Evolution Management for Interconnected Ontologies

Michel Klein and **Heiner Stuckenschmidt**
Vrije Universiteit Amsterdam
de Boelelaan 1081a, 1081 HV Amsterdam
{michel.klein, heiner}@cs.vu.nl

Abstract

Mappings between ontologies are easily harmed by changes in the ontologies. In this paper we explain a mechanism to define modular ontologies and mappings in a way that allows for local containment of terminological reasoning. We have also developed a change detection and analysis method that predicts the effect of changes on the concept hierarchy. This method determines whether the changes in one ontology affect the reasoning inside other ontologies or not. Together, these mechanisms allow ontologies to evolve without unpredictable effects on other ontologies. In this paper, we also apply these methods in a case study that is undertaken in a EU IST project.

1 Motivation

When mappings are created between ontologies, it is essential that the evolution of ontologies is managed, because a change in one ontology could have extensive effects in other ontologies. This is especially important when ontologies are used as basis for formal reasoning tasks.

To handle this problem, we have developed a mechanism to define modular ontologies and mappings between them that allows for local containment of terminological reasoning [10]. This modularization mechanism makes it possible to perform subsumption reasoning within an ontology without having to access other ontologies. We have also developed a change detection and analysis method that predicts the effect of changes on the concept hierarchy. This method determines whether the changes in one ontology affect the reasoning inside other ontologies or not. Together, these mechanisms allow ontologies to evolve without unpredictable effects on other ontologies.

In this paper, we will show how these methods work in a realistic example. For this we use the case study that is undertaken in the WonderWeb project[1]. We describe the case study and explain the overall approach in the next paragraphs. Section 2 defines the modularization approach in more detail, and shows how we use this to define mappings to the case study ontology. In section 3, we explain the change analysis mechanism and show the results for our example. Finally, in section 4 we conclude with a discussion of open issues and future work.

[1]The WonderWeb project aims at developing scalable infrastructure for the semantic web. For more information, see `http://wonderweb.semanticweb.org/`.

1.1 The WonderWeb Case Study

In the WonderWeb case study, an existing database schema in the Human Resource (HR) domain is used as the basis for an ontology. The first version of the ontology is created by a tool that automatically converts a schema into an ontology [11]. In the next phase, the quality of the ontology is improved by relating this ontology to the foundational ontology DOLCE [5]. First, the HR ontology is aligned with the DOLCE ontology, and in several successive steps the resulting ontology is further refined. During this process, the ontology changes continuously, which causes problems when other ontologies refer to definitions in the evolving ontology. Therefore, in our case study, evolution management is important during the entire life-cycle of the ontology development process.

Besides this DOLCE+HR ontology, we assume that we have another ontology (we call it the *local ontology*) that uses terms and definitions from the evolving DOLCE+HR ontology (the *external ontology*). As an example, we define a very simple ontology about employees (see Figure 1). Our example ontology introduces the concept 'FulltimeEmployee' and defines a superclass 'Employee' and two subclasses 'DepartmentMember' and 'HeadOfDepartment' using terms from the DOLCE+HR ontology.

The specific problem in our case is that the changes in the DOLCE+HR ontology could affect the reasoning in the local ontology. We want to be able to predict whether or not the reasoning in the local ontology is still valid for specific changes in the external ontology.

Changes in DOLCE+HR

The evolution of the DOLCE+HR ontology consisted of several steps, which are prescribed by the DOLCE methodology. Each of these steps involves some typical changes.

In the aligning phase, the concepts and properties in the HR ontology are connected to concepts and properties in the DOLCE ontology via subsumption relations. For example, the concept 'Departments' from the HR ontology is made a subclass of 'Social-Unit' in DOLCE.

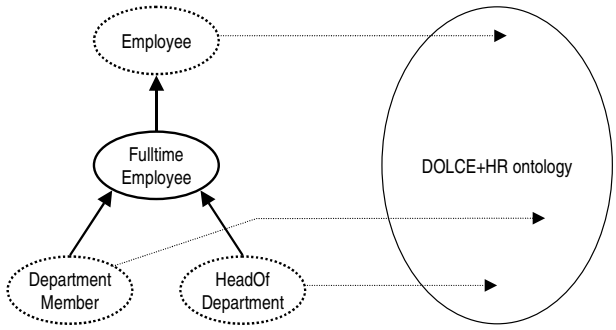

Figure 1: A simple ontology (left) with some concepts (dashed ovals) that are defined using terms from the DOLCE+HR ontology (schematically representation by a large oval).

