International Journal of Computer Applications Technology and Research
Volume 6–Issue 2, 109-116, 2017, ISSN:-2319–8656

Towards Ontology Lifecycle: Building, Matching and Evolution to Semantically Integrate Application Ontologies

Razika Driouche
National High College
of Biotechnology, Taoufik
Khaznadar, Algeria

Abstract: Semantic interoperability among applications, systems, and services are mostly based on ontology. Its increase usage in Information Systems and knowledge sharing systems raises the importance of ontology development and maintenance. It is essential for sharing information among independent organizations, exchange of information among heterogeneous systems. To make this possible, we need to carefully model the domain knowledge while preserving its semantics. Ontologies are complex in nature and often structured. Their development and maintenance incorporate research areas like: building, evolution, versioning, matching and integration where these are fundamentally different. We uncover the gap in the current research area of ontology building, matching and evolution. We propose a research direction based on ontology construction using knowledge extraction, matching evolution between versions. This paper presents system architecture to manage the lifecycle of the application ontology incorporating building, matching and evolution processes. This solution is integrated in the source ontology since its creation in order to make it possible to evolve and to be versioned.

Keywords: ontology lifecycle; ontology building; ontology matching; ontology evolution; application ontology.

1. INTRODUCTION

With growing business globalization and worldwide collaboration of manufacturing companies, a seamless exchange of products, services and information, within and across enterprises is urgently required. Both vendors and users are making serious efforts to improve enterprise interoperation [1].

Modern organizations are increasingly operating upon distributed and heterogeneous information systems, as they continuously build new autonomous systems, powered by the rapid advancement of information technology. They are facing challenges to integrate heterogeneous applications. The need to integrate heterogeneous applications, both within and across organizations, is indeed becoming pervasive.

Every day, organizations all over the world generate reports, articles, books, emails, and all kind of textual data concerning several topics. The increase of the storage capacity of computers and servers enable these organizations to keep all files. They produce without the need of deleting anything. One mainly problem they face is to know what kind of information they have, and how it is related.

The fundamental aspect of information exchange among applications, systems, and services is the development of a consistent and comprehensive model for representing the domain knowledge [2]. It is essential for sharing information among independent organizations, and exchange information among heterogeneous applications of Information Systems. To make this possible, we need to model the domain knowledge while preserving its semantics [3]. The development of ontologies is becoming a crucial part of semantic web and knowledge management in the organizations.

Interoperability among different ontologies becomes essential to gain from the power of the Semantic Web. Thus, matching of ontologies becomes a core question.

Ontology matching is a key interoperability enabler for the semantic web, as well as a useful tactic in integration tasks dealing with the semantic heterogeneity problem. It takes the ontologies as input and determines as output a set of correspondences between the semantically related entities of those ontologies.

Ontology matching is seen as a solution provider in today's landscape of ontology research. As the number of ontologies that are made publicly available and accessible on the Web increases steadily, so does the need for applications to use them. A single ontology is no longer enough to support the tasks envisaged by a distributed environment like the Semantic Web. Multiple ontologies need to be accessed from several applications. Matching could provide a common layer from which several ontologies could be accessed and hence could exchange information in semantically sound manners [4].

Thus the use of ontology is increasing in Information Systems, which in response increases the significance of ontology maintenance. Ontologies need to be kept updated for the dependent systems to remain usable. With the increase of changes occurring in the represented domains, ontology evolution becomes a necessary process.

Ontologies are often large and complex structures, whose development and maintenance give rise to certain interesting research problems. For many practical applications, ontologies change over time according to some factors, such as domain changes, adaptations to different applications, and changes to our conceptualisation or understanding of a domain. Support for change management is vital to support distributed ontologies. Preserving consistency, while accommodating new changes, is a crucial task that needs special attention [3]. Also, matching between ontologies are easily affected by changes in the ontologies because a change in one ontology could effects the others.

The paper mainly addresses the problem of cooperating enterprises trying to solve the interoperability problem by introducing ontology based reconciliation solutions. Our purpose focuses on ontology lifecycle for building, matching and evolution ontologies in the enterprise Information Systems.

The goal of this research is to present a system architecture to describe the lifecycle of the application ontology incorporating building, matching and evolution processes. The paper discusses the main features of these processes and their contributions to address the problem of interoperability. The building process is based on knowledge extraction from corpuses and databases to generate the domain ontology. For this purpose, we have developed a set of ontologies intended to capture the semantics for applications integration. The matching process tries to find semantic relationships between entities of ontologies. It takes the ontologies as input and determines as output a set of correspondences to build the matching ontology. Typically, similarity measurement strategies become necessary. In evolution process, the main focus is on keeping ontology and its dependents consistent when changes occur. It includes two sub-processes. The first one is related to the application ontology evolution to guarantee its consistency. The second one concentrates on the matching evolution to highlight consequent effects of ontology evolution on dependent ontologies.

