Chapitre 2 : Les ontologies

1. Introduction

La masse de plus en plus croissante d'information dans tous les domaines a généré un besoin capital d'organisation et de structuration des contenus de documents, disponibles généralement sur le web. Les ontologies en sont un moyen prometteur et qui ne cesse de donner ses preuves. Leurs applications sont multiples : indexation, recherche d'informations, traduction automatique, e-Learning etc.

Les principaux buts de la construction des ontologies sont la partageabilité, la portabilité, la réutilisabilité et la capitalisation de la connaissance et de l'expertise d'un domaine. Parce que l'information n'est pas statique, parce qu'elle se modifie, s'enrichisse, s'altère avec le temps et qu'elle vienne de différentes sources, nous avons besoin d'outils et de modèles qui permettent aux utilisateurs et aux experts du domaine de constituer, consulter et maintenir à jour leurs connaissances du domaine.

2. Définitions

Le mot *ontologie* qui vient du grec ***ontos*** =être et ***logos***= études, appartient à la philosophie ancienne grecque, Aristote le définit comme la science de l'Être en tant qu'être (**Welty & Guarino, 2001**). Il est difficile de définir ce qu'est une ontologie d'une façon définitive. Le mot est en effet employé dans des contextes très différents touchant à la philosophie, la linguistique ou l'intelligence artificielle.

Bien que des débats préexistent, nous parlons plus souvent d'ontologies (au pluriel) afin de refléter les multiples facettes que recouvre cette appellation (**Baneyx, 2007**). Guarino (**Guarino, 1996**) et Dameron (**Dameron, 2003**), abordent les différentes définitions de la littérature afin d'examiner le type de représentation des connaissances dénoté par le terme ontologie. En 1993, Gruber propose une première définition « *une ontologie est une spécification explicite d'une conceptualisation* », (**Gruber, 1993**). L'expression *spécification explicite* signifie, que la conceptualisation est représentée dans un langage qu'il soit naturel (arabe, français..) ou formel (logique de description, graphes conceptuels ..).

Une autre définition, peut être plus rigoureuse : « *Une ontologie implique une certaine vue du monde par rapport à un domaine donné. Cette vue est souvent conçue comme un ensemble de concepts (entités, attributs, processus, leurs définitions et leurs interrelations). On appelle cela une conceptualisation* » (**Charlet, 1996**). Une ontologie peut prendre différentes formes mais elle inclura nécessairement un vocabulaire de termes et une spécification de leur signification (**Baneyx & Charlet, 2006**). En résumé, nous pouvons définir une ontologie comme l'ensemble représentatif et exhaustif des termes d'un domaine donné avec toutes les relations qui les relient.

3. Constituants d'une ontologie

Une ontologie en tant qu'artefact informatique, comme nous l'avons défini dans la section2, est donc constituée de termes, leur définition, leurs propriétés, des contraintes sur les propriétés, des individus, et des relations. Dans ce qui suit, nous allons aborder ces concepts plus en détails.

3.1. Les concepts

Un concept, également appelé *classe* dans certains travaux ou outils, représente l'idée que l'on se fait d'un terme : le contenu. Il est porteur d'une connaissance. Il peut designer un objet concret comme :(***كتاب*** *=livre)* ou abstrait comme *:(* ***تقوى*** *= piété). Livre*, désignant dans notre contexte, un livre sacré.

3.2. Les propriétés

La propriété est une caractéristique qui qualifie un concept et qui peut généralement être dotée d'une valeur. Si nous prenons l'exemple précédent (**كتاب** =livre) nous pouvons designer quelques propriétés comme : (**الاسم_الكتاب** = titre_du_livre), (**اسم_النّبي** = nom_du_prophète) à qui il a été révélé, (**عدد_السور** = nombre_de_chapitre), ceci bien sûr, dans un certain contexte.

3.3. Les facettes

Les facettes sont des restrictions sur les valeurs des propriétés, si nous prenons l'exemple cité dans la section 3.2., La facette de (**الاسم_الكتاب** = titre_du_livre) sera une liste de tous les livres saints, en l'occurrence (**القرآن** =Coran,**الانجيل** =Bible, **التوراة**=Torah,**الزابور**=Zabur, **الصحف**= Souhouf)**.** Quant aux facettes de (**عدد_السور** = nombre_de_chapitre), ce sera tout simplement un entier.

