

Editing Learning Object Metadata: Schema Driven Input of RDF Metadata with the OLR3-Editor

Tobias Kunze, Jan Brase, Wolfgang Nejdl

Institut für Informationssysteme – Wissensbasierte Systeme,
Appelstrasse 4, 30167 Hannover, Germany
and
Learning Lab Lower Saxony (L3S),
Deutscher Pavillon, Expo Plaza 1, 30539 Hannover, Germany

{kunze, brase, nejdl}@learninglab.de

Abstract. The Institute for Knowledge Based Systems has developed several learning repositories in the last 5 years based on explicit use of metadata and (in the last two years) RDF. Using standard and self-defined metadata schemas to represent the structure and meta-information of learning objects, we decided to make our latest generation of repositories (OLR3) flexible for all kinds of schemes. To provide maximum flexibility we developed a completely schema-driven and browser-based editor in which the author can choose the type of metadata using any kind of RDF-schema available on the WWW to annotate learning resources in the OLR3 repository. In this paper we present the design, interface and implementation of this editor together with the necessary background information about metadata-standards and our learning repository, we build the editor upon.

1 Introduction

In our group we have developed several learning repositories during the last 5 years. In the latest generation, the Open Learning Repository (OLR) a course was represented completely by RDF-metadata [WOLF2001]. Using standard and self-defined metadata schemas to represent the structure and meta-information of different learning objects, we decided to make our latest version of the OLR (OLR3) flexible for all kinds of schemes and integrated a browser-based and schema-driven editor where the author can choose metadata according to any RDF-schema available on the WWW to annotate learning resources.

Chapter 2 includes the necessary background information about metadata-standards and the OLR3, followed by a presentation of this editor in Chapter 3. Chapter 4 compares the OLR3 editor with other schema editors.

2 Background

2.1 Metadata standards

One of the most common metadata schemes in the web today is “Dublin Core” (DC) by the DCMI. The Dublin Core Metadata Initiative (DCMI) is an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources, that enable more intelligent information discovery systems.

Each Dublin Core element is defined using a set of 15 attributes from the ISO/IEC 11179 standard for the description of data elements, including for example: Title, Identifier, Language, Comment.

Where “Simple Dublin Core” uses only the elements from the Dublin Core metadata set as simple attribute-value-pairs, “Qualified Dublin Core” employs additional qualifiers to further refine the meaning of a resource. The DCMI recommends a set of qualifiers simply called “Dublin Core Qualifiers” (DCQ), these include for example Name, Label, Definition or Comment as alternative qualifiers, that refine the Title element.

Since Dublin Core is designed as metadata for describing *any kind* of resource, it pays no heed to the specific needs we encounter in describing learning resources.

The “Learning Objects Metadata Standard” (LOM) [LTSC2002] by the Learning Technology Standards Committee (LTSC) of the IEEE was established as an extension of Dublin Core. Each learning object can now be described using a set of more than 70 attributes divided into nine categories responsible for general, technical, or even educational aspects of the resource.

Work on the LOM schema has started in 1998, the current version is 6.3. Once the standard has been accepted, which hopefully will happen this year, it will be LOM 1.0.

2.2 XML & RDF Bindings

While the LOM standard defines the structure of a metadata instance, it does not define how a learning technology system will represent or use a metadata instance for a learning object. A great amount of work in that area has been done by the IMS Global Learning Consortium, who have developed a XML binding and a RDF binding of LOM, the latter with cooperation from our institute under guidance of Michael Nilsson from the SweLL (Swedish Learning Lab) [NILSSON2001].

While the structure of an XML Instance is the result of choosing the most convenient syntax, and creating the element hierarchy that best matches the structure of the data, in RDF the data model is not simply syntactic, but has semantic consequences.

As a consequence of the nature of RDF, we cannot expect the RDF binding to fulfill the same purpose as the XML binding. The XML Binding defines an exchange format for metadata. The metadata might be contained in a database and an XML representation is usually generated on demand, for export to other tools and

environments. Thus, an XML metadata record is a self-contained entity with a well-defined structure.