The remainder of this paper is organized as follows. Section 2 presents the building, matching and evolution system architecture. Section 3 sketches out the proposed ontology building process. Then, we present an iterative matching process in section 4. Next, we describe the five (5) major steps of the ontology evolution process and the matching evolution process to highlight consequent effects of ontology evolution on dependent ontology. Just after that, many ideas concerning mapping evolution are mentioned where the matching evolution process is showed. Section 6 discusses some related work on ontology building as well as ontology evolution. Finally, Section7 provides concluding remarks and sketches some future work.

2. BUILDING, MATCHING AND EVOLUTION SYSTEM ARCHITECTURE

EIS (Enterprise Information Systems) is defined as an enterprise application system or an enterprise data source that provides the information infrastructure for an enterprise. An EIS can have many different types including batch applications, traditional applications, client/server applications, web applications, relational databases, and so on. These systems are often materialized in enterprise reality in the form of relational databases, ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), SCM (Supply Chain Management), and legacy systems [5].

The proposed system architecture aims at offering a support for integrating heterogeneous and distributed applications, and accessing multiple ontologies (Figure. 1). It includes building matching and evolution management of application ontologies ensured by three (3) levels respectively.

Building level: A company model is a computational representation of the structure including, activities, processes, information, resources, people, behavior, goals and business constraints. The goal is to capture the sets of the enterprise applications, the activities that they perform, the required resources, the manipulated data and the invoked messages. Then, we identify the information flow, their structure and the technical infrastructure to support them for building the application ontologies.

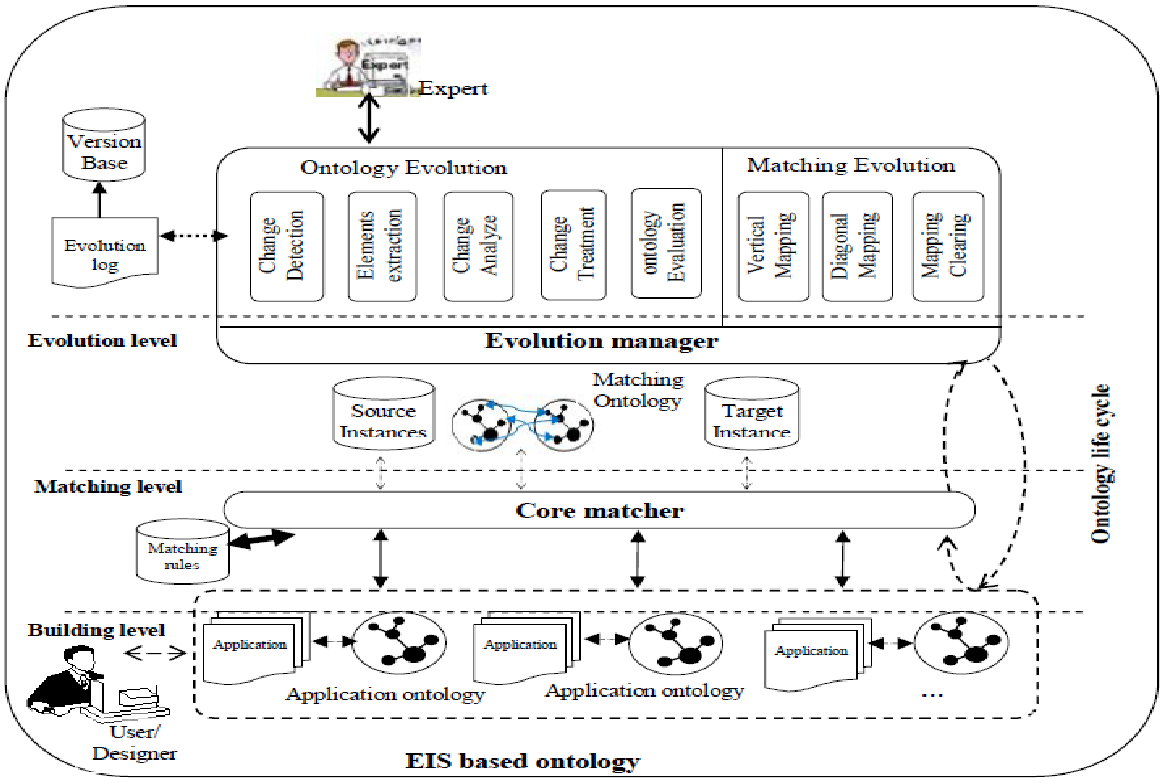

Figure 1. Building, matching and evolution system architecture.

Matching level: It concerns the application ontologies integration. The EIS based ontology consists of heterogeneous, autonomous and distributed application. Each application has its own ontology. The application ontologies are related to each other with a matching ontology. We aim at overcoming the gap between application ontologies, according to the semantic relations. A special component, named matcher, is used to perform the tasks of building the matching ontology, and transforming instances of the source ontology into instances of the target ontology.

The Matching Ontology (MO) is formally defined by a 4 tuple:
MO= (E, O, M, RT)
E: set of entities such as concepts, relations and attributes.
O: set of applications ontologies in the system.
M: $O_s \rightarrow O_t$
Mapping relation between source ontology (O_s) and target ontology (O_t)
RT: rules transformation of source instances to target instances.