3.4. Les instances

Les instances d'un concept concret sont des éléments singuliers de ce concept, aussi appelées individus dans certains travaux. (**القرآن** =Coran, **الإنجيل** =Bible, **التوراة**=Torah, **ألزبور** =Zabur, **الصحف**= Suhuf) sont des instances de (**كتاب** =livre). Les instances ne sont nécessaires que lorsque l'objectif de l'ontologie est de servir à la construction d'une base de connaissances.

3.5. Les relations

Les relations sont un type d'interaction entre deux concepts. La relation la plus utilisée est sans doute celle qui établit la hiérarchie de la structure ontologique, c'est la relation de subsomption (**عبارة_عن** = *est_un).*

B *est_un* A, exprime le fait que le concept B est un sous concept du concept A, dans le sens où B hérite de toutes les propriétés de A et a forcement des propriétés spécifiques.

Exemple : (**إنسان** *عبارة_عن* **رجل**) et (**إنسان** *عبارة_عن* **امرأة**)

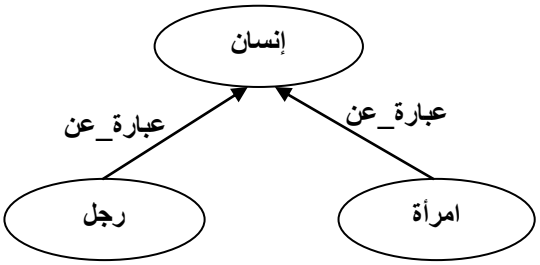

Figure 2: La relation de subsomption

Il existe deux types de relations : celles qui sont indépendantes du domaine et celles qui sont étroitement liées au domaine choisi.

Les relations indépendantes du domaine sont générales et peuvent être utilisées dans n'importe quel champ de spécialité, les plus connues sont (**عبارة_عن**= *est_un*), (**نوع_من**= *sorte_de*), (**جزء_من**= *partie_de*) etc.

Les relations dépendantes du corpus, spécifiques à un domaine, ont un sens précis dans le domaine utilisé.

Exemple :

(*أُنزل_على*= est_révélé_à) dans (**محمد** *أنزل_على* **القرآن**).

Cette relation ne peut être utilisée que dans certains domaines et elle a un sens bien précis par rapport à ce domaine.

4. Stratégies de construction d'une ontologie

Il existe deux stratégies de construction d'une ontologie : ascendante et descendante.

Dans l'ascendante, on débute par les feuilles c'est-à-dire les concepts terminaux, ce sont les plus spécifiques puis on généralise au fur et à mesure que nous montons dans la hiérarchie conceptuelle. Par exemple nous allons commencer par (**طفل** vers **رجل** vers **إنسان** vers **كائن_حي** vers **كائن**).

La stratégie descendante consiste à commencer par les concepts les plus génériques et développer la structure vers les concepts les plus spécifiques, elle nous permet de nous arrêter au niveau de détail désiré. Dans ce cas nous commençons par (**كائن** vers **كائن_حي** vers **إنسان** vers **رجل** vers **طفل**).

Toute fois, nous pouvons combiner les deux stratégies en commençant par des concepts saillants relativement au domaine, puis selon le besoin, étendre vers le spécifique ou le générique. Dans l'exemple précédent si nous considérons que (**إنسان**) est important par rapport à notre domaine, nous commençons par ce concept et on peut aller, soit vers le haut, soit vers le bas. Donc de (**إنسان** vers **رجل** vers **طفل**), ou l'inverse de (**إنسان** vers **كائن_حي** vers **كائن**).

D'après l'expérience aucune stratégie n'est meilleure par rapport aux autres. La stratégie choisie, dépend principalement du point de vue du concepteur par rapport au domaine à modéliser et à l'objectif de l'ontologie.

5. Typologies des ontologies

Il existe plusieurs typologies des ontologies, proposées par des groupes de recherche selon l'objectif principal pour lequel l'ontologie a été conçue. Nous présentons dans ce qui suit les plus importantes.

5.1. Typologie de Uschold et Grüninger

Uschold et Grüninger (**Uschold & Grüninger, 1996**) ont classé les ontologies selon leur degré de formalisme.

5.1.1. les ontologies hautement informelles

Ce sont des ontologies écrites en langage naturel, elles ne sont pas traitées par une machine.

