Ontology Evolution with Semantic Wikis

Mauro Dragoni and Chiara Ghidini

FBK-irst, Via Sommarive 18 Povo, I-38123,Trento, Italy
[dragoni,ghidini]@fbk.eu

Abstract. One of the challenges of using ontology evolution approaches is the capability of exposing the ontology with information that may be used by third-party tools for tracking the updates carried out on the ontologies. In this paper we present and enhanced version of the MoKi tool equipped with an ontology evolution approach that permits to evolve an ontology by providing a mechanism for facing the tracking challenge. By considering, as use case, the context of the Organic.Lingua EU-project, we will discuss the effectiveness of the proposed approach and possible drawbacks.

1 Introduction

Ontologies are dynamic entities that evolve over time because they are affected by the necessity of applying changes in the domain, in the conceptualization, or in their specification. As stated in [25], the ontology evolution may be defined as "the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artifacts."

The management of ontology evolution has several challenges associated, ranging from the adequate control of ontology changes to the administration of ontology versions. Ontologies evolution refers to the activity of facilitating the modification of an ontology by preserving its consistency; it can be seen as a consequence of different activities during the development and, mainly, the maintenance of the ontology.

Starting from a high level perspective, we can classify ontology evolution in two main scenarios: the first one is that only one ontology is created, maintained, and made evolve, by one or more users; the second, and more complex one is that different users work on different ontologies, and then, these ontologies are merged, and made evolve, in one single inter-ontology. The challenges raised by using an ontology evolution mechanism become more complex in scenarios where concepts defined in the ontology are used to tag resources that are then retrieved by using third-party tools or web search environments. By taking into account this scenario, the tool used for managing the ontology and its evolution has to provide the capability of injecting, into the ontology, some information for maintaining the retrieval effectiveness of the search environments used to retrieve resources that are tagged with concepts that have been updated or deleted.

In this paper we analyze the scenario explained above by considering, as use case, the context of the Organic.Lingua EU-project[1]. This project aims to create a multilingual portal where users are able to retrieve agricultural resources tagged with concepts defined into an ontology. The ontology is managed by using a semantic wiki tool, called

[1] http://www.organic-lingua.eu

MoKi, that has been equipped with an ontology evolution approach focused on the maintenance of the retrieval effectiveness of the Organic.Lingua platform.

The paper is organized as follow: Section 2 presents a review on the ontology evolution field. Section 3 presents the MoKi tool, while in Section 4 we discuss how MoKi will be used for facing the ontology evolution challenge and we compare the presented enhanced version of MoKi against the other semantic wikis presented in the literature. Section 5 shows which are the issues raised when ontology evolution approaches are used; finally, Section 6 concludes.

2 Related Work

Several approaches have been presented in the literature about ontology evolution. The aspects that are taken into account may be split in two different categories:

- **Schema evolution:** it is the ability to change the schema of the ontology without loss of data and by maintaining the consistency of the ontology, moreover, it has to be possible to access to both old and new data through the new ontology schema;
- **Schema versioning:** it is the ability to access all the data (both old and new) through all versions of the ontology. A version is a reference that labels a quiet point in the definition of a schema. Therefore, all resources have to be retrievable by using coherent concepts and by using every historic definition of the ontology schema.

Both aspects are strictly correlated with the context of the Organic.Lingua project due to the necessity of maintaining the retrieval effectiveness of the system through the changes that are carried out on the ontology.

Schema Evolution. The evolution of ontology schema is generally composed of two phases: the planning of the changes, and their implementation.

The planning of the changes consists of identifying the potential consequences of a change, and estimating what needs to be modified to accomplish a change. Such analysis, that derives from the software engineering environment, is very helpful to estimate the cost and effort required to implement the requested change. In [17] the author describes in detail how the change of a concept in the ontology might imply a cascade of changes (propagation) that may affect the entire ontology.

