologies:

Il a pour objectif de fusionner deux ontologies A et B pour obtenir une troisième ontologie C contenant toutes les informations contenues dans A en plus de toutes les informations contenues dans B et non contenues dans A évitant ainsi tout risque de redondances ou de contradictions, une information ajoutée à partir de B peut donc être:

- Un nouveau nœud: représentant une information manquante dans A.

- Une nouvelle relation: tel que l'ontologie B permet d'ajouter que « Gabriel Garcia Marquez » a écrit « The colonel has no body to write to him », en plus de l'information acquise à partir de A qui indique: «that he live one hundred years of loneliness ».
- Une relation prouvée et plus précise: par exemple A apporte que « Abraham Lincoln was born in united states », alors que B précise « that he was born exactly in Kentucky ».
- Un nouveau synonyme dans B d'un nœud courant dans A permettant d'enrichir C.
- Une relation mieux définie dans B que dans A.

III.3.6.5. Les apports de l'algorithme OM:

1- OM est totalement automatique. Il ne demande aucune intervention humaine.
2- Il permet de représenter les partitions et les sous ensembles.
3- Il permet de représenter les concepts définis même superficiellement à travers un mot, une phrase de mots ou une combinaison de phrases.
4- Une relation peut être un concept.
5- En s'aidant avec la fonction de comparaison COM, l'algorithme OM prend en compte: Les synonymes, les homonymes, les synonymes de point de vue des propriétés ainsi que les nouvelles connaissances.
6- OM évite d'assembler des relations redondantes.
7- L'algorithme OM détecte les inconsistances entre les connaissances des deux ontologies.
8- Il permet l'élimination des valeurs redondantes.

III.3.6.6. Cas d'utilisation réels de l'algorithme OM:

Cuevas-Rasgado a appliqué en 2006, l'algorithme OM pour fusionner des ontologies dérivées à partir de documents web concernant différents domaines d'application à savoir:

- Les zones géographiques: Deux documents différents concernant Oaxaca.
- Des animaux et des fleurs: Deux descriptions de turtles, et deux de poppies.

- Biographies: Deux concernant Benito Juarez, et deux concernant Neoton.
- Descriptions d'outils et de produits.
- Romans: Des portions de 100 années d'isolément (2 textes différents).

A partir de ces documents, les ontologies sont manuellement écrites dans la notation OM, obtenant ainsi deux ontologies pour chaque animal, fleur,…, puis chaque paire d'ontologies est automatiquement fusionnées par l'algorithme OM, une validation de ces résultats est faite en les comparant avec ceux obtenues avec une fusion manuelle et est résumée dans le tableau 1.

ONTOLOGIES	Temps	Relations	concepts	Erreur	efficacité
Turtles	4 sec	6(B)U8(A)=10 (C) toutes les relations sont correctement copiées	35(B)U29(A)=35(C) tous les noeuds sont correctement copiés	0	100%
Hammer	6 sec	30(B)U8(A)=36(C) toutes les relations sont correctement copiées	33(B)U24(A)=51(C) tous les noeuds sont correctement copiés	0	100%
Poppy	14 sec	20(B)U21(A)=37(C) toutes les relations sont correctement copiées	35(B)U34(A)=58(C) tous les noeuds sont correctement copiés	0	100%
100 anos de soledad	10 sec	283(B)U231(A)=420(C) 432 (-12 from 432) 12 de 432 relations ont incorrectement enlevées de C.	126(B)U90(A)=141(C) 149 (-8, 149), 8 de 149 concepts sont incorrectement enlevés	0.034	96%
Oaxaca	5 sec	43(B)U61(A)=96(C) toutes les relations sont correctement copiées	117(B)U234(A)=309(C) 310 (-1, 310) 1 de 310 concepts sont incorrectement enlevés	0.002	99.7%
Neurotransmeteur	2 sec	79(B)U51(A)=127(C) 129 (-2 from 129) 2 de 129 relations ont incorrectement enlevées de C.	56(B)U26(A)=77(C) 79 (-2, 79) 2 de 79 concepts sont incorrectement enlevés	0.019	98%
Ontologies inconsistantes	1 sec	3(B)U4(A)=7(C) 1 de 2 inconsistances sont résolues	5(B)U6(A)=9(C) 5 (C5), 5 inconsistances sont incorrectement résolues	0	100%
Tourisme d'Acapulco	20 sec	60(B)U64(A)=131(C) toutes les relations sont correctement copiées	61(B)U65(A)=124(C) tous les noeuds sont correctement copiés	0	100%
multimedia	10 sec	7(B)U6(A)=10(C) toutes les relations sont correctement copiées	8(B)U7(A)=11(C) tous les noeuds sont correctement copiés	0	100%

Table III.1: Performance de l'algorithme OM dans quelques exemples réels

A titre d'exemple, la première ligne de la table est lue comme suit: Turtles dit que OM fusionne les deux ontologies pendant 4 secondes. En manipulant 8 relations dans l'ontologie A et 6 relations dans l'ontologie B donnant ainsi 10 relations dans l'ontologie C, et toutes les relations étaient correctement copiées (même résultat que la fusion manuelle). Pour les concepts, OM fusionne 29 concepts de l'ontologie A, 35 concepts dans l'ontologie B donnant 35 concepts dans l'ontologie C, où tous les nœuds sont correctement copiés (même résultat que la fusion manuelle). L'erreur est ainsi calculée: relations + concepts faussement copiés dans C par OM/ relations + concepts correctement copiés dans C par OM = 0+0/10+35=0 → ERR=0.

Et idem pour l'efficacité:

efficiency = 100 * (relations + concepts correctement copiés dans C par OM / relations + concepts dans C obtenus manuellement)

= 100*(10+35/10+35) = 100%

III.3.7. D'autres outils de fusion d'ontologies

III.3.7.1 PROMPT

Il s'agit d'un plugin ou d'une extension de l'environnement de développement d'ontologies *Protégé 2000* qui constitue un outil interactif orienté utilisateur ayant pour objectif de guider l'utilisateur à fusionner des ontologies modélisées à travers le langage de représentation OKBC, basé sur les frames (voir chapitre 2), qui lui propose des suggestions de fusion (ToDo List). Puis l'utilisateur lance une des opérations soit à partir de la liste des propositions suggérée par PROMPT, ou en utilisant l'environnement d'édition d'ontologie pour spécifier directement l'opération voulue.

Ensuite, PROMPT exécute automatiquement l'opération lancée par l'utilisateur, tout en adaptant les changements additionnels basés sur le type de l'opération, détermine les conflits introduits par cette opération (Conflict List), tout en proposant des solutions et générer une liste des suggestions à l'utilisateur en fonction de la nouvelle situation et de la structure courante de l'ontologie, voir la figure ci-dessous.

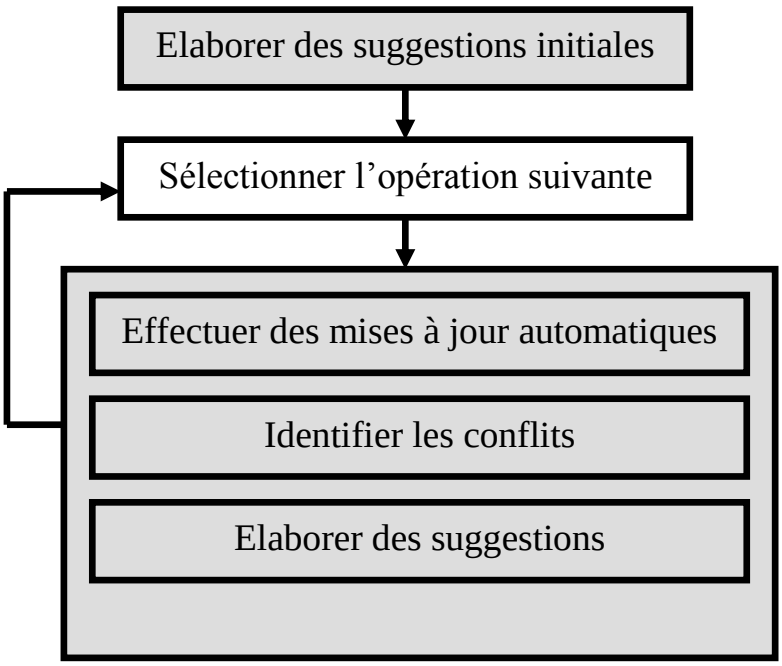


Figure III.5: L'algorithme Prompt, les boites grillées correspondent à des actions effectuées par Prompt et les autres à ceux effectuées par l'utilisateur.

Selon [NOY et al,00]et [KLE,01], pour chaque opération dans la liste ToDo, les opérations suivantes sont définies:

1. Déterminer les changements effectués automatiquement par PROMPT.
2. Déterminer les nouvelles suggestions que PROMPT présente à l'utilisateur.
3. Déterminer les conflits introduits par cette opération et qui doivent être résolus par l'utilisateur.

L'ensemble des opérations de fusion identifiées par [NOY et al,00] est constitué de:

- Fusion de classes, fusion de slots et fusion de liaisons entre un slot et une classe (fusion d'opérations d'affectation d'un slot à une classe),

- Copier profondément une classe d'une ontologie à une autre (copier la classe ainsi que toutes ses classes parentes jusqu'à la racine ainsi que toutes les classes référencées par cette classe)

- Copier superficiellement une classe d'une ontologie à une autre (copier seulement la classe mais pas ses parents ni ses classes référentielles).

Alors que la liste des conflits pouvant être introduits par ces opérations est constituée de:

- Conflit de nom (plus d'une frame avec le même nom).

- Références perdues (une frame qui réfère à une autre frame inexistante).

- Redondance dans l'hiérarchie d'une classe (plus d'un chemin à partir d'une classe vers un parent autre que la racine).

- Restriction sur la valeur du slot empêchant l'héritage de classe.

III.3.7.2 Chimaera

Il s'agit d'un outil de fusion et de diagnostique d'ontologies, développé au sein du laboratoire KSL (Knowledge System Laboratory) à l'université de Standford. L'utilisateur interagit avec Chimaera à travers un navigateur web tel que NetScape ou microsoft internet explorer, via une interface utilisateur basée web. (Voir figure 6)

Figure III.6: L'interface utilisateur de Chimaera.

Chimaera est initialement développé pour assister la fusion de différentes bases de connaissances produites par plusieurs auteurs. Puis il vise à supporter le test et le diagnostique des ontologies. Finalement, il assiste la fusion ainsi que le diagnostique d'ontologies pour éditer et naviguer les ontologies sur le web. Selon [KLE,01] et [LAM et al, 03], les deux tâches principales offertes par chimaera sont:

1. Fusionner deux termes sémantiquement identiques à partir de différentes ontologies de façon à être référencés par le même nom dans l'ontologie résultat.
2. Identifier les termes qui doivent être reliés par des relations de subsomption, de disjonction ou par des relations entre les instances tout en offrant un support pour introduire ces relations. Chimaera dispose aussi d'un support pour vérifier, valider et critiquer des ontologies.

Pour assister l'utilisateur, Chimaera génère des listes de résolution de noms à travers lesquelles, il propose des termes candidats à être fusionnés ou à avoir des relations d'hiérarchie pas encore incorporées candidats à être reconnu.
A la base de ces listes, l'utilisateur décide ce qu'il doit faire.

III.3.7.3 HCONE

Pour fusionner deux ontologies sources en entrée, HCONE commence par l'identification des s-morphism (Sémantic-morphism), entre chacune de ces ontologies sources et une ontologie intermédiaire cachée (Hidden Intermediate ontology), en appliquant la méthode d'indexation sémantique cachée, LSI (Latent Semantic Indexing) et faire correspondre les concepts de l'ontologie (intermédiaire) avec les sémantiques de WordNet qui joue le rôle d'un intermédiaire où les concepts des ontologies sources seront appareillés à travers les s-morphism. HCONE construit donc l'ontologie intermédiaire lors de l'appareillement des concepts des ontologies sources avec les sémantiques de WordNet, où il assume qu'elle se situe, et qu'elle comprend [KOT et al,05]:

1. Un vocabulaire avec un lexique des sens spécifiques des ensembles de synonymes de WordNet correspondant aux concepts des ontologies.
2. Les axiomes traduits des ontologies sources.

Donc, après l'identification des appareillements avec l'ontologie intermédiaire et la traduction les ontologies sources (O^1 et O^2), ces dernières seront alignées puis fusionnées à travers les actions de renommage, de fusion et de classification.

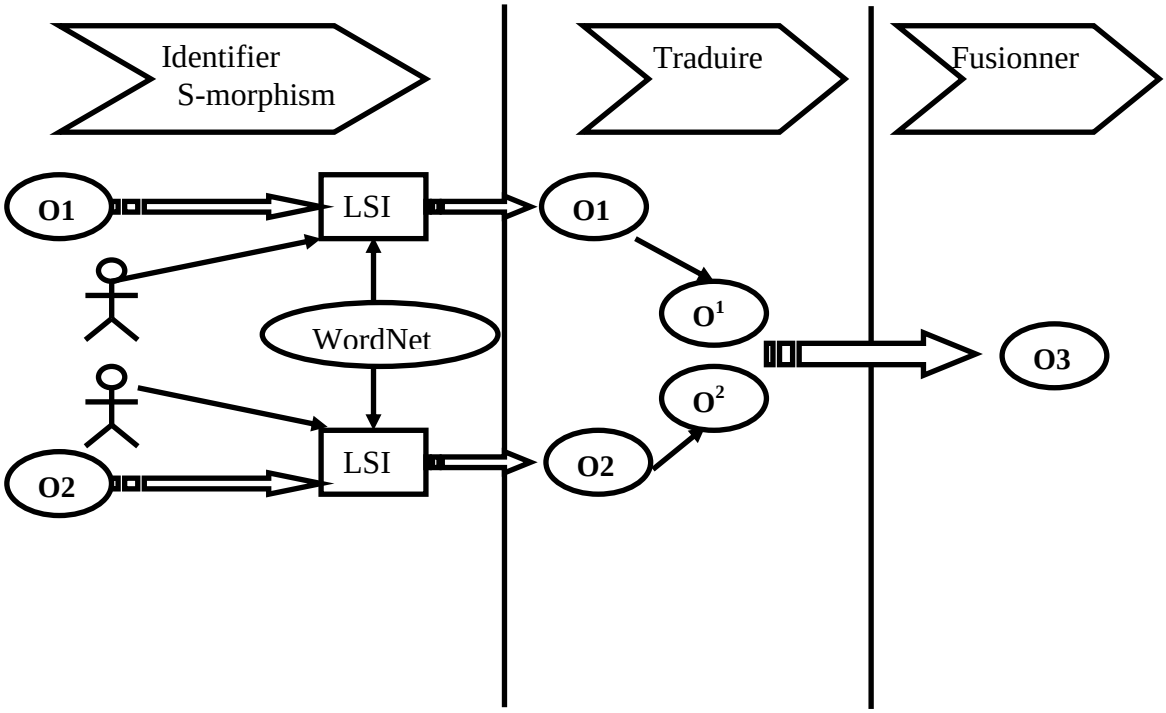


Figure III.7: L'approche HCONE.

III.3.7.4 OntoMerge

Il s'agit d'un outil de fusion et de raisonnement automatique où les parties qui se correspondent dans les deux ontologies sont reliées à travers ce qu'on appelle «les axioms Bridges », (qui sont spécifiés en logique de premier ordre), les concepts ou les termes des ontologies sources sont distingués à travers les différences dans l'espace de nom. L'ontologie résultat de la fusion risque donc d'utiliser différents espaces de noms dans ses définitions. Cela peut engendrer une confusion à l'utilisateur. Alors, pour contourner ce problème, l'ontologie résultat de la fusion par OntoMerge comprend plusieurs fatras où chacun est composé de:

1. Des termes avec différents espaces de noms.
2. Les termes similaires ou équivalents dans l'ontologie résultat.
3. « Des axioms Bridge » entre les termes reliés.

OntoMerge permet à travers la fusion d'ontologies de traduire les ensembles de données, générer une extension de l'ontologie et interroger différentes ontologies.

III.3.7.5 FCA-Merge

Il s'agit d'un outil de fusion d'ontologies basé sur l'analyse des concepts formels et suit l'approche de fusion « Bottom-Up » qui, à partir d'un ensemble de documents en entrée, extrait les instances des ontologies correspondantes à travers une analyse linguistique. Puis génère automatiquement les concepts choisis, et enfin crée interactivement avec l'utilisateur (donc semi-automatiquement) l'ontologie de la fusion.

III.3.8. Quelques applications sur la technique de la « fusion d'ontologies »:

III.3.8.1 Une nouvelle approche de fusion d'ontologies [MOH et al, 05]

En résumé, l'algorithme de la fusion appliqué par cette approche peut être résumé dans les étapes suivantes:

D'abord, pour chaque agent, une ontologie est crée, en langage naturel, par l'utilisateur non expert en ontologies indépendamment des autres agents et utilisateurs. Puis, les agents dont les ontologies sont du même domaine, sont exposés l'un à l'autre, échangent leurs ontologies, et décident de les fusionner. Ensuite, tous les concepts qui sont sémantiquement ou syntaxiquement similaires à ceux dans l'ontologie de l'agent courant sont identifiés:

1. La similarité syntaxique est donnée par l'algorithme de la programmation dynamique basé sur la distance d'édition de Levenshtein.
2. La similarité sémantique est identifiée à partir des synsets du dictionnaire sémantique, WordNet.

Après cela, ces concepts similaires (ainsi que toutes leurs relations) sont appris par l'agent courant puis sont ajoutés à son ontologie. Et en fin, tous les autres concepts (non appris par cet agent) ainsi que leurs relations sont identifiés, appris et incorporés dans l'ontologie résultat de la fusion.

III.3.8.2 Fusion des ontologies de maintenance [VIZ et al,07]

Dans cette application, A.Vizcaino et al, proposent de fusionner deux ontologies complémentaires du même domaine de maintenance, où l'une est

développée par Dias et al en 2003, identifiant les connaissances nécessaires lors d'un processus de maintenance et composée de cinq sous ontologies (*system, skills, modification process, orgnizational structure* and *application domain* subontologies). L'autre est proposée par Ruiz et al en 2004 occupée de la gestion des projets de maintenance et constituée de quatre sous ontologies (*products, activities, process* and *agents* subontologies). Cette méthode est inspirée de la méthode PROMPT. Elle consiste tout d'abord à lister les concepts de toutes les ontologies sources dans une liste, facilitant ainsi d'identifier les concepts communs dans les deux (ou plus) ontologies sources, et ceux (les concepts) existant dans une seule ontologie source.

Ensuite elle vise à déterminer quels termes ayant le même nom (label) dans les deux ontologies, mais présentant différents concepts, et quels termes ayant des noms différents dans les deux ontologies mais représentant le même concept (synonymes). Là le développeur d'ontologie doit décider quel concept doit être pris en compte, et quel concept doit être enlevé. Enfin et lorsque la nouvelle ontologie est construite elle consiste à détecter s'il y a des redondances ou des conflits. Les auteurs de cette approche proposent de fusionner itérativement les sous ontologies au lieu de fusionner les ontologies entières.

III.3.8.3 La fusion des deux ontologies TOVE et ENTREPRISE dans une optique PLM [MOS 01]

Dans cette application, l'auteur a proposé de fusionner les deux ontologies TOVE et ENTREPRISE, représentatives du PLM (Product Lifecycle Management, pour la Gestion de Cycle de vie d'un Produit), en utilisant l'algorithme de fusion PROMPT:

Après faire entrer les deux ontologies sources et choisir la fusion comme type d'intégration sous PROMPT, ce dernier suggère une liste de propositions (ToDo list) Cette liste est donc recommandée par PROMPT et soumise à l'acceptation de l'utilisateur. Ce dernier les examine une par une et fait agréer ou non leur exécution, et enfin une nouvelle ontologie de fusion est obtenue (voir Figure 8). Elle illustre une des étapes du processus de la fusion, où le concept *process* de l'ontologie TOVE et le même concept *process* de l'ontologie ENTREPRISE sont fusionnés ensemble sous le

même nom (voir la partie droite de Figure 8), et ainsi de suite, le processus de fusion continue donc jusqu'à la complète satisfaction de l'utilisateur.

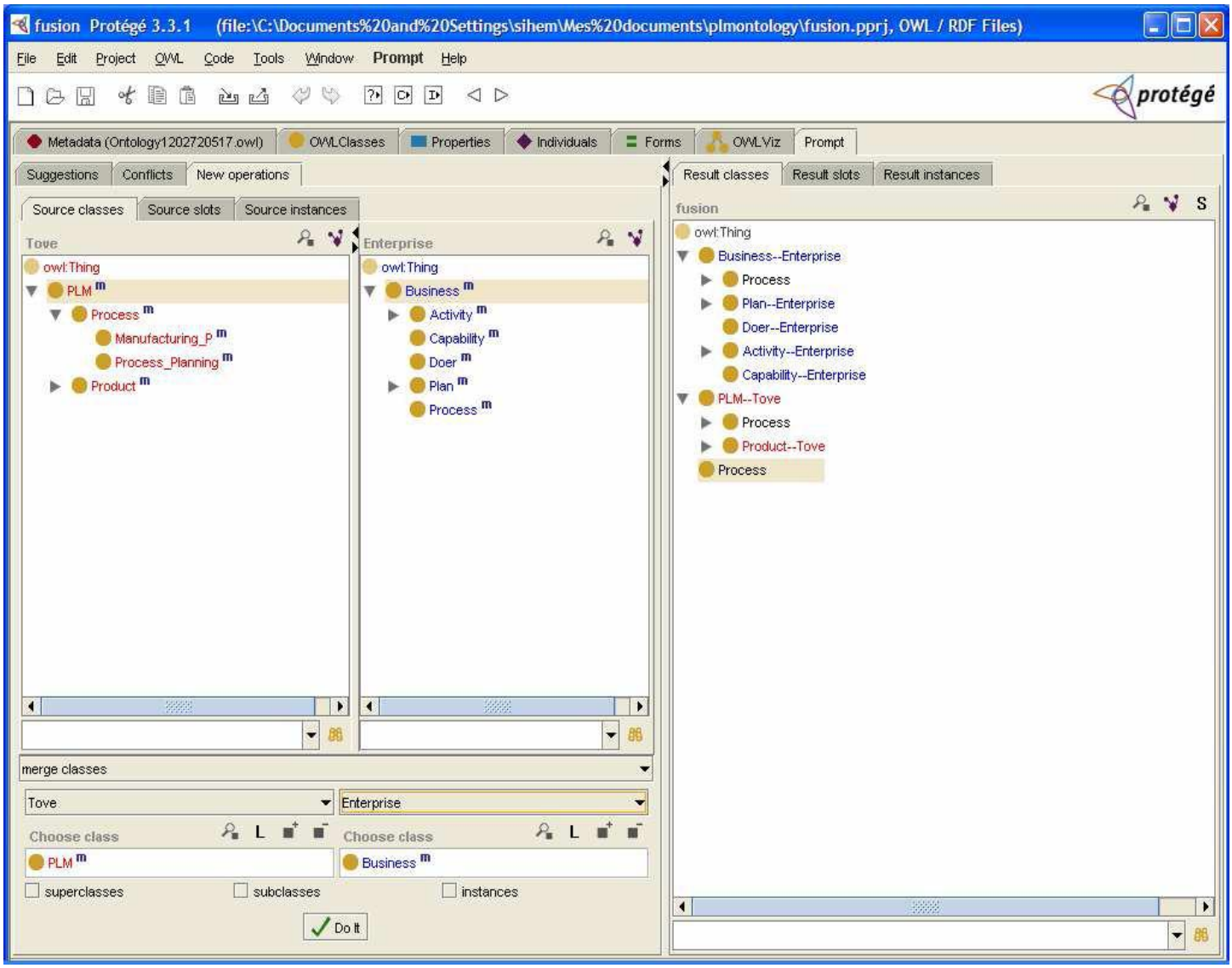


Figure III.8: Une étape dans le processus de fusion sous PROMPT [MOS, 01]

III.4. Conclusion

L'émergence de la technique de fusion d'ontologies est un moyen par lequel on surmonte les restrictions et les spécificités des informations et des connaissances lorsqu'il s'agit d'une application qui touche deux (ou plus) domaines similaires.

Dans ce chapitre nous avons donc exploré les notions d'un processus de fusion d'ontologies, ses techniques, les différents types de problèmes pouvant gêner l'opération de la fusion, et nous terminons par la représentation des algorithmes de la

fusion d'ontologies les plus connus en plus de quelques exemples d'application de la technique « fusion d'ontologies » exposées dans la littérature.

IV.1 Introduction

Une turbine est un automate rotatif, qui génère une énergie mécanique lors de la détente d'un courant d'eau, de vapeur ou de gaz. Elle est constituée principalement d'un rotor (roue) à hélice, à lame, à aube ou à augets arrangés sur son pourtour, tel que le fluide afflué applique une force tangentielle faisant tourner le rotor et générant une énergie cinétique, qui à son tour faire tourner un moteur, un compresseur, un générateur ou une hélice. Selon la nature de ce fluide on distingue donc la turbine à eau, (ou hydraulique), la turbine à gaz ou la turbine à vapeur.

Dans ce travail, nous avons choisi d'utiliser la turbine à vapeur qui constitue le fruit de la collaboration de plusieurs chercheurs et ingénieurs à la fin du XIXe siècle. Parmi ces efforts nous citons la conception du Britannique Charles Algernon Parson qui a basé sur le principe de la séparation des étages à travers lesquels le vapeur se dilate produisant ainsi de l'énergie, et le Suédois Carl Gustaf Patrick de Laval qui a conçu des jets et des augets adaptés à une utilisation performante du vapeur dilaté.

Les turbines à vapeur sont utilisées dans les centrales de production d'énergie thermique (classique ou nucléaire). Elles sont également utilisées couramment avec le moteur Diesel pour la propulsion des navires.

Dans ce chapitre, nous allons commencer par un bref historique des apparitions des turbines à vapeur. Puis nous décrivons ce que qu'une turbine à vapeur, ses principaux composants, son principe de fonctionnement, ses caractéristiques, ses types et catégories et ses différentes applications.

IV.2 Historique

En 1883: La première turbine à vapeur a été construite en 1883 par le Suédois Carl Gustaf Patrick de Laval (1845-1913)[1].

En 1884: Le Britannique Charles Algernon Parson a construit une petite turbine à vapeur en substituant l'eau de la turbine à eau par la vapeur. Cette turbine de dix cheveaux multicellulaire tournait à 18000 tours par minutes.

En 1887: Gustaf de Laval a construisait une petite machine à vapeur pour prouver que de tels appareils peuvent être fabriqués dans de semblables dimensions.

1 http://www.adkarim.ifrance.com/lu/text.htm

En 1890 Gustaf de Laval développa une tuyère permettant d'augmenter la vitesse de la vapeur entrant dans la turbine. On la connaît de nos jours sous le nom de tuyère de, de Laval et elle présente une importance particulière dans la conception de fusées. Cette turbine à vapeur, à un seul disque, tournait à 30000 tours/minutes. Pour cela, il fallut surmonter un grand nombre de difficultés: Tracé et taillage des engrenages de précision, conception de nouveaux types de réducteurs à engrenage qui sont encore utilisés de nos jours, emploi d'une solution révolutionnaire à l'époque, qui consistait en l'utilisation d'un arbre de rotation suffisamment mince pour être flexible. Les efforts mécaniques considérables et les très grandes vitesses exigeaient des outils et des matériaux qui venaient à peine d'être créés[2].

IV.3. Définition de la turbine à vapeur

Comme tout autre type de turbine, il s'agit d'une installation industrielle de production d'énergie constituant une des technologies les plus anciennes et les plus répandues pour entraîner des générateurs. La turbine à vapeur est un moteur thermique à combustion externe. Elle est composée d'une partie fixe (Stator) et une partie mobile (Rotor) constituée d'un grand nombre (quelques dizaines) de roues portant des ailettes qui vont être traversées par la vapeur qui entre dans sa partie admission avec une température élevée (jusqu'à 680°C pour certaines machines) et une pression élevée (jusqu'à 250 bars pour les plus grosses turbines), générant ainsi un mouvement de rotation, puis s'échappe de sa partie échappement.

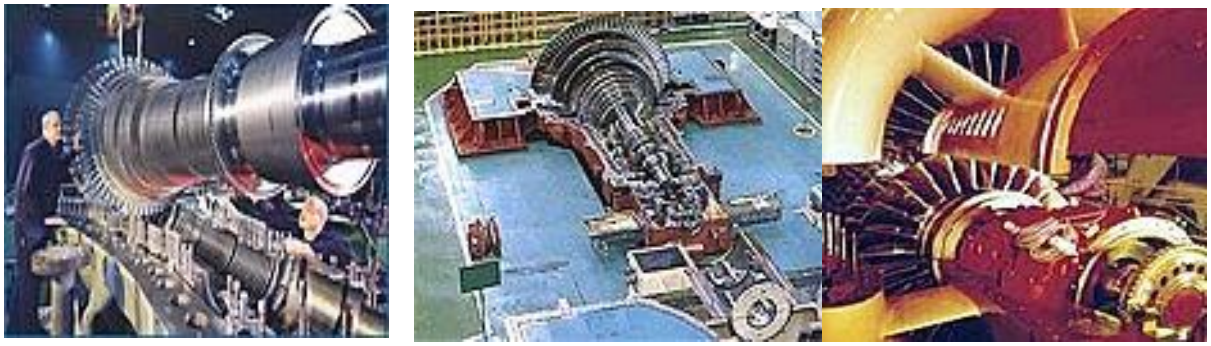


Figure IV.1Schéma illustrant quelques turbines à vapeur

2 http://www.encyclonova.com/index.php/Gustaf-de-Laval

IV.4. Les principaux composants de la turbine à vapeur[3]

IV.4.1 L'alternateur: L'alternateur est une machine électrique de type génératrice à courant alternatif qui transforme l'énergie mécanique en énergie électrique. Il est entraîné par la turbine.