Additionally, an overview about the matcher component is given in the following. The main task of the matcher is to find semantic relations between concepts of application ontologies. It involves the following tasks:

- Tries to find related concepts or attributes of ontologies and the relations between them. This can be done automatically, semi-automatically or manually with the help of domain experts.
- Represents the identified relations between ontologies based on semantic relations. It combines many algorithms to measure the similarity. Then, it adopts a multi-strategy approach to compute the concepts similarity at various levels, such as lexical, properties (roles and attributes), hierarchical and instances similarities.
- Transforms instances from the source application ontology into instances of the target application ontology by evaluating the equivalence relations defined earlier by the adaptor. Two problems that may arise are that the mappings are incomplete or the that the mapped entities differ in the context. The missing mappings can be gained through inference mechanism.

Evolution level: It concerns the evolution management. The evolution manager is composed of two parts, ontology evolution and matching evolution. The first part encompasses the set of activities which ensures that the ontology continues to meet organizational objectives and users' needs in an efficient and effective way. It includes five (5) steps; detection, elements extraction, analysis, treatment of needed change and evaluation. The second part focuses on matching evolution because dynamic environment and applications changes often have consequent effects on dependent ontologies. The role of matching evolution is to detect the new mapping between the old and new versions of the updated ontology.

3. BUILDIND PROCESS

Every day, organizations over the entire world generate reports, articles, books, emails, and all kind of textual data concerning several topics. The increase of the storage capacity of computers and servers enable these organizations to keep all files they produce without the need of deleting anything. Although this is an obvious advantage for everybody, it also implies some problems. One mainly problem they face is to know what kind of information they have, and how it is related. One way to organize information in computer science is in ontology form. This is similar to a conceptual map in which the main topics or concepts are related to each other by some kind of relations [6].

We propose an extraction and building process which includes four main phases [5]: the linguistic study and knowledge extraction, the specification, the ontology conceptualization and formalization and finally, the ontology implementation and validation (cf. figure 2).

The proposed process begins with the linguistic study and the knowledge extraction. It introduces the following steps: corpus pre-processing, extraction of terms, cleaning and filtering, and finally classification. The second phase is the ontology specification phase. It identifies the knowledge domain and the purpose of the ontology, including the operational goal, the intended users and the scope of the ontology which contain the set of terms to be represented and their characteristics. Then, the third phase consists in the conceptualization and formalization. The last phase concerns the ontology implementation and validation test. The arrival of a new corpus of text expresses the need of evolution to maintain our domain ontology. The process of evolution is invoked to guarantee the consistency and the coherence of the ontology and ensure the evolution of the data and/or the domain.

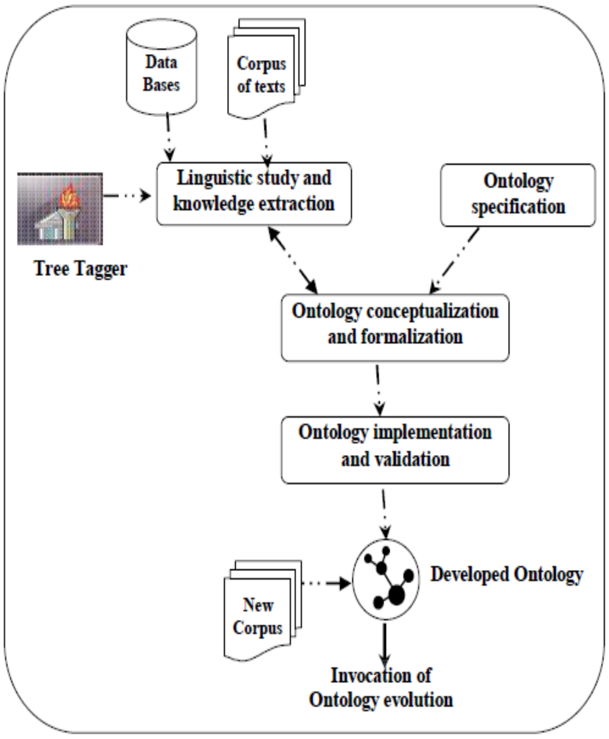
Data Bases
Corpus of texts
Linguistic study and knowledge extraction
Ontology specification
Tree Tagger
Ontology conceptualization and formalization
Ontology implementation and validation
Developed Ontology
New Corpus
Invocation of Ontology evolution

Figure 2. Application ontology building process.

