MS-ONTO
Model and System for Supporting Ontology Evolution

Emile Tawamba[1], Roger Nkambou[2], Bernabé Batchakui[1] and Claude Tangha[1]
[1]*ALOCO-LIRIMA, University of Yaoundé 1, Yaoundé, Cameroon*
[2]*GDAC, UQAM Montréal, Canada*

Keywords: Ontology Engineering, Ontology Evolution, Integrity Constraints, Semantic Web.

Abstract: Ontology is becoming the key knowledge capture structure in many domains. It plays a very important role in the area of semantic web and is widely used in multiple fields including Intelligent Tutoring Systems (ITS) and e-Learning. Ontologies are intensively used in domain knowledge modeling in specific areas which can evolve. However, current tools used to implement ontologies fail to provide functions to adequately ensure their evolution. To deal with this issue, we have developed an ontology evolution management system named «MS-ONTO», founded on a formal description of evolution operators. MS-ONTO allows the preservation of both the internal and external integrity constraints during the ontology evolution: the external integrity meaning the preservation of its usage while internal integrity means its conformity to the constraints (implicit or explicit) related to the ontology model itself. MS-ONTO should be integrated as a plug-in in existing ontology editors such as NeOn Toolkit and Protégé.

1 INTRODUCTION

Nowadays, the use of ontologies in several research areas cannot be over emphasized (Brewster, 2007). The advent of semantic web and new approaches for data web ease the access to or the sharing of web resources. The main questions that come up remain that of interoperability of the different systems of information exchange between them. Even if some languages such as XML and RDF play an important role in the creation/description of resources, there is yet no totally satisfactory solution for the terminologies and classification systems which should be used in their indexing and scouting for necessary exploitation and sharing (Psyché, 2007). Today, ontologies greatly contribute in solving this issue by providing a formal framework of conceptual modeling of different domains (Burcu, 2006).

Ontologies are proven effective in many areas such as e-Learning, e-Commerce and many others. Ontologies are mostly used in dynamic, distributed and evolutive environments/systems. It is therefore important to update ontologies in order for them to reflect the changes (changes on the state, data, and needs related to new functionalities) that affect the systems life cycle. Many other reasons can also cause a change in ontology. For instance, changes can occur when using it in different contexts or when correcting errors in conceptualization as well as changes in initial domain specification.

In order to enable ontologies to maintain their interest as regards the applications for which they have been constructed, evolution should be considered as an integral part of the life cycle of ontologies design. However, most of the tools used in ontology engineering do not include an effective ontology evolution service. In fact, most of these systems do not provide a clear framework for automatically managing the ontology evolution. The evolution process mainly relies upon the human user. A good ontology evolution process should prevent all abnormalities that can be introduced regarding both the ontology model itself (internal abnormalities) and the application contexts in which the ontology is used (external abnormalities). It is therefore important to provide a system in which ontology evolution preserves both internal and external integrity related to the initial ontology. This paper presents an ontology evolution management method based on a formal approach using description logics and which enables the preservation of internal as well as external integrity (Noy, 2004) (Zablith, 2009).

Tawamba E., Nkambou R., Batchakui B. and Tangha C..
MS-ONTO - Model and System for Supporting Ontology Evolution.
DOI: 10.5220/0005085203190326
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2014), pages 319-326
ISBN: 978-989-758-049-9

The paper is organized as follows: the second section presents some related works on the management of ontology evolution. Our solution is then introduced in the third section where a formal description of the operators of changes is given with a general presentation of our global approach towards ontology evolution management. The fourth section is the implementation and evaluation of this solution. The paper ends with some concluding remarks.

2 RELATED WORK

This section defines some ontology related concepts followed by existing ontology evolution management techniques in order to bring out their limitations.

2.1 Definitions

(Gruber, 1993) defined ontology as "an explicit specification of a conceptualization". Ontology is also considered as the result of a complete formulation and a rigorous conceptualization (hierarchical organization of pertinent concepts, relationships between these concepts, rules and axioms binding them) (Amal, 2008).

