

Open Learning Repositories and Metadata Modeling

Hadhami Dhraief, Wolfgang Nejdl, Boris Wolf, Martin Wolpers

Knowledge Based Systems

Institute of Computer Engineering

University of Hannover

Appelstr. 4, D-30167, Hannover, Germany

Tel: +49 511-762-19714

Fax: +49 511-762-19714

E-mail: {dhraief, nejdl, wolf, wolpers}@kbs.uni-hannover.de

Abstract

The Open Learning Repository (OLR) is a framework for designing and implementing modularized and distributed learning repositories on the web. In the first part of this paper, we will discuss an OLR prototype system built on meta-information stored as RDF metadata. It is one of three prototypes implemented at our Institute to enhance e-learning. The KBS Adaptive Hyperbook, our first prototype, has been used for a project-oriented course for Java Programming, whose conceptual model is implemented in the conceptual modelling language O-Telos. Building on this experience, we will discuss RDF/RDFS and O-Telos modelling in depth in the second part of this paper and will analyse similarities and differences of these two modelling languages.

Keywords:

Meta modeling, RDF/RDFS, conceptual modelling, hypermedia, learning repositories.

1 The Open Learning Repository

1.1 Motivation

Our Open Learning Repositories aim at metadata-based course portals, which structure and connect modularised course materials over the Web. The modular content can be distributed anywhere on the internet, and can be integrated in order to build courses and sets of learning materials. The modules are usually reused for other courses and in other contexts. This leads to a kind of course portal integrating modules from different sources and authors. Semantic annotation is necessary for authors to help them choose module and to bind them into the course structure.

We use a relational database to store metadata, but store no content in the database itself. The stored metadata represent information about the structure and navigational structure of a particular course, the URLs of single elements (modules, courslets, course units, subunits, etc.) and other useful metadata about the content itself (i.e. Dublin Core or IEEE LOM metadata). We are using the OLR system in the context of two courses, one in artificial intelligence and one in software engineering.

1.2 OLR functionality

The OLR repository can store RDF (Resource Description Framework) [W3C, 1999] metadata from arbitrary RDF schemas. However, we have chosen not to implement a one-size-fits-all approach, and implement different interfaces together with different schemas and metadata for different courses. Initial loading for a specific course is done by importing an RDF metadata file (using XML syntax) based on this course's RDFS [W3C, 2000] schema. Our Artificial Intelligence course prototype uses a simple schema describing course structure (units, subunits, elements and arbitrary links between these elements) and simple cataloguing of its elements using the Dublin Core metadata [DC, 2001] set. As soon as the LOM-RDF-binding is finalized, we will use the LOM metadata set (which basically includes the DC elements as a subset) to annotate our pages.

The web interface for navigating the course uses a multi-view approach. A user visiting the course currently has a choice between three different navigation schemes: The first one is a hierarchical tree-like navigation