By analyzing the propagation of changes, it is possible to estimate which is the cost of the ontology evolution. As stated in [17], the cost of evolution is a key element in the decision on propagating changes through the ontology or not, and in [23] the authors provide an overview on approaches related to the estimation of the cost of evolution and propose a parametric model for the estimation of costs to build, maintain, and evolve and ontology.

The second phase is related to the implementation of the changes. In the literature there are identified four different kinds of approach used to propagate the effects of changes implemented in an ontology:

- immediate conversion (or coercion) [12] [15]
- deferred conversion (or screening) [1] [20]

– explicit deletion [3]
– filtering [1] [20]

After the implementation of the changes, it is sometimes necessary to restructure the ontology in order to maintain the same information capacity [14] defined as the semantic equivalence between different versions of the ontology [19]. This step may introduce inconsistencies in the ontology schema that have to be managed.

There are two different schools of thought about how to face this problem: the first one is the "consistency maintenance", that it is a conservative approach in which the system is kept consistent at all costs. Some approaches related to consistency maintenance are presented in [25] and [11]. While, the second strategy simply consists in the "inconsistency management", in which inconsistencies are considered inevitable and, therefore, it is necessary to manage them. The main studies in the literature about inconsistency management are related to the localization of inconsistencies, for example [6] it is presented an approach based on the use of sub-ontologies to identify inconsistencies, while in [18] the authors present a logic-based method to detect some kinds of inconsistencies.

Schema Versioning. Versioning is in general the mechanism that allow users to keep track of all changes applied to the definition of something, and to undo changes by rolling back to previous versions. The same principle is applied to ontologies, in which it is necessary to track the changes applied to the ontology schema, in order to permit to users to roll-back to previous versions if it is necessary.

There are two main strategies that are adopted to establish the version of an ontology: "state-based" and "change-based". The "state-based" versioning consists in considering the state of the ontology at certain moment in time; a new state is created each time that a change is applied by the system to the ontology schema. An example of system that supports such a versioning strategy is described in [8].

The second way to manage schema versioning is a "change-based" approach (or "operation-based") that consisting in storing information about the precise changes or explicit operations that are performed on the ontology. The advantages of this approach, with respect to the previous one, is that it is simpler to compare different versions of an ontology and to implement undo/redo mechanisms. An example of system implementing a change-based approach is proposed in [13].

3 MoKi Tool

MoKi[2] is a collaborative MediaWiki-based [9] tool for modeling ontological and procedural knowledge. The main idea behind MoKi is to associate a wiki page, containing both unstructured and structured information, to each entity of the ontology and process model. From a high level perspective, the main features of MoKi[3] are:

[2] See `http://moki.fbk.eu`

[3] A comprehensive description of MoKi can be found in [4].

- the capability to model different types of conceptual models in an integrated manner. In particular the current version of MoKi is tailored to the integrated modeling of ontological and procedural knowledge;
- the capability to support on-line collaboration between members of the modeling team, including collaboration between domain experts and knowledge engineers.

In the context of the Organic.Lingua project, the use of MoKi has been focused on the modeling of the ontological knowledge only, while the collaborative feature is useful due to the structure of the modeling team that is composed by heterogeneous groups of domain experts and knowledge engineers situated in different geographical regions.

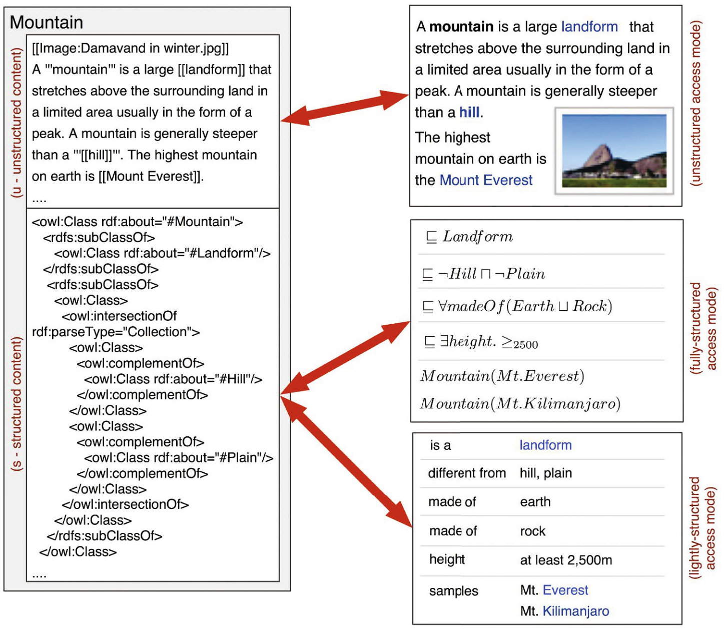
The following subsection illustrates how these features are realized in the generic MoKi architecture.

