

Annotation for an open learning repository for Computer science– Case Study & OLR3 editor

—

J. Brase, W. Nejd
Information System Institute
and LearningLab Lower Saxony
University of Hannover

22.01.2003

1 Introduction

In this case study we give you a brief overview how we annotate our computer science course and how we storage them in our learning repository. In section 2 we give you a brief introduction to the field of metadata and their binding and discuss which subset of the metadata schema LOM (Learning objects metadata) by the IEEE we find most useful to use for annotating our resources. We also present you the taxonomy we use to classify our learning resources by their content.

During the project PROMISE (Project oriented multimedia study of electric engineering and computer science), a joint project of three north german universities, founded by the government of Lower Saxony from 1999 to 2001, the department for knowledge based systems in the Information System Institute in the university of Hannover developed an open learning repository (the OLR) for the course of software engineering.

In the years 2001 to 2002 new repositories have been implemented based on the first OLR. We give you an overview of the standard OLR architecture in section 3. Taking a closer look at the content classification of the resources, we look at the metadata based queries inside our repositories in more detail.

In section 4 we finally present you the latest generation of the OLR3, that we use for our artificial intelligence course and present its new metadata schema editor.

2 Case study - Annotating courses

2.1 Finding relevant metadata standards

Metadata is data about data that helps us to receive better search results. Instead of hoping that a full text search through a learning resource will somewhere inside find the author's name nejdl for example, we can annotate the resource with a metadata attribute "author is nejdl". At once we see the two major difficulties this method holds: the technical realisation of "attaching metadata to a resource", and the standardization to avoid misunderstanding by using different attributes for the same purpose like "author is nejdl", "creator is nejdl" or "written by nejdl", etc.

Let us first discuss the idea of using a generalized vocabulary or schema for metadata.

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. To annotate the author of a learning resource we would use the element creator, and write `dc:creator = nejdl`

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 element title. For a complete description, we refer to [4]. 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) [8] 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 latest draft version was 6.4. Once the standard has been accepted, which hopefully will happen this year, it will be LOM 1.0. Since LOM defines metadata attributes for any kind of learning resource, it is obvious that one does not need to use all of the 70 attributes, but can use his own subset for annotation. On the other hand, regardless 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. Therefore colleagues from the Learning Lab Lower Saxony (L3S) work on interesting ideas to even extend LOM in the field of educational metadata. I refer to [2] for more details. For our courses we mostly use a set of roundabout 12 attributes per resource, but having the wide variety of LOM to choose from gives us the opportunity to match the set perfectly to different types of resources.

2.2 Appropriate bindings

Mainly two possible ways to realize the "binding" of metadata to a resource have been developed in the last years, XML-binding and RDF-binding. XML stands for Extensible Markup language and is derived from the document description language SGML (The international standard for structured information). RDF stands for Resource Description Framework. A great amount of work in the field of binding is 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) [10]. We will not discuss the details of these types of binding, but shortly describe, why we use the RDF-binding for our resources.

The XML Binding defines an exchange format for metadata. The metadata might be contained in a database and a XML representation is usually generated on demand, for export to other tools and environments. Thus, a XML metadata record is a self-contained entity with a well-defined hierarchical structure, so there is seldom a natural way to reuse other metadata

standards.

A RDF statement is a triple consisting of a subject, a predicate and an object, where the subject is referenced by an URL. The most common way to represent RDF is in XML. Using the syntax proposed by the W3C the statement that a resource has a certain author could be written as:

```
<rdf:RDF
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dc="http://purl.org/dc/elements/1.1#" >
<rdf:Description about=http://www.xyz.com/resource.html>
<dc:creator>nejdl</dc :creator>
</rdf:Description>
</rdf:RDF>
```