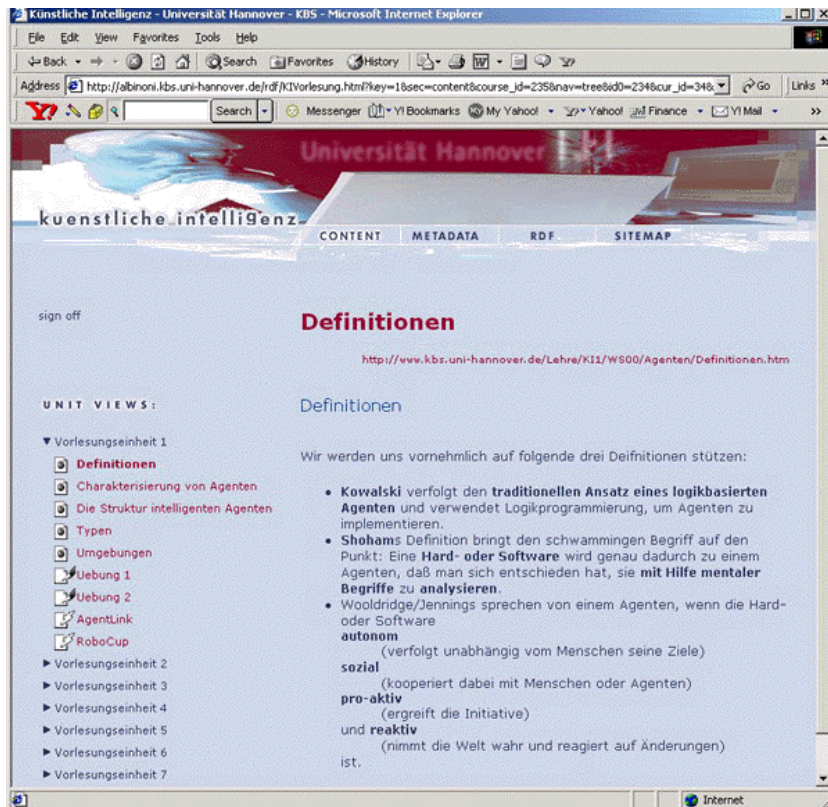


Figure 1: Structured Display of Content

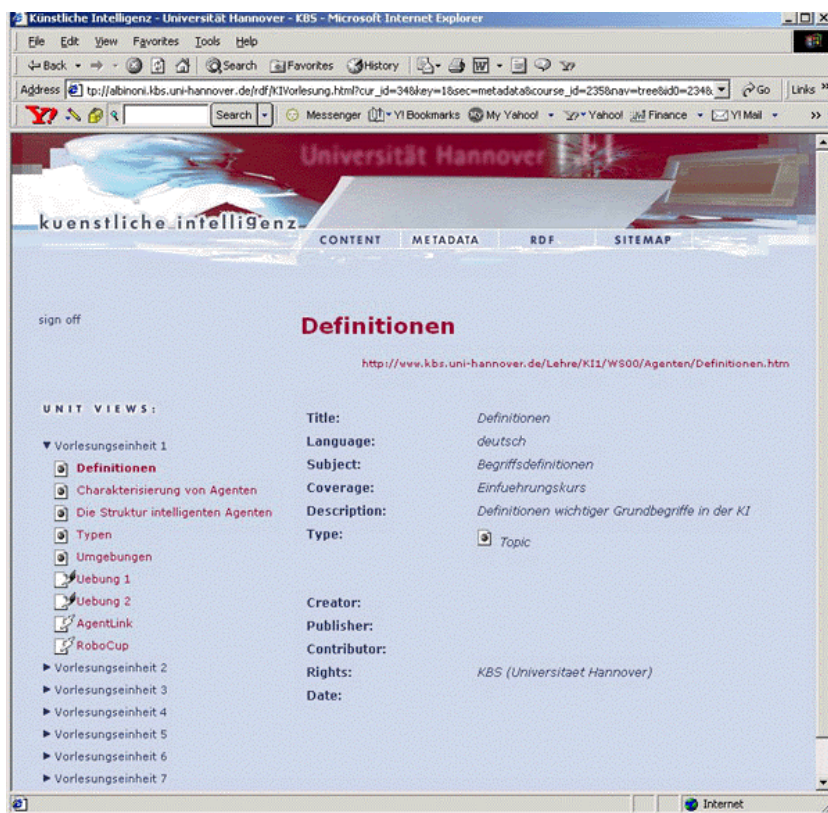


Figure 2: Display of Metadata for a Specific Resource

directly reflecting the course structure stored in the database. A visitor may open and close units and subunits to display the elements/pages of the logical document (see Figure 1). The second view provides a trail navigation

where the user has the possibility to move forward and backward on a trail. Third we are experimenting with a semantic net or context net navigation. In this approach the user can view units in different contexts, navigation is implemented as a kind of fish-eye view with the current unit located in the centre surrounded by related units and contexts. All navigation elements are created dynamically on demand.

In addition to displaying course content we are providing different ways of reviewing the metadata stored about course elements. Either the system displays metadata in a nicely formatted way suitable for a human reader or it generates the corresponding RDF source in XML notation (see Figure 2).

For content developers we are working on an enhanced web interface which allows the developer to manipulate meta-data through clearly arranged HTML forms (Figure 2). The OLR system translates all user input into suitable SQL update and insert statements hence avoiding to confront the user with having to understand the XML/RDF notation. To evaluate OLR usage, the system tracks all user behaviour in the database: which course elements are accessed and when, which updates are made and by whom. We will use this information to evaluate different navigation schemes and different types of course units.

