*N° d'ordre :*

REPUBLIQUE ALGERIENNE DEMOCRATIQUE & POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR & DE LA RECHERCHE SCIENTIFIQUE


UNIVERSITE DJILLALI LIABES
FACULTE DES SCIENCES EXACTES
SIDI BEL ABBES


# THESE DE DOCTORAT

*Présentée par*


Mlle. SETTI AHMED SORAYA


*Spécialité : Informatique*
*Option : Intelligence Artificielle*

*Intitulée*

# *Construction d'Ontologies Modulaires : Approche Basée sur le Clustering*

*Soutenue le 14/01/2016*
*Devant le jury composé de :*

*Pésident : M.RAHMOUN ABDELATIF, Professeur, UDL-Sidi Bel Abbès*
*Examinateurs : Mme. HAMDADOU Djamila, MCA, Université d'Oran 1*
*M.HAMOU Réda Mohammed, MCA, Université de Saida*
*M.BOUCHIHA Djelloul, MCA, Centre Universitaire de Naâma*
*M.BENSLIMANE Sidi Mohammed, Professeur, UDL-Sidi Bel Abbès*
*Directeur de thèse : M.MALKI Mimoun, Professeur, UDL-Sidi Bel Abbès*

*Année universitaire 2015/2016*

# Dédicace ...

*Toutes les lettres ne sauraient trouver les mots qu'il faut...*

*Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance...*

*A tous ceux qui me sont très chers je dédie ce modeste travail*
*Au doux souvenir vivant à jamais dans mon cœur, mon très cher regretté père qui m'a enseigné que l'amour, l'indulgence et la patience sont bien les clés du bonheur et de la réussite*
*« Ceux qui sont morts ne sont jamais partis».*

*A ma très chère mère*
*A mes très chères sœurs*
*A mes neveux Mohamed Taha, et ma très jolie nièce Fatima El Zahra, et en particulier, mon petit adorable ange Abdelkader.*

**Soraya**

# Remerciements

***La connaissance est en fin de compte fondée sur la reconnaissance.***
***Ludwig Wittgenstein***

Préparer une thèse est effectivement une tâche de longue haleine

L'aventure de la thèse s'achève. Au-delà de la satisfaction du travail réalisé, je garderai un excellent souvenir des années passées. Je voudrais ici exprimer ma reconnaissance envers les personnes qui m'ont aidé et soutenu durant cette période.

Tout d'abord louange à *Allah* seigneur de l'univers, de m'avoir donné la force, la patience et la volonté pour accomplir ce travail.

Mes plus sincères remerciements vont en premier lieu à Professeur. Malki Mimoun de l'université Djillali Liabes de Sidi Bel Abbès, mon directeur de thèse, non parce que c'est l'usage, mais pour m'avoir guidé dans mes travaux, pour son encadrement, sa patience pendant toutes ces années et ses nombreux conseils. La confiance que vous m'avez accordée m'a permis de m'épanouir en tant que jeune chercheuse et de mener à bon port cette thèse. Je vous remercie également pour votre soutien pendant les périodes les plus difficiles. Vous m'avez dit un jour : « Il faut partir doucement et surement ». Et oui et comme dit Mencius "Ceux qui s'avancent trop précipitamment reculeront encore plus vite" ! Vous m'avez communiqué j'espère une partie de votre expérience utile et de votre patience. Encore une fois, merci pour tout !

Un merci tout particulier à Professeur.Benslimane Sidi Mohamed de l'université Djillali Liabes de Sidi Bel Abbès et directeur de l'école ESI de Sidi Bel Abbes qui m'a aidé dans mes travaux de mémoire, pour m'avoir conseillé avec professionnalisme et une très grande expertise, sans jamais compter son temps ni perdre sa bonne humeur. J'ai beaucoup apprécie ses qualités intellectuelles et humaines. Grace à lui j'ai beaucoup appris sur la recherche. Quand nos travaux ont été remis en question vous m'avez appris à défendre mes idées avec passion mais aussi à chercher sans cesse à les améliorer.

Mes remerciements chaleureux s'adressent également aux personnes qui m'ont fait l'honneur d'accepter de siéger dans le jury de cette thèse : Professeur Rahmoun Abdelatif de l'université Djillali liabes UDL- de Sidi Bel Abbes, pour m'avoir fait l'honneur de présider le jury de cette thèse. Madame HAMDADOU Djamila , maître de conférences à l'université Oran 1, Monsieur HAMOU Réda Mohammed, maitre de conférences à l'université de Saida, Monsieur BOUCHIHA Djelloul, maître de conférences au centre universitaire de Naâma, pour l'intérêt qu'ils ont porté à ce travail en

acceptant de faire partie du jury.

Un grand merci à tous ceux qui m'ont soutenue et encouragée tout au long de ces années de thèse. J'ai une pensée spéciale pour notre très cher Prof. Ismaïl Khriss de l'UQAR (University of Quebec at Rimouski), au Canada de m'avoir convaincue de relever ce défi et pour tous vos beaux messages inhérents d'encouragements que j'avais reçus. Malgré votre temps trop chargé et la distance mais vous savez pu me consacrer un peu plus de temps par votre lecture et vos remarques judicieuses. Vous m'avez appris une partie de votre sagesse et autre de votre expérience. Vous êtes une grande école.

Je voudrais spécialement remercier la communauté des développeurs qui participent au partage libre des connaissances sur Internet. Bravo !!

Un merci particulier au docteur Mr.Dennai Abdessalam pour m'avoir donné l'autorisation d'utiliser sa mesure de calcul de similarité.

Enfin et bien évidemment, je remercie mes parents et ma famille, qui ont toujours cru en moi et qui m'ont toujours soutenue. Pendant toutes ces dernières années vous avez su être toujours près de moi pour m'encourager dans les moments difficiles et pour fêter mes réussites. Merci maman source d'amour et d'affection. Merci papa, sens de responsabilité, malgré votre mort et votre absence insurmontable, sans toi je n'aurais jamais pu goûter aux joies de la science et de l'informatique. Tu m'as toujours encouragé.
Vous avez été vous deux mes modèles pendant toutes ces années et je fais tout pour devenir quelqu'un d'aussi bien que vous.

# Résumé

Le web sémantique a pour but le partage et l'intégration des données issues de domaines et d'organisations différents.
Les ontologies présentent un savoir préalable requis pour un traitement systématique de l'information à des fins de navigation, de rappel, de précision, En outre les ontologies sont considérées comme étant des structures efficaces pour la représentation des connaissances du domaine.
Le champ d'application des ontologies ne cesse de s'élargir et couvre les systèmes conseillers (systèmes d'aide à la décision, systèmes d'enseignement assisté par ordinateur – e-Learning –, etc.), les systèmes de résolution de problèmes et les systèmes de gestion de connaissances (par exemple dans le domaine du biomédical). Un des plus grands projets basé sur l'utilisation des ontologies consiste à ajouter au Web une véritable couche de connaissances permettant, dans un premier temps, la recherche d'information aussi bien au niveau syntaxique qu'au niveau sémantique.

Et comme l'usage du web sémantique est en pleine expansion le besoin des ontologies est devenu incontournable. Toutefois, les ontologies sont confrontées de façon continue à un problème d'évolution et de complexité. Donc plusieurs opérations du web sémantique comme la réponse aux requêtes, le partage, l'intégration des données l'alignement deviennent des tâches plus difficiles. Avec la complexité des ontologies et la taille plus grande. Partitionner l'ontologie en plusieurs modules est une solution à ces problèmes.
Dans cette thèse nous proposons une approche de partitionnement de l'ontologie. C'est une amélioration et révision de l'approche de clustering K-means bâtie sur une nouvelle mesure de similarité pour partitionner une ontologie à plusieurs partitions.
Les résultats obtenus montrent que notre approche génère les meilleurs résultats en comparant avec celle de K-means standard.

**Mots clés** : Ontologie, algorithme de partitionnement, modularisation, ontologies OWL, approche K-Means, mesures de similarité

# Abstract

The semantic web goal is to share and integrate data across different domains and organizations. The knowledge representations of semantic data are made possible by ontology.
The scope of ontologies is constantly expanding and covers advisors systems (support systems in making decision, computer-assisted learning systems - e-Learning -, etc.), problem-solving systems and knowledge management systems (eg in the field of biomedical). One of the biggest projects based on the use of ontologies is to add to the Web true layer of knowledge to, firstly, the search for information as well syntactically at the semantic level.
As the usage of semantic web increases, construction of the semantic web ontologies is also increased. Moreover, due to the monolithic nature of the ontology various semantic web operations like query answering, data sharing, data matching, data reuse and data integration become more complicated as the size of ontology increases. Partitioning the ontology is the key solution to handle this scalability issue. In this work, we propose a revision and an enhancement of K-means clustering algorithm based on a new semantic similarity measure for partitioning given ontology into high quality partitions. The results show that our approach produces meaningful clusters than the traditional algorithm of K-means.

**Keywords**— Ontology, Partition Algorithm, Modularization, Ontology Owl, K-Means Clustering Algorithm, Similarity Measures

# Table des matières

## LISTE DES FIGURES

# Chapitre 2. Généralités sur les ontologies



# Chapitre 3. Ingénierie ontologique



.

# Chapitre 4. Modularisation des ontologies

**Etat de l'art**



# Chapitre 5. Approche proposée



# Chapitre 6. Implémentation

# LISTE DES TABLEAUX

## Chapitre 3



## Chapitre 4



## Chapitre 6

*La solitude est utile. Il faut parfois ne parler qu'avec soi-même. On entend alors de dures vérités ou d'agréables mensonges selon qu'on s'analyse ou qu'on s'imagine."*

Henri de Régnier

# Chapitre 1

# Introduction générale

« *La méthode de la science est une méthode de conjectures audacieuses et de tentatives ingénieuses et sévères pour réfuter celles-ci.* »

Karl Popper, La Connaissance Objective, 1972.

## 1.1 Contexte

En représentation des connaissances, la définition de l'ontologie c'est-à-dire la description formelle d'entités et de leurs propriétés, relations, contraintes et comportement dans un domaine particulier, offre une base solide au développement de nouvelles méthodes d'application de raisonnement sur les connaissances. Outre son apport en matière de réutilisabilité, de modularité et de partage de connaissances, l'ontologie permet de définir un vocabulaire précis, sur lequel est basée la communication entre les différents acteurs d'un projet. Elle est également utilisée pour la description en langage naturel et l'annotation de documents. Enfin, l'ontologie permet de simplifier l'étape d'analyse et de synthèse d'une partie de la connaissance dans le développement de systèmes et diminuent de ce fait le coût de leur conception. La réutilisabilité est une caractéristique d'autant plus importante des ontologies.

Le champ d'application des ontologies ne cesse de s'élargir et couvre les systèmes conseillers (systèmes d'aide à la décision, systèmes d'enseignement assisté par ordinateur – *e-Learning* –, etc.), les systèmes de résolution de problèmes et les systèmes de gestion de connaissances (par exemple dans le domaine du biomédical). Un des plus grands projets basé sur l'utilisation des ontologies consiste à ajouter au Web une véritable couche de connaissances permettant, dans un premier temps, la recherche d'information aussi bien au niveau syntaxique qu'au niveau sémantique.

L'enjeu de l'effort engagé est de rendre les machines suffisamment sophistiquées pour qu'elles puissent intégrer le sens des informations, qu'a l'heure actuelle, elles ne font que manipuler formellement. Mais en attendant que des ordinateurs «chargés» d'ontologies et de connaissances nous soulagent en partie du travail de plus en plus lourd de gestion des informations dont le flot a tendance à nous

submerger, de nombreux problèmes théoriques et pratiques restent a résoudre.

En outre le web sémantique est en pleine expansion, des ontologies de domaine se développent à toute vitesse surtout dans le cadre de la recherche. Même si le web sémantique est assez jeune et immature, c'est un outil puissant permettant aux agents informatiques de traiter l'information par un langage qui leur est compréhensible, OWL. Partout dans le Web, les ontologies se développent. Il faut donc créer des outils puissants permettant d'intégrer toutes ces informations. Il existe plusieurs types d'intégrations qui vont de la simple extension à la fusion d'ontologies. Les méthodes d'intégration ne sont pas uniques et sont principalement le fruit de recherches isolées. Maintenant il s'agit pour la communauté du web sémantique de décider d'une méthode d'intégration.

Les ontologies restent aujourd'hui au stade de la recherche. Elles ont un grand intérêt pour faciliter le travail par exemple des biologistes. Mais, dans le monde des affaires, le traitement efficace de l'information et des données elles deviennent une nécessité car elles confèrent un avantage concurrentiel durable par rapport aux concurrents. En effet, aujourd'hui toutes les entreprises cherchent à améliorer leurs systèmes d'information pour plus de flexibilité [BENOIT, 05].

Le champ d'application des ontologies est très vaste ; elles sont utilisées dans les Systèmes à base de connaissance (SBC), où elles peuvent servir de ressources de base pour le raisonnement, de spécification permettant le contrôle de la sémantique d'une base de connaissances ou de conceptualisation commune d'un domaine assurant la communication entre deux agents logiciels ou entre un système et un utilisateur, voire entre deux utilisateurs.

Une des applications phare des ontologies est le Web sémantique, le Web constitue un terrain idéal de mise en œuvre des ontologies considérées en tant que spécifications partagées de connaissances, les pages Web représentant une énorme masse de connaissances difficilement exploitable sans une assistance automatique. Cette masse augmente sans cesse ainsi que le nombre d'utilisateurs qui veulent pouvoir trouver facilement les informations qu'ils y recherchent.

L'introduction des ontologies dans le Web, consiste à outiller le Web pour lui adjoindre une couche de connaissances accroissant son efficacité et la puissance des applications qui s'y déploient.

Le but principal de l'utilisation des ontologies est d'avoir un moyen pour représenter les connaissances, sous une forme interprétable et utilisable par un ordinateur, en permettant leur partage et leur réutilisation. Dans cette optique de partage et d'interopérabilité, la prolifération et la disponibilité des ontologies sont cruciales. Il va sans dire que les ontologies sont devenues un modèle incontournable pour la représentation et le raisonnement sur des connaissances du domaine. Bien qu'essentielles à la gestion des systèmes à base de connaissances.

## 1.2 Problématique

Les ontologies doivent donc leur succès actuel à leur aptitude à être partageables et réutilisables. Quoiqu'avec le partage, on se retrouve avec des ontologies de plus en plus larges. La construction d'une ontologie à partir de zéro est une tâche complexe qui nécessite du temps. Ainsi, dans le but de simplifier le processus de conception, on tablera sur le principe que l'on puisse construire une ontologie à partir d'une autre déjà existante. Ceci nous amène donc à envisager la modularisation comme étant une approche qui apporterait des

solutions non seulement au problème de conception, mais aussi à celui du partage et de l'intégration des ontologies.

Une solution possible serait de modulariser les ontologies.

Dans l'ingénierie ontologique, la modularisation pourrait être perçue de deux façons. Premièrement, elle pourrait être vue comme un processus menant à la décomposition d'une ontologie large en des modules ontologiques de petite taille. Deuxièmement, elle pourrait également être perçue comme une étape du processus de construction de l'ontologie qui se réalise à travers la conception d'un ensemble de modules ontologiques indépendants les uns des autres. Modulariser revient donc à sélectionner les concepts et relations pertinents en fonction de la tâche et de l'application pour lesquelles on décide de modéliser une ontologie [PATRICK, 14]. Pour ce faire, on peut soit :

• Réutiliser toute l'ontologie en entier. Tous les concepts et relations de l'ontologie du domaine seront inclus dans l'ontologie qu'on tend à concevoir. Ceci entraîne un alourdissement de l'ontologie de l'application qui sera d'autant plus grande si l'ontologie du domaine est large. On se retrouvera dans ce cas avec des concepts et définitions qui ne sont pas forcément pertinents pour la tâche à réaliser.

• Construire une nouvelle ontologie. Ce qui représenterait un travail fastidieux. La conception d'une ontologie demeure une tâche complexe dans la mesure où il revient à l'expert de définir tous les concepts et relations nécessaires.

• Réutiliser une partie de l'ontologie initiale. C'est un compromis entre les deux alternatives précédentes. Dans ce cas, l'ontologie de l'application représenterait donc une partie ou un module de l'ontologie de domaine.

Dans la situation actuelle, on se rend assez vite compte que la modularisation serait aussi une approche efficace qui permettrait d'apporter des réponses à bons nombres de problèmes du génie de

connaissances. Nous aborderons la modularisation dans le sens d'un processus au cours duquel on identifie un ensemble de concepts pertinents et spécifiques à un domaine donné. Le module qui est aussi une ontologie sera uniquement composé de ces concepts. Toute ontologie se doit de satisfaire un certain nombre de critères afin de s'assurer qu'elle décrive de façon précise et complète les connaissances du domaine qu'on entend modéliser.

En d'autres termes, on ne saurait parler de modularisation sans aborder la question de l'évaluation des ontologies. De ce fait, la mise en place d'un ensemble de métriques de validation devient un impératif dès lors qu'il est essentiel d'évaluer la qualité des partitions obtenues.

Parmi les principales approches de modularisation des ontologies on peut citer:

**Les techniques de partitionnement des ontologies** dont on trouve :

- *Stuckenschmidt et Klein* [STUCK et KLEIN, 04] qui consiste à partitionner automatiquement des ontologies basée sur la structure de la hiérarchie des classes (concepts).
- *Cuenca Grau et al.* [GRAU et al., 00] qui proposent une approche basée sur les E-connexions pour résoudre le problème du partitionnement d'une ontologie.

**Les techniques d'extraction de module**

L'extraction basée sur le parcours de graphe. Les techniques d'extraction basées sur une approche structurelle nécessitent la représentation de l'ontologie comme un graphe afin de parcourir celle-ci et d'en extraire le module ontologique [SETTI AHMED et al., 11].

- *L'approche de d'Aquin et a* [D'AQUIN et al., 06] qui considère que la sélection des connaissances» *(Knowledge selection)* est un processus

plus complexe et l'extraction de module ontologique ce n'est qu'une étape de ce processus.

- *L'approche de Doran et al.* elle a comme perspective de pouvoir réutiliser une partie d'une ontologie existante. Elle ne prend pas en considération le type de langage dans lequel l'ontologie est représentée.

**L'extraction basée sur la logique**

En plus des techniques d'extraction basée sur le parcours de graphe, il existe d'autres techniques d'extraction basée sur la logique qui reposent sur la notion d'extension conservatrice *(conservative extension).* Une ontologie est considérée comme une extension conservatrice si les implications logiques à propos du module sont comprises dans sa signature et un module est une sous-structure de cette ontologie dont il est extrait.

On peut citer les travaux de *Lutz et al* [LUTZ et al., 07] et *Grau et al* [GRAU et al., 08]. Nous reviendrons en détail sur ces travaux dans les prochains chapitres.

## 1.3. Objectifs et contributions de la recherche

### 1.3.1 Objectifs

Nos objectifs de recherche à travers ce travail se situent au cœur des recherches actuelles sur les ontologies étant donné qu'un des plus importants aspects des ontologies est la modularisation.

Notre but donc est tout d'abord de proposer une approche de partitionnement des ontologies à base de technique de clustering. Puis, sur la base de cette dernière, il sera question de développer un système dynamique, basé sur des ontologies de domaine et capable

d'extraire des partitions ontologiques.

### 1.3.2 Contributions

Le travail présenté dans ce manuscrit ne prétend pas fournir une solution répondant aux aspects différents du problème de modularisation des ontologies. Toutefois, ce travail a apporté un certain nombre de contributions au champ de recherches sur le partitionnement des ontologies à base de clustering :

- Nous présentons un état de l'art détaillé en termes de modularisation des ontologies et de partitionnement ainsi qu'une classification des différentes approches de modularisation des ontologies selon différents niveaux.
- Nous proposons une amélioration et une révision de l'approche K-Means en fonction des premiers résultats obtenus en appliquant l'algorithme traditionnel standard.
- L'application de l'algorithme standard K-Means génère des partitions dynamiques dont le contenu change après chaque nouvelle exécution, notre proposition apporte une solution à tel problème.
- Les partitions générées de l'ontologie à partir de l'application de l'algorithme classique du K-Means donne dans certains cas des clusters (partitions) vides ce qui n'est pas conforme aux règles de partitionnement.
- Nous proposons dans ce mémoire un algorithme de partitionnement basé sur le calcul des mesures de similarités, pour ce faire nous avons utilisé sept mesures.
- Finalement, nous avons appliqué notre approche proposée sur deux ontologies Owl : Tambis avec 394 concepts et Photography avec plus de 100 concepts. Pour cela nous avons développé un système de partitionnement.

- Ensuite un protocole de validation de la qualité des partitions générées quant à lui repose sur une étude comparative en se basant sur deux métriques d'évaluation à savoir : la cohésion et la densité pour juger de la qualité des différentes partitions générées à l'aide de l'utilisation de différentes mesures de similarité.

## 1.4 Plan de la thèse

Cette thèse est organisée comme suit :

**Chapitre 2 et 3 : Généralités sur les ontologies et l'Ingénierie ontologique**

Cette partie présente une vue d'ensemble sur les concepts de base des ontologies, et se reporte à l'ensemble des activités qui concerne le processus de développement des ontologies, le cycle de vie des ontologies, les méthodes de construction, les outils et les langages qui les supportent.

**Chapitre 4 : Modularisation des Ontologies : Etat de l'art**

Ce chapitre se concentre autour de la présentation du concept de la modularisation des ontologies, les travaux qui ont été menés dans ce domaine et plus précisément, les principaux buts de la modularisation.

Les différentes techniques et outils de modularisation seront présentés tout en mettant l'accent sur les limites actuelles de ceux-ci.

**Chapitre 5 : Approche proposée**

Ce chapitre comprend le modèle et l'approche proposée, les algorithmes développés et l'architecture de notre système proposée avec les différentes étapes détaillées.

**Chapitre 6 : Implémentation et évaluation**

Ce chapitre détaille la mise en application de notre approche. Nous verrons donc le travail qui a été accompli et le système réalisé ainsi que les résultats obtenus.

En plus, nous détaillons l'ensemble des métriques de qualité sur lesquels se base l'évaluation de nos résultats de partitionnement obtenus et mettons en évidence les avantages de notre système développé.

**Chapitre 7 : Conclusion et perspectives**

Cette partie vient clôturer ce mémoire et se veut être comme l'aboutissement, la conséquence des propos précédents. L'exposé des réalisations et des perspectives liées à notre sujet de recherche, en outre, l'occasion de mettre en relation les savoirs acquis lors des chapitres précédents.

Nous discutons également les limites et les points faibles tout en planifiant les perspectives que nous envisageons pour compléter ce travail.

A l'issue de la présente thèse, il est noté que deux annexes viennent d'être rajoutées pour mieux approfondir l'étude.

La première annexe concerne la présentation des différentes mesures de similarités existantes y compris les sept mesures de similarité appliquées dans notre étude. Pour ce qui est de la deuxième annexe nous trouverons en détail les différentes interfaces de notre système développé ontology partitioning clusetring based approach (sysOPCBA).

# Chapitre 2
# Généralités sur les ontologies

*« Même un petit peu de sémantique*
*peut déjà vous emmener très loin »*

## 2.1 Introduction

L'utilisation de connaissances en informatique a pour but de ne plus faire manipuler en aveugle des informations à la machine mais de permettre un dialogue, une coopération entre le système informatique et les utilisateurs. Pour cela, le système doit avoir accès non seulement aux termes utilisés par l'être humain mais également à la sémantique qui leur est associée, afin qu'une communication efficace soit possible. Les ontologies visent à représenter cette connaissance en étant à la fois interprétables par l'homme et par la machine. Ces dernières sont un sujet de recherche populaire dans diverses communautés notamment l'ingénierie des connaissances, la recherche d'information et le traitement du langage naturel, les systèmes d'information coopératifs, l'intégration intelligente d'information et la gestion des connaissances. Elles fournissent une connaissance partagée et commune sur un domaine qui peut être échangée entre des personnes et des systèmes hétérogènes. Elles ont été définies en intelligence artificielle pour faciliter le partage des connaissances et leur réutilisation.

Actuellement, les ontologies constituent un enjeu stratégique dans la représentation et la modélisation des connaissances, récemment, elles ont été introduites pour formaliser les connaissances dans le domaine de modélisation et de simulation intelligente, elles définissent les primitives indispensables pour leur représentation, ainsi que leur sémantique dans un contexte particulier, elle permettent aussi la recherche d'informations, le traitement du langage naturel, l'intégration intelligente d'informations et la gestion des connaissances. Elles fournissent une connaissance partagée et commune sur ce domaine qui peut être échangée entre des personnes et des systèmes hétérogènes.

Dans ce chapitre, nous attachons à décrire les différentes définitions du concept d'ontologie, nous verrons aussi les différents éléments

dont elle est composée et les relations possible entre ces éléments. Nous présenterons ces différents types.

## 2.2 Origine de l'ontologie

La définition explicite et la délimitation précise du concept ontologie soulève un questionnement qui est tout à la fois d'ordre philosophique, épistémologique, cognitif et technique.

**Ontologie** est un concept qui a tout d'abord été défini en Philosophie comme une branche de la Métaphysique qui s'intéresse à l'existence, à l'être en tant qu'être et aux catégories fondamentales de l'existant. En effet, ce terme est construit à partir des racines grecques **ontos** qui veut dire ce qui existe, l'Etre, l'existant, et **logos** qui veut dire l'étude, le discours, d'où sa traduction par *l'étude de l'Etre* et par extension *de l'existence*.[AMARD, 07].
Dans la philosophie classique, l'Ontologie correspond à ce qu'Aristote appelait la Philosophie première (*protè philosopha*), c'est-à-dire *la science de l'être en tant qu'être*, par opposition aux philosophies secondes qui s'intéressaient, elles, à l'étude des manifestations de l'être (les existants). [VALERY, 07].
Bien que ce soient les Grecs qui aient inventé cette science, ils ne l'avaient pas appelé « Ontologie », le terme étant beaucoup plus récent (XVIIe siècle) que la discipline qu'il désigne. La discipline elle-même a évolué vers une voie imprévisible, il y a seulement une trentaine d'années. En effet, d'après [VALERY et al., 03] l'ontologie a été abordée dans le domaine de l'intelligence artificielle pour la première fois par John McCarthy qui reconnut le recoupement entre le travail fait en Ontologie philosophique et l'activité de construire des théories logiques de systèmes d'intelligence artificielle. Il affirmait déjà en 1980 que les concepteurs de systèmes intelligents fondés sur la logique devraient d'abord énumérer tout ce qui existe.

### 2.2.1 Définitions

Ontologie : partie de la métaphysique qui s'attache à l'étude ou à la théorie de l'être dans son essence, indépendamment des phénomènes de son existence. [ENCARTA, 07] [BEN HEBI, 12].

Plusieurs autres définitions du concept ontologie ont été proposées dans le domaine de l'Intelligence Artificielle. Ces définitions sont souvent des raffinements de définitions déjà proposées et/ou sont complémentaires avec elles.

Dans le cadre de l'intelligence artificielle, Neeches et ses collègues [NEECH, 91] furent les premiers à proposer une définition à savoir : *«une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire».*

En 1993, Gruber [GRUBER, 93] propose la définition suivante : *«spécification explicite d'une conceptualisation»* qui est jusqu'à présent la définition la plus citée dans la littérature en intelligence artificielle. Cette définition a été modifiée légèrement par Borst [BORST, 97] comme «*spécification formelle d'une conceptualisation partagée*».

Ces deux définitions sont regroupées dans celle de Studer [STUD,98] comme «*spécification formelle et explicite d'une conceptualisation partagée*».

Les principales caractéristiques des ontologies sont: [KAVEH, 04]

• *Formelles* : elles sont exprimées dans une langue qui a une syntaxe bien définie et compréhensible par la machine de façon à ce qu'elles puissent être traitées automatiquement par les programmes appropriés. Les ontologies permettent le raisonnement logique et supportent des règles d'inférence. Contrairement au modèle relationnel (ER) et UML qui sont des langages semi-formels, les ontologies peuvent être directement traitées par la machine.

• *Lisibles par les humains :* autre que les programmes informatiques, les ontologies peuvent être comprises et développées par les informaticiens.

• *Vastes* : les ontologies couvrent largement les champs de connaissance d'un domaine donné.

• *Partageables* : les ontologies sont relativement faciles à combiner et à fusionner même si elles sont développées séparément. Les ontologies permettent à différents systèmes hétérogènes de partager diverses informations.

Les performances des systèmes basés sur les ontologies dépendent formellement de la qualité de la conception du modèle ontologique. Afin d'améliorer la conception de ces modèles, certains chercheurs ont essayé de mettre en place un processus de développement des ontologies ainsi que certains critères de conception.

Pour le processus [POVEDA et al.,10 ] le développement des ontologies obéit à une méthodologie définissant les étapes et les règles à suivre. Les activités énoncées sont, dans l'ordre : l'étude de faisabilité, 1'acquisition des connaissances, la spécification des exigences de 1'ontologie et finalement la définition de cycle de vie de l'ontologie et les ressources humaines nécessaires

Concernant les critères de conception des ontologies [GRUBER, 95] considère qu'ils dépendent de l'objectif final de leur développement et propose un ensemble de critères pour qui sont comme suit :

- *Clarté:* les termes de l'ontologie doivent être définis de manière claire, objective et complète. C'est-à-dire, que la définition des termes doit être efficacement communiquée par l'ontologie et qu'un prédicat doit être défini par des conditions « nécessaires et suffisantes».
- *Cohérence* : les ontologies doivent inférer des faits qui sont cohérents avec la définition informelle de 1'ontologie. Les classes et les relations du modèle ontologique doivent suivre

une certaine logique.

- *Extensibilité : L'*extensibilité est l'une des caractéristiques majeures des ontologies. Gruber définit l'extensibilité de l'ontologie comme étant le fait *«d'être en mesure de définir de nouveaux termes pour une utilisation spéciale basée sur le vocabulaire d'une ontologie existante sans pour autant avoir recours à la révision des définitions existantes »*
- *Biais d'encodage minimal :* les ontologies sont souvent partagées par des sources hétérogènes et dont les langages de programmation diffèrent d'un système à un autre.

De ce fait, et afin qu'elles soient compréhensibles et traitables par tous les systèmes, les ontologies doivent être développées indépendamment des langages d'implémentation.

- *Engagement ontologique minimal :* l'engagement d'une ontologie doit être minimisé en se contentant juste d'énoncer les principes liés à un domaine donné, et ce en définissant juste les termes essentiels du vocabulaire et en évitant ceux qui sont hors domaine.

Pour Guarino & Giaretta [GUARI, 95] *«Une ontologie est une spécification*
*rendant partiellement compte d'une conceptualisation».* Swartout et ses collègues [SWART, 97] la définissent comme suit : *« une ontologie est un ensemble de termes structurés de façon hiérarchique, conçue afin de décrire un domaine et qui peut servir de charpente à une base de connaissances».*

La même notion est également développée par Gomez [GOMEZ, 99] comme : *« une ontologie fournit les moyens de décrire de façon explicite la conceptualisation des connaissances représentées dans une base de connaissances. »*

Pour conclure, nous pouvons donc affirmer que les définitions du terme ontologie abondent dans la littérature scientifique. Les

définitions, dans leur diversité, offrent des points de vue à la fois différents et complémentaires sur un même concept.

En clair, une **ontologie** fournit les moyens d'exprimer les concepts d'un domaine en les organisant hiérarchiquement et en définissant leurs propriétés sémantiques dans un langage de représentation des connaissances formel favorisant le partage d'une vue consensuelle sur ce domaine entre les applications informatiques qui en font usage [BEN HEBI, 12].

## 2.3 Composantes d'une ontologie

Comme cela a été mentionné, les ontologies fournissent un vocabulaire du domaine et définissent, de façon plus ou moins formelle, le sens des termes et des relations entre ces derniers. Pour cela, les connaissances intégrées dans une ontologie sont formalisées en s'appuyant sur cinq types de composants : les concepts, les relations, les fonctions, les instances et les axiomes [GRUBER, 93] :

- Concept : terme que l'on retrouve également dans la littérature sous le nom de classe, il fait référence à la représentation d'un objet matériel, d'une notion ou d'une idée dans l'ontologie.

Il s'agit, d'abstractions pertinentes d'un fragment du monde réel en fonction d'un domaine d'application. Un concept peut être classé selon plusieurs dimensions : le niveau d'abstraction (concret ou abstrait), l'atomicité (élémentaire ou composé) et le niveau de réalité (réel ou fictif) [GOMEZ, 99]. Selon Uschold [USCHOLD and KING, 95], un concept se compose de trois parties :

(1) un ou plusieurs termes, (2) une notion et (3) un ensemble d'objets. Le ou les termes permettent d'identifier le concept. La notion, aussi appelée intention du concept, désigne la sémantique du concept défini au travers de ses propriétés et de ses attributs. Enfin, l'ensemble d'objets forme l'extension du concept, il s'agit de toutes les instances du concept.

Lors de la création d'une ontologie, le problème qui se pose est la sélection de ces concepts. Quels sont ceux à intégrer ? Comment définir les hiérarchies ? Comment surmonter la variabilité des représentations ? Pour répondre à ces questions, Bachimont [BACHIMONT, 00] propose de recourir à une normalisation sémantique des concepts en s'appuyant sur le paradigme différentiel. Celui-ci permet d'éviter toute ambiguïté de sens en définissant chacun des concepts en fonction des quatre principes différentiels que sont : (1) le principe de communauté avec le père, (2) le principe de différence avec le père, (3) le principe de différence avec les frères et (4) le principe de communauté avec les frères. Enfin, un concept se définit également au travers du lien qu'il peut entretenir avec les autres concepts de l'ontologie. De ce fait, Guarino [GUARINO et WELTY, 00] propose de qualifier un concept en se basant sur des propriétés formelles de rigidité, d'identité, d'unité et de dépendance.

- Relations : Dans une ontologie, les relations correspondent aux types d'interaction possible d'interaction possibles entre les concepts d'un domaine. De ce fait, elles se définissent à la fois par leur signature mais aussi par le contenu sémantique entre les concepts qu'elles unissent [BACHIMONT, 00]. Par exemple, deux relations peuvent porter sur deux mêmes concepts (signature sémantique égale) sans toutefois avoir le même sens (contenu sémantique différent). Dans une ontologie, les relations permettent ainsi de structurer les connaissances et de définir les d'interaction possibles entre les concepts d'un domaine. De ce fait, elles se définissent à la fois par leur signature mais aussi par le contenu sémantique entre les concepts qu'elles unissent [BACHIMONT ,00].

### 2.3.1 Les propriétés portant sur un concept

**La généricité :** un concept est générique s'il n'admet pas d'extension [GUARI, 97].

*Exemple :* la vérité est un concept générique.

**L'identité :** un concept porte une propriété d'identité si cette propriété permet de conclure quant à l'identité de deux instances de ce concept. Cette propriété peut porter sur des attributs du concept ou sur d'autres concepts. [GUARI, 97]

*Exemple* : le concept d'étudiant porte une propriété d'identité liée au numéro de l'étudiant, deux étudiants étant identiques s'ils ont le même numéro.

**La rigidité :** un concept est dit rigide si toute instance de ce concept en reste instance dans tous les mondes possibles.

*Exemple* : humain est un concept rigide, étudiant est un concept non rigide.

**L'anti-rigidité :** un concept est anti-rigide si toute instance de ce concept est essentiellement définie par son appartenance à l'extension d'un autre concept. *Exemple* : étudiant est un concept anti-rigide car l'étudiant est avant tout un humain.

**L'unité :** un concept est un *concept unité,* si pour chacune de ses instances, les différentes parties de l'instance sont liées par une relation

qui ne lie pas d'autres instances du concept. [GUARI, 97]

*Exemple* : Les deux parties d'un couteau, manche et lame sont liées par une relation «Emmanché » qui ne lie que cette lame et ce manche.

### 2.3.2 Les propriétés portant sur deux concepts

**L'équivalence:** deux concepts sont équivalents s'ils ont la même extension.

*Exemple* : étoile du matin et étoile du soir.

**La disjonction:** (on parle aussi d'incompatibilité) deux concepts

sont disjoints si leurs extensions sont disjointes.

*Exemple* : homme et femme.

**La dépendance** : Un concept C1 est dépendant d'un concept C2 si pour toute instance de C1 il existe une instance de C2 qui ne soit ni partie ni constituant de l'instance de C1. *Exemple* : parent est un concept dépendant de enfant (et vice-versa).

### 2.3.3. Les relations entre concepts

Les concepts (respectivement, les instances) peuvent être reliés entre eux par des relations au sein d'une ontologie. Une relation est définie comme une notion de lien entre des entités, exprimée souvent par un terme ou par un symbole littéral ou autre. Généralement, les liens sont classés en deux catégories : des liens *hiérarchiques* et des liens *sémantiques*.

Une relation *hiérarchique* lie un élément supérieur, dit l'hyperonyme, et un élément inférieur, dit l'élément hyponyme, ayant les mêmes propriétés que le premier élément avec au moins une en plus. Dans certains cas, le couple (*Hyperonymie*, *Hyponymie*) s'interprète par (*Type*, *Sous-Type*). L'hyperonyme englobe l'hyponyme. On pourra alors écrire « HYPONYME est une sorte de HYPERONYME ». *Exemple* : Le chat est une sorte d'animal, donc, " animal " est l'hyperonyme de "chat".

Une relation *sémantique* permet de lier des instances de concepts, ou des concepts génériques.

#### 2.3.3.1 Les propriétés intrinsèques à une relation

**Les propriétés algébriques :** symétrie, réflexivité, transitivité

**La cardinalité :** nombre possible de relations de ce type entre les mêmes concepts (ou instances de concept). Les relations portant une

cardinalité représentent souvent des attributs.

*Exemple*: une pièce a au moins une porte.

#### 2.3.3.2 Les propriétés liant deux relations

**L'incompatibilité:** on dit que deux relations R1 et R2 sont incompatibles si elles ne peuvent lier les mêmes instances de concepts.

*Exemple* : « être rouge » et « être vert » sont deux relations incompatibles.

**L'inverse :** on dit que deux relations binaires R1 et R2 sont inverses l'une de l'autre si, quand R1 lie deux instances I1 et I2, alors R2 lie I2 et I1.

*Exemple* : « a pour père » et «a pour enfant» sont deux relations inverses l'une de l'autre.

**L'exclusivité :** deux relations R1 et R2 sont exclusives si, quand R1 lie des instances de concepts, R2 ne lient pas ces instances, et vice-versa. L'exclusivité entraîne l'incompatibilité.

*Exemple* : l'appartenance et le non appartenance sont exclusives.

#### 2.3.3.3 Les propriétés liant une relation et des concepts

**Le lien relationnel :** Il existe un lien relationnel entre une relation R et deux concepts C1 et C2 si, pour tout couple d'instances des concepts C1 et C2, il existe une relation de type R qui lie les deux instances de C1 et C2. Un lien relationnel peut être contraint par une propriété de cardinalité, ou porter directement sur une instance de concept. [KASS, 02] *Exemple* : Il existe un lien relationnel entre les concepts « texte » et « auteur » d'une part et la relation « a pour auteur » d'autre part.

**La restriction de relation :** Pour tout concept de type C1, et toute

relation de type R liant C1, les autres concepts liés par la relation sont d'un type imposé.

*Exemple*: si la relation « mange » portant sur une « personne » et un « aliment » lie une instance de « végétarien », concept subsumé par « personne», l'instance de « aliment » est forcément instance de « végétaux ».

### 2.3.4 Les fonctions

Elles présentent des cas particuliers de relations dans lesquelles le n*ième* élément de la relation est unique pour les *n-1* éléments précédents. [GOBEN, 99]

Formellement, les fonctions sont définies telles que :

$$F: c1 * c2 * \ldots * cn - 1 \rightarrow cn$$

*Exemple de* fonctions : « *père-de* » et « *carré* » sont des fonctions binaires. Tandis que, « *prix-de voiture-usagée* » qui calcule le prix d'une voiture de seconde main en fonction du modèle de voiture, de la date de fabrication et du nombre de kilomètres est une fonction ternaire.

### 2.3.5 Les axiomes (ou Règles)

Les axiomes sont des expressions qui sont toujours vraies. Ils ont pour but de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, les connaissances n'ayant pas un caractère strictement terminologique [STAAB, 00]. Leur inclusion dans une ontologie peut avoir plusieurs objectifs :

- Définir la signification des composants.
- Définir des restrictions sur la valeur des attributs.
- Définir les arguments d'une relation.

- Vérifier la validité des informations spécifiées ou en déduire de nouvelles.

### 2.3.6 Les instances (ou individus)

Elles constituent la définition extensionnelle de l'ontologie ; elles sont utilisées pour représenter des éléments dans un domaine.

*Exemple*: les individus *Jean* et *Marie* sont des instances du concept « Personne ».

## 2.4 Typologies d'ontologies

Les ontologies peuvent être subdivisées en plusieurs dimensions, parmi celles-ci nous en examinerons cinq, à savoir :

- Selon l'objet de la conceptualisation.
- Selon le niveau de granularité.
- Selon le niveau de formalisation de la représentation des connaissances.
- Selon le niveau de la représentation des connaissances.
- Selon l'information dont l'ontologie a besoin et la richesse de sa structure interne.

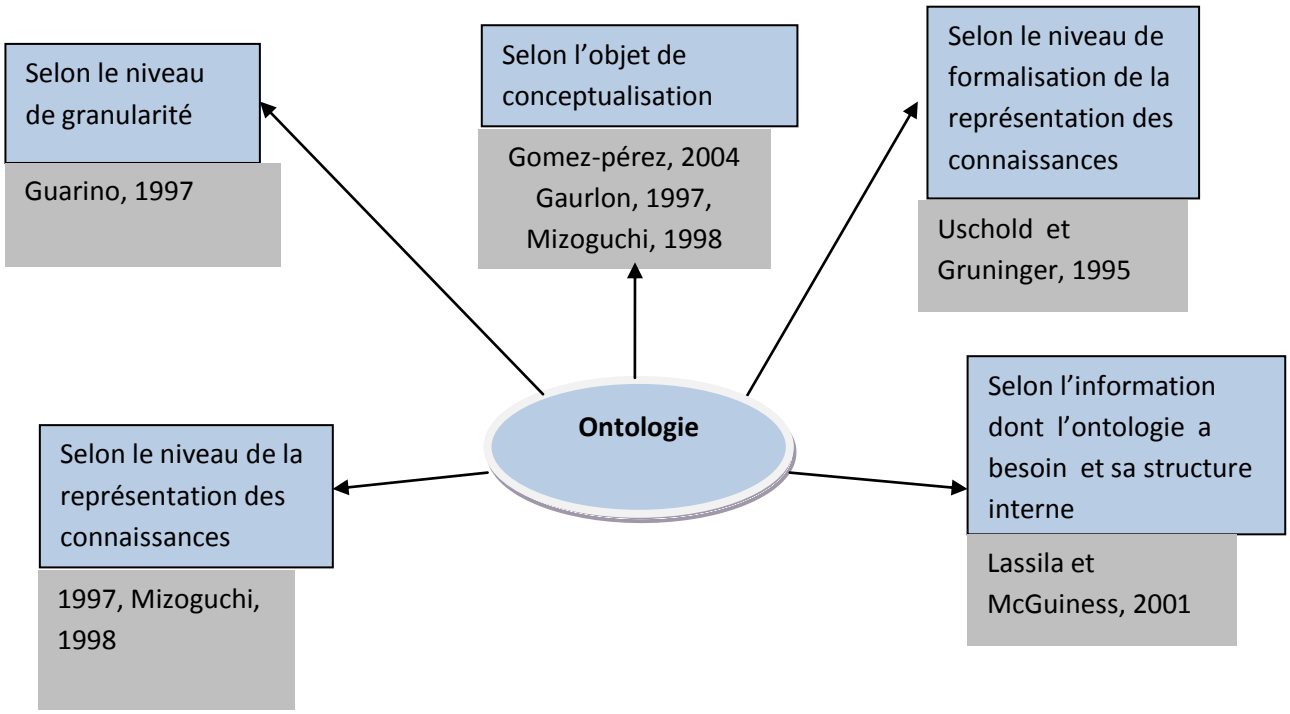



**Figure 2.1 :** Typologies des ontologies selon cinq dimensions.

### 2.4.1 Selon l'objet de la conceptualisation

Dépendamment de leur objet de conceptualisation, les ontologies sont classifiées de la manière suivante :

- **Représentation des connaissances.**
- **Haut niveau / supérieure.**
- **Générique** (dépendante du domaine ou de la tâche).
- **Domaine** (dépendante ou indépendante d'une tâche).
- **Tâche.**
- **Application** (dépendante du domaine ou de la tâche).
- **Méthode.**

#### 2.4.1.1 Ontologie de représentation de connaissances

Modélise les représentations primitives utilisées pour la formalisation des connaissances sous un paradigme donné. Par exemple, une ontologie sur le formalisme des Topic Maps comportera les concepts : Topic, Type de Topic, Association, Occurrence, Type Occurrence, etc.

#### 2.4.1.2 Ontologie de haut niveau / supérieure (Top-level / Upper-model)

Elle exprime des conceptualisations valables dans différents domaines. Elle décrit des concepts très généraux comme l'espace, le temps, la matière, les objets, les événements, les actions, etc. Ces concepts ne dépendent pas d'un problème ou d'un domaine particulier, et doivent être, du moins en théorie, consensuels à de grandes communautés d'utilisateurs.

Ce type d'ontologies est fondé sur la théorie de l'identité, la méréologie (*theory of whole and parts role*) et la théorie de la dépendance. Son sujet est l'étude des catégories des choses qui existent dans le monde. Comme les concepts de haute abstraction tels que les entités, les évènements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés, etc.

#### 2.4.1.3 Ontologie Générique (Generic Ontology)

Egalement appelée noyau ontologique modélise des connaissances moins abstraites que celles véhiculées par l'ontologie de haut niveau mais assez générales néanmoins pour être réutilisées à travers différents domaines. Cette ontologie inclue un vocabulaire relatif aux choses, évènements, temps, espace, causalité, comportement, fonction, etc. L'exemple le plus représentatif pourrait être une ontologie de méréologie qui inclurait le terme *partie-de*.