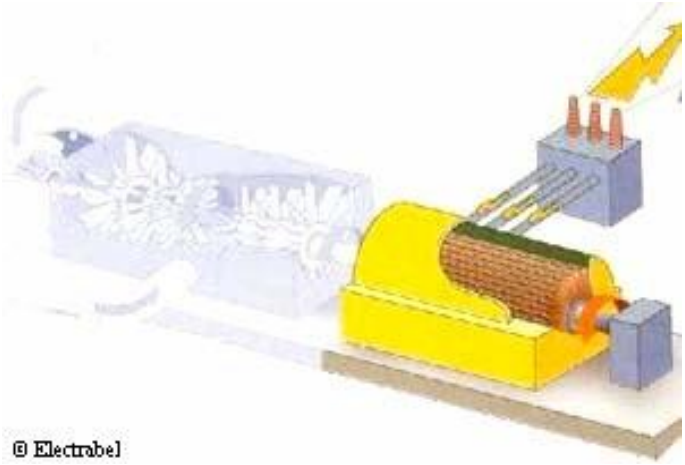


Figure IV.2: Schéma illustrant l'alternateur

IV.4.2. Les transformateurs: eux mêmes de deux types:

- 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): 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.

IV.4.3 La 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

3 http://www.retscreen.net/fr/steam_turbine_schematic.php

paramètres fixés. Il constitue la source chaude du cycle thermodynamique. Cette vapeur sera utilisée par la turbine pour fournir de l'énergie mécanique.

Figure IV.3: Schéma illustrant la chaudière

IV.4.4. Le condenseur: Afin de maximiser le rendement de la turbine à vapeur, la pression et la température de la sortie de vapeur doivent être aussi les plus 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.

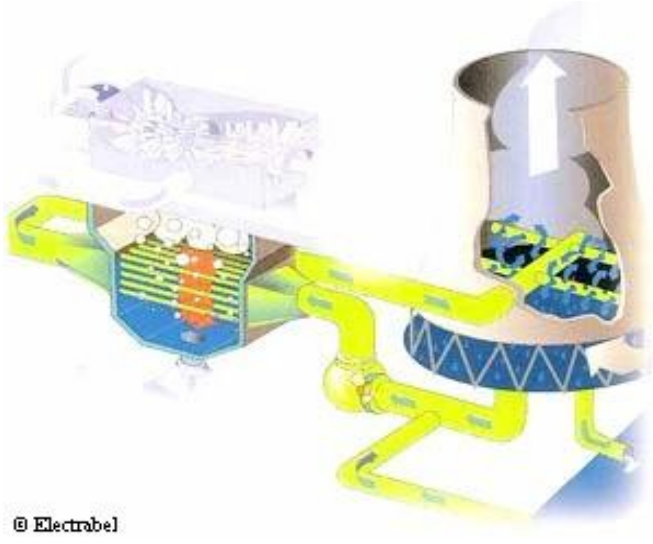


Figure IV.4: Schéma illustrant le condenseur

IV.4.5. La pompe alimentaire: La pompe KSB à très haute pression est une pompe à 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 nécessaire d'eau pour maintenir le niveau normal.

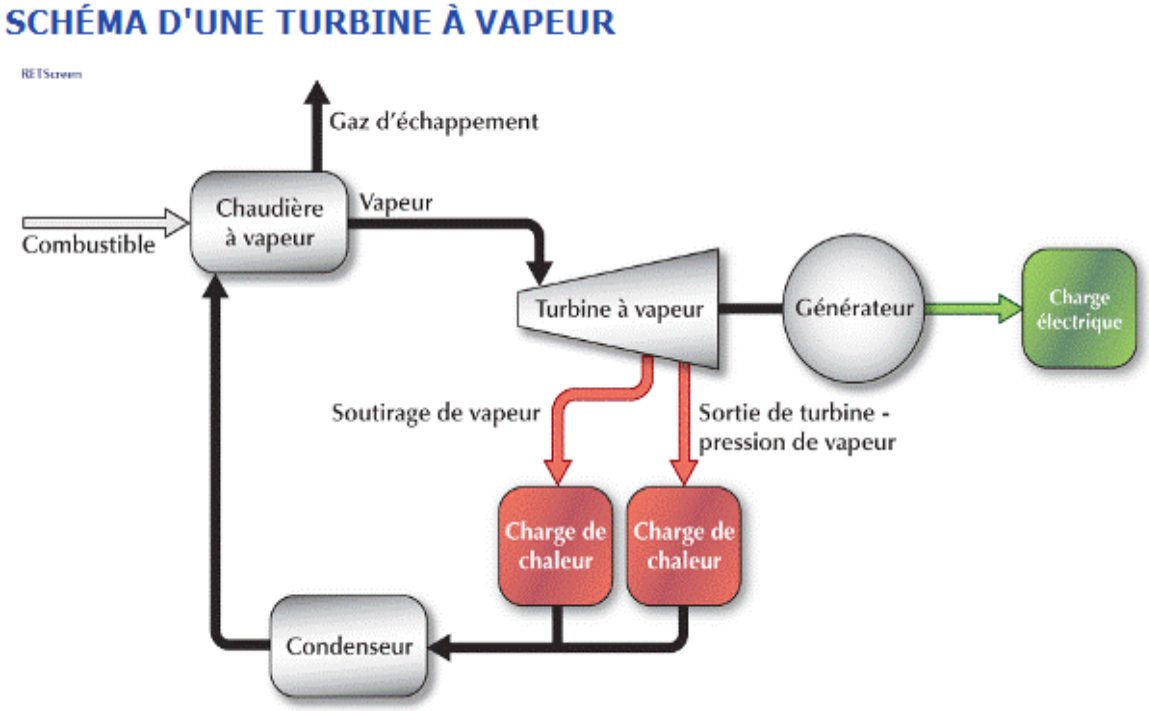


Figure IV.5: Schéma graphique de la turbine à vapeur[4]

IV.5. Réalisation pratique[5]

Une turbine est constituée d'un rotor comprenant un arbre sur lequel sont fixées des aubes et, d'un stator constitué d'un carter portant des déflecteurs fixes, généralement constitué de deux parties assemblées selon un plan axial. Elle comprend en outre un tore d'admission segmenté et un divergent d'échappement dirigé vers le condenseur. La fonction des déflecteurs fixes est d'assurer tout ou partie de la détente

4 http://www.directindustry.com/prod/ansaldo-energia/steam-turbine-29641-232047.html

5 http://www.techno-science.net

en formant un réseau de tuyères et de modifier la direction de l'écoulement sortant de l'étage précédent.

Figure IV.6: Schéma illustrant la réalisation pratique de la turbine à vapeur

Une turbine à vapeur comprend un ou plusieurs étages assurant chacun deux fonctions:

- La détente de la vapeur qui correspond à la conversion de l'énergie potentielle en énergie cinétique.
- La conversion de l'énergie cinétique en couple de rotation de la machine par le biais des aubages mobiles.

La réalisation des turbines nécessite le recours à des aciers fortement alliés (Cr-Ni-Va) pour résister aux contraintes thermiques, mécaniques (force centrifuge) et chimique (corrosion par la vapeur). Les deux premières contraintes limitent le diamètre et donc le débit capable des derniers étages. Ainsi des aubes de plus d'un mètre de longueur posent déjà de sérieux problèmes de réalisation. De plus,

l'hétérogénéité radiale des vitesses impose une incidence variable de l'aube qui présente alors une forme gauche dont l'usinage est complexe.

En pratique la température est limitée à 550 ou 580°C et le maximum mis en œuvre est de 650°C. La pression est de l'ordre de 180 bars et atteint 250 bars pour les installations supercritiques.

De ce fait, les turbines de forte puissance comprennent généralement sur un même axe (disposition tandem compound):

- Une turbine haute pression,
- Plusieurs (2 ou 3) turbines basse pression avec soutirages.
- Il est ainsi possible d'atteindre des puissances de plus de 1000 MW avec un rendement dépassant légèrement 40%.

À l'autre extrémité, les plus petites turbines ont des puissances de quelques dizaines de KW. Elles comprennent généralement un seul étage et servent à l'entraînement de machines dans l'industrie ou sur des navires. Entre les deux, existe toute une palette de turbines plus ou moins complexes et adaptées à des usages industriels spécifiques (à soutirage, à contre-pression, etc.).

IV.6. Principes de fonctionnement de la turbine à vapeur

Les turbines à vapeur sont des machines relativement simples dont la seule partie mobile importante est le rotor. Elles 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[6].

La turbine à vapeur fonctionne donc selon le cycle thermodynamique de Clausius-Rankine qui se distingue par le changement d'état affectant le fluide moteur qui est en général de la vapeur d'eau, et comprend les étapes suivantes[5]:

- L'eau liquide est comprimée par une pompe et envoyée vers la chaudière,
- L'eau est chauffée, vaporisée et surchauffée,
- La vapeur se détend dans la turbine en fournissant de l'énergie mécanique,
- La vapeur détendue est condensée au contact de la source froide sous vide partiel.

Le principe est donc le même que celui de la machine à vapeur à pistons. La turbine en constitue une évolution exploitant les principaux avantages des turbomachines à savoir:

- Puissance massique et puissance volumique élevée,
- Rendement amélioré par la multiplication des étages de détente.

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

6 http://fr.encarta.msn.com/encyclopedia_761563866/turbine.html
5 http://www.techno-science.net

d'une turbine auxiliaire intégrée au cycle thermodynamique de la turbine principale utilisant de la vapeur soutirée dans celle-ci[7].

IV.7. Caractéristiques de la turbine à vapeur[6]

IV.7.1. 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.

IV.7.2. 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 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%.

7 http://membres.lycos.fr/eaj/pde.html
6 http://fr.encarta.msn.com/encyclopedia_761563866/turbine.html

IV.7.3. 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.

IV.8. Types de turbines à vapeur

Selon la manière de la répartition de la détente de vapeur entre le rotor et le stator, on distingue les turbines à action (degré de réaction nul), où la détente du fluide est réalisé dans les aubages fixes ou tuyères en amont de la roue et les pressions en amont et en aval du rotor sont égales, et les turbines à réaction (degré de réaction égale à 0.5) où la détente est équi-répartie entre les tuyères du stator et la roue:

IV.8.1. Turbine à action:

La forme la plus simple de turbines 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 est conçue 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[6].

IV.8.2. 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

6 http://fr.encarta.msn.com/encyclopedia_761563866/turbine.html

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[6].

IV.9. Avantages et inconvénients

1. L'avantage majeur d'une turbine à vapeur, est qu'il s'agit d'un moteur à combustion externe permettant ainsi d'utiliser tous genres de combustibles (même les moins chers), tel que le gaz, le fuel, le charbon, le déchets, chaleur résiduelle pour la vaporisation, augmentant ainsi le rendement et réduisant les coûts de fonctionnements.
2. La turbine à vapeur génère directement un travail rotatif sans avoir besoin d'un vilebrequin ou de tout autre dispositif générant un mouvement rotatif à partir de mouvement de va-et-vient (contrairement à une turbine hydraulique).
3. Dans une centrale de cogénération nécessitant de la chaleur en plus de l'électricité, la vapeur est portée à haute pression dans une chaudière et extraite de la turbine à la pression et température exigées.

A côté de ces avantages, la turbine à vapeur présente quelques limites tel que:

1. Le coût et la complexité des installations industrielles croit proportionnellement en fonction de la puissance à générer (généralement les turbines à gaz sont mieux adaptées en dessus de 10 MW).

2. La restriction du domaine d'emploi aux installations fixes ou navales vu le volume nécessaire du fluide d'eau écoulé par unité de temps pour le refroidissement du condenseur.

6 http://fr.encarta.msn.com/encyclopedia_761563866/turbine.html

IV.10. La maintenance industrielle

IV.10.1. Définition de la maintenance

La maintenance est un ensemble d'actions tendant à prévenir ou à corriger les dégradations d'un matériel afin de maintenir ou de rétablir sa conformité aux spécifications. Pour une unité de production cette maintenance devrait permettre une contribution maximale à une meilleure efficacité et une meilleure rentabilité. Toutefois, elle a été jusqu'au années 1980, considérée par un grand nombre d'entreprises comme une activité non productive et par conséquent non stratégique. Au cours des vingt dernières années, la maintenance a considérablement évoluée. Actuellement, elle constitue l'un des vecteurs essentiels de compétitivité des entreprises [TAL et al, 03].

IV.10.2. Critères liés à la maintenance

Cinq critères de maintenance sont spécifiés dans la littérature selon les normes NF X 60 300 et X 60 301:

1. La surveillance de la maintenance préventive. Il est important de connaître à ce niveau l'accessibilité de la composante, sa démontrabilité et son interchangeabilité.
2. La maintenance corrective, plus particulièrement, le temps de recherche de panne ou de défaillance et le temps de diagnostic.
3. L'organisation de la maintenance, pris en compte par la périodicité du préventif, le regroupement à des périodes identiques, l'homogénéité de la fiabilité des composants, la présence d'indicateurs et de compteurs et la complexité des interventions.
4. La qualité de la documentation technique. Celui-ci comporte la valeur du contenu, la disponibilité de la documentation, le mode de transmission et les principes généraux de rédaction et de présentation de la documentation technique.

5. Le suivi du bien par le fabricant. Il sera question de l'évolution du fabricant, de la qualité du service après-vente et de l'obtention des pièces de rechange[10].

IV.10.3. Types de maintenance

Il existe plusieurs types de maintenance parmi elles, on peut distinguer:

IV.10.3.1 La maintenance préventive

Effectuée périodiquement de façon préventive, elle permet d'améliorer la fiabilité des installations mais n'évite pas les pannes. Elle ne prend pas en compte l'état d'usure des pièces remplacées, entraîne des démontages et remontages préjudiciables à la durée de vie et à la fiabilité des équipements ainsi qu'une indisponibilité pour la réalisation des travaux.

Enfin, les stocks de pièces de rechange et la main d'œuvre nécessaires à la réalisation des travaux représentent un coût important. Il existe deux principaux types de maintenance préventive:

• Maintenance préventive systématique (l'entretien de l'équipement est effectué périodiquement).

• Maintenance préventive conditionnelle (l'intervention de techniciens dépend de l'état de l'équipement) [IVA et al, 03].

IV.10.3.2 La maintenance curative ou corrective:

Elle intervient après le constat d'une panne et consiste à en diagnostiquer les causes à réparer. Ce type de maintenance nécessite des équipes d'intervention surdimensionnées pour répondre dans les meilleurs délais, sans pour autant permettre de maîtriser la disponibilité des équipements. Les dégradations engendrent généralement des coûts de réparation et des pertes de production importantes [MER et al, 05].

10 www.these.ulaval.ca/2001/19524.html. 2001

IV.10.3.3 La maintenance conditionnelle ou prédictive:

Prévenir les pannes sans démontage ou arrêt de production, tel est l'objectif de la maintenance prédictive. Cette maintenance est basée sur l'analyse des conditions de fonctionnement des machines et elle est différente de la maintenance préventive traditionnelle qui se résume à des opérations de maintenance programmées à l'avance et à des intervalles réguliers. La maintenance prédictive comporte trois formes: ***traditionnelle*** (un expert ausculte de temps en temps la machine et analyse ses mesures périodiques pour essayer de prévoir un éventuel défaut) ; ***continue*** (l'utilisateur de la machine installe des capteurs sur une machine critique et un expert réalise l'analyse des informations sur un PC. La collecte est donc automatique, mais l'analyse ne l'est pas) ; ***continue intelligente*** (et la collecte et l'analyse sont automatiques).

Les avantages économiques de ce type de maintenance sont multiples, [LAA et al, 06]**:**

- Diminution des arrêts de production intempestifs permettant d'augmenter la disponibilité des équipements,
- Suppression des arrêts systématiques pour entretien,
- Limitation de la gravité des réparations entraînant une réduction des coûts d'intervention et une amélioration de la sécurité des interventions,
- Réduction des coûts de stockage des pièces de rechange approvisionnées en fonction des besoins réels,
- Réduction des coûts de stockage des pièces de rechange approvisionnées en fonction des besoins réels,
- Planification des interventions de maintenance permettant une amélioration de l'organisation des intervenants et une réduction des coûts,
- Amélioration de la qualité des interventions grâce à des interventions ciblées,
- Motivation des personnels par la valorisation des tâches de maintenance.

IV.10.4. Les systèmes de maintenance.

L'inspection et l'entretien du matériel industriel, ainsi que le maintien de son efficacité fonctionnelle est un défi majeur que doit élever chaque industrie dans le monde entier. Alors, un programme d'entretien des turbines à vapeur est essentiel pour les raffineries et les installations, si l'on veut maintenir la profitabilité et la sécurité tout en protégeant l'environnement. Une inspection continue et un entretien régulier sont, donc deux facteurs essentiels à l'obtention de procédés de production qui ne nuisent pas à l'environnement.

Les entreprises normalisent leurs services de maintenance pour qu'elles puissent se recentrer sur leur corps de métier. Elles font appel à des sociétés de maintenance, avec qui, elles signent des contrats garantissant un taux de disponibilité de machines dans un parc de machines à maintenir. La demande étant telle, que les prestataires de services de maintenance ont une diversité de plus en plus grande d'équipements à maintenir, ce qui nécessite un savoir faire non négligeable.

Afin d'assister aux mieux les différents acteurs de maintenance, une plateforme de e-maintenance est actuellement mise au point dans le cadre d'un projet Européen PROTEUS, par 17 partenaires européens, dont le maître d'oeuvre expert en maintenance est Cegelec France et Allemagne, cette plate forme est illustrée par la figure suivante:

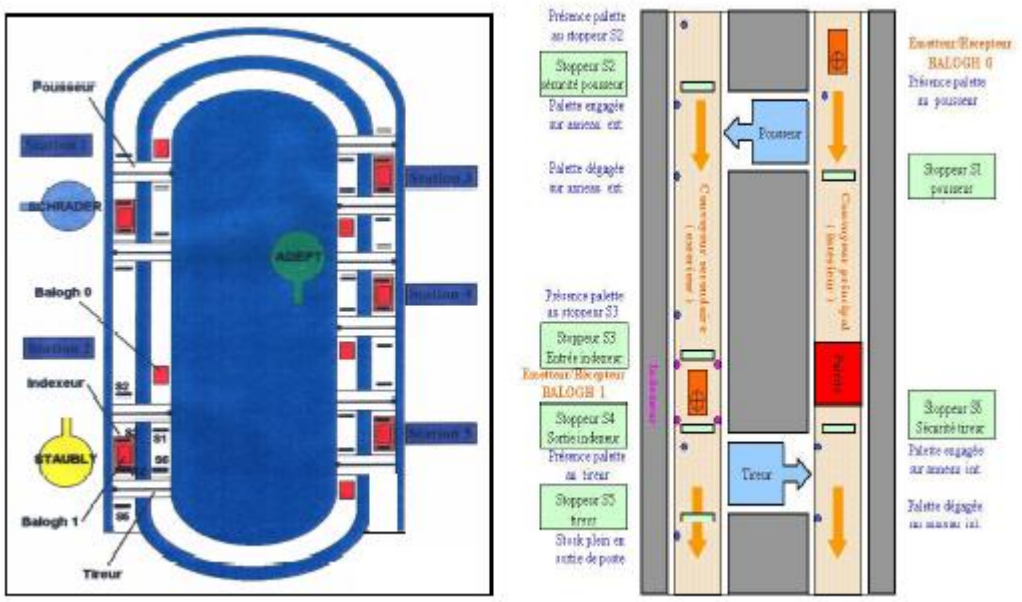


Figure IV.7: Plate forme du projet PROTEUS

IV.10.5 Les composants d'un système de maintenance

Plusieurs composants de système de maintenance peuvent être envisagés au sein d'une entreprise:

IV.10.5.1. SCADA

Un SCADA est un système d'acquisition de données ou de connaissances qui permet de suivre le dynamisme du système physique, tout en détectant les alarmes nécessitant une intervention immédiate ou sa programmation à une date ultérieure. Ce système peut être soit indépendant, soit intégré dans un système de contrôle et/ou de commande, dans des automates programmables. Un SCADA est rarement pris en compte dans un modèle de maintenance. Il modélise le processus physique à des fins de commande et/ou de supervision. Les données pertinentes de maintenance selon la stratégie envisagée ne sont pas forcément ni stockées, ni mesurées, ni calculées, et la satisfaction de besoins se fait jour de développement des adaptations en vue d'une intégration. Un cas particulier est celui des instruments intelligents (capteurs ou actionneurs) qui peuvent intégrer quelques fonctionnalités relevant de la problématique de la maintenance.

IV.10.5.2. Système de GMAO

Un système de Gestion de Maintenance Assistée par Ordinateur (GMAO) optimise les activités de maintenance et impacte ainsi la productivité du matériel maintenu. Un système de GMAO doit gérer les diverses stratégies de maintenance, que ce soit maintenance corrective, préventive ou prédictive.

Les objectifs d'une GMAO sont décomposés selon les trois axes suivants:

- La gestion proprement dite de l'activité de maintenance,
- La génération du retour d'expérience (REX),
- L'analyse des données tout au long des autres processus.

Un système de GMAO, permet alors:

- De mieux maîtriser les équipements à travers la diminution du temps d'arrêt et le nombre de défaillances, et d'augmenter la disponibilité des matériels ainsi que l'optimisation de l'efficacité du personnel.
- De mieux suivre le déroulement du processus de production.
- D'optimiser et de gérer les stocks.
- De réduire les coûts en optimisant les interventions et les stratégies de maintenance.

IV.10.5.3. Système d'aide au diagnostic

Un système d'aide au diagnostic est utilisé soit en cas d'arrêt et d'assistance à la recherche de la cause du dysfonctionnement, soit en permanence (actifs) pour prévenir les défaillances autant que possible. Ces systèmes sont construits sur des modèles très variés (modèle de Markov, modèle de Bayes, modèle neuro-mémitique, modèle de raisonnement à base de cas, etc.). Il doit, en général, s'appuyer sur des données issues d'un SCADA et sur l'expérience acquise pour assister la décision des responsables de la maintenance.

IV.10.5.4. Système de gestion de la documentation

Un système de gestion de la documentation des équipements constitue la base documentaire associée à l'installation à maintenir. Ce système doit délivrer la bonne - information, au bon moment, au bon endroit et à la bonne personne, et ce tout au long

de la procédure. Les informations ont longtemps été textuelles: procédures, relations d'anciennes expériences, notices d'utilisation ou de montage et/ou de démontage. Elles sont maintenant multimédia, incluant des films et des commentaires. Mais les systèmes de documentation contiennent maintenant plus que cela. Ils contiennent tout ce qui est relatif à la qualité, à la sécurité, aux normes, aux règlements intérieurs, et en général, à toutes les informations internes à l'entreprise.

IV.10.5.5. Système ERP

Un système ERP a pour rôle d'optimiser la productivité de l'entreprise. La méthode est basée sur une gestion, une optimisation et surtout une synchronisation des flux de matières, de produits, d'informations, de décisions et évidemment des flux financiers. De tels systèmes intègrent toutes les informations de l'entreprise, ressources humaines, gestion des achats et de la logistique, service commercial et gestion des ventes, production et gestion des matières, qualité des produits et des services, gestion comptable et financière, et évidemment la maintenance qui est en relation avec chacune des fonctions précédentes.

IV.10.6 Réparation et maintenance des turbines à vapeur

Dans un contexte économique concurrentiel, maintenir le déroulement efficace du fonctionnement des turbines à vapeur est un objectif primordial pour les industries mondiales. Comme toutes les machines, ces équipements doivent être contrôlés et renforcés de manière continue et périodique pour produire les meilleures performances. Donc, 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.

Les installations des turbines à vapeur produisent d'une part, la plus grande partie de l'électricité, et donc leurs pannes ne sont pas seulement coûteuses, mais pire encore peuvent engendrer d'énormes perturbations, et d'autres part, elles montrent leurs efficacités au sein des industries pétrolières, où un programme d'entretien est essentiel pour les raffineries et les installations pétrochimiques si l'on veut maintenir la profitabilité et la sécurité tout en protégeant l'environnement tel que l'usinage orbital des palliers de rotor de turbine, l'alésage et le fraisage de corps de turbines, de pompes

et de réducteurs, le perçage, le taraudage et le brochage des composants de turbines, etc.

De manière générale, les problèmes majeurs les plus rencontrés dans les turbines à vapeur se résument dans:

- Les aubes de turbine à vapeur basse pression ont besoin d'être réparées fréquemment en raison de dommages provoqués par les mauvaises conditions de vapeur, la perte de plaque anti_érosion brasées ou de dommages provoqués par des corps étranges. Ces aubes peuvent être en principe entièrement réparées en toute sécurité sur site ou en atelier, avec jusqu'à 50% de remplacement du profil externe, comprenant souvent:
  - Le remplacement des plaques de protection et du bord d'attaque (satellite)
  - La réparation par soudure de section endommagée du bord de fuite
  - Le remplacement complet des extrémités
  - Le remplacement/ la remise en état de tenons
  - Le repositionnement d'amortisseurs et réinstallation de fils de liaison

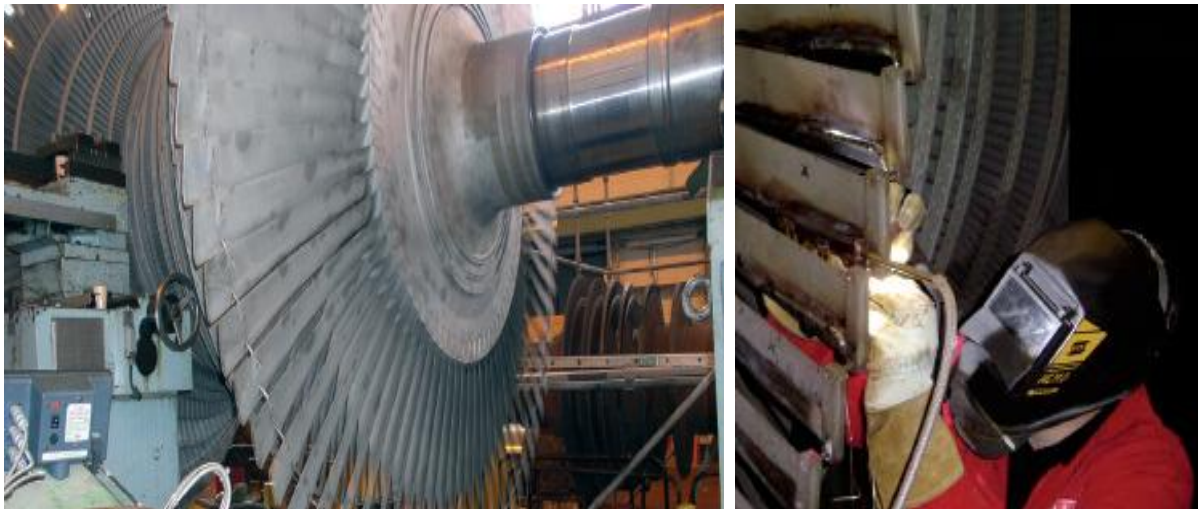


Figure IV.8: Maintenance de turbine à vapeur

- Réparation des diaphragmes et de tuyère, permet de réparer/ remettre en condition pour remplacer entièrement les ailettes ou les segments de profils externe, y compris:
  - La réparation par soudure d'ailettes/ de séparation
  - Le remplacement de segment d'ailette

- Le remplacement et la réparation de bagues internes, externes
- La réparation et la modification de goupille de positionnement
- Le remplacement de joint[8].

IV.11. Les applications des turbines à vapeur

IV.11.1. Les domaines d'application

Les domaines d'application privilégiés pour les turbines à vapeur sont principalement les réseaux de chaleurs, notamment lors de l'incinération de déchets en plus du secteur industriel:

- production d'électricité, cogénération.
- pétrole, raffinage.
- pétrochimie, chimie.
- incinération de déchets.
- sidérurgie, aluminium.
- papeteries, carton.
- agro-alimentaire.
- propulsion navale.
- …

IV.11.2. La production électrique par les Turbines à Vapeur

IV.11.2.1. Définition d'une centrale électrique

Une centrale de production d'énergie électrique est un site industriel destiné à la production d'électricité. Les centrales électriques transforment différentes sources d'énergie naturelle en énergie électrique afin d'alimenter en électricité les consommateurs, particuliers ou industriels relativement lointains. Le réseau électrique permet de transporter puis de distribuer l'électricité jusqu'aux consommateurs.

Les centrales les plus répandues sont constituées d'une chaudière et d'une turbine à vapeur. Leur carburant est le plus souvent du charbon mais on trouve aussi

8 http://www.power-technology.com/contractors/operations/wood_group/

des chaudières utilisant de la biomasse, du gaz naturel, du pétrole, ou des déchets municipaux.

IV.11.2.2. La production d'électricité

La production d'électricité est un secteur industriel vital, offrant à des clients, particuliers ou organisations, le service d'un approvisionnement régulier en énergie électrique. La production d'électricité se fait depuis la fin du XIXe siècle à partir de différentes sources d'énergie potentielles. Les premières centrales électriques fonctionnaient au bois. Aujourd'hui, la production se fait à partir du pétrole, du gaz naturel, du charbon, de l'énergie nucléaire, de l'énergie hydraulique, de l'énergie solaire et de l'énergie éolienne. Les moyens mis en œuvre sont diversifiés, et dépendent de plusieurs facteurs, tels que les technologies disponibles, la réactivité de mise en œuvre, la production nécessaire, le rendement possible, le coût des éventuelles matières premières, etc.

La plupart du temps l'électricité est produite à partir d'une source de chaleur, en utilisant la vapeur comme transporteur intermédiaire d'énergie. La vapeur fait tourner des turbines qui sont couplées à des générateurs électriques. La vapeur peut être produite en utilisant la plupart des sources d'énergie.

IV.11.2.3. Le cycle classique de production électrique

Le cycle de production d'électricité le plus répandu nécessite de disposer d'une source de chaleur permettant de chauffer de l'eau afin d'obtenir de la vapeur sous pression. Cette vapeur, en se détendant dans une turbine, entraîne un alternateur qui génère de l'électricité. Après turbinage, cette vapeur est condensée au moyen d'une source froide qui est généralement une source d'eau froide (cours d'eau, mer) ou est constituée de tours de refroidissement. La chaleur dégagée par la condensation de la vapeur peut être récupérée et exploitée à des fins de chauffage. La figure 9 représente le cycle de production classique de l'électricité.

