OWL et Terminae

Sylvie Szulman, Brigitte Biébow

OWL et Terminae*

Sylvie Szulman, Brigitte Biébow

Université de Paris-Nord, Laboratoire d'Informatique de Paris-Nord (LIPN)
Av. J.B. Clément, F-93430 Villetaneuse
ss@lipn.univ-paris13.fr et bb@lipn.univ-paris13.fr

Résumé : Cet article présente une réflexion sur l'importation et l'exportation d'une ontologie entre OWL et Terminae. Après une rapide présentation des deux langages, nous détaillons la traduction des différents éléments de OWL DL en Terminae. Nous proposons une solution pour intégrer dans OWL le lien avec les textes à partir desquels une ontologie a été créée en Terminae, ce qui est propre à Terminae.
Mots-clés : Ontologies, OWL, Web sémantique, modélisation à partir de textes

1 Introduction

Terminae est un outil d'aide à la construction d'ontologies à partir de textes (Szulman *et al.*, 2002) suivant la méthode décrite dans (Aussenac-Gilles *et al.*, 2000). Il permet au cogniticien de définir les concepts d'une ontologie à partir de l'étude terminologique d'un corpus du domaine. Il inclut à la fois des outils de visualisation de résultats d'outils de TAL (Lexter, Syntex (Bourigault & Fabre, 2000), Synoterm (Hamon, 2000), le concordancier Linguae) et un éditeur d'ontologies. Terminae conserve les liens de l'ontologie vers les textes ce qui offre une traçabilité totale et facilite la compréhension de l'ontologie. Nous ne nous intéressons pas ici à la méthode, mais au langage de l'ontologie, qui est traduisible dans un langage XML.
Actuellement, le langage OWL est devenu un standard pour la description d'ontologies. Nous nous proposons d'étudier comment adapter Terminae à ce standard tout en gardant le lien de l'ontologie avec le texte.
Nous ne définirons pas ici ce qu'est une ontologie et nous ne considèrerons pas d'autres langages du Web sémantique, le lecteur intéressé se réfèrera à (Charlet, 2002) (Baget *et al.*, 2003).

Nous présentons dans cet article comment une ontologie OWL peut actuellement être importée dans Terminae. En dehors de la réutilisation d'ontologies,

* Ce travail a été effectué dans le cadre du projet RNTL "Technolangue" ATONANT (convention 03.2.93.0639) auquel participent EADS, le CEA, le LIP6, le LaRIA, le LIPN et l'INSA de Rouen.

une telle importation permet d'effectuer une rétro-ingénierie vers des textes afin de mieux comprendre, commenter et exploiter une ontologie. Nous montrons les différences entre les deux langages, ce qui entraîne qu'importation et exportation ne sont pas inverses l'une de l'autre. Nous proposons également des solutions pour ajouter à OWL le lien de l'ontologie avec les textes et réaliser ainsi l'exportation d'ontologie construite avec Terminae. Nous nous posons enfin la question de savoir si Terminae doit s'adapter à OWL et à quel prix.

2 Présentation des langages

2.1 OWL

OWL est le langage de représentation d'ontologies élaboré par le W3C (url w3c,) et recommandé depuis fin 2003. Reposant sur les principes du Web Sémantique, XML, XMLSchema, RDF, RDFS, il intègre également les résultats de travaux en représentation des connaissances, comme les logiques de descriptions (DL), plus précisément OIL (url oil,), et DAML (url daml,). OWL se compose de trois sous langages d'expressivité croissante (OWL Lite, OWL DL, OWL Full).