The refinement step involves a large number of changes. Some property restrictions are added, and some additional concepts and properties are created to define the HR concepts more precisely. For example, the concept 'Administrative-Unit' is introduced as a new subclass of 'Social-Unit', and the concept 'Departments' is made a subclass of it. Also, the range of the property 'email' is restricted from 'Abstract-Region' to its new subclass 'Email'.

In the next step, a number of concepts and properties are renamed to names that better reflect their meaning. For example, 'Departments' is renamed to 'Department' (singular), and the two different variants of the relation 'manager_id' are renamed to 'employee_manager' and 'department_manager'.

In the final step, the tidying-up step, all properties and concepts that are not necessary anymore are removed and transformed into property restrictions. For example, the property 'employee_email' is deleted and replaced by an existential restriction in the class 'Employee' on the property 'abstract_location' to the class 'Email'.

1.2 Approach for Ontology Mappings and Change Management

The main design ideas behind our approach are the following. A detailed description with examples will be given in the next sections.

View-Based Mappings: We adopt the approach of view-based information integration. In particular, ontology modules are connected by conjunctive queries. This way of connecting modules is more expressive than simple one-to-one mappings between concept names but less expressive than the logical language used to describe concepts. We decide to sacrifice a higher expressiveness for the sake of conceptual simplicity and desirable semantic properties such as independence of the ontology langauge used.

Compilation of Implied Knowledge: In order to make local reasoning independent from other modules, we use a knowledge compilation approach. The idea is to compute the result of each mapping query off-line and add the result as an axiom to the ontology module using the result. During reasoning, these axioms replace the query thus enabling local reasoning.

Change Detection and Automatic Update: Once a query has been compiled, the correctness of reasoning can only be guaranteed as long as the concept hierarchy of the queried ontology module does not change. In order to decide whether the compiled axiom is still valid, we propose a change detection mechanism that is based on a taxonomy of ontological changes and their impact of the concept hierarchy.

2 Modular Ontologies

We will now explain the modularization mechanism and the compilation of implied subsumption relations in more detail. In Section 2.3, we show how we use these mechanisms in the case study.

In order to get a general notion of ontological knowledge, we define the general structure of an ontological module and its instantiation independent of a concrete language.

Definition 1 (Ontology Module) *A module is a triple $M = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ where $\mathcal{C}$ is a set of concept definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions. Further, we define the signature of a module $\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ to be a triple $\langle \mathcal{CN}, \mathcal{RN}, \mathcal{ON} \rangle$, where $\mathcal{CN}$ is the set of all names of concepts defined in $\mathcal{C}$, $\mathcal{RN}$ the set of all relation names in $\mathcal{R}$ and $\mathcal{ON}$ the set of all object names occurring in $\mathcal{O}$.*

2.1 Internal and External Definitions

We divide the set of concepts in a module into internally defined concepts $\mathcal{C}_I$ and externally defined concepts $\mathcal{C}_E$ resulting into the following definition of $\mathcal{C}$:

$$\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_E,\ \mathcal{C}_I \cap \mathcal{C}_E = \emptyset$$

Internally defined concepts are specified by using class expressions in the spirit of description logics [1]. We do not require a particular logic to be used.

Definition 2 (Internal Concept Definition) *An internal concept definition is an axiom of one of the following forms*

$$C \sqsubseteq D, C \equiv D$$

where $C \in \mathcal{CN}$ and D is a class expression of the form $f(t_1, \cdots, t_n)$ where the terms t_i are either class names or class expressions and f is an n-ary class building operator.

Besides the standard way of defining concepts, we consider externally defined concepts that are assumed to be equivalent to the result of a query posed to another module in the modular ontology. This way of connecting modules is very much in spirit of view-based information integration which is a standard technique in the area of database systems [6].

