

Semantic Annotation in e-Learning Systems based on Evolving Ontologies

Delia ROGOZAN, Gilbert PAQUETTE

Télé-université of Québec

4750, avenue Henri-Julien, Montreal (QC), H2T 3E4, CANADA

drogozan, gpaquett@liceq.telug.quebec.ca

Abstract. In this paper we discuss an approach for annotating e-Learning resources based on skill/performance and learning-domain ontologies and we propose a framework for managing ontology changes and their effects on the semantic annotation of resources.

Introduction

The research presented in this paper follows the initial ideas developed in ([1]) regarding the annotation of resources in e-Learning systems, and the ideas developed in ([2]) regarding the ontology evolution. In Section 1 we discuss an approach that uses skills/performance and learning-domain ontologies to annotate resources in a standard manner. In Section 2 we propose a framework for managing ontology changes in an appropriate manner; that is preserving the integrity of ontology-based annotation of resources after ontology evolution.

1. Enriching the Knowledge-based Annotation with a Competency Level

According to ([3]), a competency is the statement of a relationship among a knowledge, a skill and a performance degree: e.g. “Instantiate (*skill*) the agricultural practices (*knowledge*) in an expert manner (*performance*)”. In order to annotate resources according to their competencies, we propose an RDF-based technique that uses two ontologies: a *learning-domain ontology* specifying a consensual view of a subject-matter ([4]) and a *skill/performance ontology* specifying the generic mastery levels that may be applied to any knowledge element from a learning domain ([3]).

The skill/performance ontology has two root classes: `Skill` and `Performance`. The `Skill` class contains four principal classes (`Receive`, `Reproduce`, `Create` and `Self_Manage`), each of them having more subclasses (e.g. `Instantiate`, `Apply`). The `Performance` class refers to the performance degree (e.g. `Familiarized`, `Autonomous` OR `Expert`) of a skill when it is applied on a knowledge. In order to describe the relations between these classes, three properties have been defined: (1) the `appliedTo` property, which asserts that any `Skill` subclass may be applied to a class from a learning-domain ontology; (2) the `hasCompetency` property, which states that anything may have a competency whose value is a `Skill` applied to a knowledge; (3) the `hasPerformance` property, which links a `Skill` to a `Performance` degree.

1.1 Standard Model for Binding Knowledge and Competencies to Resources

Figure 1 (a) shows the RDF-based model we propose for annotating resources. The `PedagogicalResource` identifies a specific document, tool, operation or actor. According to Dublin Core best practices, we link the `dc:subject` property to the entry in a learning-domain ontology, namely one of its `Knowledge` element.

The `sk:hasCompetency` property connects a `PedagogicalResource` to one of its competencies defined by a `Skill` that (1) is applied to a `Knowledge` and (2) has a specific `Performance`. Therefore, the value of the `hasCompetency` property is an element from the `Skill` class-hierarchy, the value of the `appliedTo` property is an element from a learning-domain ontology and the value of the `hasPerformance` property specifies an element from the `Performance` class-hierarchy of the skills/performance ontology.

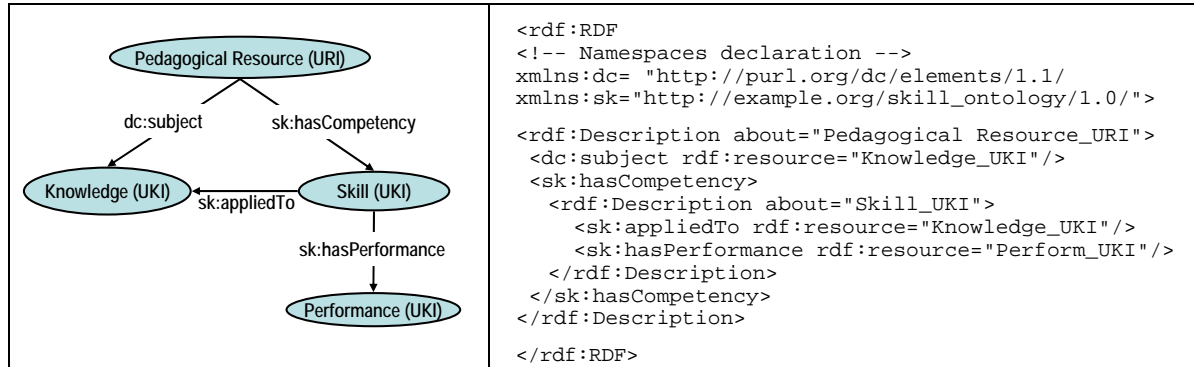


Fig. 1(a) RDF model of an annotated resource; **Fig. 1(b)** Formalization of the related RDF model

Figure 1(b) shows the formalization of the RDF-based model using the RDF syntax ([5]). The namespace declaration (i.e. `xmlns`) provides a means to locate from where the elements with the related prefix are. The `rdf:Description` statement declares a resource, its properties and their respective values.

