Querying multiple sources with OWL ontologies: an exploratory study in an automotive company

Pierre Mariot[1], Jean-Pierre Cotton[1], Christine Golbreich[2], Alain Berger[1], François Vexler[1]

[1] Ardans, 2 rue Hélène Boucher, 78286 Guyancourt, France
pmariot@ardans.fr

[2] University of Versailles Saint-Quentin, 55 Av. des Etats-Unis , 78035 Versailles, France
Christine.Golbreich@uvsq.fr

Abstract. Large companies often store information and knowledge in multiple information systems using various models and formats. A main stake is to retrieve the relevant knowledge for a specific concern. We present an exploratory study for a large automotive company faced to this problem. A first mock-up has been built using OWL ontologies and KAON2. Different questions arose during its development: how to organize the ontologies? how to communicate with knowledge sources? how to support the user for query formulation? which languages and techniques to use? This paper presents the provisory solutions adopted and possible alternatives considered.

1. Introduction

Large manufacturers are storing information and knowledge in multiple information systems using many formats. These systems are most often dedicated to specific activities and are not designed to communicate. Hence, users usually access i only one or two sources of knowledge during their work and ignore the existence of other possibly relevant sources. This prevents them from having the possibility to enlarge their view and to take into account other constraints or other solutions issued from other departments of the same company. To tackle the increasing complexity of new products design and manufacturing, large companies need to share information and knowledge through their multiple departments. This paper relates an exploratory study investigating this problem for an important automotive company. It describes the mock-up built to that end. The proposed approach is based on several ontologies designed to facilitate accessing and querying the different available knowledge sources. With a single query, the user gets the different answers issued from the different sources (I, J, K, Fig. 1).

2. Architecture

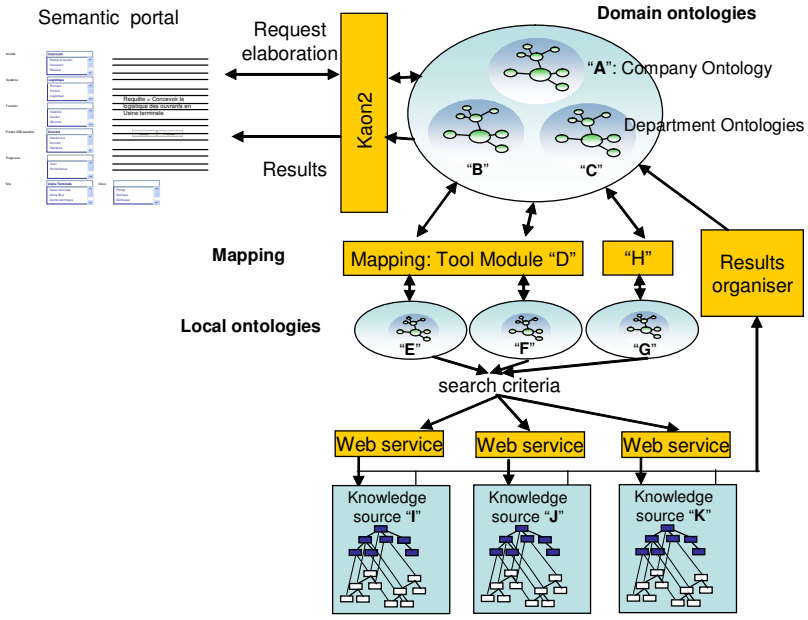


Fig. 1: Global architecture

The global architecture consists of several components: (i) existing knowledge sources, (ii) local ontologies, (iii) domain ontologies (global ontology), and (iv) mappings between global and local ontologies.

Knowledge Sources. In this experiment there were three knowledge sources to be queried: a source dedicated to Logistics (I Fig. 1), a source dedicated to Air-Conditioning Design (J), a source dedicated to Opening Design (K). They were developed independently, using different tools: I and J were created with Ardans Knowledge Maker tool [16], K with Lotus Notes.

Domain ontologies. We have built several "domain" ontologies: the *company ontology* is a general ontology transverse to the whole company, *department ontologies* are specific to each department. The company ontology (A Fig. 1) models common representations shared by all the services, such as the standard structure of the product, the standard functional analysis of the product, the standard processes. These standard representations are used by many different applications. The department ontologies (B; C) model common representations of a department, for instance: ontology of the Car Design Department (B), ontology of the Logistic Department (C). These ontologies are independent of any application.