5.1.2. les ontologies semi-informelles

Elles utilisent un langage naturel structuré et limité, elles se situent entre les ontologies formelles et informelles.

5.1.3. les ontologies rigoureusement formelles

Elles sont définies dans un langage contenant une sémantique formelle, elles sont facilement traitées par une machine peuvent être intégrées dans des applications.

5.2. Typologie de Gómez-Pérez

Cette typologie a été proposée par Gomez-Pérez, à l'université de Madrid (**Gomez-Pérez, 2004**). Elle s'intéresse aux objets que modélisent les ontologies, elle les classe ainsi :

5.2.1. Ontologies pour la représentation des connaissances

Elles sont utilisées généralement lors de la construction d'un système à base de connaissance.

5.2.2. Ontologies de domaine

Elles servent à fournir une modélisation d'un domaine de connaissance donné comme la médecine ou une spécialité en médecine comme l'ophtalmologie.

5.2.3. Ontologies de haut niveau

Elles sont généralement exprimées en termes de scenarios, événements, temps et objets. On peut y greffer n'importe quelle autre ontologie du domaine.

5.2.4. Ontologies génériques

Elles ne sont pas spécifiques à un domaine précis, Aussi appelée méta-ontologie, véhicule des connaissances génériques qui, bien que moins abstraites que celles modélisées dans l'ontologie de haut niveau, doivent être assez générales pour être réutilisées dans différents domaines. Elles organisent des connaissances factuelles ou des connaissances visant à résoudre des problèmes génériques d'un ou de plusieurs domaines**.**

5.2.5. Ontologies de tâches

Elles modélisent le processus d'une tache spécifique ou d'une activité particulière, tel un diagnostic d'une panne par exemple.

5.2.6. Ontologies d'application

Elles décrivent une tache dans un domaine bien défini (**Mizoguchi & Ikeda, 1997**), tel que l'aide au diagnostic d'une panne dans une turbine à vapeur utilisée en électricité par exemple (**Klai & Khadir, 2009**).

5.3. Typologie en fonction du niveau de granularité

5.3.1. Granularité fine

Elles sont très détaillées, possédant ainsi un vocabulaire plus riche d'un domaine, pouvant être utilisées dans des applications spécifiques.

5.3.2. Granularité large

Elles engobent un vocabulaire moins détaillé, s'arrêtant à un certain niveau de spécificité auxquelles on peut greffer d'autres ontologies plus détaillées.

6. Méthodologie de construction

Une méthodologie étant les procédures de travail, les étapes qui décrivent le pourquoi et le comment de la conceptualisation. Bien que l'on s'accorde volontiers à dire qu'il n'existe pas de méthodologie de construction d'ontologie, ceci n'est vrai que dans le sens où aucune méthodologie proposée ne fait l'unanimité de la communauté de l'ingénierie ontologique. En effet toutes les méthodologies proposées respectent juste un certain processus de l'objectif pour lequel elles ont été crées. Nous abordons dans la section suivante les méthodologies les plus importantes qui ont réussi à acquérir une certaine crédibilité dans le monde des concepteurs d'ontologies.

6.1. Méthodologie de Uschold et Grüninger

Elle propose plusieurs étapes pour construire des ontologies manuellement (**Uschold & Grüninger, 1996**), ces étapes se résument en:

a) Identifier l'objectif souhaité et spécifier le domaine concerné

b) Construire l'ontologie et pour cela définir les concepts, les relations clés et produire des définitions textuelles précises et non ambigües de ces concepts

c) Evaluer le résultat

d) Documenter le modèle en éditant des recommandations précises pour chaque étape.

6.2. Methontology

Proposée par Fernandez et son équipe à l'université de Madrid (**Fernandez, 1997**), cette méthodologie est largement utilisée dans le domaine. Pour la construction de l'ontologie, ils procèdent en dix étapes:

- **a)** Construire le glossaire des termes qui seront inclus dans l'ontologie, préciser leur définition en langage naturel, identifier leurs synonymes et leurs acronymes
- **b)** Construire des taxinomies de concepts
- **c)** Construire des diagrammes de relations binaires
- **d)** Construire le dictionnaire de concepts qui inclut, pour chaque concept, ses attributs d'instance, ses attributs de classe et ses relations
- **e)** Décrire en détail chaque relation binaire
- **f)** Décrire en détail chaque attribut d'instance
- **g)** Décrire en détail chaque attribut de classe
- **h)** Décrire en détail chaque constante (les constantes donnent des informations sur le domaine de connaissances)
- **i)** Décrire les axiomes formels
- **j)** Décrire les règles utilisées pour contraindre le contrôle et pour inférer des valeurs aux attributs.