Ontology evolution is the adaptation to changes that are brought during its life-cycle and the propagation of these changes at the level of the dependent artifact, i.e. objects referred by the ontology as well as ontologies and their related applications (Sure, 2004). There are various types of changes: changes in the modelling domain (Wohlgenannt, 2013), conceptualization and specification changes (Djedidi, 2007), etc. Given that the domain is a part of the real world, it is thus dynamic and evolves with time. Conceptualization can also change due to a new observation or a restructuring of knowledge. Ontology could be adapted to be reused in different tasks. The conceptualization of ontology could be refined through an interactive and incremental construction process. Ontology enrichment tools are another way by which an existing ontology can evolve as new concepts as well as new relationships that are extracted and added (Booshehri, 2013). It is therefore glaring that ontology must undergo changes and therefore evolve (Cuenca, 2012).

Ensuring the evolution of ontology is a costly operation which requires the services of a competent expert in this domain (Scott, 2013). However, the significant role of ontologies makes it essential that they should be kept up to date so as to reflect the changes which affect the life cycle of the systems and the applications for which they were conceived.

Many research teams and projects have worked on ontology evolution and brought out interesting findings. According to their methods, two broad categories stand out: ontology versioning and management of changes (Kondylakis, 2011).

2.2 Ontology Evolution Based on the Management of Versions

In this first group, the evolution is managed through the creation and maintenance of different versions of the same ontology. KAON, CONCORDIA and SHOE are classic examples of tools and methods that implement the view.

KAON (Gabel, 2004) offers a range of tools for ontologies and the semantic web. Presently, it is one of the rare ontology management systems which has a function dedicated to the recording of changes. During the evolution, KAON saves all the changes carried out to move from one ontology version to another in a folder as an ordered sequence of RDF/XML declarations. The main drawback in KAON approach is that, it deals only with elementary changes.

CONCORDIA (Shaban, 2010) defines a conceptual model for the management of changes of a medical terminology. This model adds to each class a unique identifier and these classes could only be subsequently and logically withdrawn but not physically erased. As such for each class, the Concordia model is capable of tracing all the parent-classes or withdrawing children through its identifiers.

HEFLIN proposes SHOE (Heflin, 1999) as a language for representing knowledge on the web. It is a language based on FOL (First order logic). SHOE is based on HTML, which offers primary terms for the management of multiple versions while allowing the association to each version of ontology, a unique identifier and a code stating the compatibility with former versions. It brings out the relevance of each revision/operation on data and requests.

As a whole, ontology evolution solutions of this group do not really treat ontology evolution. They rather provide a model for analyzing links between ontology versions, but do not pay attention either to the management of the dependent artifacts (referred resources, related ontologies, programs in which the evolved ontology is used) or to the impact of changes on the ontology internal structure and/or semantic. Moreover, the authors do not provide any functional framework to integrate the totality of methodological elements that they propose.

2.3 Evolution Based on Management of Changes

The proponents of this approach think that it is necessary to have a follow-up of changes: the request for change has to be interpreted, analyzed and executed under control. In fact, the system should be able to notify a change that might lead to an inconsistent ontology, better still, it should at least help the engineer in evolution operations if it cannot completely automate them.

The ontology must remain consistent while evolving. Ontology is consistent in relation to a model if and only if it respects all the constraints of the model. These constraints are either invariables or user-defined.

The invariables constraints are those related to the ontology structure; for instance, removing a root concept should not be allowed without a correct reorganization of the ontology. User-defined constraints are those defined by the user such as the maximum number of instance related to a given concept. A change maintains consistency only when the resulting ontology is consistent. Hence, change is defined as a function *ch(args, prec, postc)* with *args* representing the arguments, *prec* representing all the preconditions and *postc* representing all the post-conditions. Some operations could lead to several options of possible ontologies; it is therefore necessary to choose the most consistent one that responds to user's needs. However, several iterations should be needed before obtaining an acceptable resultant ontology (Stojanovic, 2004).