Definition 3 (External Concept Definition) *An external concept definition is an axiom of the form $C \equiv M : Q$ where M is a module and Q is an ontology-based query over the signature of M.*

A modular ontology is now simply defined as a set of modules that are connected by external concept definitions. In particular we require that all external definitions are contained in the modular system. Queries over ontological knowledge are defined as conjunctive queries, where the conjuncts are predicates that correspond to classes and relations of an ontology. Furthermore, variables in a query may only be instantiated by constants that correspond to objects in that ontology.

Definition 4 (Ontology-Based Queries) *Let V be a set of variables disjoint from $\mathcal{ON}$ then an ontology-based query Q over a module $M = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ is an expressions of the form $Q(\bar{X}) \leftarrow q_{1_i} \wedge \cdots \wedge q_{m_i}$ where q_i are query terms of the form $x : c$ or $(x, y) : r$ such that $x, y \in V \cup \mathcal{ON}$, $c \in \mathcal{CN}$ and $r \in \mathcal{RN}$ or are of the form $x = o$ where $x \in V$ and $o \in \mathcal{ON}$*[2].

The fact that all conjuncts relate to elements of the ontology allows us to determine the answer to ontology-based queries in terms of instantiations of the query that are logical consequences of the knowledge base.

2.2 Compilation and Local Reasoning

We now turn our attention to the issue of reasoning in modular ontologies. For the sake of simplicity, we only consider the interaction between two modules in order to clarify the basic principles. Furthermore, we assume that only one of the two modules contains externally defined concepts in terms of queries to the other module.

Implied Subsumption As mentioned in the introduction, we are interested in the possibility of performing local reasoning. For the case of ontological reasoning, we focus on the task of deriving implied subsumption relations between concepts within a single module. For the case of internally defined concepts, this can be done using well established reasoning methods [3]. Externally defined concepts, however, cause problems: being defined in terms of a query to the other module, a local reasoning procedure will often fail to recognize an implied subsumption relation between these concepts. Consequently, subsumption between externally defined concepts requires reasoning in the external module as the following theorem shows.

Theorem 1 (Implied Subsumption) *Let E_1 and E_2 be two concepts in module M_i that are externally defined in module M_j by queries Q_1 and Q_2, then $E_1 \sqsubseteq E_2$ if $Q_1 \sqsubseteq Q_2$ in module M_j.*

The result presented above implies the necessity to decide subsumption between conjunctive queries in order to identify implied subsumption relations between externally defined concepts. In order to decide subsumption between queries, we translate them into internally defined concepts in the module they refer to. A corresponding sound and complete translation is described in [7]. Using the resulting concept definition, to which we refer as *query concepts*, we can decide subsumption between externally defined concepts by local reasoning in the external ontology.

Compilation and Integrity We can avoid the need to perform reasoning in external modules each time we perform reasoning in a local module using the idea of knowledge compilation [2]. The idea of compilation is to perform the external reasoning once and add the derived subsumption relations as axioms to the local module. These new axioms can then be used for reasoning instead of the external definitions of concepts. If we want to use the compiled axioms instead of external definitions, we have to make sure that this will not invalidate the correctness of reasoning results. At the time of applying the compilation this is guaranteed by theorem 1, however, integrity cannot be guaranteed over the complete life-cycle of the modular ontology. The problem is, that changes to the external ontology module can invalidate the compiled subsumption relationships. In this case, we have to perform an update of the compiled knowledge.

2.3 Modularization and Local Reasoning in the Case Study

If we now consider the problem statement from the case study, we have a local ontology with a concept hierarchy that is built up by the following explicitly stated subsumption relations (see Figure 1 again):

$$\begin{aligned} FulltimeEmployee &\sqsubseteq Employee \\ DepartmentMember &\sqsubseteq FulltimeEmployee \\ HeadOfDepartment &\sqsubseteq FulltimeEmployee \end{aligned}$$

This ontology introduces 'Full time employee' as a new concept, not present in the case study ontology. Consequently, this concept is only defined in terms of its relation to other concepts in the local ontology.