Modeling integrated ontological and procedural knowledge The capability of modeling integrated ontological and procedural knowledge is based on different characteristics of MoKi. MoKi associates a wiki page to each *concept*, *property*, and *individual* in the ontology, and to each (complex or atomic) *process* in the process model. Special pages enable to visualize (edit) the ontology and process models organized according to the generalization and the aggregation/decomposition dimensions respectively. The ontological entities are described in Web Ontology Language (OWL [24]), while the process entities are described in Business Process Modeling Notation (BPMN [16]).

Supporting collaboration between domain experts and knowledge engineers MoKi is an on-line tool based on MediaWiki, thus inheriting all the collaborative features provided by it. In addition MoKi facilitates the collaboration between domain experts and knowledge engineers by providing different access modes to the elements described on the model, as illustrated in Figure 1 for the ontology concept “Mountain”.

MoKi allows to store both *unstructured* and *structured* descriptions of the elements of the models, as shown on the left hand side of Figure 1. The unstructured part contains a rich and often exhaustive description of knowledge better suited to humans, usually provided with linguistic and pictorial instruments. Instead, the structured part is the one which is used to provide the portion of knowledge which will be directly encoded in the modeling language used to describe the specific element (OWL in the case of the concept “Mountain”). The advantage of storing the unstructured and structured descriptions in MoKi is twofold. First, informal descriptions are usually used to provide the initial description upon which the formal model is built, and to document the elements of the model (e.g., for future access and revisions). Storing the unstructured and structured descriptions in the same tool can facilitate the interplay between these parts. Second, domain experts, who usually create, describe, and review knowledge at a rather informal/human intelligible level, may find the unstructured part their preferred portion of page where to describe knowledge, while knowledge engineers should be mainly focused on the descriptions contained in the structured part. Nevertheless, by using the same tool and accessing the same pages, all of them can be notified of what the others are focused at. Moreover, the discussion facilities of wikis, together with special fields for comments, can be used by both roles to discuss on specific parts of the model.

The reader, may found more details about the general features of MoKi in [5].


Mountain
(u - unstructured content)
(s - structured content)
(unstructured access mode)
(fully-structured access mode)
(lightly-structured access mode)

```
[[Image:Damavand in winter.jpg]]
A '''mountain''' is a large [[landform]] that
stretches above the surrounding land in
a limited area usually in the form of a
peak. A mountain is generally steeper
than a '''[[hill]]'''. The highest mountain
on earth is [[Mount Everest]].
....
```

```
<owl:Class rdf:about="#Mountain">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Landform"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf
rdf:parseType="Collection">
        <owl:Class>
          <owl:complementOf>
            <owl:Class rdf:about="#Hill"/>
          </owl:complementOf>
        </owl:Class>
        <owl:Class>
          <owl:complementOf>
            <owl:Class rdf:about="#Plain"/>
          </owl:complementOf>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
....
```

A **mountain** is a large landform that stretches above the surrounding land in a limited area usually in the form of a peak. A mountain is generally steeper than a **hill**. The highest mountain on earth is the Mount Everest

$\sqsubseteq Landform$

$\sqsubseteq \neg Hill \sqcap \neg Plain$

$\sqsubseteq \forall madeOf(Earth \sqcup Rock)$

$\sqsubseteq \exists height. \geq_{2500}$

$Mountain(Mt.Everest)$

$Mountain(Mt.Kilimanjaro)$

is a	landform
different from	hill, plain
made of	earth
made of	rock
height	at least 2,500m
samples	Mt. Everest Mt. Kilimanjaro