Plusieurs documents décrivent OWL, dont le manuel de référence (1) (url owl reference,), le guide (2) (url owl guide,) montrant comment utiliser OWL pour construire une ontologie des vins, et le document (3) sur la sémantique et la syntaxe abstraite (url owl syntax,). Le croisement de ces documents est difficile. L'ordre de présentation des éléments et leur catégorisation est différent (en 1, constructeurs sur les classes, les propriétés, les individus...; en 2, constructeurs sur les classes simples et les individus, sur les propriétés simples, sur la fusion d'ontologies, sur les classes complexes...). Le vocabulaire varie (un "axiome" en (3) est généralement appelé "définition" ailleurs (cf.(1) p.7), les constructeurs sont soit propres à OWL soit ceux de RDFS avec parfois des différences subtiles. Par exemple, le rdfs:range qui restreint une propriété entraîne que toute valeur de la propriété se trouve dans la classe définie par le range, et ce indépendamment du domaine auquel s'applique la propriété; le owl:allValuesFrom restreint également le codomaine d'une propriété, mais il est utilisé dans une description de classe et cette restriction ne vaut que lorsque cette classe est le domaine de la propriété (cf. (1) p.22). On trouve dans (3) p.17 la notion d'EnumeratedClass pour définir une classe par la donnée de ses individus ainsi que le oneOf p.18, mais seul le oneOf est présent dans les autres documents. Les notions de concepts `défini` ou `primitif` qui sont essentielles en DL car à la base des algorithmes de classification sont présentés comme des modalités complete ou partial dans (3) mais n'apparaissent pas dans les autres documents. Ces remarques montrent la complexité d'OWL et expliquent difficultés de compréhension et d'utilisation. C'est assez banal pour des documents rédigés avec une visée normative par plusieurs rédacteurs, qui doivent intégrer des langages d'expressivité variée élaborés avec des philosophies différentes.

2.2 Terminae

Le langage de Terminae était à l'origine celui d'une LD ($\top$, $\bot$, $C \cap D$, $\forall r.C$, $\exists r$, $\{\geq nr, \leq nr\}$). Ces formalismes sont toujours approfondis dans notre communauté (Kassel, 2002) et repris pour OWL. Quoique n'offrant pas de classification, Terminae propose des algorithmes de calcul et de comparaison de formes normales pendant la construction de l'ontologie pour vérifier la validité de l'insertion ou de la modification d'un concept (par exemple, la correction de la valeur d'un rôle hérité) et pour rechercher l'existence de concepts similaires à celui décrit. L'objectif de la validation n'est pas d'effectuer des raisonnements mais d'aider le cogniticien lors de la représentation d'un sens d'un terme par un nouveau concept. Ainsi, Terminae propose des concepts déjà présents dans l'ontologie et similaires au concept décrit, c'est à dire ayant des propriétés «proches» (étiquette de rôle identique, valeur de rôles compatibles, hauteur similaire dans la hiérarchie), susceptibles de représenter le sens du terme. Le cogniticien doit décider si l'un de ces concepts représente bien le sens du terme, ou s'il est nécessaire de créer un nouveau concept.

La typologie de concepts
La modélisation des concepts issus de l'étude du corpus met en oeuvre une approche linguistique, et il est apparu nécessaire de distinguer les concepts selon la façon dont ils ont été créés. Le langage a été étendu par des labels. Transparents pour l'interprétation sémantique, ce sont des commentaires utilisés par le cogniticien lors de la construction de l'ontologie et de sa maintenance. Ils mettent en évidence un point de vue sur la modélisation du concept, suivant deux axes méthodologiques: une dimension de structuration, classique, et une dimension linguistique (Aussenac-Gilles *et al.*, 2000). Brièvement, la dimension de structuration peut être TDS pour structuration descendante, BUS pour structuration ascendante, RBUS pour concept de regroupement. La dimension linguistique décrit le lien entre le concept et le texte : T pour terminologique, c'est à dire concept construit à partir du sens d'un terme dans le corpus, NT pour non-terminologique.

2.3 Principaux constructeurs des deux langages

Nous considérerons par la suite uniquement les éléments de base d'OWL DL, sans les parties "headers" (commentaires sur l'ontologie comme auteur, date, sujet...), "mapping" (opérateurs de fusion d'ontologie) et "versioning" (pour gérer les versions d'une ontologie).
Une ontologie en OWL est composée de classes, de propriétés et d'individus; une ontologie en Terminae se compose de concepts et de rôles, génériques ou individuels.