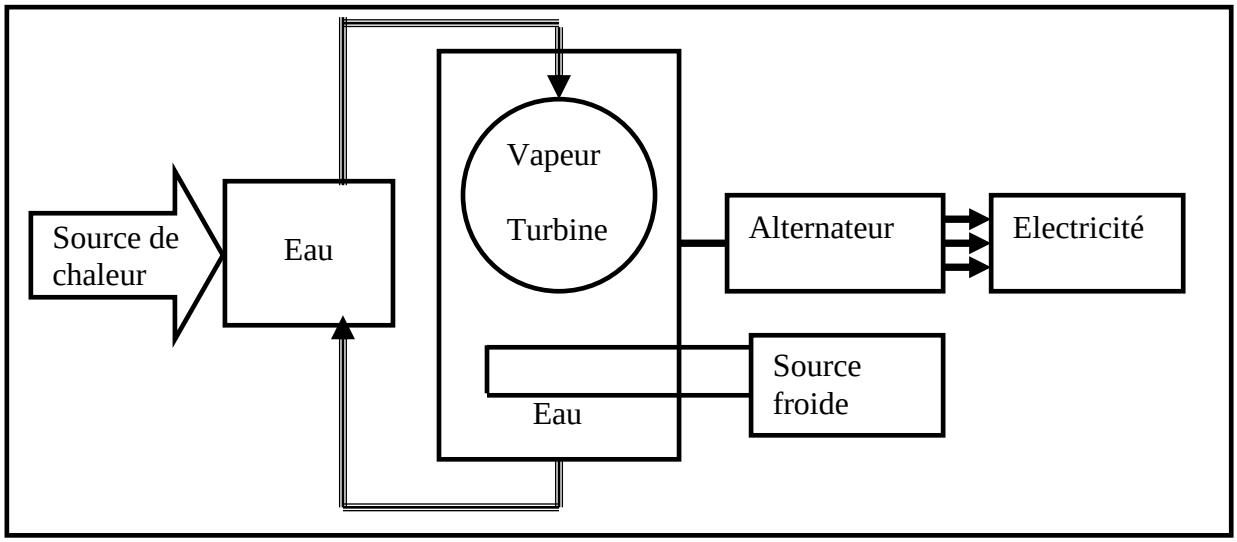


Figure IV.9: le cycle classique de la production électrique.

IV.11.2.4. Le processus de la production d'électricité[9]

Une centrale électrique produit de l'électricité à partir de la vapeur d'eau produite grâce à la chaleur dégagée par la combustion de gaz, de charbon ou de fioul, qui met en mouvement une turbine reliée à un alternateur. Un schéma analogique du processus de production d'électricité est présenté par la figure suivante:

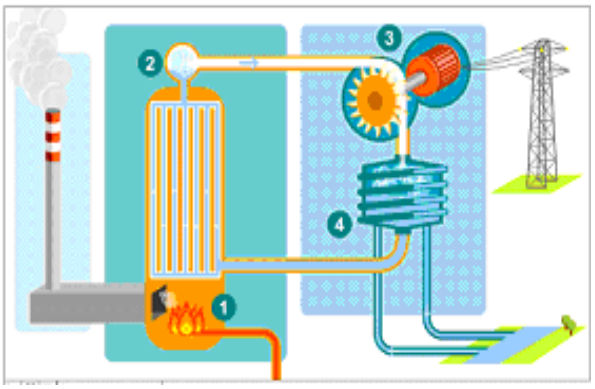


Figure IV.10: Le processus de production d'électricité[9].

Dans ce qui suit, nous décrivons séparément chacune des étapes de ce processus:

9 http://www.edf.com/html/production/thermique/fonctionnement.html

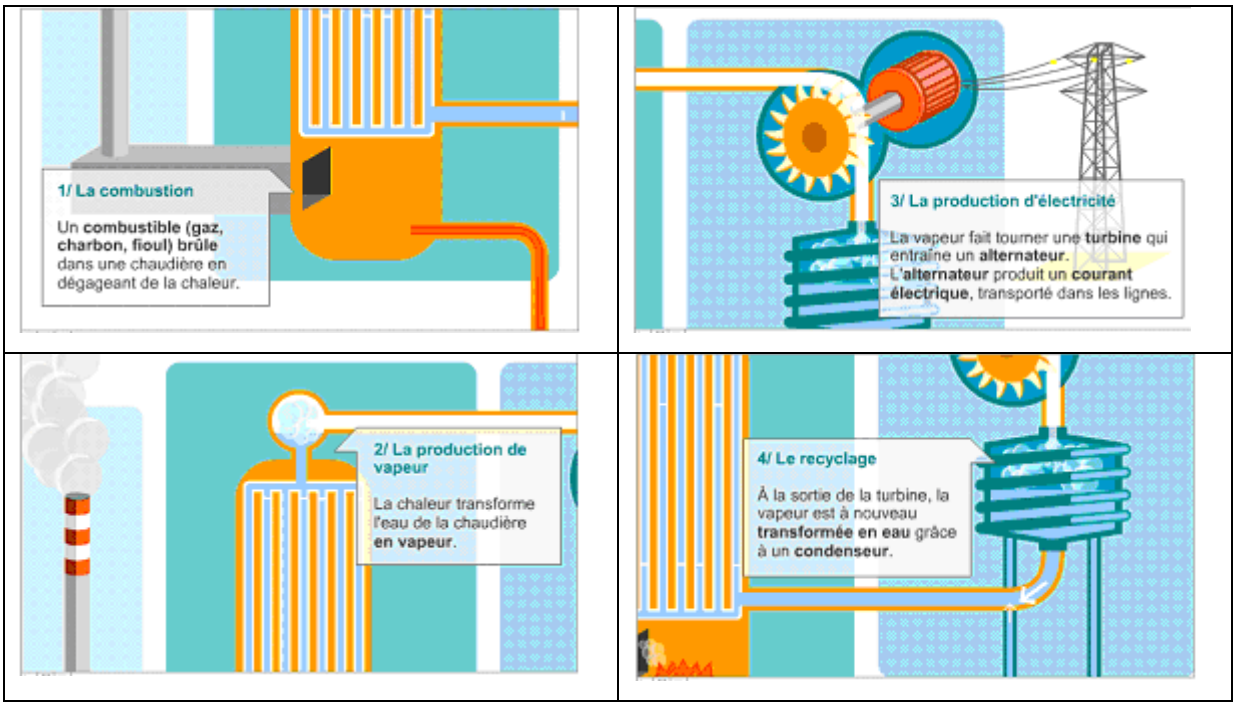


Figure IV.11: Les étapes du processus de production d'électricité[9].

IV.11.2.4.1. La combustion

Un combustible (gaz, charbon, fioul) est brûlé dans les brûleurs d'une chaudière pouvant mesurer jusqu'à 90 m de hauteur. Le charbon est d'abord réduit en poudre, le fioul est chauffé pour le rendre liquide puis vaporisé en fines gouttelettes et le gaz est injecté directement sans traitement préparatoire.

IV.11.2.4.2. La production de vapeur

La chaudière est tapissée de tubes dans lesquels circule de l'eau froide. En brûlant, le combustible dégage de la chaleur qui va chauffer cette eau. L'eau se transforme en vapeur, envoyée sous pression vers les turbines.

IV.11.2.4.3. La production d'électricité

La vapeur fait tourner une turbine qui entraîne à son tour un alternateur, tel que la turbine fait tourner l'axe sur lequel est fixé le rotor de l'alternateur, qui est composé d'une série d'électroaimants, puis, l'interaction entre ces électroaimants mobiles du

9 http://www.edf.com/html/production/thermique/fonctionnement.html

rotor et les bobines de fils de cuivre fixes qui composent le stator de l'alternateur produit un courant électrique.

Un transformateur élève la tension du courant électrique produit par l'alternateur pour qu'il puisse être plus facilement transporté dans les lignes à très haute et haute tension. L'électricité est consommée au même moment où elle est produite car elle ne se stocke pas.

IV.11.2.4.4. Le recyclage

À la sortie de la turbine, la vapeur est à nouveau transformée en eau grâce à un condenseur dans lequel circule de l'eau froide en provenance de la mer ou d'un fleuve. L'eau ainsi obtenue est récupérée et re-circule dans la chaudière pour recommencer un autre cycle. L'eau utilisée pour le refroidissement est restituée à son milieu naturel ou renvoyée dans le condenseur. Les fumées de combustion sont dépoussiérées grâce à des filtres et sont évacuées par des cheminées.

Donc, après s'être produit, l'énergie électrique est transportée puis distribuée, pour être exploitée, la figure suivante illustre l'acheminement de l'électricité du producteur au consommateur.

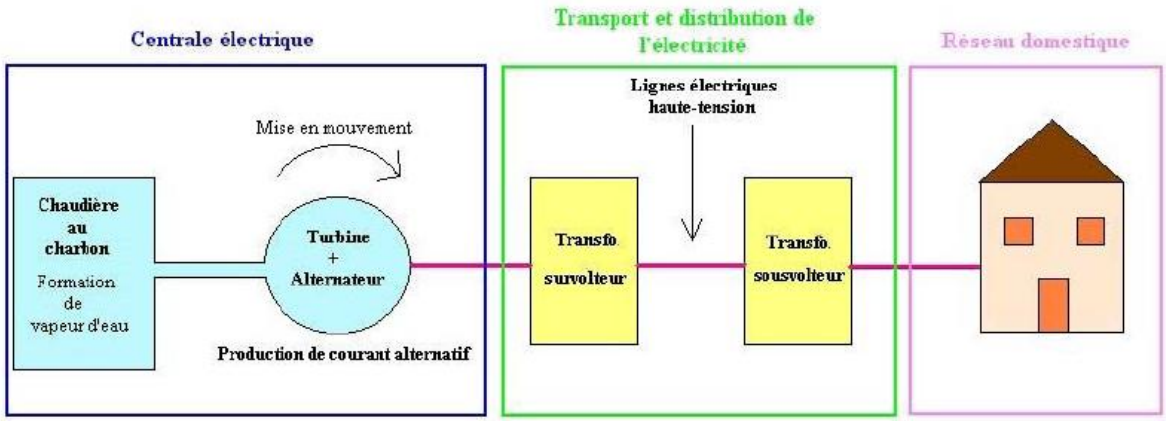


Figure IV.12: L'électricité, du producteur au consommateur…

A la fin, nous mentionnons que les installations de turbine à vapeur produisent la plus grande partie de l'électricité requise par les industries mondiales. Par exemple, elles produisent près de 70 pour cent de l'électricité consommée aux Etats-Unis. Par conséquent, les pannes ne sont pas simplement coûteuses, et elles peuvent causer d'énormes perturbations. En d'autres termes, le fonctionnement efficace des turbines à

vapeur est important pour ces industries, mais comme toutes les machines, ces équipements doivent être continuellement inspectés et entretenus pour produire les meilleures performances. Ce qui reflète alors l'importance de la maintenance industrielle pour les garder en bon état de marche et réduire au minimum les risques de panne.

Les systèmes de la Gestion de la Maintenance Assistée par Ordinateur, GMAO, peuvent donc aider les ingénieurs de centrale et les constructeurs à réparer et modifier leurs turbines à vapeur pendant les arrêts planifiés ou en cas d'urgence. Ils peuvent alors, en guider à effectuer la réparation des turbines à vapeur sur place, tels que:

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

En résumé, nous pouvons dire qu'une bonne Gestion de la Maintenance Assistée par Ordinateur, peut assurer une augmentation de la productivité de l'entreprise par la mise en place d'un système permettant de fournir, à un individu, l'information utile au moment où il en a besoin, dans les meilleurs délais, et de façon exploitable, ainsi qu'une amélioration de l'organisation en diminuant le gaspillage par une réutilisation des savoirs et des savoirs faire et des compétences développées au cours du temps.

Dans le chapitre suivant, nous allons construit deux ontologies à partir de deux différentes bases de données représentant une turbine à vapeur, puis nous utiliserons la technique de la fusion d'ontologies (vue au chapitre précédent), pour obtenir une seule ontologie plus large et plus complète permettant de répondre plus efficacement à des requêtes autour d'une turbine à vapeur.

Nous avons présenté dans le deuxième chapitre les composants de base constituant une ontologie, tout en décrivant ses différentes méthodologies de constructions, ses langages de représentation ainsi que ses éditeurs de développement. Nous avons détaillé ses étapes de construction et son processus général de développement. Dans le troisième chapitre, nous avons détaillé le processus général de fusion d'ontologies ainsi que les différents outils de fusion existant dans la littérature.

Dans ce qui suit, nous allons projeter ces notions sur nos deux BDDs représentant une Turbine à Vapeur (voir chapitre 4) selon deux différents contextes pour concevoir et développer deux différentes ontologies, puis les fusionner pour obtenir une seule ontologie plus grande et plus complète qui couvre un domaine d'application plus large, tout en spécifiant et justifiant nos choix de méthodologies de construction, langages de représentation et outils ou éditeurs de développement ou de fusion.

V.1. Description des sources de connaissances exploitées

Avant d'entamer les processus généraux de développement et de fusion proprement dits, nous concevons tout d'abord les sources de connaissances exploitées pour permettre une meilleure compréhension des processus proposés.

Donc, après l'extraction des connaissances décrivant une Turbine à Vapeur, à partir des interviews entre les ingénieurs de connaissances et les experts de domaine, en plus d'autres documents techniques, ces connaissances sont sauvegardées et structurées dans deux BDDs sous MicroSoft Access. Elles se diffèrent l'un de l'autre dans le contexte d'utilisation, tel que la première BDD, CentraleTopo.mdb décrit la Turbine à Vapeur d'un point de vue Topologique en la caractérisant par le code de chaque composant Topologique, sa description, sa zone d'emplacement, sa fonction, sa famille ainsi que le code de son parent géographique. Un extrait de cette BDD est présenté dans la figure 1:

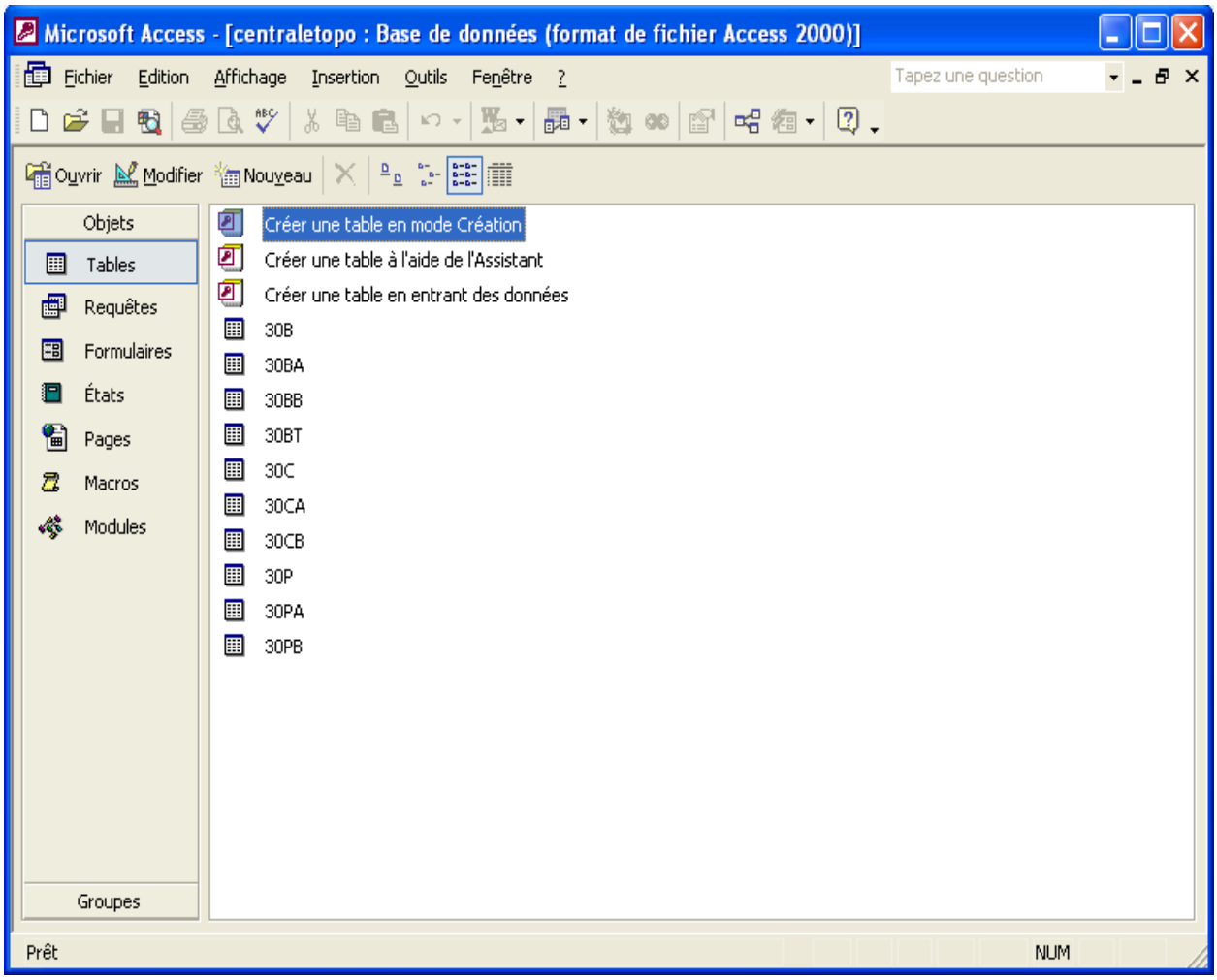


Figure V.1: Un extrait de la BDD CentraleTopo.

Alors que la deuxième BDD, Diagnostic.mdb, contient des cas de maintenance de la même Turbine à Vapeur, où seulement les composants qui sont tombés en panne, sont représentés. Chaque composant est décrit par ses cas d'intervention pour le relevé de dérangement, tout en mentionnant son code, sa description, le code de la panne qu'il présente, sa description, le code de défaut, sa description, le code du remède, sa description ainsi que le code de son parent, un extrait de cette BDD est donné par la figure 2:

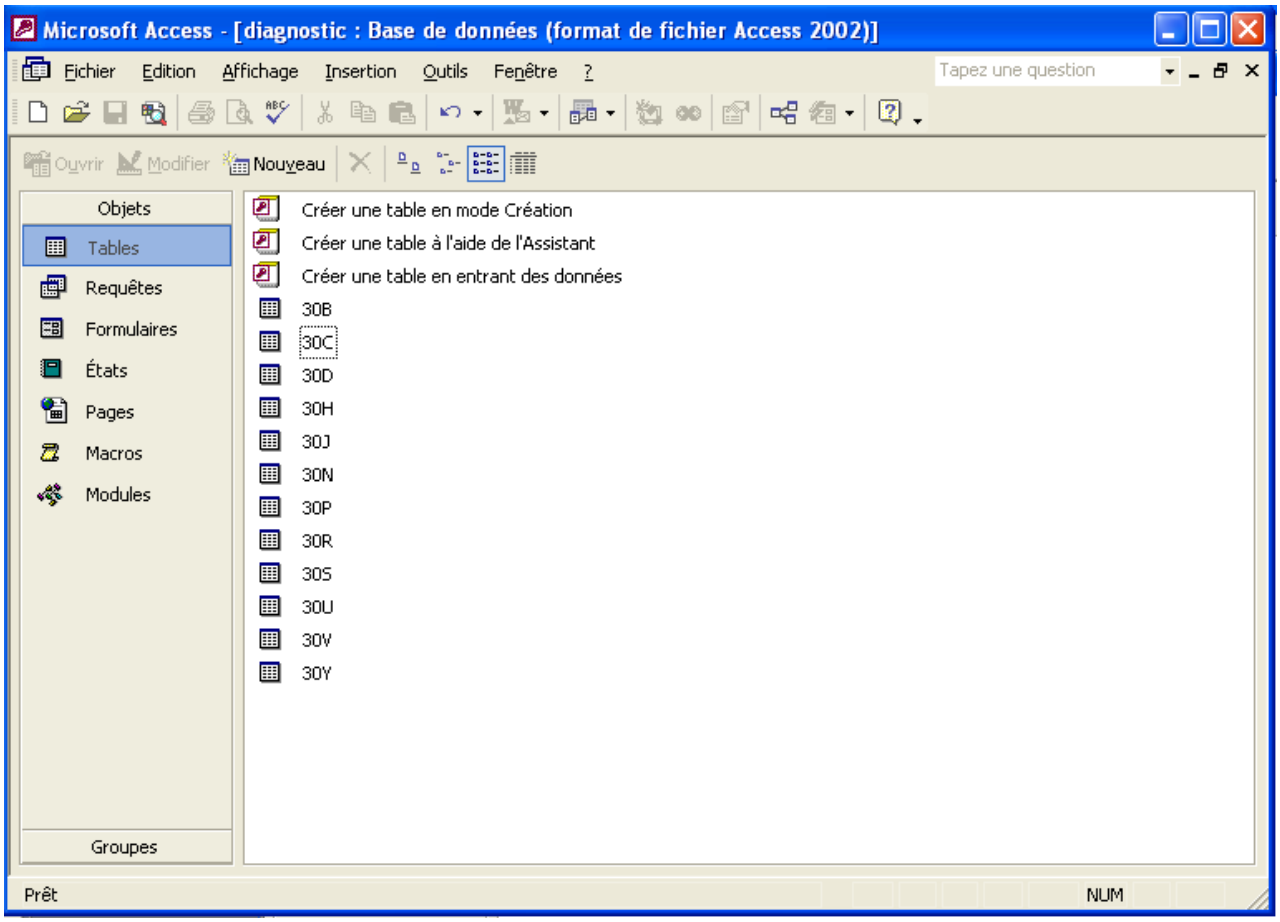


Figure V.2: Un extrait de la BDD Diagnostic

Pour une meilleure compréhension des deux BDDs, nous proposons de les décrire à travers un modèle E-A (Entité-Association) comme illustrés dans les figures 3,4 et 5:

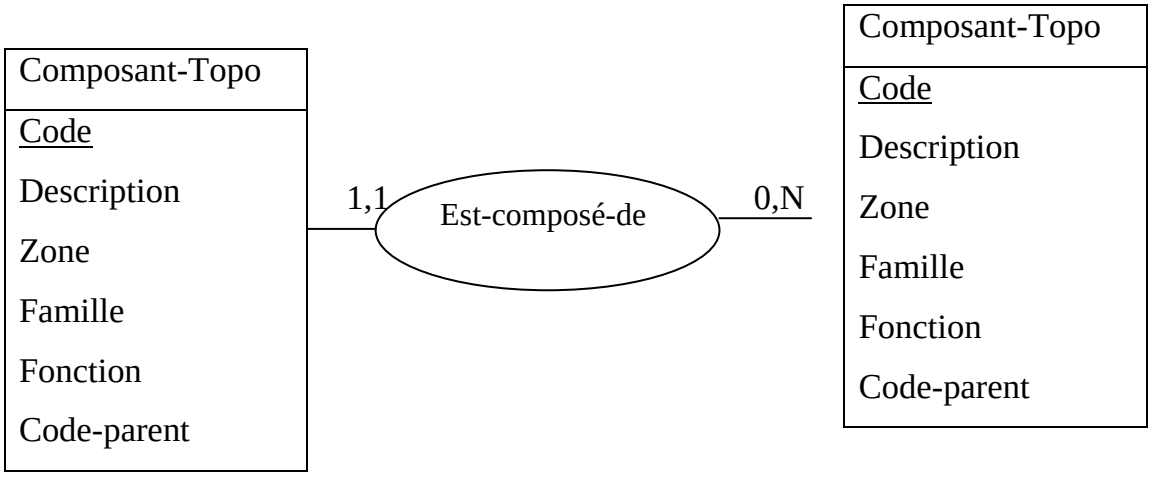


Figure V.3: Le modèle E-A représentant la BDD CentraleTopo

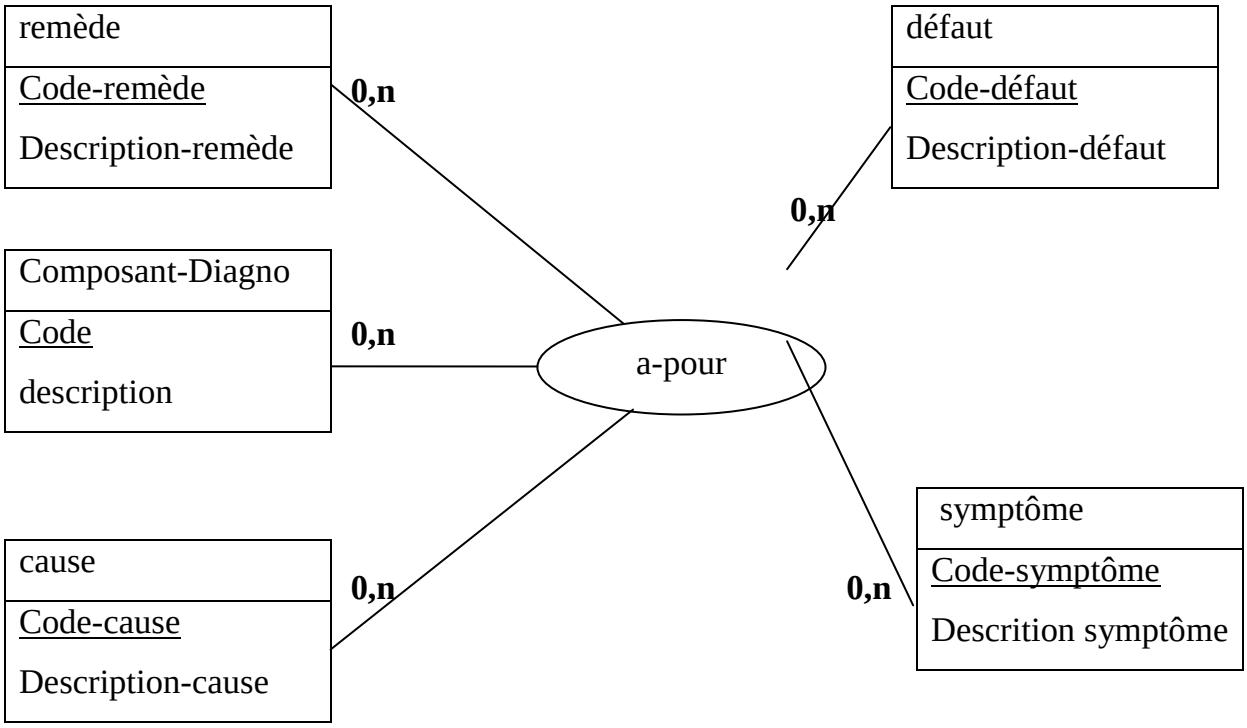


Figure V.4: Le modèle E-A représentant la BDD Diagnostic

Et si on fait la modélisation des deux BDDs concernées nous pouvons trouvé le modèle E-A de la figure 5

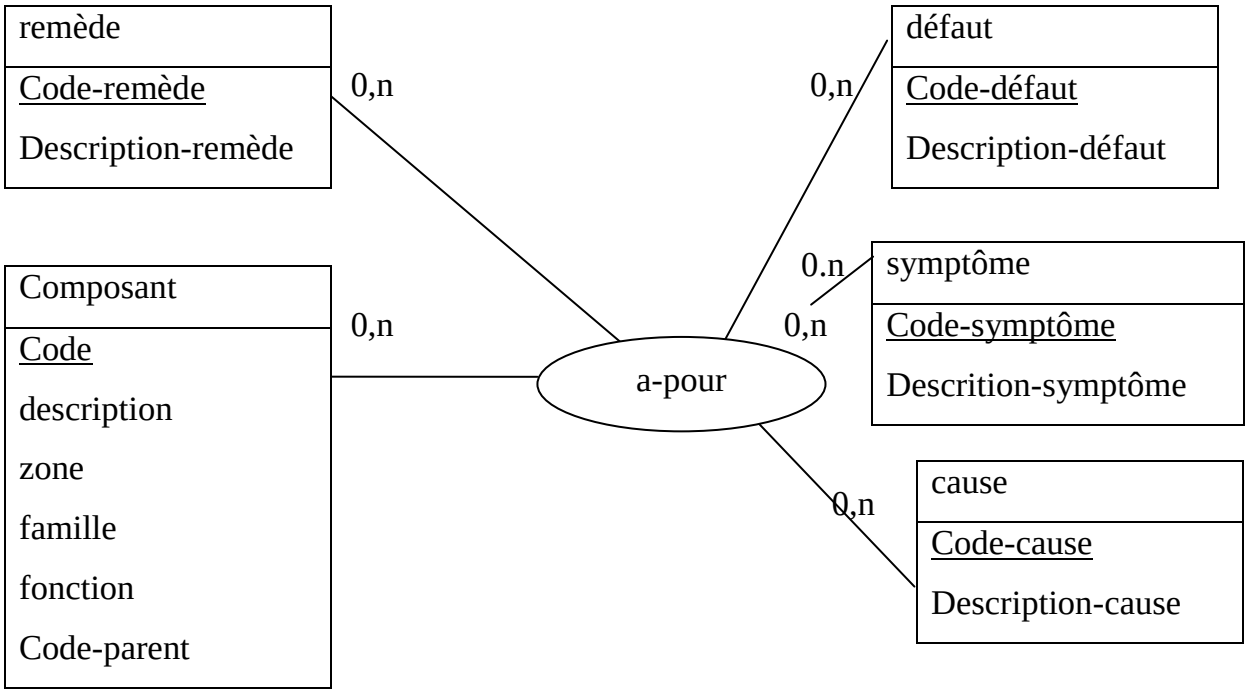


Figure V.5: Le modèle E-A représentant la fusion des deux BDD.

V.2. Méthodologies de construction d'ontologies

Comme nous l'avons déjà vu dans les chapitres précédents, pour construire ou développer une ontologie, plusieurs méthodologies de construction sont imposées dans la littérature, en l'occurrence, la méthodologie Entreprise, Tove, Methontology, Amaya Bernaras et al, Sensus et On-To-Knowledge. Le concepteur d'ontologie choisit la méthodologie qui lui correspond le plus selon le type de construction qu'il va adopter, comme construire une ontologie à partir de zéro ou construire une ontologie par intégration ou utilisation des ontologies préalablement existantes (fusion, alignement, mapping, etc). Le concepteur peut aussi construire une ontologie qui doit être dépendante, indépendante ou semi-dépendante de l'application qui va utiliser cette ontologie. Etant donné que nous voulons construire tout d'abord deux ontologies, Topo et Diagno, à partir de BDDs, puis construire une troisième ontologie en réutilisant ces deux ontologies à travers un processus de fusion, et vu qu'on souhaite construire une ontologie finale qui sera totalement indépendante de l'application qui l'exploite, nous choisissons donc de suivre la méthodologie **Methontology** qui consiste à identifier tout le processus de développement de l'ontologie tout en nous basant sur un prototype évolutif complété par des techniques particulières de contrôle et d'assurance de qualité, et de spécification, acquisition et conceptualisation de connaissances. Ceci, tout en utilisant une stratégie mixte pour l'identification de concepts de l'ontologie, où les concepts les plus importants sont identifiés en premier, puis sont spécialisés et/ou généralisés.