#### 2.4.1.4 Ontologie du Domaine (Domain Ontology)

Cette ontologie exprime des conceptualisations spécifiques à un domaine, elle est réutilisable pour plusieurs applications de ce domaine. Elle fournit les concepts et les relations permettant de couvrir les vocabulaires, activités et théories de ces domaines. Selon Mizoguchi [MIZO, 00], l'ontologie du domaine caractérise la connaissance du domaine où la tâche est réalisée. Par exemple, dans le contexte du e-learning, le domaine peut être celui de

formation.

#### 2.4.1.5 Ontologie de Tâches (Task Ontology)

L'ontologie de tâches fournit un vocabulaire systématisé des termes employés pour résoudre des problèmes liés aux tâches qui peuvent être ou non du même domaine. Elle fournit un ensemble de termes au moyen desquelles nous pouvons décrire généralement comment résoudre un type de problèmes. Elle inclut des noms, des verbes et des adjectifs génériques dans les descriptions de tâches [BEN HEBI, 12].

#### 2.4.1.6 Ontologie d'application (Application Ontology)

C'est l'ontologie la plus spécifique, elle contient des concepts dépendants d'un domaine et d'une tâche particuliers, elle est spécifique et non réutilisable. Ces concepts correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine activité [GUARI, 98]. Il s'agit donc ici de mettre en relation les concepts d'un domaine et les concepts liés à une tâche particulière de manière à en décrire l'exécution. Par exemple, dans le contexte du e-Learning, une application peut être la formation de Statistiques et Probabilités.

#### 2.4.1.7 Ontologie de méthodes

Ce type d'ontologie modélise les définitions des concepts et des relations pertinentes pour le processus de raisonnement afin d'effectuer une tâche spécifique. [LAVOI, 07].

### 2.4.2 Selon le niveau de granularité (détail)

Par rapport au niveau de détail utilisé lors de la conceptualisation de l'ontologie en fonction de l'objectif opérationnel envisagé pour l'ontologie, deux catégories au moins peuvent être identifiées [VALERY, 07] :

**Granularité fine :** ce niveau correspond à des ontologies très détaillées, elles possèdent ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche. Ce niveau de granularité peut s'avérer utile lorsqu' il s'agit d'établir un consensus entre les agents qui l'utiliseront.

**Granularité large :** ce niveau correspond à des vocabulaires moins détaillés. Par exemple, les scénarios d'utilisation spécifiques où les utilisateurs sont déjà préalablement d'accord à propos d'une conceptualisation sous-jacente. Les ontologies génériques possèdent une granularité large, compte tenu du fait que les concepts qu'elles traduisent sont normalement raffinés subséquemment dans d'autres ontologies de domaine ou d'application [BEN HEBI, 12].

### 2.4.3 Selon le niveau de formalisation de la représentation des connaissances

Par rapport au niveau de formalisation de la représentation des connaissances de l'ontologie , on peut distinguer quatre sortes d'ontologies : [USCHO et GRU, 96]

**Informelle*:*** l'ontologie est exprimée en langage naturel (sémantique ouverte). Cela peut permettre de rendre plus compréhensible l'ontologie pour l'utilisateur mais cela peut rendre plus difficile la vérification de l'absence de redondances ou de contradictions.

**Semi-informelle*:*** l'ontologie est exprimée dans une forme restreinte et structurée du langage naturel. Cela permet d'augmenter la clarté de L'ontologie tout en réduisant l'ambiguïté.

**Semi-formelle** :l'ontologie est exprimée dans un langage artificiel défini formellement.

**Formelle *:*** l'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle ainsi que des théorèmes permettant de prouver des propriétés de cette ontologie. L'intérêt

d'une ontologie formelle est la possibilité d'effectuer des vérifications telles que la complétude, la non-redondance, la consistance, la cohérence, etc.

### 2.4.4 Selon le niveau de la représentation des connaissances

L'appellation de niveaux de représentation des connaissances ainsi que leur interprétation peuvent varier selon les auteurs. À titre de comparaison, nous décrivons les typologies de Mizoguchi [MIZO, 98] et de Bachimont [BACHIMONT, 00].
D'une part, Mizoguchi propose une typologie sur trois niveaux très axée sur le processus d'ingénierie ontologique :

**Niveau 1 (**ou **niveau conceptuel) :** Il est spécifié en faisant abstraction de toute contrainte informatique et sert donc de support à l'acquisition des connaissances. Il est une collection structurée de termes. La tâche fondamentale dans la construction d'une ontologie est l'articulation du monde d'intérêt, c'est-à-dire l'extraction des concepts et leur identification dans des hiérarchies de type *is-a*. Cette étape est indispensable pour que l'on puisse parler d'ontologie. À ce stade, la modélisation étant informelle, une spécification en langage naturel complétée de graphiques est acceptable et des définitions minimales des concepts sont faites.

**Niveau 2 (**ou **niveau formel):** Il est formalisé au moyen d'un langage de représentation interprétable par une machine. En plus des
Définitions du niveau 1, des définitions formelles sont ajoutées pour empêcher des interprétations inattendues des concepts. De même, les relations ou contraintes nécessaires sont également traduites en langage machine pour former un ensemble d'axiomes. A ce stade, les relations sont plus riches que celles du niveau précédent. Les définitions sont déclaratives et formelles pour permettre aux ordinateurs de les interpréter. L'interprétation d'une ontologie à ce

niveau permet aux ordinateurs de répondre aux questions sur les modèles construits à partir de l'ontologie. La plupart des efforts de construction de l'ontologie aboutissent à ce niveau.

**Niveau 3** (ou **niveau opérationnel**) : Il est encodé au moyen d'un langage de programmation et la spécification obtenue est exécutable. L'ontologie est exécutable dans le sens où les modèles construits à partir de l'ontologie fonctionnent en utilisant des modules fournis par quelques-uns des codes abstraits associés aux concepts dans l'ontologie. Ainsi, elle peut répondre à des questions à propos de la performance du temps d'exécution des modèles.

D'autre part, Bachimont propose une typologie de trois niveaux très axée sur la construction de la signification :

**Niveau sémantique (**ou **interprétatif**) **:** tous les concepts (caractérisés par un libellé) doivent respecter les quatre principes différentiels :

Communauté avec l'ancêtre.

Différence (spécification) par rapport à l'ancêtre.

Communauté avec les concepts frères (situés au même niveau).

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 le libellé de 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.

**Niveau formel** (ou **référentiel**) **:** En ajout aux caractéristiques énoncées au niveau précédent, les concepts référentiels 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 sont identiques s'ils possèdent la même extension. *Exemple* : Les concepts « d'étoile du matin » et « d'étoiles du soir » associés à Venus.

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

### 2.4.5 Selon l'information dont l'ontologie a besoin et la richesse de sa structure interne

Lassila et McGuiness [LASSI, 01] proposent une classification des ontologies selon l'information dont l'ontologie a besoin et la richesse de sa structure interne. Cette classification consiste en un continuum allant d'ontologies légères aux ontologies lourdes:

**Vocabulaire contrôlé:** C'est un ensemble de termes définis par un groupe de personnes ou une communauté. La signification des termes n'est pas forcément définie et il n'y a pas d'organisation logique entre les termes. Ce vocabulaire peut être utilisé afin de labelliser des contenus documentaires. Les catalogues sont des exemples de vocabulaires contrôlés.

**Glossaire:** C'est un ensemble de termes avec leur signification. La définition de chaque terme est donnée en langage naturel. Cette représentation apporte plus d'informations car une personne peut lire la définition, cependant elle n'est pas interprétable par l'ordinateur.

**Thésaurus:** Glossaire contenant des descriptions sémantiques entre les termes. Ils sont principalement utilisés pour assister les

documentalistes dans la tâche d'indexation manuelle de documents. Ils sont reconnus pour présenter différents avantages dans ce contexte. Ils offrent tout d'abord une vue générale sur les termes et relations d'un domaine. Ils définissent ensuite un vocabulaire standardisé pour l'indexation. Ils permettent d'assurer qu'un seul terme d'un ensemble de synonymes soit choisi pour l'indexation (terme dit « à utiliser »). Ils sont également utilisés lors de la spécification d'une requête pour spécifier ou généraliser une recherche documentaire à partir des termes dits plus spécifiques ou plus génériques.

**Hiérarchie informelle de généralisation (is-a) :** sa structure est basée non pas sur des relations de généralisation mais sur la proximité des concepts.

**Hiérarchie formelle de généralisation (is-a):** sa structure est déterminée par des relations de généralisation.

**Hiérarchie formelle de généralisation (is-a) avec instances du domaine:** similaire à la catégorie précédente mais incluant des instances.

**« "Frame":** ontologies incluant des classes avec propriétés pouvant être héritées.

**Ontologies avec restrictions de valeur :** ontologies pouvant contenir des restrictions sur les valeurs des propriétés.

**Ontologies avec contraintes logique**s: ontologies pouvant contenir des contraintes entre constituants (exemple relations) définies dans un langage logique.

## 2.5 Utilité des ontologies en ingénierie des connaissances

Plusieurs chercheurs se sont intéressés à la finalité des exploitations des ontologies. Nous préciserons dans ce qui suit l'utilité des ontologies au sein des systèmes à base de connaissances (SBC) et du Web Sémantique :

**Les connaissances du domaine d'un SBC:** Les ontologies servent à représenter les connaissances du domaine d'un SBC. En particulier, elles servent de squelette à la représentation des connaissances du domaine dans la mesure où elles décrivent les objets, leurs propriétés et la façon dont ils peuvent se combiner pour constituer des connaissances du domaine complètes.

**La communication:** Les ontologies peuvent intervenir dans la communication entre personnes, et logiciels [USCHO et GRU, 96]. En effet, les ontologies servent par exemple, à créer au sein d'un groupe ou d'une organisation un "vocabulaire conceptuel commun". Dans ce cas, on est plutôt dans le cadre d'une ontologie informelle. Dans le cas de la communication entre personnes et systèmes, l'ontologie est formelle et sert en général une tâche précise dans le SBC ou le système d'information. L'ontologie est un puissant moyen pour lever les ambiguïtés dans les échanges.

**L'interopérabilité:** le développement et l'implantation d'une représentation explicite d'une compréhension partagée dans un domaine donné, peut améliorer la communication, qui à son tour permet une plus grande réutilisation, un partage plus large et une interopérabilité plus étendue [USCH et GRU, 96]. L'interopérabilité est donc une spécialisation de la communication qui permet de répertorier les concepts que des applications peuvent s'échanger même si elles sont distantes et développées sur des bases différentes.

**L'aide à la spécification de systèmes:** La plupart des logiciels conventionnels sont construits avec une conceptualisation implicite et

que la nouvelle génération des systèmes utilisant les travaux en intelligence artificielle devrait être basée sur une conceptualisation explicitement représentée [MIZO, 96]. En effet, l'ontologie fournit une classification des objets que doit manipuler le système.

**L'indexation et la recherche d'information:** Dans le Web Sémantique, les ontologies y sont utilisées pour déterminer les index conceptuels décrivant les ressources sur le Web.

## 2.6 Utilisation des ontologies

Les ontologies sont utilisées dans des applications différentes, que l'on peut classifier dans plusieurs domaines [USCHOLD et al., 99]. Les ontologies peuvent être utilisées dans les tâches d'intégration pour décrire la sémantique des sources d'information et pour rendre leur contenu explicite, comme elles peuvent être aussi utilisées pour d'autres besoins comme la communication…

**Intégration des ontologies**

Le mot intégration est utilisé avec plusieurs significations dans le domaine d'ontologie. [PINTO et al.,99] définie trois différentes significations :

a) **Intégration** : Construction d'une nouvelle ontologie réutilisant (en assemblant, étendant, spécialisant, ou adaptant) d'autres ontologies disponibles. Ces différentes ontologies font partie de la nouvelle ontologie.

b) **Fusion** : Construction d'une ontologie par la fusion de différentes ontologies dans une seule ontologie qui les unifie.

c) **Utilisation** : Construction d'une application utilisant une ou plusieurs ontologies pour spécifier ou implémenter un système à base de connaissance.

La distribution et l'hétérogénéité de l'information ont créé le besoin de l'intégration pour améliorer l'accès aux données et assurer une recherche pertinente. Différentes approches existent et sont évaluées dans [VGELE et al., 01] vis-à-vis de quatre caractéristiques principales:

Le rôle et l'architecture des ontologies affectent profondément le formalisme de représentation d'une ontologie. Les ontologies sont utilisées pour la description explicite de la sémantique de l'information source. Ces ontologies sont employées selon différentes méthodes :

a) ***Approche mono ontologique*** (single ontology approach) : cette approche utilise une ontologie globale fournissant un vocabulaire partagé pour la spécification de la sémantique. Toutes les sources d'informations sont reliées à une ontologie globale, cette dernière peut être une combinaison de plusieurs ontologies, la raison de cette combinaison est la modularisation d'une ontologie monolithique potentiellement grande. Une approche importante pour ce type d'intégration des ontologies est SIMS [YIGAL et al., 96]

Les approches mono-ontologie sont les approches les plus simples à mettre en œuvre lorsque les sources de données se réfèrent à des domaines similaires. Dans le cas où les sources concernent des domaines hétérogènes, il devient difficile voire impossible à concrétiser un engagement ontologique. Une autre limite importante de cette approche tient du fait que la modification des sources peut affecter l'ontologie globale.

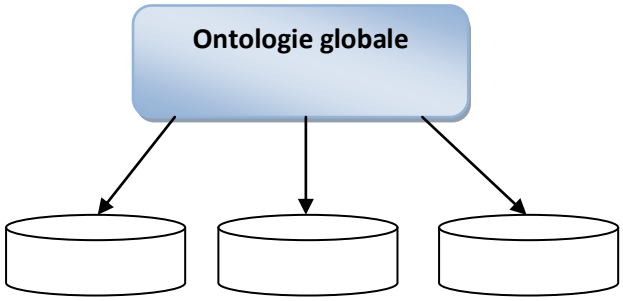



**Figure 2.2 :** Approche d'une ontologie simple

b) ***Approche multi ontologique*** (multiple ontology approach) : Dans cette approche, chaque source d'information est décrite par sa propre ontologie, Cette approche convient dans le cas où il devient difficile de trouver une ontologie commune résultant des grandes différences sémantiques existant entre les systèmes.

Des mappings inter-ontologiques sont alors nécessaires afin d'établir une interprétation commune des données. L'avantage de cette approche est l'absence d'une ontologie commune, chaque source d'ontologie est peut être développée indépendamment des autres, ce qui facilite le changement à savoir les modifications dans une source, l'ajout ou la suppression d'une autre source d'ontologie, l'inconvénient major est la difficulté de comparer les différentes ontologies due au manque d'un vocabulaire commun.

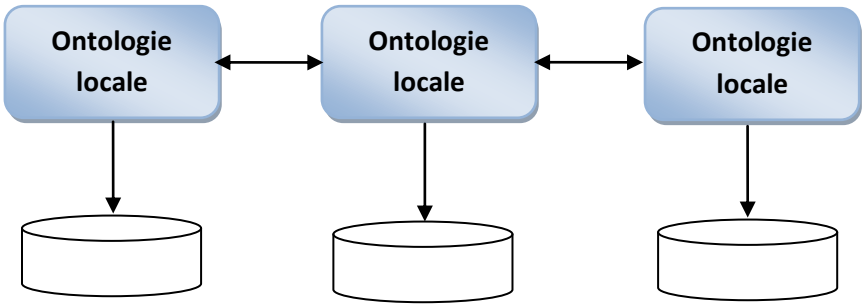



**Figure 2.3** : Approche d'ontologie multiple

b) ***Approche hybride*** : Comme l'approche multiple, la sémantique de chaque source est décrite par sa propre ontologie. Mais pour rendre les sources d'ontologies comparables, elles sont construites sur un vocabulaire partagé global [CHENG, 97] [WACHE, 99]. Ce vocabulaire partagé comprend les termes de bases ou primitives d'un domaine. Afin de construire des termes complexes, ces primitives sont combinées par des opérateurs, et les termes seront comparés plus facilement qu'une approche multiple. Parfois, ce vocabulaire partagé est représenté sous forme d'ontologie [HEINER, 00]. Dans COIN [CHENG, 97], la description locale d'une information, qu'on appelle contexte, est simplement un vecteur de valeurs d'attributs. Dans METACOTA [WACHE, 99], chaque source d'information est annotée par un label qui indique la sémantique de l'information. Ce label combine les termes primitifs du vocabulaire partagé. Dans BUSTER [HEINER, 00] le vocabulaire partagé est une ontologie générale. Une source d'ontologie est un raffinement de l'ontologie générale. L'avantage de cette approche est que l'on peut facilement ajouter une nouvelle source sans modifier le vocabulaire partagé ainsi elle supporte l'acquisition et l'évolution des ontologies alors que leur inconvénient réside dans le fait que les ontologies existantes ne peuvent pas être réutilisées facilement mais ils doivent être reconstruites à zéro parce que toutes les ontologies doivent référencier à un vocabulaire partagé.

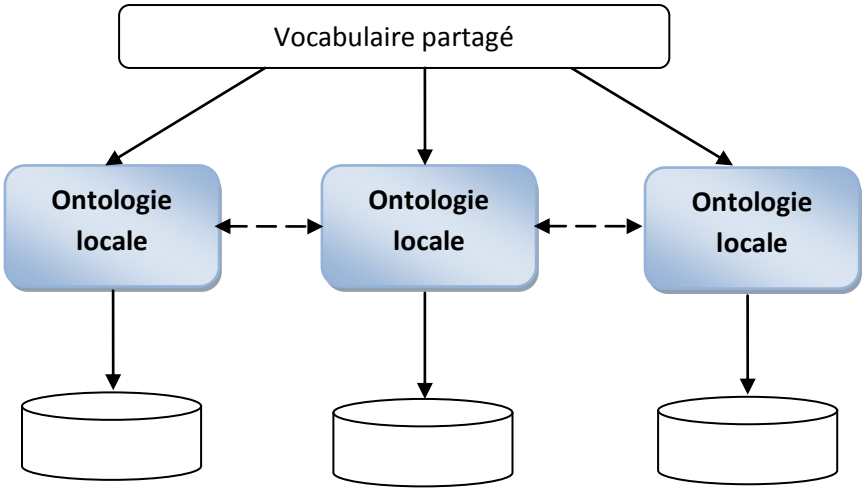



**Figure 2.4** : Approche d'ontologie hybride

## 2.7 Besoins supplémentaires

Dans cette partie nous allons voir quelques applications qui ont besoin d'utiliser les ontologies, l'idée principale est commune de toutes ces applications est l'utilisation d'ontologies afin de parvenir à un accord commun d'un domaine particulier [BOUQUET et al., 04].

- Communication : les communautés d'informations sont très utiles du fait quelles ont simplifié la communication et la coopération de ses membres par l'utilisation d'une terminologie partagée avec des significations bien définies, cependant la communication entre les membres des communautés différentes est difficile parce qu'elles ne conviennent pas sur une conceptualisation commune, elles peuvent employer le vocabulaire partagé du langage naturel alors que la majeure partie du vocabulaire utilisé dans les communautés est très spécialisé et n'est pas partagé avec d'autres communautés, cette situation exige une interprétation et une explication de la terminologie utilisé ; Les ontologies sont des bons choix pour surmonter ces problèmes [BOUQUET et al., 04], elle facilite la communication en fournissant une spécification explicite d'un domaine et elles permettent d'assurer la consistance et d'enlever l'ambiguïté dans les description des connaissances.

Interopérabilité : le domaine le plus importante d'application des ontologies est l'intégration des systèmes existants et la capacité d'échanger l'information au temps de l'exécution qui est également connue sous le nom de l'interopérabilité, la tentative de fournir l'interopérabilité souffre des problèmes similaire à ceux de la communication, la différence importante est que les acteurs ne sont pas des personnes capables de raisonner sur les sens des termes, mais des machines, afin de permettre aux machines de se comprendre, il faut bien expliquer le contexte de chaque système dans un niveau de formalité très élevé afin de rendre les machine compréhensible [BOUQUET et al., 04]. Les ontologies permettent de faciliter l'interopérabilité en intégrant les connaissances concernant différents domaines dont l'objectif est de décrire un domaine unifié ou accomplir une tâche commune.

## 2.8 Hétérogénéités des ontologies

Comme nous avons déjà vue dans les sections précédentes, les ontologies vise à constituer une représentation réelle qui puisse être acceptée par une communauté, et que les langages de représentation des ontologies et les méthodes de conceptualisation sont différents, ce qui engendre des hétérogénéités entre les ontologies ce qui limitent les possibilités d'interopérabilité entre eux, et gênent l'intégration aussi.

Dans cette partie nous allons explorer comment les ontologies peuvent se différer ; selon Klein [KLEIN., 01], l'hétérogénéité se manifeste sur deux niveaux : au niveau langage ou au niveau ontologique. Bouquet et al [BOUQUET et al., 04] ont ajouté un autre niveau d'hétérogénéité à savoir l'hétérogénéité pragmatique.

- les hétérogénéités au niveau du langage : se manifestent quand le langage de spécification diffère d'une ontologie à l'autre :

Les classes, les relations ne sont pas définies de la même manière. Les hétérogénéités de ce niveau sont situées sur le plan de la syntaxe, de la représentation logique et de l'expressivité.

• Syntaxe: la syntaxe définit la structure des représentations, les différences dans ce plan concernent le formalisme plutôt que le contenu. La solution de ce type est simple, un mécanisme de réécriture des ontologies dans le même langage.

• Représentation logique: les notions logiques sont exprimées de manière différente, par exemple dans un langage le fait que deux concepts A et B sont disjoints s'exprime par A intersect (not B)= true, et que dans un autre langage cette même relation s'exprime par A∩B=0. Ce problème est facilement résolu par des règles de translation d'une représentation logique à l'autre.

• Expressivité: implique que certains langages sont capables d'exprimer des notions alors que les autres ne peuvent pas, par exemple les listes, la négation, etc. Les hétérogénéités au niveau ontologique : Visser et al [VISSER et al., 98] ont distingué dans ce niveau, les hétérogénéités conceptuelles et celles terminologiques.

• Niveau terminologique : ce type d'hétérogénéité concerne toutes les différences liées au processus de nomination des entités (classes, propriétés, etc.). Il est représenté très souvent par la présence des synonymes, des homonymes, multilinguismes, les abréviations, etc. La difficulté major de ce niveau est la présence des homonymes qui n'est pas facile à résoudre.

• Niveau Conceptuel : est appelé aussi hétérogénéité sémantique dans Euzenat [EUZ SHV, 07], c'est la différence dans l'interprétation du domaine qui a comme conséquence différents concepts ou

différents relations entre concepts [VISSER et al., 98], Benerecitti et al [BENER et al., 01] ont identifie trois sortes d'hétérogénéités dans ce niveau :

- Couverture : qui désigne le fait que les ontologies couvrent différents portions de l'univers du discours.
- Granularité : qui signifie que les ontologies peuvent décrire les
- objets de la réalité à des degrés de détails différents.
- Perspective : qui signifie que les ontologies couvrent des points de vue différents.

Les hétérogénéités au niveau pragmatique [VISSER et al., 98]: le niveau pragmatique est le niveau le plus complexe. Il concerne les hétérogénéités d'interprétation d'une ontologie qui peuvent survenir lorsque des individus ou des communautés différentes interprètent différemment l'ontologie selon différents contextes.

## 2.9 Conclusion

L'ontologie est une conceptualisation d'un domaine à laquelle sont associés un ou plusieurs vocabulaires de termes. Les concepts se structurent en un système et participent à la signification des termes. Elle est définie pour un objectif donné et exprime un point de vue partagé par une communauté.

Conçues pour répondre aux problèmes posés par l'intégration des connaissances au sein des systèmes informatiques, les ontologies apparaissent désormais comme une clé pour la manipulation automatique de l'information au niveau sémantique.

La diversité et la puissance des applications potentielles des ontologies laissent à penser que leur place au sein des systèmes d'information ne peut que croître. Si les principaux projets utilisant des ontologies ne visent pour le moment que la gestion de connaissances au niveau sémantique, les ontologies pourraient permettre à terme la création de systèmes capables non seulement

de gérer des connaissances mais aussi de raisonner sur ces connaissances et, pourquoi pas, d'en produire de nouvelles.

Le chapitre suivant se reporte à l'ensemble des activités qui concerne le processus de développement des ontologies, le cycle de vie des ontologies, formalismes utilisés pour servir leur représentation et les outils de développement d'ontologies y compris les langages de spécification.

# Chapitre 3
# Ingénierie ontologique

*Longtemps on a pensé que l'informatique en général et les programmes d'intelligence artificielle en particulier allaient mélanger et présenter sous des angles neufs les concepts humains. Bref, on attendait de l'électronique une nouvelle philosophie. Mais même en la présentant différemment, la matière première reste identique : des idées produites par des imaginations humaines. C'est une impasse. La meilleure voie pour renouveler la pensée est de sortir de l'imagination humaine.*

*Edmond Wells, Encyclopédie du savoir relatif et absolu.*

## 3.1 Introduction

L'ingénierie ontologique (IO) a émergé de l'ingénierie de connaissances (IC). Cette dernière a longtemps été considérée comme le domaine de prédilection du développement d'expertises en conception de systèmes à base de connaissances. Malgré le fait que l'ingénierie des connaissances ait contribué à accroître cette expertise en l'organisant dans une perspective computationnelle, certains membres de la communauté de l'intelligence artificielle ont éprouvé le besoin de passer à une ingénierie basée sur d'autres fondements théoriques et méthodologiques, afin d'améliorer la conception des systèmes intelligents. [VALERY, 04]

L'ingénierie des connaissances a ainsi donné naissance à l'ingénierie ontologique, où l'« ontologie » est l'objet clé sur lequel il faut se pencher. La nécessité d'une ontologie et d'une ingénierie ontologique des systèmes à base de connaissances commence à être comprise et acceptée par la communauté. Fonder l'ingénierie ontologique exige que l'on puisse en définir l'objet et en défendre la spécificité méthodologique. [VALERY, 04].

Ce chapitre se reporte à l'ensemble des activités qui concerne le processus de développement des ontologies, le cycle de vie des ontologies, formalismes utilisés pour servir leur représentation et les outils de développement d'ontologies y compris les langages de spécification puis, nous passons en revue les différentes étapes intervenant dans la construction des ontologies. Enfin pour implémenter une ontologie, il convient d'utiliser un langage de spécification, ce travail est facilité par de nombreux outils assistant l'utilisateur en phase d'opérationnalisation.

Actuellement, il n'existe pas encore de consensus à propos des meilleures pratiques à adopter lors du processus d'ingénierie ou même de normes techniques régissant le processus de développement

des ontologies. En attendant la création d'une méthode standard de construction ontologique.

## 3.2 Le cycle de vie des ontologies

Étant donné que les ontologies sont destinées à être utilisées comme des composants logiciels dans des systèmes informatiques répondant à des objectifs opérationnels différents, leur développement doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. En particulier, elles doivent être considérées comme des objets techniques évolutifs et posséder un cycle de vie spécifique.
Les activités liées à une ontologie peuvent être regroupées en trois catégories [BLAZ, 98] :
Des activités de gestion de projet: planification, contrôle, assurance qualité.
Des activités de développement: spécification, conceptualisation, formalisation.
Des activités de support: évaluation, documentation, gestion de la configuration.

La Figure 3.1 représente les différentes activités qui expliquent que le cycle de vie préconisé est un cycle par prototypes : la vie d'une ontologie passe par les états suivants : spécification des besoins, conception (normalisation, formalisation et opérationnalisation) déploiement et diffusion, utilisation, évaluation et enfin évolution et maintenance. Le cycle de vie par évolution de prototypes permet à l'ontologiste de retourner de n'importe quel état à n'importe quel autre si une certaine définition manque ou est erronée. Ainsi, ce cycle de vie permet l'inclusion, le déplacement ou la modification de définitions n'importe quand durant le cycle de vie de l'ontologie. L'acquisition, la documentation et l'évaluation de

connaissances sont des activités de support qui sont effectuées pendant la majorité de ces états.

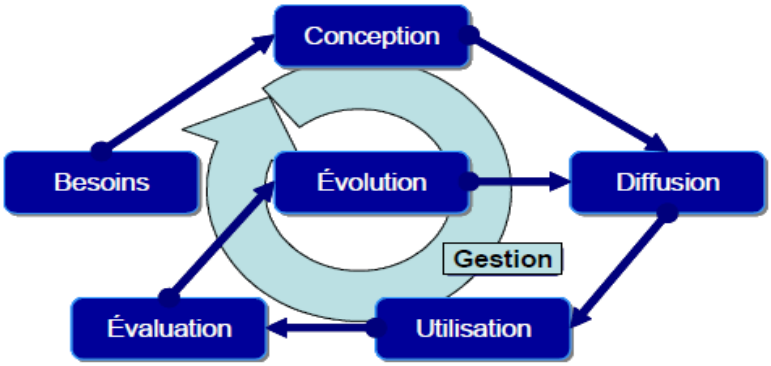



**Figure 3.1** : Le cycle de vie d'une ontologie [BANEYX, 07].

Fernandez et ses collègues [FERNA, 97] insistent sur le fait que les activités de documentation et d'évaluation sont nécessaires à l'étape du processus de construction d'ontologie, l'évaluation précoce permettant de limiter la propagation d'erreurs.

## 3.3 Fondements de l'ingénierie ontologique

### 3.3.1 Définition

L'ingénierie ontologique peut être définie comme une thématique de recherche visant à proposer des aspects pratiques, essentiellement des méthodes, des outils et des langages dédiés à l'application des résultats de la théorie des ontologies à la construction d'ontologie. [GAND, 02]

### 3.3.2 Processus

Le processus de construction d'une ontologie repose sur deux étapes : l'ontologisation et l'opérationnalisation (Figure 3.2) [BENA, 05]

L'ontologisation consiste à construire une ontologie conceptuelle. Cette étape est réalisée à partir de différentes sortes de données

telles que des glossaires de termes, d'autres ontologies, des textes, d'interviews d'experts, etc.

L'opérationnalisation consiste à coder l'ontologie conceptuelle obtenue à l'aide d'un langage de représentation de connaissances opérationnel (i.e. doté de mécanismes d'inférences).

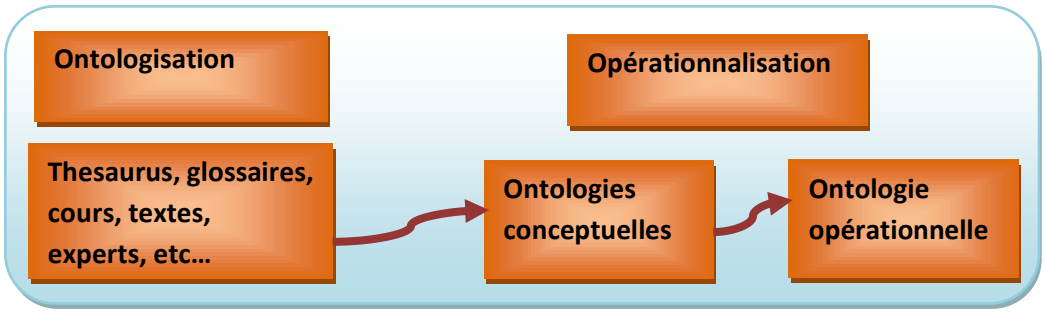



**Figure 3.2 :** Etapes du processus de développement d'une ontologie. [BENA, 05]

Il est à noter que ce processus n'est pas linéaire et que de nombreux allers-retours sont à priori nécessaires pour développer une ontologie opérationnelle adaptée aux besoins.

### 3. 3.3 Principes

Uschold et Gruninger (1996) suggèrent plusieurs critères utiles à la fois pour guider et évaluer la conception d'une ontologie : [BENA, 05]

**Clarté:** les ambiguïtés doivent être minimisées, les distinctions entre concepts motivées et la compréhension des définitions facilitée par des exemples.

**Cohérence:** les axiomes doivent être logiquement consistants. Les parties non axiomatiques des définitions doivent être cohérentes.

**Extensibilité:** de nouveaux termes et concepts, généraux et spécialisés, devraient être inclus dans l'ontologie d'une façon qui n'exige pas la révision des définitions existantes.

**Engagement ontologique minimal:** Ce principe invite à faire aussi peu de réclamations que possible au sujet du monde représenté.

**Biais de codage minimal :** il faut minimiser les choix de représentation faits pour des raisons d'implémentation.

Il existe d'autres principes tels que :

**Complétude :** une définition exprimée par des conditions nécessaires et suffisantes est préférée à une définition partielle (définie seulement par une condition nécessaire et suffisante). [GRUBER, 93]

**Distinction ontologique :** les classes de l'ontologie devraient être disjointes. Le critère utilisé pour isoler le noyau de propriétés considérées comme invariables pour une instance d'une classe est appelé le critère d'*Identité*. [BORG, 96]

**Modularité :** Ce principe vise à minimiser les couplages entre les modules. [BERNA, 96]

**Distance sémantique minimale :** il faut minimiser la distance entre les concepts enfants de mêmes parents. Les concepts similaires sont regroupés et représentés comme des sous-classes d'une classe, et devraient être définis en utilisant les mêmes primitives. Les concepts qui sont moins similaires sont représentés plus loin dans la hiérarchie. [ARPI et al., 98]

**Normalisation des noms :** ce principe indique qu'il est préférable de normaliser les noms autant que possible.

**Diversification des hiérarchies :** ce principe est adopté pour augmenter la puissance fournie par les mécanismes d'héritage multiple. Si suffisamment de connaissances sont représentées dans l'ontologie et que suffisamment de différentes classifications de critères sont utilisées, il est plus facile d'ajouter de nouveaux concepts (puisqu'ils peuvent être facilement spécifiés à partir des concepts et des classifications de critères préexistants) et de les faire hériter de propriétés de différents points de vue. [ARPI et al., 98].

### 3.3.4 Méthodes d'ingénierie ontologique

Il existe une multitude de méthodes d'ingénierie ontologique mais l'absence de directives structurées et communes ralentisse le développement d'ontologie à l'intérieur et entre les équipes, l'extension de n'importe quelle ontologie, la possibilité de réutilisation de l'ontologie. On entend par méthodologie, les procédures de travail, les étapes, qui décrivent le pourquoi et le comment de la conceptualisation puis de l'artefact construit.

#### 3.3.4.1 La méthode d'Uschold et King

Uschold et King [USCHO et KING, 95] ont proposé la première méthode de construction d'ontologie qui est basé sur l'expérience acquise lors du développement de l'ontologie d'Entreprise « the enterprise ontology ». La figure 3.3 présente le processus de la méthode qui se compose de quatre étapes :

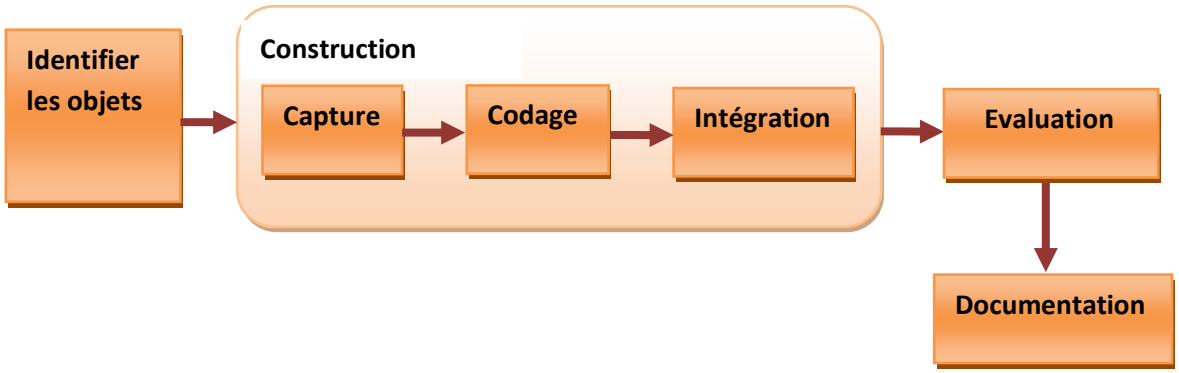



**Figure 3.3 :** Processus de la méthode d'Uschold et king [USCHO et KING, 95].

**Identification des objectifs et du contexte de l'ontologie :** Le but de cet étape est de clarifier le pourquoi de la construction de l'ontologie, les utilisations prévues et la finalité (réutiliser, partager, utiliser comme une partie d'un KB, etc.) et les utilisateurs s potentiels de l'ontologie.

**Construction de l'ontologie :** Cette étape est divisée en trois activités

**1. Capture de l'ontologie:** cette étape se fait indépendamment d'un langage de représentation (conceptualisation). Elle commence par l'identification des concepts et des relations clés ensuite produire en langage naturel les définitions précises et non ambiguës pour les concepts et les relations et se termine par l'identification des termes dénotant les concepts et les relations pour ainsi essayer d'arriver à un agrément.
Pour la catégorisation et la capture de l'ontologie, Uschold et Grüninger [USCHO et GRU, 96] proposent trois approches :

**Approche descendante** *(Top Down)* : l'ontologie est construite par généralisation en partant des concepts des basses couches taxinomiques. Cette approche encourage la création d'ontologies spécifiques et adaptées.

**Approche ascendante** *(Buttom Up)* : l'ontologie est construite par spécialisation en partant de concepts des hautes couches taxinomiques. Cette approche encourage la réutilisation d'ontologies.

**Approche intermédiaire** *(Middle Out)* : l'ontologie est construite à partir de concepts centraux puis les généralise ou les spécialise pour former les couches hautes et les couches basses de l'ontologie. Cette approche encourage l'émergence de domaines thématiques dans l'ontologie et favorise la modularité.

**2. Codage de l'ontologie :** Cette étape implique deux tâches : Représentation explicite de la conceptualisation (ex. classe, entité, relation).L'écriture du code dans un langage formel : Prolog, KL-One, OIL, CG, OWL…

**3. Intégration d'ontologies existantes :** Cette étape fait référence à comment utiliser les ontologies qui existent. Elle peut être faite en parallèle avec les étapes de capture et/ou de codage. Par exemple ; The Frame Ontology peut être réutilisée pour modéliser les ontologies de domaine qui utilise l'approche basée sur les frames.

**Evaluation de l'ontologie:** c'est la mise à l'épreuve de l'ontologie en la faisant confronter aux objectifs pour lesquels elle a été conçue et aux utilisateurs.

**Documentation de l'ontologie:** cette étape est essentielle pour l'acceptation de l'ontologie.

#### 3.3.4.2 La méthode SENSUS

C'est une méthode qui consiste à construire une ontologie de domaine à partir d'une plus grande ontologie, l'ontologie SENSUS [SWART, 97]. La méthode propose de relier les termes spécifiques du domaine à cette ontologie et de supprimer dans SENSUS, ceux qui ne sont pas pertinents pour la nouvelle ontologie. Le résultat de ce processus est le squelette de cette nouvelle ontologie, qui est générée automatiquement en utilisant ce processus et l'outil OntoSaurus.
Selon SWARTOUT, un avantage important de cette méthode est que si deux ontologies sont développées indépendamment, la large couverture de l'ontologie SENSUS agit comme une « charnière » qui couple la terminologie et l'organisation d'une ontologie avec l'autre.
Conformément à cette méthode, la construction d'une ontologie dans un domaine spécifique doit suivre les étapes suivantes :

Identifier les termes clés du domaine avec l'aide des experts.

Relier manuellement les termes clés à SENSUS.

Inclure tous les concepts qui se trouvent sur le chemin depuis le terme clé jusqu'à la racine de SENSUS.

Ajouter les nouveaux termes de domaine. Dans cette étape on ajoute manuellement tous les termes qui sont relevant pour le domaine et qui ne sont pas encore apparus. Ainsi les étapes 2 et 3 sont répétées pour inclure les concepts sur le chemin des nouveaux concepts jusqu'à la racine de SENSUS.

Ajouter le sous arbre entier. Dans cette étape, celui qui fait l'ontologie doit faire attention aux nœuds qui ont un grand nombre de chemins qui passe à travers le nouvel arbre généré.

Points forts de la méthode : elle favorise le partage de la connaissance

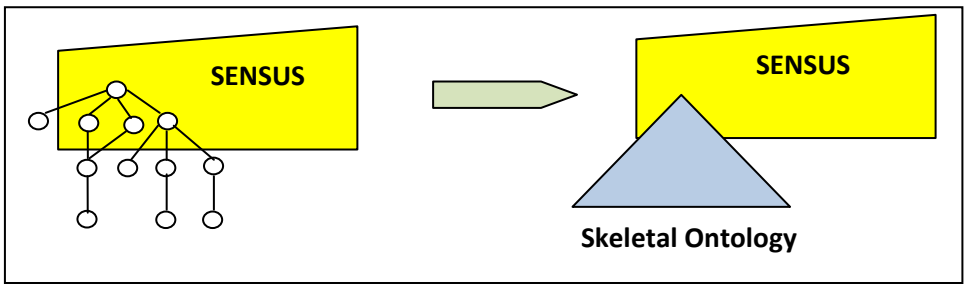



**Figure 3.4 :** Méthode SENSUS. [SWART 97]

### 3.3.4.3 La méthode On-To-Knowledge (OTK)

Le but du projet On-To-Knowledge [STAAB, 01] est d'appliquer les ontologies aux informations disponibles électroniquement pour améliorer la qualité de la gestion de la connaissance au sein des organisations larges et distribuées. Dans le cadre de ce projet, ils ont développé une méthodologie et des outils pour l'accès intelligent aux sources de large volume d'information textuelle et semi-structurée dans des environnements basé sur l'intranet, extranet et internet.

Cette méthode inclut une méthodologie pour la construction d'ontologies qui vont être utilisées par les applications de management de connaissances. Donc, la méthode On-To- Knowledge propose de construire une ontologie en tenant compte de comment l'ontologie va être utilisée par l'application plus tard. Par conséquent les ontologies développées en suivant cette méthode sont très dépendantes de l'application.

La méthode On-To-Knowledge se décompose en cinq étapes (Figure 3.5) :

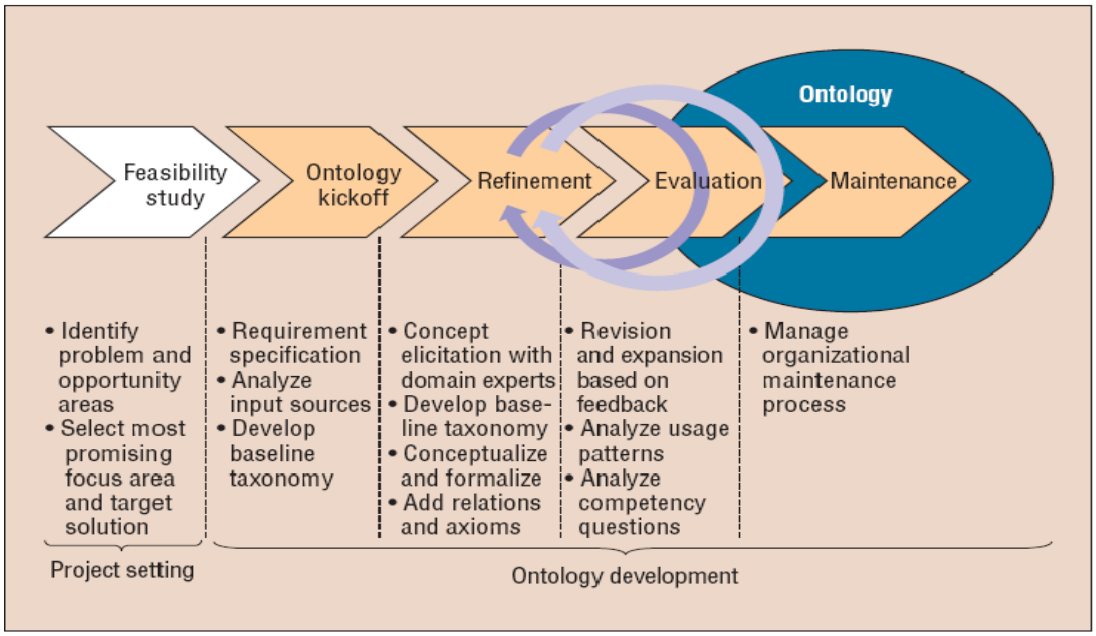



**Figure 3.5:** La méthode On-To-Knowledge. [STAAB, 01]

**Etude de faisabilité :** cette étape se fait comme suit :

Identifier les ressources de problème ainsi que les solutions potentielles et les mettre dans une perspective organisationnelle plus large.

Décrire la faisabilité économique et technique du projet, pour choisir le secteur le plus prometteur et la solution cible.

Identifier les taches appropriées, les agents qui vont exécuter et les articles de connaissances utilisés par les agents dans l'exécution des taches.

**Démarrage (Kickoff) :** Le résultat de cette étape est un document de spécifications des besoins d'ontologie. Ce document doit décrire soins Les directives de conception (par exemple, les conventions de nommage).
Source de connaissance (d'informations) valable (livres, magazines, interviews, etc.). Les utilisateurs potentiels et les cas d'utilisation.
Le démarrage (Kickoff) devrait permettre aux concepteurs d'ontologie de décider en ce qui concerne l'inclusion ou l'exclusion des concepts dans l'ontologie et de leur structure hiérarchique. Dans cette étape les développeurs devraient chercher des ontologies existantes qui sont potentiellement réutilisables.

**Raffinement :** l'objectif de cette étape est de produire une application orientée application conformément aux spécifications données à l'étape de Kickoff. Elle est divisée en deux sous étapes :

**Mise à jour de la connaissance :** le premier brouillon de l'ontologie obtenu à la deuxième étape est raffiné au moyen des interactions avec les experts du domaine. Après les axiomes sont identifiés et modélisés. Durant la mise à jour les concepts sont recueillis d'un coté et les termes pour nommer les concepts de l'autre coté. Alors les termes et les concepts sont reliés. On propose d'utiliser des représentations intermédiaires du modèle de connaissance, s'il y a plusieurs experts qui participent à la construction de l'ontologie, il est nécessaire de trouver un compromis.