1.3 OLR-Architecture

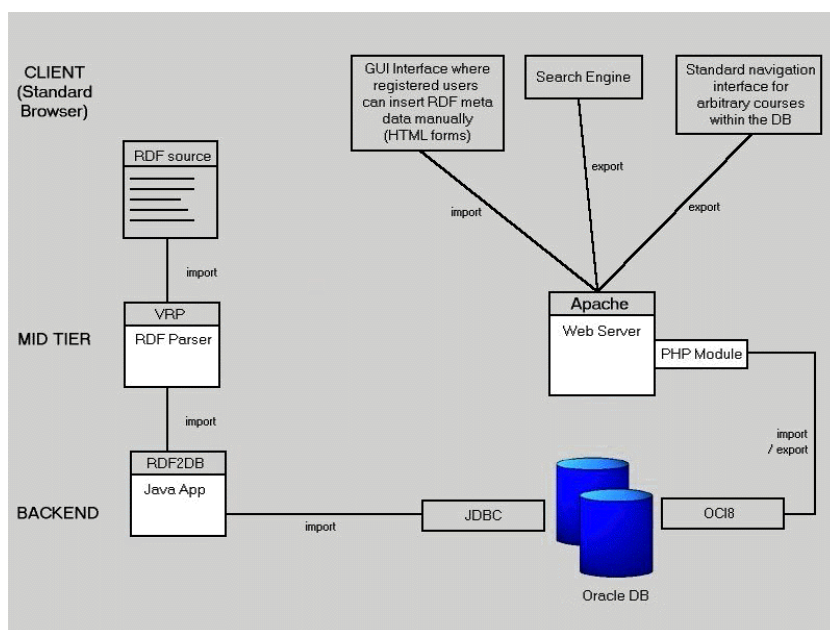


Figure 3: OLR Architecture

The OLR architecture is shown in Figure 3. It is built upon a relational database which stores RDF metadata. Displaying logical documents described by queries over these metadata or the metadata themselves is handled by an Apache Web Server and a PHP module. Different views on the logical documents are possible, realized by different queries and page designs. Importing metadata is possible by using HTML forms (see Figure 4) or a Java-based backend utilizing the VRP RDF-Parser [VRP, 2001].

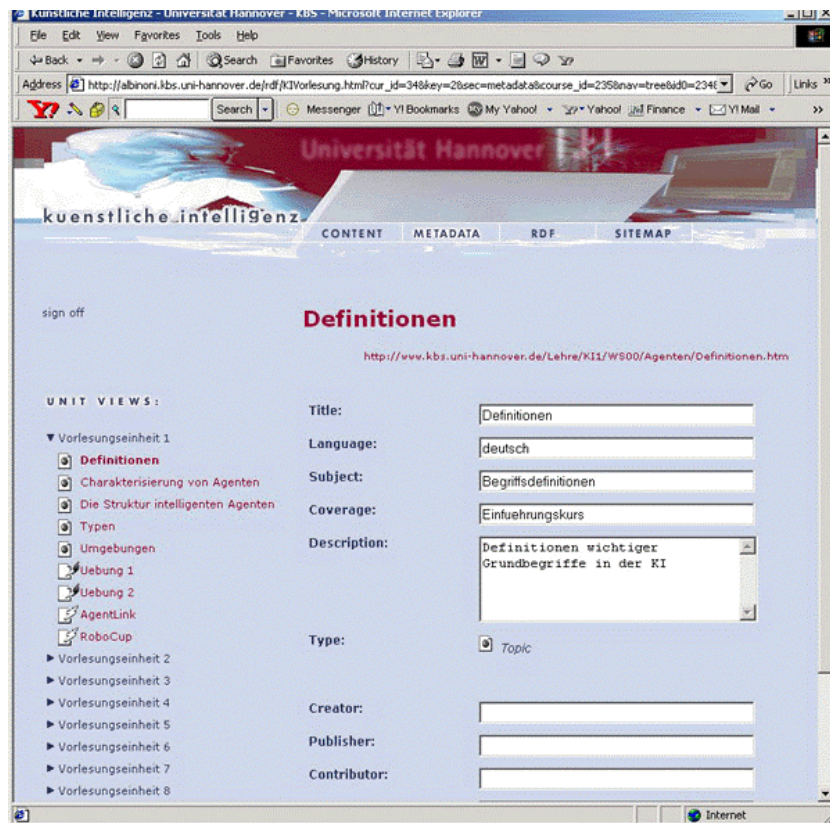


Figure 4: Adding New Metadata

1.4 Technology

1.4.1 RDF Annotation

The OLR system stores virtually anything it knows about courses as RDF metadata.