3.1 Linguistic study and knowledge extraction

This phase relies on corpus work and involves the following tasks:

a. Corpus pre-processing. It aims to define a strategy to treat the missing data. It consists in normalizing the text to obtain coherent results and also, as possible, to correct human errors by the assistant of linguistic experts. This task serves to normalize the diverse manners of writing the same word, to correct the obvious spelling mistakes or the typographic incoherence and to clarify certain lexical information expressed implicitly in texts. The textual or linguistic analysis of the corpus means systematizing and making more effective the search of terms in the texts. We also used the spellchecker to avoid errors in the corpus. Then, the text is divided into a set of sentences to allow the use of the morphosyntactic analyzer Tree Tagger [7].

b. Extraction of terms and cleaning. It aims at listing all the terms contained in a corpus. To achieve this goal, we use Tree Tagger, version 3.2 [7]. It is a tool of morphosyntactic labelling and lemmatization. It serves to assign to each term in the corpus its morph syntactic category (name, verb, adjective, article, proper noun, pronoun, abbreviation, etc.) and give for each term its lemmatization. As input, corpus of texts must be organized into a set of sentences, and stored them in a file of .txt extension. Tree Tagger is used to classify extracted terms (concepts/relations) using the annotation and lemmatization information [7]. After the mining of text, we perform the cleaning operations, such as remove the stop words, change the upper case characters to lower case and. remove the irrelevant and abbreviation terms.

Several measures are usually used to select the candidate terms, we can quote the number of appearances of a term within a corpus, as well as more complex measures such as the mutual information, tf-idf, is still used in statistical distributions methods [8]. The method is based on the syntactic analysis, and uses the grammatical techniques. They put the hypothesis that the grammatical dependences reflect semantic dependences [9].

c. Classification of terms. The terms extracted from the previous step, were then classified into two categories of terms, following this idea, we try to classify the semantic elements extracted according into two categories: the concepts and the relations. Basing on the information provided by the TreeTagger tool, we classify NAME (proper nouns) as concepts and the terms of type (verb) as relations.

3.2 Ontology specification

This phase aims at supplying a clear description of the studied problem and at establishing a document of requirements specification. We need to determine why the ontology is being built, and what is its intended uses and final users.

3.3 Ontology conceptualization and formalization

The conceptualization step comprises the following tasks:
a. Glossary of terms. It contains the definition of all terms extracted in the previous phase (concepts, instances, attributes, relations). It contains all the terms and linguistic description.

b. Concept taxonomies. The hierarchy of concepts classification shows the organization of the ontology concepts in a hierarchical order which expresses the relations sub-class and super-class.

c. Definition of binary relations diagram. It specifies which concepts are linked by each relation.

d. Concept dictionary. It contains some of the domain concepts, instances of such concepts, class and instance attributes of the concepts, relations whose source is the concept and, optionally, concept synonyms and acronyms

e. Definition of binary relations tables. The binary relations are represented in the form of properties which attach a concept to another. For each relation, we define: its name, the name of source concept, the name of target concept, the cardinality and the name of the inverse relation if it exists.

f. Definition of the attributes tables. The attributes are properties which take it values in the predefined types (String, Integer, Boolean …). For each attribute appearing in the concepts dictionary, we specify its name, the type and the domain.

g. Definition of the logic axioms table. We define for each axiom, its description in natural language, the name of the concept to which the axiom refers, attributes used in the axiom and the logic expression.

h. Definition of the instances table. For each instance identified in the concepts dictionary, we specify the instance name, the concept name to belong to it, the attributes and their values.

The formalization step consists of two parts: terminological language TBOX in which concepts and relations are defined; and an assertion language ABOX in which we introduce the instances.
a. TBOX construction: We define here concepts and relations relating to our domain, by using the constructors provided by description logic to give structured descriptions at concepts and relations [10].

b. ABOX construction: The assertion language is dedicated to the description of facts, by specifying the instances (with their classes) and the relations between them.

3.4 Ontology implementation and validation

The implementation step involves the representation of the captured concepts and its relationship in a formal language. Protégé OWL [11] is a development environment with functionality for editing classes, slots (properties) and instances. Protégé is highly extensible and customizable. To evaluate correctness and completeness of domain ontology, we use query and visualization provided by PROTÉGÉ OWL. We use the built in query engine for simple query searches and query plug-in to create more sophisticated searches. We also use visualization plug-ins to browse the application ontology and ensure its consistency. The problems of coherence, correctness and completeness are then verified using the RACER inference engine [11].

OOPS is a web application based on Java [12], used by ontology developers during the ontology validation activity [13]. OOPS! scans ontologies looking for potential pitfalls that could lead to modelling errors. We enter the URL pointing the OWL document describing the ontology to be tested. Once the ontology is parsed using the Jena API the model is scanned looking for pitfalls, from those available in the pitfall catalogue. Therefore, the ontology elements involved in potential errors are detected as well as warnings regarding OWL syntax and some modelling suggestions are generated as well as explanations describing the pitfalls [13].

Once the constructed ontology is validated, it is ready to be invoked by users' requests using SWRL language. The Protégé SWRL Editor is an extension to Protégé OWL that permits interactive editing of SWRL rules [14]. It is tightly integrated with Protégé OWL and is primarily accessible through it. When editing rules, we can directly refer to OWL classes, properties, and individuals within an OWL knowledge base.