V.3. Processus général de développement d'ontologies

Le processus général de développement d'ontologies qu'on a adopté est composé de trois phases principales, où certains outils de construction d'ontologies utilisant des formalismes variés et offrant différentes fonctionnalités sont exploités lors de chacune de ces phases:

V.3.1 La conceptualisation

Lors de cette phase primordiale, nous avons procédé par l'identification des connaissances contenues dans un corpus représentant la Turbine à Vapeur. Nous

dégageons d'abord les concepts décrivant les entités cognitives (tels que: 30B, 30C, 30P, etc), ainsi que les relations entre eux (tel que SubClassOf). Nous précisons ensuite la nature conceptuelle désignant ces entités cognitives: concepts, relations, axiomes, propriétés, etc. Cela guide l'ingénieur de connaissances qui applique son expertise de paradigmes de représentation de connaissances en machine pour aboutir à un modèle conceptuel. Ce dernier structure les concepts et les relations entre eux tout en les enrichissant par les axiomes, les propriétés, les slots, les facets, etc.

V.3.1.1 Les outils de développement orientés conceptualisation

Plusieurs outils de construction d'ontologies orientés conceptualisation sont définies dans la littérature, nous citons à titre d'exemple:

V.3.1.1.1 TERMINAE[1]: A été conçu et développé au LIPN, de l'université Paris de Nord, il consiste à extraire les concepts clés d'un domaine donné, à partir d'un corpus textuel et à travers l'outil d'ingénierie linguistique LEXTER.

V.3.1.1.2 Text-To-Onto[2]: A été conçu et développé à l'institut AIFB, de l'université Karlsruhe. Il utilise des ontologies existantes pour l'extraction de concepts de l'ontologie à partir d'un corpus ou de documents web.

V.3.1.1.3 OntoBuilder[3]: A été conçu et développé au Technion de Haifa. Comme son nom l'indique, il permet de construire des ontologies tout en utilisant des ressources web et permet de fusionner des ontologies extraites de différents sites web.

V.3.1.1.4 DataGenie[4]: Il s'agit d'un plugin de Protege qui permet d'importer des BDDs relationnelles (sous access ou excel) dans Protege Frame, mais il présente l'inconvénient de l'absence de constructeurs OWL qui permettent de résoudre le problème d'hétérogénéité sémantique entre les connaissances importées des BDDs.

[1] http://www.lipn.univ-paris13.fr.szulman.Terminae.html
[2] http://www.kmaifb.uni-karlsruhe.de.kaon2.frontpage
[3] http://www.iew3.technion.ac.il.Ontobuilder
[4] http://www.protege.stanford.edu/plugins/datagenie

V.3.1.1.5 DataMaster [NYU et al, 07]

Il s'agit aussi d'un plugin de Protege qui permet d'importer des BDDs relationnelles (sous access ou excel) dans protégé Frame, mais aussi dans protégé OWL, et contrairement à l'outil DataGénie, DataMaster ne permet pas seulement l'extraction du schéma ou du contenu d'une seule BDD relationnelle dans une seule ontologie, mais il permet aussi selon le choix de l'utilisateur de manipuler plusieurs BDDs relationnelles d'un seul coup, en les important dans plusieurs ontologies séparées, mais aussi dans une seule ontologie avec des espaces de noms différents, où les constructeurs OWL interviennent pour résoudre le problème d'hétérogénéité sémantique entre les données appartenant à différents BDDs sources. Ces caractéristiques font de DataMaster un outil très important d'importation des contenus des BDDs relationnelles vers l'environnement protégé justifiant ainsi notre choix de l'outil pour la construction automatique des ontologies Turbine à Vapeur.

V.3.1.2 Les étapes conceptuelles de construction d'ontologies

Une approche conceptuelle de construction d'ontologies en sept étapes proposée par [NOY et al, 01] est présentée par la figure 6:

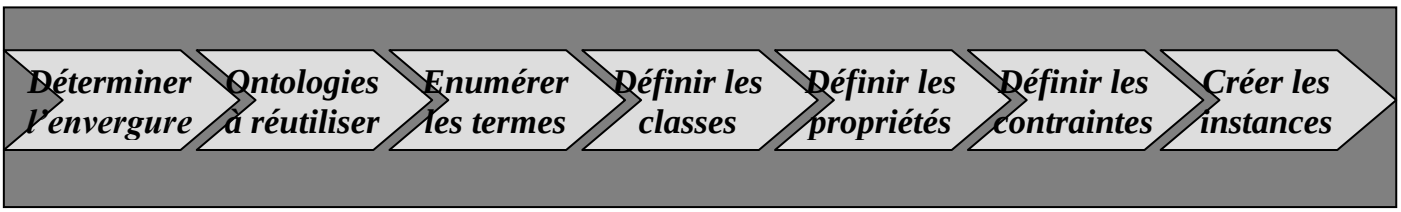


Figure V.6: Les étapes de construction d'une ontologie

Dans ce qui suit, nous détaillons chacune de ces étapes:

- Etape 1: Déterminer l'envergure: Il s'agit là de définir le domaine de l'ontologie ainsi que sa portée ou son étendue, en cherchant des réponses à quelques questions de base: Quel est le domaine à couvrir par l'ontologie ? Dans quel but cette ontologie doit être utilisée ? A quel type de questions l'ontologie devra t elle fournir des réponses ? Et qui va utiliser et maintenir l'ontologie ?

  Les réponses à ces questions peuvent varier au cours du processus de la conception de l'ontologie, consistant, pas à pas, à limiter la porté du modèle. Une des méthodes visant à déterminer la porté d'une ontologie est de rédiger

une liste de questions, dites questions de compétences, auxquelles une base de connaissances fondée sur une ontologie devrait pouvoir répondre. A partir de ces questions, l'ontologie comprendra l'information sur les différentes caractéristiques des composants, et elles serviront plus tard de test décisif.

- Etape 2: Considérer la réutilisation des ontologies existantes sur le même domaine: Il s'agit là d'identifier s'il y a des ontologies à intégrer dans l'ontologie en question, tout en étudiant leurs structures ou spécificités pour qu'on puisse les prendre en considération dans la conception de l'ontologie courante pour pouvoir les intégrer dedans, tel que par exemple si l'intégration d'une certaine ontologie dans l'ontologie courante est de type fusion d'ontologies, il faut prendre en compte le langage de représentation de l'ontologie à intégrer dans la conception de la nouvelle ontologie, car s'ils ne sont pas représentées dans le même formalisme de représentation on ne peut pas les fusionner.
- Etape 3: Enumérer les termes importants dans l'ontologie: Il s'agit de lister les termes clés dans cette ontologie sans se préoccuper du possible chevauchement des concepts qu'ils peuvent induire. L'élaboration de la liste regroupant les termes à traiter ou à expliquer à l'utilisateur de l'ontologie peut être fait tout en répondant à ces questions: Sur quels termes souhaiterons-nous discuter ? Que veut-on dire sur ces termes ? Quels sont les termes devant apparaître dans l'ontologie comme des concepts ? Et Quelles sont les propriétés de ces termes ?
- Etape 4: Définir les classes et la hiérarchie de classes: Il existe plusieurs approches possible pour développer une hiérarchie de classes, parmi eux, un procédé de développement de haut en bas commence par l'identification de concepts les plus généraux du domaine et se poursuit par la spécialisation des concepts. Pour ce faire, nous commençons par définir les classes à partir de la liste créée dans l'étape précédente, en sélectionnant les termes qui décrivent des objets ayant une existence indépendante plutôt que ceux qui décrivent ces objets. Ces termes constitueront les classes dans l'ontologie et deviendront des points d'ancrage dans la hiérarchie des classes. Ensuite, nous organisons les classes dans une taxonomie hiérarchique.

- Etape 5: Définir les propriétés associées aux classes avec les termes restants: Après avoir défini les classes de l'hiérarchie, nous procédons à la description de la structure interne des concepts à travers la définition des attributs de chaque concept aidant ainsi à fournir assez d'informations pour répondre aux questions de compétence de la première étape.
- Etape 6: Définir les facets des attributs (les contraintes qui s'appliquent sur les propriétés): Il s'agit de faire attribuer aux propriétés (attributs) leurs valeurs décrivant la valeur du type, les valeurs autorisées, en plus d'autres caractéristiques de valeurs que les attributs peuvent avoir tels que la cardinalité, la restriction des valeurs, le domaine et le co-domaine.
- Etape 7: Créer les instances de classes de la hiérarchie: La dernière étape de la construction d'une ontologie consiste à créer les instances de classes, tel que pour chaque classe, on crée une instance individuelle, et on la renseigne par ses propres valeurs d'attributs.

V.3.1.3 Le modèle en Oignon pour la conceptualisation des ontologies:

Pour la mise en œuvre de la conception proprement dite d'une ontologie, plusieurs modèles de conception d'ontologies ont été présentés dans la littérature. Vu que notre objectif est d'obtenir à la fin une ontologie globale couvrant toutes les connaissances contenant dans les deux BDDs, CentraleTopo et Diagnostic après la conception et la construction des deux ontologies qui vont être par la suite fusionnées, nous avons donc choisi de faire la conception de l'ontologie Turbine à Vapeur selon le modèle en **Oignon**, qui est un modèle en couches pour la conception d'ontologies, proposé par [JEA 07], et qui consiste à intégrer les différentes catégories d'ontologies, à savoir les Ontologies Conceptuelles Canoniques (OCC), les Ontologies Conceptuelles Non Canoniques (OCNC) et les Ontologies Linguistiques (OL) présentées dans le deuxième chapitre. Avant d'entamer la conception proprement dite de l'ontologie Turbine à Vapeur en se basant sur le modèle en Oignon, nous présentons tout d'abord les notions de base de ce modèle ainsi détaillé par [JEA 07]:

V.3.1.3.1 Relations entre les différentes catégories d'ontologies

A partir de la description de différentes catégories d'ontologies vue précédemment, l'auteur a identifié les relations suivantes entre OCC, OCNC et OL:

- Les mappings entre OCC peuvent être également définis avec des opérateurs d'équivalence conceptuels des OCNC.
- Les OCNC peuvent utiliser les constructeurs orientés OCC pour définir leurs concepts primitifs.
- Les OL peuvent définir les différentes significations de chaque mot d'un langage naturel particulier en référençant une OCNC. Cette référence fournirait une base formelle pour effectuer des inférences et des traductions automatiques de termes spécifiques à un contexte donné.

Pour aller plus loin dans ce constat à travers la mise en exploitation de ces relations entre les différentes catégories d'ontologies, l'auteur a proposé donc le modèle en Oignon qui permet d'intégrer ces différentes catégories d'ontologies.

V.3.1.3.2 Un modèle en couche pour la conception d'ontologies

L'approche présentée dans la section précédente exploite la capacité des OCNC de définir des équivalences conceptuelles et ainsi d'intégrer plusieurs ontologies traitant le même domaine. Et vu que l'auteur de cette approche, pense que les OCNC peuvent bénéficier du fait d'être articulées avec une OCC, il propose une approche alternative pour le développement d'une OCNC en commençant par la conception d'une OCC:

1. La première étape dans la conception d'une ontologie peut être de se mettre d'accord au sein d'une communauté sur une OCC. Pour atteindre cet accord, les étapes suivantes sont requises.

- Identifier clairement quel est le domaine couvert par cette ontologie en s'appuyant sur le savoir-faire des experts du domaine.
- Choisir un modèle permettant de définir précisément les concepts primitifs (classes et propriétés) existant dans le domaine et permettant d'indiquer le contexte d'évaluation des valeurs des propriétés.

- Fournir des descriptions partagées de l'ensemble de ces concepts primitifs couvrant le domaine considéré. Cette conceptualisation doit permettre de satisfaire aux besoins techniques et métiers larges et diversifiés partagés par les membres de cette communauté. Elle doit atteindre une large acceptation et reconnaissance.

2. Au sein de la communauté d'utilisateurs et/ou de développeurs, sur la base de l'OCC définie, une OCNC peut être construite afin d'être utilisée par les membres de cette communauté, soit pour construire leur propre point de vue du domaine, soit pour modéliser formellement sous forme de classes certains concepts existants dans le domaine cible et qui s'expriment sous forme de combinaisons de concepts canoniques. En procédant de cette façon, on assure de conserver la capacité d'échanger et de partager les informations exprimées en termes de l'OCC.

3. Afin de permettre l'utilisation de l'OCNC définie pour les inférences linguistiques et/ou pour fournir une interface utilisateur conviviale dans différents langages naturels, associer à chaque concept de l'OCNC une liste de termes spécifiques pour une ou plusieurs langues naturelles données. Ces relations entre termes et concepts peuvent éventuellement être qualifiées (facteur de vraisemblance, etc.). La figure 7 illustre un modèle en couches nommé le modèle en oignon d'une ontologie de domaine résultant de cette approche alternative. Une OCC fournit une base formelle pour modéliser et échanger efficacement la connaissance d'un domaine. Une OCNC fournit les mécanismes pour lier différentes conceptualisations faites sur ce domaine. Finalement, les OL fournissent une représentation en langage naturel des concepts de ce domaine, éventuellement dans les différents langages où ces concepts sont significatifs.

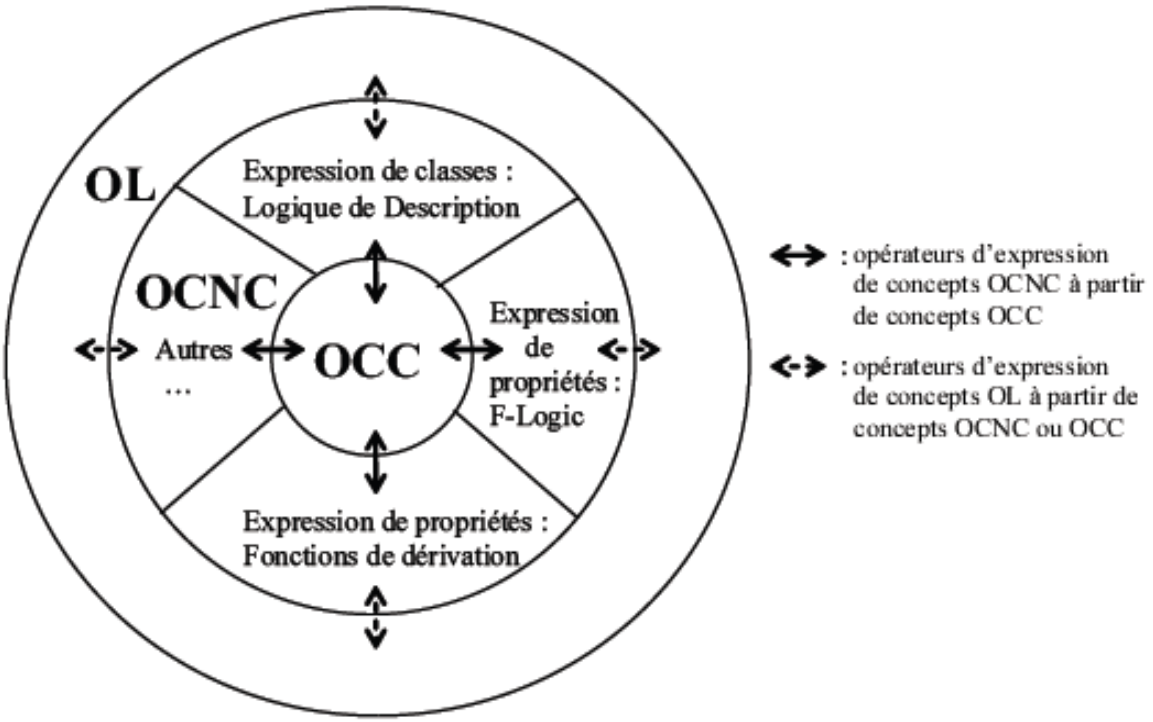


Figure V.7: Le modèle en Oignon pour les ontologies de domaine, [JEA 07].

Pour montrer l'intérêt du modèle en Oignon, l'auteur suggère un exemple d'un scénario d'échange basé sur une ontologie conçue selon ce modèle que nous présentons dans la section suivante:

V.3.1.3.3 Un scénario d'échange basé sur une ontologie en couches

Dans l'univers des bases de données, chaque base de données utilise un vocabulaire canonique. Généralement, chacune d'elles utilise un vocabulaire canonique différent. Ce scénario consiste à échanger des données entre différentes sources de données utilisant un vocabulaire canonique différent.

Plutôt que de définir une OCNC couvrant tous les termes de toutes les sources (approche illustrée par la figure 8 (A)), le modèle en oignon suggère que tous les échanges soient effectués en utilisant une OCC *consensuelle.* Chaque source contient simplement les descriptions de ses propres concepts en termes des concepts primitifs de la OCC (approche illustrée sur la figure 8 (B)). Cette approche a été mise en application pour l'intégration de bases de données dans [BEL et al, 2004]. Il est à noter que si chaque concept est représenté différemment dans les *n* sources participant à l'échange et s'il existe dans chaque source une moyenne de *p* concepts, la solution (A)

requiert d'implanter *n*p* mappings dans chaque source, alors que la solution (B) requiert seulement *p* mappings dans chaque source.

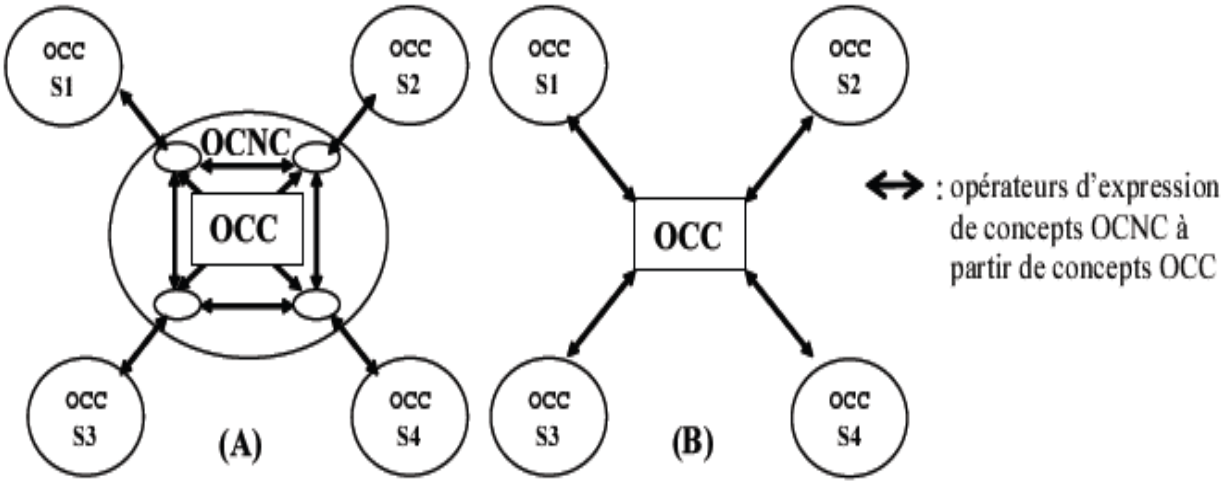


Figure V.8: Utilisation d'ontologies pour l'échange canonique de données.

Ce scénario et la démarche de conception proposée dans la section précédente montrent l'intérêt d'articuler les trois catégories d'ontologies selon le modèle en oignon tout au long du cycle de vie des ontologies de domaine. En effet, chaque catégorie d'ontologies offre des capacités particulières:

- les OCC fournissent une description canonique et précise de chaque concept d'un domaine donné. Elles fournissent une base solide pour l'échange entre différentes sources d'informations.
- les opérateurs des OCNC sont utilisés pour interagir avec ou intégrer d'autres applications ou sources ayant déjà leur propre ontologie.
- les OL offrent des capacités linguistiques sur l'ensemble des concepts (primitifs et définis) du domaine.

Dans la section suivante, l'auteur montre en quoi ces caractéristiques sont intéressantes pour les bases de données.

V.3.1.3.4 Liens entre le modèle en oignon et les bases de données

Chaque couche du modèle en oignon peut être utilisé pour résoudre différents problèmes des bases de données.

- Les OCC sont des modèles conceptuels formels et partageables. Ils peuvent être utilisés comme base pour la conception d'un modèle logique de base de

données ou comme schéma global dans un scénario d'intégration de bases de données.

- Les OCNC proposent des mécanismes similaires aux vues des bases de données, avec une théorie formelle offrant des capacités d'inférence. Ces mécanismes peuvent être utilisés pour réaliser le mapping entre différents schémas de bases de données.
- Les OL peuvent être utilisées pour localiser les similitudes existantes entre plusieurs schémas de bases de données [BEN et al, 2000], pour documenter les bases de données gérées dans les Systèmes de Gestion de Bases de Données (SGBD) ou pour enrichir le langage de dialogue personne/SGBD.

V.3.2 L'ontologisation

A travers cette phase, nous allons procéder à la structuration et à la formalisation du modèle conceptuel obtenu de la phase précédente dans un des langages formels qu'on a vu dans les chapitres précédents, tels que le langage KIF, ONTOLINGUA, LOOM, OCML, FLOGIC, OKBC, Ocycl, XML, XOL, SHOE, RDF(S), DAML+OIL, OWL (voir le chapitre 2) construisant ainsi l'ontologie tout en spécifiant la terminologie et la sémantique du domaine de l'application. Parmi ces langages de représentation formels, nous avons choisi d'utiliser le langage OWL vu ses caractéristiques syntaxiques et sémantiques permettant de bien refléter le domaine d'application avec une structure arborescente claire permettant d'hiérarchiser les concepts du domaine.

Rappelons que OWL est un langage de représentation d'ontologies destinées à être publiées et partagées sur le web. Il permet donc d'hiérarchiser les classes représentant les concepts d'un domaine particulier à travers des relations de type « is-a », tout en spécifiant leurs propriétés ainsi que leurs instances. Pour répondre aux besoin d'expressivité ontologique croissantes, OWL s'est évolué en mettant en œuvre trois sous langages d'OWL. En fonction de leurs besoins, les développeurs d'ontologies choisissent le sous langage qui leur convient le plus, qu'il s'agisse de la simplicité et de la facilité d'implémentation, (OWL Lite), de la richesse d'expression, (OWL-DL), ou d'éviter les restrictions qui leur y imposées, par exemple, une classe ne

peut pas être une instance d'une autre classe, (OWL Full). Dans notre travail, et vu qu'on s'intéresse beaucoup plus à la richesse d'expression, nous avons choisi d'utiliser OWL-DL.

L'étape de l'ontologisation peut être menée à travers plusieurs outils orientés ontologisation, tels que DOE, OntoEdit, WebODE, WebOnto, OilEd, Protege, etc, (voir chapitre 2). Nous avons donc choisi l'ontologisation sous l'éditeur protege, vu la souplesse d'implémentation à travers son interface modulaire, permettant l'édition, la visualisation, ainsi que le contrôle (vérification de la consistance des hiérarchies et des classifications) d'ontologies, en plus de la richesse fonctionnelle à travers le librairie assez vaste de plugins qu'il détient tels que DataMaster, Jambalaya, Prompt, etc.

V.3.3 L'opérationnalisation

Dans le but d'obtenir une ontologie opérationnelle sous une certaine application, cette phase consiste, comme nous l'avons cité précédemment à transcrire l'ontologie dans un langage formel et opérationnel doté de services inférentiels permettant de mettre en œuvre des raisonnements , ce qui rend l'ontologie intégrable et utilisable par un SBC. Il est à noter que notre objectif n'est pas l'opérationnalisation des ontologies formelles obtenues de l'ontologisation, mais c'est plutôt l'implémentation en premier lieu d'un outil déjà existant, PROMPT, puis la proposition d'un algorithme ainsi que son implémentation pour la fusion de ces deux ontologies, Topo et Diagno, pour obtenir une nouvelle ontologie plus grande et plus compète assurant une plus vaste couverture du domaine en question, mais cela n'empêche d'opérationnaliser l'ontologie résultat de la fusion dans un autre travail.

V.4. Le processus général de la fusion d'ontologies

Nous avons vu dans le chapitre 3 que lors de la fusion de deux ontologies deux cas de figure peuvent se présenter. Le premier cas où les deux ontologies sources vont disparaître et une nouvelle ontologie regroupant les connaissances contenues dans les deux ontologies apparaît, et le deuxième cas, où les deux ontologies sources persistent mais qui seront enrichies par des axiomes bridges représentant les correspondances et jouant les rôles d'un intermédiaire entre eux.

Dans notre cas, le processus de la fusion des deux ontologies résulte toute une nouvelle ontologie, en commençant par l'identification des points communs entre eux. Ces derniers vont alors être fusionnés dans l'ontologie résultat de la fusion. Le processus général de la fusion d'ontologies est aussi constitué de trois étapes majeures:

V.4.1 L'importation des ontologies

Les deux ontologies en entrée ne peuvent être fusionnées si elles sont spécifiées dans différents langages de représentation. Dans tel cas, elles doivent être converties en un format de représentation commun. Dans notre cas les deux ontologies sources sont conçues et spécifiées de la même manière, et toutes les deux sont représentées à travers le langage de représentation OWL, et donc, pour être fusionnées, elles sont directement importées dans un outil de fusion qui va effectuer la fusion proprement dite.

V.4.2 L'identification de similarités

Dans notre cas les similarités sont identifiées automatiquement, tout en cherchant les ressemblances et les dissemblances entre les concepts des deux ontologies, en nous basant sur les comparaisons des noms de classes.

V.4.3 La fusion d'ontologies

Lors de ce travail, nous avons réalisé la fusion proprement dite par deux manières différentes. La première utilise un outil de fusion que nous avons choisi selon le contexte d'utilisation (fusion de deux ontologies construites à partir de BDDs relationnelles représentant une Turbine à Vapeur dans le domaine de la maintenance industrielle), en plus du langage de représentation (OWL-DL), il s'agit bien donc de l'outil, PROMPT, qui utilise la technique Top-Down, et qui guide l'utilisateur à effectuer la fusion en lui proposant une liste de suggestions, à partir de laquelle, il va choisir l'action à exécuter jusqu'à l'achèvement de toutes les actions de la fusion attendues par l'utilisateur. Il s'agit bien donc, d'un processus semi-automatique où l'intervention humaine est inévitable.

Alors que lors de la deuxième partie de ce travail, nous avons proposé un algorithme de fusion qui suit également la technique TOP-DOWN, mais qui diffère de

PROMPT. La différence réside essentiellement, dans le point de l'identification de similarités, tel qu'il initie tout d'abord l'ontologie résultat de la fusion par l'une des deux ontologies, puis effectue une recherche séquentielle (par le nom du concept) des composants de l'ontologie restante, dans l'ontologie résultat. Si le concept en question existe aussi dans l'ontologie résultat alors il effectue une fusion des deux concepts en question, sinon il le copie directement dans l'ontologie résultat.

Un autre point de différence réside dans le fait que cet algorithme est totalement automatique, l'identification de concepts à fusionner est automatique ne demandant aucune intervention humaine.

Un troisième point de différence est que cet algorithme est indépendant de tout autre outil ou environnement de développement, contrairement à PROMPT qui est un plugin dans l'environnement Protege et donc on doit avoir cet environnement pour pouvoir effectuer la fusion.

Dans ce dernier chapitre nous allons décrire notre système général. Nous allons d'abord appliquer l'algorithme PROMPT pour effectuer la fusion des deux ontologies. Nous proposons ensuite un nouvel algorithme de fusion des deux ontologies indépendamment de tout environnement ou éditeurs de construction et ne demandant aucune intervention humaine. Nous implémentons enfin allons nos conceptions et discutons les résultats.

VI.1. Conception de l'approche générale proposée

Dans ce travail nous avons opté pour une implémentation du processus de fusion de deux ontologies dédiées à être exploitées dans le contexte de la maintenance industrielle. Ces deux ontologies représentent la Turbine à Vapeur et spécialement celle exploitée au sein de l'entreprise de SONELGAZ pour la génération électrique.
L'approche générale que nous avons proposée est illustrée par la figure suivante:

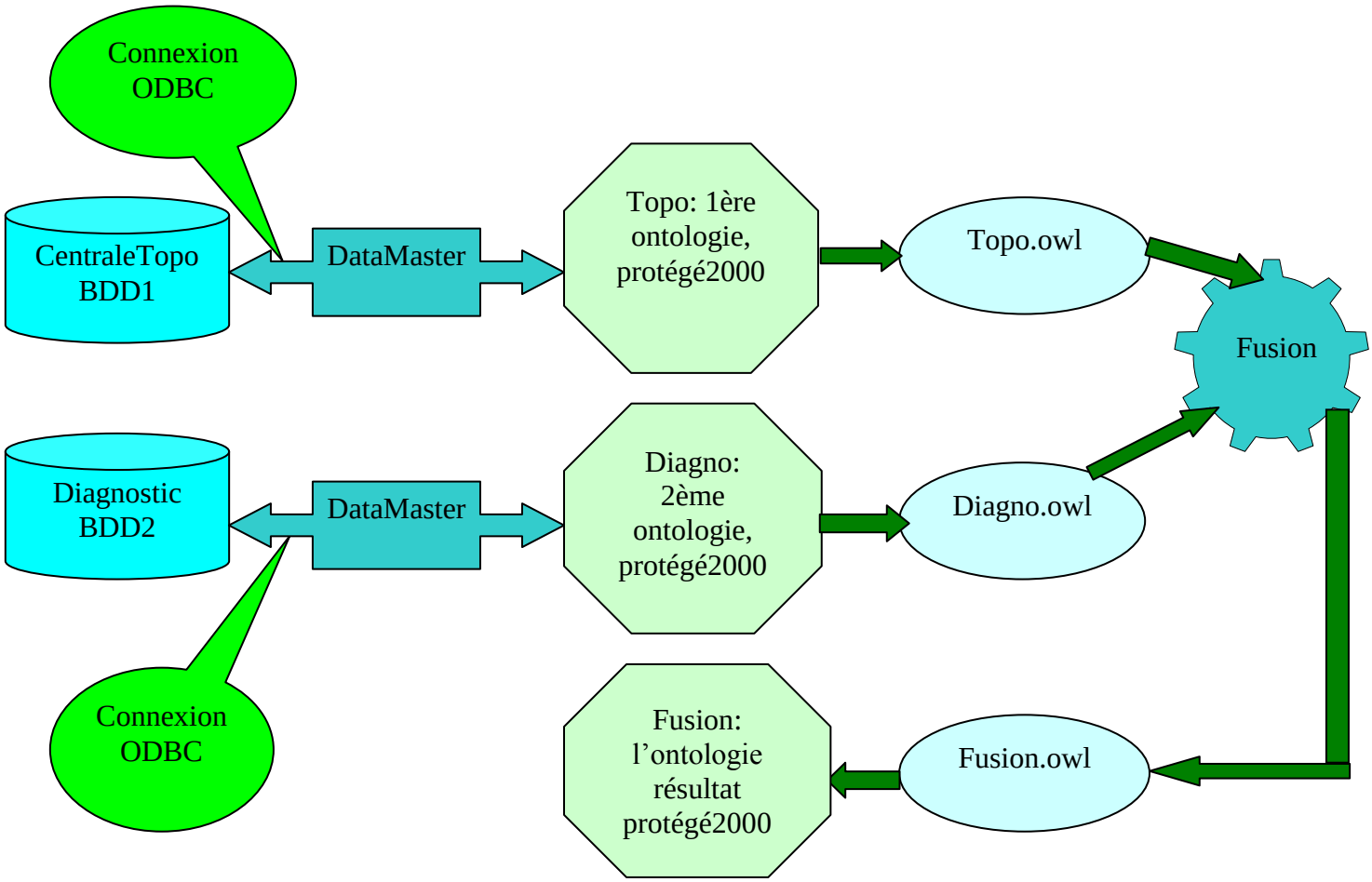


Figure VI.1: Approche générale proposée.