This approach has many disadvantages; it consists of a solution which has a high algorithm complexity due to multiple possible iterations without any specific switch-off conditions. The reason being that at each stage, choices have to be made using relevant heuristics. Another approach in this category is that of Delia (Delia, 2008). It consists of a semi automatic system where the user is involved in the ontology evolution process. It proposes versioning of changes which then serve to resolve eventual semantic referencing (based on the ontology) failures. The system records the changes during the evolution phase of the ontology and provides the user with a list of possible semantic references that are impacted. The management of changes is hereby concentrated on the control of the consistency of semantic referencing (meaning the usage of the target ontology – external integrity).

On the whole, the fore-mentioned approaches do not solve the problem of evolution on all its dimensions. Some are concentrates on structural aspects (internal integrity) while others focus on the external implications (external integrity). In addition, they lack a formal framework which entirely integrates the ontology evolution engineering.

3 THE PROPOSED APPROACH FOR MANAGING THE EVOLUTION OF ONTOLOGY

Despite the fact that several works have been carried out on the evolution of ontologies, the tools which should make it possible to ensure a guided evolution which guarantees the consistency of final ontology are still to come. The solution that we propose comprises four points: we start by taking into account the semantics behind the various operations of changes which can affect ontology during its life cycle. Secondly, we set up some integrity constraints which must be respected during any execution of an operation or of a set of operations of change. Thirdly, we control the execution of operations according to some used case we proposed, and we end by giving a detailed report of the execution of the changes with the new version of ontology.

3.1 Formal Description of Operators of Change

These operators of change are formally defined using Description Logic (DL). Evolution operations are usually done on ontology elements including T-Box axioms (concepts, roles, restrictions, attributes etc.) as well as A-Box axioms (instances).

There are two groups of operators: Basic DL operators such as negation, generalization, specialization, etc., and the other operators which we defined using DL primitives (Stojanovic, 2004) (Baader, 2007) (Gagnon, 2007). These operators include adding, removing, merging or grouping ontology elements (concept, role, restriction, etc.) Each operator is semantically defined as illustrated in the following for 3 instances of operators. Our contribution here is that, actions related to each operator are explicitly represented in the semantic.

The following examples illustrate some of these operators and their formal semantic descriptions. Let us consider (O, C, I, intscon, $H_{\mathcal{T}}$) where O is an ontology, C is a set of concepts, I is a set of instances, instcon is a function that relates a concept to the set of its instances and $H_{\mathcal{T}}$ is a relation called concept hierarchy, we have:

- AddInstance (i,C):

Add an instance to a concept C
Pré-conditions: $j \in instconc(C) \wedge (\exists i \in I \setminus \{j\}, i \notin instconc(C))$
Post-conditions: $i \in instconc(C)$
Actions :
1. $\mathcal{A}$ (ABox containing a statement $C(x)$)
2. $y \neq x$ (y an instance different from x)
3. $C(y)$ (Add an instance y)

- AddConcept C: Add a concept C
Pré-conditions: $(C, root) \notin H_{\mathcal{T}}^{*} \wedge \exists D \in \mathcal{T}$
Post-conditions: $(C, root) \in H_{\mathcal{T}} \vee (C, root) \in H_{\mathcal{T}}^{*}$
Actions:
1. $\mathcal{T}$ (TBox)
2. $C \sqsubseteq D$ (C is subsume by D)
3. $C \sqsubseteq \top$
- RemovConcept C: Remove a concept
Pré-conditions: $C \in \mathcal{T} \setminus \{root\} \wedge \exists D_1, D_2 \in \mathcal{T}((C, D_1) \in H_{\mathcal{T}} \wedge (D_2, D_1) \in H_{\mathcal{T}})$
Post-condition: $(C, D_1) \notin H_{\mathcal{T}}$
Actions:
1. $\mathcal{T}$ (TBox containing concepts $C_1, \ldots, C_n$, C and D)
2. $(\exists 1 \leq i \leq n)\ C_i \sqsubseteq C \rightarrow C_i \sqsubseteq D$
3. $C \sqsubseteq D$ (C is subsume by D)
4. $C \sqsubseteq \perp$ (Remove C)

3.2 Constraints of Semantic Integrity

We also defined a set of constraints of semantic integrity (CSI) which must be taken into account throughout the evolution process. Each single operation of change should lead to the checking of the CSI before the system can approve or reject the change. In all cases, a report must be drawn up.