4. MATCHING PROCESS

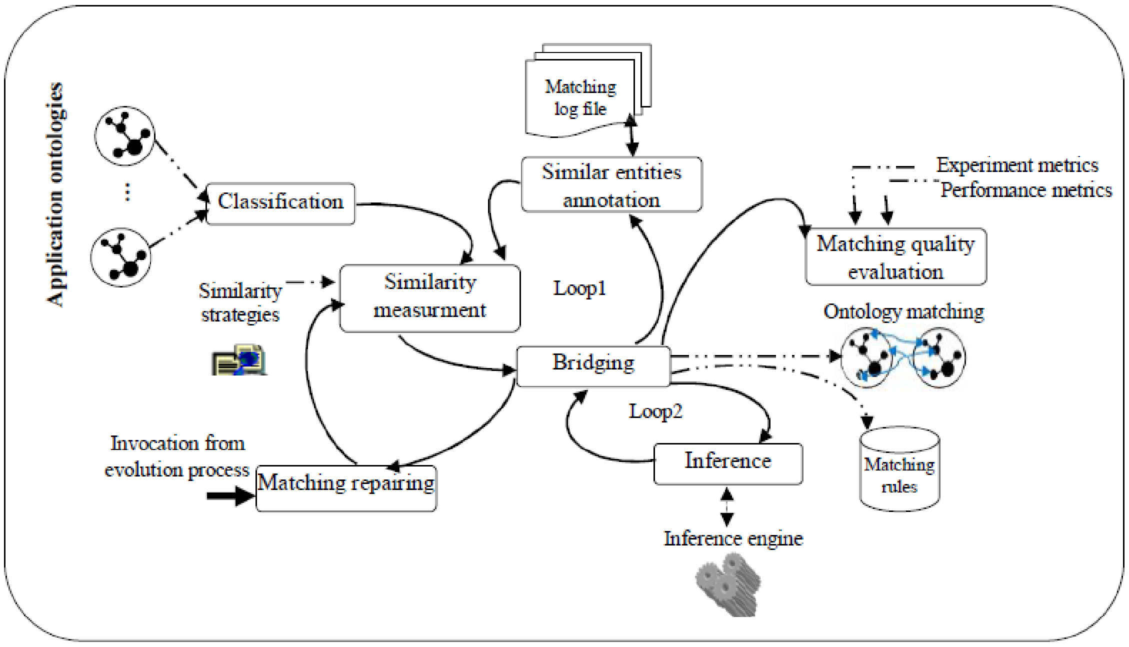

Figure 3. Iterative matching process.

Ontology matching is the process whereby semantic relations are defined between two or more ontologies to align them. It is the set of activities required to transform instances of source ontology into instances of target ontology. In this paper, we propose a matching process which includes seven main steps: classification, similarity measurement, similar entities annotation, bridging, inference, matching quality evaluation and matching repairing. In the proposed process, the classification step tries to filter the ontologies entities in order to obtain candidates entities. It is an iterative process, as described in figure 3, with a primary loop and a secondary loop. At every iteration i, a semantic bridge is created between entity ei of the source ontology and entity ej of the target ontology. In the main loop, at every iteration i, the process executes three steps: it first computes similarity sim(ei, ej) using similarity strategies, then annotates similar entities , and finally collects similar entities, selecting the most similar entity and defines a bridge. The loop ends when it becomes impossible to create a bridge between entities. The second loop concerns two steps, bridging and inference. It tries to detect new bridges basing on matching rules and human experts. These matching are then used to translate instances of source ontology into instances of target ontology. Finally, the last step focuses on experimental study to deduce some criteria to evaluate matching quality (For more detailed description of this process, see [15]).

5 EVOLUTION PROCESS

Ontology evolution is defined by Haase and Stojanovic, [16][17] as the “timely adaptation of an ontology to the arisen changes and the consistent management of these changes”. Ontology evolution is a process that supports the enrichment of the ontology by adding new entities (concepts, properties, and instances) or by modifying existing entities when new knowledge is acquired.

The usage of ontology is wide spread in Information Systems especially when building a lingua franca for resolving the terminological and conceptual incompatibilities between the enterprise applications. Ontology evolution takes place when the perspective under which the domain is viewed has changed. More specifically, ontology evolution means modifying or upgrading the ontology when there is a certain need for change as communities of practice concerned with a field of knowledge develop a deeper understanding of the domain. Ontology change management deals with the problem of deciding the modifications to perform in ontology, implementation of these modifications, and the management of their effects in dependent data structures, ontologies, services and applications [18].

One of the crucial tasks faced by practitioners and researchers in the area of knowledge representation is to efficiently encode the human knowledge in ontologies. Maintenance of usually large and dynamic ontologies and in particular adaptation of these ontologies to new knowledge is one of the most challenging problems in the Semantic Web research. This has led to the emergence of several different, but closely related, research areas such as ontology integration, merging, and versioning [3].

In our study, we focus on the ontology evolution and mappings evolution between related ontologies because they stand for the basis of all types of relations between ontologies, such as merging, integration and alignment. For this purpose, we describe two sub-processes. The first one is related to the application ontologies evolution. The second one concentrates on the matching evolution.