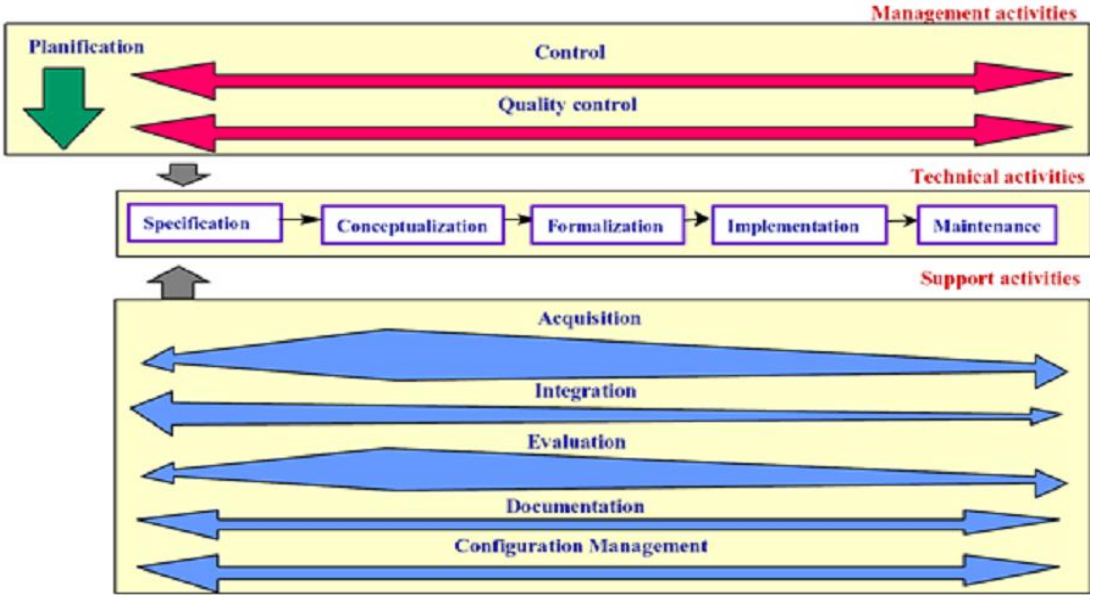

Figure 3: Processus de développement d'ontologie de Méthontology **(Corcho & al, 2005)**

6.3. Méthodologie de Guarino et Welty

C'est une méthode qui n'est pas un guide de construction d'ontologies, mais plutôt une étape permettant la vérification et la correction d'une structure ontologique construite un peu anarchiquement, entre autres les règles de subsomptions. Elle est à incorporer dans le cycle de vie d'une ontologie (**Welty & Guarino, 2001**).

6.4. Méthode ARCHONTE

ARCHONTE (ARCHitecture for **ONT**ological **E**laborating), est proposée par Bruno Bachimont (**Bachimont, 2000**), cette méthode comporte trois étapes :

- **a)** Choisir les termes pertinents du domaine et normaliser leur sens en précisant les relations de similarités et de différences que chaque concept entretient avec ses concepts frères et son concept père
- **b)** Formaliser les connaissances, c'est-à-dire ajouter éventuellement des propriétés à des concepts, des axiomes . . .
- **c)** Opérationnaliser dans un langage de représentation des connaissances

7. Formalismes de représentation des ontologies

La formalisation en elle-même n'est pas une étape incontournable de la méthodologie de construction. Une utilisation simple d'une ontologie en recherche d'information par exemple ne nécessite pas forcément une formalisation. Cependant certaines taches complexes ont besoin de cette étape, qui assure entre autre, l'intégrité, la consistance et la complétude du contenu ontologique surtout si celui-ci se trouve volumineux, elle permet par la suite l'opérationnalisation et l'intégration de l'ontologie dans les différentes applications visées.