Nous présentons dans le tableau 1 les principaux constructeurs d'OWL DL et de Terminae en regroupant ceux sur les classes puis ceux sur les propriétés. Nous détaillons la correspondance, parfois partielle, dans la section suivante.

Constructeurs OWL
Class
rdfs:subClassOf
intersectionOf
unionOf
complementOf
oneOf
disjointWith
rdfs:Property
rdfs:domain
rdfs:range
rdfs:subPropertyOf
inverseOf
transitiveProperty
symmetricProperty
allValuesFrom
someValuesFrom
maxCardinality
minCardinality
hasValue

Constructeurs Terminae
Concept
Concept_pere
complementaire_de
Concept_enumere
Role_generique
Concept_domaine
Concept_valeur
Restreint_valeur ou Restreint_cardinalité
Propriete_inverse
Propriete_transitive
Propriete_symétrique
cardinalite_maximum
cardinalite_minimum
instancie

TAB. 1 – *Constructeurs OWL et Terminae*

3 Importation d'une ontologie OWL en Terminae

Il existe plusieurs types de description d'une classe en OWL. Cependant, toute classe OWL est représentée par un concept Terminae, en lui donnant par défaut les valeurs TDS et NT pour les dimensions de structuration et linguistique.

3.1 Les classes simples

Une classe OWL correspond exactement à un concept générique en Terminae. La relation rdfs:subClassOf en OWL est traduite par la donnée du concept père en Terminae. La classe owl:Thing, sommet de la hiérarchie OWL correspond au concept générique `topConcept`. La relation à `topConcept` est explicite dans Terminae, pas en OWL. Une classe a une modalité 'complete'| 'partial', traduite en Terminae par une dimension définitionnelle `défini (D)` ou `primitif (ND)`.

```
<owl:Class rdf:ID="Winery"/>
```

```
<terminae:Concept>
  <terminae:Nom_concept> winery </terminae:Nom_concept>
  <terminae:Concept_pere> topConcept </terminae:Concept_pere>
  <terminae:Dimension_definitionnelle codeDef="ND" />
  <terminae:Dimension_linguistique codeLing = "NT" />
  <terminae:Dimension_structurelle codeStruct = "TDS" />
</terminae:Concept>
```

Une classe OWL n'est pas forcément nommée, mais un concept Terminae doit porter un nom qui est unique. Ainsi, une classe définie par restriction de propriété en OWL est une classe anonyme; elle se traduit par un concept dont le nom garde trace de l'anonymat de la classe et du nom de la propriété, et qui porte un rôle traduisant la propriété, comme ci-dessous `anonymMadeFromGrape`.

```
<owl:Class rdf:ID="Wine">
   <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
   <rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#madeFromGrape" />
    <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
      </owl:minCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
```

```
<terminae:Concept>
  <terminae:Nom_concept> wine </terminae:Nom_concept>
  <terminae:Concept_pere> potableLiquid</terminae:Concept_pere>
  <terminae:Concept_pere> anonymMadeFromGrape </terminae:Concept_pere>
</terminae:Concept>
```

```
<terminae:Concept>
  <terminae:Nom_concept> anonymMadeFromGrape </terminae:Nom_concept>
  <terminae:Concept_pere> anonymConcept </terminae:Concept_pere>
  <terminae:Role_generique>
    <terminae:Nom_role> madeFromGrape </terminae:Nom_role>
  <terminae:Concept_domaine> top-concept </terminae:Concept_domaine>
  <terminae:Concept_valeur> top-concept </terminae:Concept_valeur>
  <terminae:Restreint_min_cardinalite>1</terminae:Restreint_min_cardinalite>
  </terminae:Role_generique>  </terminae:Concept>
```