5.1 Ontology evolution process

Ontologies are not static entities but evolve over time. We aim in our work to propose an evolution management system to allow evolving, versioning and exploiting application ontologies in dynamic environments. This system helps the designer, user, and expert to supervise the required changes and provides interfaces to participate to the ontology evolution process (cf. Figure. 4).

5.1.1 Change detection

An evolution process requires some modifications to occur. It is, thus, necessary to identify the needs of evolution and the compatible changes to apply to the existing ontology. These modifications are expressed informally by different ontology actors (User, expert and ontology designer). The actors can express ambiguous, vague or redundant modifications. These needs will be expressed semi-formally according to one or several types of changes to be applied to create the new version of ontology.

The interview is commonly used to capture the possible changes. It enables the ontology designer to ask periodically questions and allows to the ontology user and experts some freedom to express their answers. The interview includes specific and general questions. The first one concerns the experts to capture specific information. This kind of questions is structured and has the dichotomous (Yes/No) answer. The second one concerns the ontology user to explore an issue or a specific need. This kind of questions is unstructured and its answer is a corpus of text.

Example 1: Expert question
a- Does change affect the ontology properties?
-Yes
-No
b- Does change concern ontology instances?
-Yes
-No
Example 2: User question
a- Does change need to adapt functional requirements?
b- How can the needed change improve the ontology use?

The output of this step is a set of corpus of texts. They enable the ontology designer to capture the needed change(s).

5.1.2 Elements extraction

We refer to linguistic study and knowledge extraction phase in the ontology building process to discover the pertinent terms and the type of change(s).

5.1.3 Change analysis

To resolve changes, we must identify and represent them in a suitable format. Changes must be formally expressed through types of changes. The composed changes which express a sequence of several elementary changes forming only one logical entity together [17].

5.1.4 Change treatment

During this step, it is necessary to determine the direct and indirect types of changes to be applied. In case of ambiguity or in presence of several possibilities, the ontology actors (user, expert and designer) decide on the action to occur.

All changes, and derived ones, confirmed by the designer are applied to the ontology. Consequently, the changes are physically applied to the ontology. The implemented changes need to be propagated to all interested parts in the ontology.

5.1.5 Ontology evaluation

It is essential to verify the consistency of the ontology in relation to the semantics of the ontology changes. At the end of the evolution process, a new ontology version is created. At this level, we decide whether to preserve the old version of ontology in the version base or not. The last task in this step is to keep track of the performed changes in the evolution log. The latter records the history of applied ontology changes as an order sequence of information.

A change in one application ontology in the system could have extensive effects on other related ontologies. This is especially important when ontologies are used as basis for semantic integration of enterprise applications. To handle this problem, we have proposed a matching evolution process that defines ontologies versions mappings and new mappings.

In order to avoid performing undesired changes, before applying a change to ontology, a temporary version of the ontology is created to support the change activities.

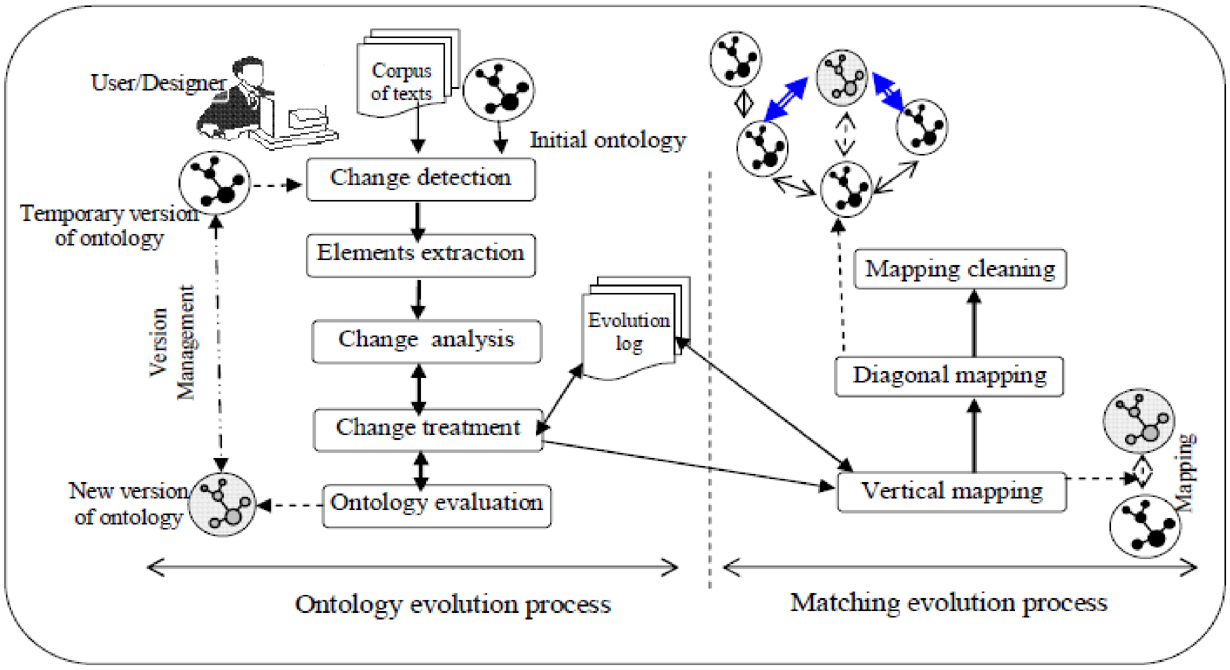

Figure 4. Ontology and matching evolution process.