Pour pouvoir raisonner sur une ontologie, il faut formaliser la connaissance. Formaliser les connaissances d'un domaine, consiste à les coder en un langage formel de description de connaissances, pour qu'elles puissent être traitées par une machine. Formaliser une ontologie c'est permettre de vérifier sa hiérarchie, sa consistance et pouvoir l'intégrer facilement dans d'autres systèmes et raisonner dessus. Il existe plusieurs formalismes de représentations des connaissances, mais les plus utilisés dans les ontologies sont principalement :

- Les graphes conceptuels
- Les logiques de description

7.1. Graphes conceptuels

Les graphes conceptuels sont introduits par (**Sowa, 1984**) et appartiennent à la famille des réseaux sémantiques. Le modèle des graphes conceptuels comprend deux parties :

a) Une partie terminologique pour les concepts, les relations et les instances. Le niveau terminologique comprend donc, trois ensembles disjoints : L'ensemble des types de concepts (Tc), l'ensemble des types de relations (Tr) et l'ensemble des marqueurs individuels (M)

b) Une partie assertionnelle pour la représentation des assertions du domaine de connaissances.

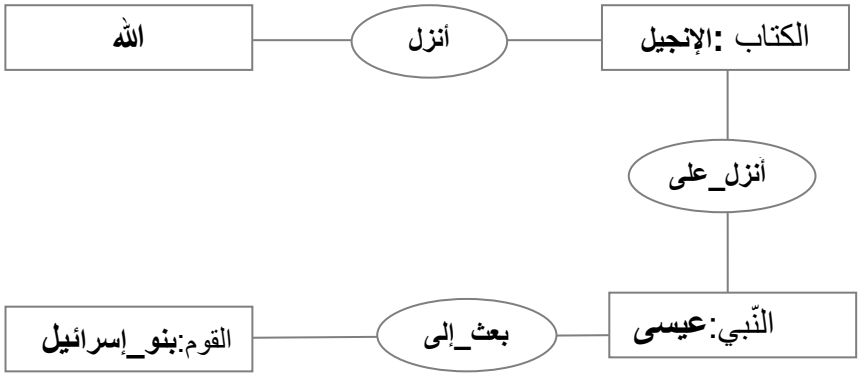

Figure 4: Un exemple de graphe conceptuel

Dans l'exemple précédent nous avons :

Tc= {**الله، النّبي،الكتاب،القوم**}

Tr={ **أُنزل_على، أنزل،بعث_إلى**}

M= { **الإنجيل،عيسى،بنو_إسرائيل** }

7.2. Les logiques de description

Les logiques de description sont des langages formels permettant de représenter des propriétés pour des ensembles d'objets. Elles se basent sur la logique du premier ordre. Elles ont pour vocation d'après leurs auteurs (**Nardi & Brachman, 2003**), d'enrichir les représentations sous forme de réseaux sémantiques ou frames, notamment au niveau des relations. Les logiques de descriptions étant le formalisme que nous avons choisi pour représenter la connaissance dans ce travail, nous y reviendrons dans la section dédiée à la formalisation de l'ontologie.

8. Langages de représentation des ontologies

L'un des principaux avantages d'une ontologie est la portabilité. Pour pouvoir exploiter une ontologie et la partager par un grand nombre d'utilisateur, il faut l'exprimer dans un langage permettant son utilisation sur différentes applications et plateformes. Ce langage doit répondre aux exigences des utilisateurs potentiels de cette ontologie. Il existe un grand nombre de langages développés à cet effet.

Tous sont basés sur la syntaxe XML, bien que XML lui-même ne soit pas un langage pour représenter les connaissances ontologiques. Nous allons citer dans

cette section les plus utilisés et donc les plus enclins à respecter une certaine standardisation.

8.1. RDF & RDFs

Les initiales RDF correspondent à « Resource Description Framework », ou cadre de description de ressources en français, le « s » de schémas est une extension de RDF. Une ressource est simplement une *chose* : Une personne, un livre, un clavier, un article de publication, un bureau, une idée, toute *chose* qui peut être décrite. RDF est un cadre d'applications utilisant l'architecture du Web pour décrire une ressource. Tel HTML qui permet de relier des documents à d'autres documents sur le Web, RDF permet de relier une ressource à d'autres ressources sur le Web.

Comme tous ses prédécesseurs, ce langage se base sur la syntaxe d'XML. Doté d'un schéma de représentation riche, incluant des classes, sous-classes, propriétés, sous-propriétés et des règles d'héritage de propriétés.