In web based learning and teaching, the trend is to encode learning materials with meaningful and machine understandable structures and semantics in order to reuse them and to facilitate the modular content building. This is also the vision of the semantic web. The Resource Description Framework RDF, the W3C standard, is considered as the way to realize this vision.

One of the practical uses of RDF, as it is described in W3C, is in Web sitemaps.

"the RDF schema specification provides a mechanism for defining the vocabulary needed for this kind of application" [W3C, 2001].

Thus, with RDF, we can describe, how modules, course units, courslets are related to each other or how an example or exercise belongs to a course unit.

Another practical use of RDF is the description of web pages, which is mandatory to build a course based on modular content, distributed all over the web. There are many standards, such as the Dublin Core Initiative and IMS, where we are currently involved in defining an RDF binding for the IEEE/LOM metadata.

As aforementioned, we decided to use RDF to annotate the content in OLR. We also decided to use RDF as a modeling language to describe the structure of a course in OLR.

RDF (Resource Description Framework) is a framework for metadata developed and maintained by the W3C. The primary target of RDF is to provide a standardized way of creating and using our own meta data vocabulary to describe information on the Web. This is useful for different purposes:

- Resource discovery (Search Engine)
- Interoperability and Knowledge Sharing (Information exchange between different applications, Software Agents.)

- Cataloguing (Description of content by attribute/value pairs e.g. "author=Smith".)
- Logical Documents (Several pieces of content physically spread over the Internet connected via RDF metadata to build one single logical document.)

Our implementation focuses on the cataloguing/annotation and logical document features of RDF. Each course is a logical document and cataloguing is used to store element information (e.g. title, author, etc.)

Everything in RDF is expressed by statements. A statement itself is a triple consisting of subject, predicate and object. In addition RDF has a class system (RDF Schema) similar to many object oriented programming and modelling systems. This class system makes it much easier to define user-specific schemas. We used RDF Schema specification in its current version 1.0 to define our own OLR schema, which describes course structures and cataloguing information.

1.4.2 Database Schema

Each course consists of a number of units that contain elements and further subunits. Each element represents any kind of Internet resource accessible through a known URL. For the first version of our introductory course on Artificial Intelligence we defined five types of basic elements: Topics, examples, slides, exercises and further references. This choice reflects the typical building blocks of a lecture at a university on an abstract level. If necessary further element types can be incorporated easily to satisfy other people's needs (we are using additional elements in our Software Engineering course). The basic building blocks (units and elements) are linked together in a tree-like structure that represents a course. Each element is described with metadata. The vocabulary describing each element is selected from the Dublin Core Metadata

In essence, everything in RDF is expressed through statements: simple triples made up of resources, namespaces and literals - no matter how complex the RDF schema behind it might be. XML syntax is the standard approach of including RDF in HTML pages. It then requires a parser to analyse the meta-information. Because of the triple approach of RDF, it is quite useful to store RDF metadata in a relational database: huge amounts of metadata can be stored and managed at one central location (using proven relational database technology) and SQL queries can be used to extract the information we are looking for.

For our OLR server, we modified the McBride schema, which is one of several suggestions discussed on the RDF/DB Page from Sergey Melnik [Melnik, 2000], also discussed within the RDF community. Currently we are running the database on Oracle 8i, but any standard relational database would be suitable. As mentioned above, storing RDF data in a relational database allows storing huge amounts of triples. Thus the main table in our database is RDF_STATEMENT. This table representing the relationship between the three parts of a statement consists of RESOURCE (stored in RDF_RESOURCE), PREDICATE (also a resource) and OBJECT (could be resource or literal). So it contains three main attributes: SUBJECT, PREDICATE and OBJECT. These attributes are references to the resource and the literal table. Since the object can be either a resource or a literal, we use two attributes for OBJECT: OBJ_RESOURCE and OBJ_LITERAL.