An example of CSI could consist in setting a maximal DL expressivity that should be preserved after the evolution process. In this way, it should be possible to preserve the logic of the initial ontology if needed. In this case, taking into account the expressivity includes three main steps:

- Identifying the structures allowed for each sub-language in a «database of structures of sub-languages». Several level of language could be obtained: For example OWL-LITE, OWL-DL, OWL-FULL (Zghal, 2007) if we refer to OWL1.
- Creating an inspector of structure, a program that is able to verify the ontology structures and to categorize them, in order to determine the ontology expressivity.
- Ensuring that the degree of expressivity of the ontology is maintained.

The expressiveness management model described above is presented in figure1. The initial ontology undergoes an audit which determines its expressiveness. Each operation of change which is applied to the ontology must absolutely respect the expressiveness. After the ontology expressiveness is verified, a report is produced (Figure 1).

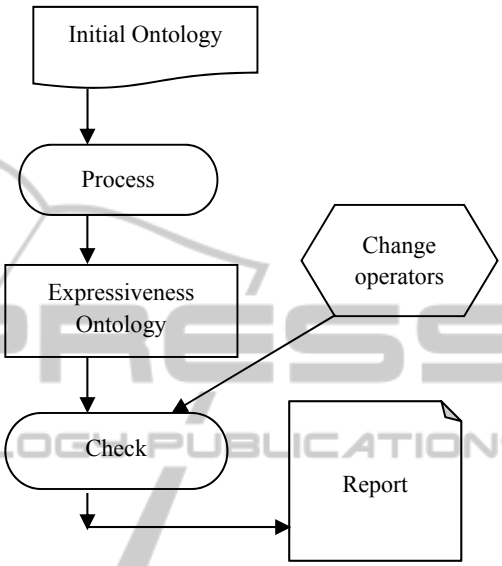

Figure 1: Model of management of the expressivity.

3.3 Execution of the Operations of Change

The execution of an operation of change or a range of change goes beyond the control of the expressivity of ontology. In fact, in addition to the CSI, each operation has a set of pre-conditions and of post-conditions to be verified before and after execution.

Our overall model of ontology evolution is presented in Figure 2.

The list of changes is a set of operations recorded after evolution of the ontology or a set of operations which one must apply to make the ontology evolve. CSI is a set of constraints of semantic integrity which must be verified to guarantee the consistency of the ontology along its evolution. The list of the operation of changes is drawn up by the operators of DL on the one hand, and the operations which we defined on the other hand as mentioned in the previous sections. The processor is a parser which carries out the operations of change while respecting constraints. A report is produced at the end of the execution.

4 CURRENT IMPLEMENTATION AND RESULTS

4.1 Implementation

MS-ONTO is implemented and current use-case helps to control the ontology evolution giving as inputs, a list of changes and the initial ontology on which these changes are applied. The constraints of semantics integrity (CSI) guaranteeing the consistence of integrity all along the evolution and their procedure of verification are also known to the system.

A parser executes each change operation while respecting predefined constraints. A report is produced at the end of the execution.

The operation class modelizes all the operations which can be applied to the ontology. The diary of changes contains the operations of change which the ontology will undergo. The list of changes (changes log) can be provided directly to the system or edited by an Expert-user using an interface (Figure 4). The initial ontology which may eventually evolve is contained in an OWL file.

The operations contained in the log are then applied to the ontology. The integrity constraints are verified and a detailed report is produced (Figure 5).