**Formalisation :** l'ontologie est implémentée dans un langage d'ontologie. Pour effectuer la formalisation On-To Knowledge recommande l'éditeur d'ontologie OntoEdit qui offre la possibilité de générer automatiquement le code de l'ontologie
dans plusieurs langages.

**Evaluation :** cette étape sert à prouver l'utilité du développement de l'ontologie et les applications associées. Elle consiste à contrôler si l'ontologie satisfait les spécifications (besoins), la tester ensuite dans le cadre de son environnement d'application. Cette étape d'évaluation est fortement liée à celle de raffinement. En effet plusieurs aller retour sont nécessaires avant d'atteindre le niveau de satisfaction souhaité.

**La maintenance :** il est important de préciser qui va s'occuper de la maintenance et comment ?

On-To-Knowledge propose que la maintenance de l'ontologie soit effectuée comme une partie de l'application.

#### 3.3.4.4. La méthode de Grüninger et Fox

La méthode de Grüninger et Fox [GRUNI et FOX, 95] est basée sur l'expérience du développement du projet TOVE (*TOronto Virtual Enterprise project Ontology)*, qui est une ontologie dans le domaine de la modélisation des activités et des processus d'affaire. La figure 3.6 présente le processus de la méthode.

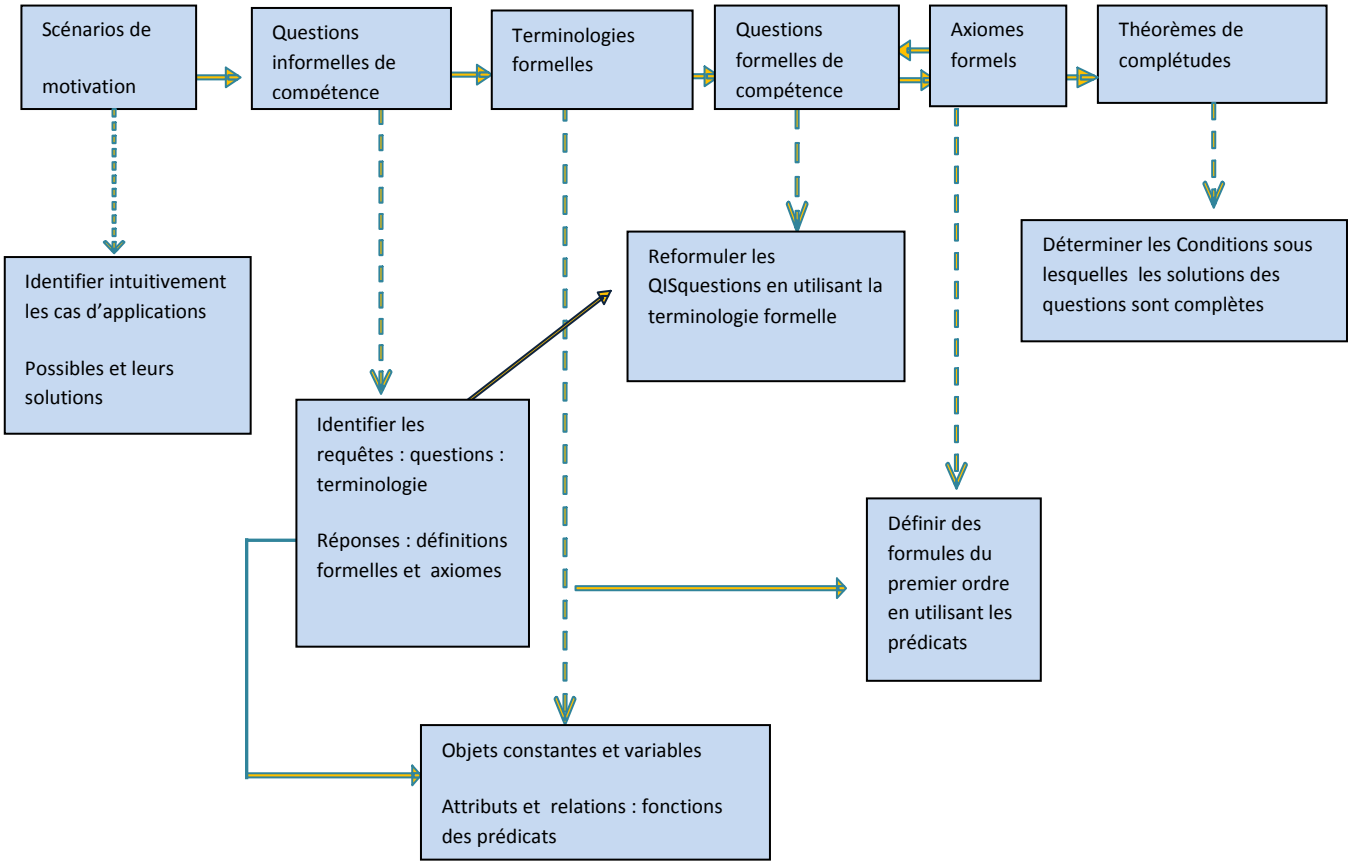



**Figure 3.6 :** Processus de la méthode de Grüninger et Fox [GRUNI et FOX, 95].

Cette méthode se compose de six étapes :

**Capture des scénarios motivants :** Le développement d'ontologies est motivé par les scénarios qui surgissent dans l'application. Un scénario de motivation fournit un jeu de solutions intuitives pour les problèmes du scénario. Ces solutions fournissent une sémantique informelle destinée aux objets et relations qui seront inclus dans l'ontologie. Pour toute nouvelle ontologie ou l'extension d'une ontologie ; un ou plus d'un scénario de motivation doivent être décrits.

**Formulation des questions informelles de compétence :** Cette étape repose sur les résultats obtenus sur les scénarios de l'étape précédente qui peuvent être réclamés comme des exigences expressives sous forme de questions. Une ontologie doit être capable de représenter ces questions en employant sa terminologie et de caractériser les réponses à ces questions en employant les axiomes et les définitions.

- **Spécification de la terminologie de l'ontologie en langage naturel:** Une fois la question de compétence formulée, la terminologie de l'ontologie doit être spécifiée dans une logique du premier ordre.
- **Formulation des questions formelles de compétence :** Les questions de compétence informelle vont être reformulées en utilisant la terminologie formelle définie à l'étape précédente.
- **Spécification des axiomes de l'ontologie dans un langage formel:** Le processus de définition des axiomes constitue la tache la plus délicate dans le processus du développement d'une ontologie, cependant ceci est guidé par les questions formelles de compétence.

**Théorème de complétude:** Dans une application donnée, il est possible de vérifier que le vocabulaire utilisé correspond à celui de l'ontologie, mais si cela garantit sa consistance il ne garantit pas pour autant sa complétude. Alors une fois les questions de compétence sont formellement exposées et les axiomes spécifiés, il faut définir les conditions qui doivent être remplies afin que les réponses aux questions de compétence soient complètes

#### 3.3.4.5 La méthode KACTUS

Cette méthode est conditionnée par le développement d'applications. Par conséquent, chaque fois qu'une application est construite, l'ontologie qui représente la connaissance requise pour l'application est construite [SCHE, 97]. La figure 3.7 présente le processus de la méthode. Les applications qui ont été créées sont dans le domaine du diagnostic d'erreur dans les réseaux électriques. Une ontologie peut être développée en réutilisant d'autres ontologies et peut aussi être intégrée dans les ontologies d'applications futures. Cette méthode repose sur trois étapes :

**Spécification de l'application :** Cette phase consiste à délimiter le contexte d'application avec l'ensemble des composants sensés être modélisé dans l'application.

**Design (conception) préliminaire :** basée sur des catégories ontologiques de haut niveau appropriées où la liste des termes et des taches développée pendant la phase précédente est utilisée pour l'obtention de plusieurs vues du modèle globale conformément aux catégories du plus haut niveau ontologique.

**Raffinement et structuration de l'ontologie :** Cette étape consiste à raffiner le travail fait dans les deux étapes précédentes afin d'assurer
une bonne conception, les principes de couplage minimum peuvent être utilisés pour s'assurer que les modules issus de la conception n'ont pas une forte relation de dépendance entre eux afin d'obtenir une homogénéité maximale dans chaque module.

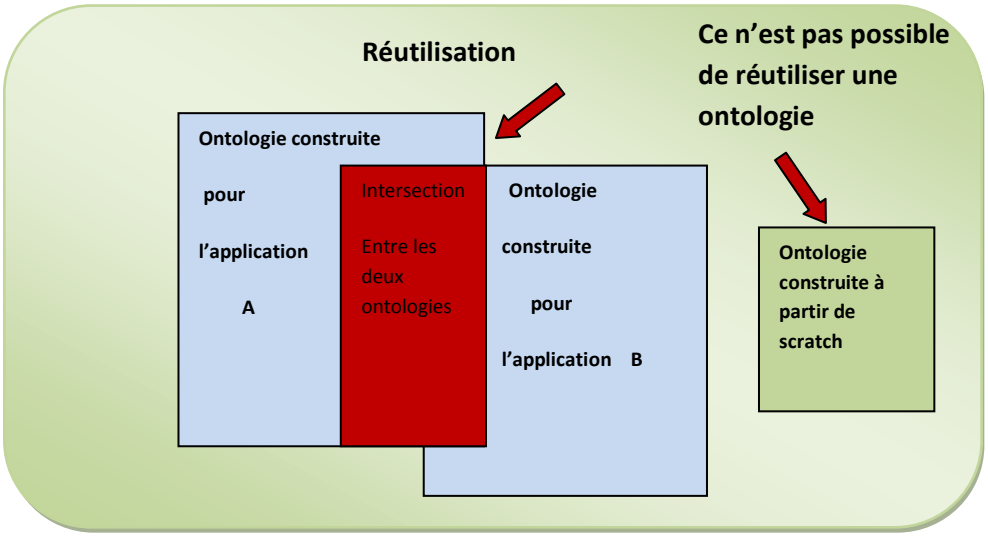



**Figure 3.7 :** Processus de la méthode KACTUS

### 3.3.4.6 La méthode de Bachimont

Cette méthode propose de contraindre l'utilisateur à un engagement sémantique en introduisant une normalisation sémantique des termes manipulés dans l'ontologie. La méthode de normalisation suit trois étapes [BACHIMONT, 00]

**Normalisation sémantique :** L'utilisateur doit choisir les termes du domaine et les normaliser en explicitant leurs propriétés et en exprimant les identités et les différences dans leur voisinage proche.

**Formalisation des connaissances:** Cette étape consiste à désambiguïser les notions de l'ontologie référentielle obtenue par l'étape précédente et choisir leurs sens pour un domaine spécifique. Cela peut nécessiter la création de nouveaux concepts, l'ajout de propriétés et d'axiomes.

**Opérationnalisation des connaissances:** Le système utilise un langage opérationnel de représentation de connaissances qui possède

les caractéristiques nécessaires pour répondre aux besoins exprimés lors de la spécification du système.

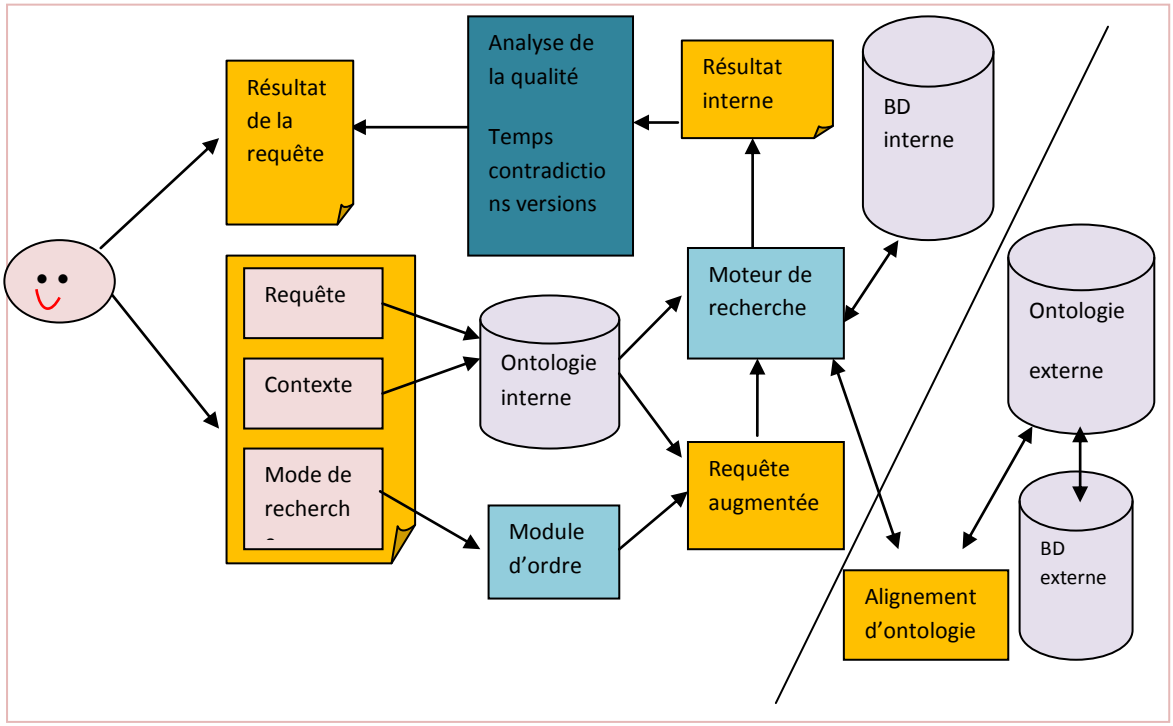



**Figure 3.8 :** Processus de la méthode Bachimont [BACHIMONT, 00]

### 3.3.4.7 La méthode METHONTOLOGY

C'est une méthodologie mise au point par l'équipe du laboratoire de l'intelligence artificielle de l'université polytechnique de Madrid.

Comme le montre la figure 3.9, cette méthode inclut :

- L'identification du processus de développement de l'ontologie.
- Le cycle de vie basé sur l'évolution de prototypes.
- Les techniques de gestion de projet (planification, assurance qualité), de développement (spécification, conceptualisation, formalisation, implémentation, maintenance) et des activités de support (intégration, évaluation, documentation).

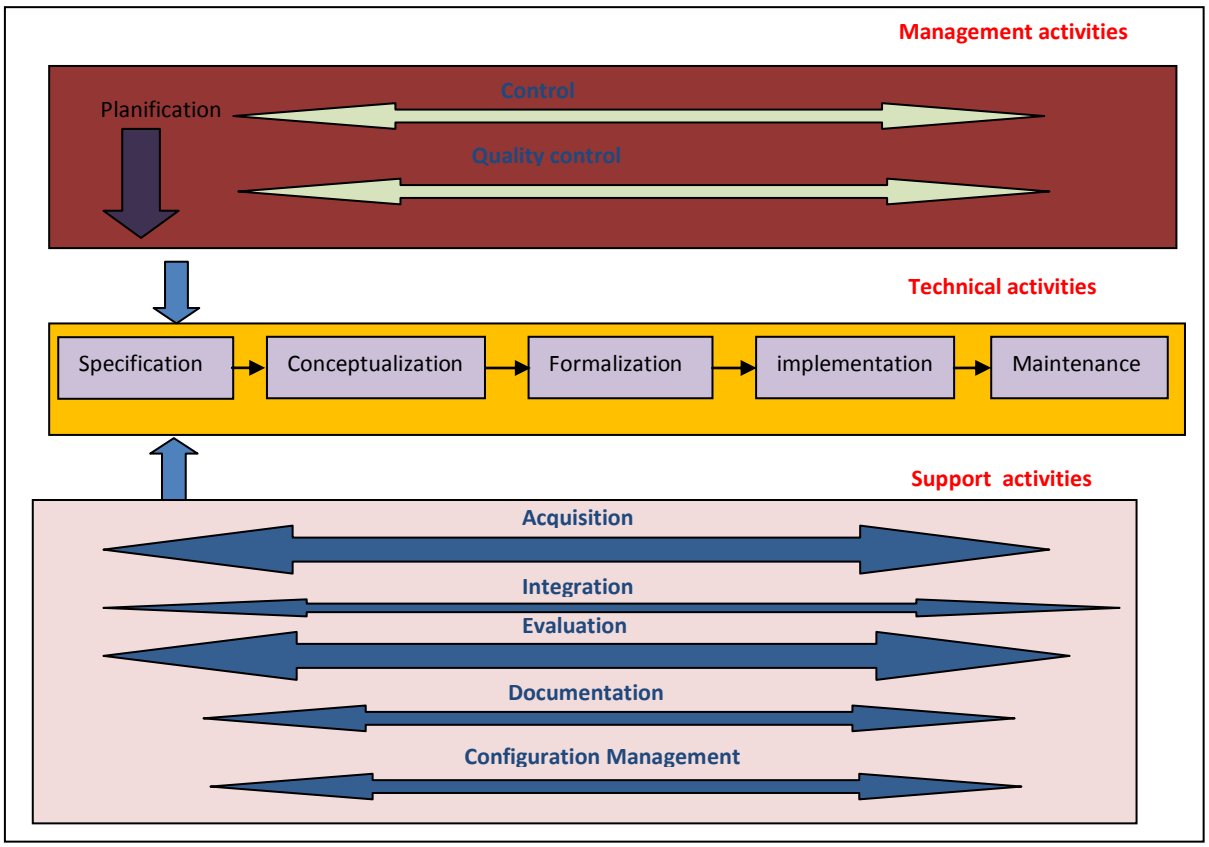



**Figure 3.9 :** Processus et cycle de vie de la méthode METHONTOLOGY. [FERNA, 97]

### 3.3.4.8 La méthode développée par l'université de STANFORD

"Ontology Development 101" [NATAL, 02] : a été développée à l'Université de Stanford, elle cherche à construire des ontologies formelles par la reprise et l'adaptation des ontologies déjà existantes. Elle comporte sept étapes qui sont les suivantes :

1. Déterminer le domaine et la portée de l'ontologie.
2. Réutiliser des ontologies existantes.
3. Enumérer les termes importants de l'ontologie.
4. Définir les classes et la hiérarchie de classes.
5. Définir les propriétés des classes (les attributs).

6. Définir les facettes des attributs.
7. Créer des instances des classes dans la hiérarchie ou exemples.

Elle utilise comme support les outils Protégé 2000 [NOY, 00] et Ontolingua [FARQ et al., 00].
Détaillons maintenant ces étapes :

**Etape 1 :** Déterminer le domaine et la portée de l'ontologie.

Cette étape se fait en répondant aux questions ci-dessous tout au long de la conception de l'ontologie et qui aident à définir la portée du domaine de l'ontologie :
Quel est le domaine que va couvrir l'ontologie ? Dans quel but utiliserons-nous l'ontologie ?
A quels types de questions l'ontologie devra-t-elle fournir des réponses?
Qui va utiliser et maintenir l'ontologie ?

Afin de mieux cerner la portée de l'ontologie on peut recourir aux questions de compétence, c'est-à-dire les questions auxquelles le système projeté est censé répondre. Ces questions permettent de savoir si l'ontologie contient suffisamment d'informations pour répondre aux questions et s'il y a une nécessité de détail dans les réponses ou de représentation d'un domaine particulier.

***Etape 2 :*** *envisager une éventuelle réutilisation des ontologies existantes.*

*Dans tout domaine de recherche, il est utile de profiter de ce que les autres ont fait afin d'en tirer les informations et ainsi permettre d'élargir le travail et l'affiner pour répondre aux besoins. Il est*

*intéressant d'importer des ontologies déjà existantes (dans le même domaine) et les adapter à nos besoins.*

***Etape 3 :*** *Enumérer les termes importants de l'ontologie.*
*Il est important d'établir en premier lieu une liste complète des termes concernant le domaine d'intérêt, et cela sans se soucier de la catégorisation de ces derniers dans des classes ou dans une hiérarchie.*

*On peut extraire ces termes à partir de produits terminologiques comme les corpus, les thésaurus, les taxonomies etc. les questions à se poser pour établir cette liste sont les suivantes :*

*Sur quels termes souhaiterons-nous discuter ? Quelles sont les propriétés de ces termes ?*

*Que veut-on dire par ces termes ?*

***Etape 4 :*** *Définir les classes et la hiérarchie de classes.*

A partir de la liste de l'étape précédente, on commence par définir les classes en sélectionnant les termes qui décrivent des objets ayant une existence indépendante. Ce sont ces termes qui constitueront les classes de l'ontologie. Il faut ensuite organiser ces classes dans une taxonomie hiérarchique en suivant la règle suivante : *« Si une classe A est super classe d'une classe B alors toute instance de B est aussi une instance de A.»*

Il existe différentes approches qui permettent la définition d'une hiérarchie des classes, il n'y a pas de meilleure méthode. Cela dépend en fait du point de vue du développeur :

Un procédé de développement de haut en bas : c'est une approche systémique qui commence par une définition des concepts les plus

généraux du domaine et se poursuit par la spécialisation des concepts.

Un procédé de développement de bas en haut : commence par la définition des classes les plus spécifiques, appelées feuilles de la hiérarchie, et se poursuit avec le regroupement de ces classes en concepts plus généraux.

Un procédé combiné de développement : représente une combinaison des deux approches précédentes, c'est l'approche la plus souvent utilisée par les développeurs en raison du fait que les concepts du milieu sont en général plus descriptifs du domaine. Au tout début, les concepts les plus saillants sont définis ensuite ils sont généralisés ou spécialisés selon le cas.

**Etape 5 :** Définir les propriétés des classes (les attributs)

Dans cette étape, on devra décrire la structure interne des concepts tirés pendant l'étape précédente. Les propriétés définissent la structure interne et les caractéristiques des classes. La plupart des termes restants (qui ne sont pas des classes) ont de fortes chances de représenter les propriétés de ces classes. Chaque propriété sera ensuite rattachée comme attribut à la classe qu'elle décrit. Il faut ensuite prendre en considération les classes et les sous-classes, ainsi un attribut doit être rattaché à la classe la plus générale pouvant avoir cette propriété et toutes les sous-classes de cette classe héritent cet attribut.
On peut citer quelques recommandations :

**Les attributs inverses :** on parle d'attributs inverses quand la valeur d'un attribut dépend de la valeur d'un autre attribut. Il est parfois plus commode de stocker l'information dans les deux sens bien que cela paraisse redondant.

**Valeurs par défaut :** une valeur par défaut d'un attribut est une valeur particulière qui est la même pour la plupart des instances d'une classe. Le système renseigne automatiquement cette valeur par défaut à chaque création d'une nouvelle instance appartenant à une classe comportant cet attribut.

**Nommer les classes et les attributs :** il faut définir une convention de nomination des classes et attributs et y adhérer complètement. Cela permet de mieux comprendre l'ontologie et d'éviter les erreurs les plus fréquentes de modélisation.

**Etapes 6 :** définir les facettes des attributs.
Les attributs peuvent avoir plusieurs facettes. Les facettes les plus communes décrivent : Le type de valeur des attributs : désigner-le type de valeur pouvant être affectée à un attribut. Les plus typiques sont les suivants : chaîne de caractère, nombre ou enveloppe ou entier, booléen, énuméré (précise une liste de valeurs spécifiques autorisées pour l'attribut), instance (permet de définir des relations entre les concepts).

**Remarque :** ici « instance » est un type d'attribut offert par l'éditeur d'ontologies Protégé. Ce type indique que la propriété est une relation entre la classe comportant cet attribut et celle spécifiée avec l'attribut. Exemple : pour modéliser la relation « avoir_père » entre les deux classes « père » et « enfant », il suffit d'ajouter une propriété de type « instance » dans la classe « enfant » qui a le nom « avoir_père ». La classe cible est la classe père.

**Le nombre de valeur ou cardinalité :** désigne le nombre de valeurs qu'un attribut peut avoir. Une cardinalité peut être unique (autorise une seule valeur) ou multiple (autorise plusieurs valeurs). Il utile de

spécifier pour un attribut une cardinalité minimale et une cardinalité maximale.

**Le domaine et le rang d'un attribut :** l'étendue ou le rang d'un attribut représente les classes autorisées pour les attributs de type « Instance ». Le domaine d'un attribut représente les classes autorisées auxquelles cet attribut est rattaché ou les classes dont l'attribut décrit les propriétés.

**Etape 7 :** créer les instances.

Cette étape consiste à créer les instances qui représentent des entités réelles des classes. On commence par créer une instance individuelle de la classe choisie puis on la renseigne avec les valeurs des attributs.

| | **CYC [LENA 90]** | **Uschold & King[USCH 95]** | **Gruninger & Fox [GRUN 95]** | **KACTUS [BERN 96]** | **METHONTOLOGY [FERN 97]** | **SENSUS [SWAR 97]** | **On-To-Knowledge [GOME04]** |
|---|---|---|---|---|---|---|---|
| Proposition de cycle de vie | Prototype en évolution | Non proposé | Prototypes en évolution ou incrémentale | Prototypes en évolution | Prototypes en évolution | Non proposé | Incrémentale et cyclique avec prototypes en évolution |
| Stratégie par rapport à l'application | Indépendante d'application | Indépendant d'application | Semi-dépendante d'application | Dépendante d'application | Indépendante d'application | Semi-dépendante d'application | Dépendante d'application |
| Stratégie d'identification de concepts | Non spécifiée | Middle-out | Middle-out | Top-down | Middle-out | Non spécifiée | Top –down-Buttom-up Middle-out |
| Utilisation d'ontologie noyau | Oui | Non | Non | Non | Dépend des ressources disponibles | Oui | Dépend des ressources disponibles |
| Outil support | Outil Cyc | Pas d'outil spécifique | Pas d'outil spécifique | Pas d'outil spécifique | ODE-WebODE-OntoEdit, Protégé-2000 | Pas d'outil spécifique (OntoSaurus) | OntoEdit avec ses plug-ins |

**Tableau 3.1 :** Les méthodes d'ingénierie ontologique

### 3.3.5 Langages de représentation des ontologies

L'une des décisions importantes à prendre dans le processus de développement des ontologies, est de sélectionner le langage dans lequel l'ontologie sera implémentée, dans les dernières années, plusieurs langages de représentations d'ontologies ont été développés
[GOMEZ PEREZ et al., 04], et autres langages et systèmes de Représentation de la Connaissance utilisés pour implémenter les ontologies alors qu'ils n'étaient pas spécifiquement créés pour ce but.

D'habitude, le choix d'un langage d'ontologie n'est pas basé sur la représentation de la connaissance et sur les mécanismes d'inférence exigés par l'application qui utilise l'ontologie, mais sur les préférences du développeur.
Notre expérience nous indique qu'un mauvais choix du langage utilisé pour implémenter une ontologie peut causer des problèmes une fois l'ontologie utilisée dans une application.
Comme indiqué dans [GOMEZ PEREZ et al., 04], des réponses plus ou moins claires aux questions suivantes permettent d'aider au choix d'un langage approprié :

- Quel(s) langage(s) devrait-on utiliser pour implémenter notre ontologie?
- Quel caractère expressif le langage doit-il avoir ?
- Quels sont les mécanismes d'inférence attachés au langage ?
- Le langage est-il soutenu par un outil de développement d'ontologie?
- Le langage est-il approprié pour échanger des ontologies entre les applications ?
- Le langage facilite-t-il l'intégration de l'ontologie dans une application ?

-Le langage est-il intégré dans d'autres langages utilisés pour représenter la connaissance et l'information sur le Web, tels que le HTML et le XML ?
- Y a-t-il des traducteurs qui traduisent l'ontologie implémentée dans un langage source vers un langage cible ?
- Comment de tels traducteurs réduisent-ils au minimum la perte de la connaissance dans le processus de traduction ?
Tous les langages existants n'ont pas le même caractère expressif et ne raisonnent pas de la même manière. Les langages d'ontologie sont basés sur des différents paradigmes de représentation de la connaissance comme des classes (ou "frames"), la logique de description, la logique de premier (et second ordre), et les réseaux sémantiques, etc... Cet état de fait rend bien plus important le choix correct du langage dans lequel l'ontologie doit être implémentée. En fait, un paradigme de représentation de connaissance peut s'avérer très approprié pour une tâche spécifique mais pas pour d'autres effectuées dans une application.
Par conséquent, avant le codage de l'ontologie nous devrions savoir d'abord ce dont nous avons besoin, en termes d'expressivité et de raisonnement, et puis les langages qui satisfont à de telles exigences.
Cette tâche d'analyse et de comparaison des langages d'ontologie, par rapport à leur expressivité et leurs possibilités de raisonnement, est décrite dans [CORC, 00].
Dans cette partie, nous voulons présenter un bref historique sur l'évolution des langages d'ontologies et les mécanismes d'inférences attachés à ces langages :

### 3.3.5.1 Evolution des langages d'ontologies

Au début des années 90, un ensemble de langages d'ontologies basées Intelligence Artificielle était créé, nous trouvons la logique du premier ordre (exemple KIF), les Frames combinés avec la logique du premier

ordre (exemple Cycl, Ontolingua,OCML, Flogic), et la logique de description (exemple LOOM), OKBC (Open Knoweldge Base Connectivity) [CHAU, 98] a été également crée comme un protocole pour accéder à des ontologies implémentées dans différents langages avec le paradigme de représentation de connaissances basé frame, la disposition globale de ces langages est présenté dans la figure suivante :

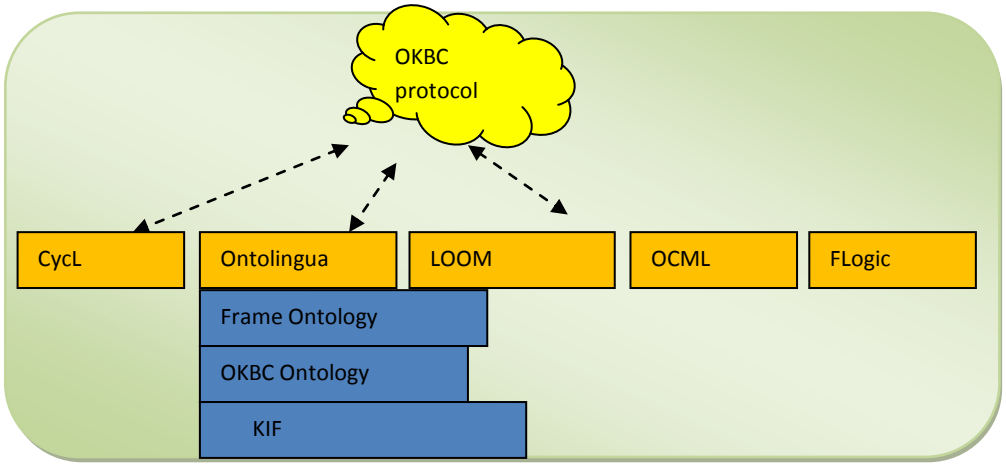



**Figure 3.10 :** Langages traditionnels d'ontologies [GOMEZ PEREZ et al., 04]

Cette catégorie représente un ensemble de langages d'ontologie basés sur l'Intelligence Artificielle et créés la plupart dans les années 1990. Nous pouvons citer entre autres :

- Cycl 1990 [LENAT et GUHA, 90] : est le premier langage qui a été développé, il est basé sur les Frames et la logique du premier ordre ; Et utilisé pour la construction de l'ontologie Cyc.
- KIF 1992 [GENES et al., 92] : il a été conçu comme un format d'échange

de la connaissance, KIF est basé sur la logique du premier ordre.

- LOOM 1991 [MACGREGOR, 91] : ce langage n'est pas conçu pour le développement des ontologies mais pour les bases de

connaissances, il est basé sur la logique de description et les règles de productions, Ce langage fournit des dispositifs de classification automatique des concepts.

OCML ("Operational Conceptual Modeling Language") [MOTTA, 99]: il a été conçu comme une extension du langage Ontolingua, une sorte de "Ontolingua opérationnel", parce que la plupart des définitions qui peuvent être exprimées dans OCML sont similaires aux définitions correspondantes dans Ontolingua. Ce langage a été construit pour développer des ontologies exécutables et des modèles dans les méthodes de résolution de problème.

- F-Logic 1995 [KIFER et al., 95] : combine aussi bien les frames que la logique de premier ordre. Il permet de représenter des connaissances avec les éléments suivants: concepts, taxonomies, relations, axiomes et règles de déduction. Il possède un moteur d'inférence, Ontobroker, qui peut être utilisé pour dériver de nouvelles connaissances.

**Les langages d'ontologie dits "web-standard" ou langages basés sur XML**

L'explosion des technologies d'Internet a mené à la création des langages pour l'exploitation des caractéristiques du Web, ces langages sont appelés généralement les langages basés Web ou les langages d'annotation d'ontologies, leur syntaxe est basée sur l'existence d'annotation comme HTML et XML, dans le but n'est pas le développement d'ontologies mais la représentation et l'échange des données [GOMEZ PEREZ et al., 04], la relation entre ces langages est représentée dans la figure suivante :

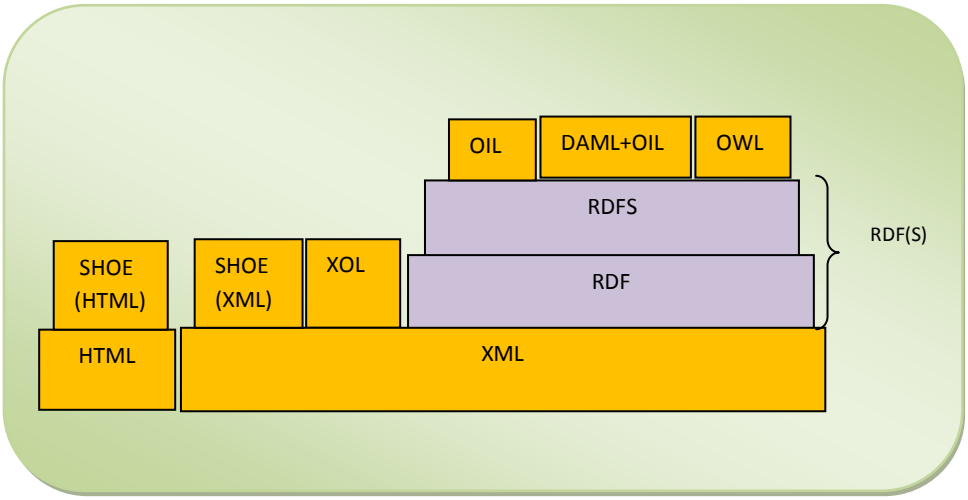



**Figure 3.11 :** Langages d'annotation d'ontologies selon Gomez Perez [GOMEZ PEREZ et al., 04]

- SHOE [HEFLEIN et al., 03] (Simple HTML Ontology Extensions) : est le premier langage d'annotation d'ontologies. Il a été créé en 1996 comme une extension de HTML. Il utilise des balises particulières qui permettent d'insérer des ontologies dans des documents HTML. Ce langage est à base de frames et les règles de production ce qui lui permet de représenter des concepts, des taxonomies, des relations, et aussi des règles. Ces dernières permettent d'inférer de nouvelles connaissances.
- XML [BRAY et al., 00] en 1998, qui a été très vite adopté comme un standard pour les échanges d'informations sur le Web par le W3C (World Wide Web Consortium), SHOE a été modifié de telle sorte qu'il puisse supporter des documents structurés décrits en XML. D'autres langages ont par la suite été créés sur la base de la syntaxe de XML.
- XOL (XML Ontology Language) : a été développé en 1999 comme un langage basé XML qui permet de spécifier des concepts, des taxonomies et des relations binaires. Bien qu'il ne permette pas d'effectuer des raisonnements, ce langage a été longtemps utilisé dans les échanges d'ontologies dans le domaine biomédical.

- RDF (Ressource Description Framework) [DAN BRICHEL et al., 04] : il a été développé par W3C comme un langage basé sur les réseaux sémantiques pour décrire les ressources du Web. Afin de renforcer ce langage, RDF Schema [DAN BRICHEL et al., 04] a été construit par W3C pour comme extension de RDF comportant des primitives basées sur des frames. RDF Schema permet notamment de déclarer les propriétés des ressources ainsi que le type des ressources. La combinaison de RDF et RDF Schema est connue sous le nom RDF(S). Bien que relativement limités dans la mesure où ils ne sont pas très expressifs, les langages RDF(S) peuvent cependant spécifier des concepts, des taxonomies et des relations binaires. De plus, certaines règles existent pour notamment définir des contraintes sur les recherches.

- DAML 2002 : Le langage DAML (*DARPA Agent Markup Language* [DAML, 02] est développé comme une extension du XML et du RDF et il permettra de créer des ontologies et de baliser l'information pour la rendre lisible par des machines. Un des objectifs du programme DAML est de mettre en place des technologies permettant aux agents d'identifier et comprendre dynamiquement des sources d'information, et de communiquer au niveau sémantique.

- OIL (Ontology Inference Layer) [FENSEL et al., 00] : basé sur RDF(S), le langage OIL permet 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 des ontologies sur le Web. Le langage OIL est limité du point de vue expressivité, puisque par exemple il ne supporte pas les types concrets, restriction qui se justifie par ailleurs par la volonté de pouvoir conserver un test de subsomption décidable.

- DAML+OIL [DAN et al., 01]: est la combinaison de ces deux langages. Il hérite des avantages de ces deux langages. En conséquence, DAML+OIL est un langage très expressif et lisible par

la machine ainsi que par un être humain avec une syntaxe basée sur RDF.

- OWL (Web Ontology Language): il est défini par le W3C dans il a été construit sur DAML+OIL. OWL est développé pour pallier les lacunes de DAML+OIL, et plus précisément pour être utilisé dans des situations où les informations contenues dans les documents Web doivent être traitées par des applications logicielles, par opposition aux situations dans lesquelles le contenu est simplement présenté aux humains, cas traditionnel de l'utilisation du Web de nos jours. Le langage OWL est basé sur la recherche effectuée dans le domaine de la logique de description. OWL permet de décrire des ontologies, c'est-à-dire qu'il permet de définir des terminologies pour décrire des domaines concrets. Une terminologie se constitue de concepts et de propriétés (aussi appelés rôles en logiques de description).

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

• OWL Lite : supporte les utilisateurs ayant besoin principalement d'une hiérarchie de classification et des contraintes simples de cardinalités 0 ou 1. Cette cardinalité correspond à des relations fonctionnelles, par exemple une personne a une adresse. Toutefois, cette personne peut avoir un ou plusieurs prénoms. OWL Lite ne suffit donc pas pour cette situation.

• OWL DL : D'après son nom OWL DL utilise la LD [BAADER et al ., 03]. Il supporte les utilisateurs qui réclament l'expressivité

maximale tout en retenant la complétude informatique (toutes les conclusions sont garanties d'être calculables), et la possibilité de décision (les calculs finiront en un temps fini). Il inclut toutes les constructions du langage OWL, qui ne peuvent être utilisées que sous certaines restrictions.

- OWL Full : a été défini pour les utilisateurs qui veulent une expressivité maximale et une liberté syntaxique de RDF sans des garanties informatiques. OWL Full permet à une ontologie d'augmenter la signification du vocabulaire prédéfini (RDF ou OWL). L'inconvénient majeur d'OWL Full est d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie.

Il existe certes d'autres langages de représentations d'ontologies, mais OWL est celui qui tend à s'imposer aujourd'hui. En outre, c'est un *standard* du W3C. C'est-à-dire qu'à défaut d'être une norme ISO, il en est l'équivalent dans le monde des industriels.

Par ailleurs, le choix du langage est fondé principalement sur des critères techniques, notamment en fonction des degrés d'expressivité et de formalisme voulus. OWL répond parfaitement à ces critères et c'est ce que nous recherchons pour la réalisation de la mission, néanmoins il serait intéressant d'étudier brièvement un autre langage qui constitue une alternative à OWL dans le cas d'un formalisme moins fort des connaissances : SKOS.

SKOS : le projet de SKOS (*Simple Knowledge Organisation System*) a été initié par l'Union européenne dans le cadre du projet SWAD-Europe.

Ces travaux ont abouti en 2003 aux premières publications de *SKOS Core Guide* et *SKOS Mapping Guide*, ainsi qu'à une mise en application destinée à valider les technologies retenues en situation réelle. La réflexion a ensuite été reprise par le W3C dans le cadre du

groupe de travail sur les bonnes pratiques et le déploiement des standards RDF.

Il s'agit d'un langage de représentation de schémas de concepts. Comme son nom l'indique, il est destiné à proposer un système permettant d'exprimer et de gérer des modèles interprétables par des machines dans la perspective du web sémantique. On le dit « simple » par opposition notamment au langage OWL, qui comme nous l'avons vu est plus à même de représenter des structures plus riches : les ontologies. En effet, SKOS est avant tout destiné à la représentation de thésaurus, classifications, taxinomies ou autres folksonomies.

Son formalisme de représentation repose sur les graphes RDF. Le concept constitue le centre du graphe auquel peuvent notamment être attachés en tant que propriétés RDF:

- **les indications portant sur le concept lui-même :** des termes préférentiels ou alternatifs, les équivalents dans d'autres langues ; des termes cachés ; et la représentation par une image
- **les différents types de notes :** notes de définition et d'application, exemples, notes historiques, etc.
- **les relations sémantiques :** hiérarchie et association

Le langage du projet OBO

Le projet OBO (Open Biomedical Ontologies) est une initiative d'un groupe de développeurs d'ontologies dans le domaine biomédical, qui s'est mis d'accord sur un nombre de principes spécifiant les bonnes pratiques pour le développement d'ontologies biomédicales. Les principes édictés reposent sur l'objectif d'interopérabilité entre les différentes ontologies développées. Un langage formel commun est fourni pour la représentation des ontologies. Il est conçu pour permettre la prise en compte de plusieurs méta-données, et comprend un mécanisme d'historisation.

Parmi les principes édictés par le projet OBO citons :

1. chaque ontologie doit avoir un identifiant unique au sein de OBO;
2. chaque ontologie doit inclure des définitions textuelles pour chacun de ses termes ;
3. les ontologies au sein d'OBO doivent être développées de façon collaborative.

Comme toutes les ontologies décrites à l'aide des langages à base de DLs, tels que OWL, une ontologie OBO comprend des classes, des propriétés (ou types de relations), des instances. La notion de classe est similaire à celle qu'on retrouve en OWL et DAML+OIL. Il existe deux niveaux de "Propriétés" : les propriétés à l'échelle des classes et celles à l'échelle des instances.

Dans le tableau 3.2 nous présentons une synthèse des trois couches de OWL.

| | OWL Full | OWL DL | OWL Lite |
|---|---|---|---|
| **Syntaxe** | - Union de la syntaxe de OWL et de RDF. | - Des fragments de logique de premier ordre (le ¼ de DAML+OIL). | - Un sous-ensemble simpliste de OWL-DL. |
| **Propriétés** | - Aucune restriction sur le vocabulaire de OWL ;<br>- Basé sur le modèle théorique de RDF ;<br>- Les inférences sont gérées par des raisonneurs de logique de premier ordre ;<br>- Les sémantiques ne remettent pas en cause celles de OWL-DL. | - Restreint l'utilisation du vocabulaire OWL (ex. un élément ne peut être à la fois classe et instance) ;<br>- Défini par une syntaxe abstraite et un mapping à RDF ;<br>- Basé sur le modèle théorique standard des DL et des logiques de premier ordre avec une sémantique « définitive » ;<br>- Bénéficie des acquis des DL (une sémantique bien définie, des propriétés formelles compréhensibles en termes de complexité et de décidabilité, des algorithmes d'inférence connus et des raisonneurs DL existants). | - Un sous-ensemble des constructeurs de OWL-DL (pas de négation ou d'union explicites, cardinalité restreinte à 0 ou1, pas de nominaux OneOf) ;<br>- Sémantique basée sur les DL, raisonnement à travers les moteurs standards de DL ;<br>- Pas très utilisé en pratique. |
| **Expressivité** | - Expressivité complète mais décidabilité non garantie. | - Expressivité incomplète mais décidable. | - Expressivité minimale facilitant la traçabilité des inférences. |

**Tableau 3.2:** Présentation synthétisée des trois couches de OWL[JEDIDI,09].

### 3.3.6 Formalismes de représentation

- **Frames**

Les schémas, connus beaucoup plus sous son terme anglais «Frame», sont apparue en 1932 dans les études psychologiques. Plus tard, les schémas ont été introduits en intelligence artificielle par Minsky afin de résoudre les problèmes de la vision par ordinateur.

Parmi les langages de représentation des connaissances à base de frame KRL (1977) et KLOne [BRACH et al., 85]. Les frames ont été définis par MINSKY [MINSKY, 75] C'est un modèle classique de l'intelligence artificielle et a été initialement proposé comme langage de représentation d'ontologies par [GRUBER, 93].

Le principe de ce modèle est de décomposer les connaissances en classes (ou frames) qui représentent les concepts du domaine. À un frame est rattaché un certain nombre d'attributs (slots), chaque attribut peut prendre ses valeurs parmi un ensemble de facettes (facets) [KIFER et al., 95]. Une autre façon de présenter ces attributs est de les considérer comme des relations binaires entre classes dont le premier argument est appelé domaine (domain) et la deuxième portée (range) [SOWA, 84]. Des instances des classes, correspondant à l'extension de chaque concept, peuvent être ajoutées, ainsi que des fonctions qui sont des types particuliers de relations liant un ensemble de classes à une valeur calculée à partir des valeurs des attributs des classes. La spécification de propriétés conceptuelles des attributs (ou relations) recourt à des formules de la logique du premier ordre.

- **Graphes conceptuels**

Introduit par J. SOWA au début des années 80 [SOWA, 84], le modèle des graphes conceptuels appartient à la famille des réseaux sémantiques [BRACH et al., 85]. Les réseaux sémantiques modélisent les connaissances sous forme de graphes, les nœuds

étant associés à des concepts et les arêtes à des relations. Ce modèle se prête bien à des présentations graphiques des connaissances mais présente, comme nous allons le voir, certains problèmes particuliers de représentation.