3.2 Les individus

En OWL, un individu appartient à au moins une classe et peut avoir un identifiant. Un concept individuel en Terminae a obligatoirement un nom l'identifiant.

```
<Region rdf:ID="CentralCoastRegion"/>
```

```
<terminae:Concept_individuel>
  <terminae:Nom_concept>centralCoastRegion</terminae:Nom_concept>
  <terminae:Concept_pere>region</terminae:Concept_pere>
</terminae:Concept_individuel>
```

3.3 Les classes complexes

Une classe complexe OWL est définie soit comme une expression comportant des opérateurs ensemblistes (union, intersection, complémentaire), soit par l'énumération de ses individus, soit par une disjonction. Terminae n'offre pas la

possibilité de décrire une classe comme une expression comportant des opérateurs. Certains opérateurs OWL sont traduisibles en Terminae, d'autres pas ou du moins la sémantique n'est pas exactement la même. En ce cas, l'utilisateur de Terminae est prévenu lors de l'importation.

Classe définie par une expression ensembliste :

* intersectionOf : une classe intersection de classes est traduite par un concept générique fils de chaque concept traduisant une classe de l'intersection.

* unionOf : une classe réunion de plusieurs classes est traduite par un concept qui subsume tous les concepts présents dans l'union.

```
<owl:Class rdf:ID="Fruit">
   <owl:unionOf rdf:parsetype="Collection"/>
  <owl:Class rdf:about="#SweetFruit"/>
  <owl:Class rdf:about="#NonSweetFruit"/>
   </unionOf>
</owl:Class>
```

```
<terminae:Concept>
  <terminae:Nom_concept>sweetFruit</terminae:Nom_concept>
  <terminae:Concept_pere>fruit</terminae:Concept_pere>
</terminae:Concept>
<terminae:Concept>
  <terminae:Nom_concept>nonSweetFruit</terminae:Nom_concept>
  <terminae:Concept_pere>fruit</terminae:Concept_pere>
</terminae:Concept>
```

Cette traduction est partielle car rien ne garantit que toute instance de `fruit` soit une instance de `sweetfruit` ou de `nonSweetFruit` ou des deux.

* complementOf: le constructeur complementOf est traduit par le constructeur `complementaire_de` qui a été ajouté à Terminae et équivaut à la négation.

Les classes disjointes : L'opérateur disjointWith exprime la disjonction entre classes, c'est à dire qu'un individu n'appartient qu'à une seule des classes disjointes. La disjonction entre deux classes est traduite par la création du complémentaire d'une des deux classes, la seconde étant sous-classe de ce complémentaire.

```
<owl:Class rdf:ID="SweetFruit">
      <rdfs:subClassOf rdf:resource="edibleThing" />
</owl:Class>
<owl:Class rdf:ID="Vegetable">
       <rdfs:subClassOf rdf:resource="edibleThing" />
       <owl:disjointWith rdf:resource="#SweetFruit" />
</owl:Class>
```

```
<terminae:Concept>
  <terminae:Nom_concept>sweetFruit</terminae:Nom_concept>
  <terminae:Concept_pere>edibleThing </terminae:Concept_pere>
</terminae:Concept>
```

```
<terminae:Concept>
  <terminae:Nom_concept>nonSweetFruit</terminae:Nom_concept>
  <terminae:Concept_pere>edibleThing</terminae:Concept_pere>
<terminae:complementaire\_de>sweetFruit</terminae:complementaire\_de>
</terminae:Concept>
```

```
<terminae:Concept>
  <terminae:Nom_concept>vegetable</terminae:Nom_concept>
  <terminae:Concept_pere>nonSweetFruit </terminae:Concept_pere>
</terminae:Concept>
```

Cette traduction entraîne qu'une instance de `edibleThing` est forcément instance de `SweetFruit` ou de `NonSweetFruit`, ce qui est plus contraignant que la disjonction.

Les classes énumérées:
L'opérateur OWL oneOf permet de définir une classe par extension; il est équivalent au constructeur Terminae `Concept_enumere`.

