1. Introduction

L'origine du langage OWL est Le World Wide Web Consortium (W3C) qui a mis sur pieds, en Novembre 2001, le groupe de travail « WebOnt », chargé d'étudier la création d'un langage standard de manipulation d'ontologies web. Le premier Working Draft «OWL Web Ontology Language 1.0 Abstract Syntax » paraît en Juillet 2002 et, au final, OWL devient une Recommandation du W3C le 10 Février 2004 ;

Le language OWL est destiné à être utilisé quand l'information contenue dans les documents doit être traitée par des applications, par opposition aux situations où le contenu doit seulement être présenté. OWL peut être employé pour représenter explicitement la signification des termes dans les vocabulaires et les relations entre ces termes.
Cette représentation des termes et de leurs corrélations s'appelle une ontologie. OWL offre plus de facilités pour exprimer la signification et la sémantique que XML, RDF et RDF-S. OWL va ainsi au-delà de ces langages dans sa capacité de représenter le contenu compréhensible par une machine sur le Web.

OWL est une révision du langage d'ontologie du Web DAML+OIL intégrant les leçons apprises de la conception et de l'application de DAML+OIL. [18]

L'objet de cette partie est de présenter les fonctionnalités offertes par OWL .

2. Présentation du langage d'ontologie OWL

Le langage d'ontologie Web OWL [25] est conçu pour décrire et représenter un domaine de connaissance spécifique, en définissant des classes de ressources ou objets et leurs relations; ainsi que de définir des individus et affirmer des propriétés les concernant et de raisonner sur ces classes et individus dans la mesure où le permet la sémantique formelle du langage OWL.
OWL est un standard basé sur la logique de descriptions[11], il est construit sur RDF et RDFS et utilise la syntaxe RDF/XML.
Le langage OWL permet d'étendre les technologies de base (XML, RDF, RDFS) pour apporter :

- ✓ Plus d'interopérabilité (équivalences) ;
- ✓ Plus de raisonnements (logique de description) ;
- ✓ Plus d'évolution (intégration d'ontologies).

Les ontologies OWL se présentent, généralement, sous forme de fichiers texte et de documents OWL.

2.1. Types du langage OWL

Le langage OWL offre trois sous langages d'expression [25][20] conçus pour des communautés de développeurs et d'utilisateurs spécifiques.

- Le langage *OWL Lite* concerne les utilisateurs ayant principalement besoin d'une hiérarchie de classifications et de mécanismes de contraintes simples. Par exemple, quoique OWL Lite gère des contraintes de cardinalité, il ne permet que des valeurs de cardinalité de 0 ou 1.
- Le langage *OWL DL* concerne les utilisateurs souhaitant une expressivité maximum sans sacrifier la complétude de calcul (inférences) et la décidabilité des systèmes de raisonnement. Le langage OWL DL comprend toutes les structures de langage d'OWL avec des restrictions comme la séparation des types (une classe ne peut pas être en même temps un individu ou une propriété, une propriété doit être un individu ou une classe).
- Le langage *OWL Full* concerne les utilisateurs souhaitant une expressivité maximum et la liberté syntaxique de RDF sans garantir le calcul (raisonnement). Par exemple, dans OWL Full, on peut simultanément traiter une classe comme une collection d'individus et comme un individu à part entière. Une autre différence significative par rapport à OWL DL réside dans la possibilité de marquer un objet owl:DatatypeProperty comme étant un objet owl:InverseFunctionalProperty.