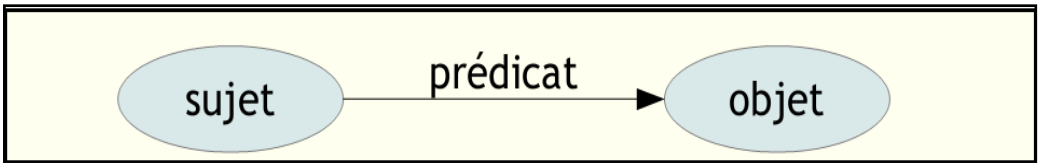

Figure 5: Le triplet RDF

Dans un graphe, chaque triplet représente l'existence d'une relation entre les choses symbolisées par les nœuds qui sont joints. La structure objet-classe des RDFS permet de définir des objets du domaine et leurs relations pour rendre compte d'une ontologie (**Baneyx, 2007**).

8.2. OIL

OIL[11] (Ontology Inference Layer) est un langage de représentation d'ontologie qui dérive de RDF. Les principaux fondements du langage OIL sont les langages de frame (tels que OKBC, XOL ou RDF) et les logiques de descriptions. OIL a été défini dans l'objectif de permettre la spécification et l'échange d'ontologies.

8.3. DAML et DAML+OIL

[11] http://www.ontoknowledge.org/oil/

DAML+OIL[12] (DARPA[13] *Agent Markup Language*) est un langage permettant la représentation d'ontologies.

DAML est une combinaison de XML et de RDF permettant de spécifier des objets mais également les relations entre ces objets. La dernière version de DAML se combine avec OIL (DAML+OIL). Ce nouveau langage supporte désormais les types de données primitifs (tels qu'on les trouve dans la norme XML Schéma) et la définition d'un certain nombre d'axiomes comme l'équivalence de classes ou de propriétés (**Baneyx, 2007**).

8.4. OWL

Nous avons vu que RDF et RDFS permettent de définir, sous forme de graphes de triplets, des données ou des métadonnées. Cependant, de nombreuses limitations bornent la capacité d'expression des connaissances établies à l'aide de RDF/RDFS. On peut citer, par exemple, l'impossibilité de raisonner et de mener des raisonnements automatisés sur les modèles de connaissances établis à l'aide de RDF/RDFS. C'est ce manque que se propose de combler OWL.

OWL[14] **(***OntologyWeb Language*) a été créé en 2001 par le W3C, hérite du langage DAML+OIL et doit permettre de représenter des ontologies sur le Web. OWL fournit en fait trois sous-langages, d'expressivité croissante, nommés OWL Lite, OWL DL et OWL Full.

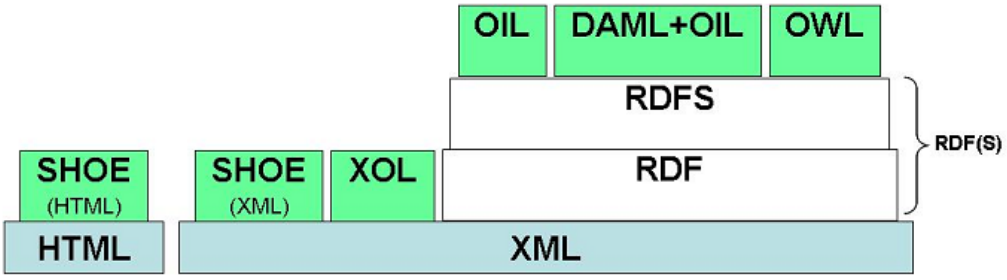

Figure 6: Les langages d'exploitation des ontologies (Gomez-Pérez, 2004)

OWL est devenu un standard du Consortium W3C qui a publié en 2004 une recommandation définissant le langage OWL fondé sur le standard RDF et en

[12] http://www.daml.org/2001/03/daml+oil-index
[13] DARPA: Defense Advanced Research Projects Agency
[14] http://owl.cs.manchester.ac.uk/

spécifiant une syntaxe XML. Plus expressif que RDFS, il tend à détrôner les autres langages et à s'imposer de plus en plus en maitre absolu.

9. Editeurs d'ontologies

Éditer une ontologie avec un outil adéquat permet son affichage sous forme arborescente, en outre l'intégration de plugins appropriés, permet la visualisation des différents concepts avec toutes les relations qui les relient, cela donne une vue plus globale de la disposition des concepts les uns par rapport aux autres. Certains éditeurs vont plus loin en permettant d'importer ou d'exporter une ontologie d'un format vers un autre, ce qui facilite largement sa portabilité et la génération automatique de fichiers OWL/XML ou RDF **(Patil, 2005).**

Nous donnons dans ce qui suit un survol sur les éditeurs les plus importants, certains d'entre eux représentent de véritables plateformes avec de multiples plugins permettant de soumettre des requêtes de vérifier la consistance et de fusionner des ontologies existantes dans différents formats.