The overall processes and information sharing in our system are depicted in Figure 3. Processing_01 helps the expert to edit or choose the change operators that will be applied to the ontology. Processing_02 checks if each operator preserves the expressiveness of the ontology. Processing_03 executes the operations under the specified CSI. At the end of all these processes, a report is produced containing explanations about operation execution failures if any.

4.2 Validation

A very initial test of our system has been done using a simple ontology on which we applied 6 change operations including 2 that had to produce erroneous results, which were successfully detected by the system. However, a thorough validation is still to come using more complex ontologies and meaningful records of change operations.

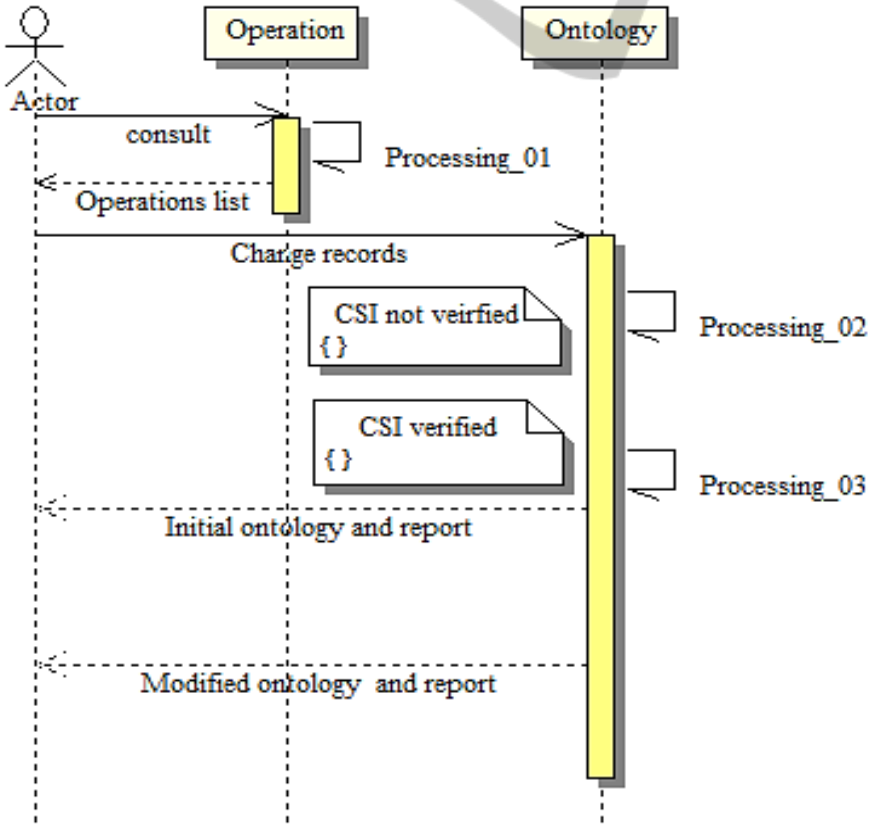

Figure 3: Sequence diagram.