Fig. 1. Multi-mode access to a wiki page

4 Ontology Evolution with MoKi

By considering the aims of the Organic.Lingua project, MoKi implements an evolution mechanism that permits to achieve the following goals:

– to modify the definition of a concept and at the same time to maintain an association of deprecation with the old definition;
– to delete a concept and at the same time to track the changes;
– to grant retrieval effectiveness after an update or a deletion of a concept.

The main issue related to the ontology evolution in the Organic.Lingua project is to maintain the retrieval effectiveness of the platform when concepts are both updated and/or deleted. In this Section we present how these goals are reached with the use of MoKi in the context of the Organic.Lingua project.

Concept Update. The concept update is intended as an action performed by a user consisting on modifying the definition with which a concept is identified in the ontology. A concept may be updated in different ways:

– the concept definition is only modified: assuming to have a concept defined as “A”, it is then renamed as concept “B”;
– the concept is split in two or more concepts: assuming to have in the ontology a concept defined as “A”, it may be split in the set of concepts “B”, “C”, and “D”.

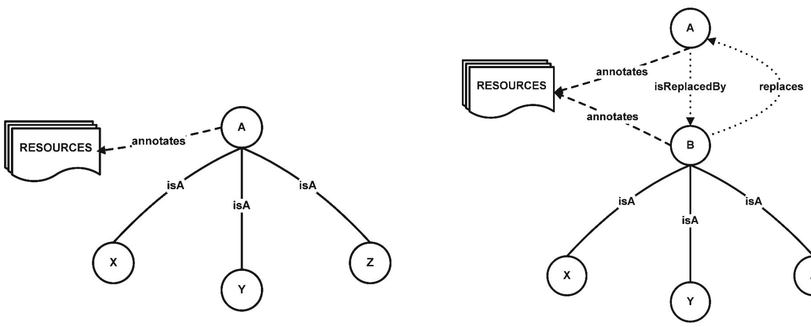


Fig. 2. Concept update by modifying its definition

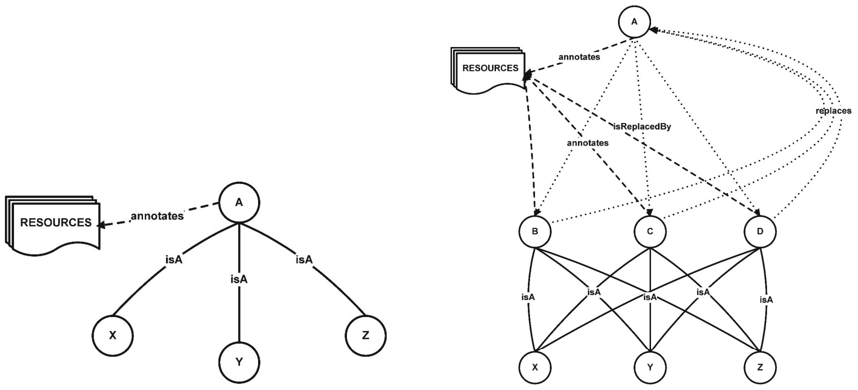


Fig. 3. Concept update by splitting the original concept in two or more new concepts

In this context, it is desirable that in both cases the old definition of the concept "A" is maintained in the ontology, because, as explained earlier, it is necessary that the resources stored in the repository, that has been annotated with the concept "A", still remain retrievable when users look for resources annotated with the concept "A".

Figures 2 and 3 present the two scenarios respectively when a concept is updated by modifying only its definition, and when a concept is split in two or more concepts.

The left parts of the images show the ontology situation before the update, while the right parts show how the ontology evolves after the two possible concept updates. In both cases, the starting point is the scenario in which a concept subsumes ("isA" relationship) other concepts, (the same strategies may also be applied for different type of relations), and it is used to annotated a set of resources stored in the repository.