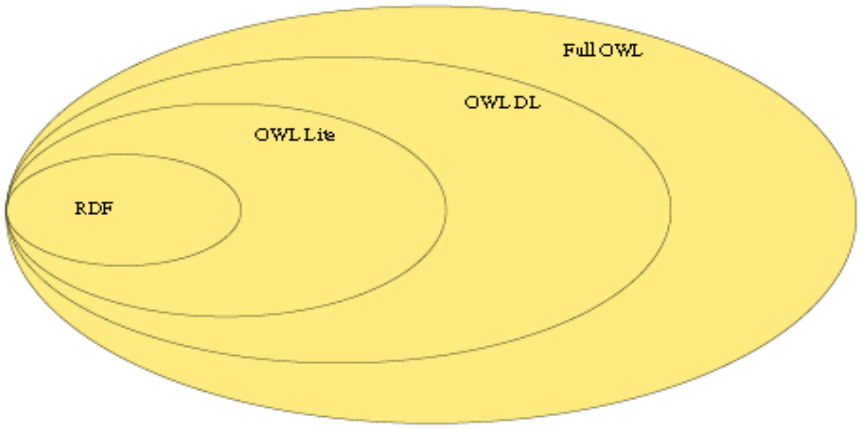

Figure (II.1): Hiérarchie de langage OWL

2.2. Eléments du langage OWL

Nous présentons ici des explications exhaustives du rôle de chacun des éléments constituant une ontologie OWL.

2.2.1. En tête OWL

Une fois les espaces de nommage établis, nous introduisons un ensemble d'assertions sur l'ontologie regroupées dans une balise *owl:Ontology*[25]. Ces balises se chargent de tâches communes comme le nom de l'ontologie (*rdf :about*), les commentaires (*rdfs:comment*), le contrôle de version (*owl:priorVersion*) et l'inclusion d'autres ontologies (*owl:imports*), comme il est détaillé dans l'exemple ci-dessous :

<owl:Ontology ***rdf:about****="Courrier">*
*<****rdfs:comment****>Une ontologie OWL de courrier qui servira d'exemple pour tout le mémoire et même pour XML, RDF et RDFS</rdfs:comment>*
<owl:priorVersion rdf:resource="20081109/Courrier"/>
*<****owl:imports*** *rdf:resource="Documents"/>*
<rdfs:label>Ontologie sur le trafic du courrier</rdfs:label>
...
</owl:Ontology>

2.2.2. Elément classe

Une classe définit un groupe d'individus possédant des caractéristiques similaires. L'ensemble des individus d'une classe est désigné par le terme extension de classe, chacun de ces individus étant alors une instance de la classe. Chaque individu du monde OWL est un membre de la classe *owl:Thing*. Chaque classe définie par l'utilisateur est donc implicitement une sous-classe de *owl:Thing* [25][20].

La déclaration d'une classe owl se fait de six manières [25][20] :

1) Nommer directement les classes : *<owl:Class rdf:ID="Courrier"/>*

Nous utilisons *rdf:ID="Courrier"* pour introduire un nom, comme partie de sa définition. Nous pouvons maintenant appeler la classe Courrier au sein du document par *#Courrier*, par exemple *rdf:resource='#Courrier'*.

Le concept d'héritage existe en OWL à l'aide de la propriété *subClassOf*. Il existe dans toute ontologie OWL une superclasse, nommée Thing, dont toutes les autres classes sont des sousclasses.

Il existe une classe nommée *noThing*, qui est sous-classe de toutes les classes OWL. Cette classe ne peut avoir aucune instance. Nous définissons, ci-dessous, la classe Courrier comme étant une sous-classe de la classe Document.

<owl:Class rdf:ID="Courrier">
*<**rdfs:subClassOf** rdf:resource="#Document" />*
</owl:Class>

2) Enumérer les individus (instances) de la classe :

<owl:Class rdf:ID="Type_destinataires">
*<**owl:oneOf** rdf:parseType="Collection">*
<owl:Thing rdf:about="#A"/>
<owl:Thing rdf:about="#Cc"/>
</owl:oneOf></owl:Class>

3) Restreindre les propriétés (valeur et cardinalité)

Une contrainte de valeur s'applique sur la valeur d'une certaine propriété de l'individu, et une contrainte de cardinalité porte sur le nombre de valeurs qui peut prendre une propriété. La propriété cardinalité permet d'exprimer exactement le nombre d'éléments dans une relation ou son intervalle (par exemple, pour un individu de la classe service, Anom_service est une propriété qui est obligatoire. La contrainte de cardinalité portant sur Anom_service restreindra donc la classe décrite aux individus pour lesquels cette propriété apparaît exactement une fois, comme il est indiqué dans l'exemple ci-dessous).