Le modèle des graphes conceptuels se décompose en deux (2) parties :

- Une partie terminologique dédiée au vocabulaire conceptuel des connaissances à représenter, c'est-à-dire les types de concepts, les types de relations et les instances des types de concepts. Cette partie correspond à la représentation du modèle conceptuel mais intègre également des connaissances sur la hiérarchisation des types de concepts et de relations.
- Une partie assertionnelle dédiée à la représentation des assertions du domaine de connaissance étudié.

- **Logiques de description**

La LD (Logiques de Description) découle directement des travaux fondateurs de Bachmann et de son système KL-ONE [NAPOLI, 97].

Depuis le début des années 90, la recherche en LD s'est considérablement développée.

La LD est une logique basée sur un formalisme de représentation des connaissances, elle s'apparente à la logique du premier ordre. Elle contient deux catégories d'éléments soit les concepts et les rôles.

Les concepts (qui peuvent être des classes) sont considérés comme des ensembles d'objet. Les rôles sont des relations binaires sur les objets.

La logique de description permet par l'utilisation de différents constructeurs de créer des concepts complexes à partir de concepts plus simples.

La LD combine des représentations intentionnelles et extensionnelles des connaissances [BAADER et al., 03]:

- Une TBox (T pour terminologique) contient les déclarations des primitives conceptuelles organisées en concepts et rôles (relations binaires entre concepts dotées d'un domaine et d'une portée similaires aux attributs des frames). Ces déclarations décrivent les propriétés des concepts et des rôles et constituent donc une définition intentionnelle des connaissances ;
- Une ABox (A pour assertionnel) contient les déclarations d'individus, instances des concepts définis dans la TBox.

Une des particularités importantes de la LD est qu'elle sépare la connaissance à propos des descriptions ou des concepts (appelée T-Box) de la connaissance sur les assertions où instances (appelée A-Box). Ce qui signifie que les instances définies dans la A-Box est basé sur des descriptions dans la T-Box.

### 3.4 Discussions

Un formalisme de représentation d'ontologie peut être jugé selon deux dimensions principales, qui sont fortement liées entre elles: la représentation de connaissances et les mécanismes de raisonnement.
A la différence des graphes conceptuels et des langages des frames, les logiques de descriptions sont caractérisées par les points suivants:
• Les LDs sont bien adaptées à la représentation d'ontologies complexes car elles disposent d'une sémantique claire, un haut pouvoir de description de concepts complexes, par exemple grâce aux restrictions de valeurs, la négation et la conjonction (disjonction) de concepts.
D'autre part, elles procurent des mécanismes d'inférences efficaces telles que la subsomption et la satisfiabilité des concepts.

• Les LDs ont l'avantage de faire apparaître explicitement les notions d'identité (individus), de structure (rôles), de classes (concepts), de sous-typage (subsomption) et d'expressivité (choix des constructeurs de concepts).

• Les LDs sont le type de formalisme retenu. La représentation des connaissances à l'aide de la famille SG est entièrement graphique. Cet aspect graphique nous semble important car il permet une interprétation intuitive des connaissances d'ingénierie tout en étant proche des langages de modélisation habituellement utilisés en IS.

Les raisonnements de la famille de SG sont basés sur des opérations de graphe. Ils s'appuient directement sur les connaissances représentées, sans passer par un langage logique. Ceci fournit potentiellement des possibilités d'explication des raisonnements à l'utilisateur puisque les raisonnements peuvent être visualisés dans le formalisme qu'il connaît et directement sur les connaissances qu'il a définies. Les raisonnements sont consistants et complets par rapport à la sémantique formelle en logique de prédicats, logique des connaissances représentées.

Il permet l'expression de connaissances variées : des descriptions, des patrons, des règles, des contraintes [BAGET et MUGNIER, 02].

Il permet une structuration et une contextualisation des connaissances grâce aux emboîtements de graphes. Dans le tableau 3.3 une comparaison entre les deux formalismes est présentée selon [CHOURAB, 09].

| **Opérationnalisation de la proposition de CMS (Système de gestion de connaissances métier)** | **Formalisme des GC et son extension avec la famille des SG du laboratoire LIRMM** | **Formalisme LD** |
|---|---|---|
| Fondements mathématiques | Les Graphes Existentiels de Charles PIERCE | L'Algèbre des relations de PEIRCE |
| Lisibilité | + | - |
| Expressivité | + (grâce aux extensions de la famille des SG) | ++ |
| Niveau Terminologique | Support de graphes conceptuels | Tbox |
| | Mécanisme de définitions de type [Badget, 2001] | Concepts et roles derives |
| | Constructeurs de type de concepts par négation et conjonction | Plus de constructeurs de concepts selon la LD choisie |
| | | |
| Niveau axiomatique | Règles et contraintes des SG | Pas de règles en LD |
| Niveau assertionnel | Graphes conceptuels simples<br>Graphes conceptuels emboités | ABOX |

**Tableau 3.3 :** Etude comparative des formalismes GC et DL

Le tableau 3.3 est une étude comparative des formalismes GC et DL pour l'opérationnalisation du système de gestion de connaissances métiers situées en ingénierie système [CHOURAB, 09].

## 3.5 Les outils de construction d'ontologie

On distingue deux familles d'outils : les outils de construction d'ontologie dépendants de formalisme de représentation et les outils

de construction d'ontologie indépendants de formalisme de représentation.

**Ontolingua :** Ontolingua est un serveur d'édition d'ontologies [FARQ et al., 00]. Il utilise des classes, des relations, des fonctions, des objets (instances) et des axiomes pour décrire une ontologie. Une relation (ou une classe) peut contenir des propriétés nécessaires (contraintes) ou nécessaires et suffisantes qui définissent la relation (ou la classe).En plus de ces fonctionnalités offertes par le langage Ontolingua, le serveur Ontolingua offre la possibilité d'intégrer les ontologies Ontolingua, ce qui permet la construction modulaire des ontologies.

Il ya trois possibilités d'intégrer les ontologies Ontolingua, ce qui permet la construction modulaire des ontologies.

Il ya trois différentes possibilités d'intégrer les ontologies [FARQ et al., 00].

Inclusion : Une ontologie inclut et utilise les définitions d'autres ontologies.

Restriction : l'ontologie importe les définitions depuis d'autres ontologies et les rend plus spécifiques.

Raffinement polymorphe : on redéfinit une définition importée depuis n'importe quelle ontologie.

**OntoSaurus :** OntoSaurus de l'Information Science Institute de l'Université de Southern California est composé de deux modules : un serveur utilisant LOOM comme langage de représentation des connaissances, et un serveur de navigation créant dynamiquement des pages HTML qui affichent la hiérarchie de l'ontologie; le serveur utilise des formulaires HTML pour permettre à l'usager d'éditer l'ontologie. Il utilise LOOM comme langage de représentation des

connaissances. On peut représenter les concepts, la taxonomie des concepts, les relations entre les concepts, les fonctions, les axiomes et les instances.

**WebOnto** [DOMINGUE, 98]: WebOnto du Knowledge Media Institute de l'Open University, est une application Web pour naviguer et développer collaborativement les ontologies. Il supporte la navigation collaborative, la création et l'édition d'ontologies sur le Web. Les ontologies WebOnto sont implémentées dans le langage OCML. Le langage OCML est une combinaison des frames et de la logique de premier ordre et permet de représenter les concepts, la taxinomie des concepts, les relations, les fonctions, les axiomes et les instances. WebOnto distingue quatre types d'ontologies : ontologie de domaine, ontologie de tâche, ontologie de méthode, et ontologie d'application. Il supporte l'inclusion d'ontologie au moyen des interfaces graphiques. En ce qui concerne l'édition collaborative d'ontologie, WebOnto est le seul outil qui procure cette fonctionnalité, il permet aux ingénieurs des connaissances de tenir des discussions sur les changements et les mises à jour des ontologies lors d'édition ou de navigation, en mode synchrone et asynchrone.

**OilEd** [BLAZ et al., 99]**:** OilEd (Oil Editor) est un éditeur d'ontologies développé à l'Université de Manchester, spécialement pour l'édition d'ontologies OIL et DAML+OIL, basées sur le formalisme des DLs. L'interface principale de OILEd est composé d'un ensemble de tables fournissant différentes options pour éditer les différents composants d'une ontologie DAML+OIL : classes, propriétés, individus (instances), axiomes, espaces de nommage.
Le classement des concepts et des instances s'effectue en utilisant le moteur d'inférence par défaut, FaCT [HORROCKS, 98]. Pour cela

l'ontologie DAML+OIL est transformée dans le formalisme des DL SHIQ [HORROCKS et SATT, 03].

OILEd n'est pas un environnement complet d'édition d'ontologies. Ainsi, il n'est pas destiné à l'édition de grandes ontologies, ni à l'intégration ou la gestion de plusieurs versions d'une ontologie. Il est essentiellement dédié à la construction de petites ontologies dont on peut ensuite tester la cohérence à l'aide de FACT, un moteur d'inférences bâti sur OIL.

**Protégé 2000** [GENNARI et al., 03]: Protégé 2000 est un éditeur d'ontologies, distribué en open source par l'institut d'informatique médicale de Stanford. C'est un éditeur hautement extensible, capable de manipuler des formats très divers. Il existe deux moyens pour modéliser une ontologie avec PROTEGE, PROTEGE-Frame et PROTEGE-OWL. Une ontologie en PROTEGE peut être exportée dans différent formats incluant RDF(s), OWL, XML schémas. PROTEGE est une plateforme Java, il est flexible et supporte plusieurs langues dont l'Anglais, le Français, l'Arabe, le Chinois le Russe etc. Une large communauté de développeurs académiques, de gouvernements et d'entreprises utilise PROTEGE dans divers domaines. L'interface permet de créer, supprimer, modifier et mettre à jour les concepts, les propriétés, les instances et les relations.

Il autorise la définition de méta-classes, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie.

**Jena** [MCBRIDE, 01] :

JENA est un environnement de travail open source en Java, pour la construction d'application web sémantique. JENA permet de manipuler des documents RDF, RDFS, OWL et SPARQL. Il fournit un moteur d'inférences permettant des raisonnements sur les ontologies. JENA est maintenant sous Apache Software Licence.

**ODE et WebOde** [ARPI et al., 01]. L'outil ODE (Ontology Design Environment) permet de construire des ontologies au niveau connaissance, comme le préconise la méthodologie METHONTOLOGY. L'utilisateur construit son ontologie dans un modèle de type frame, en spécifiant les concepts du domaine, les termes associés, les attributs et leurs valeurs, les relations de subsomption. L'ontologie opérationnelle est alors générée en utilisant les formalismes ONTOLINGUA ou FLOGIC. WEBODE est l'adaptation de l'ODE pour le Web.

**OntoEdit:** [SURE et al., 02] OntoEdit (Ontology Editor) est également un environnement de construction d'ontologies indépendant de tout formalisme. Il permet l'édition des hiérarchies de concepts et de relations et l'expression d'axiomes algébriques portant sur les relations, et de propriétés telles que la généricité d'un concept. Des outils graphiques dédiés à la visualisation d'ontologies sont inclus dans l'environnement. OntoEdit intègre un serveur destiné à l'édition d'une ontologie par plusieurs utilisateurs. Un contrôle de la cohérence de l'ontologie est assuré à travers la gestion des ordres d'édition. Enfin, un plug-in nommé ONTOKICK offre la possibilité de générer les spécifications de l'ontologie par l'intermédiaire de questions de compétences.

**KAON**

KAON (KArlsruhe ONtology) est une infrastructure d'ontologie, développée en 2002, par l'université de Karlsruhe et le centre de recherche d'information et de technologie à Karlsruhe. KAON est utilisé dans des projets de recherche qui s'intéressent particulièrement à la réalisation et à l'évolution des ontologies [AZOUAOU et al., 05]

### 3.5.1 Comparaison environnement et outils d'IO

Un ensemble d'environnements d'IO ont été développés afin de systématiser l'IO. Ces environnements offrent des services de construction, d'extraction, d'alignement, d'annotation, de raisonnement et d'évaluation d'ontologies [GÓMEZ-PÉREZ et al., 03]. Nous présentons un échantillon d'environnements d'IO, ayant fait l'objet d'une étude comparative dans [FURST, 04].

| | Oiled | OntoEdit | OntoLingua | Protégé | WebODE | WebOnto | DOE | TooCoM |
|---|---|---|---|---|---|---|---|---|
| Modèles de connaissances | DL (DAML+OIL) | Frames+FOL | Frames+FOL | Frames +FOL | Frames +FOL | Frames +FOL | Entité/Relation | Entité/Relation Syntaxe GC |
| Langage d'axiomes | Oui (DAML+OIL) | Oui (FLogic) | Oui (KIF) | Oui (PAL) | Oui (WAB) | Oui (OCML) | Non | Oui Syntaxe GC |
| Support méthodologique | Non | Oui | Non | Non | Oui | Non | Oui | Non |
| Moteur d'inférence | Fact (interne) | OntoBroker (interne) | ATP (externe) | PAL (interne) | Prolog (interne) | Oui (interne) | Non | Cogitant (externe) |
| Tests de coherence | Oui | Oui | Non | Oui | Oui | Oui | Non | Oui |
| Hiérarchies graphiques | Non | Non | Oui | Oui | Oui | Oui | Oui | Oui |

**Tableau 3.4 :** Etude comparative des outils d'IO [FURST, 2004].

**Modèle de connaissances** : paradigme et langage utilisés dans l'outil pour représenter les ontologies.

**Langages d'axiomes** : Langage utilisé dans l'outil pour représenter les axiomes des ontologies.

**Support méthodologique** : guide méthodologique pour la construction et la structuration des ontologies.

**Moteur d'inférence** : il peut être spécifique à l'outil (interne) ou développé à part (externe).

**Test de cohérence** : fonctionnalité permettant d'évaluer la cohérence des ontologies.

**Hiérarchies graphiques** : interface graphique de restitution et de manipulation des hiérarchies de concepts et de relations.

## 3.6 Conclusion

L'exposé des méthodes ci-dessus permet de distinguer deux grandes phases:

Une modélisation pour donner du sens, autrement dit, une modélisation des connaissances ontologiques conduisant à la définition d'une ontologie conceptuelle. Une modélisation pour implémenter un système conduisant à une ontologie computationnelle. D'autres auteurs proposent un certain nombre de critères et d'étapes pertinents pour la construction d'une ontologie dynamique, interopérable, facilement maintenable et indépendante du contexte. Il n'y a pas une seule et unique manière de modéliser un domaine de connaissances, il n'y a que des alternatives plus ou moins réussies. La plupart du temps, le choix d'une méthodologie adéquate dépend des objectifs et des buts poursuivis ainsi que de l'outil de construction utilisé.

A travers ce que nous avons présenté dans ce chapitre, il ressort que la notion d'ontologie constitue l'une des approches les plus efficaces pour représenter les connaissances. Nous avons montré l'apport des ontologies pour la représentation de la sémantique et assurer l'interopérabilité des systèmes. De plus, nous avons exploré les principaux langages, méthodes et outils pour la construction des ontologies.

Dans le cadre de notre mémoire, les ontologies utilisées sont décrites dans le langage OWL. Le chapitre qui suit s'articule autour de la modularisation des ontologies.

# Chapitre 4
# Modularisation des ontologies :
## Etat de l'art

*De même, comment pourra-t-on comprendre de quoi vous parlez si le message que vous exprimez au moyen de langues inconnues n'est pas clair ? Vous parlerez pour le vent ! Il y a bien des langues différentes dans le monde, mais aucune d'entre elles n'est dépourvue de sens. Cependant, si je ne connais pas la langue dans laquelle on s'adresse à moi, celui qui parle cette langue sera un étranger pour moi, et je serai un étranger pour lui.*

*I Corinthiens 14, 9-11*

## 4.1 Introduction

Ces dernières années de nombreux progrès on été réalisés pour améliorer la puissance de calcul grâce notamment à l'outil informatique. Mais cela suffit-il ? Face à la quantité de données et de connaissances accumulées, il est de plus en plus difficile de la gérer et surtout de la faire partager et de l'échanger. Une des solutions proposées est d'utiliser les ontologies.

Le web sémantique est en pleine expansion. Des ontologies de domaine se développent à toute vitesse surtout dans le cadre de la recherche. Même si le web sémantique est assez jeune et immature, c'est un outil puissant permettant aux agents informatiques de traiter l'information par un langage qui leur est compréhensible, OWL.

Le Web sémantique tend non seulement à rendre l'information accessible mais aussi lisible et utilisable par des applications de traitement de données. Celui-ci offre des outils permettant de mieux organiser l'information et ce, en extrayant, partageant et réutilisant les connaissances spécifiques à un domaine. Dans cette optique de partage et d'interopérabilité, la prolifération et la disponibilité des ontologies sont cruciales. Il va sans dire que les ontologies sont devenues un modèle incontournable pour la représentation et le raisonnement sur des connaissances du domaine. Bien qu'essentielles à la gestion des systèmes à base de connaissances, il n'en demeure pas moins que la conception, la réutilisation et l'intégration des ontologies restent des tâches complexes [PATRICK, 14].

Les ontologies doivent leur succès actuel à leur aptitude à être partageables et réutilisables. Quoiqu'avec le partage, on se retrouve avec des ontologies de plus en plus larges. La construction d'une ontologie à partir de zéro est une tâche complexe qui nécessite du temps. Ceci nous amène donc à envisager la modularisation comme étant une approche qui apporterait des solutions non seulement au

problème de conception, mais aussi à celui du partage et de l'intégration des ontologies.

La notion de modularisation est basée sur le principe de «diviser pour mieux régner» généralement appliquée dans le génie logiciel où il est question de développer une application dont la structure repose essentiellement sur des composants autonomes facilement concevables et réutilisables.

Le chapitre présent est un état de l'art sur la modularisation des ontologies, les concepts de la modularisation et les travaux que nous jugeons les plus importants réalisés dans ce domaine ainsi que les objectifs de la modularisation des ontologies. Nous partons d'une étude générale sur la modularisation, ses objectifs [SETTI AHMED et BENSL, 12]. Ensuite, nous exposons les fameux travaux dans ce domaine et nous détaillons le domaine du partitionnement des ontologies à base de clustering dont notre projet de recherche fait partie.

## 4.2 Les objectifs de la modularisation des ontologies

La modularisation est un concept qui nous renvoie simultanément à deux aspects différents de l'ontologie à savoir : «l'ontologie comme un tout» et «l'ontologie comme un ensemble de modules». Les différentes approches de la modularisation et la façon dont elles sont mises en pratique dépendent en grande partie du but visé. Ainsi, le but de la modularisation est d'apporter des solutions aux problèmes suivants [PAR et SPAC, 09]:

**1) L'extensibilité en fonction des requêtes et du raisonnement sur les ontologies**

Les raisonneurs sont plus efficaces sur des ontologies de petite taille. La performance de ceux-ci décroît au fur et à mesure que la t aille de l'ontologie croît. Travailler avec des petites ontologies permet de pallier

à la perte de performance des moteurs d'inférences, et la modularisation apporterait une solution à ce problème, dans la mesure où elle fragmenterait une ontologie qui tendrait à gagner en taille en un ensemble d'ontologies plus petites. Ainsi, des requêtes peuvent être faites uniquement sur un module précis sans pour autant avoir à explorer l'ontologie en entier.

**2) L'extensibilité en fonction de la maintenance et de l'évolution des ontologies**

À ce niveau, le but de la modularisation serait de localiser et contenir l'impact que pourrait avoir la mise à jour d'une base de connaissances *(knowledge repository)* dans les limites des modules. L'implémentation d'une bonne distribution des connaissances nécessite, dans un premier temps, une bonne compréhension du principe de propagation de la mise à jour à l'intérieur même de l'ontologie et, en second lieu, une bonne connaissance de l'information contenue dans cette mise à jour.

**3) La complexité dans la gestion des ontologies**

L'on ne saurait garantir l'efficacité et la précision des bases de connaissances de grande taille élaborées par des humains, aussi bien en termes d'objets et de relations entre objets qu'en termes d'axiomes et de règles. Les ontologies qui contiennent des milliers de concepts ne peuvent être conçues et gérées par un seul expert. D'où la nécessité de ramener la conception des ontologies à la création de différents modules manipulables et de taille raisonnable, qui pourront par la suite être liés les uns aux autres.

**4) La compréhensibilité du contenu des ontologies**

Les utilisateurs doivent pouvoir être à même de comprendre le contenu d'une ontologie afin de pouvoir la manipuler. Que ce contenu

soit représenté sous une forme graphique ou textuelle, il sera plus aisé pour des humains de le lire si l'ontologie est de petite taille. Toutefois, la compréhensibilité ne dépend pas uniquement de la taille de l'ontologie mais aussi de la façon dont les objets, relations et règles sont agencés les uns par rapport aux autres.

**5) La réutilisation des ontologies**

D'un point de vu ontologique, la modularisation est considérée comme un moyen pour structurer et pour organiser des ontologies. Le module lui même est défini comme un sous-ensemble d'un tout qui a du sens, [DORAN et al., 07] définit le module de l'ontologie comme un composant réutilisable d'une ontologie plus complexe et affirment que le module de l'ontologie est autonome mais doit toujours entretenir des relations avec d'autres modules (y compris l'original).

La réutilisation est vue donc comme l'une des motivations premières de la modularisation. Elle permet de mettre l'accent sur les mécanismes de construction de modules ontologiques de telle sorte que ceux-ci puissent être réutilisés par la suite. Il va de soi que le contenu de ces modules se doit d'être pertinent, nécessaire et compréhensible afin que ces modules soient sélectionnés pour réutilisation.

Nous partageons le point de vue de Furst [FURST, 04] selon lequel les ontologies sont destinées à être réutilisées. La sémantique qu'elles représentent est liée au cadre applicatif à partir duquel le sens des termes et concepts est défini. Cependant, la représentation ne dépend pas de l'opération faite avec l'ontologie. La sémantique de l'ontologie est liée au contexte mais la représentation n'implique pas que l'ontologie soit utilisée uniquement dans le contexte de sa création.

## 4.3 Définitions

Avant de parler de modularisation, nous allons définir ce qu'est un module. Les relations qui peuvent exister entre les modules? Leurs propriétés ? [PINTO et al., 99]

### 4.3.1 Définition et description d'un module

Un module est un sous-ensemble d'une ontologie qui a un « sens », du point de vue des applications ou des utilisateurs. On ne construira pas un module en y incluant des classes de façon aléatoire. On peut supposer qu'une classe sans aucun lien avec les autres classes du module n'est pas désirable. Mais plus concrètement on peut définir un module ayant un sens si [PINTO et al., 00]: il est valide localement : si chaque information valide dans le fragment l'est encore dans la globalité de l'ontologie.

Il est localement complet : chaque information valide dans le domaine du fragment de l'ontologie globale l'est encore dans le fragment.

Un module devrait aussi être : petit de façon à raisonner plus vite sur le module.

Le plus indépendant que possible des autres modules. L'ajout ou le retrait d'un module n'affectera pas beaucoup les autres. On caractérise donc un module comme fermé si il ne possède aucun lien avec l'extérieur et d'ouvert si il contient des liens avec d'autres modules.

Compréhensible pour toute autre personne voulant le réutiliser

Il existe deux approches de création d'un module [PINTO et al., 99]:

**Composition** : les modules sont construits indépendamment les uns des autres. Ils sont ensuite assemblés pour former une ontologie plus large.

**Décomposition** : on crée un module en partitionnant une ontologie déjà existante. Le but de cette approche est de pouvoir y arriver de façon semi-automatique pour ensuite être validée par l'administrateur

de l'ontologie.

### 4.3.2 Critères de modularité

L'approche de composition est similaire à l'intégration d'une ontologie. Mais dans une ontologie modulaire, on peut solutionner le problème de la duplication d'information en liant les redondances d'un point de vue conceptuel par des inter-modules qui diminuent la taille des modules [BENOIT, 05].

Selon [PINTO et al., 99], ils considèrent que dans l'approche de décomposition on peut se baser sur des experts du domaine pour définir les modules, mais ce n'est pas suffisant, ce qui nécessite de déterminer aussi des critères de décomposition semi-automatiques qui seraient validés par un expert. On pourrait décomposer une ontologie en déterminant un noyau de classes devant faire partie du module, pour ensuite identifier et inclure les classes rattachées à ce noyau. Dans cette méthode il faut pouvoir analyser les requêtes auxquelles doivent répondre l'ontologie et sauvegarder le chemin utilisé pour y répondre. Une autre possibilité est de décomposer une ontologie basée sur des algorithmes de décomposition de graphe pour obtenir des sous -graphes.

Il faut tout de même faire attention dans l'approche de décomposition, de ne pas perdre d'informations. Une requête doit fournir le même résultat avant et après décomposition. Mais que se passe-t-il si la perte d'information est inévitable?

#### 4.3.2.1 Relations inter-modules

Si on considère les modules comme des sous-ontologies indépendantes, il est peu probable qu'elle puisse fournir une réponse complète à une requête. Il faut déterminer les modules nécessaires à la requête et ensuite fusionner et synchroniser les réponses partielles pour avoir une réponse globale. On considère deux cas [PINTO et al., 99]:

• Si les modules sont fermés. Les méta-données peuvent être centralisées dans un « repository », contenant les informations contenues dans le module et la façon dont récupérer ces données.

• Si les modules sont ouverts, les liens inter-module peuvent être **procéduraux,** ce type de lien peut être comparé à un mécanisme de vue. On extrait la connaissance d'un module sur base d'une requête. Les liens peuvent aussi être des **relations par assertion**, ils établissent un degré de ressemblance entre les composants de plusieurs modules et peuvent donc trouver plus d'éléments concernant une information dans un autre module [BENOIT, 05].

## 4.4 Les principales approches de modularisation des ontologies

Une ontologie *O* est définie par la formule suivante [PALMIZ et al., 09]: $O = (Ax(O), Sig(O))$ où $Ax(O)$ représente l'ensemble d'axiomes composé de concepts, de relations et de fonctions (sous-classe, équivalence, inst anciation, etc ...). $Sig(O)$ est la signature de *O* qui représente l'ensemble de noms des entités qui se retrouvent dans les axiomes. En d'autres termes, il s'agit du vocabulaire de *O*. La modularisation de l'ontologie *O* permet de définir un module $\mathcal{M}$ tel que : $\mathcal{M} = (Ax(\mathcal{M}), Sig(\mathcal{M})$ où $\mathcal{M}$ fait partir de *O*, avec $\mathcal{M} \subseteq (Ax(O))^I$ et $Sig(\mathcal{M}) \subseteq Sig(O)$.

L'on ne saurait définir l'ontologie sans pour autant introduire la notion d'interprétation *I* qui est une base de la sémantique de la logique descriptive. On la symbolise par $I = (\Delta^I, \bullet^I)$, où $\Delta^I$ est le domaine d'interprétation et $\bullet^I$ la fonction d'interprétation.

Il existe deux principales approches de modularisation des ontologies. Ainsi, distingue-t-on des approches basées sur le partitionnement de l'ontologie et des approches basées sur l'extraction de module

ontologique. Le partitionnement subdivise une ontologie en un ensemble de sous-structures autonomes ou dépendantes les unes des autres qu'on a appellera partitions, tandis que l'extraction de module consiste à extraire une sous-ontologie en fonction d'une signature précise [D'AQUIN et al., 09].

### 4.4.1 Les techniques de partitionnement des ontologies

Le partitionnement est un processus au cours duquel une ontologie *O* est fractionnée en un ensemble de modules (pas forcement disjoints) *M* tel que l'union de l'interprétation de toutes les partitions ainsi crées soit équivalente à l'interprétation de l'ontologie initiale *O.* Formellement, le partitionnement de l'ontologie peut être définie comme suit :

$M = \{\{M_1, M_2, M_3, \ldots Mn\} \mid \{(Ax(M_1))^I \cup (Ax(M_2))^I \cup \ldots \cup (Ax(M_n))^I\} = (Ax(O))^I\}$

#### 4.4.1.1 La technique de Stuckenschmidt et Klein

L'approche de Stuckenschmidt et Klein [STUCK et KLEIN, 04] consiste à partitionner automatiquement des ontologies basée sur la structure de la hiérarchie des classes (concepts). Elle est basée sur l'hypothèse que les dépendances entre les concepts peuvent être dérivées de la structure même de l'ontologie. Cette dernière peut être représentée sous forme d'un graphe pondéré 0 = (C, *D,* w) où les nœuds (C) représentent des concepts et les arêtes (D) représentent des relations entre concepts, dont le poids *(ω)* varie en fonction de la dépendance. Dans cette approche le partitionnement se fait en deux étapes :

1) Extraire de l'arbre de dépendance qui est en fait un sous-graphe de l'ontologie initiale.
2) Calculer le degré proportionnel de dépendance entre les concepts (Figure 4.1). Pour ce faire, on calculera le degré proportionnel de

dépendance *w(ci ,cj)* entre deux concepts ci et *cj,* avec *aij* qui est le poids assigné à la relation qui les unit :

$$w(Ci, Cj) = \frac{aij + aji}{\sum_k aik + aki}$$

où *aij* est le poids assigné à la relation entre les concepts ci et cj. Stuckenschmidt et Klein fixent la valeur de *aij* à 1 dans leurs expériences. Ainsi, le degré proportionnel de dépendance sera égal au rapport de *aij* par le nombre de concepts auquel le concept ci est connecté [STUCK et KLEIN, 04].

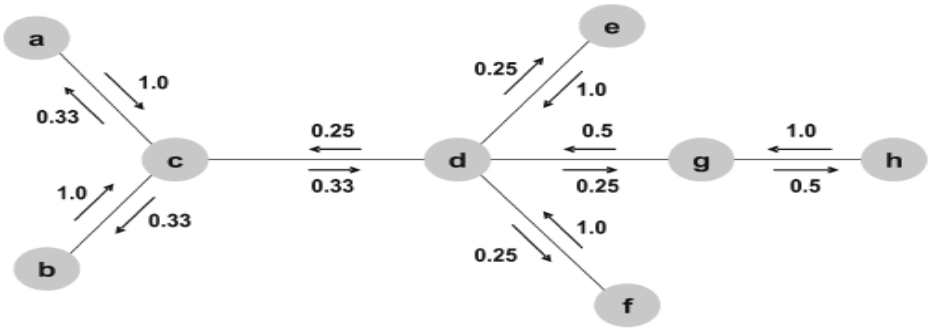



**Figure 4.1:** Un exemple de graphe avec des dépendances proportionnelles de force [STUCK et SCHL, 09]

La figure 4.1 fait état du cas où, le nœud d est utilisé dans le calcul des poids à l'aide des dépendances proportionnelles. Le nœud d a quatre voisins directs, ce qui entraine que le degré proportionnel de dépendance de la relation entre ce nœud et ses voisins est de 0.25 (la valeur *aij* qui est fixée à 1 par défaut divisée par quatre). On constate que différents niveaux de dépendances entre **d** et ses voisins proviennent de la dépendance de ces voisins là avec le nœud **d.** Toutefois, il est important de souligner que ce degré proportionnel de dépendances est non symétrique, c'est-à-dire qu'il sera différent respectivement pour les relations **d** → **g** et **g** → **d** Les nœuds e et **f** n'ont aucun voisins directs autre que **d** ou bien qui dépendent de

celui-ci. De ce fait, le poids de leur relation respective avec le nœud **d** demeurent égal à 1. Le degré proportionnel de dépendances entre les nœuds g et d est de 0.5, car g n'a que deux voisins. Quant à la dépendance entre les nœuds b et **d** est de 0.33 étant donne que **b** n'a que trois voisins [PATRICK, 14].

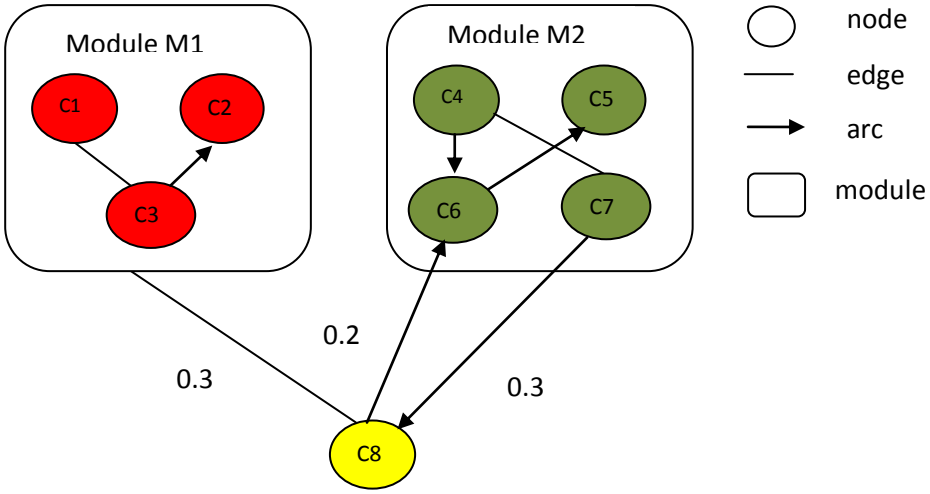



**Figure 4.2 :** Exemple de graphe pour l'attribution des nœuds restant à des modules [STUCK et SCHL, 09]

Afin de déterminer les partitions, on procède à une analyse du graphe, qui représente un réseau de dépendances proportionnelles dans lequel les nœuds fortement liés seront regroupés sous forme de "cluster" (Figure 4.2). En effet, ces derniers représentent des sous-graphes à l'intérieur desquels les arêtes sont liées plus fortement entre elles qu'à toutes autres arêtes voisines, qui sont en fait des arêtes les liant ainsi à des concepts ou sous-graphes voisins. La taille du sous graphe est un paramètre que l'utilisateur se doit de fixer au début du processus de partitionnement. Stuckenschmidt et Klein estiment que les concepts dans un module se doivent d'être fortement interconnectés. Cette propriété du module permettrait d'identifier tout d'abord les concepts qui ne le sont pas afin des les exclure du cluster. Une fois les limites du potentiel module défini, chacun des concepts qui aurait été préalablement exclu sera rajouté au dit module

uniquement dans la mesure où ce dernier contiendrai un ou plusieurs concepts auxquels ce concept potentiellement pertinent soit lié, et ce plus qu'à tout autre concept appartenant à un autre cluster ou potentiel module voisin.

#### 4.4.1.2 La technique de Cuenca Grau *et al.*

Grau et al [GRAU et al., 00] proposent une approche basée sur les E-connexions pour résoudre le problème du partitionnement d'une ontologie, qui est considéré non seulement comme un langage de représentation de connaissances (combinaison de différents formalismes logiques) mais également comme un langage de définition et d'instanciation d'ontologies OWL.

Les E-connexions ont été élaborées dans le but d'accroître l'expressivité de chaque composant logique tout en préservant la décidabilité des raisonneurs [KUTZ et al., 04]. Ce langage permet de séparer distinctement les domaines d'interprétation de n-systèmes combinés (chaque système peut être vu comme étant une base de connaissances en logique de description), où ces domaines sont liés à l'aide de *n* relations de type *«link relations».* Ces relations permettent de représenter les connexions entre différentes partitions de telle sorte que le raisonnement peut se faire sur chaque partition individuellement ou plutôt sur une combinaison de partitions liées entre elles.

Les partitions générées par les ε-connexion sont à la fois structurellement et sémantiquement compatibles [GRAU et al., 05]. Soient une ontologie *O* et une ε-connexion ∑ ayant respectivement pour vocabulaires *V* et *Vn* qui est structurellement compatible avec *O (∑~O)* si et seulement si :

1. V*n* est un vocabulaire partitionné de *O.* Ce qui sous-entend que ∑ contient exactement les mêmes entités (concepts, propriétés et individus) et axiomes que *O.*

*2.* $A \in \sum \Leftrightarrow A \in O$

$\sum$ est sémantiquement compatible avec *O* ($\sum \approx O$) si et seulement si :

1. *O* est partitionnable pour Vn.
2. Si $x$ (Vn) $\leftrightarrow M$, *alors* $M \vDash \Sigma$ *ssi* $x$ (Vn $\vDash O$)

Les ε-connexion assurent deux types de compatibilité : structurelle et sémantique.

La compatibilité structurelle garantie qu'aucune entité ou axiome ne sera ajouté, retiré ou modifié lors du partitionnement. En d'autres termes, on tient à s'assurer que chaque axiome existant dans la ε-connexion existe aussi dans l'ontologie. La compatibilité sémantique représente ici la relation souhaitée entre l'ontologie initiale et l'ontologie résultante du processus de partitionnement. En effet, le rôle de la compatibilité sémantique est de garantir que l'interprétation de l'ontologie partitionnée est équivalente à l'interprétation de la même ontologie non partitionnée, donnant lieu ainsi à la préservation du modèle d'interprétation. On s'assure principalement que chaque ontologie équivalente corresponde exactement au même ensemble de ε-connexions compatibles. L'approche de partitionnement de Cuenca Grau *et al. p*ermet ainsi d'identifier les propriétés dans *O* qui lient *O* à $\sum i$ ou $\sum i$ à *O,* tout en garantissant à la fois les compatibilités structurelle et sémantique [PATRICK, 14].

**Analyse des techniques de partitionnement**

L'approche de Stuckenschmidt et Klein ne considère pas la sémantique de l'ontologie, elle est applicable donc à travers différents langages d'ontologie; tandis que l'approche de Cuenca Grau *et al.* s'appuie uniquement sur la logique de description, qui apporte à leur approche la garantie qu'il n'y aura aucune altération de l'information contenue dans les partitions par rapport au contenu de l'ontologie d'origine.

Mais il reste à signaler que le calcul de degré de dépendance dans l'approche de Stuckenschmidt et Klein pourrait dans un certain sens être vu comme un facteur limitant l'efficacité de leur algorithme, dans la mesure où en plus de faire abstraction de la sémantique de l'ontologie, il ne considère aucunement le cadre dans lequel la partition crée entend être utilisée. À l'instar de l'approche de Stuckenschmidt et Klein, l'algorithme de partitionnement de Cuenca Grau *et al.* n'est pas paramétrable, mais son réel défaut vient du fait qu'il ne s'applique pas à toutes les ontologies, car certaines ne sont pas partitionnables à l'aide des ε-connexions, et ce dans les cas où les compatibilités structurelle et sémantique ne pourraient être garanties lors du processus de partitionnement.

### 4.4.2 Les techniques d'extraction de module

L'extraction de module ontologique est un processus au cours duquel un module $\mathcal{M}$ couvrant une signature spécifique est extrait d'une ontologie *0,* telle que *Sig(*$\mathcal{M}$*)*⊆ *Sig(*0). $\mathcal{M}$ est la partie pertinente de *0* qui couvre l'ensemble d'éléments définis par Sig ($\mathcal{M}$*).* Le module $\mathcal{M}$ est une ontologie en soi, et comme tel, d'autres modules peuvent être extraits à partir de lui. Formellement, l'extraction du module peut être définie comme suit:

$Extraction(0, Sig(\mathcal{M})) \rightarrow \mathcal{M}|(Ax(\mathcal{M})i \subseteq (Ax(0))i$

#### 4.4.2.1 L'extraction basée sur le parcours de graphe

Les techniques d'extraction basées sur une approche structurelle nécessitent la représentation de l'ontologie comme un graphe afin de parcourir celle-ci et d'en extraire le module ontologique. Parmi ces approches nous distinguons :

- ***L'approche de d'Aquin et al.***

D'Aquin *et al.* [D'AQUIN et al., 06] considère que la sélection des connaissances»*(Knowledge selection)* est un processus plus complexe

et l'extraction de module ontologique ce n'est qu'une étape de ce processus. Ce dernier a comme objectif de récupérer les composants pertinents des ontologies disponibles en ligne pour l'annotation automatique du page Web retournée par le navigateur. Cette approche a été implémentée au sein d'un outil appelée *Magpie,* qui est un plugin pour navigateur. *Magpie* permet d'identifier les instances de classes d'ontologies sur une page Web en surlignant chacune d'elles avec une couleur qui lui est associée.

*Magpie* permet aussi d'extraire les parties les plus utiles et pertinentes des ontologies qui décrivent les classes des instances sur une page Web.

Le processus de sélection de connaissances se fait en trois étapes (Figure 4.3) : [D'AQUIN et al., 06]

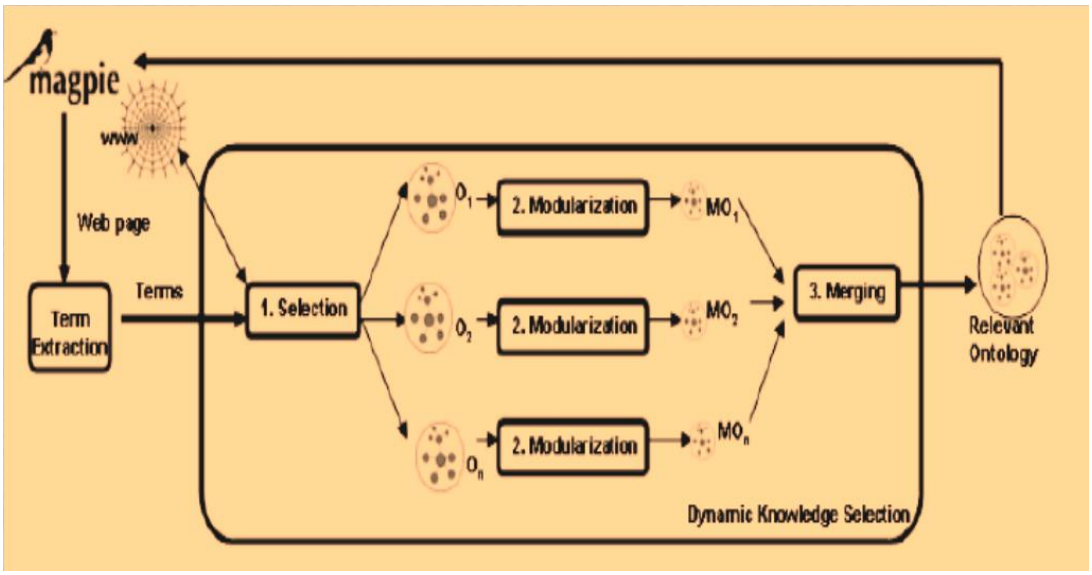



**Figure 4.3:** Le processus de sélection de connaissances et son utilisation avec *Magpie* [D'AQUIN et al., 06].

**1. *La sélection des ontologies pertinentes.***

Tout d'abord, on rentre un ensemble de termes pour lesquels on recherche une ontologie. Ensuite, on identifie les ontologies ou les ensembles d'ontologies qui couvrent les termes qui ont été rentrés. Par couverture, on entend que l'ensemble des ontologies contenant

des concepts, propriétés, ou instances qui sont sémantiquement liés aux termes donnés doit être retourné par l'algorithme.

**2. *La modularisation des ontologies sélectionnées.***

La technique de partitionnement de Stuckenschmidt et Klein est appliquée sur les ontologies retournées lors de l'étape précédente afin d'identifier les différentes sous-ontologies qui contiennent une connaissance suffisamment pertinente pour la tâche à réaliser [STUCK et KLEIN, 04]. Le principe d'extraction de cette approche est de rajouter au fur et à mesure dans les modules tous les éléments participant directement ou indirectement à la définition des entités déjà incluses dans chacun de ceux-ci. En d'autres termes, si un concept impliqué dans une définition n'est pas encore inclus dans l'un des modules, alors celui-ci y est systématiquement rajouté [PATRICK, 14].

**3. *La fusion des modules ontologiques pertinents.***

Dans la mesure où plusieurs ontologies seraient retournées après la phase de sélection, les modules extraits individuellement de celles-ci sont fusionnés afin de produire une ontologie unique, résultat final du processus de sélection de connaissances. D'Aquin *et al.* ne fournit aucun détails sur le déroulement du processus de fusion des modules.

- ***L'approche de Doran et al.***

L'approche de Doran *et al.* a comme perspective de pouvoir réutiliser une partie d'une ontologie existante. Elle ne prend pas en considération le type de langage dans lequel l'ontologie est représentée, dès lors que celui-ci est transformable en un *modèle de graphe abstrait* [DORAN et al., 07]. Ce graphe abstrait de Doran *et al.* est un graphe orienté et étiqueté *C* avec un alphabet donné *E,* tel que *C = (V, E)* où :

• $V$ est un ensemble fini de sommets.

• $E \subseteq V * \sum_E * V$ est une relation ternaire décrivant les arêtes.

La particularité de l'approche de Doran *et al.* est que dans ce *graphe* il ya un ensemble d'arêtes à parcourir et un autre à ne pas parcourir, du moins lors de la première itération de l'algorithme.

Par exemple, lorsqu'on extrait un module d'une ontologie OWL-DL, les arêtes owl:disjointWith ne sont pas parcourues à la première itération, elles le sont uniquement lors des itérations suivantes. Les arêtes sont étiquetées en fonction du langage de l'ontologie.

En utilisant le modèle abstrait de Doran *et al.,* il serait donc possible de définir un module ontologique *GM* tel que *GM= (VM, EM* ), avec $VM \subseteq V \wedge VM \neq \emptyset$ et $EM \subseteq E$. Ce qui implique que $GM \subseteq G$.Cette approche permettrait de ramener le problème d'extraction de module à un parcours de graphe avec pour point de départ un sommet $x$ tel que $x \in VG$. La seule condition est que les arêtes étiquetées *« disjoint»* du sommet $x$ ne soient pas parcourues à la première itération. Deux concepts sont disjoints s'ils n'ont aucun ancêtre commun. Toutefois, Doran *et al.* supposent que si l'utilisateur souhaite inclure les concepts disjoints dans le module lors de la première itération, alors il devra tout simplement choisir pour point de départ leur ancêtre commun.