When a concept is updated (Figure 2), the ontology is modified by inserting a new concept that replaces the old one; however, the old one is maintained in the ontology. The associations between the old concept and its subsumed concepts are moved, and they are placed as subsumptions of the new concept. Two relations are created between the old concept and the new one: "isReplacedBy" and "replaces". The relations "isReplacedBy" and "replaces" are used for retrieval purposes because they permit to navigate through the old concept definitions that are maintained in the ontology in order to preserve the retrieval effectiveness. In fact, the resources annotated with the definition of the old concepts still remain retrievable when users perform queries containing the definition of the old concepts. Moreover, in order to preserve also the efficiency of the platform, it is possible to annotate the resources, previously annotated with the definition of the old concept, with the definition of the new one. This way, the resources are retrievable by using only the definition of the new concept; otherwise, it would be necessary to perform the retrieval operation in different steps.

The second case is the split of the original concept in two or more concepts (Figure 3). In this case, the operations described in the previous case, are repeated for the all concepts that have been created after the split of the original one. Therefore:

– the subsumptions relationships are copied for all new concepts;
– the couple of relations "isReplacedBy" and "replaces" are created for each new concept;
– the actions to preserve the effectiveness and the efficiency of the platform are performed.

Concept Deletion. Similarly to the concept update, the concept deletion has to be managed differently based on the relationships between the deleted concept and the other ones. Two different scenarios are expected: (i) the concept is a middle node of the ontology; and, (ii) the concept is a leaf of the ontology.

The case in which the concept is at the top of the ontology is not managed because it is supposed that the top concept is never been deleted.

Figures 4 and 5 show the two scenarios respectively when the deleted concept is a middle node or a leaf of the ontology.

In the first case, it is supposed that the concept "B" (middle node) is deleted, while in the second case, it is supposed that the deleted concept is "Y" (leaf node). It is desirable that in both cases the concepts "B" and "Y" are somehow maintained in the ontology, because, as explained earlier, it is necessary that the resources stored in the repository, that has been annotated with the concept "B" (or "Y"), still remain retrievable when users look for resources annotated with the concept "B" (or "Y").

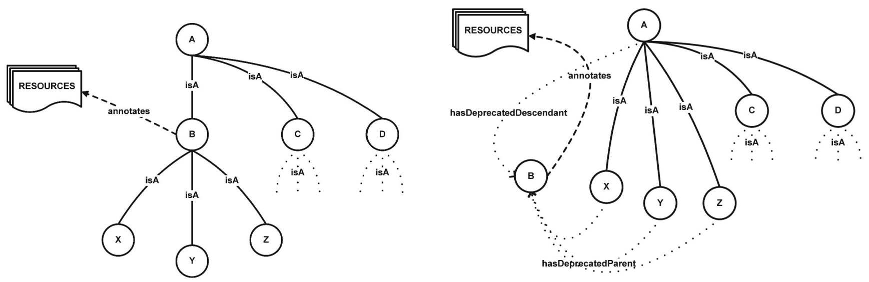


Fig. 4. Deletion of a middle-node concept

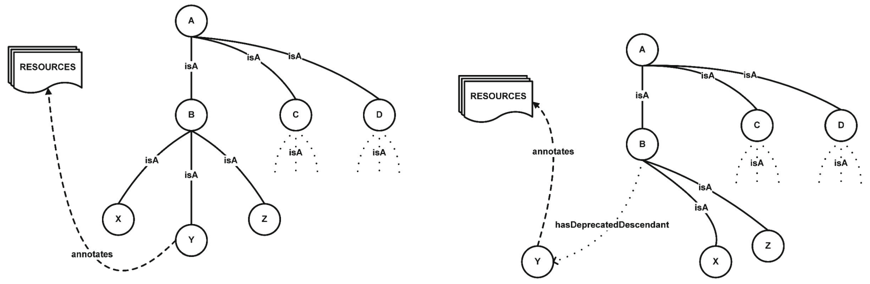


Fig. 5. Deletion of a leaf concept

The left parts of the images show the ontology situation before the deletion, while the right parts show how the ontology evolves after the two possible concept deletions. In both cases, the relation used to associated the concepts is the "isA" relationship, however, the same strategies may also be applied for different type of relations. It is also supposed that the resources are annotated by using only the deleted concept.