The Open Learning Repository is a repository to integrate, manipulate and annotate more than one course. Thus, we need to store large amounts of statements for every course. For this purpose, we utilize the table RDF_MODEL. Each model currently corresponds to one course.

Distinctions to the McBride schema

Because OLR is used in a learning context, we established different user categories with different roles and rights. Every category has a specific view on courses and metadata. Hence we define a table for user administration which is connected to the other tables via the attribute USR. We also add the attribute MODIFIED representing the last modification date.

In OLR, we also have different visualizations of a course. All dynamic content is created using SQL queries stored in the table SQL_QUERY together with a short description to facilitate the reuse of such queries and to support the PHP interface. From a developers perspective this greatly enhances reusability and maintainability of the underlying PHP source code.

In order to evaluate the different visualizations and navigation possibilities in OLR, we define a table RDF_TRACK to record the user behaviour while accessing course elements (which resources have been visited, in which order, how often, in which view). Our current database schema is shown in Figure 5.

The model of the above example declares the following two O-Telos frames:

```
Class Course isA WWWResource
end
```

```
Class WWWResource with
  attribute
    title : String;
    description : String;
    course : Course;
end
```

The frame `Course` declares a class named `Course` as a subclass of `WWWResource`. A `WWW` resource is declared by the frame `WWWResource` stating that it has a `title` and a `description`, both of type `String`. Furthermore a `WWW` Resource has a `course` attribute holding the course the resource belongs to.

The next frame declares a `WWW` resource with the title "lecture unit 1", the description "Definitions of agents in AI". This resource belongs to the course "Introduction to AI 1" which is an introductory course about the basics of AI. Also this resource belongs to the another course "AI 2" which is a high-level course about AI.

```
Individual IntroAIIecture in Course with
  title
    t1 : "Introduction to AI 1"
  description
    d1 : "Introductory course about the basics of AI"
end
```

```
Individual HighLevelAIIecture in Course with
  title
    t1 : "AI 2"
  description
    d1 : "High-level course about AI"
end
```

```
Individual IntroAIIectureUnit1 in WWWResource with
  title
    t1 : "Lecture Unit 1"
  description
    d1 : "Definitions of agents in AI"
  course
    c1 : IntroAIIecture;
    c2 : HighLevelAIIecture
end
```

The frame `IntroAIIectureUnit1` shows how the declared attributes `title`, `description` and `course` are instantiated. Especially the `course` attribute shows that O-Telos attributes usually are multi-valued. O-Telos allows to declare the attributes in their own frame, e.g.

```
Attribute IntroAIIecture!t1 in WWWResource!title
end
```

As regular frames such declarations allow the definitions of attributes themselves:

```

Attribute IntroAIIecture!t1 in WWWResource!title with
  comment
    com : "this is a comment to an instantiation link between attributes"
end

```

The frames are translated to sets of propositions which are stored in the ConceptBase database. The definition of O-Telos propositions is a relation $P(oid,x,l,y)$ with oid being the identifier, x being the source, l being the label and y being the destination. Consequently $P(oid,x,l,y)$ states a relationship called l with ID oid from object x to object y .

O-Telos states natural interpretations for four predefined types of propositions. The first of these types is the object declaration $P(oid,oid,l,oid)$ declaring an object named l . As second predefined type an instance relationship is expressed using the proposition $P(oid,x,*instanceof,y)$ stating that x is an instance of y . The third type declares the inheritance relationship by stating propositions of the kind $P(oid,x,*isa,y)$ saying that x is a specialisation of y . The fourth predefined type of proposition states the attribute therefore specifying all other possible propositions.