All other concepts are externally defined in terms of ontology based queries over the case study ontology. The first external definition concerns the concept 'Employee' that is equivalent to the 'Employee' concept in the case study ontology. This can be defined by the following trivial view:

$$Employee \equiv HR : Employee(x)$$

Another concept that is externally defined is the 'Head of Department' concept. We define it to be the set of all instances that are in the range of the 'department manager' relation. The definition of this view given below shows that our approach is flexible enough to define concepts in terms of relations.

$$HeadOfDepartment \equiv HR : \exists y[departmentManager(y, x)]$$

An example for a more complex external concept definition is the concept 'department member' which is defined using a query that consists of three conjuncts, claiming that a department is an employee that is in the has_member relation with a Department.

$$DepartmentMember \equiv HR : \exists y[Department(y) \wedge has\_member(y, x) \wedge Employee(x)]$$

[2]Note that this may include data-type expressions as the type itself is can be considered to be a class, the actual value an instance of that class and the comparison operator a special relation.

Implied subsumption relations

If we now consider logical reasoning about these external definitions, we immediately see that the definition of Employee subsumes the definition of DepartmentMember, as the former occurs as part of the definition of the latter.

$$\models DepartmentMember \quad \sqsubseteq \quad Employee \qquad (1)$$

At a first glance, there is no relation between the definition of a Head of Department and the other two statements as it does not use any of the concept- or relation names. However, when we use the background knowledge provided by the external ontology we can derive some implied subsumption relations. The reasoning is as follows. Because the range of the department_manger is set to 'Department' and the domain to 'Manager', the definition of HeadofDepartment is equivalent to:

$$\exists y[Department(y) \wedge department\_manager(y, x) \wedge Manager(x)]$$

As we further know that Manager is a subclass of Employee and department_manager is a sub-relation of has_member, we can derive the following subsumption relation between the externally defined concepts:

$$\models HeadOfDepartment \quad \sqsubseteq \quad Employee \qquad (2)$$
$$\models HeadOfDepartment \quad \sqsubseteq \quad DepartmentMember \qquad (3)$$

When the relations 1–3 are added to the local ontology, it possible to do subsumption reasoning without having to access the DOLCE+HR ontology anymore.

3 Change Detection and Analysis

The changes in the DOLCE+HR ontology could invalidate the local reasoning. In principle, testing the integrity of the mappings might be very costly as it requires reasoning within the external ontology. In order to avoid this, we propose a heuristic change detection procedure that analyzes changes with respect to their impact on compiled subsumption relations, i.e. relations 1–3 from the previous section. This is a three-steps procedure: 1) find out what the differences are between two distinct versions of the ontology, 2) characterize the effect of these changes on individual concepts, and 3) determine the impact of changes of individual concepts on the compiled subsumption relations. The next sections describe these steps.

3.1 Finding Changes

To find changes in ontologies, we have developed a mechanism and a tool to compare ontologies. This change detection mechanism is described in [8]. The algorithm that we developed works for all ontology languages that can be represented in the RDF data model [9], including RDF Schema and OWL. For each changed definition, it produces a list of change operations that are necessary to transform the old version into the new version.

To standardize the description of changes, we have developed an ontology of all possible change operations for an OWL-lite ontology. An actual description of a change between two versions of an ontology can be seen as an instantiation of the ontology of change operations. The change ontology is extendable to other knowledge models. We have chosen the OWL-Lite model because of its simplicity and the central role of OWL in the WonderWeb project. A snapshot of the change ontology can be found online.[3]

Apart from *atomic change operations* — like add range restriction or delete subclass relation — our change ontology also contains some *complex change operations*, which consist of multiple atomic operations and/or incorporate some additional knowledge. The complex changes are often more useful to specify effects than the basic changes. For example, for operations like concept moved down, or range restricted, we can specify the effect more accurately than for the atomic operations subclass relation changed and domain modified.

The case study ontology in our example is expressed in OWL-Lite, which is based on RDF. Therefore, we can use rule-based change detection mechanism. If we look at the changes in the definition of 'Departments', we see that three things happened:

- the comment is reformulated,
- the superclass is changed from 'Social-Unit' to 'Administrative-Unit', and
- there is a property restriction added for 'temporary-component-of' to the class 'Organization'.

