Using Ontologies for Extracting Differences in the Dynamic Domain: Application on Cancer Disease

Nora Taleb
Laboratory for Electronic Document Management LABGED, Badji Mokhtar University, Algeria

Abstract: *Over time, the data representatives a given domain can change, both the data model reflecting the area. In this situation, the presence of strategies that can summarize the produced changes is mandatory. This study presents an implemented approach based on data mining techniques in order to extract the differences, the model is domain ontology and the changes are represented by two ontology's versions. The results are summarized in changes report. An experimentation of the tool was made on the ontology of the cancer disease and satisfactory results were obtained.*

Keywords: *Ontology change, ontology versioning, web ontology language scheme, retrieval information.*

Received February 27, 2013; accepted September19, 2013; published online March 8, 2015

1. Introduction

The analysis of changes between ontologies is an important service for ontology engineering [16] it presents a very critical task in information retrieval. As one of important management of ontology versioning, detecting changes provide information about differences between versions of ontologies. Generally, the differences between ontology's versions are caused by changes in the domain itself [2٢] they can arise from different situations:

- The increase in the number of ontologies specifying the same field.
- The need to establish correspondences, to trace the evolution of ontology through the comparison of the various versions [29].
- The generation of mediators for queries.
- Several versions can result, when a distributed ontology is developed in an independent way [8].

An ontology resulting from the application of a change can be considered as a new version as shown in Figure 1.

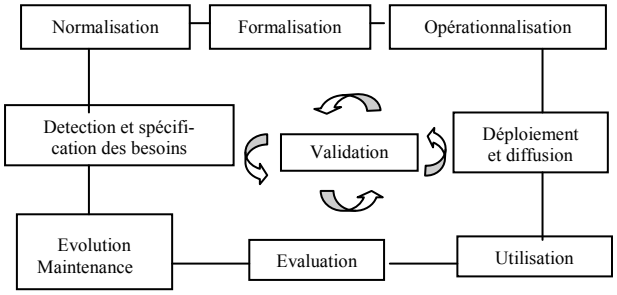

Figure 1. Ontology development process [2٥].

The analysis task must be carried out in a completely automatic way as it is not realistic to perform the mapping of ontologies by hand, a fortiori when these ontologies exceed a certain size or complexity. Most analysis studies compare file based ontologies to find differences between them [3]. However, version comparison by comparing text files is an approach which does not work [2]. In order to, cope with the complex problem of ontology's versions differences, several related research disciplines have emerged (such us: Ontology alignment, margins, mapping, etc.,), each dealing with a different facet of the problem [3]. The current state of the art in ontology engineering ignores logically changes and lacks any further characterization of even significant changes. The problem of computing the difference between pairs of ontologies has been approached both syntactically and semantically.

The ideal would be to preserve the various versions of ontology and to keep all the information concerning the differences and compatibilities between them. This requires methods of versions identification of versions differentiation (based on the same principles as the methods of measurement of semantic similarities in ontology alignment) and of specification of relations between versions; procedures of update of ontology; and of the mechanisms of access to the various versions of ontology [3].

Two major aspects of ontology differences can be distinguish: The detection of changes and the presentation of changes to the user.

In this paper, we propose a framework both logical and probabilistic allowing the automatic comparison of ontologies. The method extract the differences between ontology's versions both syntactic by using the characteristics of Web Ontology Language (OWL) scheme [30] and semantic by using some similarity measures in order to detect the distance between the entities of ontology versions (concepts, relations). In particular, we present a software tool using the

semantics of the definitions and axioms of OWL in order to establish the journal of changes between two ontological versions.

The remainder of this paper is structured as follows: We start with a short discussion of related works section 2. We continue with the presentation of OWL language in section 3. Then, we present our proposed approach in detail in section 4 with two correspondent algorithms. We introduce the software tool used for extraction differences in section 5. Before the conclusion, we evaluate the performance of our software tool by using two versions of the same ontology for cancer disease, *V*1 includes information concerning inflammatory cancer and *V*2 for Non inflammatory in section 6 and finally we conclude in section 7.