We still can identify the triple-structure of RDF: the resource, referenced by the URL "http://www.xyz.com/resource.html" is the *subject*, "dc:creator" is the *predicate* and the literal value "nejdl" is the *object*. The namespace "dc:" refers to an URL containing a RDF file that describes the structure of the metadata attributes of Dublin Core, dc:creator in this case. That file is called the RDF-binding for DC. If we want to describe a resource with RDF and use LOM, we write triples like the one above in a rdf-file and refer to the RDF binding of the LOM attributes we use. A final metadata description is just a set of these triples. The use of namespaces makes it a part of a global network of information, where anyone has the capability of adding metadata to any resource, using standardized or specialized schemas describing these metadata. This modularity of the architecture leads to naturally reusable constructs. The LOM RDF binding is directly compatible with Dublin Core RDF binding, and therefore Dublin Core elements are used directly instead of defining new LOM elements for these DC properties. This of course helps a lot to enhance the interoperability between resources that are annotated with DC and others that are annotated with LOM. In the latest draft for the RDF binding of LOM ([10]) for example, about 80 percent of the LOM elements are defined using DC and DCQ.

2.3 An example course for software engineering

In our system a whole course is represented by a single rdf-file. In the following you see a simple example of a course consisting of one lecture-unit with the slides and the streaming video of the lecture as learning resources, containing description, title, rights, language and a classification entry

```
01 <?xml version='1.0' encoding='ISO-8859-1'?>
02<rdf:RDF
03 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
05 xmlns:dc="http://purl.org/dc/elements/1.1/"
06 xmlns:dcq="http://dublincore.org/2001/08/14/dcq#"
07 xmlns:lom_cls="http://www.imsproject.org/rdf/imsmd_classificationv1p2#"
08 xmlns:lom="http://ltsc.ieee.org/2002/09/lom-base#"
09 xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#">
10
11 <rdf:Description rdf:ID="SELecture">
12 <rdf:type rdf:resource="http://telemann.kbs.uni-hannover.de:3333/
13         olr/olr_v9#Course" />
14     <dc:title>SE Lecture in winter 2001</dc:title>
15     <dc:creator>
16         <lom:entity>
17             <vCard:FN>Friedrich Steimann</vCard:FN>
18         </lom:entity>
19     </dc:creator>
20     <dcq:created>
21         <dcq:W3CDTF>
22             <rdf:value>2001-09-15</rdf:value>
23         </dcq:W3CDTF>
24     </dcq:created>
25     <dcq:hasPart>
26         <rdf:Seq>
27             <rdf:li rdf:resource="#Unit1" />
28         </rdf:Seq>
29     </dcq:hasPart>
30 </rdf:Description>
31
```

```

32 <rdf:Description rdf:ID="Unit1">
33 <rdf:type rdf:resource="http://telemann.kbs.uni-hannover.de:3333/
34         olr/olr.v9#Unit"/>
35 <dc:title>1. Lecture unit 22.10.2001</dc:title>
36 <dc:description>Introduction to the course</dc:description>
37 <dcq:isPartOf rdf:resource=" #SELecture"/>
38 <dcq:hasPart>
39 <rdf:Seq>
40 <rdf:li rdf:resource="http://www.kbs.uni-hannover.de/Lehre/SWT1/
41         /S1T1.pdf"/>
42 <rdf:li rdf:resource="http://www.mml.uni-hannover.de/
43         meta/kbs.cgi?SWT1WS01S01T1.rm"/>
44 </rdf:Seq>
45 </dcq:hasPart>
46 </rdf:Description>
47
48 <rdf:Description rdf:about="http://www.kbs.uni-hannover.de/Lehre/
49         SWT1/OLR/S1T1.pdf">
50 <rdf:type rdf:resource="http://telemann.kbs.uni-hannover.de:3333/
51         olr/olr.v9#PDF"/>
52 <dc:title>Slides for the first lecture</dc:title>
53 <dc:description>Overview of the discipline. And a brief intodruction
54         to the Function-Point-Method</dc:description>
55 <dc:rights>balzert</dc:rights>
56 <dc:language rdf:resource="http://www.kbs.uni-hannover.de/~brase/
57         lang.rdf#en"/>
58 <dc:subject rdf:resource="http://www.kbs.uni-hannover.de/~brase/
59         SWT_Ontologie.rdf#MeasSWDevelopment"/>
60 <dc:subject rdf:resource="http://www.kbs.uni-hannover.de/~brase/
61         SWT_Ontologie.rdf#FuncDesMeasures"/>
62 <dcq:hasFormat rdf:resource="http://www.mml.uni-hannover.de/meta/
63         kbs.cgi?SWT1WS01S01T1.rm"/>
64 <dcq:requires rdf:resource="http://www.kbs.uni-hannover.de/Lehre/
65         SWT1/OLR/S3T2.pdf"/>
66 <dcq:isPartOf rdf:resource=" #Vorlesungseinheit4"/>
67 </rdf:Description>
68
69</rdf:RDF>