Le module extrait est un graphe représentant respectivement les ensembles de sommets et d'arêtes. Le module ontologique généré par cette approche reste transitivement fermé et ce, même en tenant compte des différentes relations parcourues. Afin de garantir cette fermeture transitive, l'ontologie dont on veut extraire le module se doit elle aussi d'être transitivement fermée. La notion de fermeture transitive ici suggère que toute relation existante entre concepts est considérée, et ce même ci cette relation lie un concept du module à un concept intermédiaire (concept qui n'appartient pas au module mais

qui est lié à un autre appartenant au module). Il convient tout de même de noter que les concepts hiérarchiquement supérieurs au concept à partir duquel le processus commence ne sont pas parcourus; il revient à l'utilisateur de définir le concept de départ et selon son choix, s'il souhaite inclure des concepts disjoints dans le module lors de la première itération. En considérant les concepts plus «généraux» lors du processus d'extraction, on risquerait de se retrouver avec un module dont les proportions sont équivalentes à celles de l'ontologie à modulariser [DORAN et al., 07].
L'avantage de cette approche est que le modèle de graphe abstrait est applicable à toutes ontologies, quelque soit leur langage.

***L'approche de Noy et Musen***

Noy et Musen dans leur approche considère un module ontologique comme étant une vue d'ontologie, en se basant sur le principe d'encapsulation. Comme la notion de vue issue du domaine des bases de données. [NOY et MUSEN, 04]. Ce module représente une sous-structure qui résulte comme réponse à une requête de l'utilisateur sur l'ontologie initiale. Les composants de la requête sont la liste des termes qui représente en fait la signature du module à obtenir.
Le point de départ du processus d'extraction doit être fixé par l'utilisateur en désignant un concept dont les relations seront parcourues récursivement afin d'inclure l'ensemble des entités qui lui sont liées.
Aussi, les relations à parcourir sont sélectionnées par l'utilisateur qui attribue à chacune de celles-ci une profondeur de parcours : il s'agit de la directive de parcours *(traversal directive) TD.* Lorsque L'algorithme arrête de parcourir la relation sélectionnée une fois la profondeur est atteinte. La directive de parcours *D* d'une ontologie *0* est définie par la paire $(C_{st}, PT)$ où :

• $C_{st}$ est le concept de départ du parcours.

• *PT* est un ensemble de directives de propriétés *(property directives).*

Chaque directive de propriété est une paire $< P, n >$ ou P est une propriété de *0* et *n* est un entier non-négatif ou infini (∞) qui définit la profondeur de parcours de la propriété *P.* Si $n = (\infty)$, alors le parcours inclura aussi une couverture transitive pour *P* à partir de *Cst·*

Noy et Musen définissent une spécification de la vue de parcours *(traversal view specification) T* comme étant un ensemble de directives de parcours. Soit une ontologie *0* et une spécification de la vue de parcours *T* constitué d'un ensemble de directives de parcours *TV.* Le résultat d'une spécification de la vue de parcours *TV(O, T)* est aussi une vue de parcours qui en fait, représente l'union de tous les résultats des directives de parcours $D$ , tel que $D \in TD$. On rappellera qu'une vue de parcours contient toutes les classes et instances rencontrées tout le long du parcours.

La technique de Noy et Musen a été implémentée et incorporée dans l'outil *PROMPT,* qui est un plugin pour l'éditeur d'ontologie *Protégé* et qui donne à l'utilisateur la possibilité de gérer plusieurs ontologies et ce, en lui permettant de comparer les versions, de fusionner et d'extraire les modules ontologiques [NOY et MUSEN, 03].

Cette approche est flexible et permet à un ingénieur de connaissances de construire un module de façon itérative, mais pour ce faire, il doit avoir des connaissances approfondies sur l'ontologie à manipuler afin d'être capable de définir lui même des directives de parcours appropriées [NOY et MUSEN, 04].

- ***L'approche de Seidenberg et Rector***

L'approche de Seidenberg et Rector est une technique d'extraction de module ontologique à partir d'une ontologie médicale GALEN [SEID et RECTOR, 05].

Soit un concept *A* de l'ontologie et une signature de *M* tel que *Sig(M)* =*A*. Le processus d'extraction se fait en deux phases : (Figure 4.4)

1) L'ontologie est parcourue vers le haut afin d'inclure toutes les superclasses de *A*.
2) La hiérarchie de l'ontologie est parcourue vers le bas dans le but de rajouter les sous-classes de *A*. Il est à noter que les classes ayant un ancêtre commun et se trouvant à la même profondeur dans la hiérarchie sont ignorées. Il faut les rajouter dans *Sig(M). p*our les inclure dans le module. Les restrictions, intersections, unions et toutes classes équivalentes de celles déjà incluses sont aussi rajoutées au module.

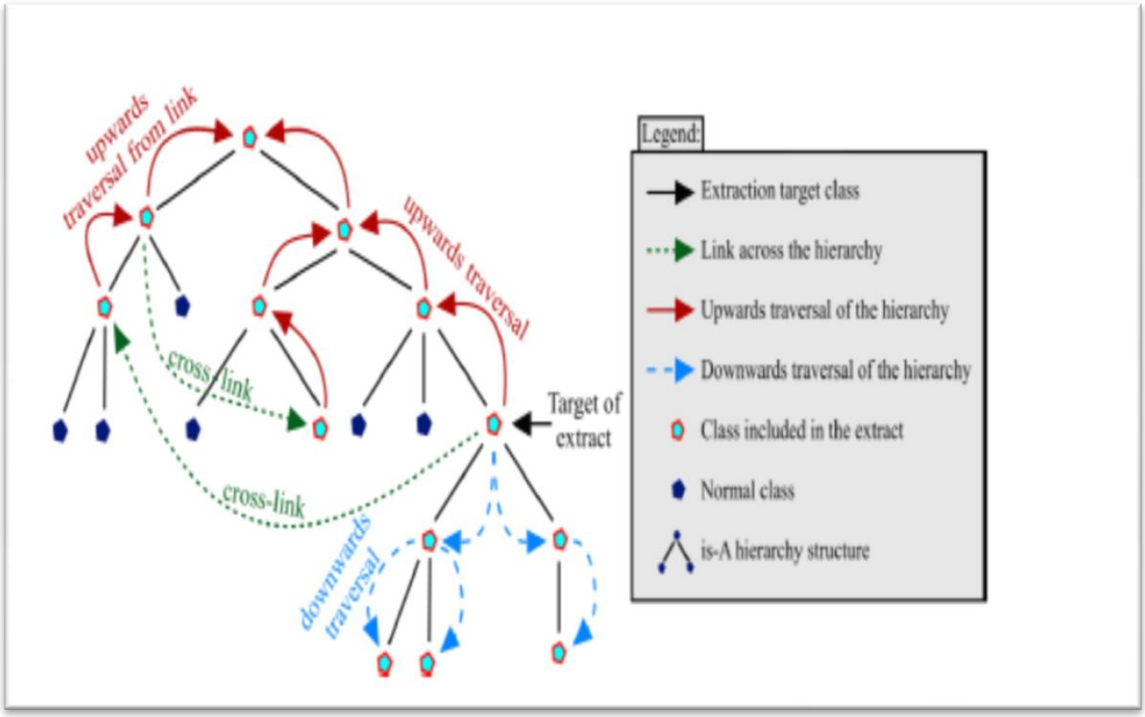



**Figure 4.4:** Parcours de la hiérarchie des classes à travers les liens [SEID et RECTOR, 05]

En dernier lieu, les propriétés des classes précédemment sélectionnées sont aussi parcourues vers le haut afin de rajouter d'autres classes au module.

Seidenberg et Rector propose un algorithme amélioré en introduisant deux notions essentielles qui sont le filtrage de propriétés et l'utilisation de classes frontières, pour pouvoir contrôler les proportions du module dans le cas où on travaillerait avec une ontologie dense et large. Le processus de filtrage de propriétés consiste à supprimer les propriétés choisies par l'utilisateur [SEID et RECTOR, 06].

Pour illustrer cette approche [PATRICK, 14], considérons un cas où l'utilisateur ne serait pas intéressé par l'ensemble des maladies modélisé dans l'ontologie GALEN. Celui ci choisira alors d'exclure toutes les propriétés locatives, c'est-à-dire uniquement des propriétés qui lient des maladies à des parties précises du corps humain. Par exemple, l'utilisateur pourrait, afin d'éliminer l'ensemble de maladies juste supprimer toutes les propriétés *hasLocation* dans des relations similaires à celle-ci: "IschaemicCoronaryHeartDisease *hasLocation* Heart".

Le filtrage de propriétés passe aussi par la suppression de toutes les restrictions de classes dans lesquelles ces propriétés apparaissent. Toutefois, il arrive fréquemment qu'en supprimant une restriction, la définition de la classe concernée devienne, soit impossible à distinguer, soit équivalente à une autre définition de classe similaire. On assiste ainsi à l'apparition dans l'ontologie de longues séries de classes équivalentes qui, bien que correctes sont impossibles à visualiser dans un éditeur d'ontologie tel que Protégé. Pour résoudre ce problème, Seidenberg et Rector proposent une méthode permettant de transformer les classes équivalentes en classes primitives qui conservent leur position dans la hiérarchie, devenant ainsi facile à visualiser dans les éditeurs. L'exemple suivant illustre le filtrage d'une propriété avec le retrait d'une définition :

$$SkinOf\ Frontal\ Scal \equiv (SkinofScalp \sqcap \exists hasSpecificProximity.FrontalBone$$

$$SkinOf\ Frontal\ Scalp \equiv SkinOf\ Scalp\ SkinO\ f\ Frontal\ Scalp \sqsubseteq SkinO\ f\ Scalp$$

Lors du processus de filtrage, la restriction présente sur la classe *SkinO ƒ Scalp* est retirée. La résultante est une classe équivalente, dont la définition est convertie par la suite en classe primitive.

La seconde méthode de délimitation du module proposée par Seidenberg et Rector se fait à l'aide de deux éléments préalablement définis par l'utilisateur à savoir la profondeur de récursion et le concept cible à partir duquel le processus de segmentation débute. Le parcours des liens à travers l'ontologie en partant du concept cible s'arrête lorsque la classe qui se trouve à la frontière est atteinte :

on parlera alors de classe frontière (*boundary class).* En effet, une classe frontière est la classe sur laquelle l'algorithme de segmentation s'arrête lorsqu'une certaine profondeur de récursion est atteinte, entraînant ainsi la suppression de tous les liens de cette classe (Figure 4.5). Il s'agit d'une méthode efficace qui permet de limiter la taille du module dès lors que les classes frontières sont considérées comme étant la condition d'arrêt de l'algorithme.

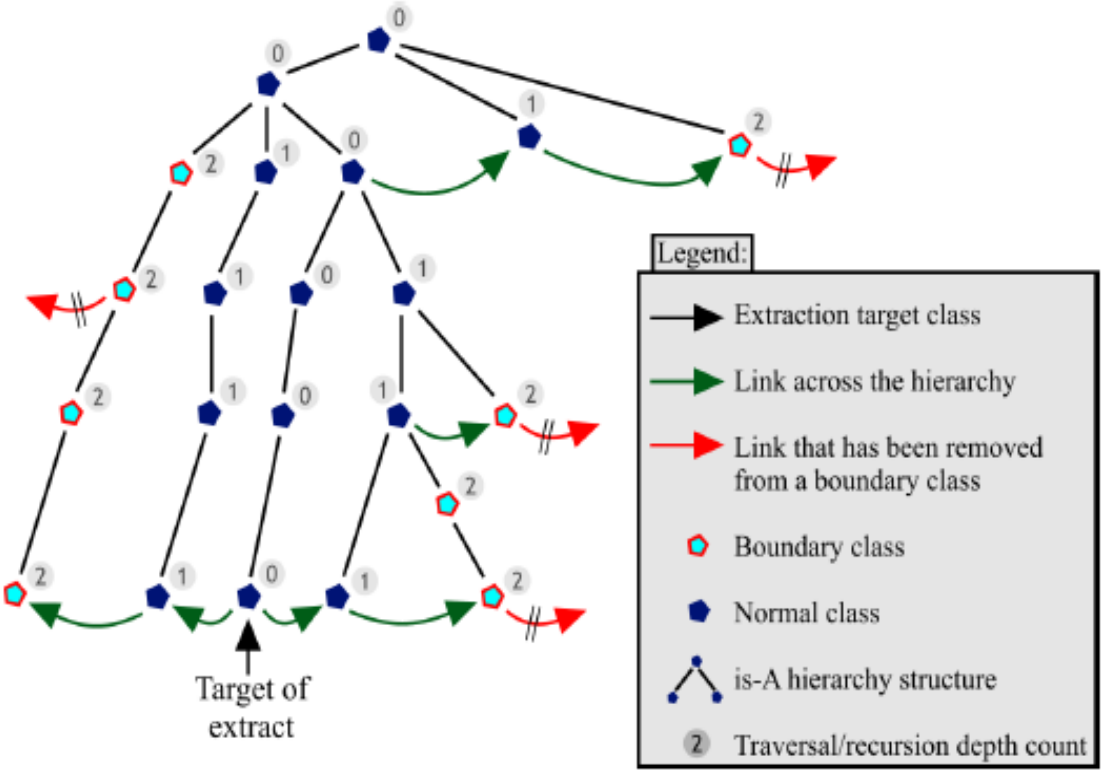


**Figure 4.5 :** Extraction de segment avec profondeur de 2 [SEID et RECTOR, 06]

L'algorithme basique d'extraction de module ontologique de Seidenberg et Rector est implémenté dans l'outil *SegmentationApp*. C'est une application contenue dans un fichier .jar

Cet outil prend en entrée deux éléments à savoir l'ontologie à modulariser et le fichier texte contenant la signature du module à extraire. Le module ontologique ainsi généré est automatiquement exporté dans un fichier .owl. L'algorithme basique de Seidenberg et Rector n'est pas paramétrable.

**L'extraction basée sur la logique**

En plus des techniques d'extraction basée sur le parcours de graphe, il existe d'autres techniques d'extraction basée sur la logique qui reposent sur la notion d'extension conservatrice *(conservative extension).* Une ontologie est considérée comme une extension conservatrice si les implications logiques à propos du module sont comprises dans sa signature et un module est une sous-structure de cette ontologie dont il est extrait.

Formellement, et selon Lutz *et al.* une extension conservatrice est définit comme suit [LUTZ et al., 07] :

Soient $\mathcal{T}_1$ *et* $\mathcal{T}_2$ des T-Box formulées en logique descriptive $\mathcal{L}$ et soit $\Gamma \subseteq Sig(\mathcal{T}_1)$ une signature. Alors $\mathcal{T}_1 U \mathcal{T}_2$ est une Γ-extension conservatrice de $\mathcal{T}_1$, si pour tout $C_1, C_2 \subseteq \mathcal{L}(\Gamma)$, on a $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$.

Cette définition sous-entend que toutes les implications logiques de la signature d'un module ontologique sont les mêmes que si on fait l'union de ce module et de l'ontologie dont il a été extrait.

Déterminer si une ontologie *O* est une extension conservatrice est un problème indécidable en langage OWL-DL c'est l'inconvénient avec la définition de Lutz *et al.*

Pour pallier à ces limites, Grau *et al.* proposent d'utiliser des contraintes moins strictes : on parlera de modules basés sur la localité (*locality-based modules)* [GRAU et al., 08]. Grau *et al.* introduisent également deux nouvelles notions: la couverture ( *coverage)* et la sécurité ( *safety)* qui sont des propriétés garanties par des modules basés sur la localité. Ces propriétés sont définies en des termes d'un module importé par une ontologie locale *(L)* comme suit [GRAU et al., 07] :

• La couverture garantit que l'ensemble des termes qui se rapportent aux termes spécifiés sera extrait de l'ontologie. Un module $0'$ couvre une ontologie $0$ pour les termes d'une certaine signature *Sig* dans la mesure où pour toutes les classes $A$ et $B$ tels que $A, B \in Sig(0')$, si $L \cup 0 \models A \sqsubseteq B$ alors $L \cup 0' \models A \sqsubseteq B$.

• La sécurité garantit que la signification des termes extraits ne sera pas modifiée. $L$ utiliserait les termes de la signature *Sig* de façon sécuritaire dans la mesure où pour toutes les classes $A$ et $B$ tels que $A, B \in Sig(0')$ si $L \cup 0' \models A \sqsubseteq B$ alors $0' \models A \sqsubseteq B$.

Grau *et al.* décrivent aussi deux types de localité : la localité syntaxique qui peut être calculée en temps polynomial, et la localité sémantique dont le calcul est un problème PSPACE-complet [GRAU et al., 09]. À la différence de la localité syntaxique qui est calculée à partir de la structure syntaxique des axiomes, la localité sémantique repose uniquement sur l'interprétation *(I)* de l'axiome.

Jimènez *et al.* quant à eux proposent deux conditions différentes de localité afin d'extraire les modules ontologiques [JIME-RUIZ et al., 08]

• La $\perp$-localité qui extrait un module approprié pour un raffinement et celui-ci contient tous les super-concepts de la signature.

•La $\top$-localité qui extrait un module approprié pour une généralisation et qui contient tous les sous-concepts de la signature.

L'approche d'extraction basée sur la logique de Grau *et al.* est implémentée dans l'outil *OWL Module Extractor*. C'est une application Web développée par Rafael Gonçalves dans le cadre d'un projet en ingénierie de connaissances à l'université de Manchester (Royaume-Uni). OWL Module Extractor prend en entrée deux paramètres : l'URI ou le contenu de l'ontologie à modulariser et la signature qui représente la liste de termes à couvrir par le module à extraire.

### 4.4.3 Comparaison

Dans cette section nous présentons une étude comparative entre les approches d'extraction de modules basées sur le parcours de graphe Tableau 4.1 et les approches d'extraction de module basées sur la logique selon un certain nombre de critères [SETTI AHMED et al., 11].

| | Coverage | Minimality | DL EXP | Tractable |
|---|---|---|---|---|
| **Whole Ontology** | + | - | Any | + |
| **Locality Based** | + | - | Owl 1 | + |
| **MEX** | + | + | EL++ | + |
| **Conservative** | + | + | Any | - |

**Tableau 4.1 :** comparaison des approches d'extraction de modules basée sur le parcours de graphe.

| | Interactive | Traversal direction | Property filtring | Least common subsumer | Assume inferred Model |
|---|---|---|---|---|---|
| **Whole Ontology** | _ | Up& Down | _ | _ | _ |
| **d'Aquin et al** | _ | Up& Down | _ | + | + |
| **Doran et al** | _ | Down | _ | _ | _ |
| **Noy and Musen** | + | Up& Down | _ | _ | _ |
| **Seidenberg and Rector** | _ | Up | + | _ | _ |

**Tableau 4.2 :** comparaison des approches d'extraction de modules basée sur la logique.

**L'extraction basée sur SPARQL**

Les approches proposées par Borgida et Giunchiglia ainsi que d'Aquin *et al.* nécessitent que l'utilisateur doit avoir des connaissances sur les formalismes non standards [BOR et GIUNCH, 07], [D'AQUIN et al., 07].

En revanche le travail de Doran *et al.* utilise RDF et SPARQL comme base pour un framework commun d'extraction de module, car tous deux sont des standards W3C [DORAN et al., 08].

Toutes les ontologies OWL peuvent être représentées en un graphe RDF et SPARQL est le langage permet tout particulièrement, l'interrogation de descriptions RDF. A l'image de RDQL (dont il est une extension), SquishQL ou TriQL, SPARQL exploite, en premier lieu, RDF au travers de la notion de triplets ou d'ensembles de triplets. La réponse à la requête posée va ainsi correspondre à la restitution du sous-graphe RDF satisfaisant le filtre exprimé au travers d'opérations de mise en correspondance de patterns de graphe (possiblement optionnels) et d'opérations basées sur les connecteurs logiques (conjonction, disjonction).

Doran *et al.* présentent le framework SOMET (Figure 4.6) et montrent ainsi qu'il est possible de classer les approches d'extraction de module basées sur le parcours comme étant une série de requêtes SPARQL sur un graphe RDF.

Le framework SOMET contient donc des représentations SPARQL des différentes techniques d'extraction de module par parcours de graphe. Cet outil permet à l'utilisateur d'ajouter, de modifier ou de retirer des requêtes SPARQL de l'ensemble de requêtes qui doivent passer au moteur d'extraction *(Traversal Extraction Engine)*. Ce qui lui juge flexible. Étant donné que les différentes approches d'extraction sont assimilées à un ensemble de requêtes SPARQL, ces ensembles ne sont pas disjoints et leurs intersections permettent ainsi de mettre en évidence les points communs entre les différentes techniques.

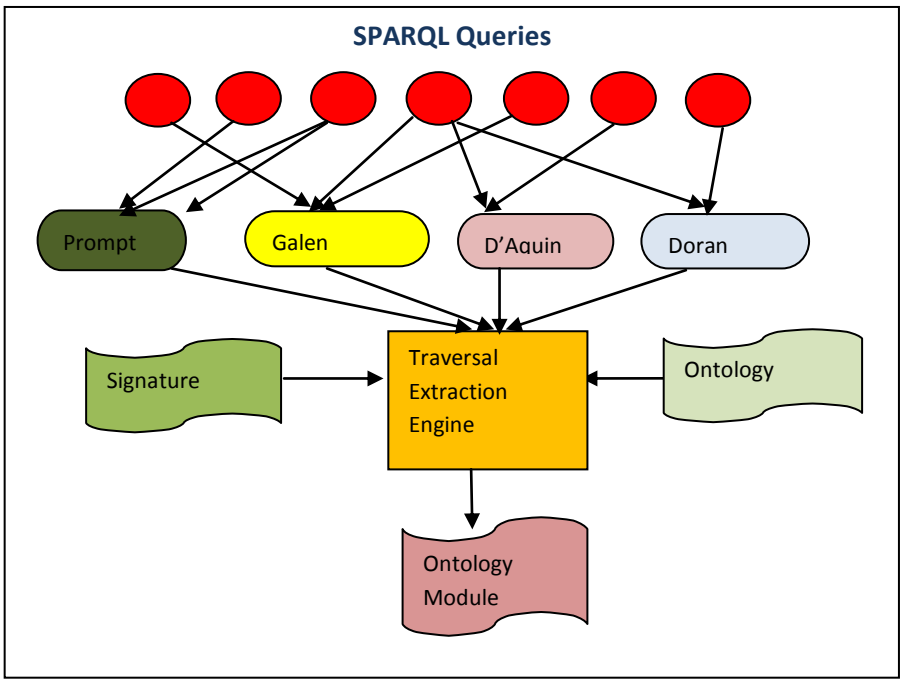



**Figure 4.6 :** Le framework SOMET [DORAN et al., 08].

SOMET utilise un algorithme de parcours de graphe dont les paramètres sont : l'ontologie dont on veut extraire le module, la signature qui décrit le module et un ensemble de requêtes SPARQL. Il

s'agit d'un algorithme itératif qui, dans un premier temps, applique des requêtes sur les éléments de la signature afin de construire le module désiré, et dans un deuxième temps de nouvelles requêtes sont appliquées sur chacun des éléments retournés par les requêtes précédentes. Ainsi de nouveaux éléments sont rajoutés au module à chaque itération [DORAN et al., 08].Dans le cas de l'approche de Doran *et al.* , dont le processus d'extraction se fait à partir d'un concept unique, on distingue les requêtes SPARQL suivantes :

• DESCRIBE ? *c* : cette requête permet de décrire le concept sur lequel elle est appliquée et ce, en lisant toutes les déclarations (*statements)* dans lesquelles le concept? *c* apparaît comme étant le sujet.

• CONSTRUCT { ? *y* rdf s : domain ? c} WHERE { ? *y* rdf s : domain ? c} : retourne toutes propriétés *?y* dont le concept ? *c* est l'espace de départ *(domain)* de la propriété.

• DESCRIBE ? *y* WHERE { ? *y* rdf s subClassOf ? c} : retourne toutes les sousclasses de ? *c*

• CONSTRUCT *{ ?y* owl : equi valentClass ? c}

WHERE *{ ?y* owl : equivalentClass ? c} : retourne tous les concepts *?y* où *?y* est une classe équivalente à ?c.

## 4.5 Partitionnement d'une ontologie à base de clustering

Dans cette section nous passons en revue d'autres approches de partitionnement d'ontologie à base de clustering. Pour ce faire nous faisons tout d'abord un survol sur les concepts de base du clustering ensuite nous exposons quelques approches car peu de travaux existe dans ce domaine de partitionnement des ontologies OWL.

La problématique générale du regroupement (ou *clustering*) consiste à organiser un ensemble d'objets en groupes de façon à ce que deux objets similaires se retrouvent dans un même groupe et deux objets

dissimilaires dans des groupes distincts. De nombreuses stratégies ont été proposées pour répondre à cette problématique, comme par exemple les approches par partitionnement (k-moyennes), les algorithmes hiérarchiques (agglomératifs ou divisifs), les méthodes utilisant des mélanges de densités de probabilité, des découpages en grilles, etc.

Donc Le Clustering consiste à créer une partition ou une décomposition de cet ensemble en sous parties (clusters) telle que :

- Les données appartenant au même groupe se ressemblent,
- Les données appartenant à deux groupes différents soient peu ressemblantes.

Mathématiquement, on a un ensemble X de N données décrites chacune par leurs P attributs.

**1- Exemple :**

On utilise souvent le clustering en traitement d'images pour fixer les divers objets qu'elles contiennent (segmentation) : routes, villes, rues, des organes humaines (pour les images médicales).

**2-Exemples d'applications :**

- Classification de documents
- Regroupement de documents par leur contenu
- k dimensions : nombre d'occurrences d'un mot dans un document (k mots en tout → beaucoup de dimensions)

### 4.5.1 Clustering K-Means

La méthode de K-Means ou algorithme de partitionnement par centre mobile permet d'effectuer un partitionnement d'un ensemble de données en K clusters.

Un cluster regroupe plusieurs concepts similaires. Chaque cluster (partition) est décrit par son centre. Les centres des clusters sont

mobiles au cours de l'exécution de l'algorithme.

L'algorithme peut se présenter comme suit :

- Le nombre de clusters, le paramètre K, est fournit au départ.
- Un ensemble de K centres choisi dans l'ensemble des données,
- Les K clusters sont formés en regroupant dans chaque centre l'ensemble des données plus proches du centre courant que de tout autre centre,
- Le centre de chaque cluster est calculé et devient le nouveau centre,
- L'algorithme boucle alors sur l'étape précédente : Les données sont réaffectées en fonction de ces nouveaux centres et la condition d'arrêt est que les centres deviennent immobiles.

### 4.5.2 Méthodes de Clustering

Il existe une trentaine de méthodes et leurs variations incluent dans différentes familles. Nous présentons les principales familles de méthodes de regroupement des données en clusters. Les méthodes peuvent être séparées en quatre groupes :

**4.5.2.1 Les méthodes basées sur une distance** : Ces méthodes se basent sur la notion de distance entre objets du jeu de données, en posant que si deux objets sont proches suivant cette distance, ils doivent être regroupés ensemble dans un même cluster. Les algorithmes K-means [KAROL et MANGAT, 13] et Fuzzy-c-means [RTK et al., 95] sont les algorithmes les plus connus de cette famille d'algorithme. Ces méthodes permettent de trouver des formes de clusters convexes et sont très utilisées notamment à cause de leur coût d'exécution faible.

**4.5.2.2 Les méthodes basées sur une grille** : Leur processus consiste à regrouper les cellules denses les plus proches. Ces méthodes ont été proposées pour réduire l'explosion combinatoire des

méthodes à base de densité qui fait suite à l'augmentation du nombre d'objets.

**4.5.2.3 Les méthodes probabilistes** : Ces méthodes supposent que les données suivent une certaine loi de probabilité. L'objectif est d'estimer les paramètres de cette loi et de définir un modèle de mélange de lois pour représenter les différents clusters.

**4.5.2.4 Les méthodes hiérarchiques** : Ces méthodes construisent une hiérarchie de clusters. Chaque nœud contient ses clusters enfants, et les nœuds frères partitionnent les objets contenus dans leurs parents. Ce type d'approche permet d'explorer les données à différents niveaux de granularité. Les méthodes de clustering hiérarchique sont décomposées en deux types d'approches, les approches ascendantes où l'algorithme part d'un grand nombre de clusters et ceux-ci sont ensuite fusionnés jusqu'à n'obtenir plus qu'un unique groupe contenant tous les objets du jeu de données et les approches descendantes qui partent, de l'ensemble des données, et le divisent en clusters qui sont ensuite divisés récursivement.

## 4.6 Travaux dans le domaine de partitionnement des ontologies à base de Clustering

Parmi les travaux dans le domaine de partitionnement à base de clustering on peut citer le travail de [SARULAD et al., 12] : dans ce travail un algorithme de partitionnement est appliqué pour la décomposition d'une vaste ontologie. C'est une révision de l'algorithme d'AHSCAN. (**A**gglomerative **H**ierarchical **S**tructure **C**lustering **A**lgorithm for **N**etworks) est un algorithme de clustering avec une complexité de O(n). Dans cet algorithme un concept (V) est considéré comme (neighbor) d'un autre concept (W) s'il existe un lien entre eux. V est le subsumeur de W. Le concept V est le descendant ou le fils

du concept W. Deux concepts reliés qui n'ont pas de nœud (edge) subsumeur entre eux ne sont pas considérés comme ascendant ou descendant. Dans cet algorithme un nouveau paramètre est introduit (paramètre de mesure de la qualité de la modularisation) afin de trouver la meilleure partition.

Dans les travaux de Kolli [KOLLI, 08] l'ontologie est représentée sous forme d'un graphe. L'algorithme de clustering est traversé en commençant par la racine (R) et assemblant MB nombres de nœuds avec un ensemble (2*MB) nombre de nœuds.

L'objectif de cette approche comme indiqué par les auteurs est juste pour diviser l'ontologie afin de permettre son traitement dans la réalité.

Dans les travaux de Hu et al [HU et al., 06] l'algorithme de partitionnement est réalisé sur un graphe construit en se basant sur les dépendances de la hiérarchie de classes. Dans cette approche, un poids est associé à ces dépendances en utilisant des informations linguistiques et structurelles des entités. Ensuite l'algorithme de Rock [GUHA et al., 99] est appliqué (c'est un algorithme agglomératif de clustering) pour partitionner le graphe. Dans l'étape finale des partitions sont générées chacune est appelée bloc.

Dans l'étude réalisée par [HUNG et LAI, 06] pour partitionner une ontologie qui est représentée sous forme d'un graphe, les auteurs génèrent une matrice d'incidence. Ici la valeur de mesure de similarité entre deux entités est partiellement déterminée par le nombre des nœuds commun entre eux.

Après partitionnement, les partitions générées avec similarités supérieurs sont alignées ensemble.

L'approche introduite par [KUNGER et PURAJI, 09] est une revue sur les techniques de partitionnement des graphes en appliquant la notion de similarité sur une ontologie représentée sous forme d'un graphe.

Dans [SANG et LAVANYA, ??] un rapport de recherche non publié, un algorithme de partitionnement de données est utilisé. En exploitant les relations entre les concepts d'une ontologie dans le domaine (PHC) (preventive health care domain). Des tables d'attributs et de concepts ainsi qu'une autre table de relations (liens) entre concepts sont générés pour implémenter l'algorithme CBRO (clustering based relational ontology) un algorithme de clustering à base de relations ontologiques.
Les données sont partitionnées au niveau des concepts ensuite en fonction des relations ontologiques existantes, d'autres nouvelles relations seront dérivées entre deux concepts.
L'algorithme est appliqué dans le domaine de PHC.
Trouver les correspondances sémantiques à travers des attributs multiples est plus important dans plusieurs applications tellesque (l'intégration de schéma).

Dans les travaux de [DING et SAN, 13] les auteurs utilisent l'algorithme de K-means pour réaliser l'alignement entre plusieurs attributs.
En premier lieu, les auteurs convertirent les attributs en points, ensuite exécutent l'algorithme K-means pour partitionner les attributs à plusieurs clusters.
Les attributs dans le même cluster ont la même sémantique.
Dans cet algorithme, le nombre de K objets est choisi aléatoirement comme des centres de clusters ensuite la méthode TFIDF est utilisée pour calculer le poids. Aussi le modèle de l'espace vectoriel est appliqué comme métrique de calcul de distance entre les points d'attributs).
Finalement, une partie expérimentale a été réalisée.

Le travail de [JIANG et al., 06] propose une méthode de clustering d'une ontologie par la définition d'une nouvelle mesure pour calculer

la similarité. L'arbre d'agrégation crée a une forte sémantique.
Cet algorithme combine la similitude de l'ontologie avec la valeur de l'objet et décide quel objet doit être remplacé.
L'arbre d'agrégation est créé entièrement différemment selon les cas d'application.

## 4.7 Discussion

**Avantages et limites des approches structurelles et sémantiques**

La façon la plus évidente pour l'Homme de visualiser une ontologie est de la représenter comme un graphe dont les sommets sont des concepts et individus et les arêtes sont des relations (hiérarchiques et sémantiques). Le problème d'extraction de module ontologique serait donc ainsi ramené à l'extraction d'un sous-graphe contenant les concepts les plus pertinents par rapport à la liste de termes entrée par l'utilisateur [DUPONT et al., 06]. Mais alors, on se retrouve dans la nécessité d'évaluer la pertinence des concepts les uns par rapport aux autres. On pourrait se référer à la notion de distance en ce sens que les plus pertinents seraient les plus proches des concepts de départ dans un certain rayon prédéfini par l'utilisateur, et dont la distance inter-concepts pourrait être évaluée par le calcul du plus court chemin de *Dijkstra* [ZIDI, 06]. Cette approche serait réalisable, d'autant plus qu'il existe une bibliothèque Java qui s'appelle JUNG, qui permet de transformer une ontologie en graphe.
Les avantages de cette approche structurelle sont multiples. Tout d'abord, il existe déjà des algorithmes efficaces d'extraction de sous-graphs. En supposant que l'ontologie dont on veut extraire le module est de qualité satisfaisante, cette approche ne déstructure pas l'ontologie dans la mesure où le module extrait conserve la même structure interne que l'ontologie d'origine. Et pour peu que le graphe ne soit pas dense, la complexité de l'algorithme s'en trouve réduit. [PATRICK, 14]

Par contre, JUNG considère par défaut que toutes les arêtes du graphe sont équivalentes. De ce fait, il devient impossible d'évaluer correctement les nœuds et les arcs, d'autant plus que dans une ontologie les relations entre différents concepts ne sont jamais équivalentes. De plus, cette méthode ignorerait les concepts éloignés des concepts de départ mais qui pourraient être tout aussi pertinents.
La deuxième façon de voir une ontologie serait de la considérer à travers son utilité qui est la représentation des connaissances. En d'autres termes, lorsqu'il est question de définir des concepts qui sont utiles les uns par rapport aux autres afin de comprendre le domaine que l'ontologie décrit, on parlera de sémantique du domaine. Ceci dit, en faisant totalement abstraction de la structure de l'ontologie, on serait à même de prendre des concepts deux à deux afin de voir s'ils sont sémantiquement proches [JIANG et CON, 97]. À la différence de l'approche structurelle, on fait allusion ici à une distance sémantique qui nous permettrait d'évaluer la proximité sémantique des concepts de l'ontologie par rapport à ceux correspondant aux termes rentrés par l'utilisateur.
Les avantages de l'approche sémantique sont la pertinence et la précision, car ce qui fait le principal atout d'une ontologie demeure sa sémantique. Il suffit qu'un concept soit lié à un concept qui ne cadre aucunement avec le contexte pour lequel l'ontologie a été crée pour que la qualité de celle-ci s'en trouve altérée, car on se retrouverait ainsi avec une ontologie qui ne cadre plus avec la réalité. L'approche sémantique nous permet donc de mettre l'ontologie à plat et de tester la similarité sémantique de tous les couples de concepts de l'ontologie.
La structure et la sémantique de l'ontologie sont des aspects dont on ne saurait faire abstraction lors de la modularisation [PATRICK, 14].
La structure nous permet de guider le processus de recherche de concepts sémantiquement pertinents, en ciblant concepts et paires de concepts dont la proximité sémantique se doit d'être évaluée. De

plus, la structure initiale de l'ontologie guide le processus d'ajout de la descendance des concepts jugés pertinents au potentiel module.
Après l'état de l'art détaillé et réalisé dans le domaine de modularisation des ontologies et après le survol des travaux du partitionnement à base de clustering on peut dire que peu de travaux existe dans ce domaine de recherche à base de K-means , d'ailleurs parmi quelques travaux réalisé dans ce cadre nous avons trouvé le travail de recherche mené par [DING et SUN, 13] qui utilise le K-means pour l'alignement des ontologies. Notre approche se diffère de celle de [DING et SUN, 13] en termes d'approche et d'objectif.
A partir du constat fait, nous étions encouragés à attaquer ce domaine de recherche sur le partitionnement des ontologies à base de clustering.

Nous manipulons dans notre approche des ontologies opérationnelles écrites dans le format OWL qui seront transformées par la suite à des graphes de nœuds. Nous référons ensuite à la notion de distance qui est illustrée par la notion de mesure de similarité. Dans notre cas, il s'agit de sept mesures de similarité que nous allons utilisées et testées sur nos ontologies qui sont respectivement (Tbk, Jaccard, Cosine, Dice, Euclidienne, Wu and Palmer et Dennai). En pratique, la similarité est évaluée, en général par une mesure de similarité ou une distance. La multitude de mesures de similarité existantes dans la littérature (des nouvelles mesures sont toujours proposées aujourd'hui) est à mettre en rapport avec la multitude de méthodes et de domaines où une mesure de similarité intervient. Ce constat nous a motivé pour étudier quelques mesures de similarités dans notre approche de partitionnement proposée et de comparer les résultats obtenus afin de mieux appréhender leur comportement et de mieux choisir une mesure qui génère les meilleurs résultats en terme de partitions.

## 4.8 Conclusion

Nous avons vu dans le présent chapitre que dans les environnements distribués comme le web sémantique, les ontologies sont construites au travers de différents points de vues indépendamment les unes des autres. On peut donc supposer qu'il existe une hétérogénéité de celle-ci.

En outre plus qu'une ontologie est grande plus il est difficile de maintenir l'exactitude du modèle. Le concepteur crée un modèle compréhensible et appréhendé pour ensuite l'intégrer dans l'ontologie finale.

En plus et dans le cadre de la réutilisation il est intéressant de réutiliser les mêmes ontologies dans diverses applications. C'est pourquoi la modularisation est une solution.

D'après l'étude et l'état de l'art réalisé, nous avons constaté que peu de travaux existent dans le domaine de partitionnement des ontologies à base de clustering. Nos recherches se sont donc essentiellement concentrées sur l'application d'une méthode de partitionnement des ontologies en se basant sur des mesures de similarités.

Le domaine de l'identification de la similarité a été considéré comme un sujet de recherche fortement recommandé dans les domaines du Web sémantique, de l'intelligence artificielle et de la littérature linguistique.

L'identification de la similarité dans les ontologies est un concept fondamental qui est adopté par plusieurs techniques telles que le regroupement, la fouille de données (data mining), le Web sémantique et en particulier, le domaine de la recherche de l'information. Ce dernier repose largement sur des mesures pour l'identification de la similarité entre les documents [BAR et al., 99] [SAL et MCG, 83].

L'idée essentielle dans notre approche prend en compte les rapports ontologiques entre concepts. Les rapports ontologiques entre concepts peuvent être détectés par un processus de calcul de similarité entre des paires d'objets contenus dans l'ontologie.

Dans le chapitre suivant nous présenterons la conception et l'architecture globale de notre système ainsi que les différents algorithmes utilisés.

# Chapitre 5

# Approche proposée

*À partir de prémisses fausses, tout peut se démontrer. Ce n'est pas parce qu'un raisonnement est implacable qu'il n'est pas délirant.*

François Lelord

## 5.1 Introduction

Dans ce chapitre nous proposons notre approche de partitionnement des ontologies écrites en OWL à base de clustering. En fait c'est une adaptation et une amélioration de l'approche classique K-means. Nous proposons ici des algorithmes pour l'adaptation et l'amélioration de l'approche traditionnelle. Nous présentons notre architecture globale de notre approche proposée et nous détaillons les différents algorithmes utilisés.

## 5.2 Architecture globale de notre système

L'architecture globale de notre approche proposée est illustrée dans la figure 5.1.

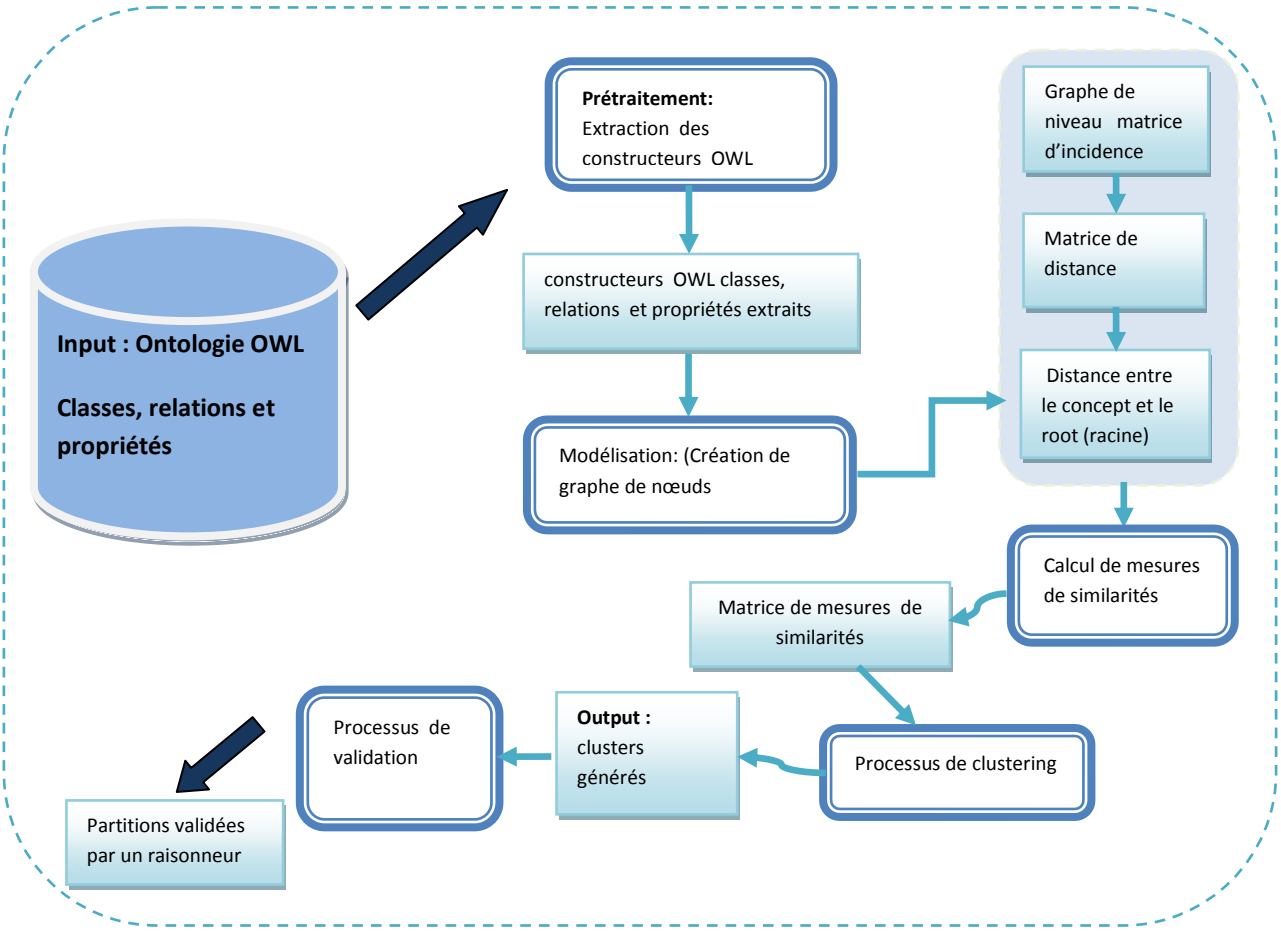



**Figure 5.1 : Architecture globale de notre système**

### 5.2.1 : Description de notre approche

Notre approche est composée de cinq phases [SETTI AHMED et al., 15], les phases de notre approche sont:

1) l'étape de prétraitement,
2) la phase de création de graphe de nœuds,
3) l'étape de calcul de mesures de similarités,
4) La phase de partitionnement et enfin la validation.

#### 5.2.1.1 La phase de Prétraitement

Dans la phase de prétraitement nous avons en entrée une ontologie OWL , ensuite on doit parcourir toute la hiérarchie de l'ontologie, pour extraire tous les constructeurs qui se trouvent dans cette dernière (classes, axiomes ; relations taxonomique et non taxonomiques et les propriétés). Pour ce faire nous avons développé l'algorithme d'extraction des constructeurs OWL suivant :

***Algorithme 1: Algorithme d'extraction des constructeurs OWL***

***Entrées : Fichier OWL***

***Sorties : Un ensemble de constructeurs OWL***

**Begin**

```
1.  Insert_Axiom(url, concept,Axioms)
2.  Add(url,concept,Axioms) In BDDAxiom
3.  Insert Properties(CanonicalName, LocalName)
4.  Add(CanonicalName, LocalName) In BDDPrperties
5.  Insert_Dataname(CanonicalName, LocalName)
6.  Add(CanonicalName, LocalName) In BDDDataName
7.  Begin
8.  OntModel model = ModelFactory.createOntologyModel(OntModelSpec.OWL MEM,null);//
    the model creation
9.  clas = it.next();
10. while (Not EOF) Do
11. Begin
12. if (URI(clas) is Not empty)
13. Begin
14. OntClass onto = model.getOntClass( URI(clas) )
15. Add(URI(clas))
16. for each (Iterator j ) To listDisjointWith(onto) Do
17. Begin
18. OntClass c = (OntClass) j.next();
19. if (URI(clas) is Not empty) and (LocalName(c) is Not empty)
20. Insert Axiom( URI(clas),LocalName(c),"DisjointWith")
21. End for
22. for each (Iterator j ) To listDifferentFrom(onto) Do
23. Begin
24. OntClass c = (OntClass) j.next();
25. if (URI(clas) is Not empty) and (LocalName(c) is Not empty)
26. Insert_Axiom( URI(clas),LocalName(c),"DifferentFrom")
27. End for
28. for each (Iterator j ) To listEquivalentClasses(onto) Do
29. Begin
30. OntClass c = (OntClass) j.next();
31. if (URI(clas) is Not empty) and (LocalName(c) is Not empty)
32. Insert_Axiom( URI(clas),LocalName(c),"EquivalentClasses")
33. End for
34. For each (Iterator j ) To listSameAs(onto) Do
35. Begin
36. OntClass c = (OntClass) j.next();
37. if (URI(clas) is Not empty) and (LocalName(c) is Not empty)
38. Insert_Axiom( URI(clas),LocalName(c),"SameAs")
39. End for
40. For each (Iterator j ) To listSubClasses(onto) Do
41. Begin
42. OntClass c = (OntClass) j.next();
43. if (URI(clas) is Not empty) and (LocalName(c) is Not empty)
44. Insert_Axiom( URI(clas),LocalName(c),"SubClasses")
45. End for
46. for each (Iterator j ) To listSuperClasses(onto) Do
47. Begin
48. OntClass c = (OntClass) j.next();
49. if (URI(clas) is Not empty) and (LocalName(c) is Not empty)
50. Insert_Axiom( URI(clas),LocalName(c),"SuperClasses ")
51. End for
52. End if
53. End while
54. clas = it.next()
55. End
56. For each element of Concept Do
57. Begin
58. Iterator iter = model.listObjectProperties()
59. ObjectProperty prp = (ObjectProperty) iter.next()
60. Insert Properties(iter.getClass().getCanonicalName(),prp.getLocalName())
61. Iterator iter1 = model.listDatatypeProperties();
62. while (iter1.hasNext())
63. Begin
64. DatatypeProperty prp1 = (DatatypeProperty) iter1.next();
```

```
65. Insert_Dataname(iter.getClass().getCanonicalName(),prp1.getLocalName())
66. End while
67. End for End
```

**Explication de l'algorithme**

Les primitives pour lire et parcourir des ontologies OWL constituent un outil plus important. Nous allons expliquer les éléments les plus essentiels.