9.1. PROTEGE

PROTEGE[15] 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.

[15] http://protege.stanford.edu

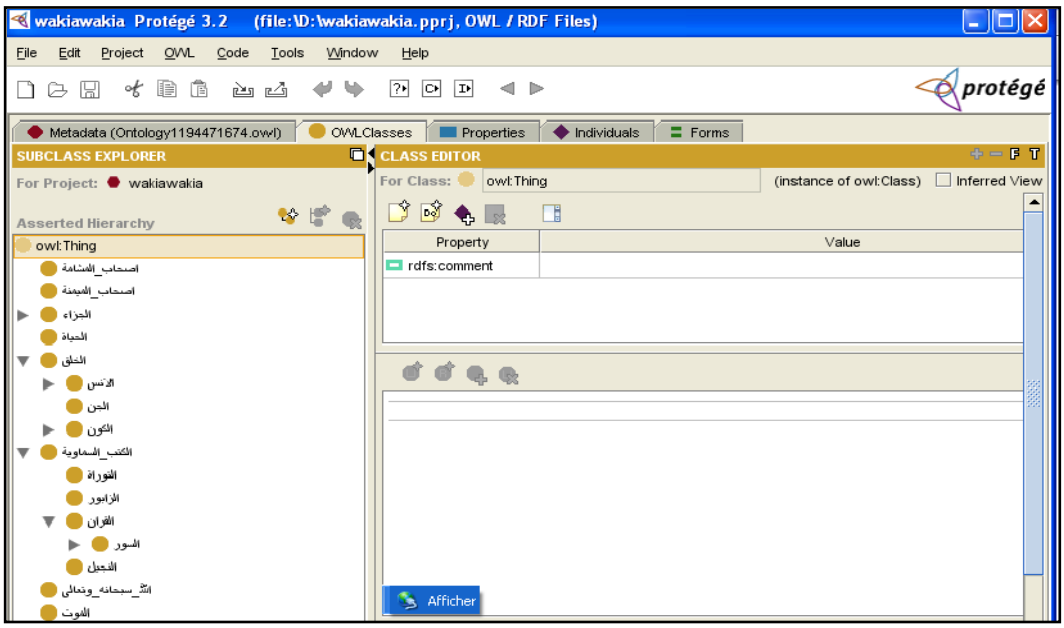

Figure 7: La hiérarchie Ontologique sous PROTEGE

En plus de la visualisation de la hiérarchie ontologique, PROTEGE permet une visualisation graphique à l'aide de plugins comme OntoGraph ou OWL-Viz[16], il dispose de raisonneurs comme Racer[17], Fact++, Hermitt, Pellet. OntoGraph, Fact++, Hermitt, Pellet sont fournis avec PROTEGE.