```

Line 11-30 define the structure of the complete course, Line 32-46 of the unit, using `rdf:Sequence`. Also between these description tags stands all metadata concerning the whole course respectively the unit. Between line 48 and 67 is the metadata concerning the pdf-slides as one of the learning resources.

For a complete description of the attributes used, we refer to [10] and [4]. In the following we give you a brief overview of the subset of LOM, we use to annotate our courses in software engineering.

rdf:type (line 12, 33, 50): The type *course* or *unit* stands for a self-created structure in contrast with a learning resource, which is physically present somewhere in the internet. Also `rdf:type` is used to ensure that the resources are displayed in the best possible way. For type *pdf* or *video*, a new window will open, including a small navigation bar for "Play", "Pause" and "rewind" in case of type *video*.

rdf:ID (line 11, 32): Each structure element like the course or a unit, that can not be physically located via its URI, needs a unique ID to be referenced by inside the database.

dc:title (line 13, 35, 52): The title of the resource will be displayed in the navigation tree. Often the title is displayed as the result of a query. Therefore it is very useful to choose a title that is short, but rich and understandable.

dc:creator, dcq:created (line 15, 20): These attributes to describe author and time are usually only used once in the course metadata description and then inherited to every unit and sub-unit inside the system.

dc:description (line 36, 53): Contains a short description that can be displayed, when opening the resource.

dcq:isPartOf, dcq:hasPart (line 25, 37, 38, 66): This attributes are used to define the structure of a course.

dc:rights (line 55): Some parts of the lecture "Software Engineering I" are based on a lecture by Prof. Balzert from the university of Bochum. We use this attribute to cope with any problems of copyright by identifying resources from other authors.

- dc:language** (line 56): A useful attribute for querying resources to make sure that you will be able to understand the language of the learning resources you get as a result.
- dcq:hasFormat** (line 62): As the lectures are represented by a pdf-file with the slides and a streaming video we use `dcq:hasFormat` and `dcq:isFormatOf` to receive better search results by eliminating resources that are only a different version of an already known search-result.
- dcq:requires** (line 64): This metadata instance from the Relation category binding (see [10] for the details) is needed for an adaptive version of the script. It shall prevent readers from starting with chapters that build up on other chapters the reader has not yet looked at. This adaptive version of the script is developed as a diploma and will be presented in a later paper.
- dc:subject** (line 58, 60): This attribute is used to classify the content of the learning resource. We will discuss it in detail in the next chapter.

The list above represents our best practice list. It turned out that these 12 attributes are enough to richly annotate our resources and gain a high quality in querying the system, but they are far less than the over 70 attributes that form the LOM schema.

2.4 The SWEBOK taxonomy

As mentioned before we are interested in classifying the content of our resources as well. It is obvious that using self-defined keywords can only be a first solution to this problem. To receive the best search results the used keywords should be part of an international accepted and used classification system, such as the ACM CSS [1] for example. For the discipline of software engineering, we decided to use the SWEBOK as a taxonomy.