Voici par exemple les éléments pour la création d'un modèle d'ontologie dans notre cas ontologie OWL.

`OntModelmodel=ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM, null`

Pour lister les classes d'une ontologie, la méthode `listClasse()` retourne un littérateur des objets de type `OntoClasses`. Voici un exemple pour lister les classes :

Liste des classes

```
for (Iterator j= model.listClasses()j.Next();
```

Prendre toutes les classes de l'ontologie

```
OntoClass c = (OntClass) j.next();
```

Nom local de la classe par la méthode `getLocalName())`

Avec une `OntoClass` nous pouvons obtenir les super classes et les sous classes avec les méthodes `listSuperClasses()` et `listSubClasses()`. Ces deux méthodes retournent des littérateurs contenant des `OntClass`, il est possible ainsi de parcourir la hiérarchie des classes d'une ontologie en utilisant ces méthodes pour trouver les sous-classes et les super classes.

### 5.2.1.2 Modélisation : Création de graphe de nœuds :

Après la première phase d'extraction des constructeurs de l'ontologie, la deuxième étape est la modélisation (création de graphe de nœuds). Avant de détailler cette étape il est primordial de donner quelques définitions de base intéressantes que nous avons utilisées dans notre travail.

**Définition1 :**

Notre ontologie est transformée en un graphe de nœuds (Directed acyclic graphe) qui est définie par l'ensemble (V, E, Lab), d'où :

$V=\{r, v_{2,...,} v_n\}$ est un ensemble fini de nœuds, où quelques nœuds sont identifiés uniquement par un identificateur d'objets (Object identifier OID), alors que le r est le nœud racine.

$E= \{(v_i,v_j)\} \mid vi, vj \in V$ est un ensemble fini de nœuds.

Labv est un ensemble fini des étiquettes de nœuds, ces étiquettes sont de type chaines de caractère pour décrire les propriétés des éléments et les attributs des nœuds telsques les noms des nœuds ainsi que leurs attributs.

**Définition 2**

(neighbor)Neighbors(c)= {Sub(c) ∪Sup(c)} avec Sub(c)= {c'|c' le sous-concept de c} et Sup(c)= {c'|c' le sup-concept de c}.

Dans cette étape nous avons besoin de transformer notre ontologie en graphe.

**a) Création de graphe de niveaux : (Matrice d'incidence)**

Lors de cette étape une matrice d'incidence est générée. S'il existe un chemin d'un nœud donné vers n'importe quel autre nœud dans le graphe la valeur est 1 sinon 0.

**a) Graphe de liaison en XML**

Dans cette phase, le format XML est utilisé pour la sauvegarde des concepts de l'ontologie manipulée sous forme d'entités graphiques

élémentaires, des relations en tant que liaisons. Ensuite nous optons pour l'algorithme de Djikstra [ZIDI, 06] pour le calcul du plus court chemin.

**b) Calcul de la matrice de distance**

Dans cette étape, nous utilisons l'algorithme de Djikstra [ZIDI, 06] pour le calcul du plus court chemin. Comme notre ontologie est transformée sous forme d'un graphe, l'algorithme de Dijsktra utilise le parcours en largeur d'abord » et l'approche « gourmande » (*greedy*) pour trouver les chemins les plus courts entre un nœud et toutes les destinations du graphe (les autres nœuds).

Puisqu'un graphe est composé de nœuds et d'arcs, l'algorithme de Dijsktra peut trouver les chemins les plus courts liant un nœud quelconque à tous les autres nœuds du graphe en une seule exécution.

**c) Calcul de distance entre un concept et la racine (root R)**

Comme certaines mesures de similarités utilisées dans ce travail sont basées sur le principe de calcul de distances qui sépare les nœuds désirées du nœud racine R et la distance qui sépare le concept subsumant (CS) de ces nœuds (vu dans le chapitre précédent). A titre d'exemple la mesure de Wu and Palmer. Cette phase génère une distance minimale entre un concept et son nœud racine R.

### 5.2.1.3 Calcul des mesures de similarité

Lors de cette étape, nous appliquons les sept mesures de similarités (Tbk,Jaccard,Cosine,Dice, Euclidienne,Wu and palmer et Dennai) (voir annexe A) et cela d'un nœud donné vers n'importe quel autre nœud dans le graphe. Le nombre et le choix des mesures est personnel, juste pour voir laquelle des mesures de similarité génère les meilleurs résultats.

#### 5.2.1.4 Processus de partitionnement

Il est à noter que toutes les phases précédentes sont une préparation à la phase présente dans le sens ou toutes les matrices générées sont utilisées dans les algorithmes que nous avons développés pour partitionner nos ontologies OWL.

Pour ce faire nous avons développé trois algorithmes de partitionnement [SETTI AHMED et al., 15] à savoir :

- Un algorithme d'adaptation de l'approche K-means à notre domaine des ontologies, car les objets manipulés sont des concepts de l'ontologie. (Algorithme 2)
- Une extention de l'algorithme de l'approche traditionnelle K-means. L'algorithme est modifié pour intégrer les sept mesures de similarités pour calculer le poids du centre. (Algorithme 3).
- Un algorithme de révision et d'amélioration de K-means en
- ajoutant d'autres Paramètres telques le seuil et le K-max. Dans cet algorithme proposé nous générons un nombre maximum de K,
- mais à la fin nous allons obtenir un nombre minimum de partitions selon le seuil que nous avons fixé. (Algorithme 4). Mais avant ça on va voir le principe de l'algorithme K-means.

  La méthode de K-means ou algorithme de clustering par centre mobile permet d'effectuer un clustering d'un ensemble de données en K clusters. Chaque cluster est décrit par son centre. L'algorithme est décrit comme suit :
- Le nombre de clusters, le paramétre K, est fourni au départ,
- Un ensemble de K centres est choisi aléatoirement dans l'ensemble des données,
- Les K clusters sont formés en regroupant dans chaque centre l'ensemble des données plus proches du centre courant que de

tout autre centre.

- Le centre de chaque cluster est calculé et devient le nouveau centre.

- L'algorithme boucle alors sur l'étape précédente : Les données sont réaffectées en fonction de ces nouveaux centres et la condition d'arrêt est que les centres deviennent immobiles.

L'algorithme (2) est une adaptation de l'algorithme K-means dans le domaine des ontologies (les données sont les concepts de l'ontologie). Cependant, le problème de calcul du centre d'un cluster se pose car dans les méthodes usuelles de K-means, le centre d'un cluster se calcule à partir des cordonnées des données. Or la notion de cordonnées n'existe pas dans la structure d'une ontologie. C'est pourquoi nous avons proposé l'algorithme 2 détaillé.

***Algorithme 2: Algorithme d'Adaptation de l'approche K-Means***

**Inputs**

```
nb cluster , nb iter max,  n
Output  nb_cluster
BEGIN
1. h = 1
2. While (h < nb_iter_max) and (fin = false) Do //loop for the nb_iter_max
3. Begin
4. For i =0 to n Do //for each concept of ontology
5. Begin
6. For s =0 to nb_cluster Do
7. Begin
8. center[s].weight = |concept[i]. weight - center[s].weight |
9. End
10.min = center[0].weight
11.ind=0
12.For s = 0 to nb_cluster Do
13.Begin
14.If center[s].weight < min Then
15.Begin
16.min = center[s]. weight
17.ind=s
18.End
19.End
20.concept[i].cluster = ind
21.module[i] = ind
22.END
23.center_x =  0
24.nbx  =  0
25.cond =  true
26.For s = 0 to nb_cluster Do //compute the new center
27.Begin
28.center_x = 0
29.nbx = 1
30.For i = 0 to n Do
31.Begin
32.If concept[i].cluster = s Then
33.Begin
34.center x = center x + concept[i].weight
35.nbx = nbx +1
36.End
37.End
38.center[s].weight = center x / nbx   // div (center x, nbx).quot
39.If center1[s]. weight <> center[s] .weight Then cond = false
40.End
41.If cond Then fin = true;
42.Else
43.Begin
44.For i = 0 to nb_cluster Do
45.center1[i].weight = center[i]. weight
46.End
47.h = h +1
48.End While
49.Output= nb_cluster{C1,C2,……,Ck}
END
```

Le but est qu'on doit avoir dans la même partition des concepts reliés entre eux avec forte dépendance.

***Algorithme 3 : Extension de l'algorithme de l'approche K-Means :***

***Application des mesures de similarités***

**Inputs:** nb_cluster, nb_iter_max, //Similarity_of_Tbk(Concept_i,Concept_j) // our contribution
**Output:** nb_cluster
**BEGIN**
1. h = 1
2. **While** ((h less then nb_iter_max) and (fin equal false)) Do //loop for nb_iter_max
3. **Begin**
4. **For** each i equal 0 **to** n **Do** //for each concept of ontology
5. **Begin**
6. **For** each s equal 0 to nb_cluster Do //Similarity measure
7. **Begin**
8. center(s).weight = Similarity Tbk(Concept i,Concept s)// our contribution
9. **End**
10. min = center(0).weight
11. ind=0
12. **For** each s equal 0 to nb_cluster Do
13. **Begin**
14. **If** (center(s).weight less then min) Then
15. **Begin**
16. min = center(s).weight ; ind=s
17. **End**
18. **End**
19. concept(i).cluster = ind // assign the minimum
20. module(i) = ind
21. **End**
22. center_x = 0 ; nbx = 0 ; cond = true
23. **For** each s equal 0 to nb_cluster Do //compute the new center
24. **Begin**
25. center_x = 0 ; nbx = 1
26. **For** each i equal 0 to n Do
27. **Begin**
28. **If** (concept(i).cluster equal s) Then
29. **Begin**
30. center_x = center_x + concept(i).weight
31. nbx = nbx +1
32. **End**
33. **End**
34. center(s).weight = center_x / nbx // div (center_x, nbx).quot
35. **If** (center1(s).weight Not equal center(s).weight) Then cond = false
36. **End**
37. **If** cond Then fin = true
38. **Else**
39. **Begin**
40. **For** each i equal 0 to nb_cluster Do
41. **Begin**
42. center1(i).weight = center(i).weight
43. **End**
44. **End**
45. h = h + 1
46. **End While**
47. **Output** = nb_cluster {C1,C2,……,Ck}.
**END**

En plus de l'algorithme (2), nous cherchons à éliminer les partitions non significatives (partitions contenant peu de concepts dans le cas par exemple de deux ou trois).

- Dans l'algorithme 3 nous avons introduit la notion de mesures de similarités pour calculer la distance de similarité. Il est à noter que les sept mesures : (Jaccard,Cosine,Dice,Euclidienne,

(Tbk, Wu and Palmer et Dennai) voir (annexe A) sont utilisées dans cet algorithme

Pour calculer un nombre optimal de clusters nous avons introduit l'algorithme 4. Si au moins une partition contenant un nombre de concept inférieur à la valeur du seuil qu'on a introduit, donc cette partition sera écartée et doit être déplacée avec d'autres partitions et le nombre K-max sera décrémenté à K-max-1, sinon le nombre de partition est affiché avec le contenu et la taille de chaque partition. (Algori4).

***Algorithme 4 : Algorithme de révision de l'approche K-Means***

```
Inputs
max_ nb_clusterx, threshold

Output
optimal_number_of_cluster
BEGIN
1. Algorithm K means with Max  nb cluster
2. Compute the number of clusters
3. If  at last  one cluster has size < threshold Then
4. Begin
5. max_ nb_cluster = Max_ nb_cluster -1
6. goto  1
7. end
8.  Else Visualize  the  number and the size of clusters
9. Output=optimal  nb cluster{C1,C2,……,Ck}
END
```

#### 5.2.1.5 Processus de validation

Après le processus de partitionnement, l'ensemble des partitions générées seront validées en utilisant par exemple un raisonneur comme Pellet, pour voir est ce qu'il ya une perte de sémantique après partitionnement ou pas.

## 5.3 Conclusion

Dans ce chapitre, nous avons présenté en détail les étapes suivies pour arriver à la réalisation de notre objectif de partitionnement d'ontologies à base de clustering. Nous avons opté pour une architecture fonctionnelle afin de construire notre futur système.

L'étude théorique nous a permis de mettre en évidence les étapes nécessaires et les différents composants du système à implémenter.
La notion de similarité est mise en avant dans plusieurs domaines d'activités liés à l'ingénierie des ontologies tels que l'apprentissage, l'alignement ou encore le peuplement d'ontologies. La notion de similarité intervient ainsi dans de nombreux domaines de l'intelligence artificielle comme l'apprentissage automatique, la recherche d'information, le raisonnement. C'est pourquoi nous avons opté dans notre approche de partitionnement à base de clustering sur quelques mesures de calcule de similarités entre concepts d'une ontologie.
dans l'approche des arcs [RADA 89] [WU et PALMER 94] [LEE 93] [EHR 04]. Le principe de calcul de similarité avec cette approche est basé sur l'idée suivante : plus le chemin entre deux nœuds est court plus ils sont plus semblables. L'autre notion qui caractérise cette deuxième approche est que les arcs d'une taxonomie représentent des distances uniformes, par conséquent, cette approche présente l'inconvénient que tous les liens sémantiques possèdent le même poids ce qui impose des difficultés au niveau de la définition et du contrôle des distances des liens. Est ce n'est la qu'un exemple parmi les points faibles de cette mesure ? Pour cela nous avons opté pour la mesure sémantique de Dennai qui est une amélioration de celle de Wu et Palmer avec d'autres mesures pour voir leur efficacité en les appliquant dans un algorithme de partitionnement.
Dans le chapitre suivant, nous présentons la réalisation de notre système de partitionnement des ontologies écrites en OWL en appliquant en premier lieu l'algorithme standard K-means et ensuite notre algorithme révisé et amélioré en introduisant les sept mesures de similarités.

# Chapitre 6

# Implémentation et évaluation du SysOPCBA

*Nous passons notre vie à apprendre des choses, mais nous trouvons toujours des exceptions et des erreurs. La certitude semble demeurer constamment hors de portée. Il faut donc que nous prenions quelques risques si nous ne voulons pas rester paralysés par la lâcheté. Cependant, pour éviter tout accident, nous devons accumuler deux types de connaissances supplémentaires :*
*Nous cherchons des "îlots de cohérence" dans les limites desquels le raisonnement ordinaire semble sûr.*
*Nous nous efforçons également de trouver et de marquer les frontières hasardeuses de ces domaines.*
*Marvin Minsky (La société de l'esprit)*

## 6.1 Introduction

Nous avons présenté dans le chapitre précédent notre approche sur le partitionnement des ontologies OWL. Dans ce chapitre, nous allons présenter l'implémentation de notre système de partitionnement des ontologies «SysOPCBA» « Ontology Partitioning : Clustering Based Approach ». Nous allons tout d'abord présenter l'environnement de développement ainsi que les différents outils utilisés, puis nous donnons une description de notre système «SysOPCBA » à travers quelques fenêtres de capture qui représentent les interfaces de ce dernier, qui sont conçues de manière à être conviviales et simples d'utilisation. Cette étape nous a aussi permis de nous familiariser avec les outils utilisés pour le développement de notre système. Ensuite une étude expérimentale est réalisée en appliquant notre système sur des ontologies réelles. Nous évaluons finalement les résultats acquis à travers les ontologies expérimentées en utilisant deux métriques d'évaluation à savoir : la cohésion et la densité et nous conclurons ce chapitre.

## 6.2 Environnement de développement

Avant de commencer l'implémentation de la phase conceptuelle de notre système de partitionnement, nous allons tout d'abord spécifier les outils utilisés qui nous ont semblés être un bon choix de part les avantages qu'ils offrent.

### 6.2.1 Choix du Système d'exploitation

Notre application a été développée sous le système d'exploitation Windows 7, mais comme elle est développée en langage java, elle peut être intégrée dans n'importe quel autre système d'exploitation supportant la machine virtuelle java (Linux, …).

### 6.2.2 Matériel et langage de programmation utilisés

Toutes les expériences et les tests réalisés dans ce mémoire sont faits sur une machine portable Intel Core TM 2 Duo 2.00 GHz avec 2 GB de mémoire RAM sous Windows 7 et Java NetBeans IDE 7.1.1.

## 6.3 Description du système SysOPCBA

L'interface homme/machine représente l'élément clé dans l'utilisation de tout système informatique. Les interfaces de notre système de partitionnement des ontologies sont conçues de manière à être simples, naturelles, et de compréhension et d'utilisation faciles. Pour pouvoir utiliser notre application l'utilisateur doit d'abord charger le fichier de l'ontologie OWL après validation par protégé (toutes les ontologies utilisées ici sont validées par l'outil protégé).

### 6.3.1 Interface principale

L'interface illustrée par la figure ci-dessous représente l'interface principale de notre système de partitionnement SysOPCBA .

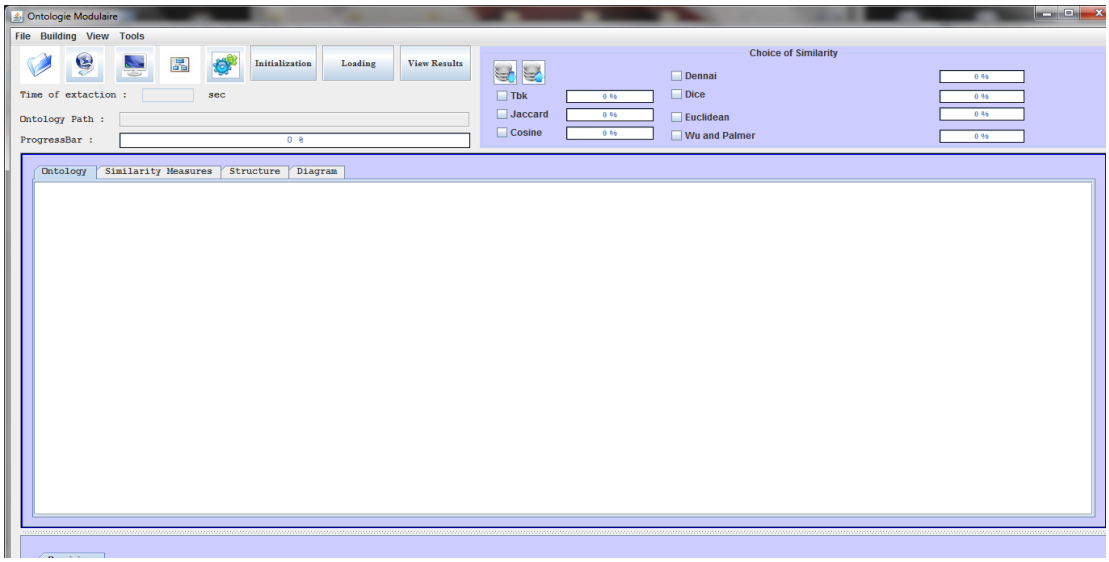



**Figure .6.1 :** Copie d'écran de l'interface principale de notre système.

Après chargement du fichier .owl de notre ontologie (voir figure 6.2), on lance le module d'analyse et d'extraction d'ontologie qui permet d'extraire les informations utiles de notre ontologie telques les concepts, les relations, les propriétés et les axiomes (voir annexe B). et d'éliminer tout ce qui est bruit tel que les tags. Car dans un fichier OWL, les constructeurs sont délimités par des balises. Pour réaliser cette étape nous avons utilisé un algorithme de cleansing.

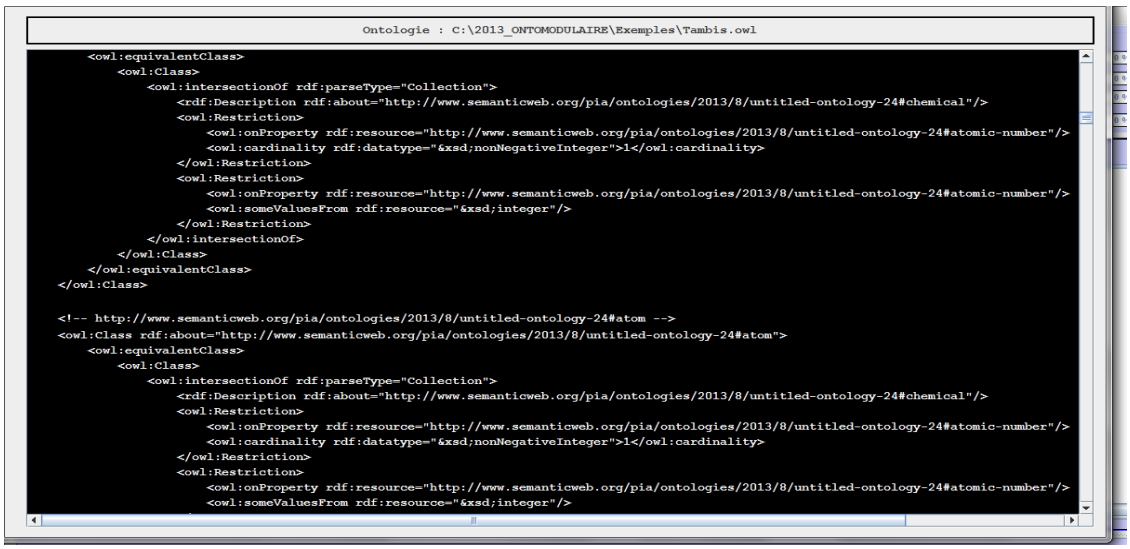



**Figure 6.2 :** Chargement de l'ontologie

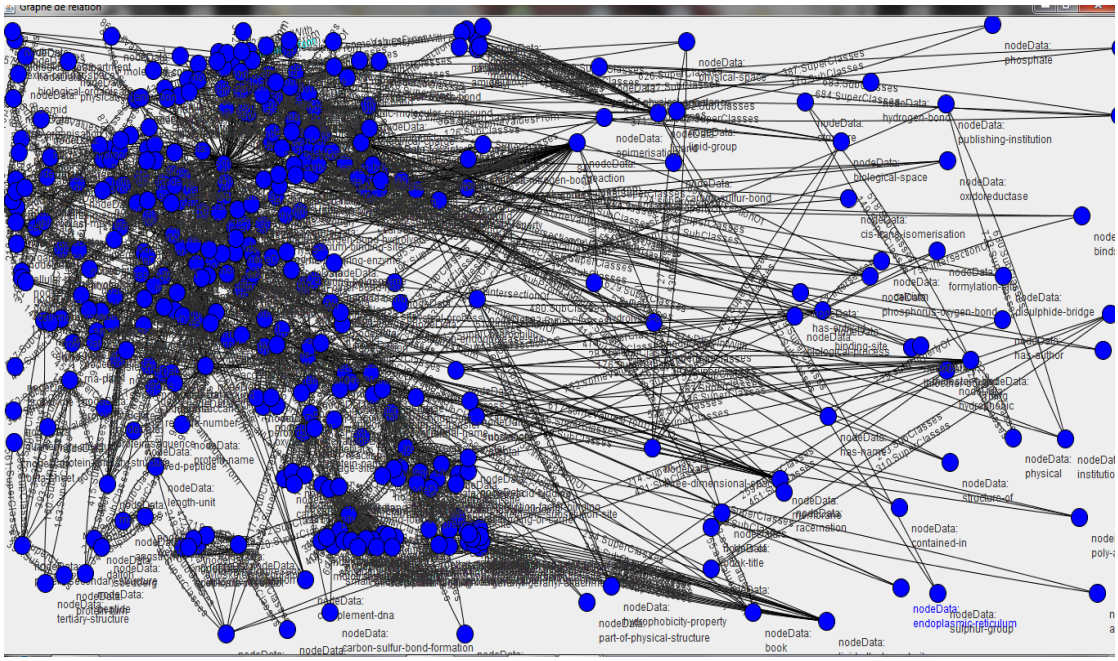


**Figure .6.3** Copie d'écran de l'interface de notre ontologie OWL sous forme de graphe de nœuds.

## 6.4 Expérimentations et évaluation

Afin de tester la fiabilité et de montrer la faisabilité de notre approche, nous avons développé le système de partitionnement SysOPCBA dont les fonctionnalités principales ont été présentées dans les sections précédentes. Ce système a besoin d'être expérimenté sur des ontologies réelles afin de vérifier ses fonctions.

### 6.4.1 Expérimentations

Nous avons effectué deux types d'expériences. Dans la première expérience nous avons utilisé l'ontologie TAMBIS contenant 393 concepts reliés entre eux par des relations de subsumption et 597 axiomes. Ces entités décrivent des entités moléculaires et biologiques. La deuxième expérience est réalisée sur une ontologie de photography contenant plus de 185 classes, 268 relations et différents types d'objets et attributs.

Il est à noter que le temps d'analyse et d'extraction des constructeurs des deux ontologies par notre système est réalisé entre 30 and 20 second respectivement.

Au départ notre première ontologie Tambis était validée par le raisonneur Pellet, puis par notre système d'extraction, nous avons obtenu les mêmes résultats en termes de classes, de propriétés et d'axiomes ce qui prouve la puissance et la performance de notre système. Il n' ya pas une perte d'information de l'ontologie voir (Figure 6.4.).

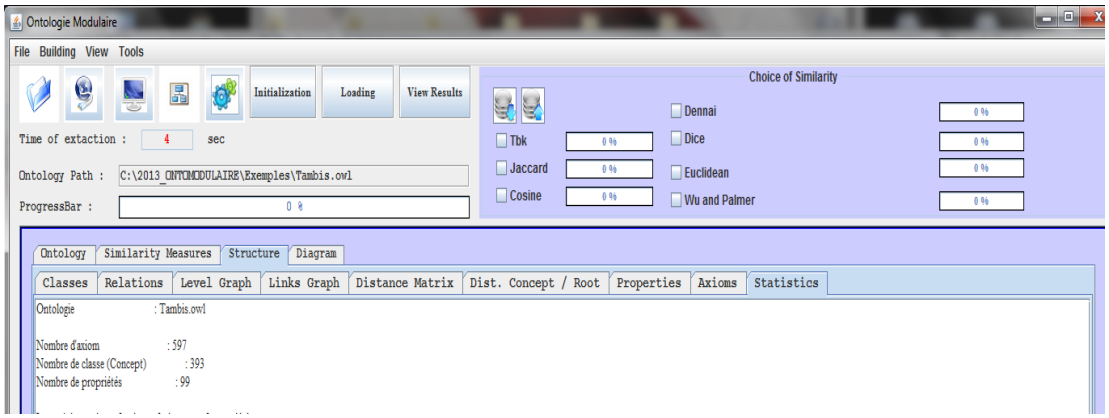



**Figure.6. 4 :** Capture d'écran de l'interface de visualisation de l'ontologie Tambis après validation par le raisonneur Pellet

Nous décrivons ensuite les processus d'expérimentation et évaluons les résultats acquis à travers l'expérimentation et l'utilisation de deux métriques d'évaluation. Nous avons utilisé deux métriques dans l'évaluation de notre système de partitionnement, à savoir la cohésion et la densité.

La cohésion représente le dégrée de dépendance des classes OWL qui sont sémantiquement et conceptuellement reliées par des propriétés dans l'ontologie (relatedness of elements in ontologies.)

Haute cohésion meilleurs modules [WAYNE et al., 74].

La densité est définie comme étant la présence des clusters des classes avec plusieurs relations non taxonomiques entre plusieurs relations qui se trouvent entre eux.

**Les résultats expérimentaux**

Dans cette section nous présentons quelques résultats obtenus.

Les premières expériences sont faites sur l'ontologie Tambis avec un K-max=40 et un seuil =5. Les résultats obtenus sont représentés dans les figures (Figure 6. 5) et (Figure 6. 6) respectivement et résumés dans la table 1.

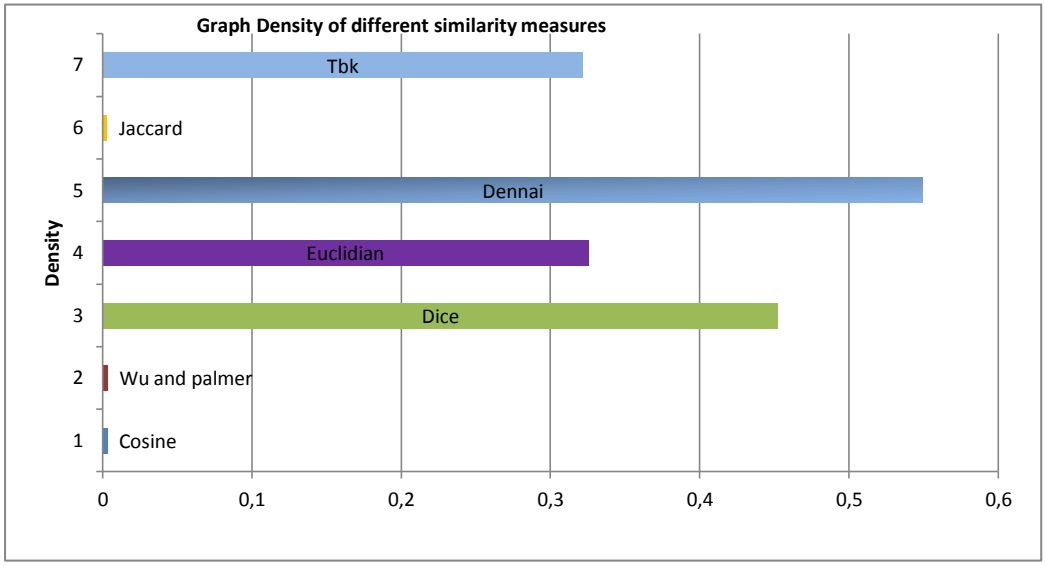



**Figure 6. 5 :** Graphe de densité pour les sept mesures de similarité pour Tambis

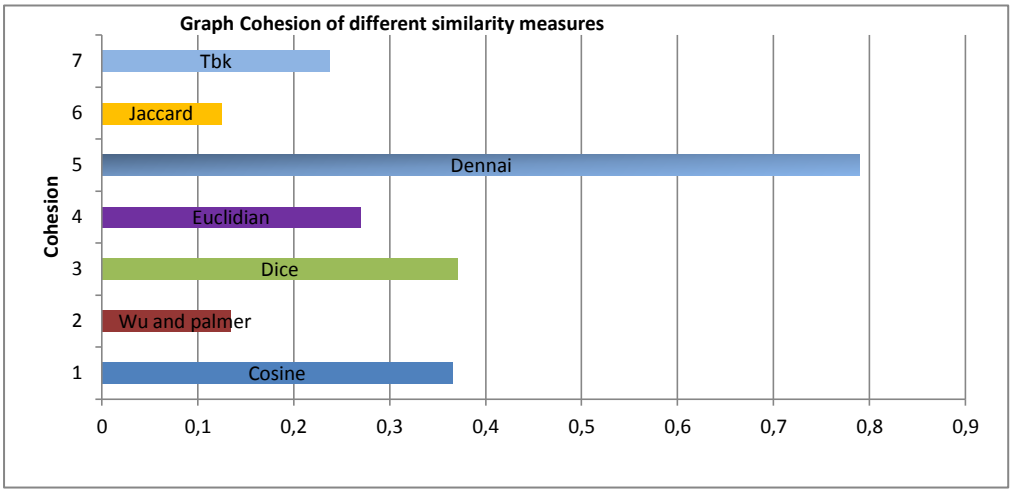



**Figure.6. 6 :** Graphe de cohésion pour les sept mesures de similarité pour Tambis

Dans la Table6.1. Il s'agit d'une autre expérimentation pour l'ontologie Tambis pour un nombre K-max=40 et un seuil=5.

| Mesure | Nombre de partition | Seuil | Nombre d' itération | Cohésion | Densité |
|---|---|---|---|---|---|
| **Cosine** | 2 | 5 | 100 | 0,365 | 0,003 |
| **Wu/palmer** | 2 | 5 | 100 | 0,133 | 0,003 |
| **Dice** | 3 | 5 | 100 | 0,370 | 0,452 |
| **Euclidian** | 3 | 5 | 100 | 0,269 | 0,325 |
| **Dennai** | 4 | 5 | 100 | 0,790 | 0,549 |
| **Tbk** | 5 | 5 | 100 | 0,237 | 0,321 |
| **Jaccard** | 6 | 5 | 100 | 0,124 | 0,002 |

**Table 6.1 :** Expérimentation de l'ontologie Tambis pour K-max=40 et un seuil=5.

Une autre expérience est faite sur une deuxième ontologie Photography pour K-max=50 et un seuil =7. Les résultats obtenus sont résumés dans la table Table 6.2.

| Mesure | Nombre de partition | seuil | Nombre d' iteration | Cohésion | Densité |
|---|---|---|---|---|---|
| Cosine | 4 | 7 | 100 | 0,365 | 0,005 |
| Wu/palmer | 3 | 7 | 100 | 0,032 | 0,203 |
| Dice | 4 | 7 | 100 | 0,321 | 0,459 |
| Euclidian | 4 | 7 | 100 | 0,321 | 0,321 |
| Dennai | 6 | 7 | 100 | 0,856 | 0,655 |
| Tbk | 2 | 7 | 100 | 0,137 | 0,004 |
| Jaccard | 3 | 7 | 100 | 0,325 | 0,005 |

**Table 6.2 :** Expérimentation de l'ontologie Tambis pour K-max=50

En outre, une autre expérience a été réalisée sur la même ontologie Tambis avec K-max=30, 20 et un seuil =10, 7 respectivement. Les résultats obtenus sont interprétés dans les Figures 6.7 et 6.8 respectivement.

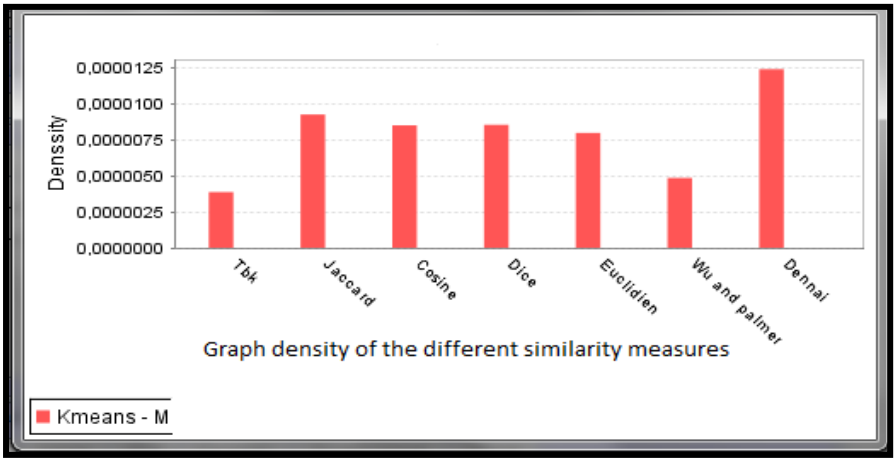



**Figure 6.7** : Graphe de densité pour les différentes mesures de similarité pour Tambis avec K-max=30 et seuil 10

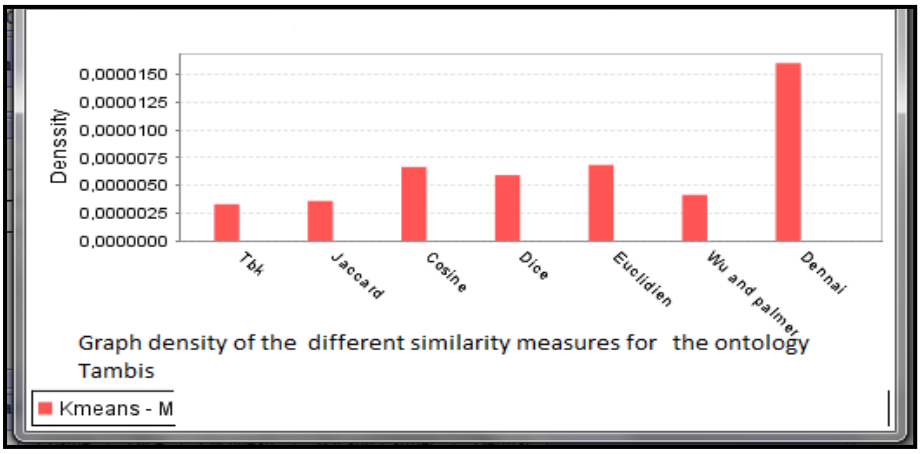



**Figure 6. 8** : Graphe de densité de l'ontologie Tambis pour les différentes mesures de similarités avec K-max=20 et seuil=7

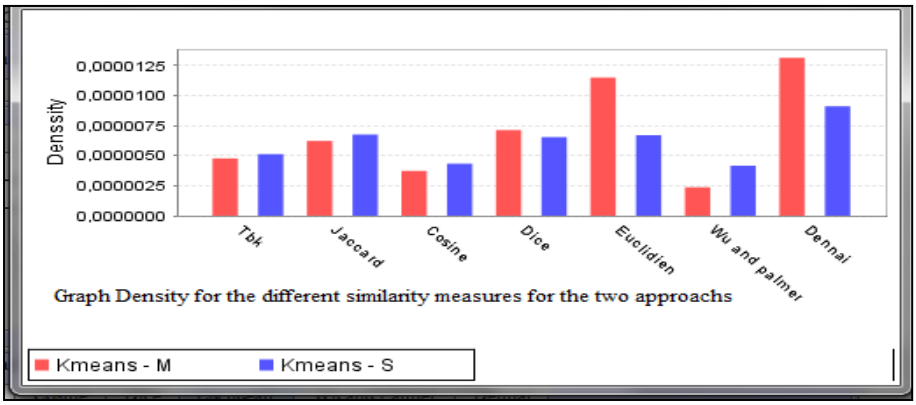



**Figure 6.9** Graphe de densité de l'ontologie Tambis pour les deux approches

La figure **6. 9** représente une autre expérience sur l'ontologie Tambis.

Dans cette expérience nous comparons l'algorithme standard et traditionnel de l'approche K–means (Kmeans_S) avec notre algorithme révisé (Kmeans_M) pour un nombre de K_max=30 et un seuil de 10.

Comme nous avons confirmé dans la problématique que l'application de l'algorithme K-means standard donne un ensemble non significatif de partitions avec seulement deux ou trois concepts dans des partitions et dans certains cas nous obtenons des clusters ne

contenant que le centre du cluster. En plus et après l'implémentation de K-means nous avons constaté que le contenu des partitions change après chaque exécution, pourtant nous gardons les mêmes valeurs de paramètres.

Notre algorithme révisé résout ces problèmes le fait que nous avons introduit dans l'algorithme un nouveau paramètre qui est le seuil.

Rappelons que le seuil a un impact sur la taille de la partition. De ce fait, un seuil faible entraînerait systématiquement un *Nombre de concepts* bas.

Nous avons refait l'exécution pour la quatrième fois et nous avons conclu que les résultats ne changent pas. En outre, et dans les quatre expériences effectuées la mesure de similarité sémantique de Dennai donne toujours les meilleurs résultats coté densité et cohésion.

### 6.4.2 Discussion

Pour l'évaluation de la qualité des partitions générées par notre approche proposée, nous avons utilisé deux métriques à savoir : la densité et la cohésion.

Pour comparer les deux approches nous exécutons les deux approches pour la 4ème fois avec les deux ontologies Tambis et Photography.

Nous avons confirmé que l'algorithme K-means génère des clusters (partitions) dynamiques et dans quelques cas nous avons trouvé des partitions complètement vides ou seulement avec un seul concept qui représente le centre du cluster, dans certains cas deux ou trois concepts seulement. Cela n'a aucun sens et aucune sémantique.

En contrepartie notre algorithme amélioré donne des partitions fixes en plus aucune partition n'est vide dans les 4 exécutions.

En outre, la mesure de Dennai donne dans la majorité des cas des partitions de fortes dépendances en termes de densité et cohésion en

comparant avec les six autres mesures car c'est une amélioration de la mesure de Wu et palmer.

## 6.5 Conclusion

Dans ce chapitre, nous avons présenté le cadre applicatif de nos travaux. Ensuite, une présentation été faite du système que nous avons développé comme support à notre approche. Enfin nous avons présenté et discuté quelques expérimentations mises en place. Le but principal de cette implémentation, est d'évaluer nos propositions, de tester la faisabilité de notre approche améliorée, et de démontrer que l'approche proposée peut contribuer au processus de partitionnement d'ontologie en comparant avec celle standard.

Dans le chapitre suivant nous conclurons notre mémoire en ouvrant quelques perspectives et pistes de recherche dans le domaine de la modularisation des ontologies.

# Chapitre 7
# Conclusion générale

*Il est peu de personnes qui ne se soient amusées, à un moment quelconque de leur vie, à remonter le cours de leurs idées et à rechercher par quels chemins leur esprit était arrivé à de certaines conclusions. Souvent cette occupation est pleine d'intérêt et celui qui l'essaye pour la première fois est étonné de l'incohérence et de la distance, immense en apparence, entre le point de départ et le point d'arrivée.*

*Edgar Poe - traduction de Charles Baudelaire*
*(Double assassinat dans la rue Morgue)*

## 7.1 Conclusion

Ainsi, ce mémoire touche à sa fin.

Dans les deux chapitres 2 et 3, nous nous sommes intéressés à ce qu'était une ontologie. Pour cela, nous sommes partis des origines philosophiques du terme pour ensuite définir son sens en ingénierie des connaissances. Ensuite, nous avons étudié la manière de concevoir et de réaliser une ontologie en ingénierie des connaissances en énumérant ses composants et en détaillant les différentes méthodes de construction. Nous avons également abordé deux points plus techniques en présentant d'une part, le standard OWL du W3C, le langage plus répandu pour représenter des ontologies, et d'autre part, en exposant les différents logiciels permettant d'éditer des ontologies. Une bonne partie du quatrième chapitre a été consacrée à la modularisation des ontologies, ses concepts de base et ses objectifs ainsi que les différentes techniques de modularisation. Par la suite nous avons évoqué le partitionnement à base de clustering ; aussi les fondements théoriques et les concepts de bases ont été exposés.

Enfin, dans le chapitre 5, nous avons proposé une nouvelle approche de partitionnement des ontologies à base de clustering, qui est bâtie sur des mesures de calcul de similarité. Une nouvelle mesure est appliquée pour la première fois. Cette approche est une adaptation, une amélioration et une révision de l'approche K-means.

Sur la base de cette approche, nous avons implémenté un outil de partitionnement que nous avons baptisé SysOPCBA. Des expérimentations ont été menées sur deux ontologies réelles. Afin de qualifier le travail effectué et de démontrer l'efficacité de notre approche dans la production des modules (partitions) de bonne qualité, des tests sur les différentes partitions générées ont été effectués en utilisant sept mesures de similarités via l'outil de partitionnement SysOPCBA proposée.

Finalement nous avons discuté les résultats obtenus en soulignant les points positifs de cette expérimentation, mais aussi les points faibles de notre approche.

## 7.2 Perspectives

Dans cette section, nous présentons les possibilités d'étendre et d'approfondir les recherches réalisées dans cette thèse en vue d'un travail futur. Face aux limites actuelles de SysOPCBA et dans la perspective d'un développement futur, des améliorations pourraient être faites tant au niveau de l'algorithme qu'au niveau de l'implémentation de l'outil.

Ainsi, les points suivants devraient être abordés :

- Proposer une approche empirique permettant de fixer le seuil sémantique. L'expert se doit d'effectuer un certain nombre de tests afin de trouver le seuil idéal, d'où la nécessité d'élaborer un protocole suivant lequel ces tests devront être menés.
- S'inspirer des méthodes d'exploration de graphe basées sur des heuristiques pour améliorer les performances de l'approche proposée.
- Appliquer l'algorithme de partitionnement proposé dans l'alignement des ontologies (matching) pour répondre aux requêtes (Query answing).
- Appliquer notre approche sur d'autres ontologies réelles de grandes tailles et plus complexes.

De plus, et à fin de juger de la qualité des partitions générées, nous souhaitons :

- Appliquer d'autres métriques d'évaluation telles que : la Précision, le Recall, et la F-mesure.

- Tester également d'autres métriques pour mesurer la qualité de modularisation, tel que le couplage.
- Valider les partitions générées par un raisonneur tel que Pellet.

Ce projet de recherche a été une expérience très enrichissante qui nous a permis d'apporter des tentatives de solution à un problème d'actualité, et de poser des problématiques intéressantes pour une étude future.