2. Related Works

2.1. Tools to Manage Ontology Changes

Within the framework of management of change suggested for ontologies, several specialized prototypes have been developed [9, 10, 13, 17]. Noy *et al*. [٢١, ٢٤] presents a web-based tool the onto view tool implements a procedure of detection of changes for ontologies in Resource Description Framework (RDF) [٢٠]. Its principle consists in observing rules in order to discover specific operations of change and to produce sets of transformation between the versions of ontology [12] it compares ontologies on the data model not the representation to detect changes. By grouping RDF-triples per definition, the necessary representational knowledge can be retained. Djedidi *et al.* [6, 7, 8] have developed Two extensions of the PROMPTdiff tool-one plug-in developed for protected to research mappings between frames while basing itself on heuristics [٢٤] were proposed in [٢٤] their role is to define the relations of evolution between the elements of two ontologies versions. The user interface makes it possible to visualize certain complex changes between versions of ontology.

A more complete system of ontology evolution is described in [15, ٢٦]. The core of the system is based on the CHAO ontology of changes and annotations (Exchange and Ontology Annotation). The instances of the CHAO ontology represent the changes between two versions of the ontology and the annotations users related to these changes. The system is implemented in the form of two protected plug-in:

- Plug-in management of change giving the access to a list of changes and allowing the users to add individual or grouped annotations of change and to consult the history of the concepts.
- PROMPT Plug-in providing comparisons between two versions of an ontology and information on the users having made these changes and facilitating the acceptance or the rejection of the changes [14, 31].

These methods perform comparison based on file-storage ontologies. When ontologies continue to evolve, There is Much redundancy among all versions of ontologies. It is not space efficient [**2**٦].

2.2. Classification of the Changes

The objective of classification changes is to define, for a given language of representation of ontologies, taxonomy of changes specifying of the classes of changes and their properties. The principal classifications defined in the literature were proposed for languages KAON2 [4, 5, 7] and OWL3 [14]. The KAON ontology of changes classifies the changes according to three levels of abstraction [16]:

- *Elementary Changes*: Applying modifications to only one entity of ontology.
- *Composite Changes*: Applying modifications in the direct neighborhood of an entity of ontology.
- *Complex Changes*: Applying modifications to an arbitrary set of entities of ontology.

Moreover, the KAON changes were also classified according to their effect [6]:

- *Additive Changes*: Adding new entities to ontology without deteriorating the existing entities.
- *Subtractive Changes*: Removing certain entities or parts of entities. Thought like a minimal and complete unit, the ontology of changes KAON does not take into account the modifications of entities. Klein and Fensel [14] differentiates from the basic operations of changes and complex changes:

1. The basic changes are simple and atomic changes which can be specified while being based only on the structure of ontology and which modify only one characteristic of the model of knowledge OWL [28] i.e., only one entity of the ontology (like an addition operation of a class or a relation suppression “is-a”).
2. The complex changes correspond to composite and rich changes grouping logical sequences of basic changes and incorporating information on their impact on the logical model of ontology (as for example moving up subclasses, widening the same domain of a property object to its super classes, amalgamating classes, etc.,). In addition to, their specification, complexity also appears in the effects of these changes. If the effects of the basic changes remain relatively minor, the cumulated effects of all the intermediate changes carrying out change complexes can be important.

2.3. Methods of Ontological Comparison

There are four methods of comparison of ontologies [19]:

1. *Comparison of the Internal Structures*: Compare the internal structure of the entities (e.g., value interval, cardinality of attributes, etc.,).

2. *Comparison of the External Structures*: Compare the relations entities with others. It is composed of methods of comparison of the entities within their taxonomies and methods of comparison of the external structures by taking the cycles into account.
3. *Comparison of the Instances*: Compare the extensions of the entities, i.e., it compares the set of the other entities which are attached to them (concepts of ontology).
4. *Semantic Method*: Compare interpretations (or more exactly the Models) of the entities.