The SWEBOK has been developed as a Guide to the Software Engineering Body of Knowledge in a context of an IEEE/ACM working group. On their webpage [14] we find the definition "The purpose of this guide is to provide a consensually-validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of

Knowledge supporting that discipline.” Almost 500 software engineering professionals from 41 countries have hierarchically structured the field of software engineering in 10 Knowledge Areas and almost 300 topics, based on their number of publications. The Knowledge areas are:

- Software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering process
- Software engineering tools and methods
- Software quality

Compared to the already established ACM classification, that tries to catalogue the whole discipline of computer science, the SWEBOK classification is more detailed and covers more theoretical aspects of software engineering.

Therefore we decided to use the SWEBOK Classification to receive a larger amount of annotation entries. Since using more than one classification system to annotate learning resources is very useful, we consider an automated mapping from SWEBOK to ACM.

2.5 A machine readable version of the taxonomy based on LOM

To classify a resource the IEEE learning object RDF binding guide (Draft version) [10] suggests the use of `dc:subject` with elements of a taxonomy that must be found on the Internet. Such a taxonomy hierarchy is an instance of `lom-cls:Taxonomy` and must be formatted in a RDF file where the topics and subtopics are separated following the example below.

Based on the SWEBOK we created a RDF-file with a taxonomy for the area of software engineering called mySWEBOK. The complete ontology can be found at [3], The main structure is as follows:

```

<dcq:SubjectScheme rdf:ID="mySWEBOK">
<rdfs:label>Software Engineering Book of Knowledge Field Classification</rdfs:label>
</dcq:SubjectScheme>

<lom_cls:Taxonomy>
  <lom_cls:rootTaxon>
    <swtOnt:mySWEBOKrdf:about="http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf#SWEnginManagement">
    <rdf:value>Software engineering management</rdf:value>
      <lom_cls:taxon>
        <swtOnt:mySWEBOK rdf:about="http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf#SWEnginMeasurement">
        <rdf:value>Software engineering measurement</rdf:value>
          <lom_cls:taxon>
            <swtOnt:mySWEBOK rdf:about="http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf#MeasSWDevelopment">
            <rdf:value>Measuring software and its development</rdf:value>
            </lom_cls:taxon>
          </lom_cls:taxon>
        </lom_cls:taxon>
      </lom_cls:rootTaxon>
    </lom_cls:Taxonomy>

```

This extract from the ontology shows the definition of one knowledge area (Software engineering management) divided in this example in one subtopic (Software engineering measurement), which is itself divided in one subtopic (Measuring software and its development).

All the RDF Triples building the ontology are stored in the database of the OLR as well. Since the SEWBOK is more a classification than an ontology we find almost no dependencies between the different topics, as we would aspect of an ideal ontology.

Nevertheless mySWEBOK is exactly what we need for our purposes, an universal structure of the discipline of software engineering that is widely accepted.

3 OLR

Our Open Learning Repositories, Versions 1 to 3, are connected to an Oracle Database. In the database we store the metadata of a course, never the content itself. Therefore, it is easy to include material from different sources throughout the internet to a course, if the copyright is granted (Otherwise we recommend the use of dc:rights). As shown above, a course is represented by one single rdf-file containing the structure and all metadata of a course. The rdf-Triples are stored as triples in the database and are used by the system to display the system.

For further information about the OLR we refer to [16].

3.1 Architecture

The basic architecture of the different OLR versions is the same. The first two versions were php-based. Our newest Open Learning Repository, the OLR3 system however, that we use for our artificial intelligence course, is implemented in Java and works as a JavaServlet, running on an Enhydra [5] 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. The resulting RDF statements are defined on basis of 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 [12], whereas imported RDF files are parsed by a VRP RDF parser [15], which provides semantical checks against given RDF schema rules.