2.3 Simple mapping of RDF to O-Telos

Let us start with a simple mapping of an RDF declaration to an O-Telos declaration:

```

<rdf:Description ID="LectureUnit1">
  <rdf:type resource="http://.../olr_schema_6#Unit"/>
  <dc:title>Lecture Unit 1</dc:title>
  <dc:description>Introduction to intelligent agents</dc:description>
  <olr:parentCourse rdf:resource="#AIIecture"/>
  <olr:child rdf:resource="http://.../Agents/Definitions.htm"/>
  <olr:child rdf:resource="http://.../Agents/Characterisation.htm"/>
  <olr:child rdf:resource="http://.../Agents/Structur.htm"/>
  <olr:child rdf:resource="http://.../Agents/Types.htm"/>
</rdf:Description>

```

The RDF declaration can be mapped to the following O-Telos frame which contains basically the same information:

```

Individual LectureUnit1 in olr_unit with
  dc_title
    t1: "Lecture Unit 1"
  dc_description
    d1: "Introduction to intelligent agents"
  olr_parent_course
    pc1: AIIecture
  olr_child
    c1: "http://www.kbs.uni-hannover.de/.../Definitions.htm";
    c2: "http://www.kbs.uni-hannover.de/.../Characterisation.htm";
    c3: "http://www.kbs.uni-hannover.de/.../Structur.htm";
    c4: "http://www.kbs.uni-hannover.de/.../Types.htm"
end

```

The above example shows that the `rdf:type` property is mapped to the O-Telos relationship in (`instanceof`). Also the property declarations `dc:title`, `dc:description`, etc. are mapped to the respective O-Telos attributes. Note that both declarations require the declarations of the objects/classes `Unit/olr_unit` and the course `AIIecture`.

This mapping is pretty straight forward. It does not take into account any sequencing of the `olr:child` properties. Furthermore the `rdf:about` construct is not reflected leaving out the fact that RDF allows metadata about resources which are referenced by their URI.

2.4 Enhancing the simple mapping (descriptions and aggregations)

A more in-depth examination of the RDF Model and Syntax Specification and our OLR Schema shows that we can distinguish two types of general classes in RDF. The first type are classes whose instances group/aggregate other instances. We will call these classes "aggregation classes". In RDF an "aggregation class" is defined using the following statement:

```
<rdf:Description ID="...">
</rdf:Description>
```

These aggregation classes possibly define supplemental attributes for their aggregates. As shown in the above example these types of classes can directly mapped to O-Telos constructs.

The second type of the general classes in RDF are classes whose instances are assigned to web pages directly. We will call these classes "annotation classes". In RDF an "annotation class" is defined using the following statement:

```
<rdf:Description about="http://...">
</rdf:Description>
```

These annotation classes define attributes to describe the assigned web pages. Annotation classes can be used in various RDF schemas to declare attributes on the same resource (referenced by its URI). Thus annotation classes can be mapped to O-Telos constructs only if there is no other annotation class stating some attribute about the same resource. In this case, the O-Telos object takes the URI as its ID and all other attributes are referenced as above. However, in general this cannot be assured, as RDF (in contrast to the frame syntax of O-Telos) is a property centric language, where properties about a given resource can be declared in different locations. To reflect this modularity, we need a different approach for mapping RDF annotation classes to O-Telos.

As said before, resources which are described by the RDF declaration `<rdf:Description about="http://...">` have no ID property. These resources can be represented as instances of OLR annotation classes. They are groupings of attributes as the following example shows:

```
<rdf:Description about="http://.../Agents/Definitions.htm">
  <rdf:type resource="http://.../rdf/olr#Topic"/>
  <dc:title>Definitions</dc:title>
  <dc:description>Definitions of the basics of AI</dc:description>
  <dc:subject>Definitions</dc:subject>
  <dc:language>german</dc:language>
  <dc:coverage>Introductory course</dc:coverage>
  <dc:rights>KBS (Universität Hannover)</dc:rights>
  <olr:parentUnit rdf:resource="#LectureUnit1"/>
</rdf:Description>
```

Seven attributes are assigned to the web page, which is defined by the URL "<http://.../Agents/Definitions.htm>".

Number	Subject	Predicate	Object
1	http://.../Agents/Definitions.htm	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://albinoni.kbs.uni-hannover.de/rdf/olr#Topic
2	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#title	Definitions
3	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#description	Definitions of the basics of AI
4	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#subject	Definitions
5	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#language	German
6	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#coverage	Introductory course
7	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#rights	KBS (Universität Hannover)
8	http://.../Agents/Definitions.htm	http://.../rdf/olr_schema_5#parentUnit	online:#LectureUnit1

The table shows the RDF-triples representing the RDF declaration of "<http://.../Agents/Definitions.htm>". The triples are generated by the SIRPAC [Saarela, 2001] parser.