It enables the ontology engineer to accept or reject the suggested changes and eliminate the changes that can cause ontology inconsistency. Moreover, ontology designer should check the results of a change request on the temporary version to ensure consistency. At the end, the designer can perform successively all the changes on the concerned ontology.

5.2 Matching evolution process

Multi-ontology means the existence of multiple ontologies related to each other in many ways [19]; reuse fusion, alignment and integration, to adapt to the various tasks of the EIS. These ontologies must be accessible by different applications and must even exchange semantic information. That is achieved through the mapping of ontologies which is necessary for the management of multiple and heterogeneous ontologies. It is due to its capacity to provide a common layer allowing the access to ontologies and the semantic exchange of information. The problem is how to manage the ontology versions in the system when the ontology evolves?

Additionally, our work is articulated around the ontological mappings evolution after an ontology evolution. Therefore, we define the three following types of mappings:

-The horizontal mappings are the set of existing mappings between the old version of evolved ontology and related ontologies.

-The vertical mappings are mappings between the old version of the evolved ontology and its new version.

-The diagonal mappings are those mappings that exist between the new version of the evolved ontology and all related ontologies. These diagonal mappings are new mappings that are generated when ontology evolves. Therefore, the detection of these diagonal mappings in an automatic way constitutes the principal objective of our work. The diagonal mapping is the composition of horizontal mapping and the vertical mapping.

We have proposed a matching evolution process composed of three (3) steps. The first one is the detection of the vertical mappings between the evolved ontology versions (old, new). For that reason, we studied the effects and the correspondences derived from the application of the change operations. Then, the diagonal mappings are obtained by composition of vertical mappings with the horizontal ones existing between evolved ontology and related ontologies. Finally, we eliminate the invalid and useless correspondences of the obtained mappings.

5. DISCUSS AND RELATED WORK

A range of methods and techniques have been reported in the literature regarding ontology building methodologies. We have selected some methodologies whose proposals meet the design criteria mentioned above. Given that ontologies are mainly used in ontological engineering, many of the existing methodologies are geared to the organization and exchange of information in computer systems, as well as in the Semantic Web. Nevertheless, we consider that it is possible to adapt those methodologies to the aims of data and text-mining. Some of them are, for instance, Uschold and King’s [20], METHONTOLOGY [21], On-To-Knowledge [22] and Noy and McGuinness’ [23]. Other methodologies arose from the work by terminology researchers interested in taking advantage of the features of ontologies for extracting knowledge from local resources. The most relevant is TERMINAE [24]

Based on these results, METHONTOLOGY meets the most criteria, with the exception of corpus based knowledge extraction. TERMINAE also complies with all the requirements. Therefore, we propose to create a methodology that combines the best characteristics of METHONTOLOGY, on one side, and of TERMINAE on the other. The proposed process is completely suitable to the domain of ontological engineering and knowledge extraction from corpuses and databases.

According to Stojanovic [17], “Ontology Evolution is the timely adaptation of ontology to the arisen changes and the consistent propagation of these changes to dependent artefacts (i.e. Dependent ontologies, ontology instances, applications using ontology)”. Ontology evolution is a complex process, due to the variety of sources and consequences of changes. Ontology evolution requires taking into account the effects of each change on the ontology to ensure uniformity in the basic ontology and all dependent objects.

Research on ontology evolution is being carried out by different researcher’s groups, and their approaches overlap with each other. The current state of the art can be found in [3], [16]. While some of these tools are ontology editors, others provide more specialized features to the user, like the support for evolution strategies, collaborative edits, change propagation, transactional properties, intuitive graphical interfaces, undo/redo operations etc.

Despite these features, our work focuses on the ontology change and matching evolution between related ontologies. Furthermore, we propose two processes. The first one is related to the application ontologies evolution. The second one concentrates on the matching evolution. We have also address the problem of undo/redo operations by using temporary version of the concerned ontology.

6. CONC LUSION

Semantic interoperability among applications, systems, and services are mostly based on ontology. A solution is to use an ontology based approach associated to enterprise applications. It provides a semantic layer to encapsulate the applications’ heterogeneity. In this paper, we have outlined architecture for application ontologies lifecycle for building, matching and evolution management.

The goal of this research study is to extract knowledge by mining corpus of text to build application ontology. This article deals with knowledge extraction using a text mining approach. More precisely, we concentrate on the extraction and construction process which includes four main phases: the specification, the linguistic study and knowledge extraction, the ontology conceptualization and finally, the implementation of the developed ontology. We use also tools of terminological extraction such as Tree Tagger for the morpho-syntactic labelling and Protégé OWL for the implementation of the ontology.