3.4 Les propriétés

En OWL, une propriété est une relation binaire, qui peut n'être définie que par son nom. OWL s'appuie sur rdfs pour décrire les propriétés.
Une propriété OWL sera traduite par un rôle Terminae. Un rôle doit obligatoirement avoir un domaine et un codomaine, ainsi qu'une cardinalité minimum et une cardinalité maximum. La notion de sous-rôle existe grâce à la restriction de rôle, qui porte sur le codomaine ou la cardinalité. Toutefois, Terminae n'offre pas la possibilité de définir une hiérarchie de rôles indépendante de celle des concepts. C'est une différence majeure avec OWL, dont les propriétés forment une hiérarchie en utilisant les constructeurs super ou owl:subPropertyOf, sans contrainte sur les classes domaine et codomaine.
OWL distingue deux types de propriétés selon le type du codomaine :
les ObjectProperty dont le codomaine est une classe. Elles se traduisent exactement par un rôle générique entre concepts représentant les domain et range.

```
<owl:ObjectProperty rdf:ID="hasWineDescriptor">
  <domain rdf:resource="#Wine"/>
  <range rdf:resource="#WineDescriptor"/>
</owl:ObjectProperty>
```

```
<terminae:Role_generique>
  <terminae:Nom_role>hasWineDescriptor</terminae:Nom_role>
  <terminae:Concept_domaine>wine</terminae:Concept_domaine>
  <terminae:Concept_valeur>wineDescriptor</terminae:Concept_valeur>
  <terminae:Cardinalite_minimum> 0</terminae:Cardinalite_minimum>
  <terminae:Cardinalite_maximum>100</terminae:Cardinalite_maximum>
</terminae:Role_generique>
```

* les DatatypeProperty dont le codomaine est un literal RDF ou un type prédé-

fini de XML Schema. Terminae n'accèdant pas aux types de données du langage d'implémentation (java), il faut créer à chaque fois les concepts correspondants, sous-concepts d'un concept de regroupement `typeXMLSchema`. Une DatatypeProperty est donc traduite par un rôle générique du nom de la DatatypeProperty entre le concept représentant la classe domain et le concept correspondant au type XML Schema.

L'instanciation d'une propriété correspond exactement à l'instanciation d'un rôle générique par un rôle individuel, qui doit respecter les restrictions de cardinalité et de codomaine. En OWL, la cardinalité d'une propriété est par défaut [0,∞], elle peut être précisée en utilisant l'élément owl:cardinality ce qui correspond à la cardinalité d'un rôle en Terminae.

En OWL, les propriétés peuvent avoir des caractéristiques (FunctionalProperty, SymmetricProperty, TransitiveProperty, inverseOf). Les trois dernières existent également sur les rôles (`Propriete_symetrique`, `Propriete_transitive`, `Propriete_inverse`).

Une propriété en OWL peut être décrite comme restriction d'une autre. La restriction de valeur owl:allValuesFrom définit une contrainte sur le codomaine (classe ou donnée) d'une propriété, sans imposer l'existence de la propriété pour un individu. C'est l'équivalent du quantificateur universel : la valeur de la propriété pour un individu devra instancier cette classe ou cette donnée. En Terminae, cela correspond exactement à la restriction du codomaine d'un rôle;lors de la restriction du rôle, l'éditeur de Terminae vérifie immédiatement qu'elle est compatible avec le codomaine défini initialement.

La restriction de valeur owl:someValuesFrom définit une contrainte sur le codomaine (classe ou donnée) d'une propriété, en imposant l'existence d'au moins une instanciation de cette propriété pour un individu. C'est l'équivalent du quantificateur existentiel. Elle est traduite en Terminae par une double restriction du rôle, sur le codomaine et sur la cardinalité qui sera au minimum de 1. Cela n'empêche pas l'existence d'un rôle ayant même nom, même domaine et un codomaine différent.

La restriction owl:hasValue indique la valeur (individu ou donnée) d'une instanciation de la propriété. Elle correspond à la notion de rôle individuel en Terminae défini sur un concept générique.