The above example could also be expressed in two separate RDF declarations about the resource "<http://.../Agenten/Definitionen.htm>". Both declarations assign values to attributes but are different objects anyway.

```
<rdf:Description about="http://.../Agents/Definitions.htm">
  <rdf:type resource="http://.../rdf/olr#Topic"/>
  <dc:title>Definitions</dc:title>
  <dc:description>Definitions of the basics of AI</dc:description>
  <dc:subject>Definitions</dc:subject>
</rdf:Description>

<rdf:Description about="http://.../Agents/Definitions.htm">
  <rdf:type resource="http://.../rdf/olr#Topic"/>
  <dc:language>german</dc:language>
  <dc:coverage>Introductory course</dc:coverage>
  <dc:rights>KBS (Universität Hannover)</dc:rights>
  <olr:parentUnit rdf:resource="#LectureUnit1"/>
</rdf:Description>
```

Nr.	Subject	Predicate	Object
1	http://.../Agents/Definitions.htm	http://www.w3.org/1999/02/22-rdf-syntax-s#type	http://.../rdf/olr#Topic
2	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#title	Definitions
3	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#description	Definitions of the basics of AI
4	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#subject	Definitions

The number of triples = 4

Number	Subject	Predicate	Object
1	http://.../Agents/Definitions.htm	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://.../rdf/olr#Topic
2	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#language	german
3	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#coverage	Introductory course
4	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#rights	KBS (Universität Hannover)
5	http://.../Agents/Definitions.htm	http://.../rdf/olr_schema_5#parentUnit	online:#LectureUnit1

The number of triples = 5.

Both tables contain the RDF triples describing the two RDF declarations of attributes to "<http://.../Agents/Definitions.htm>". The comparison of the RDF triples of both declarations (the single and the divided declarations of attributes about the resource to "<http://.../Agents/Definitions.htm>") shows that the declarations are identical. This is a reflection of the fact that RDF is property-centric, i.e. the properties are the basic concept of RDF, classes are just an add-on to define domain and range of these properties.

The advantage of property-centric approach is that properties can be assign to websites in a modular way. Furthermore it is semantically unimportant whether all properties are instantiated at once. As a result the properties are alwas multivalued , i.e. the expression <rdf:description ID="?"> for a specific web page can be used repeatedly in a RDF file (possibly in several RDF files!)

A disadvantage of this modularity is of course that we cannot define single-value in RDF. For instance, it is not possible to define a property with a single value to represent the size of a resource. Thus several people can define different (in this case inconsistent) RDF-Statements for the size of the resource which leads to inconsistent information about the resource.

In contrast this modularity cannot be expressed in O-Telos, as schema definitions in O-Telos are class-centric and not property-centric (although the representation as tuples would allow this). In O-Telos it is not possible to use e.g. "<http://?/AgentenDefinitionen.htm>" as ID for two instances. In order to declare several O-Telos objects about the same resource it is necessary to introduce an additional attribute "about" holding the URI of the

resource. Thus each O-Telos object describing a resource has its own ID as required by the O-Telos axioms and still can describe the same resource. A similar approach would have to be used in XML Schema, by the way.

The previous RDF example of the resource "<http://?/Agents/Definitions.htm>" is then declared in O-Telos by the following single frame:

```
Individual AgentDefinition1 in Topic with
  about
    a : "http://?/Agents/Definitions.htm"
  language
    l : "german"
  coverage
    c : "Introductory course"
  rights
    r : "KBS"
  parentUnit
    pu : online:#LectureUnit1
end
```

In order to represent the above object AgentDefinitionen1 by two frames the about-attribute is used. Note that the about attributes of both instances hold the same value <http://?/Agents/Definitions.htm>, but the Ids are different.

```
Individual AgentDefinition1 in Topic with
  about
    a : "http://?/Agents/Definitions.htm"
  language
    l : "german"
  coverage
    c : "Introductory course"
end
```

```
Individual AgentDefinition2 in Topic with
  about
    a : "http://?/Agents/Definitions.htm"
  rights
    r : "KBS"
  parentUnit
    pu : LectureUnit1
end
```

Using this approach it is possible to declare various objects about the same resource in the same model. Because O-Telos does not have a feature like the namespace declaration of RDF it is not yet possible to declare objects about the same resource in different models.