In contrast, with an RDF representation, the metadata is not really self-contained, but rather part of a global network of information, where anyone has the capability of adding metadata to any resource. It is also not the case that all metadata for one resource needs to be contained in a single RDF document. Translations might be administrated separately, and different categories of metadata might be separated.

Thus, we expect to see much richer structures on many levels in an RDF representation than in the corresponding XML binding instance. In XML, there is seldom a natural way to reuse other metadata standards. By contrast, RDF is designed in a way that leads to naturally reusable constructs. So, the LOM RDF binding is directly compatible with Dublin Core, and indeed uses Dublin Core elements directly instead of defining new LOM elements for these DC properties

2.3 Beyond LOM and SCORM

Regardless of the very useful work that has been done in developing the LOM standard, the standard still fails to specify important educational aspects of learning resources. As a lot of work in this area is done by the Learning Lab Lower Saxony (L3S) [ALLERT2001], it is important for us to be open for new standards as well. Similar remarks are valid for the copyright and law-related attributes in the current LOM standard. Thus, obviously, tools for working with learning objects and learning object metadata definitively need to be flexible and able to use different kinds of schemas, some not yet specified.

2.4 OLR3

Our Open Learning Repository, Version 3, the OLR3 system, is implemented in Java and works as a JavaServlet, running on an Enhydra [ENHYDRA] Application Server (open source software). It is connected to an Oracle Database via JDBC, which is used to store the metadata entered by course authors and students. RDF schemes, needed for either the annotation of metadata or the import of externally prepared metadata, can come from anywhere in the internet.

The database does only hold the metadata annotated by the user or imported from RDF files, that are prepared to act as a data source for the courses metadata. Those RDF statements are defined from arbitrary RDF schemes, which are used as a guideline to the user whenever he wants to add new metadata.

The central part of the system is a storage called "StatementPool". It holds all metadata that is known to the system at runtime. When an author starts working on a course, the pool is filled with the already existing data about that course from the database, and all statements from the used RDF schemes.

Any referenced RDF schema will be parsed using the SiRPAC RDF parser [SIRPAC], whereas imported RDF files are parsed by a VRP RDF parser [VRP], which provides semantical checks against given RDF schema rules.

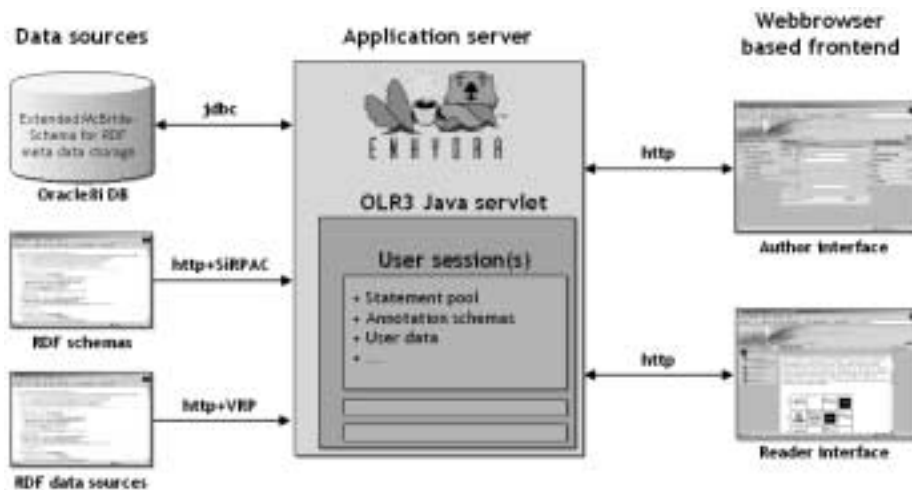


Fig. 1. The architecture of the OLR3 system

3 OLR3 Schema Editor

3.1 General Design Considerations

OLR3 provides a web-browser based metadata editor/viewer and provides two major user interfaces: One for readers with a more graphically oriented view and only minor functions for manipulation of the underlying metadata. The other one designed for authors to provide a schema-driven and browser-based metadata editor with flexible binding to different RDF schemes.