OLR3 offers a web-browser based metadata editor/viewer and provides two major user interfaces: One for readers with a more graphically oriented

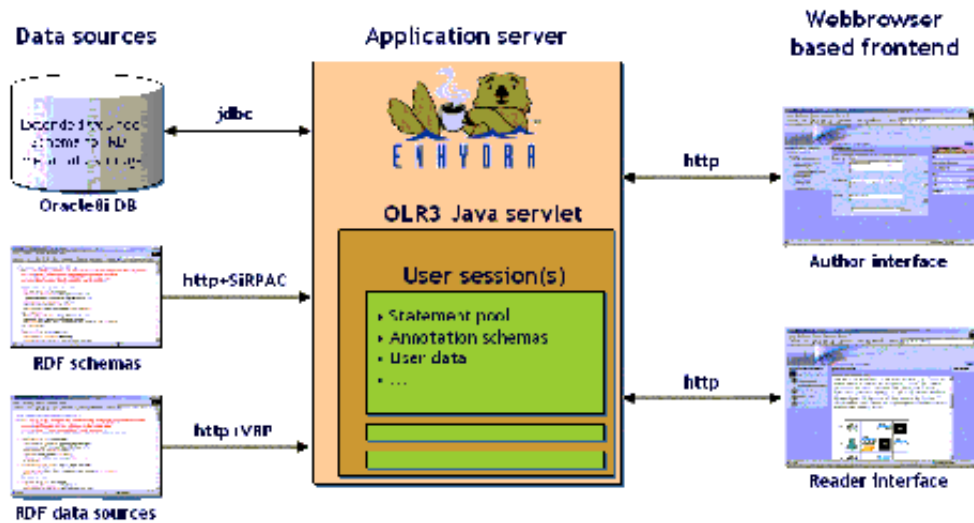


Figure 1: The architecture of the OLR3 system

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 elements 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 offers 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.

3.3 Querying the course meatdata

We assume that we can access the triples with a view `STATEMENT(Subject,Predicate,Object)`. Let us also consider that every resource has an ID that was automatically generated, when the RDF file was stored in the database. We will now have look at how the metadata is queried in more detail. Therefore we return to our software engineering course, and the way it is represented in the OLR2.

Parts of our little software engineering course and of the mySWEBOK Taxonomy can be found in the database as:

```
...
STATEMENT(http://www.kbs.uni-hannover.de/Lehre/SWT1/OLR/
           S1T1.pdf, ID, 577)
STATEMENT(577, dc:title, "Slides for the first lecture")
STATEMENT(577, dc:language, http://www.kbs.uni-hannover.de/
           ~brase/lang.rdf#en)
STATEMENT(577, dc:subject, http://www.kbs.uni-hannover.de/~brase/
           SWT_Ontologie.rdf#FuncDesMeasures)
...
STATEMENT(http://www.kbs.uni-hannover.de/~brase/
           SWT_Ontologie.rdf#FuncDesMeasures, ID, 1004)
STATEMENT(1004, rdf:value, "Measuring software and its development")
STATEMENT(1017, lom_cls:taxon, 1004)
STATEMENT(http://www.kbs.uni-hannover.de/~brase/
           SWT_Ontologie.rdf#SWEnginMeasurement, ID, 1017)
...
```

When we display a learning resource, one can decide between looking at the content, by choosing a layer labelled "content" or looking at the classification-entry, by choosing the layer labelled "taxon".

If you choose "taxon" the system will first retrieve the ID of the current learning resource, which is 577 in this example and then look if there are any `dc:subject` entries in the Oracle-Database via:

```
SELECT Object FROM STATEMENTS WHERE Subject='[577]' AND
Predicate='dc:subject'
```