This results in three change operations: 1) superclass changed (from 'Social Unit' to 'Administrative-Unit', 2) comment changed, and 3) property restriction added.

3.2 Characterizing Changes

Now we have detected the change operations that are required to transform the old version of the ontology into the new version, we look at the effect of the change operations on individual concepts. Assuming that C represents the concepts under consideration before and C' the concept after the change there are four ways in which the old version C may relate to the new version C':

1. the meaning of concept is not changed: $C \equiv C'$ (e.g. because the change was in another part of the ontology, or because it was only syntactical);
2. the meaning of a concept is changed in such a way that concept becomes more general: $C \sqsubseteq C'$
3. the meaning of a concept is changed in such a way that concept becomes more specific: $C' \sqsubseteq C$
4. the meaning of a concept is changed in such a way that there is no subsumption relationship between C and C'.

We want to know what the effect of specific operations on the interpretation of a concept is (i.e. whether it becomes more general or more specific). As our goal is to determine the integrity of mappings without having to do classification, we describe what theoretically could happen to a concept as result of a modification in the ontology. To do so, we have

[3] http://ontoview.org/changes/1/3/

determined the effect for all possible change operations that we distinguish in the 'finding changes' phase.

Table 1 contains some examples of operations and their effect on the classification of concepts. The table only shows a few examples, although our full ontology of change operations contains around 120 operations. This number is still growing as we define new complex changes.

	Operation	**Effect on C**
1	Attach a slot to class C	Specialized
2	*Complex:* Change the superclass of class C to a class lower in the hierarchy	Specialized
3	*Complex:* Restrict the range of a slot S (*effect specified for all classes C that have a slot restriction with S*)	Specialized
4	Remove a superclass relation of a class C	Generalized
5	Change the class definition of C from primitive to defined	Generalized
6	Add a class definition A	Unknown
7	*Complex:* Add a (not further specified) subclass A of C	No effect

Table 1: Some ontology change operations and their effect on the classification of concepts in the hierarchy.

If we apply this to our example, we can only give a useful characterization of the effect to some of the concepts. For example, the concept 'Departments', underwent several changes during the whole process: its superclass has changed to a subclass of the original superclass (change 2 in Table 1) but there are also some property restrictions removed. Both changes have an opposite effect. As a result, we have to characterize the effect of the change as "Unknown". On the contrary, the effect on the relation 'department_manager', is clear: the relation is renamed from 'manager_id' — which has no conceptual effect — and the range is changed from 'Employee' to 'Manager'. Because 'Manager' is a subclass of 'Employee', this change makes it more specific (change 3 in Table 1).

3.3 Update Management

With the elements that we described in this section, we now have a complete procedure to determine whether compiled knowledge in other modules is still valid, and thus whether the mappings are still usable. The complete procedure is as follows:

1. create a list of concepts and relations that are part of the "subsuming" query of any compiled axiom;
2. create another list of concepts and relations that are part of the "subsumed" query of any compiled axiom;
3. achieve the modifications that are performed in the external ontology;
4. use the modifications to determine the effect on the interpretation of the concepts and relations.
5. check whether there are concepts or relations in the first, "subsuming", list that became more specific, or concepts or relations in the second, "subsumed", list that became more general, or concepts or relations in any of the lists with an unknown effect; if not, the integrity of the mapping is preserved.

All the steps can be automated. The tool that we mentioned in the previous section currently helps with steps 3 and 4. It detects the changes between two versions and produces a list op change operations.

We can now use this procedure to check whether the implied subsumption relations in our case study are still valid. For the sake of simplicity, we restrict us here to relation 3:

$$\models HeadOfDepartment \sqsubseteq DepartmentMember$$

For this compiled axiom, the list of 'subsuming' concepts and relations would contain 'Department', 'has_member', and 'Employee', while the list of subsumed concepts and relations would be 'Department', 'department_manager', and 'Manager'.

We will now illustrate that the conclusions of the procedure are correct by studying the impact of changes mentioned in the problem statement.

Example 1: The Employee Concept The first change we observed is the removal of properties from the Employee concept. Our rules tell that this change makes the new version more general compared to its old version:

$$Employee \sqsubseteq Employee'$$

According to our procedure, this shouldn't be a problem because Employee is in the 'subsuming list'.