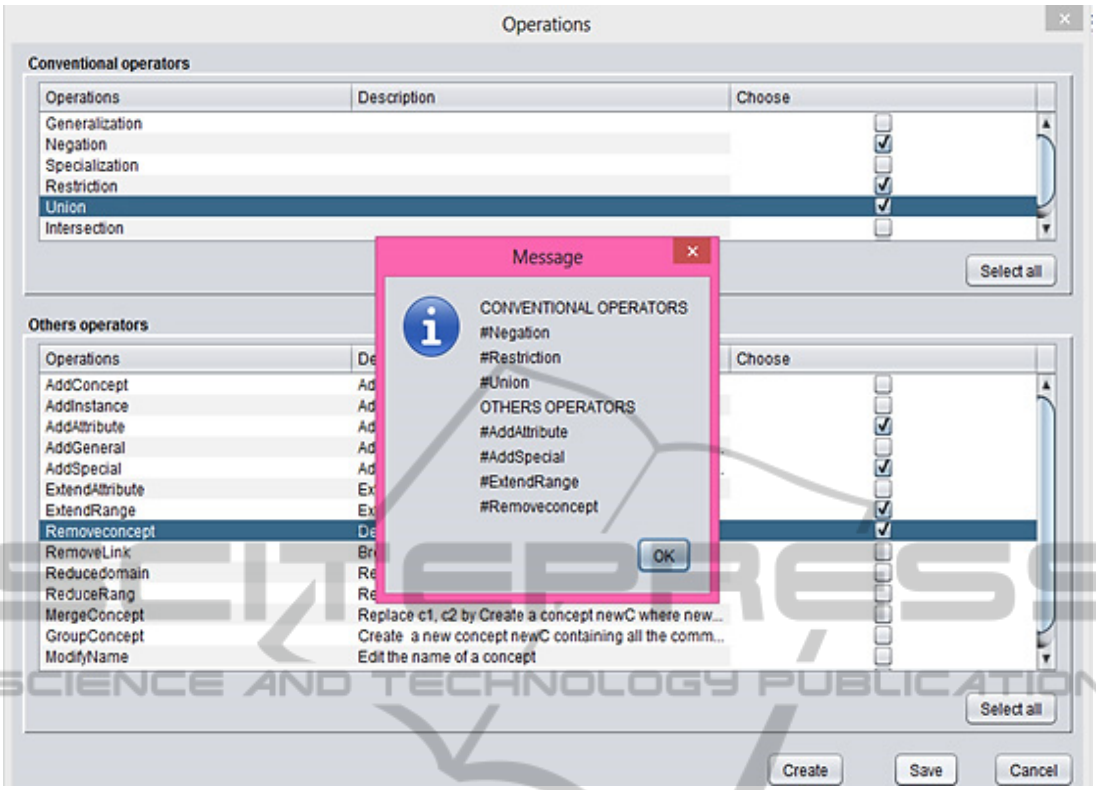

Figure 4: The interface that allows the user to build up its journal of change.