<owl:Class rdf:ID="service">
<rdfs:subClassOf ><owl:Restriction>
<owl:OnProperty rdf:resource="#Anom_service"/>
*<**owl:cardinality** rdf:datatype="&xsd:int">1 <**rdfs:cardinality** />*
<owl:Restriction/><rdfs:subClassOf />
<rdfs:subClassOf ><owl:Restriction>
<owl:OnProperty rdf:resource="#Acode_service"/>

*<owl:cardinality rdf:datatype="&xsd:int">1<**rdfs:cardinality** />*
<owl:Restriction/><rdfs:subClassOf />
<owl:Class/>

4) Déclarer une nouvelle classe par intersection de deux ou plusieurs classes :

<owl:Class>
*<**owl:intersectionOf** rdf:parseType="Collection">*
<owl:Class rdf:about="#classe1"/>
.....<owl:Class rdf:about="#classen"/>
</owl:intersectionOf>
</owl:Class>

5) Déclarer une classe par union de deux ou plusieurs classes :

<owl:Class rdf:ID="classe">
*<**owl:unionOf** rdf:parseType="Collection">*
<owl:Class rdf:about="#classe1" />
....<owl:Class rdf:about="#classen" />
</owl:unionOf>
</owl:Class>

6) Déclarer une classe qui contient tous les individus dans le domaine de discours n'appartenant pas à une autre classe.

<owl:Class rdf:ID="classe" />
<owl:Class rdf:ID="Nonclasse">
*<**owl:complementOf** rdf:resource="#classe"/>*
</owl:Class>

2.2.3. Individus d'une classe

Il suffit de déclarer l'appartenance d'un individu (instance) comme membre d'une classe pour l'introduire[25][20]. Un individu s'exprime de la manière suivante :

<Courrier rdf:ID="Courrier14582">

L'exemple suivant est identique au précédent du point de vue de la signification :

*<owl:Thing **rdf:ID=" Courrier14582"** />*
<owl:Thing rdf:about="#Courrier14582">
<rdf:type rdf:resource="#Courrier"/>
</owl:Thing>

L'élément *rdf:type* est une propriété RDF liant un individu à une classe dont il est membre.

Une difficulté peut apparaître dans le nommage des individus concerne la non-unicité éventuelle des noms attribués aux individus. Par exemple, un même individu pourrait

être désigné de plusieurs façons différentes. C'est la raison pour laquelle OWL propose un mécanisme permettant de lever cette ambigüité, à l'aide des propriétés : *owl:sameAs*, *owl:diffrentFrom* et *owl:allDifferent*[25].

2.2.4. Elément propriété OWL

Les propriétés nous permettent d'affirmer des faits généraux sur les membres des classes et des faits particuliers sur les individus. Une propriété est une relation binaire. On distingue deux types de propriété [25][20] :

- Les propriétés d'objets (*owl:ObjectProperty*), qui sont des relations entre les instances de deux classes.
- Les propriétés de types de données (*owl:DatatypeProperty*), qui sont des relations entre des instances de classes, des littéraux RDF [11] et des types de donnée du schéma XML[3].

a. Propriétés d'objets

Il existe plusieurs façons de restreindre la relation, lorsque nous définissons une propriété. Nous pouvons définir un domaine et une image. Nous pouvons définir la propriété comme une spécialisation (sous-propriété) d'une propriété existante. Des restrictions plus élaborées sont possibles [25][20], comme il est formulé dans l'exemple ci-dessous :

```
<owl:ObjectProperty rdf:ID="EnvoyéPar">
<rdfs:domain rdf:resource="#courrier"/>
<rdfs:range rdf:resource="#emetteur"/>
<rdfs:subPropertyOf rdf:resource="#implique"/>
</owl:ObjectProperty>
```

Dans OWL, une séquence d'éléments sans opérateur explicite représente une conjonction implicite. La propriété *"EnvoyéPar"* a pour domaine la classe *"courrier"* et pour image la classe *"emetteur"*. C'est-à-dire qu'elle relie des instances de la classe *"courrier"* à des instances de la classe *"emetteur"*. Plusieurs domaines signifient que le domaine de la propriété est constitué par l'intersection des classes identifiées (idem pour les images).

b. Propriétés de types de données

Dans le cas d'une propriété de type de donnée, l'image de la propriété est un type de donnée. C'est à dire l'image est un ensemble d'individus possédant des types de données. Les propriétés de types de données recouvrent des littéraux RDF ou des types simples définis conformément aux types de données du schéma XML. Le langage OWL utilise la plupart des types de données intégrés du schéma XML [25][20][3].