Nous avons commencé par la construction ou le développement des deux ontologies Topo et Diagno. Etant donné que les connaissances extraites à partir des interviews entre les ingénieurs de connaissances et les experts du domaine en plus d'autres documents techniques concernant les Turbines à Vapeur sont déjà stockées dans deux bases de données relationnelles sous l'éditeur Microsoft Office Access, et qui se diffèrent l'une de l'autre dans leurs contextes d'utilisation. Etant donné le gros volume de ces BDDs (plus de 2000 composants de la Turbine à Vapeur), une construction manuelle de ces deux ontologies semble longue et coûteuse. Pour contourner ce problème, tout en optimisant les coûts de développement, nous avons opté pour une construction automatique des ontologies en utilisant l'outil DataMaster, qui permet d'importer les concepts et leurs attributs ainsi que instances à partir d'une base de données relationnelles vers un éditeur d'ontologies. Dans notre cas, Protege qui permet de sauvegarder les deux ontologies sous format OWL, pouvant ainsi être fusionnées en utilisant l'outil de fusion d'ontologies PROMPT:

DataMaster: Il s'agit d'un plugin de Protege permettant d'extraire les données à partir d'une seule ou de plusieurs BDDs relationnelles en une seule ontologie avec des espaces de noms différents. Il produit une ontologie sous format .frame mais aussi sous format .owl, selon le choix de l'utilisateur, offrant ainsi un enrichissement sémantique et/ou syntaxique des données tout en assurant une interopérabilité des bases de données.

VI.1.1 Développement des ontologies Turbine à Vapeur

Nous allons suivre le processus général de développement d'ontologies vu dans la section précédente pour développer nos deux ontologies Topo et Diagno:

VI.1.1.1 Conceptualisation de l'ontologie Turbine à Vapeur:

VI.1.1.1.1 Les étapes de construction des ontologies Turbine à Vapeur

Dans ce qui suit nous suivons l'approche de conception d'ontologies proposée par [NOY et al, 01], pour concevoir l'ontologie Turbine à Vapeur, Topo, (resp Diagno):

- **Etape 1:** Déterminer l'envergure: Nous visons à construire deux ontologies nommées respectivement, Topo et Diagno, décrivant chacune une Turbine à Vapeur dans deux différents contextes, cette dernière est utilisée pour la génération électrique au niveau de l'entreprise de SONELGAZ. Ces deux ontologies sont destinées à être exploitées après leur fusion fournissant ainsi des réponses aux questions des techniciens et des opérateurs de cette Turbine à Vapeur dans le domaine de la maintenance industrielle.
- **Etape 2:** Considérer la réutilisation des ontologies existantes sur le même domaine: Cette étape est reportée pour la deuxième partie de notre système proposé, après la conception et la construction des deux ontologies Topo et Diagno, qui sont toutes les deux représentées sous le langage OWL, permettant ainsi leur intégration à travers l'opération de la fusion, sans aucun empêchement.
- **Etape 3:** Enumérer les termes importants dans l'ontologie: Dans notre cas, nous avons pris le nom de chaque table dans les deux BDDs Access, comme étant le nom d'un concept, et dans chaque table, chaque ligne représente une instance de ce concept, et les noms des colonnes représentent les attributs des concepts. Dans ce qui suit, nous énumérons les termes importants dans chacune des deux ontologies Topo et Diagno:

  L'ontologie Topo: 30B, 30BA, 30BB, 30BT, 30C, 30CA, 30CB, 30P, 30PA, 30PB, code, description, zone, famille, fonction et code-parent.

  L'ontologie Diagno: 30D, 30Y, 30B, 30C, 30R, 30V, 30J, 30S, 30H, 30N, 30U, 30P, code, description, code-composant, description-composant, code-symptôme, description-symptôme, code-défaut, description-défaut, code-cause, description-cause, code-remède, description-remède, et code-parent-géographique.
- **Etape 4:** Définir les classes et la hiérarchie de classes: A partir de la liste créée lors de l'étape précédente, nous identifions les classes de chaque ontologie comme suit:

L'ontologie Topo a pour classe: 30B, (ayant pour sous classes: 30BA, 30BB, et 30BT), 30C, (ayant pour sous classes: 30CA, et 30CB), et 30P, (ayant pour sous classes: 30PA, et 30PB).

L'ontologie Diagno a pour classes: 30D, 30Y, 30B, 30C, 30R, 30V, 30J, 30S, 30H, 30N, 30U, et 30P, tous de même niveau (il n'y a pas des sous classes entre eux).

Dans ce qui suit, nous présentons graphiquement les hiérarchies de classes dans chaque ontologie:

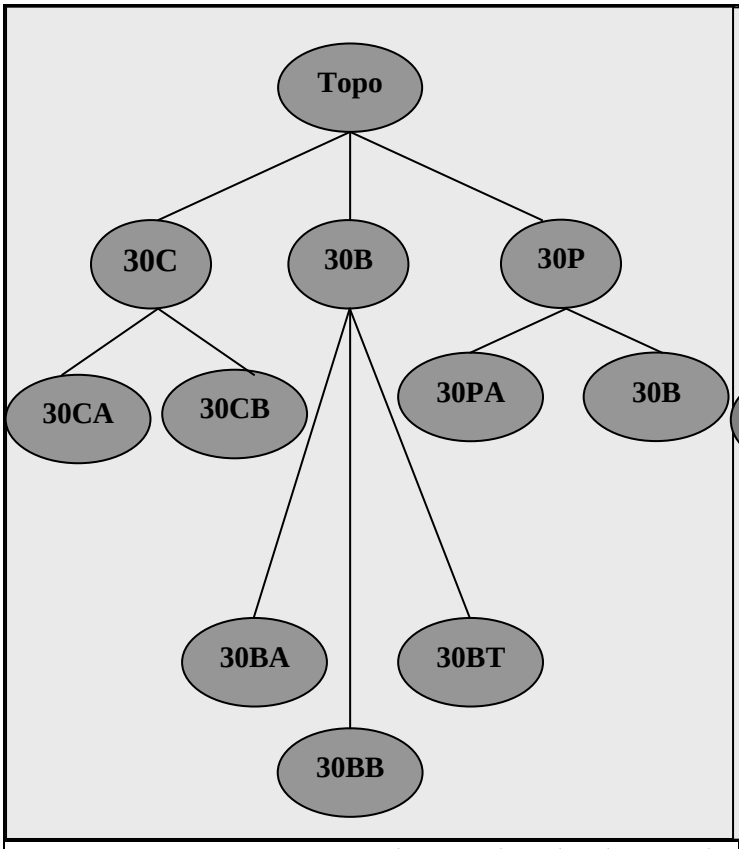


Figure VI.2: Figure 11: L'hiérarchie de classes de l'ontologie Topo

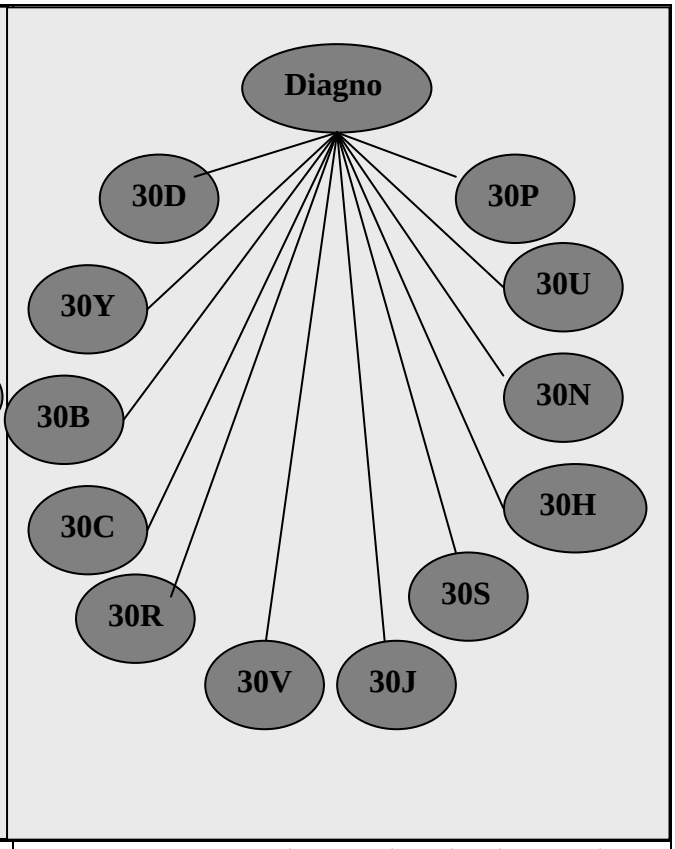


Figure VI.3: L'hiérarchie de classes de l'ontologie Diagno

- **Etape 5:** Définir les propriétés associées aux classes avec les termes restants: A partir des termes restants dans la liste énumérée dans l'étape 3 nous déduisons les propriétés associées aux classes de chaque ontologie comme suit:

  Dans l'ontologie Topo, chaque concept est décrit par les propriétés suivants: code, description, zone, famille, fonction et code-parent.

Dans l'ontologie Diagno, chaque concept est décrit par les propriétés suivants: code, description, code-composant, description-composant, code-symptôme, description-symptôme, code-défaut, description-défaut, code-cause, description-cause, code-remède, description-remède, et code-parent-géographique.

- **Etape 6:** Définir les facettes des attributs (les contraintes qui s'appliquent sur les propriétés): Dans ce qui suit nous présentons un exemple d'une facet appartenant à l'ontologie Topo:

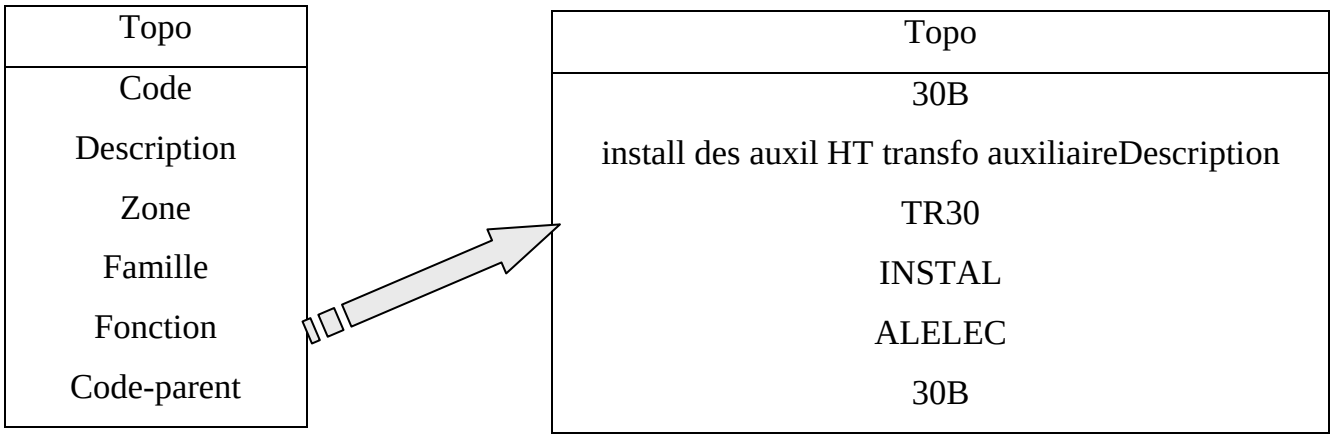


Figure VI. 4: Exemple d'une facet appartenant à l'ontologie Topo.

Code	Description	Zone	Famille	Fonction	Code-parent
30B	install des auxil HT transfo auxiliaire Description	TR30	INSTAL	ALELEC	30B

Figure VI.5: Exemple d'une instance appartenant à l'ontologie Topo.

- **Etape 7:** Créer les instances de classes de la hiérarchie: Dans ce qui suit, nous présentons un exemple d'instance du concept 30B dans l'ontologie Topo:

VI.1.1.1.2 Conceptualisation de l'ontologie Turbine à Vapeur selon le modèle en Oignon

Rappelons que notre objectif n'est pas seulement la conception et la construction des deux ontologies à partir de deux différentes sources de données décrivant une Turbine à Vapeur selon deux différents contextes, Topologique et

Diagnostique. C'est en plus, sur ces deux ontologies nous allons appliquer un algorithme de fusion pour obtenir une seule ontologie plus large et plus complète couvrant un domaine d'application plus vaste. La multitude des sources de données qu'on va traiter a guidé notre choix vers la conception basée sur le modèle en Oignon.

Alors, pour le développement de l'ontologie Turbine à Vapeur, en nous basant sur le modèle en Oignon chaque couche de l'ontologie va être représentée comme suit:

- **La couche de l'Ontologie Conceptuelle Canonique:** Les OCC représentent les concepts importés de chaque BDD, qui utilise chacune un vocabulaire canonique qui est différent l'un de l'autre, sous forme de modèles conceptuels formels et partageables entre eux. Cela revient à dire que nous devons concevoir une OCC par sources de données ne contenant que les concepts primitifs de cette dernière. Comme nous avons deux sources, nous allons établir deux OCCs. En d'autres termes, chaque BDD source va représenter un sous ensemble de l'OCC. Par exemple, nous obtenons OCCs1 qui représente la première source « CentraleTopo » et OCCs2, qui repésente la deuxième source « Diagnostic ». Chaque sous ensemble OCCs1 et OCCs2, offre les mêmes capacités formulés par [JEA 07], tels que:
  - La description canonique et précise de chaque concept par source de données.
  - La base solide pour l'échange d'informations entre différentes sources.
- **La couche de l'Ontologie Conceptuelle Non Canonique:** Les OCNC sont dans notre cas utilisées comme un intermédiaire d'échange de données entre les différents sous ensembles des OCC, OCCs1 et OCCs2, là, et comme le suggère le modèle en Oignon, après avoir conçu les sous ensembles OCCs1 et OCCs2, nous concevons une *ontologie consensuelle* contenant le consensus de concepts primitifs de toutes les sources de données, tout en effectuant des mappings entre les deux ontologies à travers les mécanismes offerts par l'OCNC (aux vues des BDDs), ces mappings jouent un rôle primordial pour l'identification des concepts similaires dans les deux ontologies sources désignées à être fusionnées dans l'ontologie résultante après l'exécution de l'algorithme de fusion d'ontologies.

Il est à noter que ces mappings représentent les opérateurs d'expression des concepts OCNC à partir des concepts OCC, et que les OCCs1 et OCCs2, sont des sous ensembles de l'OCC *consensuelle* Turbine à Vapeur.

En résumé, les opérateurs OCNC servent dans notre cas à l'insertion des constructeurs OWL (désignant chaque source de données), et des mappings entre les différents concepts qui se correspondent entre les deux ontologies, tout en offrant les mêmes capacités formulées par [JEA 07] tel qu'ils sont utilisés pour intégrer d'autres sources ayant déjà leurs propres ontologies.

- **La couche de l'Ontologie Linguistique:** Dans notre cas le rôle des OL se limite à localiser les similitudes existantes entre les différents noms de concepts entre les deux ontologies pour désigner les noms de classes décrivant les mêmes composants pour les fusionner en une seule classe lors du processus de la fusion, et donc cette couche de l'OL offre aussi les mêmes capacités formulés par [JEA 07], tel qu'elle offre des capacités linguistiques sur l'ensemble des concepts (primitifs et définis) du domaine.

Le modèle en Oignon pour notre ontologie est illustré par la figure suivante:

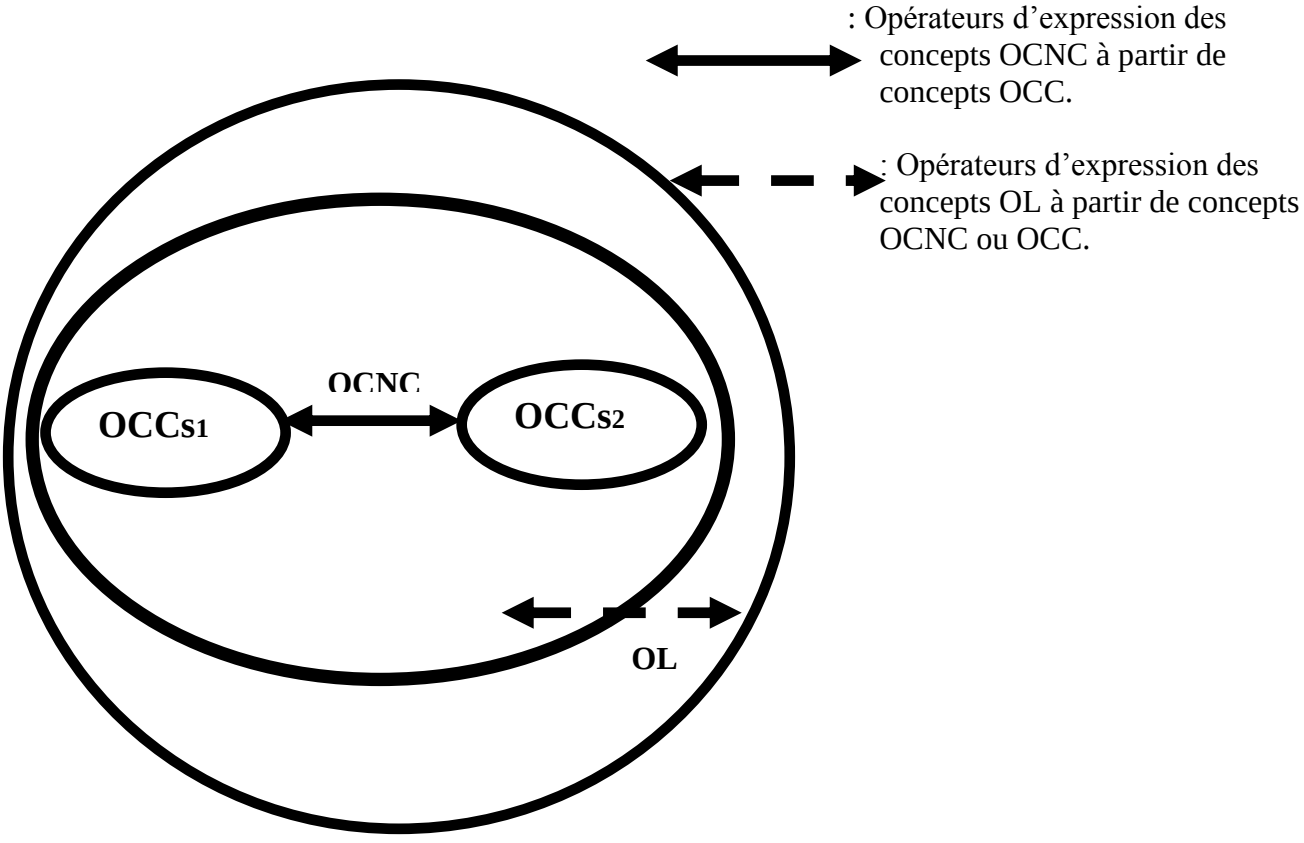


Figure VI.6: Représentation dl'ontologie Turbine à Vapeur selon le modèle en Oignon.

VI.1.1.2 L'ontologisation de l'ontologie Turbine à Vapeur:

Après avoir conçu l'ontologie Turbine à Vapeur lors de la phase précédente, à travers le modèle conceptuel en Oignon, nous procédons dans cette étape à son structuration et sa formalisation sous l'éditeur d'ontologie Protege qui génère son code OWL, dont le mécanisme de raisonnement est basé sur la logique de description, et sa syntaxe est supportée par XML/XML Schéma et donc par son extension RDF/RDF Schéma (voir chapitre 2). Ce dernier permet de représenter et de coder les connaissances selon deux approches différentes:

VI.1.1.2.1 Les approches de représentation de connaissances en RDF/RDF-Schéma

VI.1.1.2.1.1 La représentation par triplet (représentation générique):

Où les connaissances sont codées à travers un ensemble de triplets (subject, predicate, object) eux-mêmes sont de deux formes:

- Des triplets (élément, rdf-schéma: type, entité): Décrivant chaque élément de l'ontologie à travers son type qui est un des entités définies en RDF-Schéma ou OWL, tels que owl: class, ou rdfs: property.
- Des triplets (élément, attribut, valeur): Décrivant chaque élément de l'ontologie en lui assignant des valeurs d'attributs RDF-Schema , tels que rdfs: label, rdfs: comment.

Cette représentation a pour avantage de faire évoluer le modèle d'ontologie mais présente l'inconvénient de ne pas prendre en compte la sémantique des constructeurs d'ontologie.

VI.1.1.2.1.2 La représentation par séparation ontologie/contenu ou ontologie/instances

Elle représente le modèle d'ontologie à travers un modèle relationnel ou relationnel-objet, supporté par un SGBD, et comporte les tables suivantes:

- Class: pour stocker les classes des ontologies.
- SubClassOf: Pour stocker la hiérarchie des classes en indiquant pour chaque classe ses super-classes.
- Property: Pour stocker les propriétés.

Elle contourne les limites de l'approche précédente en prenant en compte la sémantique des constructeurs de l'ontologie, mais le modèle reste figé.

Ces approches de représentation de connaissances permettent alors de définir les ressources, ainsi que les liens entre eux, tout en offrant une richesse sémantique dans la représentation du domaine qui sera encore plus enrichi à travers les notions de base, (décrivant les ressources ainsi que les liens entre eux), offerts par le vocabulaire exploité par l'environnement protege, à savoir: nothing, thing, subclass, domain, range, property, object property, type, etc, tels que nothing, thing, property, object-property sont de type class, subclass a pour domaine la ressource class et a pour co-domaine une autre ressource aussi de type class, une classe peut être une sous classe d'une autre classe, un domaine a pour co-domaine une classe et un co-domaine a pour co-domaine une classe. Cette structure permet donc de classifier et de catégoriser les termes importants décrivant un domaine donné.

Pour mieux éclairer les relations liant les différentes ressources décrivant les connaissances sous l'environnement protégé2000, nous avons proposé de schématiser cette structure à travers le graphe présenté en figure 15:

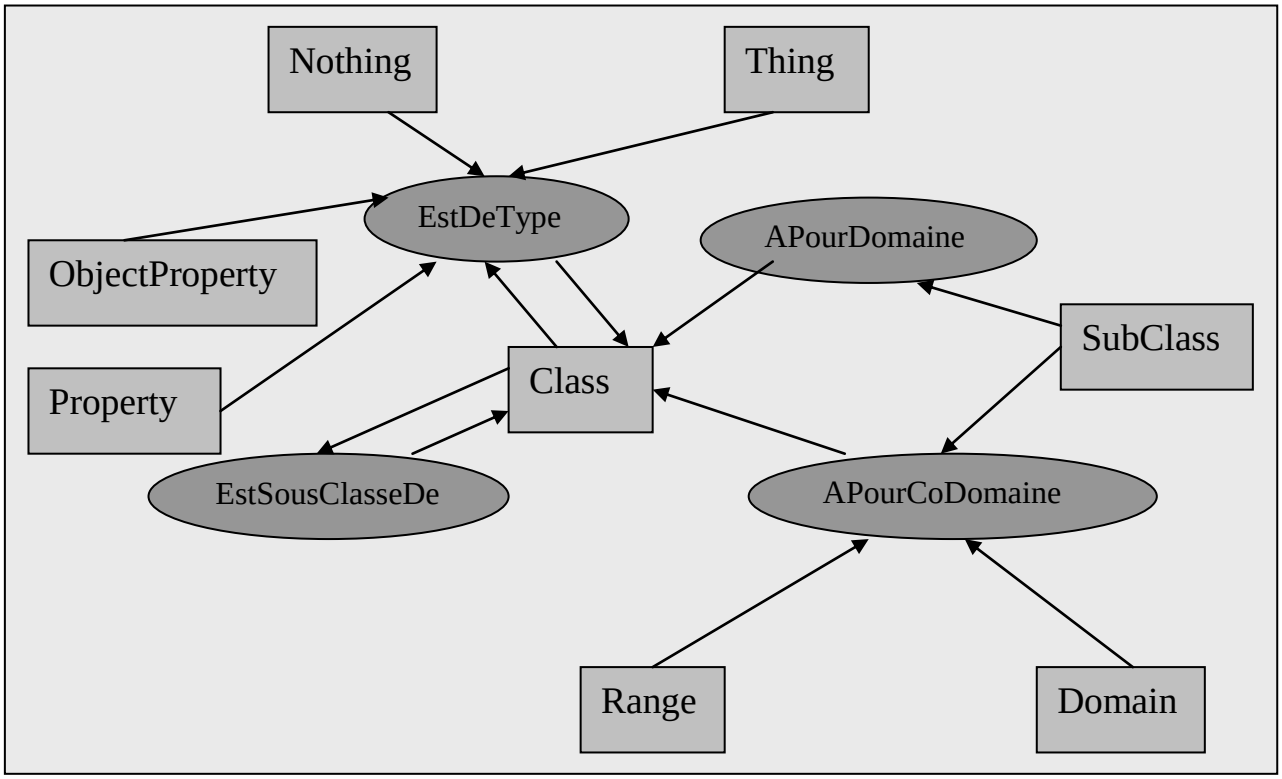


Figure VI.7: Un graphe représentant les liens entre les différentes ressources exploitées par protege.

Un autre format de représentation de connaissances plus particulier, qui détaille la structure de chacune des ressources représentées dans la figure 15, tout en représentant les concepts appartenant au domaine d'application (un concept peut être une tranche, un équipement principal ou un équipement composant), leurs attributs, les types de classes et des attributs ainsi que les relations hiérarchiques et spécifiques entre eux, un schéma graphique illustrant ce format de représentation est donné par la figure 7:

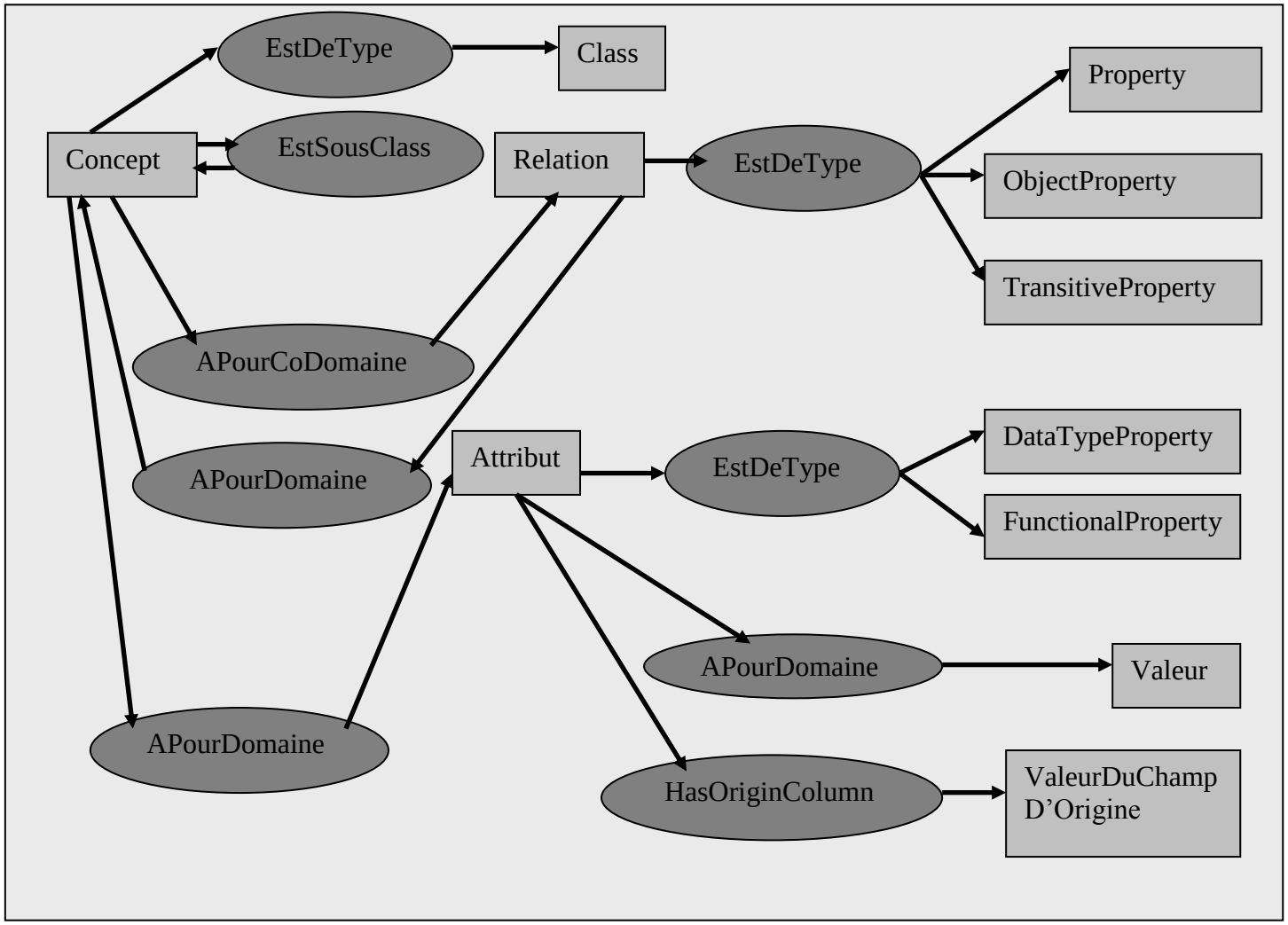


Figure VI.8: Un graphe représentant les liens entre les structures internes des ressources exploitées par protege.