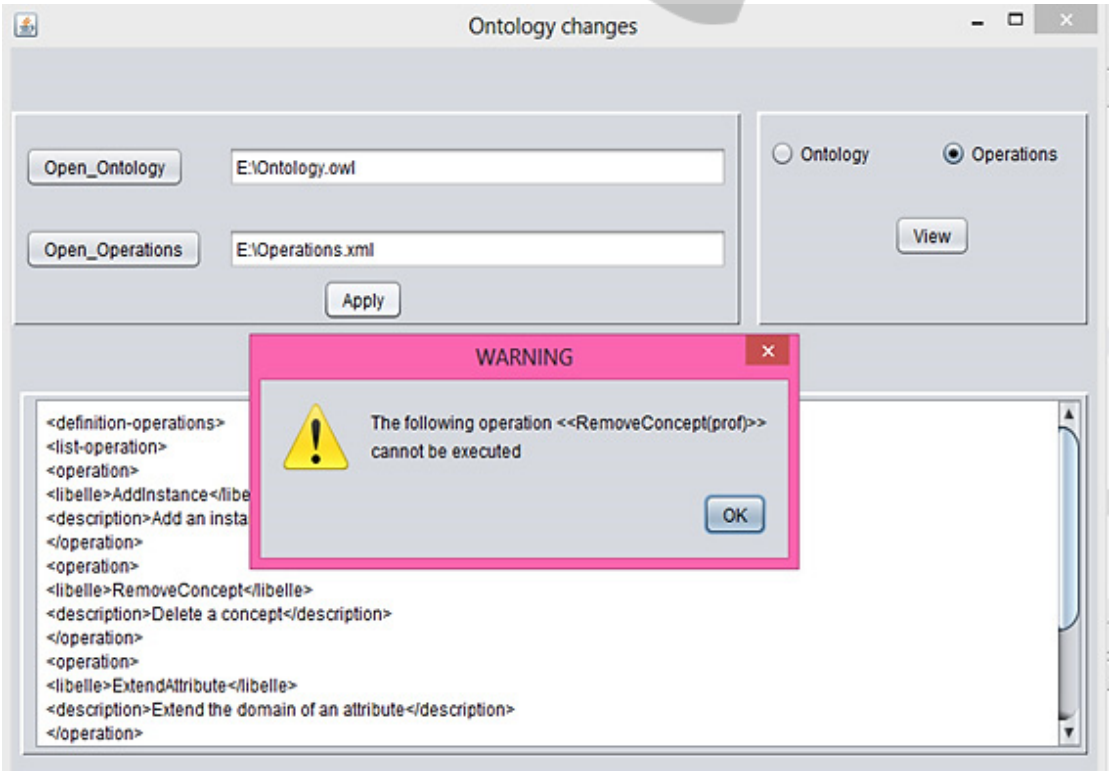

Figure 5: Viewing the log of changes / Initial Ontology / Execution of operations of change.

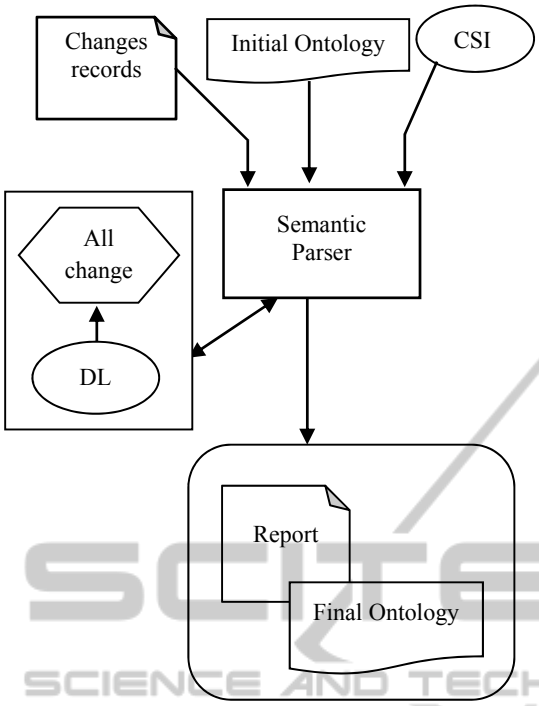

Figure 2: Overall model of MS-ONTO.

5 CONCLUSION AND FUTURE WORKS

We have described a new system which makes it possible to follow and control the evolution of ontologies. Our solution is founded on the formal description of the semantic associated to each operation of change based on the Description Logics. Each operation of change contains a set of preconditions and a set of post-conditions to fulfill in order to guarantee the consistency of the ontology during the evolution. Furthermore, a set of constraints of semantic integrity was defined to better ensure the consistency and the integrity of the final ontology.

We intend to add to these constraints, a set of metric of quality in order to measure the quality of the final ontology. Also, many works remain to cope with the external validation of the evolution which is usage-context dependant. The key issue here is to look for a possible generic core that can ease the specification of context-dependant CSI related to the ontology use in a specific domain. A first step could consist in reconsidering Delia's work on semantic reference of learning resources using our own system.

This work is still in its initial stage but we have provided a framework where both internal and external validation of the evolution process can be managed. This is a contribution as such integrated solution does not exist. We intend to implement SM-ONTO as a plug-in in order to ease its integration with classic ontology engineering tools such as Protégé. We should also validate the system and collect some data for its improvement. Finally, the current implementation is used as a validation tool to analyze a list of changes operated on a given ontology and provide feedbacks on that. Our next step here will be to use the system in other use-cases where it could act as an active coach during a live ontology evolution (even building) process.

REFERENCES

Amal, Z., Nkambou R., 2008. Building Domain Ontologies from Text for Educational Purposes. *IEEE Translations on Learning Technologies, vol. 1, N° 1,* p.49-62.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., 2007. The Description Logic Handbook: Theory, Implementation, and Applications. *Cambridge University Press,* 622 pages.