When we analyze this change, we see that it has an impact on the definition of the concept DepartmentMember as it enlarges the set of objects allowed to take the first place in the has_member relation. This leads to a new definition of $DepartmentMember'$ with $DepartmentMember \sqsubseteq DepartmentMember'$. As DepartmentMember was already more general than HeadOfDepartment and the Employee concept is not used in the definition of the latter the implied subsumption relation indeed still holds.

Example 2: The department_manager Relation The second example, we have to deal with a change affecting a relation that is used in an external definition. The relation department_manager is specialized by restricting its range to a more specific concept making it a subrelation of its previous version:

$$department\_manager \sqsupseteq department\_manager'$$

Again, this is harmless according to our procedure, as department_manager is in the 'subsumed list'.

The analysis shows that this change has an impact on the definition of the concept HeadOfDepartment as it restricts the allowed objects to the more specific Class Manager. The new definition $HeadOfDepartment'$ is more specific that the old one: $HeadOfDepartment' \sqsubseteq HeadOfDepartment$.

As the old version was already more specific than the definition of DepartmentMember and the department_manager relation is not used in the definition of the latter the implied subsumption is indeed still valid.

Example 3: The Department Concept The different changes of the definition of the department concept left us with no clear idea of the relation between the old and the new version. In this specific case, however, we can still make assertions about the impact on implied subsumption relations. The reason is that the concept occurs in both definitions. Moreover, it plays the same role, namely restricting the domain of the relation that connects an organizational unit with the set of objects that make up the externally defined concept. As a consequence, the changes have the same impact on both definitions thus not invalidating the implied subsumption relation. *In summary, an implied subsumption relation is still valid if the changed concept occurs in and plays the same role in both definitions involved.*

4 Discussion

In this paper we discussed the problem of ontology evolution in situations where mappings between ontologies existed. We presented two main contributions towards a better understanding and management of dependencies in the light of changes to an ontology.

- We presented a formal model for describing dependencies between different ontologies. We proposed conjunctive queries for defining concept using elements from another ontology and presented a model-based semantics in the spirit of distributed description logics that provides us with a notion of logical consequence across different ontologies. This clear semantic account of dependence makes it possible to study the impact of changes on a semantic level.
- We described a method for detecting changes in an ontology and for assessing their impact. The main feature of this method is the derivation of conceptual changes from purely syntactic criteria. These conceptual changes in turn provide input for a semantical analysis of the effect on dependent ontologies, in particular on the validity of implied subsumption relations.

The effect analysis procedure that we have proposed uses quite coarse-grained heuristics. As a result, it often concludes that a validity of a subsumption relation cannot be guaranteed, while it is in fact still valid. In order to be able to provide more precise answers we will have to develop a more formal characterization of changes like it has been done in the area of schema evolution for database systems [4]. Based on such a formal characterization, we have to investigate conditions under which implied knowledge is still valid in a more generic way.

References

[1] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] M. Cadoli and F. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[3] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[4] E. Franconi, F. Grandi, and F. Mandreoli. A sematic approach to schema evolution and versioning in object-oriented databases. In *Proceesing of CL 2000*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 1048–1062. Springer Verlag, 2000.

[5] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, volume 2473 of *Lecture Notes in Computer Science*, page 166 ff, Sigüenza, Spain, Oct. 1–4, 2002.

[6] A. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[7] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.

[8] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology versioning and change detection on the web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, number 2473 in LNCS, page 197 ff, Sigüenza, Spain, Oct. 1–4, 2002.

[9] O. Lassila and R. R. Swick. Resource Description Framework (RDF): Model and Syntax Specification. Recommendation, World Wide Web Consortium, Feb. 1999. See http://www.w3.org/TR/REC-rdf-syntax/.

[10] H. Stuckenschmidt and M. Klein. Integrity and change in modular ontologies. In *Proceedingso of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, Aug. 2003.

[11] R. Volz, D. Oberle, S. Staab, and R. Studer. Ontolift prototype. Deliverable D11, EU/IST Project WonderWeb, 2002.