We have also developed an evolution management system to allow evolving, versioning and exploiting application ontologies in dynamic environments. This system allows the designer, the user and the expert to supervise the required changes, and provides interfaces to participate to the ontology evolution process.

For the future, we identified a number of open issues, we to address in future work. We will improve tool support in the building process by investigating ways of automatic ontology extraction from data base schema. Particularly interesting is the question of how to combine a top-down modeling approach (the way humans think) with a bottom-up approach (which results from automatic ontology extraction). Furthermore, we intend to integrate our matching tool with (semi-) automatically generated data dictionaries, in order to help domain and/or modeling experts faster understand foreign domains, during the matching process.

7. REFERENCES

[1] Jochem, R. 2010. Enterprise interoperability assessment. In Proceedings of the 8th International Conference of Modeling and Simulation, Hammamet Tunisia. 10-12,

[2] T. R. Gruber, “Towards principles for the design of ontologies used for knowledge sharing”, International Journal of Human-Computer studies, Vol 43, 907-928, 1995.

[3] A. M. Khattak, K. Latif, and S. Y. Lee, "Change management in evolving web ontologies", Knowledge based Systems, (SCI IF: 1.574), ISSN: 0950-707051, 2012.

[4] Hong-Hai. Do. Melnik. S. Erhard. R. 2002. Comparison of Schema Matching Evaluations. In Proceedings of International Workshop on Web Databases, German, Informatics Society.

[5] Driouche, R. Bensassi, H. Kemcha, N. 2015. Domain ontology building process based on text Mining from medical structured corpus. In Proceedings of the International Conference on Digital Information Processing, Data Mining and Wireless Communications, Dubai.

[6] J. I. Toledo-Alvarado, A. Guzmán-Arenas, G. L. Martínez-Luna, “Automatic building of an ontology from a corpus of text documents using data mining tools “ Journal of Applied Research and Technology, Vol. 10(3), 398-404, 2012.

[7]http://www.cis.unimuenchen.de/~schmid/tools/TreeTagger/

[8] Winkler. W. E. 1990. String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In Proceedings of the Section on Survey Research Methods (American Statistical Association). 354–359.

[9] H. Luong, S. Gauch, Q. Wang, and A. Maglia, “An ontology learning framework using focused crawler and text mining”. International Journal on Advances in Life Sciences, Vol 1(23), 99-109, 2009.

[10] F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, P. F. Patel-Schneider. 2003. The description logic handbook: Theory, implementation and applications. Cambridge University Press, Cambridge, UK.

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

[12]http://www.oracle.com/technetwork/java/javaee/overview/index.html

[13] M. Poveda. M.C. Suárez-Figueroa. A. A. Gómez-Pérez. 2010. Double classification of common pitfalls in ontologies. In Proceedings of the Workshop on Ontology Quality at the 17th International Conference on Knowledge Engineering and Knowledge Management. 1-12. Lisbon, Portugal.

[14] Golbreich, C. Imai, A. 2004. Combining SWRL rules and OWL ontologies with Protégé OWL Plugin, Jess, and Racer. In Proceedings of the 7th International Protégé Conference, Bethesda, MD.

[15] Driouche. R. 2012. A Mapping process for semantic integration of enterprise applications. In Proceedings of the 3rd international Arab conference on e-technology, Zarqa University, Jordan, 100-107.

[16] Haase, P. and Sure. Y. 2004. State of the art on ontology evolution. SEKT Deliverable.

[17] Stojanovic, L 2004 Methods and tools for ontology evolution. Doctoral thesis, University of Karlsruhe.

[18] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, G. Antoniou. “Ontology change: classification and survey”. Knowledge Engineering Review, Vol 23(2), 117-152, 2008.

[19] N. Choi, I. Song, H. Han, “A survey on ontology mapping”, ACM SIGMOD Record, Vol 35 (3), 2006, 34-41.

[20] Uschold, M. and King, M. 1995. Towards a methodology for building ontologies. In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, D. Skuce (Ed.), Montreal, Canada.

[21] Gómez-Pérez, A., Fernández-López, M., and Corcho, Ó. 2004. Ontological engineering: with examples from the areas of knowledge management. London: Springer Verlag. Greenwood, Edition. Metodología de la investigación social. Buenos Aires, Argentina: Paidós.

[22] S. Staab, H. P. Schnurr, R. Studer, and Y. “Sure, knowledge processes and ontologies”. IEEE Intelligent Systems, Vol. 16 (1), 26-34, 2001.

[23] Noy, N. F. and McGuinness, D. L. 2001. Ontology development 101: A guide to creating your first ontology [online]. Technical Report Stanford Knowledge Systems Laboratory.

http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noymcguinness.html.

[24] Aussenac-Gilles, N. Després, S. and Szulman, S. 2008. The TERMINAE method and platform for ontology engineering from texts. In Proceedings of the Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge. Amsterdam: IOS Press.