3.2 Reader Interface Layout

Readers of courses using this interface can navigate through an existing course structure, displayed as a tree and extended by additional, metadata-defined images for better understanding. Within that tree they may select single course elements, whose content will be shown in the center of the screen. A specific engine prepares and filters the element metadata (“content”), and displays it in a certain manner - e.g. show inline links to linked web pages, or display the course elements title at the top of the content screen.

The reader interface also offer the reader the possibility of making minor additions to the metadata of a selected course element by providing functions like “add comment”, “add bookmark”, etc. All those additions can be made private or public to other course readers.

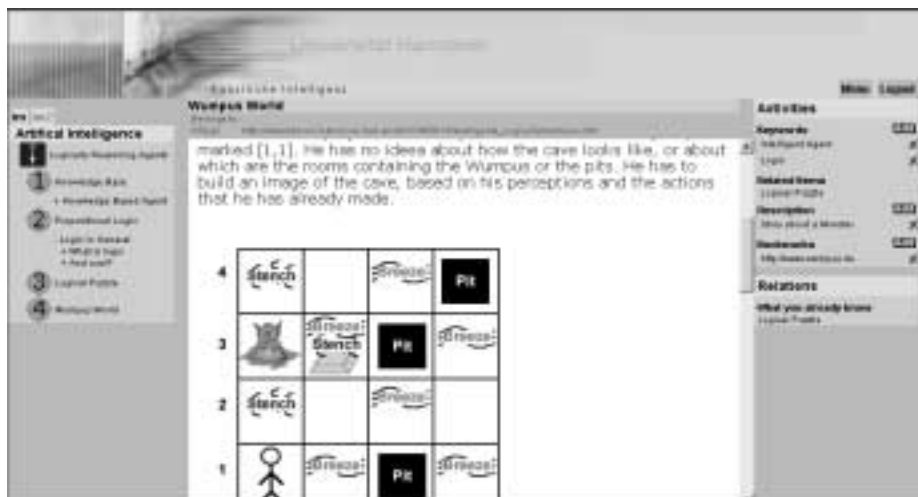


Fig. 2. The user interface for readers

3.3 Author Interface Layout

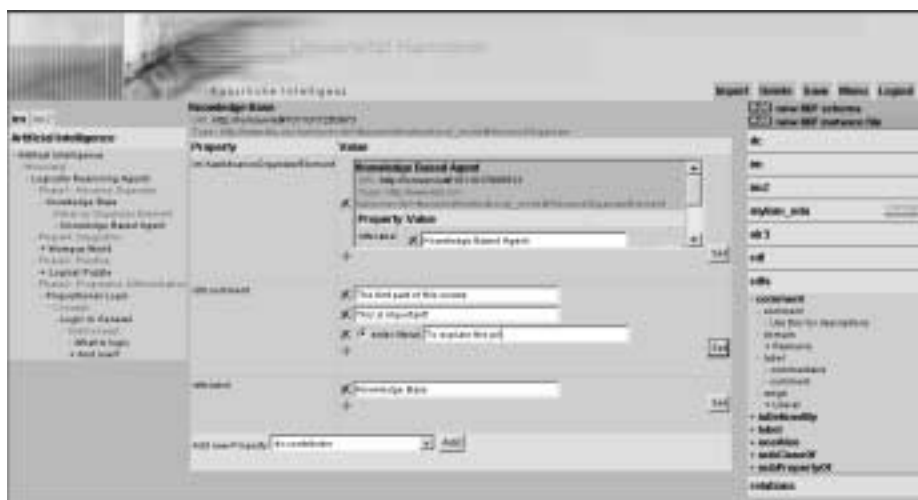


Fig. 3. The user interface for authors with the editor in the center

The second interface is the actual metadata editor which is intended for course authors, who can navigate through the structure tree of a course and select any sub-element. All existing editable metadata for this element are shown in the center of the screen, and the user can choose from a set of existing RDF properties, to add to the metadata, or to modify the existing data. The author can also bind RDF schemes (e.g. DC, DCQ, LOM) from anywhere in the Internet to extend the set of available

properties for annotation, or unbind RDF schemes, that are not needed anymore. A “toolbar” holds those bound RDF schemes and offers the possibility of navigating through their structure by displaying an expandable tree view of any available property.