Local ontologies e.g., E, F, G (Fig. 1) include all concepts that can be used to query the sources. For example, E includes all concepts available to query the source I, e.g., `RechercheParVue`. These concepts are associated to class instances of the tool module D which is imported by E (see below).

Mappings. Mappings modules have been defined to map the domain ontologies concepts to concepts enabling to query the sources. A so called "tool" mapping module, e.g., D, H (Fig. 1) models the different types of search that can be achieved by using a given tool to access a knowledge source. For example, D models the searches that are possible with the Ardans Knowledge Maker tool, e.g., SearchByView or FullTextSearch, to access the knowledge source I. Its main property is `aCommeRecherche` (i.e. `hasSearchMode`) which links domain concepts to their corresponding tool concepts.

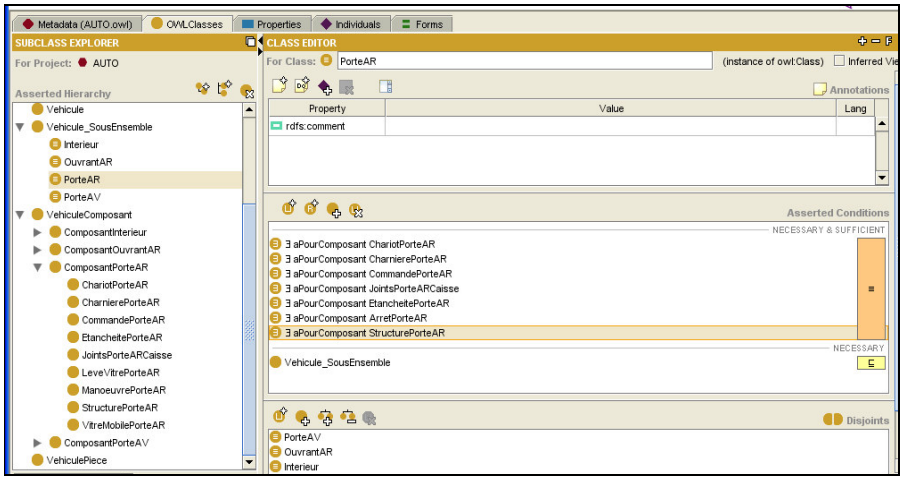


Fig. 2: Company ontology[1] (A)

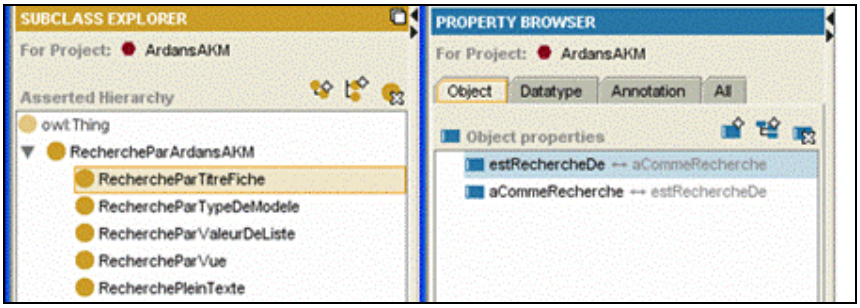


Fig. 3: Tool module (D)

The class `RechercheParTitreFiche` (SearchByFormTitle) has a property estRecherchede (whose inverse is aRecherchede – `hasSearchMode`)

[1] The OWL ontologies were developed using the Protégé OWL editor [14]

3. Mapping process

For example, assume that a user is looking for knowledge about Cost (in French "Coûts") for a Front Door Structure (Structure Porte AV).

How does the mapping work for "Front Door Structure"?
The skeleton of a car is a structure made of steel. Components of the car, e.g.; glass, seals, seats, are assembled on the structure. The class `Front Door Structure` represents the steel component of the door. The Company ontology A (Fig.2) contains.

- `Vehicle`
- `Engine`
- `Rear Door Component`
    - `Rear Door Glass`
    - `Rear Door Seals`
    - `Rear Door Structure`
    - …
- `Front Door Component`
    - `Front Door Glass`
    - `Front Door Seals`
    - `Front Door Structure`

…

Logistics department deals only with Structure components and Assembly Components. Structure components correspond to the structure of the car which is first stamped, then painted, and assembled with Assembly components issued from external providers. Logistics department ontology (C) contains a representation of a car from the logistic specific point of view:

- `Vehicle`
    - `Structure component`
    - `Assembly component`

Design department for Opening has also its own representation of a vehicle (B):

- `Openings`
    - `Front Door or Rear Door`
    - `Openings Glass`
    - `Openings Seals`...