URI (*Uniform Resource Identifier*) ([6]) provides a standard mechanism for resource identification. In order to clearly distinguish between ontology terms and other type of resources (e.g. documents, services), we introduced the expression UKI (*Uniform Knowledge Identifier*). A UKI is as a compact sequence of characters, which explicitly and completely specifies a link to a unique element, in a specific ontology. An example is shown in Figure 2.

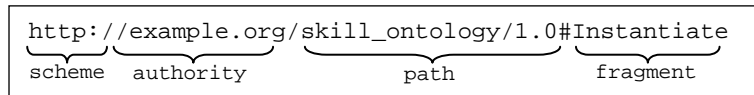


Fig. 2. UKI Example. The path includes the ontology name (i.e. `skill_ontology`) and version (i.e. `1.0`). The fragment specifies a unique ontology entity (i.e. `Instantiate`)

In this section, we have described an RDF-based technique to annotate resources (i.e. documents, tools or human resources) in a modular approach, according to their competencies (i.e. knowledge + skill/performance). Seeing that we can therefore search from resources according to their knowledge (e.g. documents describing the `AgriculturalPractices`) and/or competencies (e.g. actors being able to `Instantiate` the `AgriculturalPractices` in an `Expert` manner), this modular annotation will enable search-agents to suggest resources adapted to users' knowledge and competency.

2. Ontology Evolution in the Educational Semantic Web

Ontologies often evolve over time ([2, 7]). Changes in ontology domain, adaptations to different learning tasks, or changes in conceptualization require the modification of learning-domain ontologies. We define the *ontology evolution* as the process through which a former ontology version (V_N) is changed into a new version (V_{N+1}), while preserving the integrity of the ontology-based annotation of resources. This is an essential issue since changes such as the removal, merge or splitting of classes and properties may hinder the access to resources that

were annotated with terms from corresponding ontology ([8, 7]). Hence, we propose a framework that supports the consistent ontology evolution in the Semantic Web context.

2.1 Framework for Managing Ontology Changes

The conceptual framework we propose to manage ontology changes is depicted in Figure 3. The **Ontology Workbench** allows the elementary (e.g. Add, Delete) and complex (e.g. Merge, Split) change editing.

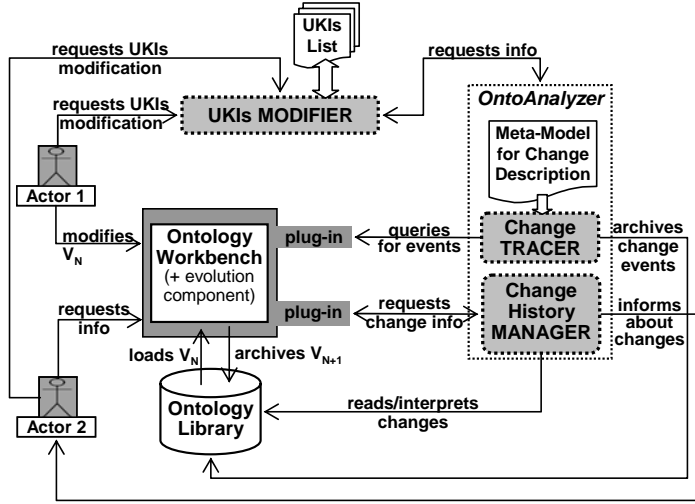


Fig. 3. Ontology Change Management Framework

The main functionalities of the **Change TRACER** are: (1) to track changes during ontology evolution and (2) to archive them to allow later retrieval.

At any time, the **Change History MANAGER** can provide users and software agents with information about: (1) the changes applied to V_N to obtain V_{N+1} and (2) the effects of these changes on the semantic annotation of resources.

The Change Tracer and the Change History Manager form together the **OntoAnalyzer System** we described in ([9]).

The goal of the **UKIsModifier** is to preserve the access to semantically annotated resources after the ontology evolution. The next section gives an account of this function.

2.2 Aim of Preserving the Integrity of Semantic Annotation based on Evolving Ontologies

Nowadays, a major issue emerges in the ontology research: “the interlinkage between objects and evolving ontologies need to be managed” ([10]). Consequently, we propose the **UKIsModifier** whose major function is to maintain the integrity of the ontology-based annotation of resources after the ontology evolution. To illustrate our proposal, let us consider a simple example of an ontology evolution (see Figure 4).