# Bibliographie

# *Bibliographie*


**[AMARD, 07]** Florence AMARDEILH. « Web Sémantique et Informatique Linguistique : propositions méthodologiques et réalisation d'une plateforme logicielle. » Thèse de doctorat, Université Paris, 2007.

**[ARPI et al., 98]** Arpirez J., Gómez-Pérez A., Lozano A. et Pinto S. « (ONTO) 2Agent: An ontology based WWW broker to select ontologies. » Paper presented at *Workshop on Applications of Ontologies and PSMs*, Brighton, England, 1998.

**[ARPI et al., 01]** ARPIREZ J., CORCHO O., FERNANDEZ-LOPEZ M., AND GOMEZ-PEREZ A. (2001). WebODE : a Workbench for Ontological Engineering. In First international Conférence on Knowledge Capture (K-CAP'Ol), pages 6-13, Victoria, Canada, 21-23 Octobre 2001. ACM

**[AZOUAOU et al., 05]** F. Azouaou, T. Cao, S. Dehors, C. Desmoulins, R. Dieng-Kuntz, and C. Farou-Zucker. Les outils du web semantique et du e-learning. In *Plate-forme Afia*, May 2005.

**[BAADER, 03]** F. Baader, D. Calvanese, D.McGuinness, D. Nardi, P.F. Patel-Schneider.: The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

**[BACHIMONT, 00]** Bachimont B. «Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances.» In *Ingénierie des connaissances. Évolution Récentes et nouveaux défis,* Charlet J., Zacklad M., Kassel G. et Bourgault D., Paris: Eyrolles, 2000.

**[BAGET et MUGNIER, 02]** Baget, J.F. and Mugnier, M.L. (2002).Extensions of Simple Conceptual Graphs: the Complexity of Rules and Constraints. *Journal of Artificial Intelligence Research (JAIR),* vol. 16, pages 425-465. http://www.jair.org/papers/paper918.html.

**[BANEYX ,07]** Baneyx, A. (2007). Construire Une Ontologie De La Pneumologie : Aspects Théoriques, Modèles Et Expérimentations. Thèse de doctorat, UNIVERSITÉ PIERRE ET MARIE CURIE.

**[BAR et al., 99]** R.Baeza-Yates et B.Ribeiro-Neto. Modern Information Retrieval. ACM Press; Addison-Wesley: New York; Harlow, England; Reading, Mass., 1999.

**[BAR et RIBEIRO, 99]** R.Baeza-Yates et B.Ribeiro-Neto. Modern Information Retrieval. ACM Press; Addison-Wesley: New York; Harlow, England; Reading, Mass., 1999.

**[BENA, 05]** Ahcene BENAYACHE. « Construction d'une mémoire organisationnelle de formation et évaluation dans un contexte e-learning : le projet MEMORAE. » Thèse de doctorat, université de technologie de Compiègne, 2005.

**[BENER et al., 01]** M. Benerecetti, P. Bouquet, C. Ghidini.: On the dimensions of context dependence: partiality, approximation, and perspective. In Proc. 3rd International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), volume 2116 of Lecture notes in computer science. Dundee (UK), pp 59–72, 2001.

**[BEN HEBI, 12]**Ben Hebireche H., Proposition d'une ontologie formelle pour la modélisation et la simulation intelligente. Thèse de Magister, 2012, Université Kasdi Merbah Ouargla.

**[BENOIT, 05]** V. BENOIT :Modularisation et intégration d'ontologies dans le domaine de la bioinformatique, thèse de Master, univrsité du QUEBEC à Montréal. 2005

**[BERNA, 96]** Bernaras A., Laresgoiti I., Corera J. « Building and Reusing Ontologies for Electrical Network Applications.»,*In Proceedings of the European Conference on Artificial*

*Intelligence (ACAI' 96)*. ECAI96. Publisher: John Wiley & Sons, 1996.

**[BLAZ, 98]** Blazquez M., Fernandez M., Garcia-Pinar J. M. et Gomez-Perez A. «Building Ontologies at the Knowledge Level using the Ontology Design Environment» In Proceedings of the Banff Workshop on knowledge Acquisition for Knowledge Acquisition for knowledge-based Systems, KAW, Banff, Canada,1998.

**[BORG, 96]** Borgo S., Guarino N. et Masolo C. « *Stratified Ontologies: the case of physical objects.* » Paper presented at the ECAI96. Workshop on Ontological Engineering, Budapest, 1996.

**[BOR et GIUNCH, 07]** Borgida, A. et Giunchiglia, F. (2007). Importing from functional knowledge basesa preview. Dans B. C. Grau, V. Honavar, A. Schlicht , et F. Walter (dir.) . *Proceedings of th e 2nd International Workshop on Modular Ontologies, WoMO 2007,* volume 315 de *CEUR Vvorkshop Proceedings,* Whistler, Canada.

**[BORST, 97]** Borst W. N. «Construction of Engineering Ontologies.» Center for Telematica and Information Technology, University of Tweenty, Enschede, NL, 1997.

**[BOUQUET et al., 04]** P. Bouquet, M. Ehrig, J. Euzenat, E. Franconi, P. Hitzler, M. Krotzsch, L. Serafini, G. Stamou, Y. Sure, S. Tessaris.: Specification of a common framework for characterizing alignment. Deliverable D2.2.1, Knowledge web NoE, 2004

**[BRACH et al., 85 ]**.R.J. Brachman, J.G. Schmolze.: An overview of the KL ONE knowledge Representation System. Cognitive Science, 9(2): pp 171–216, April 1985.

**[BRAY et al., 00]** T. Bray, J. Paoli, C.M. Sperberg-McQueen and E. Maler, Editors. Extensible Markup Language (XML) 1.0, Second Edition, World Wide Web Consortium. 6 October 2000. http://www.w3.org/TR/REC-xml.

**[CHA, 04]** Charlet J., Bachimont B. et Troncy R. « Ontologies pour le Web Sémantique.» In Le Web sémantique, CHARLET J., LAUBLET P. et REYNAUD C. (Ed.), Hors série de la Revue Information - Interaction - Intelligence (I3), 4(1), Cépaduès, Toulouse, 2004.

**[CHAU, 98]** CHAUDRHRI VK., FARQUHAR A., FIKES R., RICE JP. "Open Knowledge Base Connectivity 2.0.3. Technical Report. [en ligne] sur :<http://www.ai.sri.com/~okbc-2.0.3.pdf>, (consulté le 2 Novembre 2006)

**[CHE et al., 04]** F. Chen, A. Farahat et T. Brants. Multiple Similarity Measures and Source-Pair Information in Story Link Detection. In Proceedings of HLT,2004.

**[CHENG, 97]** Cheng Hian Goh. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources. *Phd, MIT*, 1997.

**[CHOURAB, 09].**Olfa Chourabi.Un cadre ontologique générique de modélisation, de capitalisation et de partage de Connaissances Métiers Situées en Ingénierie Systéme. Computer Science [cs]. Conservatoire National Des Arts et Métiers, Péaris; Ecole Nationale des Sciences de l'Informatique, Tunisie,2009. French.

**[CORC, 00]** CORCHO O., GOMEZ-PEREZ A. "A Roadmap to Ontology Specification Languages". In Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00), Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp 80-96, 2000.

# *Bibliographie*


**[DAN BRICHEL et al., 04]** Dan Brickley and R.V. Guha, Editors. RDF Vocabulary Description language1.0:RDF Schéma, W3C Recommendation, 10 February2004.http://WWW.W3.org/TR/ rdf-schema/

**[D'AQUIN et al., 06]** D'Aquin, M., Sabou, M. et Motta, E. (2006). Modularization : a key for the dynamic selection of relevant knowledge components. Dans *Proceedings of the ISWC 2006* Workshop on Modular Ontologies, Athens, Georgia, USA.

**[D'AQUIN et al., 07]** d'Aquin, M., Doran, P. , Motta, E. et Tamma, V. (2007). Towards a parametric ontology modularization framework based on graph transformation. Dans B. C.

**[D'AQUIN et al., 09]** d'Aquin, M., Schlicht, A. , Stuckenschmidt, H. et Sabou, M. (2009) . Criteria and evaluation for ontology modularization techniques. Dans *Modular Ontologies,* volume 5445 de *Lecture Notes in Computer Science,* 67- 89. Springer-Verlag.

**[DAML,02]**DAML.(2002):DarpaAgentMarkupLanguage. http://www.daml.org/2000/10/daml-ont.html.

**[DAN et al., 01]** Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL Reference Description. *W3C Note* 18 December 2001. http://www.w3.org/TR/daml+oil-reference.

**[DAN BRICHEL et al., 04**] Dan Brickley and R. V.Guha, Editors. RDF Vocabulary Description Language 10: RDF Schéma ,W3C Recooendation, 10 February 2004.http://www.w3.org/TR/rdf-schema/

**[DEN et BENS, 13]** Abdeslem Dennai, Sidi Mohamed Benslimane**.** Toward an Update of a Similarity Measurement for a Better Calculation of the Semantic Distance between Ontology Concepts. The Second International Conference on Informatics Engineering & Information Science (ICIEIS2013). Kuala Lumpur, Malaysia, November 12-14, 2013

**[DING et SAN, 13]** G. Ding, T. Sun, Y. Xu, —Multi-Schema Matching Based On Clustering Techniques. In the 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). 2013.

**[DORAN et al., 07]** ] Doran P., Tamma V., Iannone L. (2007). Ontology module extraction for ontology reuse: An ontology engineering perspective. In Proceedings of the sixteenth acm conference on conference on information and knowledge management, p. 61–70. New York, NY, USA, ACM.

**[DORAN et al., 08]** Doran, P**.**, Palmisano, I. et Tamma, V. (2008). Somet : algorithm and tool for sparql based ontology module extraction. Dans U. Sattler et A. Tamilin ( dir.) . *Proceedings of the 2008 ESWC International Workshop on Ontologies : Reasoning and Modularity (WORM-08),* volume 348 de *CEUR Workshop Proceedings,* Tenerife, Spain.

**[DOMINGUE, 98]** Domingue J. : Tadzebao and WebOnto : Discussing, Browsing, and Editing Ontologies on the Web. In 11th Banff Knowledge Acquisition Workshop (KAW'98), Banff, Canada, 1998.

**[DUPONT et al., 06]** Dupont, P., Callut, J., Dooms, G., Monette, J. et Deville, Y. (2006). *Relevant subgraph extraction from random walks in a graph.* Rapport technique, Université catholique de Louvain, UCL/ INGI.

# *Bibliographie*


**[EHRI et al., 04]** M.Ehrig, P.Haase, M.Hefke et N.Stojanovic. Similarity for ontology-a comprehensive framework. In Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, 2004.

**[ESPOSITO et al.,?? ]** F. Esposito, N. Fanizzi, and C. d'Amato, ―Conceptual Clustering Applied to Ontologies by means of Semantic Discernability‖, technical report, unpublished.

**[EUZN et SHVAIKO, 07 ]** J. Euzenat, P. Shvaiko.: Ontology Matching. ISBN 978-3-540-49611-3, Springer- Verlag Berlin Heidelberg 2007

**[FARQ et al., 00]** FARQUHAR A., FIKES R. & RICE J. « Ontolingua server : a tool for collaborative ontology construction. » In International journal of Human- Computer studies, 2000.

**[FENSEL et al, 00]** D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein. Oil in a nutshell. In *12th International Conference on Knowledge Engineering and Knowledge Management EKAW2000*, Juanles-Pins, France, 2000.

**[FENSEL, 01]** D. Fensel.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer, 2001.

**[FERNA, 97]** Fernandez M., Gomez-Pérez A., Juristo N. «METHONTOLOGY: From Ontological Art Towards Ontological Engineering.» *Proceedings of the AAAI- 97 Spring Symposium Series on Ontological Engineering*, Stanford, CA, USA,1997.

**[FURST, 04]**Frédéric FÜRST,(2004). Contribution à l'ingénierie des ontologies : une méthode et un outil d'opérationnalisation. Thèse d'Informatique, Université de Nantes.

**[GAND, 02]** Gandon F., Dieng R., Corby O., Giboin A. (2002) « Web Sémantique et Approche Multi-Agents pour la Gestion d'une Mémoire Organisationnelle Distribuée. », *In Actes de IC'*
*2002*, Rouen, France, pp. 15-26.

**[GARI, 97]** GuarinoN. (1997). Some organizing principles for a unified top-level ontology. In *National Conference of the American Association on Artificial Intelligence (AAAI)*, p. 57–63, Stanford, United-States.

**[GUARI, 98]** Guarino , N. (1998). Formal onthology in information systems. Dans *Proceedings of the First International Conference (FIOS '98)* , volume 46, Trento, Italy. lOS press.

**[GUARINO et WELTY, 00]** Guarino, N. et Welty, C. (2002). Evaluating ontological decisions with ontoclean.*Communications of the ACM,* 45(2) , 61- 65.

**[GENES et al., 92]** M.R. Genesereth, R.E. Fikes.: Knowledge Interchange Format. Version 3.0. Reference Manual. Technical Report Logic-92-1. Computer Science Department. Stanford University, California.1992.

**[GENNARI et al., 03]** Gennari, J., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., Noy, N. F., et Tu, S. W. (2003). The evolution of protégé : An environment for knowledge-based systems development. International Journal of HumanComputer Studies, 58 :1:89-123**.**

**[GOBEN, 99]** Gomez Pérez A. et Benjamins V.R. «Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods.» Proceedings of the IJCAI-99, workshop on Ontologies and Problem-Solving Methods (KRR5), Chandrasekaran B., Benjamins V.R., Gómez-Pérez A., Guarino N. et Uschold M., Stockholm (Suède) 1999

# *Bibliographie*


**[GOMEZ-PEREZ, 99]** Gomez Pérez A. « Développements récents en matière de conception, de maintenance et d'utilisation d'ontologies.» 3èmes rencontres Terminologie et intelligence artificielle TIA, 1999.

**[GOMEZ-PEREZ et al., 03]** A.Gomez-Pérez, M. Fernández-López, O. Corcho. O.(2003). Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web, First edition. New York: Springer.

**[GOMEZ-PEREZ et al., 04]** A. Gómez-Pérez, M. Fernández-López, O. Corcho.: Ontological Engineering, ISBN 1-85233-551-3 Springer-Verlag London Limited 2004.

**[GRAU et al., 05]** Grau**,** B.C., Parsia, B., Sirin, E. et Kalyanpur, A. (2005).Automatic partitioning of owl ontologies using e-connections. Dans Proceedings of the 2005 International Workshop on Description Logics (DL-2005).

**[GRAU et al., 07]** Grau, B. C., Horrocks, I., Kazakov, Y. et Sattler, U. (2007). Just the right amount: Extracting modules from ontologies. Dans Proceedings of the 16th International Conference on World Wide Web, 717-726. , Banff, Alberta, Canada. ACM.

**[GRAU et al., 08]** Grau, B. C., Horrocks, I. , Kazakov, Y. et Sattler, U. (2008). Modular reuse

**[GRAU et al., 09]** Grau, B. C., Horrocks, I., Kazakov, Y. et Sattler, U. (2009). Extracting modules from ontologies : A logic-based approach. In M odular Ontologies 159- 186. Springer.

**[GRUBER, 93]** Gruber T. «A translation approach to portable ontology specifications.» Knowledge. Acquisition 5(2): pp 199–220, 1993.

**[GRUBER, 95]** Gruber T. « A translation approach to portable ontology specifications.» Knowledge. Acquisition 5(2): pp 199–220, 1993

**[GRUNI et FOX, 95]** Grüninger, M., & Fox, M. (1995). Methodology for the design and evaluation of ontologies.In D. Skuce (Ed.), IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing,(pp. 6.1–6.10).

**[GUARI, 95]** Guarino N., et Giaretta P. «Ontologies and Knowledge Bases: Towards a Terminological Clarification.» In *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing,* Mars N. J. I., Amsterdam: IOS Press, 1995.

**[GUARI, 97]** Nicola GUARINO. «Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology.» 1997.
SCIE 1997,M.T.Pazienza (Eds.),Springer Verlag, pp 139-170.
www.ladseb.pd.cnr.it/infor/ontology/Papers/OntologyPapers.html.

**[GUARI, 98]** Guarino N. « Formal Ontology and Information Systems.» Formal Ontology in Information Systems. IOS Press, 1998.
of ontologies : Thcory and practice. *Journal of Artificial Intelligence Research,* 31(1) , 273- 318.

**[GUHA et al., 99]** S. Guha, R.Rastogi, and Shim, K. ROCK, —A Robust Clustering Algorithm for Categorical Attributes‖. In Proceedings of the 15th International Conference on Data Engineering (Sydney, Australia. March 23-26 1999.

# *Bibliographie*


**[GUS et STRUBE, 04]** I.Gurevych et M. Strube. Semantic similarity applied to spoken dialogue summarization. In Proceedings of the 20th International Conference on Computational Linguistics, Geneva, Switzerland, 23 - 27, pp. 764-770, 2004.

**[HEFLIN et al., 03]** Heflin, J., Hendler, J. A. et Luke, S. (2003). Shoe : A blueprint for the semantic web. In Fensel, D., Hendler, J. A., Lieberman, H. et Wahlster, W., éditeurs : Spinning the Semantic Web : Bringing the World Wide Web to Its Full Potential [outcome of a Dagstuhl seminar], pages 2963. MIT Press.

**[HEINER, 00]** Heiner Stuckenschmidt, Holger Wache, Thomas V¨ogele, and Ubbo Visser. Enabling technologies for interoperability. In Ubbo Visser and Hardy Pundt, editors, *Workshop on the 14th International Symposium of Computer Science for Environmental Protection*, pages 35–46, Bonn, Germany, 2000. TZI, University of Bremen.

**[HERNA, 05]** Nathalie HERNANDEZ. «Ontologies de domaine pour la modélisation du contexte en recherche d'information.» Thèse de doctorat, université Paul Sabatier de Toulouse, 2005.

**[HIB et BUDAN., 04]** G. Hirst et A. Budanitsky Correcting real-word spelling errors by restoring lexical cohesion. Natural Language Engineering 2004.

**[HIRS et ONGE, 98]** G.Hirst et D.St Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In Christiane Fellbaum (editor), WordNet: An electronic lexical database, Cambridge, MA:The MIT Press .1998.

**[HORROCKS, 98]** Horrocks, I. (1998). The FaCT system. Lecture Notes in Computer Science, 1397.

**[HORROCKS et al., 00]** I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, Goble, F. van Harmelen,M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, Sept. 2000.

http://www.ontoknowledge.org/oil/.

**[HORROCKS et SAT, 03]** Horrocks, I. et Sattler, U. (2003). Decidability of shiq with complex role inclusion axioms. IJCAI'2003, pages 343-348.

**[HU et al., 06]** W.Hu, Y. Zhao, Y.Qu, ―Partition-based block matching of large class hierarchies‖. In Asian Semantic Web Conference, pp 72-83. 2006.

**[HUANG et LAI, 06]** X.Huang, W.LAI. Clustering graphs for visualization via node similarities

Journal of Visual Languages and Computing 17 (2006) 225–253

**[JEDIDI, 09]** R.Jedidi. Approche d'évolution d'ontologie guidée par des patrons de egstion de changement. Thèse de doctorat.Université Paris sud XI Orsay. 2009.

**[JEF et al., 05]** H. Jeffrey, L. William et D. John. A Semantic Similarity Measure for Semantic Web Services. In proceedings of WSS05. 2005.

**[JIANG et al., 06]** Z. Jiang, S. Qingguo T. Tang, Li Y ongiang, ―An aggregation cache replacement algorithm based on ontology clustering‖. Journal of natural sciences. Vol. 11 NO.5 1141-1146.2006

**[JIANG et CON, 97]** Jiang, J. et Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. Dans *Proceedings of 1 Oth International Conference on*

*Research in Computational Linguistics (ROCLING X),*19- 33. Computing Research Repository CoRR. Récupéré de http: *1* /www. ci te base. org/ abstract? id=oai :arXiv .org :cmp-lg/9709008.

**[JIME-RUIZ et al., 08]** Jiménez-Ruiz, E., Grau, B. C., Sattler, U. , Schneider, T. et Berlanga, R (2008). Safe and economie re-use of ontologies : A logic-based methodology and tool support. In *The Semantic Web : Research and Applications,* volume 5021 de *LNCS* 185-199. Springer.

**[KAROL et MANGAT, 13]** S. Karol, V. Mangat. ―Evaluation of a Text Document Clustering Approach based on Particle Swarm Optimization‖. IJCSNS International Journal of Computer Science and Network Security, Vol.13 No.7, July 2013.

**[KASS, 02]** Kassel G. «OntoSpec: une méthode de spécification semi-informelle d'ontologies.» In Actes des journées francophones d'Ingénierie des Connaissances, 2002.

**[KAVEH, 04]** Kaveh Bazargan. 2004. "Le rôle des ontologies de domaine dans la conception des interfaces de navigation pour des collections en ligne de musées : évaluation et proposition". Mémoire de DEA en Management et Technologies des Systèmes d'Information (MATIS), Genève, Université de Genève.

**[KIFER et al., 95]**. M. Kifer, G. Lausen, J. Wu.: Logical Foundations of Object-Oriented and Frame- Based Languages. Journal of the ACM, 42(4): pp 741–843, 1995.

**[KLEIN, 01]**.M. Klein.: Combining and relating ontologies: an analysis of problems and solutions. In Proc. IJCAI Workshop on Ontologies and Information Sharing, Seattle (WA US), 2001.

**[KOLLI, 08]** R. Kolli. ―Scalable Matching Of Ontology Graphs Using Partitioning‖. M.S.Thesis, University of Georgia. Kunjir MP, 2008.

**[KUNJIR et Puraji, 09]** M.P. Kunjir, MD Puraji,. Project Report on Effective and Efficient computation of Cluster Similarity. Indian Institute of Science, Bangalore 2009.

**[KUTZ et al., 04]** Kutz, 0 ., Lutz, C. , Wolter, F. et Zakharyaschev, M. (2004) . E-connections of abstract description syst ems. *Artificial Intelligence, 156(1)* ,1- 73..

**[LASSI, 01]** O. Lassila et D. McGuiness «The role of frame-based representation on the semantic Web.» Rapport technique KSL-01-02, Knowledge Systems Laboratory, Stanford University, 2001.

**[LAVOI, 07]** Benoit LAVOIE. «Notion d'ontologie et construction d'ontologie à partir de corpus de textes.» Programme de Doctorat en Informatique Cognitive, Université du Québec à Montréal, 2007.

**[LEC, 98]** C. Leacock et M. Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. In WordNet: An Electronic Lexical Database, C. Fellbaum, MIT Press, 1998.

**[LEE et al., 93]** J.H.Lee, M.H.Kim et Y.J.Lee. Information Retrieval Based on Conceptual Distance in IS-A Hierarchy. Journal of Documentation 49, pp. 188-207, 1993.

**[LENAT et GUHA, 90]** Lenat, D. B. et Guha, R. V. (1990). Building large knowledge based systems. Reading, Massachusetts : Addison Wesley.

**[LIN, 98]** D. Lin. An Information-Theoretic Definition of similarity. In Proceedings of the

Fifteenth International Conference on Machine Learning (ICML'98). Morgan- Kaufmann: Madison, WI, 1998.

**[LOR et al., 03]** P.W. Lord, R.D. Stevens, A. Brass et C.A.Goble. Semantic Similarity Measures as Tools for Exploring the Gene Ontology. Pacific Symposium on Biocomputing 8, pp.601-612, 2003.

**[LUTZ et al., 07]** Lutz**,** C., Walther, D. et Walter, F. (2007). Conservative extensions in expressive description logics. Dans *Proceedings of the 20th International Joint Conference on Artificial Intelligence,* volume 7, 453- 458. , Hyderabad, India.

**[MCBRIDE, 01]** B. McBride.Jena : Implementing the rdf model and syntax specification. In *Semantic Web Workshop WWW2001*, 2001.

**[MACGREGOR, 91]**.R. MacGregor.: Inside the LOOM clasifier. SIGART bulletin, 2(3): pp70–76, 1991.

**[MIL et al, 93]** G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, et K. Miller. Introduction to WordNet: An On-line Lexical Database. Cognitive Science Laboratory,Princeton University, Princeton, Technical Report 1993.

**[MINSKY, 75]**.M. Minsky.: A framework for representing knowledge, in Winston, P. H. The Psychology of Computer vision, New York, McGraw-Hill, pp 211-277,1975.

**[MIZO, 96]** Mizoguchi R. et Ikeda M. «Towards Ontological Engineering (AI-TR-96-1).», Osaka: ISIR, Osaka, 1996.

**[MIZO, 98]** Mizoguchi R. «A Step Towards Ontological Engineering.» *12th National conference AI of JSAI 1998. En ligne*<http://www.ei.sanken.osakau.ac.jp/english/steponteng.html>.

**[MIZO, 00]**Mizoguchi R. et Bourdeau J. «Using Ontological Engineering to Overcome Common AI-ED Problems.» *International Journal of Artificial Intelligence in Education, IJAIED.* Vol. 11, no 2, 2000. En ligne <http://cbl.leeds.ac.uk/ijaied/>.

**[MOTTA, 99]** E. Motta.: Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design. IOS Press, Amsterdam. The Netherlands. 1999.

**[NAPOLI, 97]**.A. Napoli.: Une introduction aux logiques de descriptions, Rapport de recherche N°3314, INRIA, 1997.

**[NATAL, 02]** NATALYA F.NOY et DEBORAH L. MCGUINNESS. « Développement d'une ontologie 101 : Guide pour la création de votre première ontologie. » Université de Stanford, Stanford, CA, 94305, USA, 2002.

**[NEECH, 91]** Neeches R., Fikes R. E., Finin T., Gruber T. R., Senator T. et Swartout W. R. «Enabling technology for knowledge sharing.» *AI Magazine*. vol. 12, no 3, 1991.

Noy, N. et Musen, M. (2004). Specifying ontology views by traversal. *The Semantic Web-ISWC 2004 ,* 713-725.

**[NOY, 00]** N. Noy, R.W. Fergerson, M.A. Musen.: The knowledge model of Protégé2000: combining interoperability and flexibility, in Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW'00),2000.

**[NOY et MUSEN, 03]** Noy, N. F.et Musen, M. A. (2003) .The prompt suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies, 59(6),* 983-1024.

# *Bibliographie*


**[NOY et MUSEN, 04]** Noy, N. et Musen, M. (2004). Specifying ontology views by traversal. *The Semantic Web-ISWC 2004 ,* 713-725.

**[PALMIZ et al., 09]** Palmisano, I., Tamma, V., Payne, T. et Doran, P. (2009). Task oriented evaluation of module extraction techniques. In *The Semantic Web-ISWC 2009* 130- 145. Springer.

**[PAR et SPAC, 09]** Parent, C. et Spaccapietra, S. (2009). An overview of modularity. Dans *Modular Ontologies,* volume 5445 de *Lecture Notes in Computer Science,* 24-55.Springer-Verlag.

**[PATRICK, 14]** PATRICK,T.P. Modularisation des ontologies. Mémoire préparé comme exigence partielle de la maitrise en informatique. Université du Quebec à montréal. 2014

**[PINTO et al., 99]** Pinto, H. S., G´omez-P´erez, A.,and Martins, J. P. (1999). Some issues on ontology integration.In Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99, Stockholm, Sweden.

**[PINTO et al., 00]** H. Sofia Pinto, J.P. Martins « Reusing Ontologies », Portugal, 2000

**[POVEDA et al.,10]** M. Poveda-Villalon, M. Carmen Swirez-Figueroa, R. Garcia-Castro, and A. GomezPérez.201 O. "A Context Ontology for Mobile Environments". In *Proceedings of the Second Workshop on Context Information and Ontologies (CIA02010),* Tilbur University, The Netherlands: CEUR-WS.org, vol. 626.

**[QUILLIAN, 68]**. M.R. Quillian.: Semantic Memory. In Semantic Information Processing. MIT Press, pp 227–270, 1968.

**[RADA et al., 89]** R. Rada, H. Mili, E. Bichnell et M. Blettner, Development and application of a metric on semantic nets. IEEE Transaction on Systems, Man, and Cybernetics: pp 17-30. 1989.

**[RES, 95]** P. Resnik. Using information content to evaluate semantic similarity in taxonomy. In Proceedings of 14th International Joint Conference on Artificial Intelligence, Montreal, 1995.

**[RES, 99]** P. Resnik. Semantic similarity in a taxonomy: An information based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research, 11:95-130, 1999.

**[RTK et al., 95]** P.J. Rousseeuw, E. Trauwaert, and L. Kaufman. Fuzzy clustering with high contrast. *J. Comput. Appl. Math.*, 64 :81–90, November 1995.

**[SAL et MCG, 83]** G.Salton et M. J.McGill, Introduction to modern information retrieval. McGraw-Hill. New York, 1983.

**[SANG et LAVAN, ?? ]** C. Sang, G. Suh Lavanya, ―Role of Clustering of Ontology Relations for Preventive Health Care through Nutrition‖. Technical report, unpublished.

**[SARULAD et al., 12]** K. Saruladha, G. Aghila, B. Sathiya, ―A Partitioning Algorithm for Large Scale Ontologies‖. International Conference on Recent Trends In Information Technology (ICRTIT), 2012

**[SCHE, 97]** SCHREIBER, G. V. HEIJST, A. B. WIELINGA. « Using explicit ontologies in KBS development ». In « International Journal of Human-
Computer Studies», volume 46, 1997, pages 183-298.

**[SEID et RECTOR, 05]** Seidenberg, J. et Rector, A. (2005) . Techniques for segmenting large

description logic ontologies. Dans *Workshop on Ontology Management : Searching, Selection, Ranking, and Segmentation. 3rd International Conference on Knowledge Capture,* 49-56.

**[SEID et RECTOR, 06]** Seidenberg, J. et Rector, A. (2006). Web ontology segmentation : analysis, classification and use. Dans *Proceedings of the 15th International Conference on World Wide Web,* 13- 22., NY, USA. ACM.

**[SETTI AHMED et al., 11]** Soraya Setti Ahmed, Mimoun Malki, Sidi Mohamed Benslimane. Extracting Views From Domain Ontology: An Existential Dependency Driven Approach. International Conference on Knowledge Engineering and development, KEOD 2011. 26-28 Octobre, 2011, Paris – France.

**[SETTI AHMED et BENSL, 12]** Soraya Setti Ahmed, Sidi Mohamed Benslimane: Reverse Engineering Process for Extracting Views from Domain Ontology. International Conference of Web and Information Technologies (ICWIT'2012), 29-30 April, 2012, Sidi Bel Abbes, Algeria.

**[SETTI AHMED et al., 15]** Soraya Setti Ahmed, Mimoun Malki, Sidi Mohamed Benslimane. Ontology Partitioning: Clustering Based Approach. International Journal of Information Technology and Computer Science, 2015, 06, 1-11 Published Online May 2015 in MECS (http://www.mecs-press.org/) DOI: 10.5815/ijitcs.2015.06.01

**[SLIMANI et al., 07]** T.Slimani,B.Ben Yaghlane et Y.Mellouli : A New Similarity Measure based on edge counting, word academy of science , Engineering and technology. 2008

**[SOWA, 84]** J. Sowa.: Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley,1984.

**[STAAB, 00]** S. Staab, A. Maedche. «Axioms are objects too: Ontology engineering beyond the modeling of concepts and relations.» Research report 399, Institute AIFB, Karlsruhe, 2000.

**[STAAB, 01]** Steffen Staab, Rudi Studer, Hans-Peter Schnurr, York Sure, « Knowledge Processes and Ontologies. », IEEE Intelligent Systems, v.16 n.1, January 2001.

**[STUCK et KLEIN, 04]** Stuckenschmidt, H. et Klein, M. (2004). Structure-based partitioning of large concept hierarchies. In *The Semantic Web-ISWC 2004* 289-303. Springer.

**[STUCK et SCHL, 09]** Stuckenschmidt, H. et Schlicht, A. (2009). Structure-based partitioning of large ontologies. Dans *Modular Ontologies,* volume 5445 de *Lecture Notes in Computer Science,* 187- 210. Springer-Verlag.

**[STUD, 98]** Studer R., Benjamins V. R. et Fensel D. «Knowledge engineering: Principles and Methods.» Data Knowledge Engineering, vol. 25, no 1-2,1998.

**[SURE et al., 02]** SURE Y., ERDMANN M., ANGELE J., STAAB S., STUDER R., AND WENKE D. (2002). OntoEdit : Collaborative Ontology Engineering for the Semantic Web. In I. Horrocks and J. Hendler, editors, First International Semantic Web Conférence (ISWC'02), volume (2342) of Lecture Notes in Computer Science, pages 221-235, Chia, Sardaigne, Italie, 9-12 Juin 2002. Springer Verlag.

**[SWART, 97]** Swartout B., Patil R., Knight K. et Russ T. «Towards Distributed Use of Large Scale Ontologies.» Spring Symposium Series on Ontological Engineering, Stanford University, CA,1997.

**[USCHO et KING, 95]** Uschold M., King M. « Towards a Methodology for Building Ontologies. », in Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing

(IJCAI95), Montreal, 1995.

**[USCHO et GRU, 96]** Uschold M. et Grüninger M. «Ontologies: Principles, Methods And Applications». *Journal of Knowledge Engineering Review*. vol. 11, no 2, 1996.

**[USCHO et al., 99]** M. Uschold, R. Jasper.: A Framework for Understanding and Classifying Ontology Applications. In: Benjamins VR (ed) IJCAI'99 Workshop on Ontology and Problem Solving Methods: Lessons Learned and Future Trends. Stockholm, Sweden. CEUR Workshop Proceedings 18:11.1–11.12. Amsterdam. The Netherlands.1999. (http://CEUR-WS.org/Vol-18/).

**[VALERY et al., 03]** VALERY PSYCHÉ, Olavo Mendes et Jacqueline Bourdeau.(2003).Apport de l'ingénierie ontologique aux environnements de formation à distance. Article de recherche, Revue STICEF, Volume 10, ISSN : 1764-7223

**[VALERY, 04]** VALERY PSYCHÉ. Proposition d'une méthode d'ingénierie ontologique pour les EIAH : application aux systèmes auteurs. Aspects théoriques, modèles et expérimentations. Une proposition de recherche doctorale Programme de doctorat en informatique cognitive Département d'informatique. Université du Québec à Montréal. Canada.

**[VALERY, 07]** VALERY PSYCHÉ. « Rôle des ontologies en ingénierie des EIAH : Cas d'un système d'assistance au design pédagogique.» Thèse de doctorat, Université du Québec, Montréal, 2007.

**[VGELE et al., 01]** T. Vgele, H. Wache, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hbner:"Ontology-based integration of information A survey of existing approaches". In Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing. 2001.

**[VISSER et al., 98]** P.Visser, D.Jones, T. Bench-Capon, M. Shave.: Assessing heterogeneity by classifying ontology mismatches. In Proc. 1st International Conference on Formal Ontology in Information Systems (FOIS ).Trento (IT), pp 148–162,1998.

**[WACHE, 99**] H. Wache, Th. Scholz, H. Stieghahn, and B. K¨onig-Ries. An integration method for the specification of rule–oriented mediators. In Yahiko Kambayashi and Hiroki Takakura, editors, *Proceedings of the International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 109–112, Kyoto, Japan, November, 28-30 1999.

**[WACHE et al., 01]** H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hübner.: Ontology-Based Integration of Information – A Survey of Existing Approaches. In Stuckenschmidt, H.,editor, IJCAI-2001 Workshop on Ontologies and Information Sharing, pp 108-117, 2001.

**[WAYNE et al., 74]** Wayne P. Stevens, Glenford J. Myers, and Larry L. Constantine. Structured system. IBM Systems Journal, 13(2):115–139, 1974.

**[WU et PALMER., 94]** Z. Wu et M. Palmer. Verb semantics and lexical selection. In Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics, pp 133-138. 1994.

**[YIGAL et al., 96]**Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the SIMS information mediator. In Advanced Planning Technology. AAAI Press, California, USA,1996.

# *Bibliographie*

**[ZIDI, 06]** K.ZIDI.Système interactif d'aide au déplacement multimodal (SIADM), Ecole centrale de Lille, Université de lille 2006.

# Annexes

# *ANNEXE A : Mesures de similarités*

## A.1 Introduction

La notion de similarité a été très tôt perçue comme un concept clé en intelligence artificielle, comme le souligne Rissland [RISSLAND, 06]. Elle est au cœur du paradigme qui énonce qu'un transfert de connaissances d'un cas connu vers un cas inconnu est possible dans la mesure où ils sont suffisamment similaires. Rissland appelle ce paradigme l'< heuristique du CBR > (*Case-Based Reasoning* ou raisonnement à partir de cas).

Le domaine de l'identification de la similarité a été considéré comme un sujet de recherche fortement recommandé dans les domaines du Web sémantique, de l'intelligence artificielle et de la littérature linguistique. Dans le domaine du Web sémantique où les ontologies interviennent pour la modélisation des connaissances.

Dans cette annexe nous présentons les différentes mesures de similarités existantes et dont nous avons utilisé quelques une.

A l'heure actuelle, la notion de similarité est mise en avant dans plusieurs domaines d'activités liés à l'ingénierie des ontologies tels que l'apprentissage, l'alignement ou encore le peuplement d'ontologies. La notion de similarité intervient ainsi dans de nombreux domaines de l'intelligence artificielle comme l'apprentissage automatique, la recherche d'information, le raisonnement. Elle est aussi étudiée dans de nombreux domaines liés à l'intelligence artificielle comme l'analyse de données, les sciences cognitives ou la psychométrie.

Ces dernières années, de nombreuses mesures dédiées à la définition de la (dis-)similarité entre concepts ont été proposées. Ces mesures peuvent être classées suivant deux approches : (i) les mesures de type extensionnel telle que Resnik, Lin, Jiang et Conrath ou d'Amato et (ii) les mesures de type intensionnel telle que Rada, Leacock et Chodorow ou Wu et Palmer. La plupart de ces mesures se focalisent sur un seul

aspect de la conceptualisation sous-jacente à une ontologie de domaine, soit l'intension – au travers de la structure de la hiérarchie de subsomptions, soit l'extension – au travers des instances de concepts ou des occurrences de termes dénotant les concepts au sein d'un corpus. De plus, ces mesures sont majoritairement sensibles à la structure de la hiérarchie de subsumptions (par l'utilisation du subsumant commun le plus spécifique) et, par conséquent, dépendantes des choix de modélisation.

## A.2 Mesures de similarité : revue de l'existant

Une mesure de similarité est un concept de base dans de nombreux domaines qu'une distinction s'impose selon les types de données auxquelles elles s'appliquent.
Les deux grands types de données simples sont les données symboliques et les données numériques.

**Généralités**

Les mesures de similarité étant très largement utilisées dans des domaines très divers, leur terminologie varie considérablement (ressemblance, proximité, …) et l'auteur d'une mesure particulière n'est pas forcément unique. Malgré la confusion qui règne dans la littérature, certaines conditions sur les mesures de similarité sont plus fréquemment mentionnées que d'autres.

La similarité joue un rôle central dans le processus de mapping ; Elle se rapporte à la comparaison des éléments d'ontologies. Elle renvoie une valeur numérique indiquant si les deux éléments ont un degré élevé ou bas de similitude. Cette similarité peut être calculée selon plusieurs façons (terminologique 'syntaxique linguistique', structurelle, etc.)

D'autres propriétés peuvent être requises comme la normalisation qui impose que les valeurs appartiennent à l'intervalle [0; 1]. Des mesures non normalisées peuvent subir une transformation de normalisation pour obtenir leur version normalisée. Dans la suite, nous considérons le cadre des mesures normalisées.
Dans la littérature, plusieurs travaux sur la mesure de similarité sémantique entre les objets d'une ontologie ont été développés dans différents contextes.

On peut distinguer trois grandes familles d'approches pour l'identification de la similarité sémantique. Les approches basées sur les nœuds [RES, 95] [LIN, 98] [JIAC, 97] utilisant typiquement des mesures du contenu informationnel pour déterminer la similarité conceptuelle. En plus, la similarité entre les concepts est déterminée par le degré de partage de l'information. L'autre famille d'approches repose uniquement sur la hiérarchie ou sur les distances
des arcs [RADA, 89] [WU et PALMER, 94] [LEE et al., 93] [EHR, 04]. Le principe de calcul de similarité avec cette approche est basé sur l'idée suivante : plus le chemin entre deux nœuds est court plus ils sont plus semblables. L'autre notion qui caractérise cette deuxième approche est que les arcs d'une taxonomie représentent des distances uniformes, par conséquent, cette approche présente l'inconvénient que tous les liens sémantiques possèdent le même poids ce qui impose des difficultés au niveau de la définition et du contrôle des distances des liens. Finalement l'approche hybride [LEC, 98] [RES, 99] qui combine entre les deux premières approches. Avec cette approche, il y a plusieurs manières de déterminer la similarité conceptuelle de deux mots dans un réseau sémantique hiérarchique.
Notons que les travaux recensés dans la littérature placés sous la bannière de la deuxième approche [WU et PALMER, 94] présentent l'inconvénient suivant : Dans certaines situations on peut obtenir une

valeur de similarité de deux éléments d'une ontologie contenus dans le voisinage qui dépasse la valeur de similarité de deux concepts contenus dans la même hiérarchie. Cette situation est inadéquate dans le cadre de la recherche de l'information. Dans le présent travail nous avons utilisé sept mesures de similarité à savoir : (Tbk, Jaccard, Cosine, Dice, Euclidienne, Wu and palmer, Dennai)

## A.3 Classification des approches de mesure de similarité

Dans cette section nous présentons les approches principales de la mesure de similarité.

### 1) Approches basées sur les arcs

La mesure de similarité la plus intuitive des objets dans une ontologie est leurs distances [RADA, 89] [LEE, 93] [WU et PALMER, 94]. Evidemment, un objet X est plus similaire à un objet Y qu'un objet Z. Cette similarité est évaluée par la distance qui sépare les objets dans l'ontologie.

Ces mesures se servent de la structure hiérarchique de l'ontologie pour déterminer la similarité sémantique entre les concepts. Le calcul des distances dans l'ontologie est basé sur un graphe de spécialisation des objets. Dans chaque graphe, la distance de l'ontologie doit être caractérisée par le plus court chemin qui fait intervenir un ancêtre commun ou le plus petit généralisant, connectant potentiellement deux objets à travers des descendants communs. Parmi les travaux classifiés sous cette bannière on peut citer :

**Mesure de Wu & Palmer**

La mesure de similarité de [WU et PALMER, 94] est basée sur le principe suivant :

Etant donnée une ontologie W formée par un ensemble de nœuds et un nœud racine R (Figure A.1). Soit $C_1$ et $C_2$ deux éléments de

l'ontologie dont nous allons calculer la similarité. Le principe de calcul de similarité est basé sur les distances (N1 et N2) qui séparent les nœuds $C_1$ et $C_2$ du nœud racine et la distance qui sépare le concept subsumant (CS) de $C_1$ et de $C_2$ du nœud R.

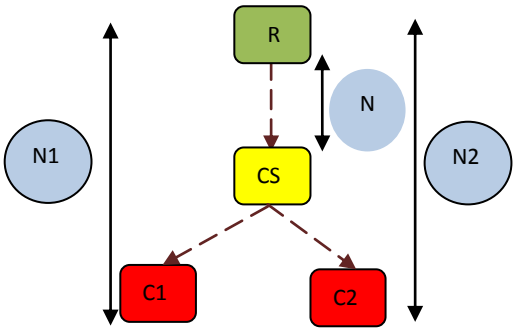



**Figure A.1 :** Exemple d'un extrait d'ontologie formé d'un nœud commun CS et un nœud racine R

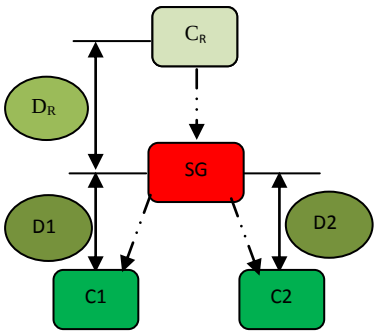



**Figure A.2** Exemple de calcul de distance dans une ontologie.

La mesure de Wu et Palmer est définie par la formule suivante :

$$\mathbf{SimWP(C1, C2)} = \frac{\mathbf{2 * DR}}{\mathbf{D1 + D2 + 2 * DR}}$$

[LIN, 98] a effectué une comparaison entre les méthodes des mesures de similarité. Il en ressort que la mesure de Wu et Palmer [Wup 94] a l'avantage d'être simple à calculer en plus des performances qu'elle présente, tout en restant aussi expressive que les autres, c'est pour cette raison qu'on a adopté cette mesure comme fondement de nos travaux.

La mesure de [WU et PALMER, 94] est intéressante mais présente une limite car elle vise essentiellement à détecter la similarité entre deux concepts par rapport à leur distance de leur plus petit généralisant, ce qui ne permet pas de capter les mêmes similarités que la similarité conceptuelle symbolique.

**Mesure de Rada et al.**

Cette mesure [RADA, 89] est adoptée dans un réseau sémantique et elle est fondée sur le fait qu'on peut calculer la similarité en se basant sur les liens hiérarchiques «is-a».

Pour calculer la similarité de deux concepts dans une ontologie, on doit calculer le nombre des arcs minimums qui les séparent. Cette mesure, basée sur le calcul de la distance entre les nœuds par le chemin le plus court, présente un moyen des plus évidents pour évaluer la similarité sémantique dans une ontologie hiérarchique.

**Mesure d'Ehrig et al.**

Un travail de mesure de similarité pour les ontologies a été introduit par [EHR, 04]. Ce travail introduit trois couches : les données, l'ontologie et le contexte. La similarité des entités est mesurée au niveau des données en considérant les valeurs de données de type simple ou complexe (entiers, caractères). Les relations sémantiques entre les entités sont mesurées au niveau de la couche de l'ontologie. Finalement la couche du contexte spécifie comment les entités de l'ontologie sont utilisées dans un certain contexte externe, plus spécifiquement, le contexte de l'application.

**2) Approches basées sur les nœuds**

Ces approches adoptent une nouvelle mesure en termes de la mesure entropique de la théorie de l'information [LIN, 98] [RES, 99].