Brewster, C., and O'Hara, K. 2007. Knowledge representation with ontologies: Present challenges-Future possibilities. *ScienceDirect Int. J. Human-Computer Studies 65*, p.563–568.

Booshehri, M., Malekpour, A., Luksch, P., Zamanifar, K., and Shariatmadari, S. 2013. Ontology Enrichment by Extracting Hidden Assertional Knowledge from Text. *International Journal of Computer Science and Information Security, vol. 11, No.5*, p. 64-72.

Burcu, Y., 2006. Ontology Evolution and Versioning: The state of art. *Vienna University of Technologies, Asgaard-TR-2006-3,* 28 pages.

Cuenca, G., Jimenez-Ruiz, B., Kharlamov, E., Zheleznyakov, E., and Dmitry, 2012. Ontology Evolution under Semantic Constraints. *Proceedings of the 30th International Conference on Principles of Knowledge Representation and Reasoning,* p.137-147.

Delia, R., 2008. Gestion de l'évolution d'une ontologie : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions de l'ontologie. *Thèse de Doctorat, Université du Québec à Montréal,* p.68-180.

Djedidi, R., Aufaure, M., and Abboute, H., 2007. Evolution d'ontologie: Validation des changements basée sur l'évaluation. *Journées Francophones sur les Ontologies (JFO),* p. 55-73.

Gabel, T., Sure, Y., and Voelker, J., 2004. KAON-ontology management infrastructure. *SEKT informal deliverable,* 58 pages.

Gagnon, M., 2007. Logique descriptive et OWL. *Ecole Polytechnique de Montréal,* 55 pages.

Gruber, T., 1993. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *In International Journal Human-Computer Studies 43,* p.907-928.

Heflin, J., Hendler, J., and Luke, S., 1999. A knowledge Representation Language for Internet Applications. *Technical Report, University of Maryland at College Park,* 30 pages.

Kondylakis, H., Plexousakis, D., 2011. Exelixis: Evolving ontology-based data integration system. *Proceeding of the international conference on Management of data,* p. 1283-1286, ACM.

Noy, N., Klein, M., 2004. Ontology Evolution: Not the same as Schéma Evolution. *In Knowledge and Informations Systems, vol.6 Issue 4,* p.428-440.

Psyché, V., 2007. Rôle des ontologies en ingénierie des EIAD: Cas d'un système d'assistance au design pédagogique. *Thèse de Doctorat, Université du Québec à Montréal,* p.13.

Scott, B., Pete, B., Mark, B., David, K., and Debra, S. 2013. PRONTOE - A Case Study for Developing Ontologies for Operations. In: *5th International Conference on Knowledge Engineering and Ontology Development (KEOD). SciTePress, Vilamoura, Portugal*, 9 pages.

Shaban-Nejad, A., 2010. A frame work for analyzing changes in health care lexicons and nomenclatures. *PHD Thesis, Concordia University, Montreal, Quebec, Canada,* p.86.

Stojanovic, L., 2004. Methods and Tools for Ontology Evolution. *Thèse de Doctorat, Karlsruhe,* 249 pages

Sure, Y., Tempich, C., 2004. State of art in ontology engineering methodologies. *SEKT informal deliverable 7.1.2, Institut AIFB,* University of Karlsruhe.

Wohlgenannt, G., Belk, S., and Schett, M., 2013. A Prototype for Automating Ontology Learning and Ontology Evolution. *In: 5th International Conference on Knowledge Engineering and Ontology Development (KEOD). SciTePress, Vilamoura, Portugal*. p. 407-412.

Zablith, F., 2009. Evolva : A Comprehensive Approach to Ontology Evolution. *Proceedings of the 6th European Semantic Web Conference, LNCS 5554, eds. L. Aroyo et al., Springer-Verlag, Berlin, Heidelberg*, p. 944-948.

Zghal, S., Yahia, B., Nguifo E., and Slimani. Y., 2007. SODA : une approche structurelle pour l'alignement d'ontologies OWL-DL. *Actes 1ères journées francophones sur les ontologies (JFO),* p.1-20.