Les types de données intégrés du schéma XML plus le type *rdfs:Literal* constituent les types de données intégrés du langage OWL [25].

L'exemple ci-dessous montre que la propriété objet relie les éléments de la classe *"courrier"* à des valeurs de type chaine de caractère :

```
<owl:DatatypeProperty rdf.id="objet">
<rdfs:domain rdf:resource="#courrier" />
<rdfs:range rdf:resource="xsd:string"/>
</owl:DatatypeProperty>
```

c. Propriétés des individus

Dans l'exemple ci-dessous, nous décrivons d'abord les individus de la classe *"courrier"* puis nous définissons leurs objets :

```
<courrier rdf:ID=" Courrier14582">
<objet rdf:resource="#visite présidentielle" />
<numéro_courrier rdf:resource="#14582"/>
…….
</courrier>
```

d. Caractéristiques de propriété

Il existe d'autres moyens pour lier des mécanismes utilisables pour définir des propriétés supplémentaires. Il est possible d'ajouter des caractéristiques de propriété, qui fournissent un mécanisme puissant pour améliorer le raisonnement lié à cette propriété.

Parmi les caractéristiques de propriétés principales, nous trouvons la transitivité, la symétrie, la fonctionnalité et l'inverse [25].

Propriété	Sémantique de la propriété	Exemple
owl:TransitiveProperty	Une propriété P est dite transitive si pour tout x, y, z, P(x,y) et P(y,z) implique P(x,z).	<owl:ObjectProperty rdf:ID="localisé-à"> <rdf:type rdf:resource="owl:TransitiveProperty"/> <rdfs:domain rdf:resource="owl:Thing" /> <rdfs:range rdf:resource="#Région" /> </owl:ObjectProperty> < Région rdf:ID="Massinissa"> <locaklisé rdf:resource="#Elkhourb" /> </Region> <Région rdf:ID="Elkhroub"> <locakisé-à rdf:resource="#Consatntine" /> </Region>
owl:SymmetricProperty	Une propriété P est dite symtrique si pour tout x, y P(x,y) implique P(y,x).	<owl:ObjectProperty rdf:ID="ami"> <rdf:type rdf:resource="owl:SymmetricProperty" /> <rdfs:domain rdf:resource="#Algérien" /> <rdfs:range rdf:resource="#Tunisien" /> </owl:ObjectProperty>
owl:FunctionalProperty	Si une propriété P est considérée fonctionnelle, alors pour tout x, y, z, P(x,y) et P(x,z) implique y = z.	<owl:ObjectProperty rdf:ID="nombre_courrier_envoyé "> <rdf:type rdf:resource="owl:FunctionalProperty"/> <rdfs:domain rdf:resource="#courrier" /> <rdfs:range rdf:resource="#Valeur" /> </owl:ObjectProperty>
owl:inverseOf	Une propriété P1 est dite inverse du P2 si pour tout x, y P1(x, y) équivalent P2(y, x)	</owl:ObjectProperty> <owl:ObjectProperty rdf:ID="envoyé-à"> <owl:inverseOf rdf:resource="# envoyé-par" /> </owl:ObjectProperty>
owl:InverseFunctionalProperty	Si une propriété P est considérée inverse fonctionnelle, alors pour tout x, y, z, P(y,x) et P(z,x) implique y = z	<owl:ObjectProperty rdf:ID=" nombre_courrier_envoyé " /> <owl:ObjectProperty rdf:ID=" envoyé-à "> <rdf:type rdf:resource= "&owl;InverseFunctionalProperty"/> <owl:inverseOf rdf:resource="# envoyé-par " /> </owl:ObjectProperty>

Tableau (II.1): Caractéristiques de propriété

2.2.5. Restrictions de propriété

Il est possible de contraindre de plusieurs façons l'image d'une propriété en fonction de contextes particuliers [25]. Nous le faisons avec des restrictions de propriété ci-dessous. Nous ne pouvons employer les diverses formes décrites que dans le contexte d'un élément *owl:Restriction*. L'élément *owl:onProperty* désigne la propriété restreinte.