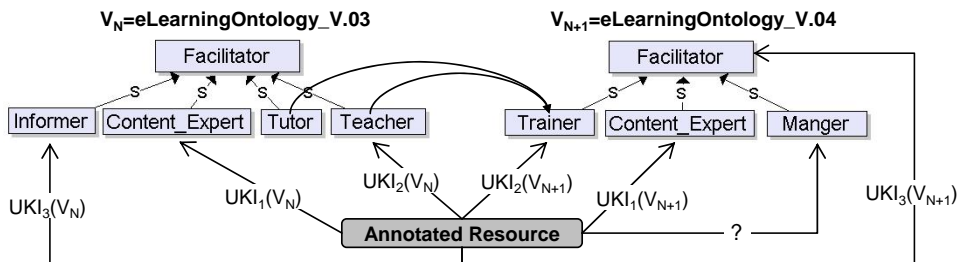


Fig. 4. Ontology changes: (1) the class *Informer* was deleted, (2) the classes *Tutor* and *Teacher* were merged into *Trainer* in the fourth version and (3) the class *Manager* was added in the fourth version.

In order to preserve the access to and the interpretation of annotated resources, the **UKIsModifier** modifies the UKIs associated to resources according to the changes applied to V_N to obtain V_{N+1} (see Table 1 and Table 2). These changes are previously identified by **OntoAnalyzer**, which also discovers the logical relations that may exist among the entities belonging to V_N and those belonging to V_{N+1} . For example: the class *Facilitator* from the fourth version includes the meaning of its subclass *Informer* from the third version; the class

Trainer includes the meaning of merged classes Tutor and Teacher belonging to the previous ontology version.

Table 1. UKIs modification for *full compatible* and *backward compatible* changes for which the interpretation of annotated resources via V_N is the same as when using V_{N+1}

Change Example	UKIsModifier function
Add an instance to the class Content_Expert	Modifies ONLY the ontology path (i.e. version name and/number) in the UKIs $UKI_1(V_N) = http://www.example.org/eLearningOntology/3.0\# Content_Ex$ $\hookrightarrow UKI_1(V_{N+1}) = http://www.example.org/eLearningOntology/4.0\#Content_Ex$
Add class Manager to class Facilitator	Assists the users in defining new UKIs for the new knowledge (e.g. highlights the new class "Manager" and supports users to bind it to a resource).

Table 2. UKIs modification for *incompatibles* changes for which the interpretation of annotated resources is invalid or the access to them is hindered via V_{N+1}

Change Example	UKIsModifier function
Modifies the ontology path (i.e. version name and/or number) AND the fragment identifier in the UKIs	
Merge classes Tutor and Teacher into Trainer	$UKI_2(V_N) = http://www.example.org/eLearningOntology/3.0\# Teacher$ $\hookrightarrow UKI_2(V_{N+1}) = http://www.example.org/eLearningOntology/4.0\#Trainer$
Delete class Informer from the subclasses of Facilitator	$UKI_3(V_N) = http://www.example.org/eLearningOntology/3.0\# Informer$ $\hookrightarrow UKI_3(V_{N+1}) = http://www.example.org/eLearningOntology/4.0\#Facilitator$

3. Conclusion and Future Work

In this article we have presented an RDF-based model for annotating resources in a more fine-grained manner; that is annotating resources by their knowledge or competencies. We have also described a framework for managing ontology changes and for maintaining the integrity of semantic annotation of resources after the ontology evolution. The goal of our future work is to complete the development of the UKIsModifier system and to implement it within the TELOS system being built in the LORNET (Learning Object Repository Network) project.

References

1. G. Paquette and I. Rosca, *Ontology-based indexing for selecting actors, operations and resources in an on-line learning system*, SW-EL'04: Semantic Web for E-Learning, 2004.
2. D. Rogozan, G. Paquette and I. Rosca, *Gestion de l'évolution de l'ontologie utilisée comme référentiel sémantique dans un environnement de téléapprentissage*, TICE International Symposium, 2004, 245-252.
3. G. Paquette, *Instructional engineering for network-based learning*, Wiley-Pfeiffer, 2003.
4. J. Breuker, A. Muntjewerff and B. Bredewej, *Ontological modelling for designing educational systems*, AI-ED'99 Workshop on Ontologies for Educational Systems, IOS Press, 1999.
5. W3C Recommendation, "RDF/XML syntax specification," 2004, <http://www.w3.org/TR/rdf-syntax-grammar/>.
6. T. Berners-Lee, "Uniform resource identifier (URI): Generic syntax", 2005, <http://www.gbiv.com/protocols/uri/rfc/rfc3986.html>.
7. M. Klein and D. Fensel, *Ontology versioning for the semantic web*, International Semantic Web Working Symposium (SWWS), 2001.
8. J. Heflin and J. Hendler, *Dynamic ontology on the web*, 17th National Conference on AI, 2000.
9. D. Rogozan and G. Paquette, *Managing ontology changes on the semantic web*, The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), (submitted paper), 2005.
10. S. Staab, A. Maedche and S. Handschuh, *An annotation framework for the semantic web*, The First International Workshop on MultiMedia Annotation, 2001.