Cependant, comme nous le disions en début de cette section, toutes les relations entre propriétés ne sont pas exprimables en Terminae :

```
<owl:ObjectProperty rdf:ID="hascolor">
  <subPropertyOf rdf:resource="#hasWineDescriptor"/>
  <range rdf:resource="#WineColor"/>
</owl:ObjectProperty>
```

Dans cet exemple, la classe wineColor est une sous-classe de WineDescriptor. Une relation hiérarchique est décrite entre les propriétés hasWineDescriptor et hascolor. Il n'y a pas de traduction satisfaisante de cette relation. Les propriétés sont traduites par des rôles génériques. La propriété hasWineDescriptor a été traduite précédemment par un rôle `hasWineDescriptor` dont le domaine est `wine` et le codomaine `WineDescriptor`. La propriété hasColor est traduite par un rôle `hasColor` dont le codomaine est color mais le domaine n'est pas précisé, il doit cependant être $\subseteq$ `wine`. On définira le concept `wine` comme domaine, mais sans indiquer la hiérarchie entre les rôles. En effet, deux rôles ayant même domaine en Terminae ne peuvent être en relation hiérarchique.

L'exemple met en évidence que Terminae privilégie la hiérarchie des concepts pour la modélisation. Cela vient de l'approche linguistique qui commence par l'étude des termes et des concepts terminologiques en découlant. Les rôles servent à décrire les concepts et ne sont pas définis a priori, ce qui pourrait cependant être le cas si on partait des relations présentes dans le texte. Notre expérience nous a conduites vers un type particulier de modélisation auquel nous avons adapté l'outil, et réciproquement : l'outil finit aussi par biaiser la modélisation.

3.5 Conclusion

L'importation d'une ontologie OWL se fait donc avec une perte partielle d'information. Les éléments non traduits ou traduits partiellement sont indiqués en commentaire lors de l'importation. Le plus gros problème reste la différence d'expressivité sur les propriétés. Toutefois, il est possible de récupérer des ontologies OWL existantes. L'expérience montre (par exemple, pour l'ontologie sur les vins de (url owl guide,)) que la majeure partie des ontologies existantes est importable en Terminae. Outre leur réutilisation pour créer nos propres ontologies, nous pouvons les relier à des textes portant sur le thème de l'ontologie. Cette sorte de rétro-ingénierie est en cours sur l'ontologie Wine et un glossaire sur le vin, ce qui permet d'améliorer la compréhensibilité de l'ontologie. Nous expliquons dans la section suivante la liaison entre ontologie et corpus.

4 Ajout à OWL pour exportation

L'exportation de Terminae en OWL ne pose pas de problème d'un point de vue DL, car OWL est plus expressif que Terminae: les opérateurs DL de Terminae ont un équivalent immédiat en OWL. Cependant, le principal intérêt de Terminae consiste à lier un corpus décrivant un domaine et une ontologie sur ce domaine. Le lien ontologie-texte se fait par la relation concepts-termes. Un concept terminologique de l'ontologie correspond à un sens d'un ou plusieurs termes présents dans un texte, il est associé à tous ces termes dans l'ontologie (synonymes). L'ensemble des occurrences de ces termes dans le texte est conservé avec leur contexte. On peut donc à partir d'un concept terminologique retrouver

tous les termes associés et leurs occurrences dans le texte. Ces informations sont acquises à l'aide d'outils de TAL, en particulier Syntex et Synoterm.

Pour conserver ces informations dans OWL, nous proposons d'utiliser les annotations. Le corpus séquencé contenant les références des occurrences de terme, serait disponible sur le Web et son adresse fournie dans l'en-tête de l'ontologie. Les identifiants de séquence des termes associés à un concept terminologique seraient eux-aussi décrits dans une URI, dont l'adresse serait donnée dans une annotation `occurrence` liée au concept. L'annotation `synonym_list` donnerait la liste des termes associés au concept, et l'annotation `term` le terme désignant préférentiellement le concept.