3.4 Structural Course Viewer

The OLR3 user interface offers an additional view on the metadata: It is used for navigation through the existing metadata of a given course, and is called the “Course Viewer”. For one selected RDF schema, it will display all system-wide instances of those schema classes in a tree view. Any such instance is represented by a node in that tree, beginning at a defined root (the course class) and forming a new branch at every occurrence of a property with a schema-class-instance. Initially, the tree is collapsed, but one can expand any desired node that contains sub elements. Additionally, in the author mode one can select any item in the tree for further annotation. The structure does not necessarily need to form a tree – it may also contain loops, which are then unfolded as a tree.

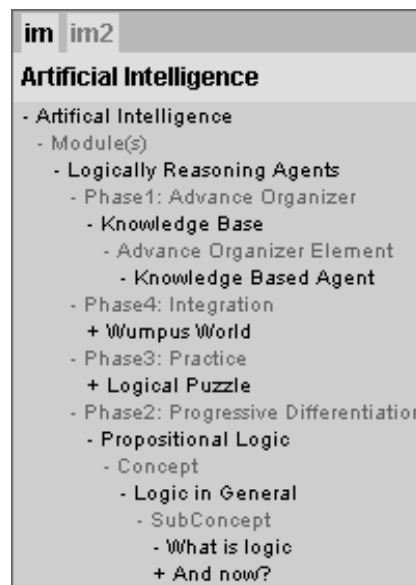


Fig. 4. The Course Viewer for the RDF schema “im”, and an alternative viewer for “im2” in the background

As described, the elements and the structure displayed by the Course Viewer are entirely RDF schema driven, with only one element statically implemented: the root, which is defined to be represented by the class “Course”. The category labels within the viewer depend on the class names defined by the underlying RDF schema, whereas the instance labels are given by the `rdfs:label` attribute of each instance. In the reader mode, the category labels are replaced by small icons.

The order of elements within the Course Viewer is arbitrary in the author mode, but can be specified by additional sequence definitions within the metadata to show a desired order in the reader mode.

OLR also enables users to have several Course Viewers at the same time. Each viewer may base on a different RDF scheme and thus provide a different view on a given course. The user can then choose between these several viewers to highlight the favored one.

3.5 Structural Schema Viewer



Fig. 5. The Toolbar here contains several schemes, with “rdfs” expanded and showing the subitems for the property “comment”

The “Toolbar” within the author interface does not only hold all system-bound RDF schemes, but also provides the possibility of investigating the structure of any included schema. One can bind or unbind RDF schemes given by their URI. Then each bound schema may be expanded to show all included properties. A mouse-click on a property or subitem exposes all its attributes with their corresponding values.

Thus, one can navigate through all elements defined by the RDF schema. The underlying technique used here is very similar to the one used for the Course Viewer.

3.6 Interface – Input Types

Whenever the user selects an item from the course viewer in the author mode, all its editable attributes will be displayed in the content area of the editor. In this context, editable means: The property is contained in one of the RDF schemes of the toolbar. This way, its possible to reduce the shown properties to a subset of significant ones by simply removing some bound RDF schemes from the toolbar.

A property can be displayed in several different manners, depending on the properties range settings and the state of the object.

3.6.1 Properties with Literals

If the value of a property is a literal value, the user interface will show a plain entryfield, containing the literal.

If the property does not have a value yet (object is empty) and there is no range definition given for the property, the interface will show an empty entryfield, where the user may enter any valid URI.