En attribuant des valeurs aux différents attributs affectés à une classe, nous obtenons des instances individuelles de concepts. La figure 17 représente les instances de l'ontologie et les liens existants entre elles, ainsi que les liens d'appartenance avec les classes concernées tels que: une instance est de type concept, et a pour attribut la valeur de l'attribut qui elle-même peut être de type littéral ou chaîne de caractères.

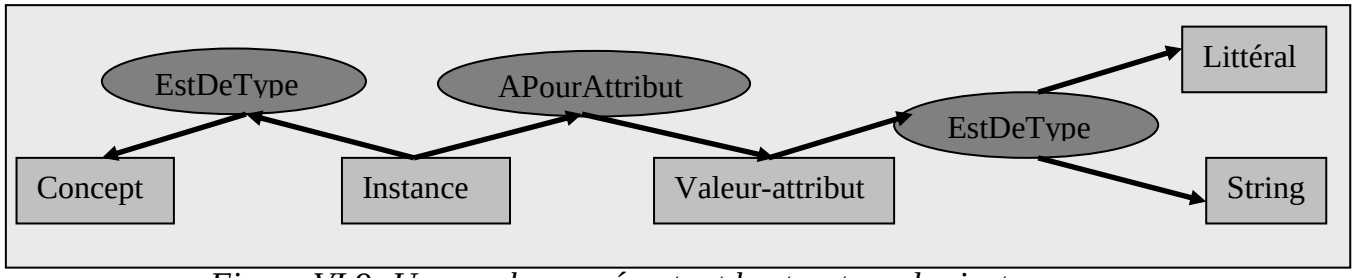


Figure VI.9: Un graphe représentant la structure des instances.

VI.1.1.2.2 Les langages de représentation de l'ontologie Turbine à Vapeur

Après avoir importé l'ontologie Turbine à Vapeur vers Protege, ce dernier génère son code OWL-DL qui se base sur la logique de description et utilise un syntaxe supportée par RDF/RDF-Schéma exploitant des triplets (subject, predicate,object) pour représenter la sémantique des assertions . D'un point de vue syntaxique, RDF utilise le mécanisme d'URI (Uniform Resource Identifier) pour identifier les champs de chaque triplet et plus précisément de référence par URI, par exemple:

http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:Diagnostic#_30B

URI référence

Parmi les lacunes de rdfs qui ont été résolues par owl on trouve:

- Rdfs ne permet pas d'exprimer la disjonction de classe.
- Rdfs ne permet pas de créer de nouveaux classes par combinaison ensemblistes d'autres classes (insertion, union,…).

La portion de code ci-dessous représente un extrait de l'ontologie Diagno sous format owl définissant la syntaxe rdf/xml pour représenter les espaces de noms de la BDD source:

```
<rdf:RDF
   xmlns:dbs="http://www.dbs.cs.uni-duesseldorf.de/RDF/relational.owl#"
   xmlns="http://www.owl-ontologies.com/Ontology1241595998.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:db="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:Diagnostic#"
  xml:base="http://www.owl-ontologies.com/Ontology1241595998.owl">
```

VI.1.1.2.3 Les principales ressources de connaissances sous Protege

VI.1.1.2.3.1 La ressource classe

OWL: Thing et OWL: Nothing sont deux classes prédéfinies, toute classe OWL est une sous-classe d'OWL: Thing et une super-classe d'OWL: Nothing. Les classes sont définies avec un élément OWL: class (OWL: class est une sous classe de rdfs:class).

VI.1.1.2.3.2 La ressource propriété

Les propriétés OWL donnent la capacité d'exprimer des faits au sujet de ces classes et de leurs instances. OWL fait la distinction entre deux types de propriétés

- Les propriétés d'objet qui permettent de relier des instances à d'autres instances
- Les propriétés de type de données qui 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 données est une instance de la classe OWL:DatatypeProperty. Ces deux classes sont elles mêmes sous-classe de la classe RDF rdf:Property.

La portion de code ci-dessous représente un extrait de l'ontologie Diagno sous format owl décrivant le concept 30B:

```
<owl:Class
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:Diagnostic#_30B">
  <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</db:isBridgeTable>
  </owl:Class>
```

La portion de code ci-dessous représente un extrait de l'ontologie Topo sous format owl décrivant le concept 30B:

```
<owl:Class
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:centraleTopo#_30B"/>
 </rdfs:subClassOf>
 <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
 >false</db:isBridgeTable>
 </owl:Class>
```

La portion de code ci-dessous représente un extrait de l'ontologie Topo sous format owl décrivant l'attribut code_parent du concept 30C:

```
<owl:FunctionalProperty
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:centraleTopo#_30C.code_parent"
>
   <db:hasOrigColumnName rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >code_parent</db:hasOrigColumnName>
   <rdfs:domain
rdf:resource="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:centraleTopo#_30C"/>
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 </owl:FunctionalProperty>
```

La portion de code ci-dessous représente un extrait de l'ontologie Topo sous format owl décrivant l'instance 25 du concept 30PB:

```
<db:_30PB
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:centraleTopo#_30PB_Instance_2
5">
   <db:_30PB.zone rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >TR30</db:_30PB.zone>
   <db:_30PB.famille rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >VAN</db:_30PB.famille>
   <db:_30PB.fonction rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >ALCOMB</db:_30PB.fonction>
   <db:_30PB.description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >VAN AMONT FILTRE DOUBLE B GR3</db:_30PB.description>
   <db:_30PB.code_parent rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >30PB61</db:_30PB.code_parent>
   <db:_30PB.code rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >30PB61S103</db:_30PB.code>
 </db:_30PB>
```

VI.1.1.3 Opérationnalisation de l'ontologie Turbine à Vapeur

Après avoir conçu et développer les deux ontologies Topo et Diagno tout en les présentant dans un langage formel, il s'agit lors d'une étape d'opérationnalisation de les transcrire dans un langage formel et opérationnel doté de services inférentiels permettant la mise en œuvre des raisonnements pour qu'elle puisse être intégrées dans un SBC. Néanmoins, nous rappelons que notre objectif n'est pas l'exploitation des deux ontologies Topo et Diagno dans un système opérationnel, mais c'est plutôt les utiliser sous un processus de fusion d'ontologies pour obtenir une ontologie globale plus large et plus complète couvrant un domaine d'application plus vaste, permettant ainsi de répondre plus efficacement aux questions de compétences des techniciens et

des opérateurs utilisant cette Turbine à Vapeur. Lors de l'exploitation de cette ontologie sous un SBC, cette dernière étape reste comme une perspective à atteindre.

VI.1.2 Le processus général de fusion des deux ontologies Topo et Diagno

Dans cette partie, nous allons suivre les étapes du processus général de fusion d'ontologies pour fusionner les deux ontologies Topo et Diagno précédemment construites. Nous rappelons que nous allons faire la fusion des deux ontologies de deux différentes manières, dans la première partie nous allons effectuer la fusion des deux ontologies en utilisant l'outil semi-automatique de fusion d'ontologies, PROMPT, et dans la deuxième partie, nous allons proposer un algorithme de fusion totalement automatique, ne demandant aucune intervention humaine. Nous allons donc faire la distinction entre ces deux algorithmes au fur et à mesure du processus général de fusion des deux ontologies, Topo et Diagno:

VI.1.2.1 L'importation des deux ontologies, Topo et Diagno

Lors du développement des deux ontologies, Topo et Diagno sous l'environnement Protege, ce dernier génère leurs représentations sous le même format et langage de représentation, OWL-DL, n'imposant ainsi aucune contrainte pour la fusion, les deux ontologies sont alors directement importées vers l'outil de fusion de la même manière pour les deux algorithmes.

VI.1.2.2 Identification de similarités

Il s'agit là d'identifier les composants similaires dans les deux ontologies Topo et Diagno. Nous rappelons que cette étape n'est obligatoire que si l'algorithme de fusion est totalement automatique. Lors de la fusion par PROMPT, qui n'est pas purement automatique, cette étape n'est pas prise en compte, PROMPT propose, à travers son interface, à l'utilisateur toutes les combinaisons possible de classes des deux ontologies, et c'est à lui de décider quelles sont les classes similaires qui vont être fusionnées. Alors que dans la deuxième partie où nous allons faire la fusion à travers un nouveau algorithme de fusion qui est totalement automatique, les similarités sont identifiées automatiquement en comparant les noms de classes entre eux, les

noms similaires (par exemple 30B-Topo et 30B-Diagno) correspondent aux mêmes composants et vont être donc automatiquement fusionnés.

VI.1.2.3 La fusion d'ontologies

VI.1.2.3.1 La fusion par PROMPT

Dans cette partie, la fusion des deux ontologies Topo et Diagno se fait à travers une interface interactive orientée utilisateur, où l'algorithme PROMPT propose des suggestions de fusion des concepts appartenant aux deux ontologies en question (To Do List). L'utilisateur lance une des opérations soit à partir de la liste des propositions suggérée par PROMPT, ou en utilisant l'environnement d'édition d'ontologies pour spécifier directement l'opération voulue. Par exemple on sélectionne la classe 30Bde l'ontologie Topo et la classe 30B de l'ontologie Diagno, et on sélectionne *fusion de classes,* comme le type de la fusion à effectuer, puis on lance la fusion en exécutant la commande *« exécuter ».* Ensuite Prompt exécute automatiquement l'opération lancée par l'utilisateur, tout en adaptant les changements additionnels basés sur le type de l'opération, détermine les conflits introduits par cette opération (Conflict List) tout en proposant des solutions et régénère une nouvelle liste de suggestions à l'utilisateur en fonction de la nouvelle situation et de la structure courante de l'ontologie. Et on continue ainsi jusqu'à ce que tous les composants similaires soient fusionnés.

Les deux premières portions de code ci-dessous représentent, respectivement, des extraits des ontologies Topo, et Diagno sous format owl décrivant les concepts similaires, 30B dans les deux ontologies, et la troisième portion de code représente le résultat de la fusion de ces deux classes par l'algorithme PROMPT:

```
<owl:Class
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:centraleTopo#_30B"/>
 </rdfs:subClassOf>
 <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
 >false</db:isBridgeTable>
 </owl:Class>
```

```
<owl:Class
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:Diagnostic#_30B">
  <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</db:isBridgeTable>
  </owl:Class>
```

```
<owl:Class rdf:ID="_30B">
   <isBridgeTable--diagProject rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
   >false</isBridgeTable--diagProject>
  </owl:Class>
```

VI.1.2.3.2 Un algorithme de fusion d'ontologies

Après avoir importé les deux ontologies Topo.owl et Diagno.owl et en nous basant sur les résultats de l'étape de l'identification de similarités, cet algorithme fusionne les classes similaires en une seule classe dans l'ontologie résultat, et copie directement les classes différentes. Dans ce qui suit nous présentons le pseudo code de l'algorithme de fusion que nous avons proposé:

```
Entrées: ontologie A, ontologie B ;
Sortie: ontologie R ;
Début
    R ← A
    booléen ← faux ;
          pour chaque concept Cb de l'ontologie B faire
               pour chaque concept Cr de l'ontologie R faire
                    si Cb.nom = Cr.nom alors     booléen ← vrai
                     sinon  booléen ← faux
                    fin-si
               fin-pour
          fin-pour
      si booléen = vrai alors fusionner-classes (Cb , Cr)
        sinon copier-classe (Cb , R) ; // copier la classe Cb dans l'ontologie R.
      fin-si
fin
```

Les deux premières portions de code ci-dessous représentent respectivement des extraits des ontologies Topo, et Diagno sous format owl décrivant les concepts similaires, 30B dans les deux ontologies, et la troisième portion de code représente le résultat de la fusion de ces deux classes par l'algorithme que nous avons proposé:

```
<owl:Class
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:centraleTopo#_30B"/>
 </rdfs:subClassOf>
 <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
 >false</db:isBridgeTable>
 </owl:Class>
```

```
<owl:Class
rdf:about="http://biostorm.stanford.edu/db_table_classes?DSN=jdbc:odbc:Diagnostic#_30B">
  <db:isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</db:isBridgeTable>
  </owl:Class>
```

```
<owl:Class rdf:about="_30B">
    <isBridgeTable rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</isBridgeTable>
  </owl:Class>
```

VI.2. Implémentation

Nous avons commencé le développement de ce projet par la création d'un projet sous l'environnement Protege, tout en utilisant le langage OWL dont la syntaxe est supportée par le vocabulaire de RDF. Vue qu'on s'intéresse beaucoup plus à la richesse d'expression nous avons choisi d'utiliser OWL-DL, (voir la figure ci-dessous):

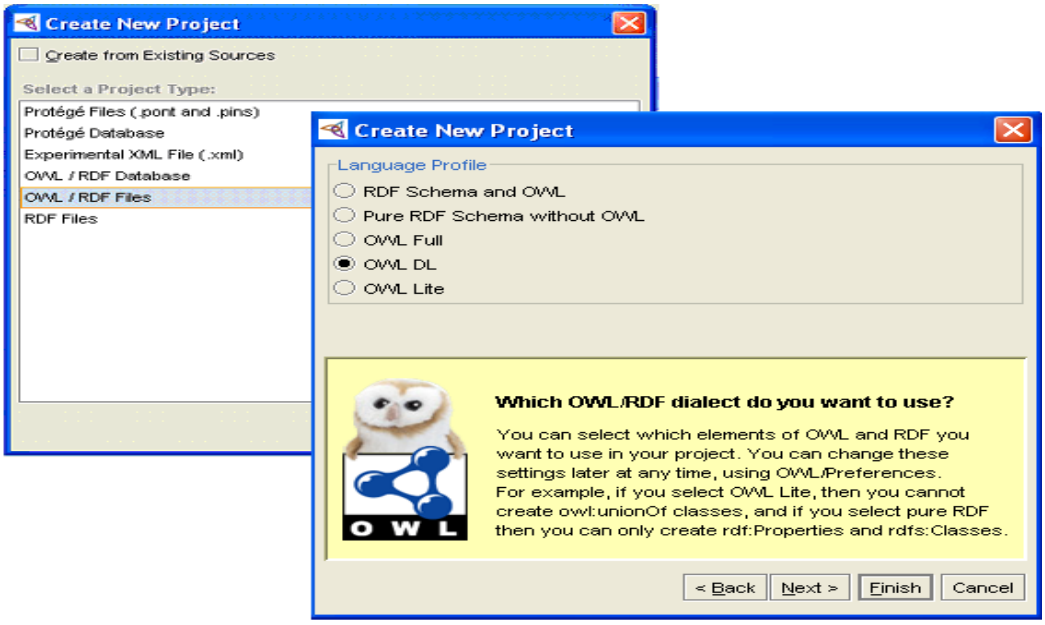


Figure VI.10: Sélection du langage de représentation ontologique, OWL-DL.

Puis nous procédons à la construction automatique des deux ontologies Topo et Diagno à travers le plugin sous Protege, DataMaster. Nous faisons donc appel à ce dernier, (ainsi que d'autres plugins qu'on va utiliser plus tard) à travers une petite configuration sous Protege, (voir la figure ci-dessous):

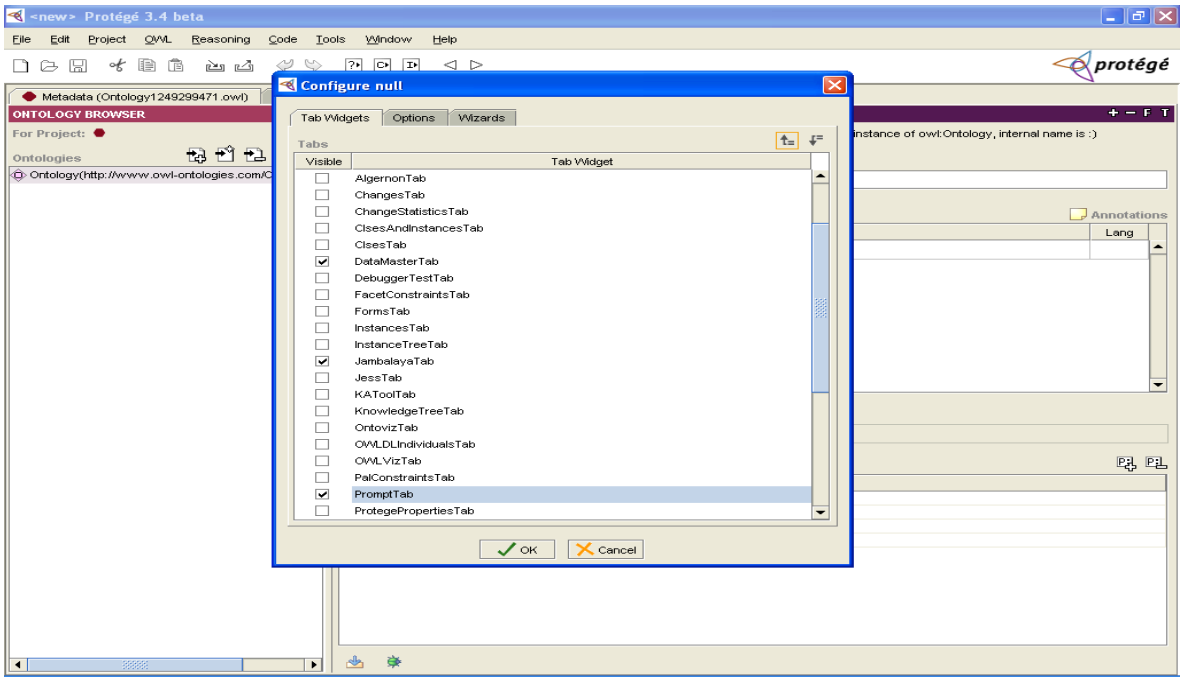


Figure VI.11: Appels aux plugins (DataMaster, Jambalaya et Prompt) à exploiter sous protege.

Pour pouvoir connecter aux BDDs, CentraleTopo et Diagnostic à travers l'outil DataMaster, nous commençons par la création d'une connexion ODBC entre le système d'exploitation que va utiliser DataMaster, et nos BDDs, à partir du fichier système, Sources de Données ODBC, du paneau de configuration, (voir la figure ci-dessous):

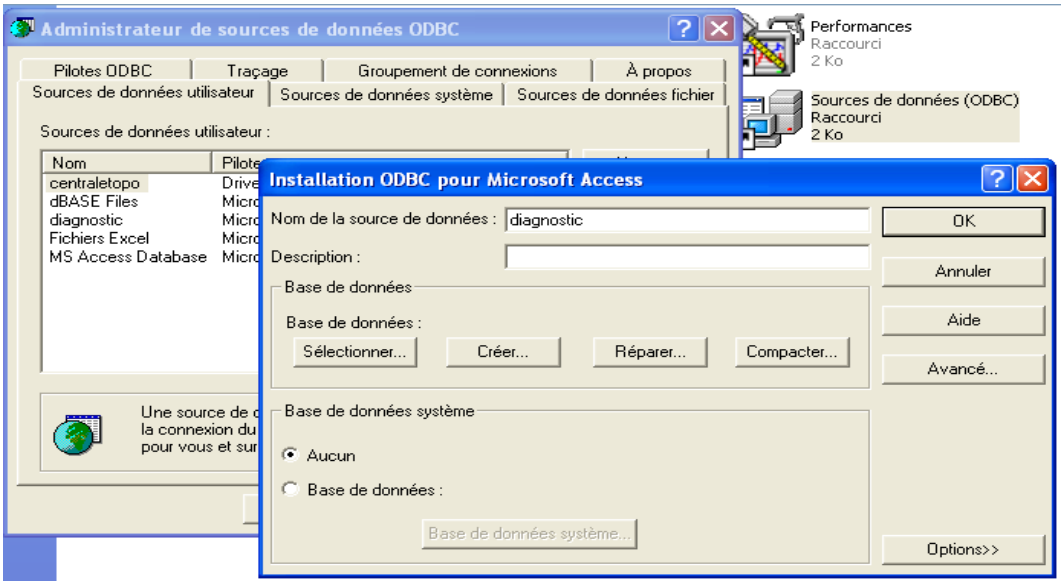


Figure VI.12: Création de la connexion ODBC entre le système d'exploitation et la BDD Diagnostic.

Après s'être connecter aux BDDs, on commence la construction automatique des deux ontologies, après une petite configuration de DataMaster, tout en sélectionnant les options d'importer la BDD connectée dans l'ontologie courante en incluant les noms des colonnes (les attributs) ainsi que le contenu des tables (les instances), (voir la figure ci-dessous):

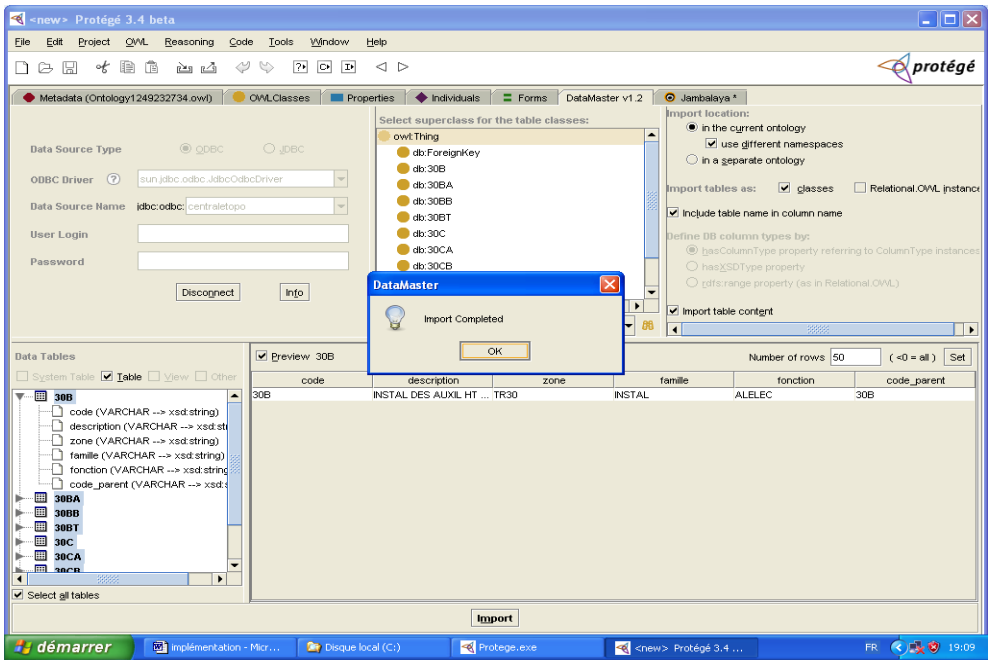


Figure VI.13: Construction automatique de l'ontologie Topo sous DataMaster.

L'ontologie Diagno, (y compris sa hiérarchie de concepts, les attributs ainsi que les instances de chaque concept), est présentée dans la figure suivante:

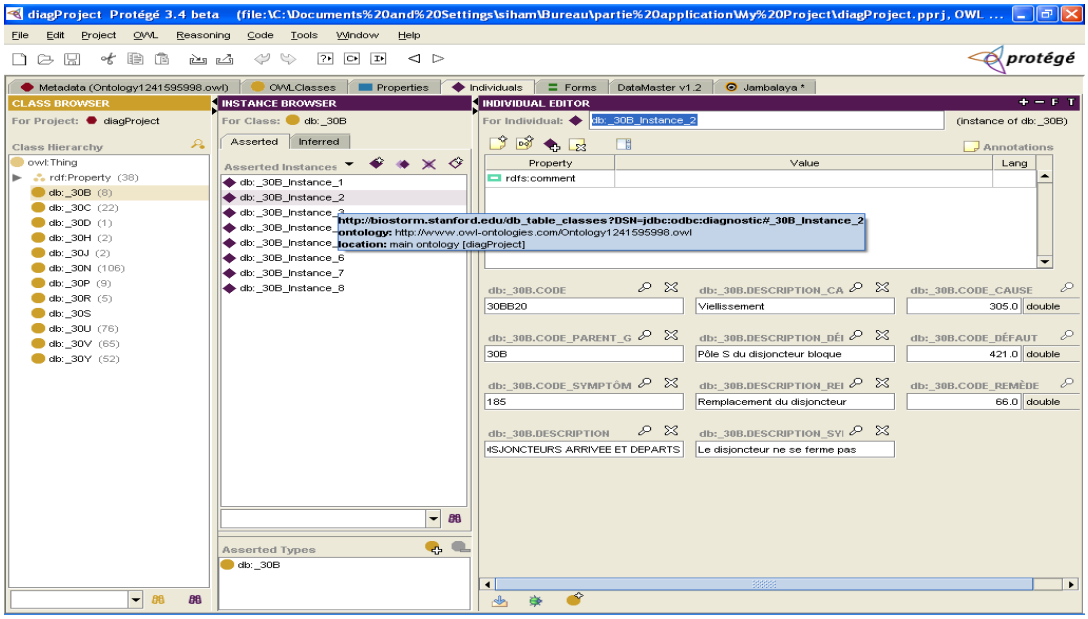


Figure VI.14: Représentation de l'ontologie Diagno sous Protege.

L'hiérarchie de concepts ainsi obtenue peut être visualisée graphiquement sous le plugin de visualisation, jambalaya, (voir la figure ci-dessous):

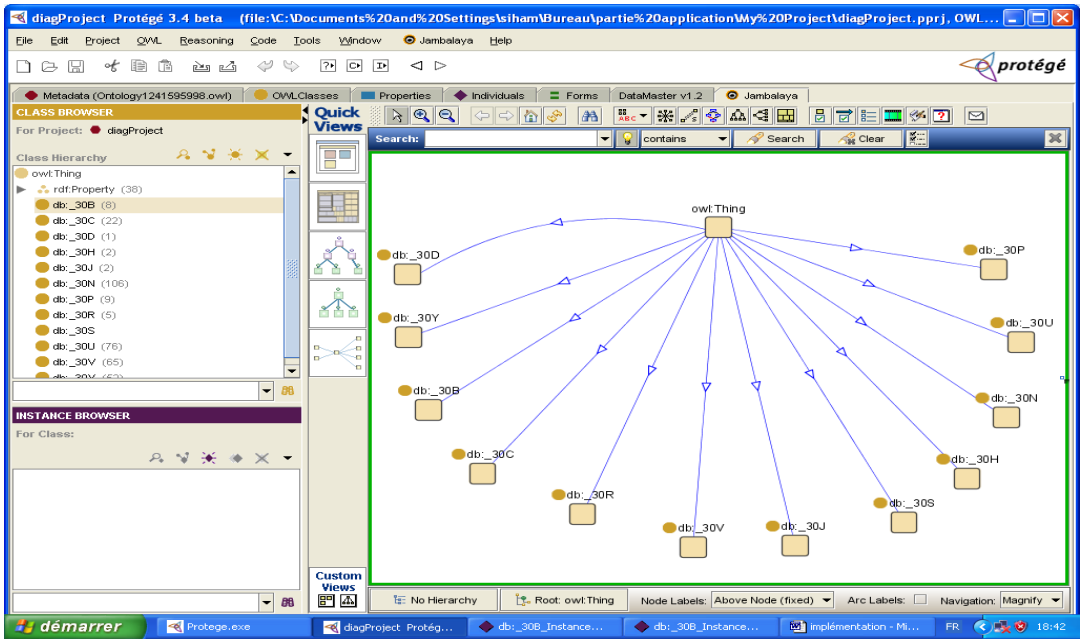


Figure VI.15: La visualisation de l'ontologie Diagno par jambalaya.

Idem pour l'ontologie Diagno, nous présentons l'ontologie Topo ainsi que sa visualisation par jambalaya dans les figures 16 et 17:

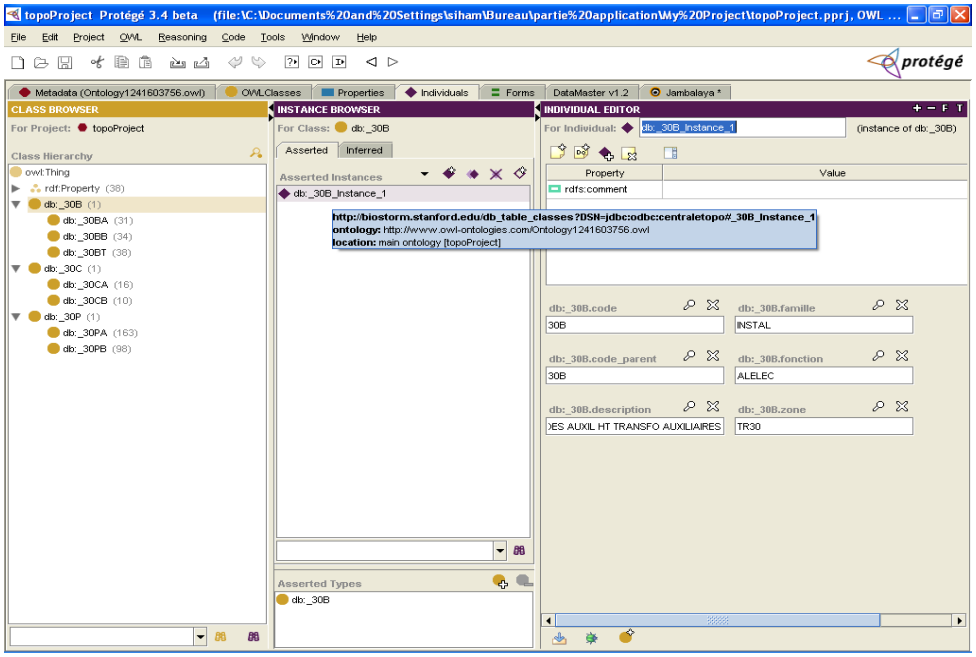


Figure VI.16: Représentation de l'ontologie Topo sous protege.