3. Web Ontology Language

The OWL is a standard OWL proposed by W3C recently. OWL is intended to be used by applications to represent terms and their interrelationships. It is an extension of RDF and goes beyond its semantics. OWL provides a richer set of vocabulary by further restricting on the set of triples that can be represented. An OWL document can include an optional ontology header and any number of class, property and individual descriptions or axioms [2٧]. A named class in an OWL ontology can be described by a class identifier, for example, "<owl: Class rdf:ID="Cancer_disease"/>" defines a class "Cancer_disease" which is an instance of "owl: Class". There are two kinds of properties can be defined in OWL: Object property (owl: ObjectProperty) which links individuals to individuals and data type property (OWL: DatatypeProperty) which links individuals to data values.

The semantics of OWL is defined in the way analogous to the semantics of description logic [12, 2٢] as shown in Figure 2.

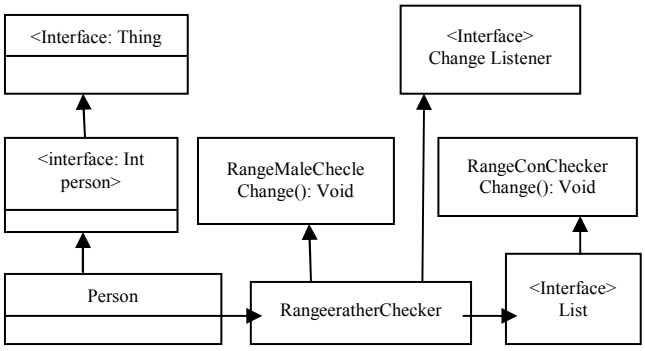

Figure 2. Multiple range OWL properties in Java [2٣].

4. Proposed Approach for the Comparison between Versions of Ontologies

We propose CompOnto method for extracting the differences between the versions of ontology, it is divided into two stages: The first concerns a syntax extraction, based on the formalism of OWL syntax, and the second is using a measure of similarity which calculates the distance between two ontological entities.

Before starting, We must present our definitions for: Ontology, version of ontology, and how a change can be represented as shown in Figure 3.

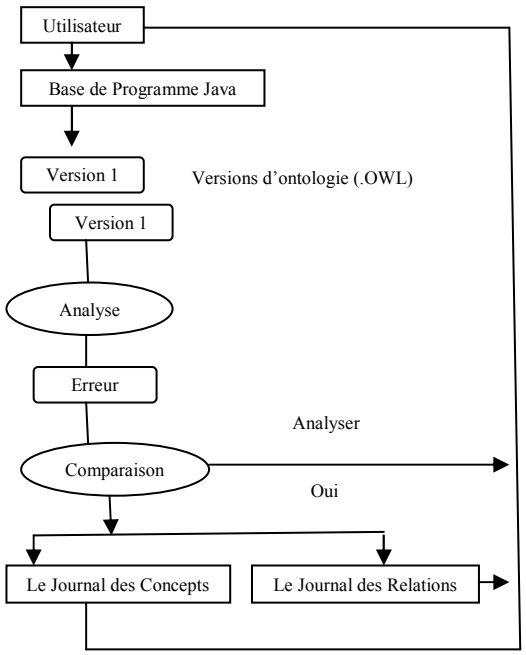

Figure 3. Fonctionnel architecture of compdiff method.

- ***Definition* 1**. In OWL ontology, concepts are arranged in hierarchical structure called classes and relate each other with properties and axioms. The structure of an OWL ontology is represented by a tuple $S:=\{C, R, <, X\}$

  Where C, R: Are separated sets containing the concepts and no taxonomic relationships. <: CXC is a partial order on C, which defines the concept hierarchy. X: R CXC is the signature of a taxonomic relationships.

  The lexicon of the ontology is the tuple $L:=\{Lc, Lr, F, G\}$ where Lc, lr: Are separated disjoint sets. F, G: Are two references relationships.

  The hierarchy of concepts is defined by a structure: $S0:=\{C, <\}$

The A concept is defined by $L0:=\{Lc, F\}$.

- ***Definition* 2**. One version of an ontology is a four-tuple, $= < CV\ I, PV\ I, AV\ I, IVi >$. Each has the same meaning as defined in Definition 1.

  In our approach, we extend Definition 1 in which version numbers are applied to every ontology components.

- ***Definition* 3**. Given an ontology O and two arbitrary versions $V1$ and $V2$.