Therefore, a query about `Front Door Structure` has to be mapped to `Structure component` from a Logistic point of view, and to `Front Door or Rear Door`, from the Opening Design point of view. This is achieved by asserting Necessary and Sufficient conditions using existential restrictions in the ontology C, which imports the ontology A.

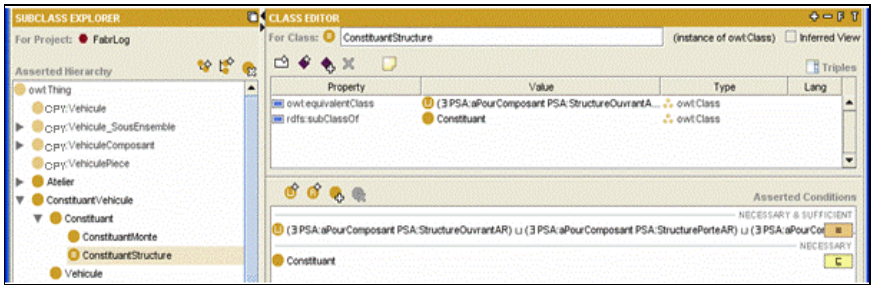


Fig. 4: Logistics Department ontology (C)

Running an OWL reasoner, e.g., RACER [5], PELLET [6], FACT++ [7], automatically computes the inferred hierarchies, for example, the following hierarchies are inferred for the Logistic and the Opening design ontology respectively:

- `Vehicle`
  - `Structure component`
    - `Rear Door Structure`
    - `Front Door Structure`
- `Assembly component`

- `Openings`
  - `Front Door or Rear Door`
    - `Rear Door Structure`
    - `Front Door Structure`
  - `Openings Glass`, etc.

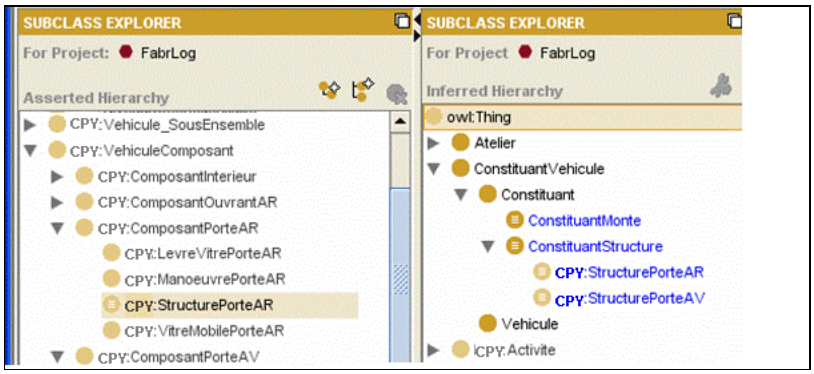


Fig. 5: Asserted and inferred hierarchy of the Logistic Department ontology

Hence, the system knows that `Front Door Structure` is a subclass of `Structure component` in ontology C and a subclass of `Front Door` or `Rear Door` in ontology B, and the system has to generate a query for both sources I and K. For example, let be a query that needs to access the knowledge source I dedicated to theknow-how about Logistics. Usually, to get such knowledge about "Structure

component", a user has to clic on a "View" (a kind of category used to index and serach the knowledge stored in the application) called "Structure". Therefore, on order to simulate the same, our integration system has to map the class "Structure component" of ontology C to a query corresponding to a clic on the View "Structure" of the source I. The class `StructureComponent` of the ontology C contains a single instance `StructureComponent_1`. Ontology E contains an instance `SearchByViewStructure_1` of the class `SearchByView` belonging to module D. In ontology E, the instance `searchByViewStructure_1` is related by the property `isSearchModeOf` to the instance `StructureComponent_1` of ontology C. Instance `searchByViewStructure_1` of ontology E has properties like `AKM:identifiant(ID)` which allows to access the source I content.