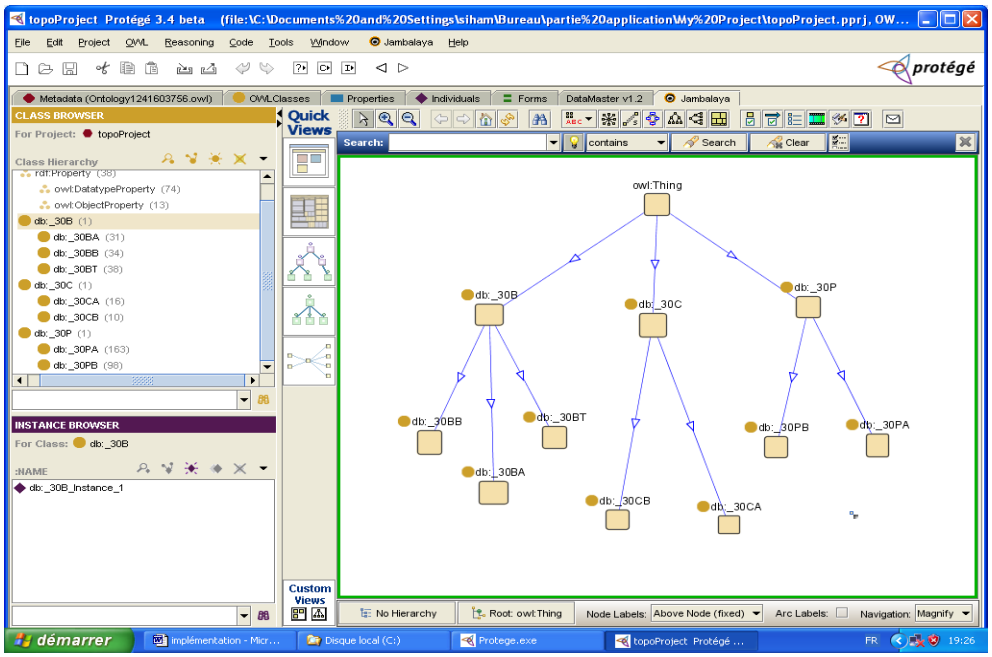


Figure VI.17: La visualisation de l'ontologie Topo par jambalaya.

Maintenant et après la construction automatique des deux ontologies Topo et Diagno, nous procédons en premier lieu à leur fusion via le plugin « PROMPT », en commençant par la sélection du type d'opération, Fusion (Merge), parmi les cinq types proposés. Nous importons ensuite les deux ontologies à fusionner tout en spécifiant leurs chemins, (voir la figure ci-dessous):

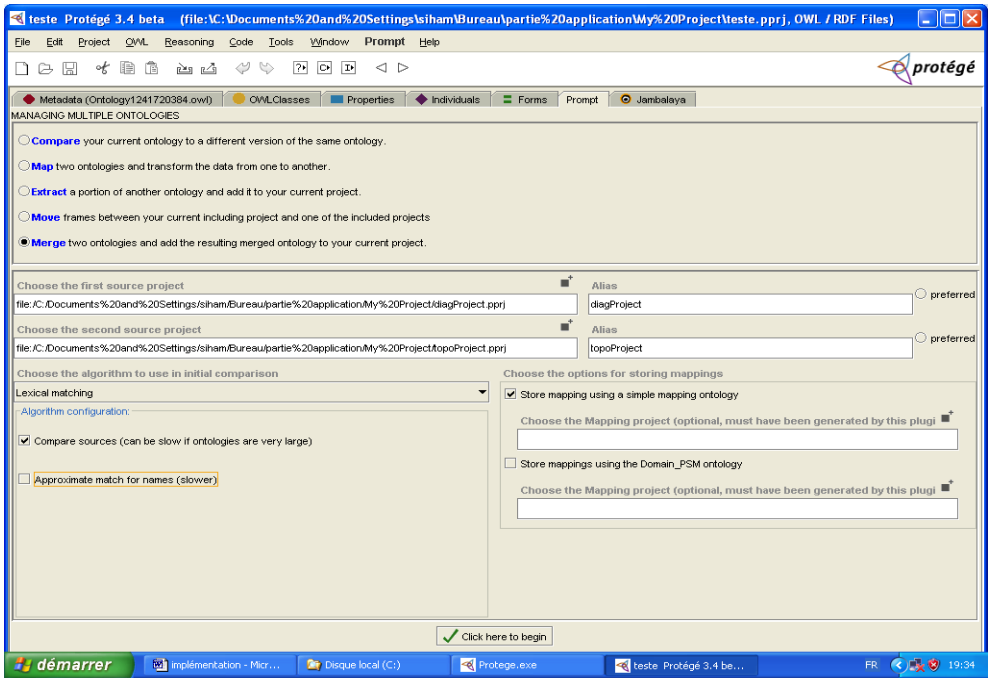


Figure VI.18: Configuration de l'outil Prompt pour la fusion.

Commençons maintenant la fusion proprement dite en spécifiant d'abord le type de la fusion par la fusion de classes. Puis nous identifions les deux classes à fusionner dans les deux ontologies (en rouge et en bleu) et nous exécutons la commande *« Do it »* pour la fusion effective. À la fin de la fusion, nous obtenons l'ontologie en noir (voir la figure ci-dessous):

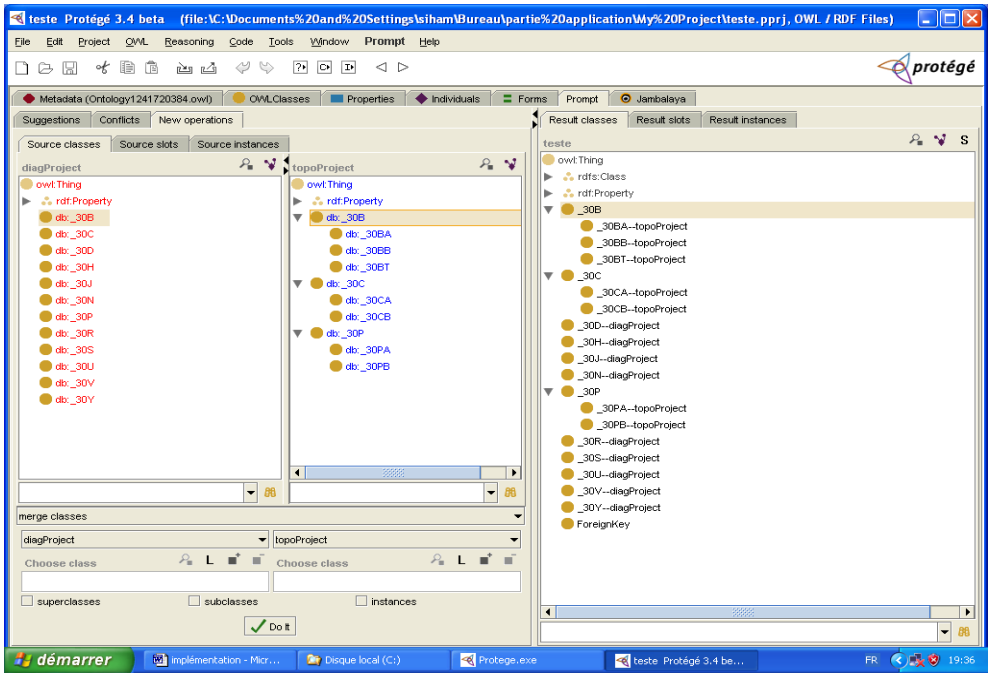


Figure VI.19: La fusion des deux ontologies par PROMPT.

L'ontologie test (ses concepts, ses attributs et ses instances) résultat de la fusion est représentée dans la figure ci-dessous:

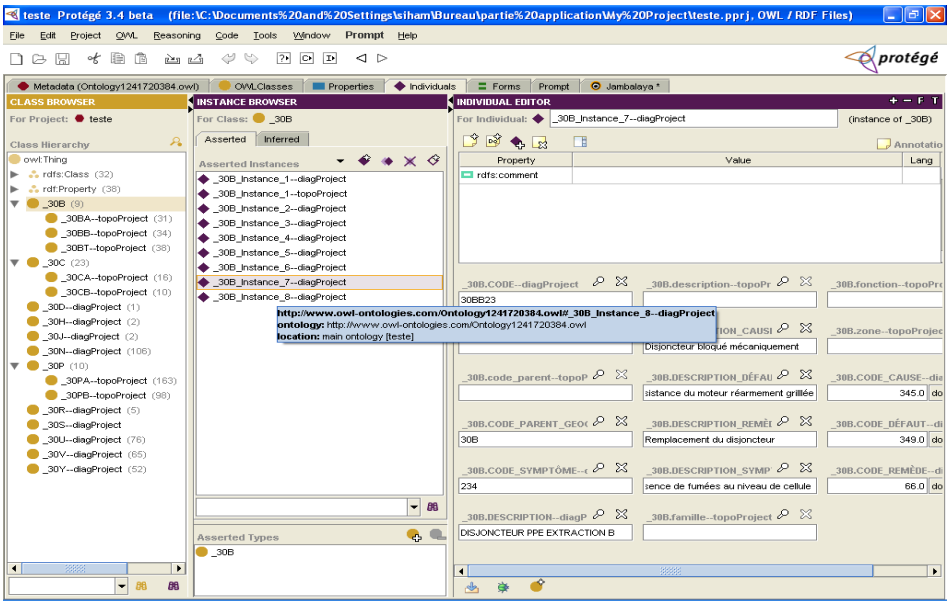


Figure VI.20: Représentation de l'ontologie, résultat de la fusion sous protege.

Une vérification de la consistance de l'ontologie de la fusion obtenue à travers le raisonneur Pellet 1.5.1 est donnée par la figure suivante:

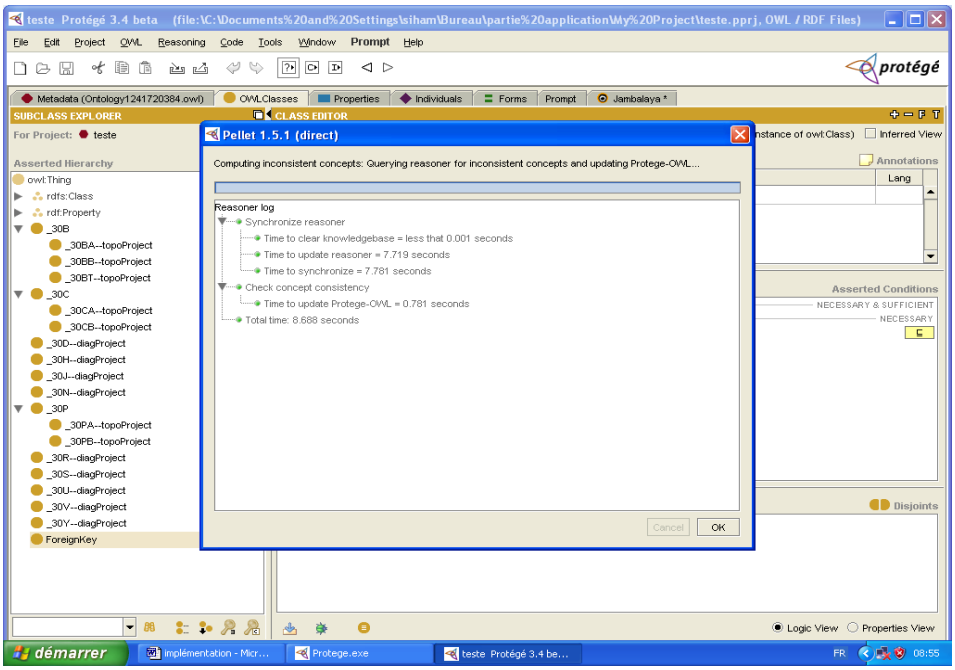


Figure VI.21: Vérification de la consistance par le raisonneur Pellet 1.5.1.

Ensuite, la vérification de la cohérence de l'hiérarchie de classes par le même raisonneur Pellet 1.5.1 est donnée par la figure suivante:

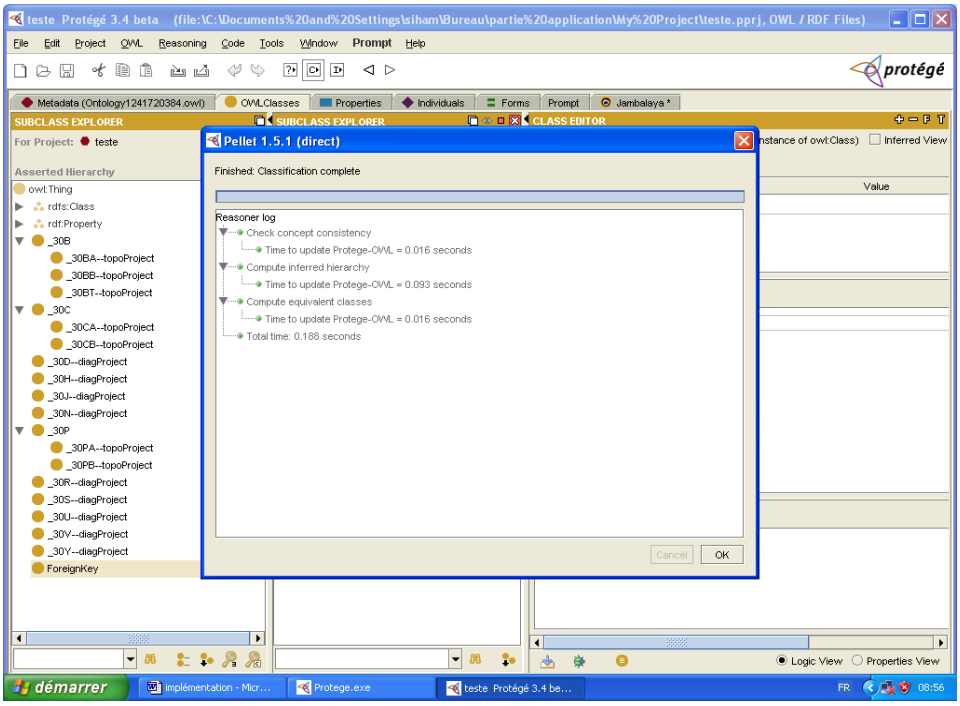


Figure VI.22: Vérification de la cohérence de la hiérarchie de classes par le raisonneur Pellet 1.5.1.

Une visualisation graphique de la fusion donnée par jambalaya est illustrée par la figure suivante:

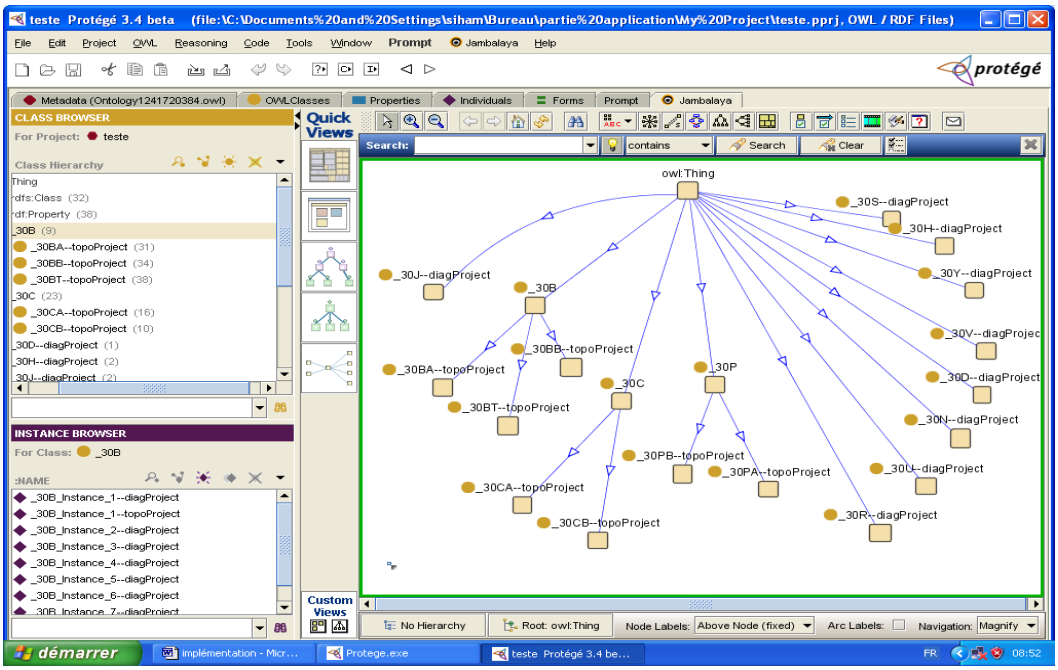


Figure VI.23: La visualisation de l'ontologie test par jambalaya.

Maintenant, nous allons refaire la fusion à travers un nouvel algorithme que nous avons proposé. Pour l'implémenter nous avons développé un macro sous excel en utilisant le langage de programmation ***Visual Basic*** dont l'interface est donnée par la figure suivante:

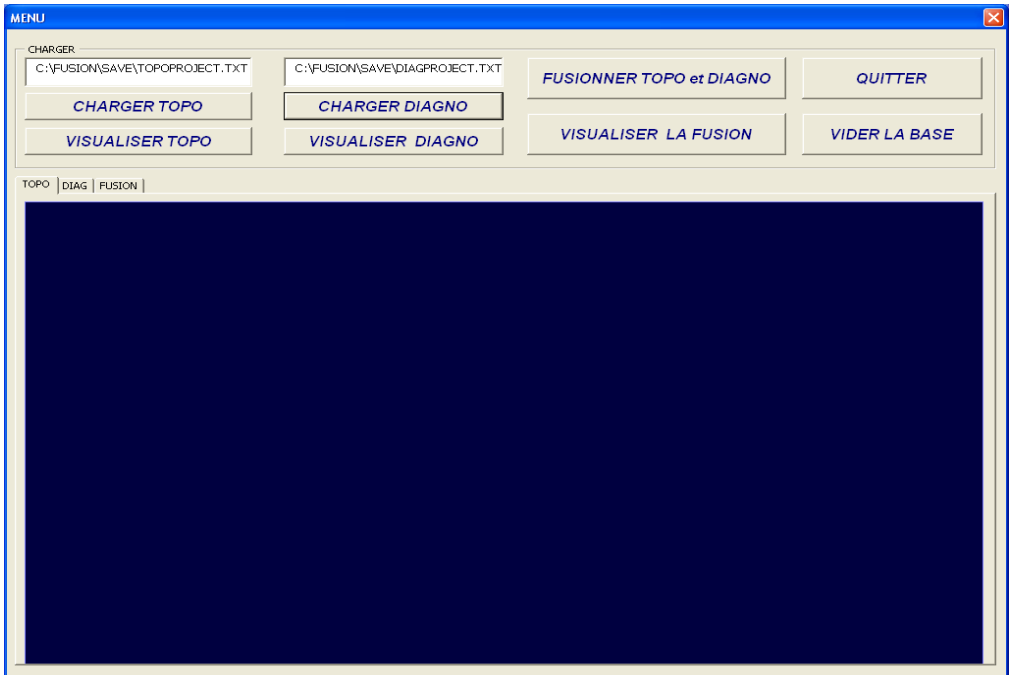


Figure VI.24: Interface de l'outil de la fusion d'ontologies proposé.

Comme on a vu précédemment, nous commençons le processus de la fusion d'ontologies par l'importation des deux ontologies en exécutant les commandes « charger Topo » et « charger Diagno », voir les figures 25, et 26:

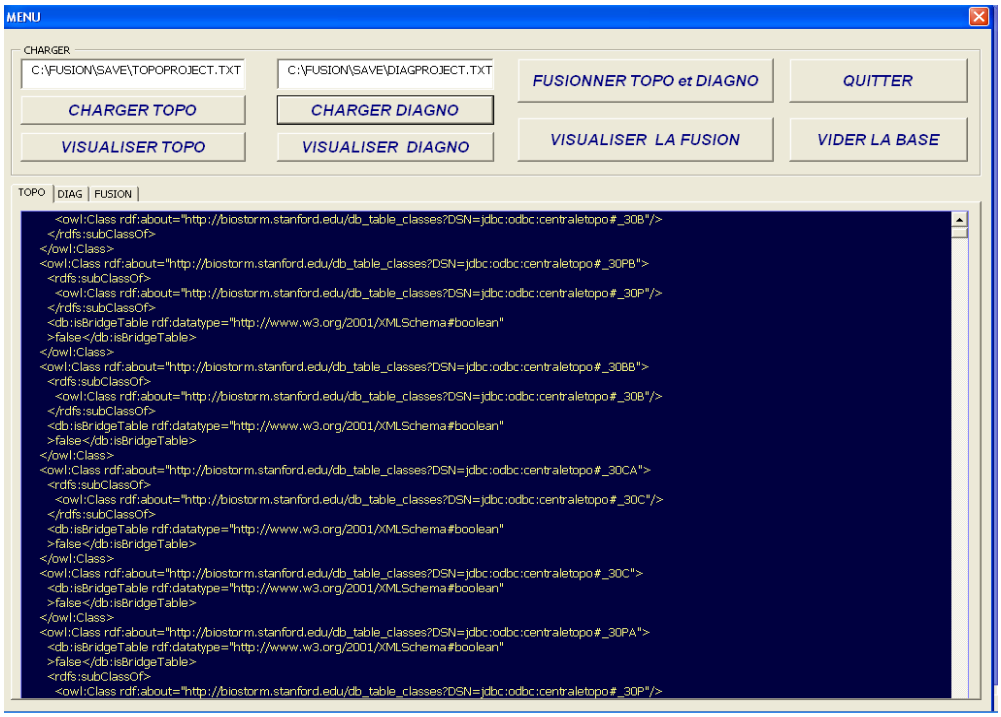


Figure VI.25: L'importation de l'ontologie Topo.

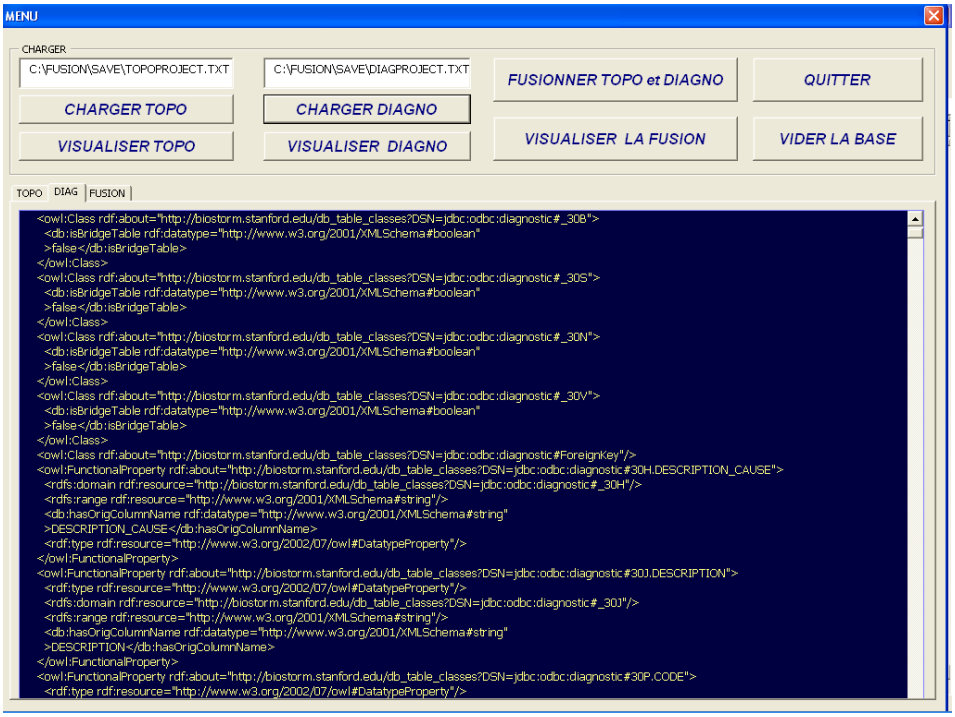


Figure VI.26: L'importation de l'ontologie Diagno.

Le résultat de la visualisation de l'ontologie Topo donnée par jambalaya est donné par la figure suivante:

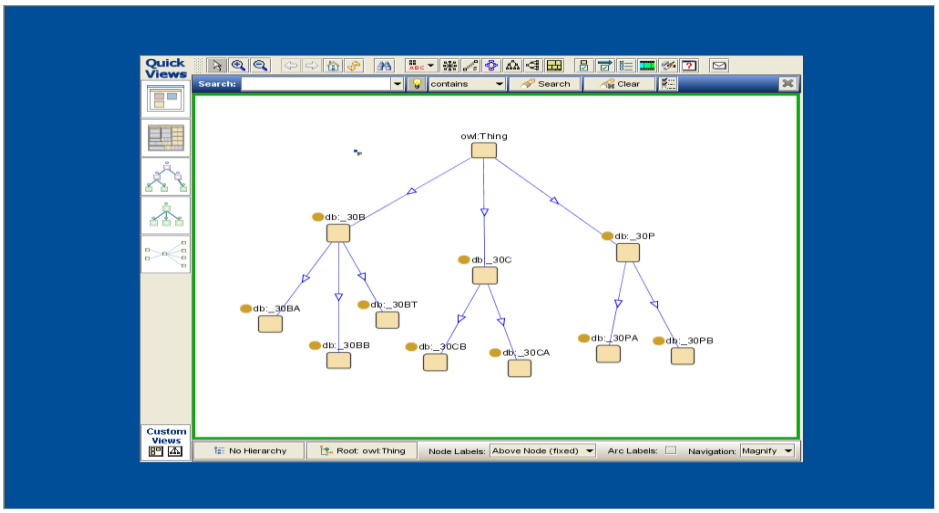


Figure VI.27: La visualisation de l'ontologie Topo par jambalaya.

Le résultat de la visualisation de l'ontologie Diagno donnée par jambalaya est donné par la figure suivante:

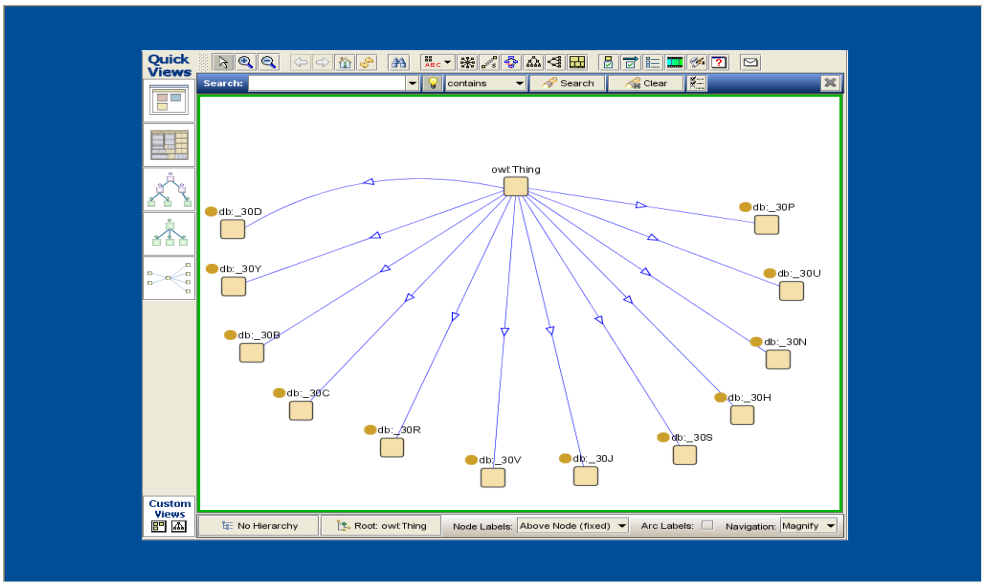


Figure VI.28: La visualisation de l'ontologie Diagno par jambalaya.

Puis, l'étape de l'identification de similarités est transparente par rapport à l'utilisateur, alors que l'étape de la fusion d'ontologies est achevée en exécutant la commande « Fusionner Topo et Diagno ». Le résultat de la fusion sera affiché dans le panneau, Fusion, voir la figure ci-dessous:

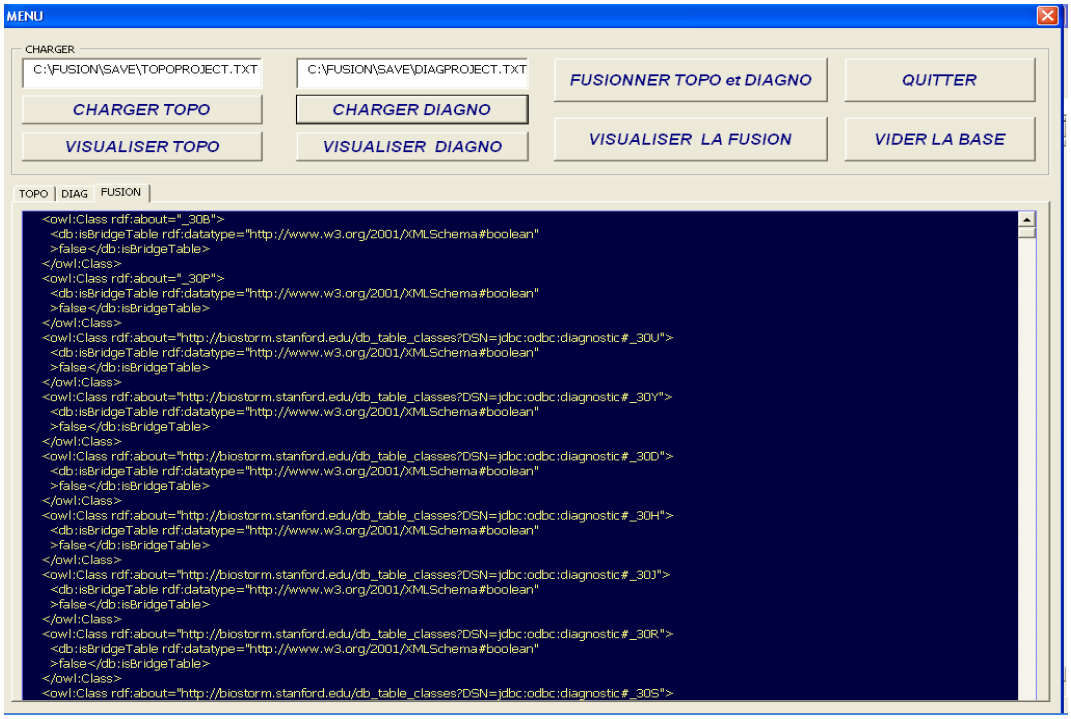


Figure VI.29: La fusion des deux ontologies, Topo et Diagno.