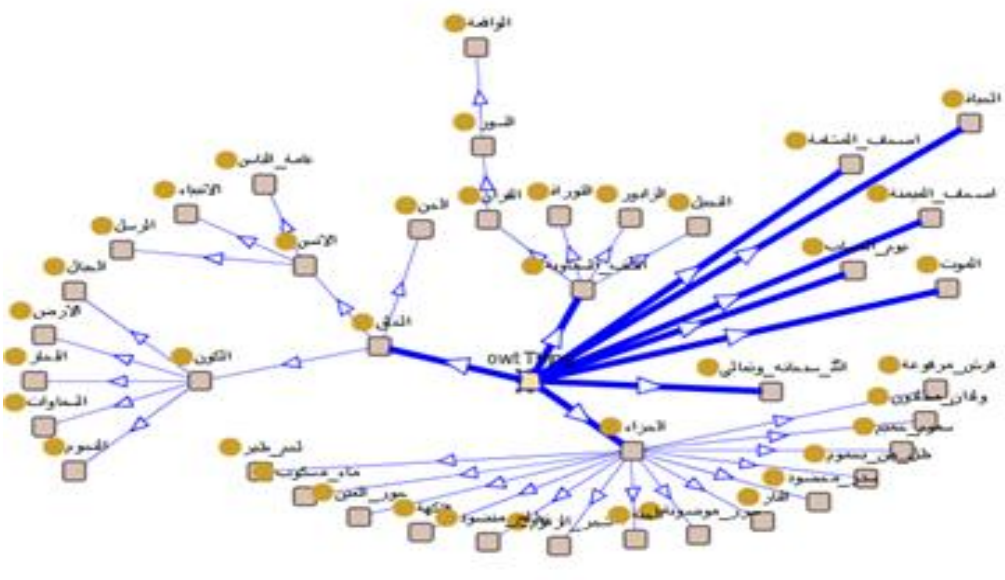

Figure 8: Visualisation d'ontologie en arabe avec PROTEGE **(Zitouni, 2010)**

[16] http://www.co-ode.org/downloads/owlviz/
[17] http://www.racer-systems.com/

9.2. JENA

JENA[18] 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.

9.3. OilEd

OilEd[19] est un éditeur qui a été développé par l'université de Manchester pour éditer des ontologies dans les langages de représentation OIL[20]. Les versions disponibles de OilEd ne constituent pas un environnement complet pour le développement d'ontologies d'envergure.

9.4. OntoEdit

OntoEdit[21] est un outil mis au point par l'institut AIFB de l'université de Karlsruhe et qui est maintenant commercialisé par la société Ontoprise GmbH[22]. Il s'inspire de l'approche par frames. OntoEdit est un des seuls éditeurs, avec DOE (voir *section 9.6*), à s'attaquer au problème de la synonymie.

9.5. WebOde

WebOde[23] est une plateforme en ligne développée par le groupe Ontological Engineering du département d'intelligence artificielle de la faculté d'informatique de l'université polytechnique de Madrid. C'est un éditeur qui assurait le support de Methontology (voir *section 6.2)*.

9.6. DoE

DoE[24] (Differential Ontologies Editor) a été développé à l'Institut National de l'Audiovisuel par R. Troncy et A. Isaac en 2002 (**Isaac, 2005**). L'éditeur DOE offre des interfaces de création, modification et suppression de concepts et de relations, une représentation graphique de l'arbre ontologique, et des fonctionnalités de recherche et de navigation dans la structure créée. L'ontologie

[18] http://incubator.apache.org/jena
[19] http://oiled.semanticweb.org/
[20] http://www.w3.org/TR/daml+oil-reference
[21] http://ontoserver.aifb.uni-karlsruhe.de/ontoedit
[22] http://www.ontoprise.de/
[23] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/downloads/60-webode
[24] http://www.eurecom.fr/~troncy/DOE/

est documentée par des définitions encyclopédiques avec des synonymes et les principes différentiels en plusieurs langues (**Baneyx, 2007**).

9.7. ONTOLINGA

ONTOLINGA[25] fournit un environnement collaboratif distribué pour chercher, créer, éditer, modifier, et utiliser des ontologies en ligne. Le serveur peut supporter jusqu'à 150 utilisateurs actifs. Certains fournissent une description de leurs projets. L'environnement assiste l'utilisateur dans les taches de développement et le maintien et leur permet de partager leur ontologie avec d'autres utilisateurs.

9.8. KAON

KAON[26] (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, TextToOnto[27] et Text2Onto[28] sont open source et codés en Java. KAON2[29] n'est pas open source mais l'exécutable peut être téléchargé.

10. Conclusion

Nous avons abordé dans ce chapitre un tour d'horizon sur les différentes technologies utilisées dans le développement d'ontologies. Nous n'avons présenté qu'une liste succincte mais nous avons tenu à ce qu'elle soit la plus représentative possible des outils existant pour chaque phase de la création d'ontologie. Nous avons mis l'accent autant que cela était possible sur les outils libres et open source, qui peuvent servir beaucoup plus, dans le domaine de la recherche.

Nous n'avons pas développé les sections concernant les outils ou techniques que nous avons adoptées dans ce travail et que nous allons abordées dans les prochains chapitres puisqu'elles y seront exposées plus en détail comme la logique de description par exemple.

[25] http://www.ksl.stanford.edu/software/ontolingua/
[26] http://kaon.semanticweb.org/
[27] http://sourceforge.net/projects/texttoonto/
[28] http://code.google.com/p/text2onto/downloads/
[29] http://kaon2.semanticweb.org/#download