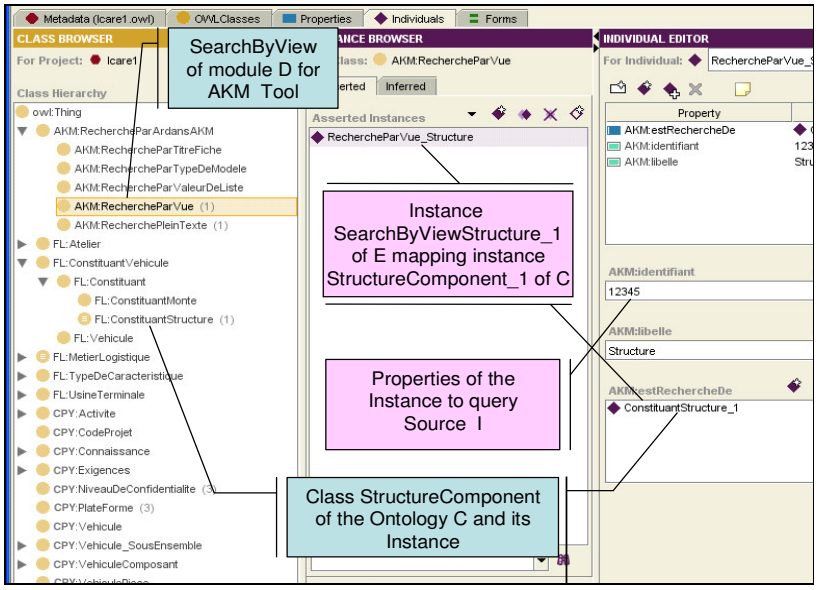


Fig. 6: Global ontology importing all the ontologies and modules

How does the mapping work for "Costs"?

"Cost" is an unknown concept of both knowledge sources. It is defined in the Company ontology as a kind of Requirement (in French "Exigence"). The mapping has to find out a way to query the Logistic knowledge source with the "Cost" concept. This point was solved using the local ontology defined for the Logistics source. The concept `Cost` is linked to an instance of the `FullTextSearch` class. This instance has a property related to the Full Text Search query. In our case: `Chiffrage` or `Coût Logistique` (logistic cost). So the query `Cost` is translated for the Logistics source into a full text search with the query "`Chiffrage OR coût logistique`". This illustrates that when a local ontology is designed for a Knowledge source, the knowledge engineer has to identify implicit concepts, and to define how to translate them into the relevant Full text queries.

2.3 Query processing

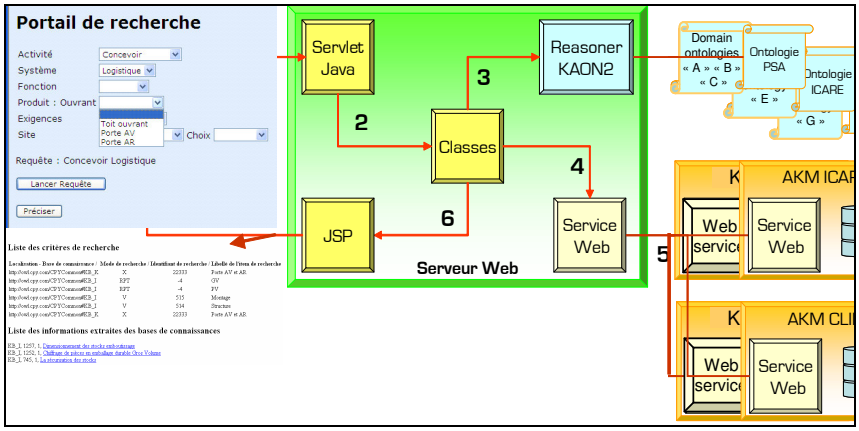


Fig. 7: Technical architecture

1 The user selects relevant domain classes (search criteria) through the search portal Fig. 7 (in French "Portail de recherché").
2 The query is transmitted to the processing server (Fig. 7)
3 The processing server uses domain ontologies to identify the one or the several knowledge bases concerned by the search criteria
4 Servlet JAVA extracts query parameters and create an instance of the processing class which generates SPARQL [13] queries.
5 SPARQL queries are processed through KAON2 reasoner [15] and provides relevant Knowledge bases with their search criteria
6 Web services are processed on the relevant source of knowledges
7 Results are gathered in an HTML page and sent to the user (lower left corner Fig. 7). This HTML page is made of the initial question criteria with the corresponding results.Characteristics (title, URL) of relevant knowledge elements