When the deleted concept is a middle node (the concept "B" in Figure 4), the subsumed concepts (in this case the concepts "X", "Y", and "Z") are directly associated with the parent concepts of "B" in order to preserve the consistency of the taxonomy. However, there is the possibility that a set of resources is annotated by using only the concept "B". To make these resources retrievable it has been decided to use the "hasDeprecatedParent" and "hasDeprecatedDescendant" relationships. The "hasDeprecatedParent" relationship associates the concepts that was descendants of the deleted concept, with the deleted concept itself. This way, when users perform queries containing the concept "X" and/or "Y" and/or "Z", the platform will also looks for resources that are annotated by using the concept "B". In the same way, when a user perform queries containing the concept "A", the "hasDeprecatedDescendant" relationship is exploited to retrieve the resources that have been annotated by using the concept "B"

In the second case (Figure 5), it is supposed that the deleted concept "Y" is a leaf node of the ontology. In this case, the evolution of the ontology is more simpler than the previous case. The "hasDeprecatedDescendant" relationship is created between the concepts "B" and "Y", and this relationship is exploited in the same way explained for the previous case. Therefore, when users look for resources annotated with the concept "Y", these resources still remain retrievable, while, when users look for resources annotated with the concept "B", the resources annotated with the concept "Y" will be retrieved too.

4.1 Ontology Evolution with Semantic Wikis: A Comparison

In the literature, wiki systems and semantic wikis have been mainly applied to support collaborative creation and sharing of ontological knowledge.

AceWiki [10] was developed in the context of logic verbalization, that is, the effort to verbalize formal logic statements into English statements and vice-versa. AceWiki is based on Attempto Controlled English (ACE), which allows users expressing their

Table 1. Comparison of state-of-the-art modeling wikis

	OWL support	Evolution mechanism
AceWiki	X	
SMW+	X	
IkeWiki	X	
OntoWiki	X	X
MoKi v.2	X	X

knowledge in near natural language (i.e. natural language with some restrictions). *Semantic MediaWiki+* [7], which includes the Halo Extension, is a further extension on Semantic MediaWiki with a focus on enhanced usability for semantic features. Especially, it supports the annotation of whole pages and parts of text, and offers "knowledge gardening" functionalities, that is maintenance scripts at the semantic level, with the aim to detect inconsistent annotations, near-duplicate entries etc.

IkeWiki [22] supports the semantic annotation of pages and semantic links between pages. Annotations are used for context-specific presentation of pages, advanced querying, consistency verification or drawing conclusions. *OntoWiki* [2] seems to focus slightly more directly on the creation of a semantic knowledge base, and offers widgets to edit/author single elements/pages and whole statements (subject, predicate, object).

We have compared the tools mentioned above, together with the versions of MoKi presented in this paper, against the ontology evolution features. The results are displayed in Table 1, where the columns refer to the capability of: (i) representing entities by using OWL syntax; and (ii) providing an ontology evolution algorithm when changes are carried out on pages.

As we can see from the table, all the compared wikis support the use of the OWL language for representing the entities, while only OntoWiki implements and ontology evolution approach. The approach implemented in OntoWiki is based on the EvoPat algorithm [21] that permits to define evolution patterns that may be applied to an ontology for evolving it. This strategy is merely related to the evolution of the ontology, but it does not take into account the necessity of tracking the historical aspect of the ontology. This way, with respect to the ontology evolution approach implemented in MoKi, an external tool is not able to infer which was the previous structure of the ontology.