Le résultat de la visualisation de l'ontologie Fusion donnée par jambalaya est donné par la figure suivante:

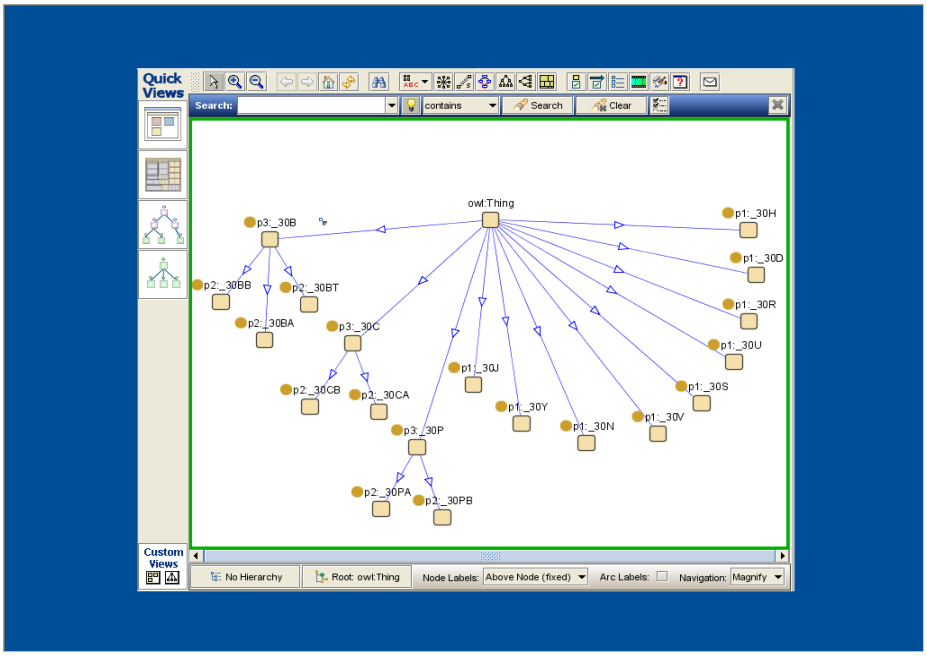


Figure VI.30: La visualisation de l'ontologie fusion par jambalaya.

Après l'atteinte de notre objectif de fusionner deux ontologies sous format OWL en une seule ontologie sous le même format OWL, indépendamment de tout

outil d'édition ou environnement de développement, nous avons essayé d'importer l'ontologie résultat sous l'environnement Protege pour voir ce qu'elle nous donne comme résultats:

L'hiérarchie de concepts de notre ontologie résultat ainsi que la visualisation graphique sous le plugin jambalaya sont données par la figure suivante:

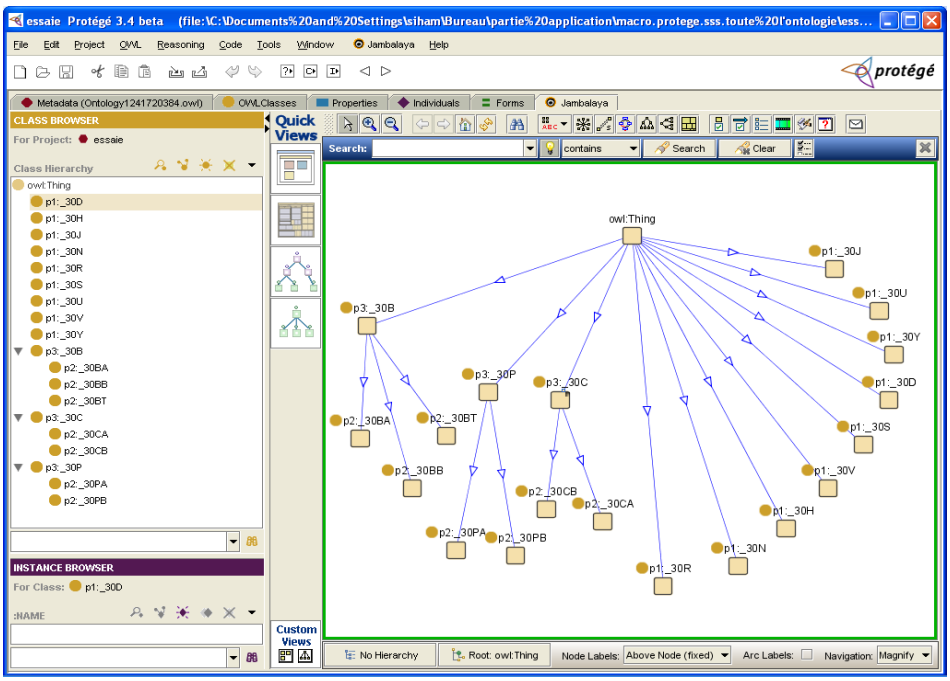


Figure VI.31: le résultat de la fusion sous protege.

La vérification de la consistance de l'ontologie de la fusion par le raisonneur Pellet 1.5.1 est donnée par la figure suivante:

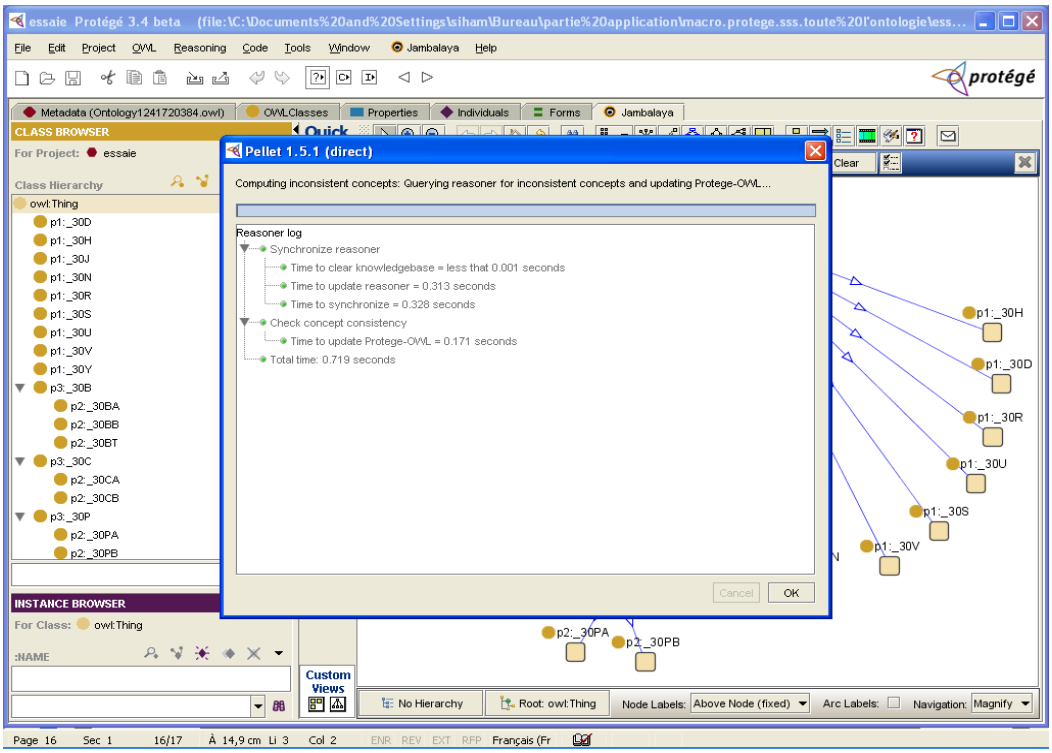


Figure VI.32: Vérification de consistance de l'ontologie par le raisonneur Pellet 1.5.1

Alors que la vérification de la consistance d'hiérarchie de concepts, toujours par le raisonneur Pellet 1.5.1 est donnée par la figure suivante:

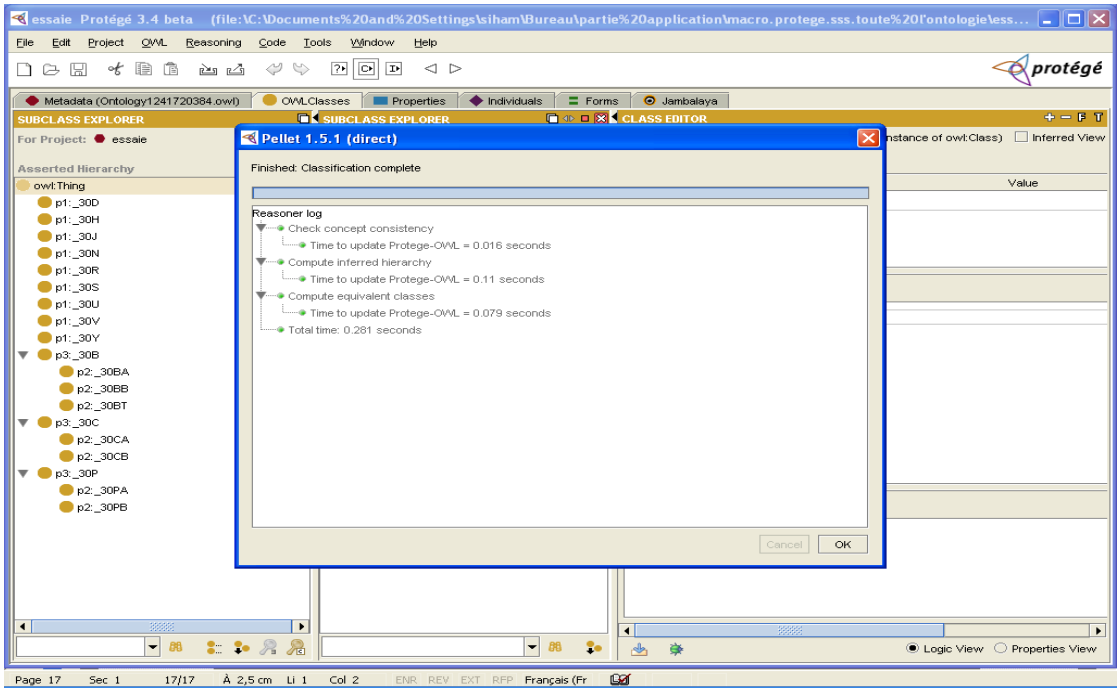


Figure VI.33: Vérification de la cohérence de la hiérarchie de classes par le raisonneur Pellet 1.5.1.

En résumé, les résultats obtenus lors des deux parties de notre approche, (la première concernant la fusion des deux ontologies en utilisant l'algorithme PROMPT, et la deuxième concernant la fusion des ontologies en utilisant l'algorithme de la fusion qu'on a proposé) se rencontrent au point qui reflète la structure formelle de l'ontologie résultat. Comme PROMPT, notre algorithme résulte une ontologie de fusion qui préserve la structure formelle d'une ontologie présentée en langage OWL, où le premier bloc structurel représente la séquence de tous les concepts (ou composants) issus des deux ontologies sources tout en évitant toute possibilité de répétition ou de redondance de concepts. Ensuite, le deuxième bloc représente l'ensemble de tous les attributs ou propriétés décrivant les concepts des deux ontologies source. Enfin, le dernier bloc décrit la séquence de toutes les instances ou individus des concepts issus des deux ontologies initiales tout en évaluant les attributs par les valeurs correspondant à chaque instance.

Cependant, et contrairement aux résultats obtenus par PROMPT qui est un plugin de l'éditeur d'ontologies Protege, les résultats de notre algorithme sont obtenus indépendamment de tout environnement de développement ontologique. En plus, notre algorithme est plus simple à exploiter et plus facile à utiliser car il ne demande aucune intervention humaine vue que l'identification des concepts qui vont être fusionnés en un seul concept se fait automatiquement par comparaison de leur identification (le code du concept), ce qui n'est pas le cas pour PROMPT où l'intervention de l'utilisateur est inévitable pour la spécification des concepts qui doivent être fusionnés.

Conclusion et perspectives

Les ontologies deviennent, de plus en plus, des modèles de représentation et de stockage d'informations très efficaces facilitant le traitement et la gestion des connaissances à travers les techniques de l'IA. Elles offrent le potentiel d'assemblage d'une grande quantité d'informations appartenant à différentes sources d'informations ou de données à travers ce qu'on appelle la Fusion d'Ontologies. Préalablement, chaque source de données (base de données) peut faire l'objet d'une construction ontologique.

Pendant la fusion, des problèmes d'hétérogénéité et de contradiction entre les concepts peuvent apparaître d'où la nécessité d'automatiser le processus de fusion à travers un algorithme de fusion d'ontologies.

La contribution répertoriée dans ce mémoire consiste à partir de deux BDDs décrivant une Turbine à Vapeur selon deux différents contextes, l'un topologique décrivant les caractéristiques topologiques de chaque composant, tels que zone, famille, fonction, etc, et l'autre diagnostique décrivant les cas d'intervention sur chaque composant pour maintenance ou réparation, tout en spécifiant le défaut, son cause, ses symptômes et ses remèdes. Donc, à partir de ces deux BDDs, nous avons construit automatiquement à travers l'outil DataMaster, deux ontologies séparées que nous avons utilisées par la suite à travers un outil de fusion pour obtenir à la fin une seule ontologie plus large et plus complète couvrant un domaine d'application plus vaste.

L'importance du domaine d'application est la nécessité d'une prise en charge efficace, la facilité de stockage, et de traitement ainsi que la gestion de connaissances, ce qui permet d'acquérir facilement et d'une manière efficace les réponses des techniciens et des opérateurs à des requêtes de différents niveaux et dans différents buts de maintenance, d'exploitation et d'adaptation.

La partie de la contribution visant à fusionner les deux ontologies ainsi construites est réalisée de deux différentes manières. Dans la première, nous avons fait la fusion à travers le plugin (sous Protege) semi automatique, PROMPT qui demande une intervention humaine de l'utilisateur pour identifier les concepts similaires qui doivent être fusionnés. Ceci impose alors certaines difficultés délicates : D'abord, on

doit toujours avoir l'environnement de développement ontologique, Protege, pour pouvoir fusionner les ontologies en question. En plus, ces deux ontologies sont assez volumineuses (plus de 2000 composants), ce qui rend difficile l'application de PROMPT, où l'utilisateur doit intervenir pour étudier ce gros nombre de composants afin de pouvoir identifier les concepts similaires qui vont être fusionnés ensemble.

Alors, pour contourner ces difficultés, nous avons créé un algorithme de fusion purement automatique et indépendant de tout environnement de développement, où l'identification de similarités entre les concepts qui doivent être fusionnés est totalement automatique, ainsi aucune intervention humaine ni requise. Cet algorithme de fusion a donné de bons résultats tout en préservant la structure formelle d'une ontologie représentée sous le langage de représentation OWL tel que après l'entête d'axiomes et de faits représentant les méta données décrivant l'ontologie (voir chapitre 2), le premier bloc structurel représente la séquence de tous les concepts issus des deux ontologies sources tout en évitant toutes possibilités de redondances. Ce bloc est suivi par un autre décrivant l'ensemble de toutes les propriétés attribuées à ces concepts. Enfin, on a le bloc énumérant les instances ou les individus des concepts où les attributs de chaque individu sont évalués par leurs valeurs correspondantes.

En perspective, et vu qu'il s'agit d'une voie de recherche récemment entamée, ce domaine reste toujours un domaine ouvert et ambivalent où plusieurs perspectives de recherche sont imposées, et parmi celle qui complètent notre contribution nous proposons :

- Etendre PROMPT, pur qu'il prend en compte l'étape de l'identification de similarités rendant ainsi le test sur les concepts à fusionner totalement automatique et ne demandant aucune intervention humaine.
- L'extension de PROMPT, ainsi que notre algorithme proposé, pour qu'ils fassent aussi la fusion d'attributs et d'instances.
- L'intégration de l'ontologie résultat de la fusion dans un Système Expert ou un SBC, tout en effectuant un enrichissement sémantique et/ou syntaxique de l'ontologie de la fusion (et/ou ontologies sources) par la création de nouvelles

relations et/ou constructeurs owl entre les différents concepts et / instances d’ontologies.

Bibliographie

[ABD et al, 04]: B. Abdulahad, G. Lounis *"A user interface for the ontology merging tool SAMBO",* PHD Thesis, Departement of Computer and Information Science, Université de Linkopngs, 2004.

[AMR et al, 09]: S. Amrouch, M.T. Khadir, *"Fusion d'ontologies de domaine pour la gestion de connaissances d'une turbine à vapeur»,* Seconde Conférence International En Informatique Appliqué, (ICAI 2009), Bordj Bouariridj, Algérie, 2009.

[AUB 07]: S. AUBRY, *«annotations et gestion des connaissances en environnement virtuel collaboratif»,* thèse de doctorat, université de technologies de compiegne, 2007.

[BAC 00]: B. Bachimont, *"Engagement sémantique et engagement Ontologique: conception et réalisation d'ontologies en ingénierie de connaissances".* Eyrolles, Paris, 2000.

[BAC 01]: B.Bachimont, *"Modélisation linguistique et modélisation logique des ontologies: l'apport de l'ontologie formelle".*12ème Journées Francophones d'Ingénierie des Connaissances (IC'01), pages 349-268, Grenoble, 2001.

[BAK 07]: M. Bakillah, *«Ontologies et similarité sémantique», du cours – « Développement d'une approche géosémantique intégrée pour ajuster les résultats des requêtes spatiotemporelles dans les bases de données géospatiales multidimensionnelles évolutives »*, université de LAVA, Québec, Canada, 2007.

[BEL et al, 04]: L. Bellatreche, , G. Pierra, D. Nguyen-Xuan, D. Hondjack, Y. Ait Ameur, *«A Priori Approach for Automatic Integration of Heterogeneous and Autonomous Databases".* In Proceedings of the 15th International Conference on Database and Expert Systems Applications (DEXA 04), pages 475–485, 2004

[BEN et al, 00]: D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, M. Vincini, *"Information Integration: The MOMIS Project*

Demonstration". In Proceedings of 26th International Conference on Very Large Data Bases (VLDB'00), pages 611–614. Morgan Kaufmann, 2000.

[BOR 97]: Borst W. N., *"Construction of Engineering Ontologies for Knowledge Sharing and Reuse"*. PhD thesis, *Tweente University*, Enschede, NL- Centre for Telematica and Information Technology, 1997.

[BRU et al, 06]: J. Bruijn, M. Ehrig, C. Feier, F. Martin-Recuerda, F. charffe, M. Weiten *« Ontology mediation, merging and aligning»*, 2006.

[CAH 05]: L. Casely-Hayford, *"Environments, Methodologies and Languages for supporting Users in building a Chemical Ontology",* PHD Thesis, University of Manchester, 2005.

[CEC 01]: L. CECCARONI, *"ONTOWEDSS, An ontology based environmental decision support system for the management of WASTEWATER treatment plants", Thèse de doctorat en IA,* Université Polythechnique de CATALUNYA, Barcelon, Espagne, 2001

[CHA et al, 04]: J. Charlet, B. Bachimont, R. Troncy, *« Ontologies pour le Web sémantique»*, Mission de recherche STIM, AP-HP & INSERM ERM 0202, Université Technologique de Compiègne, 2004.

[COL et al, 07]: R. M. Colomb, M. N. Ahmad, *"Merging ontologies requires interlocking institutional worlds"*, School of Information Technology and Electrical Engineering, The University of Queensland, Queensland 4072, Australia, 2007.

[COR et al, 02]: O. Corcho, M. Fernandez-Lopez, A. Gomez-Perez, *" methodologies, tools and languages for building ontologies. where is their meeting point?",* Article Elsevier, doi: 10.1016/S0169-023X(02)00195-7, Faculté d'Informatique, Université polytechniques de Madrid, Espagne, 2002.

[CUE et al, 06]: B. Cuenca, C. Lutz, U. Sattler, A.Y. Turhan, *« Tasks for Ontology Integration and Merging"*, Deliverable TONES-D11, The University of Manchester, 2006.

[DIE et al, 01]: R. Dieng, O. Corby, F. Gandon, A. Giboin, J. Golebiowska, N. Matta, et M. Ribière, *"Méthodes et outils pour la gestion des connaissances: une approche pluridisciplinaire du knowledge management"*. Dunod Edition Informatiques, Séries Systèmes d'Information, 2001.

[DOU et al, 02]: D. Dou, D. McDermott, P. Qi, *"Ontology Translation by Ontology Merging and Automated Reasoning »,* In Proceedings of EKAW 2002 Workshop on Ontologies for Multi-Agent Systems (OMAS 2002). pp. 3-18, Yale University, USA, 2002.

[FER et al. 97]: M. Fernandez, A. Gomez-Perez, N. Juristo, *"METHONTOLOGY"*. Proceedings of the AAAI97, Spring Symposium Series on Ontological Engineering, Stanford, USA, pp. 33-40, 1997.

[FUR02]: F. Furst, *"L'ingénierie ontologique"*. Rapport de recherche N°02-07. Institut de Recherche en Informatique de Nantes, 2002.

[FUR 04]: F. Furst *« Contribution à l'ingénierie des ontologies: une méthode et un outil d'opérationalisation »*, Thèse de doctorat, École Polytechnique de l'Université de Nantes (EPUN), 2004

[GAË 02]: L. Gaëlle *« État de l'art Ontologies et Intégration/Fusion d'ontologies »*, rapport de stage, Laboratoire Dialogue et Intermédiations Intelligentes de la Direction des Interactions Humaines DIH/D2I, centre de Recherche et Développement de France Télécom, Lannion, 15 avril -13 septembre 2002.

[GAN 02]: F. Gandon, *"Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web"*. PhD Thesis, INRIA and University of Nice - Sophia Antipolis, 2002.

[GOM 99]: A. Gomez-Perez, *"Ontological Engineering: a State of the Art"*, Expert Update. British Computer Society. Vol. 2. nº 3. (1999), pp. 33 – 43. 1999.

[GRU 93]: T. Gruber, *"A Translation Approach to Portable Ontology Specifications". Knwledge Acquisition*, Vol. 5 (No. 2) pp. 199-220, 1993.

[GRÜ et al, 96]: M. Grüninger, M. Uschold. *"Ontologies: Principles, Methods and Applications"*, Knowledge Engineering Review, 1(2), pp. 93--155. 1996.

[GUA 95]: N.Guarino, "*Formal Ontology, Conceptual Analysis and Knowledge Representation"*. International journal of Human and Computer Studies, Vol. 43 (No. 5/6), pp. 625-640,1995.

[HEM 05]: M. HEMAM, « *Un processus de développement d'ontologies dans le cadre du web sémantique »*, Mémoire de magister, Institut des sciences exactes, Centre Universitaire Larbi Ben M'hidi -Oum El Bouaghi, Algérie, 2005.

[IVA et al, 03]: A.IVANOV, C.VARNIER, N. ZERHOUNI *«Ordonnancement des activités de maintenance dans un contexte distribué »*. 4e Conférence Francophone de MOdélisation et SIMulation "Organisation et Conduite d'Activités dans l'Industrie et les Services" MOSIM'03 – Toulouse (France), 2003.

[IZZ 06]: *S. IZZA,* «*intégration des systèmes d'information industriels, Une approche flexible basée sur les services sémantiques »,* thèse de doctorat, l'Ecole Nationale Supérieure des Mines de Saint-Etienne, Saint-Etienne, 2006.

[JEA, 07]: S. Jean, *« OntoQl, un langage d'exploitation des bases de données à base ontologique »*, Thèse de Doctorat, ENSMA: Ecole Nationale Supérieure de Mécanique et d'Aérotechnique, 2007.

[KHA 07]: M.T. KHADIR, *«Les ontologies concepts, méthodes et outils»***,** Support de cours, Université de Annaba, Algérie, 2007.

[KLE, 01]: M. Klein, *"Combining and relating ontologies: an analysis of problems and solutions",* Vrije Universiteit Amsterdam, 2001.

[KOT et al, 05]: K. Kotis, A. G. Vouros, K. Stergiou, *"Towards Automatic Merging of Domain Ontologies: The HCONE-merge approach"*, Department of Information and Communications Systems Engineering", University of the Aegean, Karlovassi, Samos, 83200, Greece, 2005.

[LAA et al, 07]: F. Z. Laallam , M. Sellami, *« Modélisation et gestion de la maintenance dans les systèmes de production »*, thèse de doctorat, université badji mokhtar annaba, Algérie, 2007.

[LAM et al,03]: P. LAMBRIX, A. EDBERG, *«Evaluation of ontology merging tools in bioinformatics »*, Department of Computer and Information Science, Linköpings universitet, 581 83 Linköping, Sweden, 2003.

[MAE et al, 02]: A. Maedche, S. Staab, *"Measuring Similarity between Ontologies"*. In Proceeding of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW- 02, 2002.

[McG et al, 00]: D. L. McGuinness, R. Fikes, J. Rice, S. Wilder. *"An Environment for Merging and Testing Large Ontologies".* Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning. Breckenridge, Colorado, 2000.

[MER et al, 05]: A. MERGHNI, N. MESBAHI, F.Z. LAALLAM. *« Conception et réalisation d'un sous système de GMAO pour la direction de la maintenance SONATRACH »* Mémoire d'ingéniorat d'état, université de Ouergla, Algerie, 2005.

[MIZ 03]: R. Mizuguchi, *"Tutorial on ontological engineering – Part1: Introduction to Ontological Engineering"*, New Generation Computing, Springer, Vol.21, No.4 (2003), pp.365-384, 2003.

[MOH et al, 05]: M. Mohsenzadeh, F. Shams, M. Teshnehlab, *« A New Approach for Merging Ontologies"*, Proceedings Of World Academy Of Science, Engineering And Technology Volume 4, ISSN 1307-6884, 2005.

[MOS 01]: S. Mostefai, *« Fusion d'ontologies dans une optique PLM »*, Labo LIRE, Université Mentouri, Constantine 25000, Algérie, Nom de la revue. Volume X – n° X/2001, pages 1 à X, 2001.

[NOY et al, 00]: N. F. Noy, A. M. Musen, *"PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment"*, Stanford Medical Informatics, Stanford University, Stanford, CA 94305-5479, 2000.

[NEE et al, 91]: R. Neeches, T. Finin T. Gruber, T. Senator, W. Swartout, *"Enabling Technology for Knowledge Sharing"*, AI Magazine, 12, pp. 35 – 56, 1991.

[NOY et al, 03]: N. F.Noy, M. A.Musen, *"The Prompt suite: Interactive tools for ontology merging and mapping"*. International Journal of Human Computer Studies, 59(6): 983-1024, 2003.

[NYU et al, 07]: C. Nyulas, , M. O'Connor, T. Samson, *"DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé"*, Stanford Medical Informatics, Stanford University School of Medicine, Stanford,CA 94305, 2007.

[OZD 04]: P. ÖZDEN, *"An Ontology of Socio-Cultural Time Expressions",* mémoire de magister, Institut d'informatique, Lehr- und Forschungseinheit für Programmier- und Modellierungssprachen Oettingenstrabe 67 D-80538 München, 2004.

[PIN et al, 99]: H. S. Pinto, A. Gomez-Perez, J. P.Martins, *"Some Issues on Ontology Integration",* Proceedings of the IJCAI-99 workshop on ontologies and problem solving methods (KRR5), Stockholm, Sweden, 1999.

[PRE et al, 05]: L. Predoiu, C. Feier, F. Scharffe, J. Bruijn , F. Martin-Recuerda, D. Manov, M. Ehrig, *"D4.2.2 State-of-the-art survey on Ontology Merging and Aligning V2"*, EU-IST Integrated Project (IP) IST-2003-506826 SEKT, Digital Enterprise Research Institute, University of Innsbruck, 2005.

[RAS et al, 08]: A. D. C. Radgado, A Guezman-Arenas, *"A language and algorithm for automatic merging of ontologies",* Chapter XXII, book "Ontologies for Business Interaction", Instituto Politécnico National, Mexico, ISBN: 978-1-59904-660-0, 452 pages, copyright 2008.

[SHN 05]: A. Shaban-Nejad, *"Design and Development of an Integrated Formal Ontology for Fungal Genomics",* Master thesis, Department of Computer Science and Software Engineering, Université Concordia, Montreal, Quebec, Canada, 2005.

[SOW 95]: J. Sowa, "*Distinction, combination and constraints"*, proceedings of IJCAI-95, Workshop on Basic Ontological Issues in Knowledge Sharing, Stockholm, Sweden, 1995.

[STU 98]: R. Studer, V. Benjamins, D. Fensel. *"Knowledge Engineering: Principles and Methods"*. In Data and Knowledge Engineering, 1998.

[STU et al, 01]: G. Stumme, A. Maedche, *"FCA-MERGE: Bottom-Up Merging of Ontologies"*, Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany, 2001

[SUX 04]: X. Su, *"Semantic Enrichment for Ontology Mapping"*, PHD Thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, N-7491Trondheim, Norway, ISBN 82-471-6453-1, ISSN 1503-8181, 2004.

[SWA et al. 97]: B.Swartout, R.Patil, K.Knight, T.Russ, *"Use of Large Scale Ontologies"*. Spring Symposium Series on Ontological Engineering. Stanford University, CA, pp. 138-148, 1997.

[TAL et al, 03]: A.TALBI, A. HAMMOUCHE, C. TAHON. *« Méthode intégrée de planification des tâches de production et de maintenance »*. Revue Française de Gestion Industrielle Vol.22, N°3, P39-59, 2003.

[VIZ et al, 07]: A. Vizcaíno, N. Anquetil, K. Oliveira, F. Ruiz, M. Piattini, *"Merging Software Maintenance Ontologies: Our Experience"*, University of Castilla-La Mancha, Spain, 2007.

Résumé

Les connaissances initiales dans un domaine donné se développent au fur et à mesure des acquis de l'environnement à partir de différentes sources d'informations.

Pendant la fusion, des problèmes d'hétérogénéité et de contradiction entre les concepts peuvent apparaître, d'où la nécessité d'automatiser le processus de fusion à travers un algorithme de fusion d'ontologies qu'on va appliquer sur des ontologies (deux ou plus) représentant une turbine à vapeur.

L'importance du domaine d'application est la nécessité d'une prise en charge efficace facilitant le stockage, le traitement ainsi que la gestion de connaissances, permettra ainsi d'acquérir facilement et d'une manière efficace les réponses des techniciens et des opérateurs à des requêtes de différents niveaux et dans différents buts de maintenance, d'exploitation et d'adaptation.

Notre but, est d'appliquer un algorithme de fusion sur deux ontologies sources, issues du même domaine de la maintenance industrielle et décrivant la même Turbine à Vapeur exploitée au sein de l'entreprise de SONELGAZ pour la production de l'électricité, mais selon deux différents contextes, l'un Topologique et l'autre de Diagnostique. Cet algorithme résulte donc une seule ontologie plus grande et plus complète couvrant un domaine d'application plus large. D'abord, nous avons fusionné les deux ontologies en appliquant le plugin de fusion sous Protege, PROMPT, qui est un algorithme semi automatique guidant l'utilisateur à fusionner les ontologies sources.

Notre contribution a été de créer un outil automatique de fusion d'ontologies, indépendant de tout environnement de construction d'ontologies et ne demandant aucune intervention humaine.

Mots Clés:

Fusion d'ontologies, Turbine à vapeur, DataMaster, Protege 2000, OWL, PROMPT.