Exemple:

```
<owl:Class rdf:ID="Aroma">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />
   <owl:annotation>
     <terminae:term> aroma </terminae:term>
     <terminae:synonym_list>
       <terminae:synonym> bouquet</terminae: synonym>
     </terminae:synonym_list>
     <terminae:occurrence rdf:http://lipn.
      univ-paris13.fr/~Szulman/ontoWine#aroma/>
  </owl:annotation>
</owl:Class>
```

On trouverait les occurrences suivantes, extraites du glossaire sur le vin :
Aroma: Traditionally defined as the smell that wine acquires from the grapes and from fermentation. Now it more commonly means the wine's total smell, including changes that resulted from oak aging or that occurred in the bottle– good or bad. "Bouquet" has a similar meaning.
Dumb: Describes a phase young wines undergo when their flavors and aromas are undeveloped.
...

Pour que ces informations linguistiques soient prises en compte lors d'une importation d'ontologie Terminae, il faudrait étendre les éditeurs OWL existants. Cela permettrait toutefois de préserver la traçabilité entre ontologies, termes et textes comme l'offre Terminae ou le système basé sur XTerm et TROEPS décrit dans (Cerbah & Euzenat, 2000).

5 Etat de l'art

Terminae implémente une méthodologie de construction d'ontologie à partir de textes émanant du groupe TIA (url tia,), proche de celle mise en oeuvre dans l'éditeur DOE (Troncy & Isaac, 2002). DOE repose sur des principes linguistiques (la sémantique différentielle) qui permettent au concepteur d'exprimer la signification linguistique de concepts dans une ontologie différentielle où les concepts

sont définis par leurs différences et ressemblances. Ces concepts sont repris dans une ontologie référentielle qui correspond à la notion classique d'ontologie de domaine, mais respectant la hiérarchie de l'ontologie différentielle (Bachimont, 2000). Pour le moment, la méthode part de textes du domaine pour construire les ontologies, prend en compte les résultats d'outils de TAL mais ne conserve pas le lien avec le texte. Le langage de DOE étant insuffisamment expressif, l'éditeur propose différents formats d'exportation de l'ontologie référentielle, dont OWL. L'ontologie peut être complétée dans un éditeur comme Protégé ou OiLed.

L'éditeur d'ontologie Protégé (url protege,) développé à Stanford offre un *plugin* permettant l'importation et l'exportation d'ontologies OWL, en traduisant OWL en Protégé. Les deux langages ne sont pas équivalents comme le montre (url protege mapping,), mais tous les éléments d'OWL sont présents. Aucune méthodologie spécifique n'est associée, et les raisonneurs sont externes. On pourrait étendre le *plugin* OWL en intégrant des informations textuelles liées à un concept comme dans Terminae. Il manquerait toutefois la partie d'intégration des résultats de TAL sur lesquels repose la méthode de modélisation.

6 Conclusion

Nous avons comparé OWL DL et Terminae, en vue d'importer et exporter des ontologies. Nous avons montré la possibilité d'importer partiellement une ontologie OWL; l'un de nos objectifs est la rétro-ingénierie, c'est à dire trouver des textes portant sur le domaine et en lier les termes aux concepts afin de mieux comprendre l'ontologie, ce qui est nécessaire pour envisager sa réutilisation.

L'approche de modélisation en OWL est différente de celle de Terminae, ce qui apparaît dans l'existence de certains constructeurs difficiles à traduire. Par exemple, il n'a jamais été indispensable dans nos expériences de modélisation de créer une propriété en dehors d'un concept, ou un concept anonyme, qu'il soit générique ou individuel. Pour nous, le nom d'un concept ou d'un rôle porte la trace de sa signification. Bien sûr, notre méthode de modélisation ne s'applique pas à toute application pour laquelle est développée une ontologie, même si l'accès aux textes du Web offre un large champ de travail.
L'importation d'OWL présentée dans cet article est réalisée, l'exportation est en cours. Cependant, cette étude nous a montré que les extensions apportées au langage de Terminae ne permettaient pas une importation complète et réversible. Si nous désirons vraiment que Terminae puisse bénéficier des avancées sur le Web sémantique et y contribuer en retour, nous devons nous orienter vers une autre solution. Nous pouvons conserver l'éditeur Terminae bien adapté à la modélisation à partir de textes, mais en remplaçant le langage par OWL en utilisant une des API OWL disponibles. Nous pouvons également remplacer dans la plate-forme de modélisation l'éditeur Terminae par un éditeur OWL. Les deux solutions ont un coût non négligeable que nous étudions.