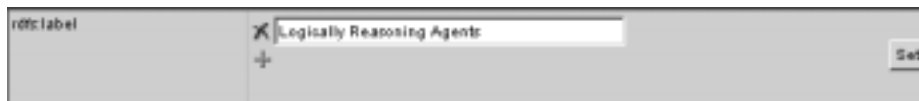


Fig. 6. A property with a literal value

3.6.2 Properties with Range

If a property has no value yet (the object is empty) but has a range definition, the user will see – depending on the existing resources in the system – a select list with all instantiable classes for that range and/or a select list with all existing resources, that fit into the range. If `rdfs:Literal` is part of the instantiable classes, one will see an additional empty entry field for any literal. In any case, the system respects `rdfs:subClassOf` and `rdfs:subPropertyOf` definitions to find all valid classes and resources.

One can either choose to create a new instance from one of the offered classes, select an existing resource or enter a literal as the new value for the given property.

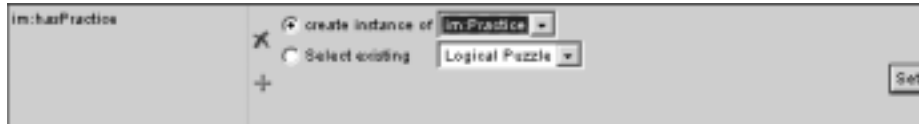


Fig. 7. A property with range and an empty value

3.6.3 Properties with Resources

Properties with an existing object resource that is known to the system, will show an internal frame that contains the attributes of that resource



Fig. 8. A property with a resource

For any subject, the user can extend the existing annotation by adding properties from the toolbar that fit to the type of the resource. “Fit” means all properties with either no domain definition, or a domain definition that somehow (by respecting super-classes) refers to the origin class of the given subject. This way, OLR3 ensures the construction of valid statements.

3.6.4 Adding Properties



Fig. 9. The list with valid properties for annotation

One relevant element of the editor is the list containing all properties available for annotation. For a selected resource (course element), the editor will filter and provide only those properties from the toolbar schemes with either no domain attribute or a domain attribute that somehow (by respecting subClassOf and subPropertyOf definitions) includes the type of the selected resource. Thus, a user automatically gets a list with “valid” properties for annotation, from which he can choose to add attributes to the given resource.

4 Comparison with other Schema Editors

We decided to develop OLR3 on the basis of regular web-browsers, to give every student the opportunity to access our courses without further software installation. This is a different approach from document viewers like CREAM from the University of Karlsruhe [HANDSCHUH2002], although the schema-driven metadata approach is very similar

On the other hand, we found out that the most editors or viewers for learning resources that use the LOM standard, like the course editor from the IPSI [IPSI2002], concentrate only on the LOM standard and lack an flexible structure to adapt to new standards.

Conzilla the Concept-Browser, developed by the CID [NILSSON1999], is also a metadata-focused tool, with an editor using the LOM standard, but no course editor. Its main goal is to present complete fields of science and their concepts.

References

- [ALLERT2001] H.Allert, H.Dhraief, W. Nejd. *How are Learning Objects Used in Learning Process?* L3S, 2001
- [ENHYDRA] Enhydra Open Source Java/XML Application Server, <http://enhydra.enhydra.org>
- [HANDSCHUH2002] S.Handschuh, S. Staab, A.Maedche. *CREAM- Creating relational metadata with a component-based, ontology-driven annotation framework*
- [IPSI2002] S. Hoermann, A. Faatz, et.al *Ein Kurseditor für modularisierte Lernressourcen auf der Basis von LOM zur Erstellung von adaptierbaren Kursen.*
- [LTSC2002] *Draft Standard for Learning Objects Metadata* IEEE P1484.12/D6.3 12 January 2002
- [NILSSON1999] Nilsson, M. & Palmér M., *Conzilla - Towards a Concept Browser*, (CID-53), KTH, 1999.
- [NILSSON2001] M. Nilsson. *IMS Metadata RDF binding guide*. May 2001
- [SIRPAC] SiRPAC RDF Parser, Stanford, <http://www-db.stanford.edu/~melnik/rdf/api.html>
- [VRP] K.Tolle. VRP RDF Parser, ICS Forth, Greece, <http://www.ics.forth.gr/proj/isst/RDF>
- [WOLF2001] B.Wolf, H. Dhraief, M. Wolpers, W. Nejd. *Open Learning Repositories and Metadata Modeling* SWWS, 2001