  The comparison of two versions ($V1$ and $V2$) of ontology consists in identifying the various types of change: Consequently, the various types of comparisons being able to be drawn are as follows:

  1. Exact Comparison: $V1 \cap V2$ this comparison consists in finding the concepts (or the relations)

with the same name which exists in the two versions of the ontologies. We name this case "the stability".

2. Comparison Different: $V1 \neq V2$ this comparison consists in finding the concepts (or relations) which exist in the first version and do not exist in the second one. It can results from adding or deleting concepts (relations) from the first version $V1$.
3. Approximate Comparison: This comparison is more interesting because it consists in finding synonymous terms for concept names or concepts with similar names. The method is based on hybridization of the algorithms of fusion with a dictionary of the synonyms.
4. Similar Comparison: The similar comparison is a manual comparison; the user selects the names of concepts which he considers similar. We are based on the syntax of OWL scheme in order to compare versions of ontology.

In order to, calculate the distance between the two versions of ontology (expressed by owl files), we use a string metric, which is a metric that measures similarity or dissimilarity (distance), between two text strings for approximate string matching or comparison and in fuzzy string searching.

The most widely known string metric is Levenshtein Distance. It operates between two input strings, returning a score equivalent to the number of substitutions and deletions needed, in order to, transform one input string into another.

The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion or substitution of a single character. Mathematically, the Levenshtein distance between two strings *a* and *b* is given by *lev a*, *b* (|*a*|, |*b*|) where:

$$Lev_{a,b}(i,j) = \begin{cases} Max(i,j) \\ Min \begin{cases} Lev_{a,b}(i-1,j)+1 \\ Lev_{a,b}(i,j-1)+1 \\ Lev_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{cases} \end{cases} \quad (1)$$

Based on Levenshtein edit distance we propose a *syntactic similarity measure* for strings, in order to, compare the two OWL files (which represent the ontology's versions). As follows is Distance Algorithm based on Levenshtein edit distance for extracting differences between ontologies.

Algorithm 1: Distance.

```
Input
V1, V2 : :={C, R, <, X} // OWL files
int n=V1.length(); // length of Version V1
int m=V2.length(); // length of Version V2
char t_j; // j^th character of t
int I, j, cost;
Output
if (n==0) return m
  else if (m==0) return n;
   int p[]=new int[n+1]; //'previous' cost array, horizontally
   int d[]=new int[n+1]; // cost array, horizontally
   int _d[]; //placeholder to assist in swapping p and d
   for (i=0; i<=n; i++) p[i]=i;
    for (j=1; j<=m; j++)  t_j = t.charAt(j-1);
        d[0]=j;
   for (i=1; i<=n; i++) {
      a.  cost=s.charAt(i-1)==t_j ? 0 : 1;
      b.  // minimum of cell to the left+1, to the top+1,
          diagonally left and up+cost
      c.  d[i]=Math.min(Math.min(d[i-1]+1, p[i]+1) and nbsp;
          p[i-1]+cost);
   End
   // copy current distance counts to 'previous row' distance
  counts
   _d = p; p = d; d = _d;
   // our last action in the above loop was to switch d and p,
  so p now
   // actually has the most recent cost counts
    return p[n];
    End distance.
```

5. Un Algorithm for the Management of the Ontological Comparison

Our project of ontology versions comparison makes it possible to compare two versions selected by the user. As follows is the implemented algorithm.

5.1. Definition of the Algorithm

Figure 3 illustrates the COMP Algorithm; it takes two ontologies as input in the form of OWL files as follows:

- Traverse the two versions of ontology.
- For each version, extract the concepts and the relations.
- Compare the two versions.
- Post the comparison result in the form of a journal changes (concepts, relations) the following architecture explains in detail the tool: The system is a tool made up of several complementary components where each one has a precise task to achieve. Among the major functionalities which these components offer:

1. Analysis of two input versions.
2. Localization of the differences between these two versions through the posting of the journal of the changes of concepts (or relations) for each one.

The COMP algorithm is composed of two principal modules: Analyze and comparison.