Remerciements Nous remercions nos relecteurs pour leurs remarques pertinentes et leurs corrections qui nous ont permis d'avancer dans notre réflexion.

Références

Aussenac-Gilles N., Biébow B. & Szulman S. (2000). Revisiting ontology design : a methodology based on corpus analysis. In R. Dieng & O. Corby, Eds., *Knowledge Engineering and Knowledge Management : Methods, Models, and Tools. Proc. of the 12th International Conference, (EKAW'2000)*, LNAI 1937, p. 172–188: Springer-Verlag.

Bachimont B. (2000). Engagement sémantique et engagement ontologiqueă: conception et réalisation d'ontologies en ingénierie des connaissances. In J. Charlet, M. Zacklad, G. Kassel & D. Bourigault, Eds., *Ingénierie des Connaissances, évolutions récentes et nouveaux défis*, p. 305–323, Paris: Eyrolles.

Baget J., Canaud E., Euzenat J. & Saïd-Hacid M. (2003). Les langages du web sémantique. In J. Charlet, P. Laublet & C. Reynaud, Eds., *Action spécifique CNRS-STIC "Web sémantique" , Rapport final, CNRS, Paris (FR)*, p. 9–24.

Bourigault D. & Fabre C. (2000). Approche linguistique pour l'analyse syntaxique de corpus. *Cahiers de Grammaire , Numéro spécial "sémantique et corpus"*, **25**, 131–151.

Cerbah F. & Euzenat J. (2000). Integrating textual knowledge and formal knowledge for improving traceability. In R. Dieng & O. Corby, Eds., *Knowledge Engineering and Knowledge Management : Methods, Models, and Tools. Proc. of the 12th International Conference, (EKAW'2000)*, LNAI 1937, p. 296–303: Springer-Verlag.

Charlet J. (2002). *L'Ingénierie des connaissances Développements, résultats et perspectives pour la gestion des connaissances médicales.* Habilitation à diriger des recherches, Université Pierre et Marie Curie.

Hamon T. (2000). *Vérification sémantique en corpus spécialisé : acquisition de relations de synonymie à partir de ressources lexicales.* Thèse d'informatique, Université Paris 13, Villetaneuse, France.

Kassel G. (2002). Une méthode de spécification semi-informelle d'ontologies. In *Actes des 13ièmes journées francophones d'Ingénierie des Connaissances (IC 2002)*, p. 75–87.

Szulman S., Biébow B. & Aussenac-Gilles N. (2002). Structuration de terminologies à l'aide d'outils de tal avec Terminae. In A. Nazarenko & T. Hamon, Eds., *Traitement automatique des langues. Structuration de terminologie*, volume 43, p. 103–128: Hermes.

Troncy R. & Isaac A. (2002). Doe: une mise en oeuvre d'une méthode de structuration différentielle pour les ontologies. In *Actes des 13ièmes journées francophones d'Ingénierie des Connaissances (IC 2002)*, p. 229–238.

url daml. http://www.daml.org/.

url oil. http://www.ontoknowledge.org/oil/.

url owl guide. http://www.w3.org/tr/2004/rec-owl-guide-20040210/.

url owl reference. http://www.w3.org/tr/2004/rec-owl-ref-20040210/.

url owl syntax. http://www.w3.org/tr/2004/rec-owl-semantics-20040210/.

url protege. http://protege.stanford.edu/.

url protege mapping. http://protege.stanford.edu/plugins/owl/mapping.html.

url tia. http://www.biomath.jussieu.fr/tia.

url w3c. http://www.w3.org/2001/sw/webont/.