5 Challenges on the Ontology Quality and Exposure

The use of ontology evolutions mechanisms needs further activities for checking the updated version of the ontologies that are generated as well as for verifying the impact of the carried out changes on the functionalities of third-party tools that, eventually, exploit the ontology for their activities (for instance, resource tagging). In this Section we want to highlight which are the challenges raised by using an ontology evolution mechanism and to verify if the approach implemented in MoKi faces them in an effective way.

5.1 Ontology Quality

The evolution of ontologies implies the possibility of introducing some mistakes into the ontology definition. For instance, after deletion operations, there is the probability that some concepts may remain orphans due to the removal of some relationships, or that there are individuals defined without using concepts. Some examples of imprecisions are: concepts and properties that do not have verbal descriptions, orphaned concepts, concepts that are not used to tag individuals, properties that do not have domain and/or range definitions, presence of non-shared concepts and/or properties, and individuals with no type defined.

In order to avoid the existence of these imprecisions, MoKi implements a service that permits to knowledge engineers to identify the elements that contains some errors.

This service checks the quality of the ontology by performing the following actions in order to avoid the issues listed above:

- Concepts checking: this check consists in verifying that every concept has a verbal description, that there are not orphaned concepts, concepts without individuals, and non-shared concepts.
- Individuals checking: this check consists in verifying that every individual has a type defined.
- Properties checking: this check consists in verifying that every property has a verbal description, that both domain and range are defined in each property, and that there are not non-shared properties.

Besides this automatic check, MoKi implements a further service that helps the knowledge engineer to discover ontology imprecisions. It consists in the use of questionnaires containing list of statements that have been automatically inferred from the domain model defined in the ontology used by the tool. Only statements formulated by using already existing complex concept expressions are displayed. A knowledge engineer analyzes the explanation of each inferred statement in order to understand how it has been inferred and if the statement is correct with respect to which should be the content of the ontology. This way, the knowledge engineer may exploit this service in order to identify possible imprecisions in the ontology and to adjust the ontology by providing/removing elements that cause the imprecisions.

5.2 Ontology Exposure

The challenge of exposing the ontology is a very important aspect not only in the context of the Organic.Lingua project.

Indeed, this challenge impacts on all scenarios in which there are tools that exploit ontologies produced and managed by third-party providers and that they use the concepts defined in the ontology for several activities, like the annotating one.

In these scenarios, the evolution of an ontology might be critical from the point of view of resource retrieval. Indeed, the external tools have to be able to understand how the changes are represented in the ontology in order to update the set of concepts available for the tagging activity, as well as, to provide a mechanism in the search environment for retrieving the resources tagged with the old versions of the ontology.

The current version of MoKi exposes the ontology in Linked Data format to external tools that use it for supporting the tagging activity of the resources that are deployed on the Organic.Lingua portal.

The approach implemented in MoKi that is described in Section 4, provides a set of relationships that permit to reconstruct the changes carried out on the ontology. This way, external tools may exploit these relationships for updating the interface provided to their users, while the search environment may use these relationships for maintaining its effectiveness.

6 Conclusions

In this paper we presented how the challenge of ontology evolution may be faced by using semantic wiki tools. In particular, the use case that we have considered takes into account the scenario in which an ontology is used for tagging resources available on the web. We described how the presented version of MoKi is able to manage the ontology evolution mechanism by providing a set of relationships that permit both to external tools and to search environment, to maintain the knowledge of the changes carried on the ontology.

Acknowledgments. Organic.Lingua is funded under the ICT Support Program of the EU Commission (Grant Agreement Number 270999).