5.2. Analysis

To analyze the two versions of the program written in OWL, we benefited from the capacities of Java; in extracting the concepts and the relations of these two

versions, it returns as result the Journal of the changes for these concepts and relations for each version.

5.3. Comparison

The entities of the two versions which we want to compare are the concepts and the relations. To compare these entities, we distinguish the following classes:

- ***Class*** 1: This class is the main class. It is responsible for the initialization of COMP; it manages the graphic interface and the interaction with the user. It offers a window with various menus specific to precise functionalities (to extract the concepts (relations) of an OWL file², to analyze the program, comparison of two ontology versions).
- ***Class*** 2: This class is regarded as one of the most important classes in COMP. It provides the means and the structures necessary to extract the concepts (relations) of ontology which exist in the form of an OWL file and visualizes them in a Journal of changes as shown in Figure 4.
- ***Class*** 3: This class deals with the detection of the differences (to make the comparison itself).

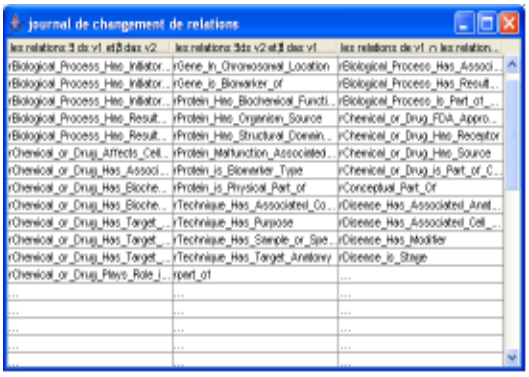

Figure 4. The results of comparision algorithm.

6. Evaluation

6.1. Evaluation Environment

Our application was developed in Java; it can be integrated in other resource supporting the Java virtual machine. For the implementation of our ontology, we chose Protégé.3.4; several reasons justified our choice: Protégé-3.4. Source is a free open editor, it allows to import and to export ontologies in various implementation languages (RDF-Schema, OWL, DAML, OIL, ..., etc.,), it has a modular interface, which allows its enrichment by additional modules (plug-in), Protégé-3.4.1 allows the edition and the visualization of ontologies. Finally, it is provided with API written in Java, which makes it possible to develop applications being able to access Protégé anthologies and to handle them.

Our ontology is implemented in OWL. However, OWL files are not easily exploitable in their rough form because of their complex structure. In order to, be able to exploit it we needed a "translator" able to translate the mark-up tags and the semantics conveyed by OWL files into objects easy to handle by programs. For this purpose, we used the JENA API [11].

In order to, build the instance of a Java class, we use the standard Java-beans [8] approach to access the values of the properties of the class (*set/ get methods*). In order to, maintain class relationships present in the ontology (including multiple inheritance), we use Java interfaces to define the OWL classes.

6.2. Evaluation Results

To evaluate the efficiency of our proposed ontology comparison, we run the evaluated ontology comparison tools on two versions of cancer disease ontology [18] as shown in Figure 5. The result of the algorithm is the Journal of the changes. The first column contains the concepts which exist in the version1 and does not exist in the second version. The second column contains the concepts which exist in the version and does not exist in the first version. The third column contains the intersection of the concepts between the two versions of ontologies (common concepts). The first column presents the relations which exist in the version1 and does not exist in the second version. The second column shows the relations which exist in the version and does not exist in the first version. The third column shows the intersection of the relations between the two versions of ontology as shown in Figure 6.

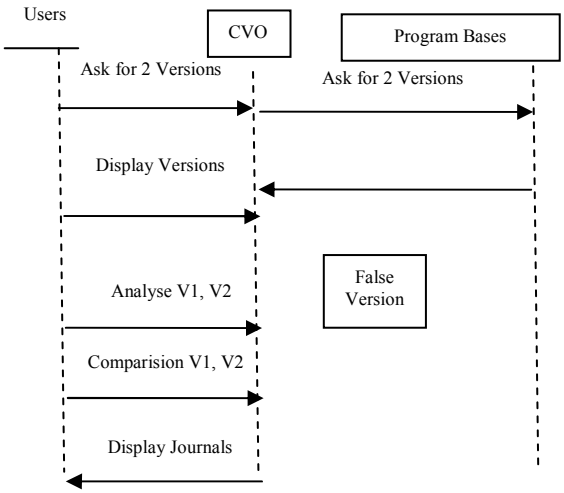

Figure 5. Sequence diagram.