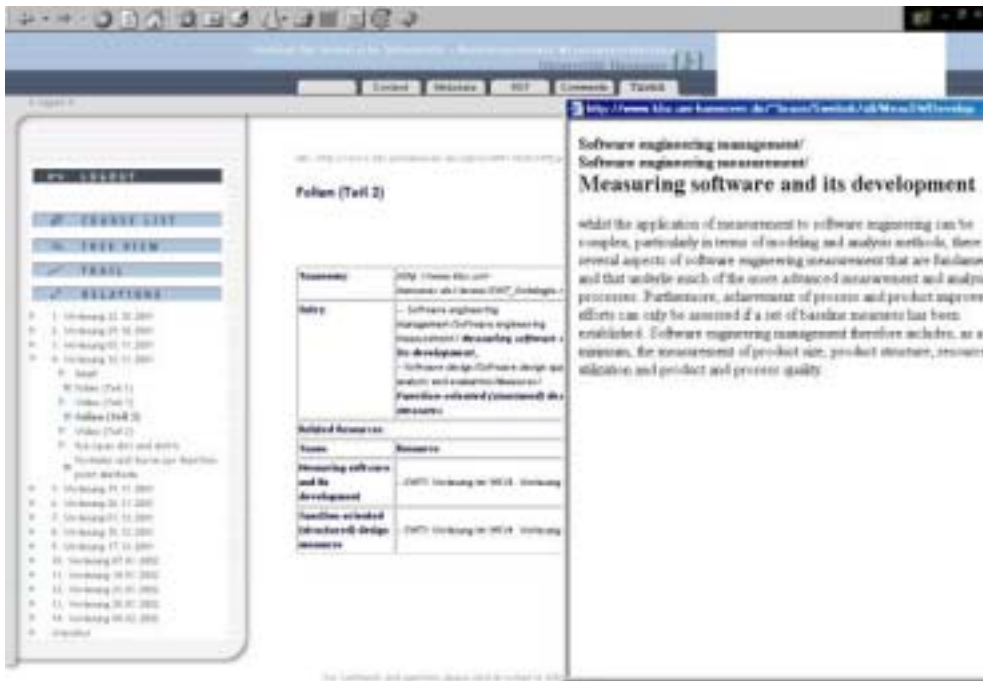


Figure 2: Taxon entries and related resources for a specific resource in the OLR2

The taxon entry is displayed not with its original entry but the system will retrieve the `rdf:value` entry of it from the taxonomy as the title. The system will also retrieve the taxon-structure identified by the taxon-attribute. Remember that the complete SWEBOK taxonomy is also stored in triples in the database. In our example the entry *MeasSWDevelopment* will be displayed as:

Software engineering management / Software engineering measurement / Measuring software and its development

We decided to display the entry as a hyperlink to the user leading to a brief description of the subject, that this resource was described as belonging to by the entry (see Fig 2).

3.3.1 Related Resources

Furthermore, the related resources (Resources with the same classification entry *MeasSWDevelopment* from the same taxonomy mySWEBOOK) are retrieved from the database via:

```
SELECT Subject FROM STATEMENTS WHERE Predicate='dc:subject'  
AND Object='[http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf  
#MeasSWDevelopment]')
```

and displayed as a hyperlink to the resource (see Fig 2).

3.3.2 Annotating chapters

Since a course is only represented by a RDF-Schema, a single learning resource can be annotated via its RDF description. A whole course, or a single lecture exists only inside the schema. Of course it has a description that we could annotate, but since we want to be as flexible as possible in creating different views of different lectures from the same learning resources, lectures and courses inherit their taxon-entries from their child-units. If the user chooses the layer "taxon" while he is not in a learning resource, but looking at the structure of a unit, the sub-units of this unit are retrieved via:

```
SELECT [The sub-units names] FROM STATEMENTS WHERE Subject IN  
(SELECT Object FROM STATEMENTS WHERE Predicate ='dcq:hasPart'  
AND Subject=[The ID of the current unit])
```

Then their taxon entries are displayed together with the learning resource they come from. Displaying the taxon entries of a whole course works in the same way, it only takes one iteration more to receive the taxon entries from the "grandchildren"

4 OLR3 Schema Editor

The major difference between the OLR3 and its ancestors is the schema editor. Mainly two considerations led to the development of the OLR3 metadata editor: Firstly, the need for an editor which allows to edit content online from anywhere in the world without additional client software, plug-ins or configuration. And secondly, the demand for an extensible and flexible authoring interface, which should not be limited to a certain metadata standard or schema, but be open for arbitrary future developments, ideas and resulting schemes.