References

1. Andany, J., Léonard, M., Palisser, C.: Management of schema evolution in databases. In: Lohman, G.M., Sernadas, A., Camps, R. (eds.) VLDB, pp. 161–170. Morgan Kaufmann (1991)
2. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 736–749. Springer, Heidelberg (2006)
3. Banerjee, J., Kim, W., Kim, H.-J., Korth, H.F.: Semantics and implementation of schema evolution in object-oriented databases. In: Dayal, U., Traiger, I.L. (eds.) SIGMOD Conference, pp. 311–322. ACM Press (1987)
4. Ghidini, C., Rospocher, M., Serafini, L.: Moki: a wiki-based conceptual modeling tool. In: ISWC 2010 Posters & Demonstrations Track: Collected Abstracts. CEUR Workshop Proceedings (CEUR-WS.org), Shanghai, China, vol. 658, pp. 77–80 (2010)
5. Ghidini, C., Rospocher, M., Serafini, L.: Conceptual modeling in wikis: a reference architecture and a tool. In: The Fourth International Conference on Information, Process, and Knowledge Management, eKNOW 2012 (2012)
6. Haase, P., Stojanovic, L.: Consistent Evolution of Owl Ontologies. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 182–197. Springer, Heidelberg (2005)
7. Hansch, D., Schnurr, H.-P.: Practical applications of semantic mediawiki in commercial environments - case study: semantic-based project management. In: 3rd European Semantic Technology Conference, ESTC 2009 (2009)
8. Klein, M.C.A., Fensel, D., Kiryakov, A., Ognyanov, D.: Ontology Versioning and Change Detection on the Web. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 197–212. Springer, Heidelberg (2002)

9. Krotzsch, M., Vrandecic, D., Volkel, M.: Wikipedia and the semantic web - the missing links. In: Proc. of the 1st Int. Wikimedia Conference, Wikimania 2005 (2005)
10. Kuhn, T.: AceWiki: A Natural and Expressive Semantic Wiki. In: Proceedings of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges (2008)
11. De Leenheer, P., de Moor, A., Meersman, R.: Context dependency management in ontology engineering: A formal approach. J. Data Semantics 8, 26–56 (2007)
12. Lerner, B.S., Habermann, A.N.: Beyond schema evolution to database reorganization. In: OOPSLA/ECOOP, pp. 67–76 (1990)
13. Maedche, A., Motik, B., Stojanovic, L.: Managing multiple and distributed ontologies on the semantic web. VLDB J. 12(4), 286–302 (2003)
14. Miller, R.J., Ioannidis, Y.E., Ramakrishnan, R.: The use of information capacity in schema integration and translation. In: Proceedings of the 19th International Conference on Very Large Data Bases, VLDB (1993)
15. Nguyen, G.T., Rieu, D.: Schema evolution in object-oriented database systems. Data Knowl. Eng. 4(1), 43–67 (1989)
16. OMG. Business process modeling notation, v1.1, www.omg.org/spec/BPMN/1.1/PDF
17. Plessers, P.: An Approach to Web-Based Ontology Evolution. PhD thesis. Department of Computer Science, Vrije Universiteit Brussel, Brussel, Belgium (2006)
18. Plessers, P., De Troyer, O.: Resolving Inconsistencies in Evolving Ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 200–214. Springer, Heidelberg (2006)
19. Proper, H.A., Halpin, T.A.: Conceptual schema optimisation: Database optimisation before sliding down the waterfall. Technical Report 341. Department of Computer Science, University of Queensland, Australia (1998)
20. Ra, Y.-G., Rundensteiner, E.A.: A transparent schema-evolution system based on object-oriented view technology. IEEE Trans. Knowl. Data Eng. 9(4), 600–624 (1997)
21. Rieß, C., Heino, N., Tramp, S., Auer, S.: Evopat - pattern-based evolution and refactoring of rdf knowledge bases. In: International Semantic Web Conference (1), pp. 647–662 (2010)
22. Schaffert, S.: Ikewiki: A semantic wiki for collaborative knowledge management. In: 1st Int. Ws. on Semantic Technologies in Collaborative Applications, STICA 2006 (2006)
23. Simperl, E., Popov, I.O., Bürger, T.: ONTOCOM Revisited: Towards Accurate Cost Predictions for Ontology Development Projects. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 248–262. Springer, Heidelberg (2009)
24. Smith, M.K., Welty, C., McGuinness, D.L.: Owl web ontology language guide. W3C Recommendation, February 10 (2004)
25. Stojanovic, L.: Methods and Tools for Ontology Evolution. PhD thesis. University of Karlshrue, Karlshrue, Germany (2004)