Figure 6. The results of comparision algorithm 2.

7. Conclusions

This paper is a refinement and extension of the ontology technology in the medical field; it presents an implemented approach for detecting changes between two versions of ontology. We described how to obtain a sketch of our comparison algorithm. The evaluation of our prototypic implementation gives promising numbers, which outrun the results from existing approaches.

An experimentation of the tool was made on the ontology of the cancer disease and satisfactory results were obtained. We aim to use the tool to make the comparison in more complicated fields such as sociology in order to be able to find the differences between two companies and to trace the common points and the points in disjunction between them.

We hope that more research continues in this direction in order to realize practical widely used applications based on semantic web technologies.

References

[1] Almeida R. and Guizzardi G., "Knowledge Engineering an Ontological Approach to Domain Engineering," *in Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, pp. 351-358, 2002.

[2] Benslimane S., Malki M., and Rahmouni M., "Towards Ontology Extraction from Data," *the International Arab Journal of Information Technology*, vol. 5, no. 1, pp. 34-44, 2008.

[3] Charlet J., Bachimont B., and Troncy R., "Ontologies Pour Le Web Sémantique. Le Web Sémantique," available at: http://www.eurecom.fr/~troncy/Publications/Troncy-revue_i304.pdf, last visited 2005.

[4] Di Pinto F., Lembo D., Lenzerini M., Mancini R., Poggi A., Rosati R., Ruzzi M., and Savo F., "Optimizing Query Rewriting in Ontology-Based Data Access," *in Proceedings of the 16th International Conference on Extending Database Technology*, pp. 561-572, 2013.

[5] Ding Z., "A Probabilistic Extension to Ontology Language OWL," *in Proceedings of the 37th International Conference on System Sciences*, Hawaii, pp. 1-10, 2004.

[6] Djedidi R. and Aufaure A., "Change Management Patterns (CMP) for Ontology Evolution Process," available at : http://ceur-ws.org/Vol-519/djedidi.pdf, last visited 2009.

[7] Djedidi R. and Aufaure A., "Ontology Change Management," *in: A. Paschke, H. Weigand, W. Behrendt, K. Tochtermann, Pellegrini (Eds.),* I-Semantics 2009, Proceedings of IKNOW '09 and I-SEMANTICS, 611-621, Verlag der Technischen Universitt Graz. 2009.

[8] Doan A., Halevy Y., and Ives G., Principles of Data Integration, *Morgan Kaufmann*, 2012.

[9] Fethallah H. and Mohammed C., "Automated Retrieval of Semantic Web Services: A Matching Based on Conceptual Indexation," *the International Arab Journal of Information Technology* , vol. 10, no. 1, pp. 61-66, 2013.

[10] Ghobadi A. and Rahgozar M., "An Ontology-Based Semantic Extraction Approach for B2C Ecommerce," *the International Arab Journal of Information Technology*, vol. 8, no. 2, pp. 163-156, 2011.

[11] Groner G., Parreiras S., and Staab S., "Semantic Recognition of Ontology Refactoring," *in Proceedings of the 9th International Semantic Web Conference*, Shanghai, pp. 273-288, 2010.

[12] Kalyanpur A., Pastor J., Battle S., and Padget J., "Automatic Mapping of OWL Ontologies into Java," *in Proceedings of the 16th International Conference on Software Engineering and Knowledge Engineering*, 2004.

[13] Klein B., Cosmides L., Tooby J., and Chance S., "Decisions and the Evolution of Memory: Multiple Systems, Multiple Functions," *Psychological Review*, vol. 109, no. 2, pp. 306-329, 2002.

[14] Klein M. and Fensel D., "Ontology Versioning on the Semantic Web, " *International Semantic Web Working Symposium*, USA, pp. 75-91, 2001.

[15] Klein M. and Noy N., "A Component-Based Framework for Ontology Evolution," available at : http://se-pubs.dbs.uni-leipzig.de/files/Klein2003Acomponentbasedframeworkfor.pdf, last visited 2001.