La probabilité P(.) pour l'identification de l'utilisation d'une classe ou de ses descendants dans un corpus désigne l'information de la classe. On définit l'entropie d'une classe par la formule suivante :

$$E(C) = -\log(P(C))$$

où P est la probabilité de trouver une instance du concept c. La probabilité d'un concept c est calculée en divisant le nombre des instances de c par le nombre total des instances.

En associant des probabilités aux concepts d'une taxonomie, il est possible d'éviter le manque de fiabilité des distances des arcs. Cette caractéristique quantitative de l'information fournit une nouvelle façon de mesurer la similarité sémantique. Plus l'information est partagée par deux concepts, plus ils sont similaires.

Parmi les travaux, recensés dans la littérature, sous cette bannière on peut citer :

**Mesure de Resnik**

La notion du contenu informationnel (CI) a été initialement introduite par [RES, 95] qui a prouvé qu'un objet (mot) est défini par le nombre des classes spécifiées et que la similarité sémantique entre deux concepts est mesurée par la quantité de l'information qu'ils partagent. Pour évaluer la pertinence d'un objet il faut calculer le contenu informationnel. Le contenu informationnel est obtenu en calculant la fréquence de l'objet dans le corpus (Wordnet). La formule proposée par Resnik est définie par:

$$Sim(X,Y) = Max[E(CS(X,Y))] = Max[-Log(p(CS(X,Y))])$$

où CS(X, Y) représente le concept le plus spécifique (qui maximise la valeur de similarité) qui subsume (situé à un niveau hiérarchique plus élevé) les deux concepts X et Y dans l'ontologie.

Cette mesure est un peu sommaire car elle ne dépend que du concept le plus spécifique.

**Mesure de Lin**

Lin a défini une mesure de similarité légèrement différente de celle de Resnik :

$$SimL(X,Y\ ) = \frac{2 * \log\left(P(AC(X,Y))\right)}{\log\big(P(X)\big) + \log\left(P(Y)\right)}$$

Cette mesure utilise une approche hybride qui combine deux sources de connaissances différentes (Thesaurus, corpus).

En plus, elle représente la similarité comme degré probabiliste de chevauchement des concepts descendants de X et Y. Les travaux de [MIL et al., 93] ont évalué cette mesure à travers une expérience qui utilise des sujets humains pour évaluer la similarité entre 30 paires de noms, il en ressort que cette méthode offre une amélioration significative.

**Mesure de Hirst et Onge**

L'idée de cette mesure est que deux concepts lexicalisés sont sémantiquement étroits si leurs ensembles synonymes (synsets) de WordNet sont reliés par un chemin qui n'est pas trop long et qui "ne change pas la direction trop souvent". Avec cette mesure, toutes les relations contenues dans un réseau Wordnet sont prises en considération. Dans le travail de [HIRS et ONGE., 98], les auteurs ont classé la direction des liens en lien haut (super-classe), lien bas (sous-classe) et lien horizontal (antonyme). Le calcul de la similarité, avec cette méthode, s'effectue entre objets (mots) par le poids du plus court chemin allant d'un terme à un autre, en plus des classifications qui indiquent les changements de direction. La force du rapport est donnée par :

$$\mathbf{SimHt = T - PCC - K * nd}$$

où T et K sont des constantes, PCC est la distance du plus court chemin en nombre d'arc et nd le nombre de changements de direction.

**Approches hybrides**

Ces approches sont fondées sur un modèle qui combine entre les approches basées sur les arcs (distances) en plus du contenu informationnel qui est considéré comme facteur de décision.

**Mesure de Jiang et Conrath**

Pour remédier au problème présenté au niveau de la mesure de Resnik, [JIANG et CON, 97] a apporté une nouvelle formule qui consiste à combiner l'entropie (contenu informationnel) du concept spécifique à ceux des concepts dont on cherche la similarité (combine entre les techniques basées sur les arcs et les techniques basées sur les nœuds qui consistent à compter les arcs afin d'améliorer les résultats par des calculs basés sur les nœuds). La mesure adoptant cette méthode est basée sur la combinaison d'une source de connaissance riche (thesaurus) avec une source de connaissance pauvre (corpus). Notons que cette formule est définie par l'inverse de la distance sémantique.

$$Sim(X,Y) = \frac{1}{distance(X,Y)}$$

Sachant que la distance entre X et Y est calculée par la formule suivante:

$$distance(X,Y) = E(X) + E(Y) - (2.E(CS(X,Y)))$$

**Mesure de Leacock et Chodorow**

Une autre méthode présentée par [LEC, 98] qui combine entre la méthode de comptage des arcs et la méthode du contenu informationnel. La mesure proposée par Leacock et Chodorow [Lec 98] est basée sur la longueur du plus court chemin entre deux synsets de Wordnet. Les auteurs ont limité leur attention à des liens hiérarchiques «is-a » ainsi que la longueur de chemin par la profondeur globale P de la taxonomie. La formule est définie par :

$$SimLc(X,Y) = -\log\left(\frac{cd(X,Y)}{2 * M}\right)$$

Où M est la longueur du chemin le plus long qui sépare le concept racine, de l'ontologie, du concept le plus en bas. On dénote par cd (X, Y) la longueur du chemin le plus court qui sépare X de Y.

### 3) Approches basées sur l'espace vectoriel

Dans le domaine de la recherche de l'information, les modèles de l'espace vectoriel sont largement adoptés [BAR et al., 99][ SAL et MCG, 83]. Ces approches utilisent un vecteur caractéristique, dans un espace dimensionnel, pour représenter chaque objet et calculent la similarité en se basant sur la mesure de cosine ou la distance euclidienne. Le modèle de l'espace vectoriel est employé pour un arrangement des objets complexes en les représentant comme des vecteurs de k-dimensions. La définition de la similarité entre deux vecteurs d'objets est obtenue par leurs contenus internes. Parmi les approches citées dans la littérature on peut citer :

#### Similarité de Jaccard

La mesure de similarité de Jaccard est définie par le nombre des objets communs divisé par le nombre total des objets moins le nombre d'objets communs :

$$SimJ(X,Y) = \frac{x * y}{\|x\|_2^2 + \|y\|_2^2 - x * y}$$

Tels que x et y sont des vecteurs extraits à partir des concepts X et Y.

$\| X \| = \Omega x\ i = 1$ désigne la norme du vecteur X.

#### Similarité de Cosine

Cette mesure utilise la représentation vectorielle complète, c'est-à-dire la fréquence des objets (mots).

Deux objets (documents) sont similaires si leurs vecteurs sont confondus. Si deux objets ne sont pas similaires, leurs vecteurs forment un angle (X, Y) dont le cosinus représente la valeur de la

similarité. La formule est définie par le rapport du produit scalaire des vecteurs x et y et le produit de la norme de x et de y.

$$SimC(X,Y) = Cos(X,Y) = \frac{x * y}{\|x\|_2 * \|y\|_2}$$

La mesure de Cosine quantifie donc la similarité entre les deux vecteurs comme le cosinus de l'angle entre les deux vecteurs.

**Similarité euclidienne**

La similarité euclidienne est basée sur le ratio de la distance euclidienne augmenté de 1. La distance euclidienne est définie par la formule suivante :

$dE = \|x - y\|^2$

La mesure de similarité est donc définie par :

$$SimE(C_1, C_2) = \frac{1}{1 + dE}$$

**Similarité de Dice** La similarité de Dice [LIN, 98] est définie par le nombre des objets communs multipliés par 2 sur le nombre total d'objets. La mesure de Dice est donc définie par la formule suivante :

$$SimD(C_1, C_2) = \frac{2 * x * y}{\|x\|_2^2 + \|y\|_2^2}$$

## A.4 Quelques approches de la littérature

Dans la table1 nous montrons quelques branches de similarité qui accordent les critères de classification présentés dans la section précédente.

| | *[KOZ 93] [KOZ 97][WU et PALMER 94]* | *[DIE 02] [LEC 98]* | *[JIANG 97] [RES 95] [LIN 98]* | *[OKU 94]* |
|---|---|---|---|---|
| ***Type de similarité*** | Lexicale | Lexicale | Lexicale | lexicale |
| ***Source de données*** | Dictionnaire Monolingue | Wordnet | Wordnet et corpus | Thesaurus |
| ***Performance*** | Assez bien | Très bien | Très bien | Très bien |
| ***Langues d'utilisation*** | Plusieurs Langues | Quelques Langues | Quelques langues | Quelques langues |
| ***Type de données*** | Non Structuré | Structuré | structuré | Structuré |

**Tableau A.1-** Classification des caractéristiques de quelques travaux de similarités

**La mesure de similarité de TBK**

La mesure de [WU et PALMER, 94] est intéressante mais présente une limite car elle vise essentiellement à détecter la similarité entre deux concepts par rapport à leur distance de leur plus petit généralisant. Cette mesure présente l'avantage de la rapidité du temps d'exécution, mais l'inconvénient de la production d'une valeur de similarité de deux concepts voisins qui dépassent la valeur de deux concepts dans la même hiérarchie. Donc les auteurs ont opté pour une autre mesure celle de TBK [SLIMANI et al., 07].

$$\mathbf{Sim}TBK(C1,C2) = \frac{2 * Profendeur\ (C)}{Profendeur\ (C1) + \mathbf{Profondeur(C2)}} * \mathbf{Fp}(C1,C2)$$

Fp(C1,C2) est la fonction de pénalisation des concepts C1 et C2. L'utilité de cette fonction est de pénaliser la similarité de deux concepts éloignés qui ne sont pas situés dans une même hiérarchie.

$$\mathbf{Fp}(C1,C2) = \frac{1}{|Profendeur\ (C1) - \mathbf{Profondeur(C2)}| + 1}$$

Cette formule assure une valeur de similarité Simtbk(C1,CV) toujours inférieure ou égale à Simtbk(C1,CF), tels que CV est un concept voisin

(non inclus dans les noeuds fils de C1) et CF un nœud fils subsumé par C1.

**Théorème concernant la fonction de pénalité**

Etant donnés une ontologie Ω et deux concepts C1 et C2 contenus dans l'ontologie. Soit P1 la profondeur de C1 par rapport au concept racine de Ω et P2 la profondeur de C2 par rapport au concept racine de Ω .

1) Si $|P1 - P2| \longrightarrow K$ alors $fp\,(C1, C2) \longrightarrow 0$
2) Si $|P1 - P2| = 0$ alors $fp\,(C1, C2) \longrightarrow 1$

Constat 1 : La valeur de fp(C1, C2) est toujours dans l'intervalle ]0, 1],
Constat 2 : De plus en plus que la distance entre les concepts subsumant de C1 et C2 est éloignée de plus en plus que la fonction de pénalité diminue et de ce fait la valeur de similarité diminue.
Preuve : Soit Simbtbk = Simwp*EP. Les valeurs de Simwp et Simwp sont limitées par l'intervalle [0,1]. Il en ressort que fp est limité par [0,1]. Cette mesure aest utilisé également dans ce présent travail pour la comparer avec les six autres mesures de similarités.

**Mesure de Dennai**

[LIN, 98] a effectué une comparaison entre les méthodes des mesures de similarité. Il en ressort que la mesure de Wu et Palmer [WU et PALMER, 94] a l'avantage d'être simple à calculer en plus des performances qu'elle présente, tout en restant aussi expressive que les autres, cette mesure de [WU et PALMER, 94] est intéressante et a l'avantage d'être simple et plus rapide mais elle présente une limite car elle vise essentiellement à détecter la similarité entre deux concepts par rapport à leur distance de leur plus petit généralisant, ce qui ne permet pas de capter les mêmes similarités que la similarité conceptuelle symbolique. En outre son inconvénient réside dans la

valeur de la similarité calculée entre deux concepts voisins (near by concepts) qui dépasse celle de deux concepts qui se trouvent dans la même hiérarchie.

C'est pourquoi une nouvelle mesure de similarité a été introduite par Dennai et Benslimane [DENN et BENSL, 13] pour pallier à ce problème. Aussi cette mesure est utilisée au dans notre travail.

**Principe de la mesure**

Le travail de [DENN et BENSL, 13] est une mise à jour et amélioration de la mesure de Wu AND Palmer. Son expression est représentée par la formule suivante:

$$\text{Sim } DB(C1, C2) = \frac{2 * \text{D}}{D1 + D2 + 2 * D + FP_{SG(C1,C2)}}$$

$$FPD_{SG(C1,C2)} = \begin{cases} 0 & if\ C1\ is\ ancestor\ of\ C2\ or\ conversely \\ Dist & if\ C1\ and\ C2\ are\ closed\ by\ a\ CS \end{cases}$$

Ou $Dist = (D + D1) * (D + D2)$ et FPD_SG (Function Produces Depths by Smaller Generalizing). FPD_SG (C1, C2) est la fonction de pénalisation des concepts C1 et C2. L'utilité de cette fonction est de pénaliser la similarité de deux concepts reliés qui ne sont pas situés dans une même hiérarchie.

Dans le cas de deux concepts

FPD_SG donne la distance Dist qui est égale au produit des profondeurs de deux concepts (C1, C2) comparés à leur racine(R) en passant par le concept (CS). ou D est la distance entre CS et la racine R et Di représente la distance entre un concept Ci et son concept subsumeur CS. En utilisant cette fonction, la mesure de similarité entre deux concepts dans la même hiérarchie est supérieure avec la similarité de deux concepts reliés avec un CS.

FPD_SG (C1, C2) est la fonction de pénalisation des concepts C1 et C2. L'utilité de cette fonction est de pénaliser la similarité de deux concepts éloignés qui ne sont pas situés dans une même hiérarchie.

## A.5 Conclusion

Dans cette annexe nous avons motivé la notion de similarité. Elle est au cœur du paradigme qui énonce qu'un transfert de connaissances d'un cas connu vers un cas inconnu est possible dans la mesure où ils sont suffisamment similaires. Rissland appelle ce paradigme l'< heuristique du CBR > (*Case-Based Reasoning* ou raisonnement à partir de cas).

A l'heure actuelle, la notion de similarité est mise en avant dans plusieurs domaines d'activités liés à l'ingénierie des ontologies tels que l'apprentissage, l'alignement ou encore le peuplement d'ontologies. La notion de similarité intervient ainsi dans de nombreux domaines de l'intelligence artificielle comme l'apprentissage automatique, la recherche d'information, le raisonnement. Elle est aussi étudiée dans de nombreux domaines liés à l'intelligence artificielle comme l'analyse de données, les sciences cognitives etc…

# *ANNEXE B : Quelques interfaces de notre système développé SysOPCBA*

## B.1 Introduction

Dans cette partie nous présentons quelques interfaces de notre système développé SysOPCBA.

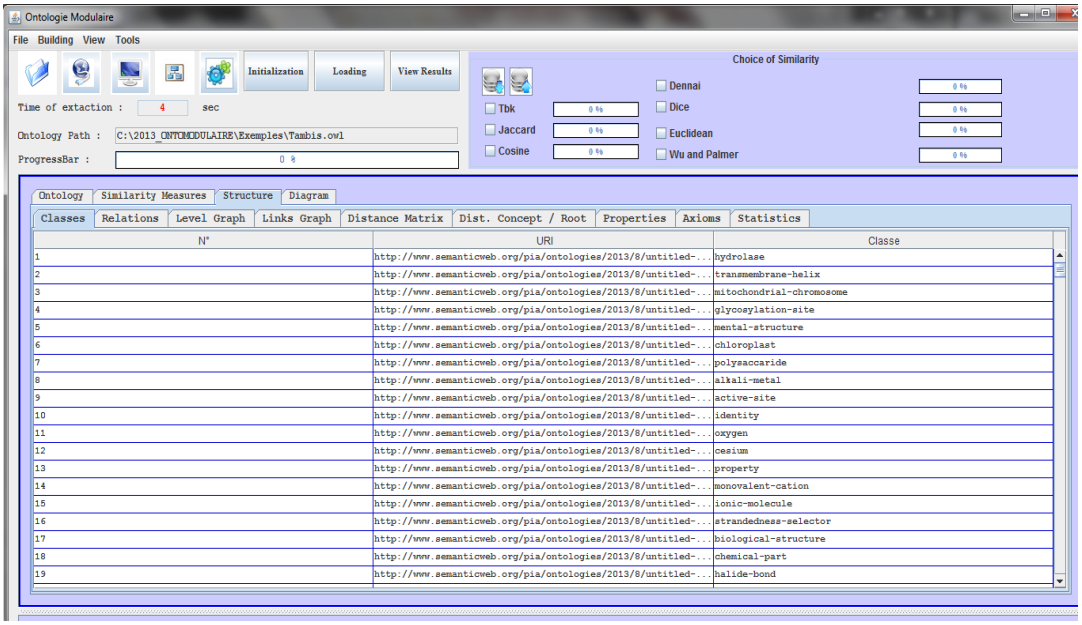



**Figure.B.1 :** Copie d'écran de l'interface d'analyse et d'extraction de notre ontologie (extraction des classes)

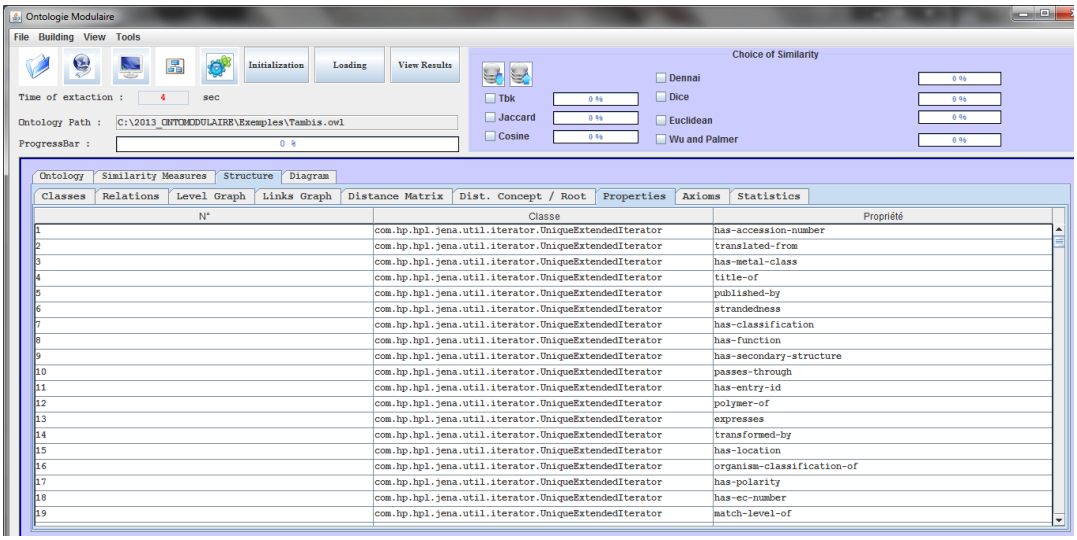



**Figure.B.2 :** Copie d'écran de l'interface d'analyse et d'extraction de notre ontologie (extraction des propriétés)

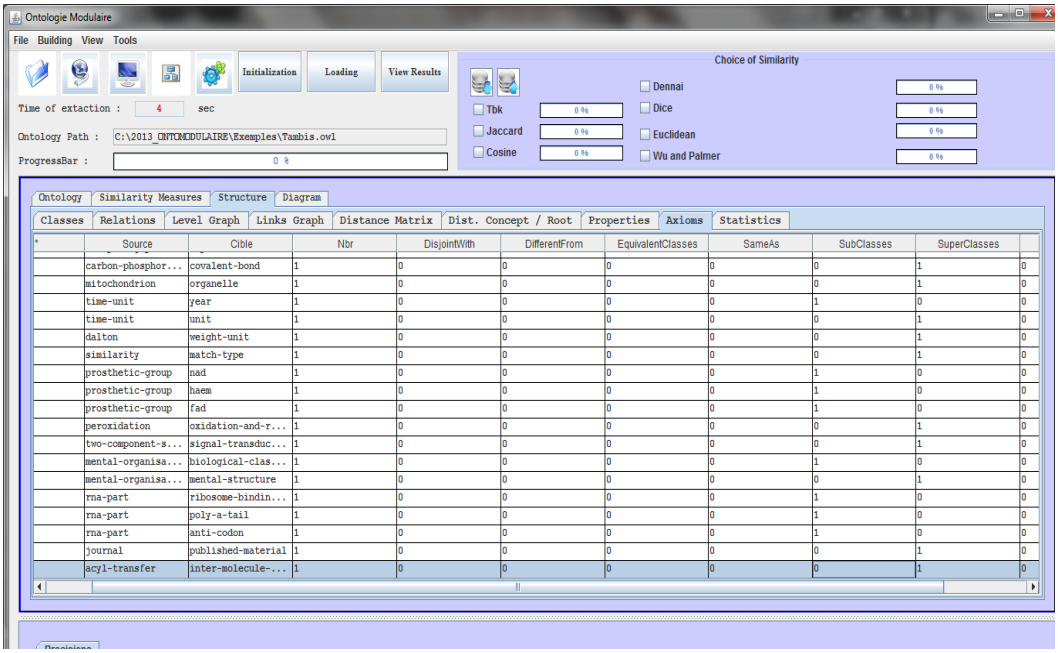



**Figure.B.3 :** Copie d'écran de l'interface d'analyse et d'extraction de notre ontologie (extraction des axiomes)

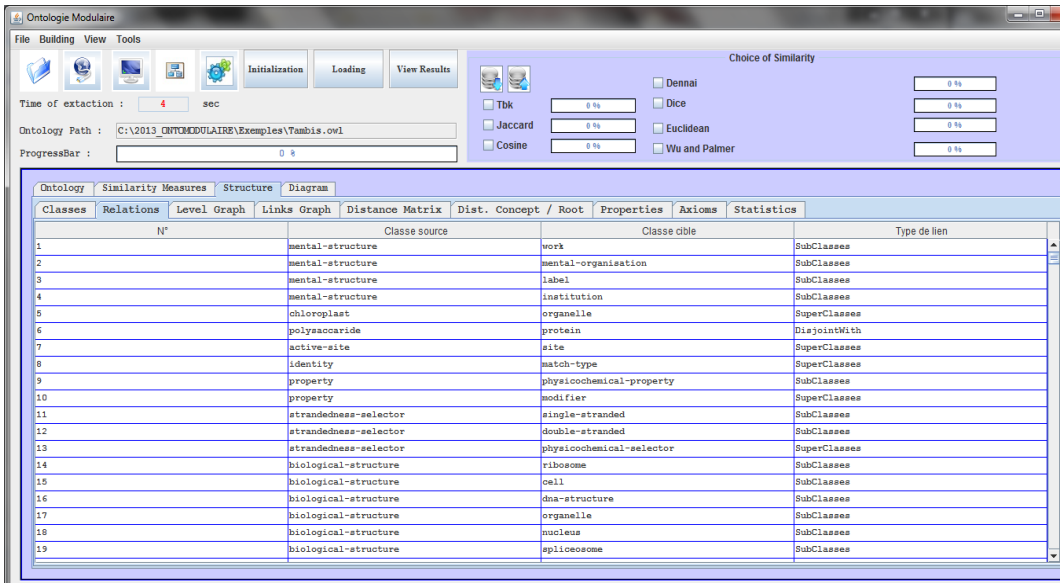



**Figure.B.4 :** Copie d'écran de l'interface d'analyse et d'extraction de notre ontologie (extraction des classes sources, classes cibles et le type de lien).

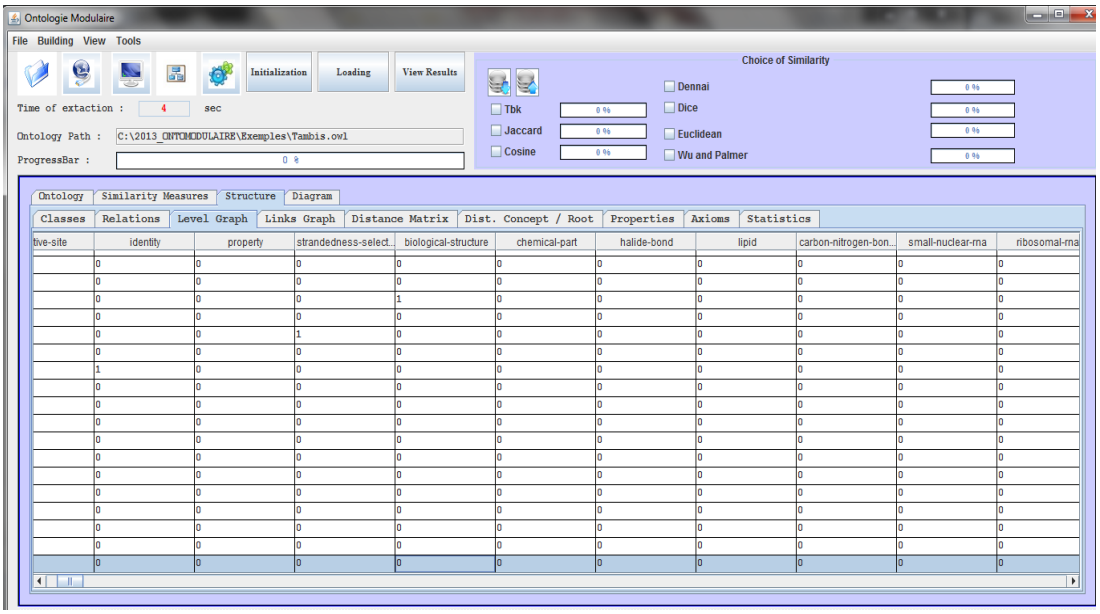



**Figure.B.5 :** Copie d'écran de l'interface de graphe de niveau (Level graph)

Comme nous avons vu dans l'architecture de notre système ; nous avons un module de création de graphe de liens mais avant de passer à cette étape, il faut charger notre ontologie en format XML les concepts sous forme d'entités graphiques (voir Figure.B.6).

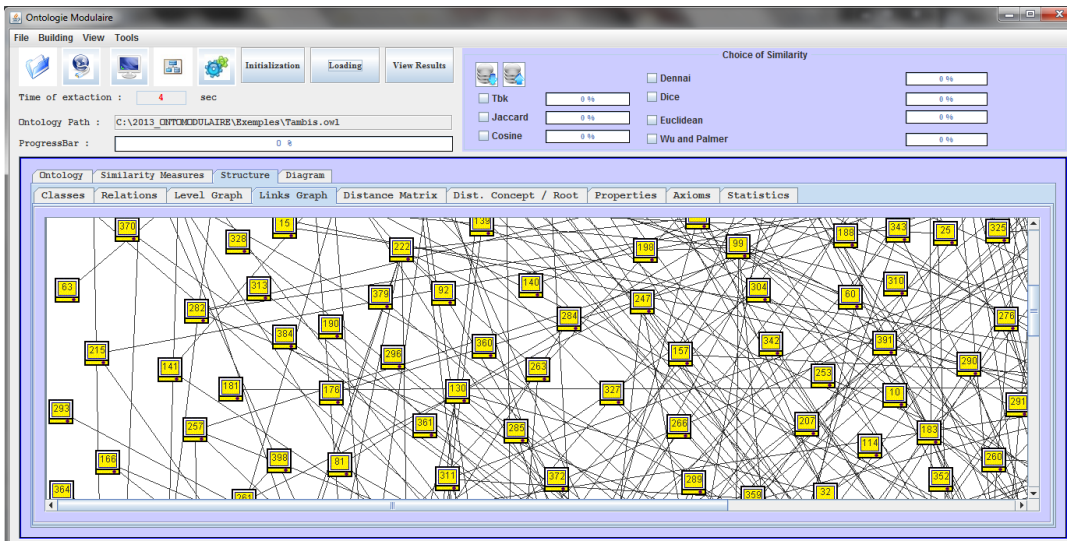
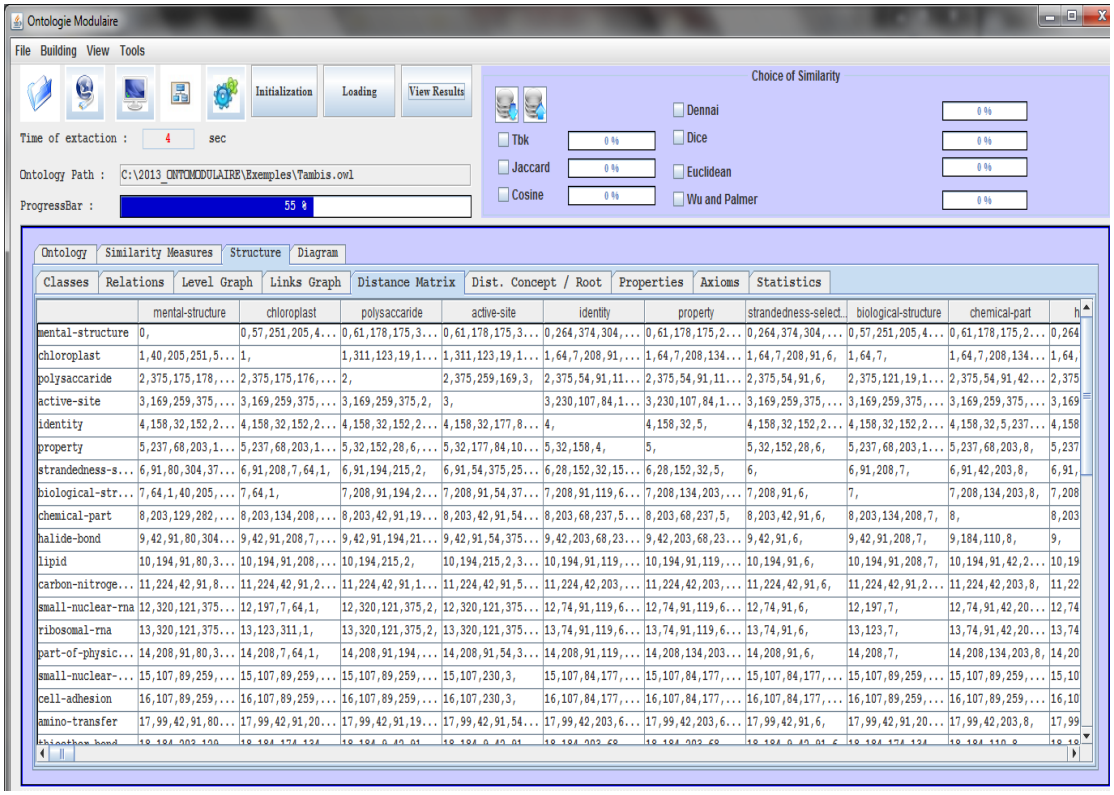



**Figure.B.6 :** Copie d'écran de l'interface de chargement de l'ontologie sous format XML.

Après cette étape, le processus de calcul de matrices de distance est lancé. Le résultat est illustré dans la figure suivante (voir Figure.B.7)




| | mental-structure | chloroplast | polysaccaride | active-site | identity | property | strandedness-select... | biological-structure | chemical-part | h |
|---|---|---|---|---|---|---|---|---|---|---|
| mental-structure | 0, | 0,57,251,205,4... | 0,61,178,175,3... | 0,61,178,175,3... | 0,264,374,304,... | 0,61,178,175,2... | 0,264,374,304,... | 0,57,251,205,4... | 0,61,178,175,2... | 0,264 |
| chloroplast | 1,40,205,251,5... | 1, | 1,311,123,19,1... | 1,311,123,19,1... | 1,64,7,208,91,... | 1,64,7,208,134... | 1,64,7,208,91,6, | 1,64,7, | 1,64,7,208,134... | 1,64, |
| polysaccaride | 2,375,175,178,... | 2,375,175,176,... | 2, | 2,375,259,169,3, | 2,375,54,91,11... | 2,375,54,91,11... | 2,375,54,91,6, | 2,375,121,19,1... | 2,375,54,91,42... | 2,375 |
| active-site | 3,169,259,375,... | 3,169,259,375,... | 3,169,259,375,2, | 3, | 3,230,107,84,1... | 3,230,107,84,1... | 3,169,259,375,... | 3,169,259,375,... | 3,169,259,375,... | 3,169 |
| identity | 4,158,32,152,2... | 4,158,32,152,2... | 4,158,32,152,2... | 4,158,32,152,2... | 4, | 4,158,32,5, | 4,158,32,152,2... | 4,158,32,152,2... | 4,158,32,5,237... | 4,158 |
| property | 5,237,68,203,1... | 5,237,68,203,1... | 5,32,152,28,6,... | 5,32,177,84,10... | 5,32,158,4, | 5, | 5,32,152,28,6, | 5,237,68,203,1... | 5,237,68,203,8, | 5,237 |
| strandedness-s... | 6,91,80,304,37... | 6,91,208,7,64,1, | 6,91,194,215,2, | 6,91,54,375,25... | 6,28,152,32,15... | 6,28,152,32,5, | 6, | 6,91,208,7, | 6,91,42,203,8, | 6,91, |
| biological-str... | 7,64,1,40,205,... | 7,64,1, | 7,208,91,194,2... | 7,208,91,54,37... | 7,208,91,119,6... | 7,208,134,203,... | 7,208,91,6, | 7, | 7,208,134,203,8, | 7,208 |
| chemical-part | 8,203,129,282,... | 8,203,134,208,... | 8,203,42,91,19... | 8,203,42,91,54... | 8,203,68,237,5... | 8,203,68,237,5, | 8,203,42,91,6, | 8,203,134,208,7, | 8, | 8,203 |
| halide-bond | 9,42,91,80,304... | 9,42,91,208,7,... | 9,42,91,194,21... | 9,42,91,54,375... | 9,42,203,68,23... | 9,42,203,68,23... | 9,42,91,6, | 9,42,91,208,7, | 9,184,110,8, | 9, |
| lipid | 10,194,91,80,3... | 10,194,91,208,... | 10,194,215,2, | 10,194,215,2,3... | 10,194,91,119,... | 10,194,91,119,... | 10,194,91,6, | 10,194,91,208,7, | 10,194,91,42,2... | 10,19 |
| carbon-nitroge... | 11,224,42,91,8... | 11,224,42,91,2... | 11,224,42,91,1... | 11,224,42,91,5... | 11,224,42,203,... | 11,224,42,203,... | 11,224,42,91,6, | 11,224,42,91,2... | 11,224,42,203,8, | 11,22 |
| small-nuclear-rna | 12,320,121,375... | 12,197,7,64,1, | 12,320,121,375,2, | 12,320,121,375... | 12,74,91,119,6... | 12,74,91,119,6... | 12,74,91,6, | 12,197,7, | 12,74,91,42,20... | 12,74 |
| ribosomal-rna | 13,320,121,375... | 13,123,311,1, | 13,320,121,375,2, | 13,320,121,375... | 13,74,91,119,6... | 13,74,91,119,6... | 13,74,91,6, | 13,123,7, | 13,74,91,42,20... | 13,74 |
| part-of-physic... | 14,208,91,80,3... | 14,208,7,64,1, | 14,208,91,194,... | 14,208,91,54,3... | 14,208,91,119,... | 14,208,134,203... | 14,208,91,6, | 14,208,7, | 14,208,134,203,8, | 14,20 |
| small-nuclear-... | 15,107,89,259,... | 15,107,89,259,... | 15,107,89,259,... | 15,107,230,3, | 15,107,84,177,... | 15,107,84,177,... | 15,107,84,177,... | 15,107,89,259,... | 15,107,89,259,... | 15,10 |
| cell-adhesion | 16,107,89,259,... | 16,107,89,259,... | 16,107,89,259,... | 16,107,230,3, | 16,107,84,177,... | 16,107,84,177,... | 16,107,84,177,... | 16,107,89,259,... | 16,107,89,259,... | 16,10 |
| amino-transfer | 17,99,42,91,80... | 17,99,42,91,20... | 17,99,42,91,19... | 17,99,42,91,54... | 17,99,42,203,6... | 17,99,42,203,6... | 17,99,42,91,6, | 17,99,42,91,20... | 17,99,42,203,8, | 17,99 |

**Figure.B.7 :** Copie d'écran de l'interface de la matrice de distance après calcul.

Comme nous avons mentionné dans les chapitres précédents que certaines mesures de similarités utilisées dans notre travail nécessitent le calcul de la distance minimale entre un concept et la racine (root), nous avons un processus de calcul de distance de n'importe quel concept vers sa racine R. Cela est illustré par figure suivante (Figure.B.8).

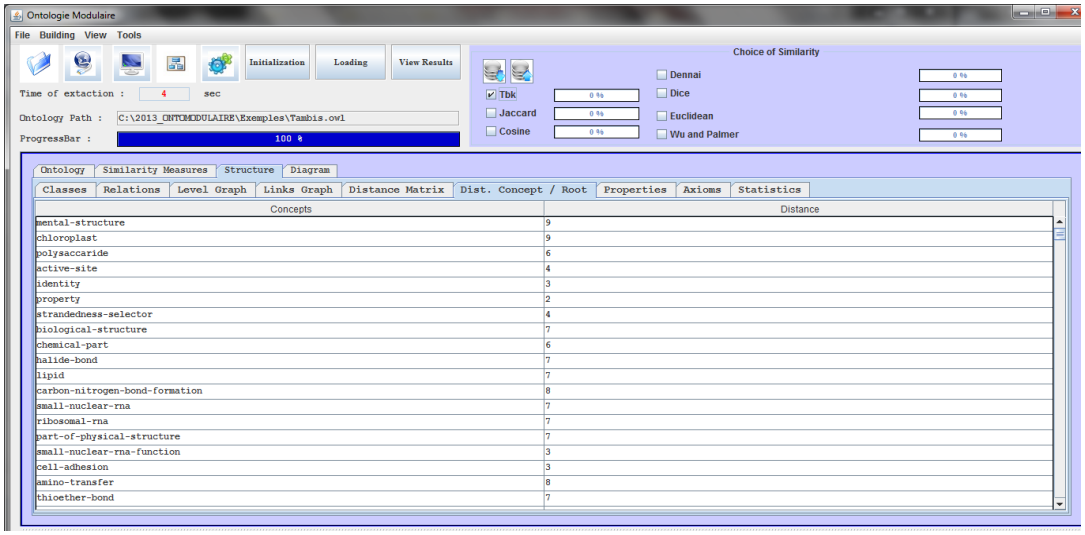



**Figure.B.8 :** Copie d'écran de l'interface de calcul de distance con/root

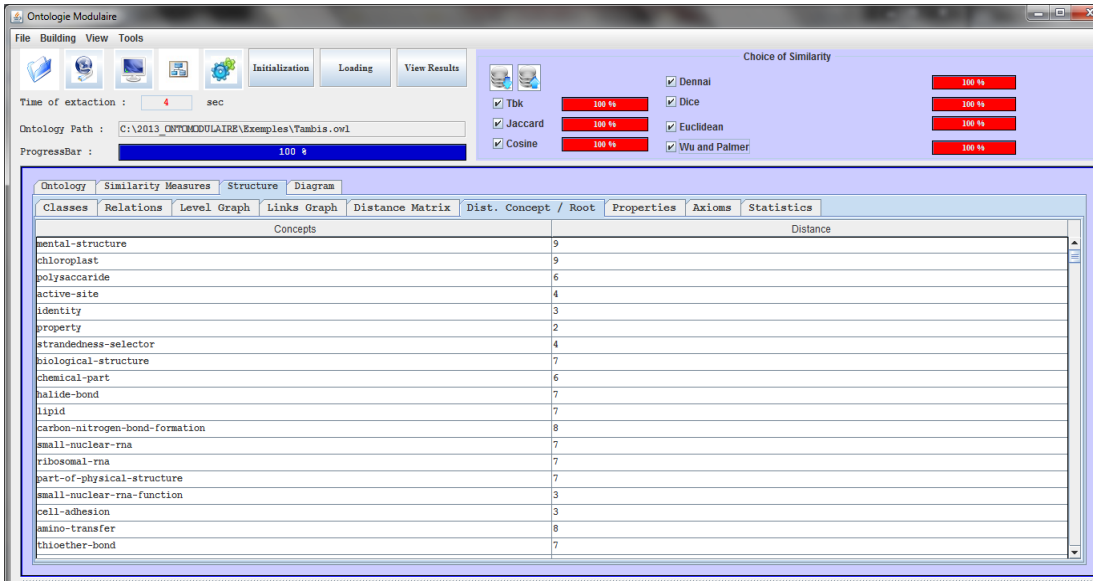
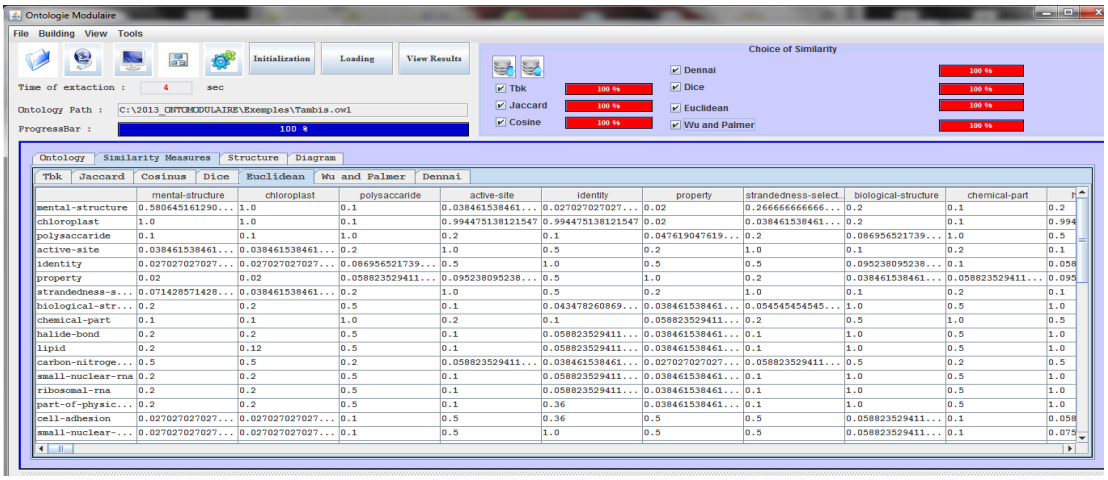



**Figure.B.9 :** Capture d'écran de l'interface de calcul des mesures de similarités.

Avant de calculer les mesures de similarité, il faut tout d'abord cocher les cases associées à ces mesures (voir Figure.B.10).

Ensuite, les résultats de calcul sont affichés. Voir (Figure.B.10)




| | mental-structure | chloroplast | polysaccaride | active-site | identity | property | strandedness-select. | biological-structure | chemical-part | |
|---|---|---|---|---|---|---|---|---|---|---|
| mental-structure | 0.580645161290... | 1.0 | 0.1 | 0.038461538461... | 0.027027027027... | 0.02 | 0.266666666666... | 0.2 | 0.1 | 0.2 |
| chloroplast | 1.0 | 1.0 | 0.1 | 0.994475138121547 | 0.994475138121547 | 0.02 | 0.038461538461... | 0.2 | 0.1 | 0.994 |
| polysaccaride | 0.1 | 0.1 | 1.0 | 0.2 | 0.1 | 0.047619047619... | 0.2 | 0.086956521739... | 1.0 | 0.5 |
| active-site | 0.038461538461... | 0.038461538461... | 0.2 | 1.0 | 0.5 | 0.2 | 1.0 | 0.1 | 0.2 | 0.1 |
| identity | 0.027027027027... | 0.027027027027... | 0.086956521739... | 0.5 | 1.0 | 0.5 | 0.5 | 0.095238095238... | 0.1 | 0.058 |
| property | 0.02 | 0.02 | 0.058823529411... | 0.095238095238... | 0.5 | 1.0 | 0.2 | 0.038461538461... | 0.058823529411... | 0.095 |
| strandedness-s... | 0.071428571428... | 0.038461538461... | 0.2 | 1.0 | 0.5 | 0.2 | 1.0 | 0.1 | 0.2 | 0.1 |
| biological-str... | 0.2 | 0.2 | 0.5 | 0.1 | 0.043478260869... | 0.038461538461... | 0.054545454545... | 1.0 | 0.5 | 1.0 |
| chemical-part | 0.1 | 0.1 | 1.0 | 0.2 | 0.1 | 0.058823529411... | 0.2 | 0.5 | 1.0 | 0.5 |
| halide-bond | 0.2 | 0.2 | 0.5 | 0.1 | 0.058823529411... | 0.038461538461... | 0.1 | 1.0 | 0.5 | 1.0 |
| lipid | 0.2 | 0.12 | 0.5 | 0.1 | 0.058823529411... | 0.038461538461... | 0.1 | 1.0 | 0.5 | 1.0 |
| carbon-nitroge... | 0.5 | 0.5 | 0.2 | 0.058823529411... | 0.038461538461... | 0.027027027027... | 0.058823529411... | 0.5 | 0.2 | 0.5 |
| small-nuclear-rna | 0.2 | 0.2 | 0.5 | 0.1 | 0.058823529411... | 0.038461538461... | 0.1 | 1.0 | 0.5 | 1.0 |
| ribosomal-rna | 0.2 | 0.2 | 0.5 | 0.1 | 0.058823529411... | 0.038461538461... | 0.1 | 1.0 | 0.5 | 1.0 |
| part-of-physic... | 0.2 | 0.2 | 0.5 | 0.1 | 0.36 | 0.038461538461... | 0.1 | 1.0 | 0.5 | 1.0 |
| cell-adhesion | 0.027027027027... | 0.027027027027... | 0.1 | 0.5 | 0.36 | 0.5 | 0.5 | 0.058823529411... | 0.1 | 0.058 |
| small-nuclear-... | 0.027027027027... | 0.027027027027... | 0.1 | 0.5 | 1.0 | 0.5 | 0.5 | 0.058823529411... | 0.1 | 0.075 |

**Figure.B.10 :** Capture d'écran de l'interface de visualisation des résultats de calcul des sept mesures de similarités (Tbk ;Jaccard,Cosine,Dice,Euclidienne,Wu and Palmer et Dennai).

## B.2 Conclusion

Dans cette annexe B, nous avons vu les différentes interfaces de notre système de partitionnement des ontologies sont conçues de manière à être simples, naturelles et de compréhension et d'utilisation faciles.