Restruction de propriété	Explication	Exemple
owl: allValuesFrom	Exprimer une restriction universelle	<owl:Class rdf:about="#Cours_pg"> <rdfs:subClassOf rdf:resource="#Cours"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#enseigné-par"/> <owl:allValuesFrom rdf:resource="#Proffesseur"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
owl: someValuesFrom	Exprimer une restriction Existentielle	<owl:Class rdf:about="#Enseignant"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#Enseigne"/> <owl:someValuesFrom rdf:resource="#Cours_1ere_année"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
owl: minCardinality owl: maxCardinality owl: Cardinality	Permet d'exprimer exactement le nombre d'éléments dans une relation ou son intervalle.	<owl:Class rdf:about="#cours_pg"> <rdfs:subClassOf><owl:Restriction> <owl:onProperty rdf:resource="#Enseigné-par"/> <owl:minCardinality rdf:datatype="xsd:nonNegativeInteger">1 </owl:minCardinality> </owl:Restriction></rdfs:subClassOf> <rdfs:subClassOf><owl:Restriction> <owl:onProperty rdf:resource="#enseigné-par"/> <owl:maxCardinality rdf:datatype="xsd:nonNegativeInteger">2 </owl:maxCardinality> </owl:Restriction></rdfs:subClassOf></owl:Class>
owl: hasValue	permet de définir des classes selon l'existence de valeurs de propriétés particulières. elle impose une valeur spécifique à toutes les instances de la classe en question.	<owl:Class rdf:about="#Cours"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#enseigné-par"/> <owl:hasValue rdf:resource="#BOURBIA"/> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

Tableau (II.2) : Restrictions de propriété

2.2.6. Equivalence entre classes, propriétés et individus

Pour lier un ensemble d'ontologies dans une nouvelle ontologie, il se révèle souvent utile de pouvoir indiquer qu'une classe (ou une propriété) particulière dans une ontologie est équivalente à une classe (ou une propriété) dans une autre.

La propriété *owl:equivalentClass*[25][20] sert à indiquer que deux classes ont précisément les mêmes instances. Remarquez que dans OWL DL les classes représentent simplement des ensembles d'individus et qu'elles ne sont pas elles-mêmes

des individus. Par contre, dans OWL Full, nous pouvons employer la propriété *owl:sameAs* entre deux classes pour indiquer qu'elles sont identiques.
De la même façon, nous utilisons *owl:equivalentProperty* pour lier des propriétés.
Le mécanisme d'identité des individus est similaire à celui pour les classes, mais déclare deux individus identiques [25][20]. Voici un exemple :

```
<Cours rdf:ID="BD">
<owl:sameAs rdf:resource="#Base de données" />
</Cours>
```

Le contraire de la relation *sameAs* est la relation *differentFrom*.
Egalement, au contraire de la propriété *owl:equivalentClass*, nous pouvons définir le caractère *disjoint* d'un ensemble de classes au moyen du constructeur *owl:disjointWith*[25][20]. Il garantit que l'individu membre d'une classe ne puisse pas être simultanément l'instance d'une autre classe définie. Voici un exemple :

```
<owl:Class rdf:ID="Etudiant">
<rdfs:subClassOf rdf:resource="#Ressources Humaines"/>
<owl:disjointWith rdf:resource="#Enseignant"/>
<owl:disjointWith rdf:resource="#Administrateur"/>
</owl:Class>
```

Conclusion

Cette partie a donné une vue d'ensemble du langage d'ontologie du Web OWL en fournissant une brève introduction sur la nécessité d'un langage d'ontologie du Web. Il a aussi présenté les trois sous langages d'OWL à savoir OWL Lite, OWL DL et OWL Full avec une synthèse des caractéristiques de ces langages.