[16] Klein M., "Change Management for Distributed Ontologies," *PhD Thesis*, *Vrije Amesterdam University*, 2004.

[17] Kondylakis H., Plexousakis D., and Tzitzikas Y., "Ontology Evolution in Data Integration," available at: http://ceur-ws.org/Vol-651/paper4.pdf, last visited 2010.

[18] Kun L., Shengqun T., Zhang L., and Tian H., "A Method Based on RDB for Detecting Changes Between Multi-version Ontologies," *Journal of Computational Information Systems*, vol. 8, no. 8, pp. 45-52, 2012.

[19] Maedche A. and Staab A., "Measuring Similarity between Ontologies," *in Proceedings of the 13th European Conference on Knowledge Acquisition and Management*, Madrid, pp. 251-263, 2002.

[20] Nguyen P., Kaneiwa K., and Nguyen M., "Ontology Inferencing Rules and Operations in Conceptual Structure Theory, available at: http://krr.meraka.org.za/~aow2010/Nguyen-etal.pdf, 2010.

[21] Noy N. and Kelin M., "Ontology Evolution: Not the Same as Schema Evolution," *the Knowledge and Information Systems*, vol. 6, no. 4, pp. 428-440, 2004.

[22] Noy N. and Musen M., "Ontology Versionning as an Element of an Ontology Management Framework," *IEEE Intelligent System*, 2003.

[23] Noy N. and Musen M., "Promptdiff: A Fixed Point Algorithm for Comparing Ontology Versions," *in Proceedings of the 8th National Conference on Artificial Intelligence*, Menlo Park, CA, pp. 744-750, 2002.

[24] Noy N., Chugh A., Liu W., and Musen M., "Framework for Ontology Evolution in Collaborative Environments," *in Proceedings of the 5th International Semantic web Conference*, *Lecture notes in computer science*, USA, pp. 544-558, 2006.

[25] Noy N., Kunnatur S., Klein M., and Musen A., "Tracking Changes During Ontology Evolution," *in Proceedings of the 3rd International Semantic Web Conference*, Hiroshima, pp. 259-273, 2004.

[26] Noy N., Kunnatur S., Klein M., Mark A., and Musen., "Tracking Changes During Ontology Evolution," *in Proceedings of the 3rd International Semantic Web Conference*, pp. 259-273, 2004.

[27] OWL Web Ontology Language Semantics and Abstract Syntax, available at: http://www.w3.org/TR/owl-semantics/, last visited 2004.

[28] Rafael S., Parsia B., and Sattler U., "Categorising Logical Differences Between OWL Ontologies," *in Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, New York, pp. 1541-1546, 2011.

[29] Stojanovic L., "Methods and Tools for Ontology Evolution," *Thesis of the Karlsruhe University*, Germany, 2004.

[30] Suntisrivaraporn B., "Modularization-based Approach to Finding All Justifications for OWL DL Entailments," *in Proceedings of the 3rd ASWC*, Bangkok, pp. 1-15, 2008.

[31] Wiggisser K. and Eder J., "Detecting Changed in Ontologies via DAG Comparison," *in Proceedings of the 19th International Conference*, Trondheim, pp. 21-35, 2007.

Nora Taleb was Engineer in computer science in 1996. She received a MS degree in artificial intelligence: l'acquisition des connaissances pour la construction d'un système à base de connaissance" in 2003. She received a PhD degree in knowledge acquisition for managing an evolving ontology. Currently, she is searcher and lecture at the Department of Computer Science in Annaba University (Algeria). She participated at several internationals conferences: ISPS (International symposium on programming and systems, Algeria, 2003), NACET (North African conference of enginnering technology, Algeria, 2003), TIA'07 (Terminology and artificial intelligence, Toulouse, 2007), ICOSE'09 (International conference on ontology and semantic engineering, Italy, 2009), and I-semantics (International conference on semantic systems, Austria, 2009). She is member of the project of Management of the knowledge, the ontologies and the system of decision-making: Application in the industrial maintenance. Her research interests are: Ontology, knowledge acquisition, and expert systems.