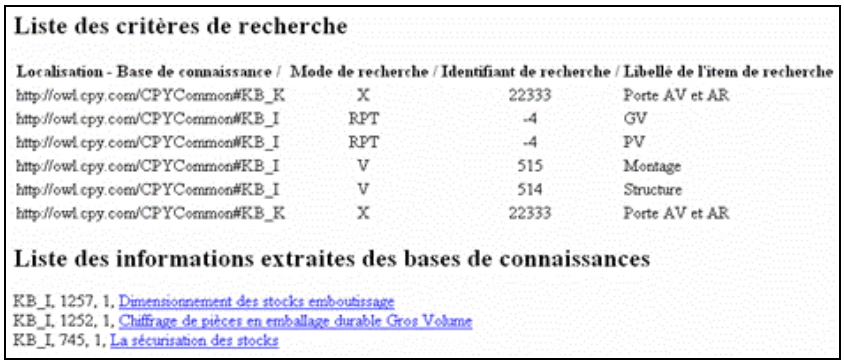
Example 1 of query:
Find all search modes for RearOpening, and for each one give its identiant and the knowledge source to be accessed

```
SELECT ?z ?d ?s
WHERE {RearOpening_1 a:hasSearchMode ?z ?z a:identifier ?d ?z
a:hasSource ?s } ORDER BY ?z
```

Example 2 of a more complex query:

```
SELECT ?x ?y ?z ?d ?c ?e ?s WHERE
{RearOpening a:estComposantDe ?x . /Select classes whose
RearOpening is component of/
?x a:aPourComposant ?y . /For each select its components/
?y a:aPourRecherche ?z . /Retrieve corresponding search concept/
?z a:libelle ?c . /Retrieve corresponding label/
?z a:code ?d . /Retrieve corresponding identifier/
?z a:typeRecherche ?e . /Retrieve corresponding search mode/
```

```
?z a:aPourApplication ?s }/Retrieve corresponding knowledge
source/
ORDER BY ?z
```



Liste des critères de recherche

Localisation - Base de connaissance /	Mode de recherche /	Identifiant de recherche /	Libellé de l'item de recherche
http://owl.cpy.com/CPYCommon#KB_K	X	22333	Porte AV et AR
http://owl.cpy.com/CPYCommon#KB_I	RPT	-4	GV
http://owl.cpy.com/CPYCommon#KB_I	RPT	-4	PV
http://owl.cpy.com/CPYCommon#KB_I	V	515	Montage
http://owl.cpy.com/CPYCommon#KB_I	V	514	Structure
http://owl.cpy.com/CPYCommon#KB_K	X	22333	Porte AV et AR

Liste des informations extraites des bases de connaissances

KB_I, 1257, 1, Dimensionnement des stocks emboutissage
KB_I, 1252, 1, Chiffrage de pièces en emballage durable Gros Volume
KB_I, 745, 1, La sécurisation des stocks

Fig. 8: Answer to example 2 query and information retrieved in sources I, K

Figure 10 (top) shows the results of the query example 2: knowledge source, search mode, identifier, label of search. The answer is derived from the following facts of the Abox: `RearOpening isComponentOf Opening` (asserted in ontology A); `Opening hasComponent Porte AV et AR` defined as labels (in ontology B); `RearOpening isComponentOf Constituant` (in ontology C); `Constituant hasComponent Structure and Montage` defined as labels (in ontology C). Figure 10 (bottom) shows the list of retrieved information next extracted from the mentionned knowledge sources, thanks to the above results.

3. Choices, success and limits

The mock-up architecture that has been defined is quite classical for heterogeneous information integration, and partly similar to the architecture presented in [8]. A main difference is that the sources store non structured information that can be accessed only by specific tools, e.g., Ardans Knowledge Maker, and not structured data of databases. We successfully designed and implemented a mock-up using OWL [1] and KAON2 [15] for knowledge sources integration in a large automotive company. Several work-around were needed. The adopted solutions are provisory and improvements may be considered for any future development.

3.1 Query formulation

The user is guided by menus issued from the domain ontologies. Other solutions are possible: to allow natural language questions interpreted thanks to domain ontologies; to propose a refinement of the query driven by real data of the applications and not only by the domain ontologies.

3.2 Languages

SPARQL query language. Several query languages do exist: SPARQL [13], OWLQL, NRQL etc. We selected SPARQL. KAON2 provides for research purpose a Java library to trigger SPARQL queries. However, SPARQL has a strong limitation, reasoning is possible only with instances. Because of this limitation we were obliged to define "prototypical" instances instead of using classes in the local ontologies. For future development, query language extensions allowing reasoning directly with classes would be useful.
OWL 1.1 [11]. We used OWL DL as the ontology language so as to use KAON2. Obviously, using OWL 1.1 qualified cardinalities (and other extensions) will be useful, allowing for example to express that a car has exactly two rear doors.