The OLR3 editor can handle any given RDF scheme and - once it is registered to the editor - use it for metadata input. Thus, the set of available RDF properties will only be limited by the number of available schemes that define the properties. An author can choose any desired property from existing standards (e.g. LOM, DC) and compose his own set of metadata attributes to annotate learning resources.

4.1 Author Interface Layout

The OLR3 offers a second interface, 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 is 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.

4.2 Structural Schema Viewer

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



Figure 3: The Toolbar here contains several schemes, with "rdfs" expanded and showing the subitems for the property "comment"

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.

4.3 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



Figure 4: A property with a literal value

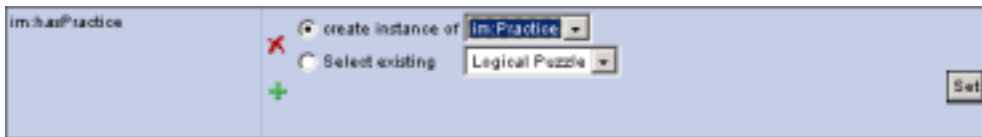


Figure 5: A property with range and an empty value

the properties range settings and the state of the object.

4.4 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.

4.5 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.



Figure 6: A property with a resource

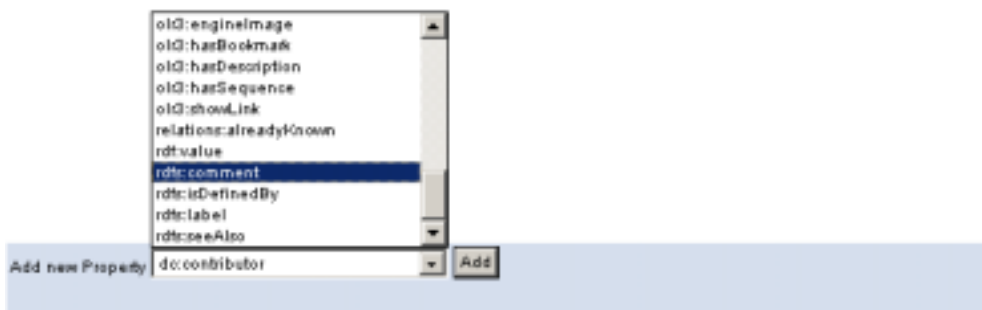


Figure 7: The list with valid properties for annotation

4.6 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

4.7 Adding Properties

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.

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 ei-

ther 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.

5 Related work

Even though the number of digital libraries and Learning repositories have started to increase rapidly, we haven't yet encountered a similar approach to classify the resources than our use of the LOM Standard together with a machine readable rdf-file as a classification scheme. Whereas the use of metadata to annotate resource is widely accepted and common, the traditional way of annotating the content of a resource is by using keywords, and query these keywords later via a full text search.

Even Portal the Digital library of the ACM [11] itself uses the ACM CSS Classification for a simple keyword annotation of the resources, the same as for example the SMETE Digital Library [13].

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 [6], although the schema-driven metadata approach is very similar.

A similar approach was developed in the K-Med project by the university of Darmstadt [7]. The courses presented in their editor are also only represented by metadata-schemes, using LOM metadata. The system however is focused only on LOM and therefore lacks our possibility to include new, self-created metadata elements in the course schema.

Conzilla the Concept-Browser, developed by the CID [9], is a very interesting metadata-focused tool, with an editor using the LOM standard, which is very similar to our approach. It is however no course editor. Its main goal is to present complete fields of science and their concepts.