3.3 Mapping representation

We used OWL expressiveness to define the mappings by subclass and equivalence axioms. However, it may be useful to specify more complex mappings. As first evaluations using Hoolet [12] showed rather satisfying performance with the rule language extension SWRL, SWRL may be a good candidate for implementing more powerful translators. We had initially considered to implement the mappings in another language, e.g., XSL, but this option was cancelled. We prefer a coherent framework where modeling is expressed using a single language that allows for representing an ontology enriched by rules, e.g., SWRL [10] or a DL-safe rule extension of OWL [4]. As far as we know, without restrictions imposed on the form of queries and mappings, query rewriting is undecidable. A more careful analysis of the requirements is needed in the future to know what should be preferred, SWRL undecidable extension or DL-safe rules as supported by KAON2.

4. Conclusion

A friendly framework allowing to query distributed knowledge is an important needs. This mock-up has successfully demonstrated the usefulness of OWL and KAON2 reasoner for knowledge integration in a large automotive company. Ontologies are crucial components to access knowledge spread over multiple knowledge sources. But the use of OWL and KAON2 for this purpose in industrial companies is still an emergent technique. We mainly relied on the online documentation [2; 9]. Though we have more than twenty years of practice in knowledge engineering, it turned out that it was not obvious to understand and use OWL and protypical tools such as KAON2, from the presently available information without some support. This mock-up would never have existed without the assistance of an "OWL specialist". In conclusion, OWL ontology seems a promising technique. But its success in industrial companies will critically depend on the industrialization and standardization of the languages and tools (editors and reasoners), coupled with the availability of reliable documentation, knowledge engineering methodology, training, and tutorials.

References

1 Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: The making of a Web Ontology Language. J. of Web Semantics, 1(1):7–26, 2003

2 OWL (2004), OWL Web Ontology Language : Overview, http://www.w3.org/tr/owl-features/.

3 OWL 1.1 (2006). OWL 1.1 Web Ontology Language Syntax: http://www-db.research.belllabs.com/user/pfps/owl/syntax.html.

4 Motik, B., U. Sattler, et R. Studer (2004). Query answering for OWL DL with rules. *ISWC 2004, LNCS 3298 Springer*. Publishers, Inc.

5 Volker Haarslev and Ralf Moller. RACER system description. In Proc. of the Int.joint Conf. on Automated Reasoning (IJCAR 2001), volume 2083 of Lecture Notes in Artificial Intelligence, pages 701–705. Springer, 2001.

6 Evren Sirin and Bijan Parsia. Pellet: An OWL DL reasoner. In Proc. of the 2004 Description Logic Workshop (DL 2004), 2004.

7 Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), volume 4130 of Lecture Notes in Artificial Intelligence, pages 292–297. Springer, 2006.

8 Haase, P., Hitzler, P., Krötzsch, M., Angele, J., Studer, R. : Practical Reasoning with OWL and DL-Safe Rules. Half-day tutorial at the 3rd European Semantic Web Conference, ESWC 2006. http://km.aifb.uni-karlsruhe.de/ws/prowl2006/

9 Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe Matthew Horridge A Practical Guide To Building OWL Ontologies Using The Prot´eg´e-OWL Plugin and CO-ODE Tools Edition 1.0, 2004.

10 Ian Horrocks, Peter F. Patel-Schneider, Sean Bechhofer, Dmitry Tsarkov. OWL Rules: A Proposal and Prototype Implementation. Journal of Web Semantics, Vol. 3, No. 1, pp 23-40, 2005.

11 OWL 1.1 Peter Patel-Schneider and Ian Horrocks. OWL 1.1 Web Ontology Language overview. W3C Member Submission, 19 December 2006. Available at http: //www.w3.org/Submission/owl11-overview/

12 Hoolet: http://owl.man.ac.uk/hoolet/

13 SPARQL: http://www.w3.org/TR/rdf-sparql-query/

14 Protégé OWL: http://protege.stanford.edu.

15 KAON2: http://kaon2.semanticweb.org/

16 Pierre Mariot, Christine Golbreich, Jean-Pierre Cotton, François Vexler, Alain Berger Méthode, Modèle et Outil Ardans de capitalisation des connaissances, 2007, 7èmes journées francophones Extraction et Gestion des Connaissances, Namur, Belgique http://www.ardans.com/