6 Conclusion

In this chapter we have discussed the annotation of courses with metadata. We have presented an example RDF file for a course description and introduced the metadata standard LOM from the IEEE we use to describe our learning resources. We identified a subset of 12 attributes that are enough to richly annotate our resources and gain a high quality in querying the system, but are far less than the over 70 attributes that form the LOM schema.

The content of a resource is classified using the attribute `dc:subject`. Instead of having self-defined keywords as a value, `dc:subject` points to a taxon in a taxonomy hierarchy. We have identified the Software Engineering Body of Knowledge (SWEBOK) developed in context of an IEEE/ACM working group as a content classification for this discipline and have implemented a machine readable version of it as a taxonomy. We have discussed how to use this content classification inside our open learning repository to easily identify related resources for example.

To add and edit metadata annotation in a comfortable way, we have included a metadata schema editor in the latest version of our open learning repositories, which supports us in linking different schemas and creating our own metadata sets. This editor is server based and accessible for authors via the internet by simply using a browser.

7 Acknowledgements

The basic versions of the OLR were created and implemented by Boris Wolf. The OLR3 and the Schema-editor were implemented by Tobias Kunze. We also gratefully acknowledge important input and discussion on the design and use of the OLR3 editor from Hadhami Dhraief

References

- [1] *The ACM Computing Classification System–1998 Version* valid in 2002
<http://www.acm.org/class/1998/>
- [2] H. Allert, H. Dhraief, W. Nejd. *How are Learning Objects Used in earning Process?* ED-MEDIA 2002, World Conference on Educational Mul-

timedia, Hypermedia & Telecommunications, Denver Colorado, United States, June 24-29, 2002

- [3] J. Brase *mySWEBOK – A classification for Software engineering, based on the SWEBOK*
http://www.kbs.uni-hannover.de/~brase/SWT_Ontologie.rdf
- [4] The Dublin Core Metadata Initiative
<http://dublincore.org/>
- [5] *Enhydra Open Source Java/XML Application Server*
<http://enhydra.enhydra.org>
- [6] S.Handschuh, S. Staab, A.Maedche. *CREAM- Creating relational meta-data with a component-based, ontology-driven annotation framework ACM K-CAP*, First International Conference on Knowledge Capture, October 2001 , Vancouver.
- [7] S. Hoermann, A. Faatz, et.al *Ein Kurseditor fr modularisierte Lernressourcen auf der Basis von LOM zur Erstellung von adaptierbaren Kursen*.LLWA 01 - GI-Workshopwoche "Lernen-Lehren-Wissen-Adaptivitt", 2002
- [8] Learning Technology Standards Comittee of the IEEE: *Draft Standard for Learning Objects Metadata IEEE P1484.12.1/D6.4* 12. June 2002).
http://ltsc.ieee.org/doc/wg12/LOM_1484.12.1_v1_Final_Draft.pdf/
- [9] Nilsson, M. & Palmr M., *Conzilla - Towards a Concept Browser*, (CID-53), KTH, 1999.
- [10] M. Nilsson.*IMS Metadata RDF binding guide*May 2001
<http://kmr.nada.kth.se/el/ims/metadata.html>
- [11] *PORTAL to computing literature*
<http://portal.acm.org/portal.cfm>
- [12] *SiRPAC RDF Parser, Stanford*
<http://www-db.stanford.edu/~melnik/rdf/api.html>
- [13] *SMETE Digital Library* <http://www.smete.org/>

- [14] *The guide to the Software engineering body of Knowledge*
<http://www.swebok.org>
- [15] K.Tolle *VRP RDF Parser, ICS Forth, Greece*
<http://www.ics.forth.gr/proj/isst/RDF>
- [16] B. Wolf, H. Dhraief, M. Wolpers, W. Nejdl. *Open Learning Repositories and Metadata Modeling* International Semantic Web Working Symposium (SWWS) Stanford University, California, USA, July 30 - August 1, 2001