2.5 Sequences in RDF and O-Telos

We have mentioned before that our approach does not take sequences of attributes into account. As an example we use the following RDF declaration of the resource "LectureUnit1" which we will translate to O-Telos. "LectureUnit1" defines a sequence for values of the olr:child property. Its RDF declaration is given below:

```
<rdf:RDF xml:lang="en"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:olr="http://.../rdf/olr_schema_5#"
  xmlns:dc="http://purl.org/dc/elements/1.0#">

  <rdf:Description ID="LectureUnit1">

    <rdf:type resource="http://?/rdf/olr_schema_5#Unit"/>
    <dc:title>Lecture Unit 1</dc:title>
    <dc:description>Introduction to intelligent agents</dc:description>
    <olr:parentCourse rdf:resource="#AllLecture"/>
```

```

    <olr:child>
      <rdf:Seq>
        <rdf:li rdf:resource="http://.../Agents/Definitions.htm" />
        <rdf:li rdf:resource="http://.../Agents/Characterisation.htm" />
        <rdf:li rdf:resource="http://.../Agenten/Structur.htm" />
        <rdf:li rdf:resource="http://.../Agenten/Types.htm" />
      </rdf:Seq>
    </olr:child>
  </rdf:Description>
</rdf:RDF>

```

In the above example the order of resources of the property `olr:child` is defined with the container object RDF sequence (`rdf:Seq`). This order is used for the visualisation of the course hierarchy. While it is a convenient way to represent sequences, it is conceptually questionable, as `rdf:seq` is used as range of `olr:child`, instead of the more explicit ranges describing the specific type of the child resource (like `topic`, `example`, `slide`, etc. which we use in OLR).

O-Telos does not define such a construct for stating sequences. Instead an ordered list can be used to explicitly represent a sequence. Its declaration results in difficult to read and unnecessary complex O-Telos frames. But O-Telos also represents sequences implicitly by the order of the attributes in the O-Telos frames. Of course it is not insured that each application of O-Telos interprets the frames in the same way so that the attribute order (the sequence) might vary from an application to another.

If we want to state our RDF example without using RDF sequence but still represent sequences, we could use an attribute `"ordinal"` for the RDF-statements representing the sequence of the property values. These statements then look like:

`<oid , ordinal , i>`, with `i:integer` and `oid:ID` is the ID of a statement `<s , p , o>` with `s:subject`, `p:predicate` and `o:object`.

In other words we need the possibility to make statements about statements, e.g. by referring to the IDs of statements in statements. According to the RDF Model and Syntax Specification statements do not have IDs. Instead the specification introduces the declaration of higher-order statements which are some kind of statements about statements:

`<s , p , o , t>` with `s:subject`, `p:predicate`, `o:object` and `t:type`.

Applied to our example this could be written as follows:

```

<olr:Unit rdf:ID="LectureUnit1">
  <olr:topic rdf:resource="http://.../Agents/Definitions.htm" />
  <olr:topic rdf:resource="http://.../Agents/Characterisation.htm" />
</olr:Unit>

<rdf:Description>
  <rdf:subject resource="#LectureUnit1" />
  <rdf:predicate resource="http://.../topic" />
  <rdf:object rdf:resource="http://.../Agents/Definitions.htm" />
  <rdf:type resource="http://.../22-rdf-syntax-ns#Statement" />
  <olr:ordinalNo>1</olr:ordinalNo>
</rdf:Description>

<rdf:Description>
  <rdf:subject resource="#LectureUnit1" />
  <rdf:predicate resource="http://.../topic" />
  <rdf:object rdf:resource="http://.../Agents/Characterisation.htm" />
  <rdf:type resource="http://.../22-rdf-syntax-ns#Statement" />
  <olr:ordinalNo>2</olr:ordinalNo>
</rdf:Description>

```

This kind of modelling sequences resembles the way how sequences could be modelled in O-Telos as well. Of course the disadvantage is the lost simplicity of the model and a rather complex und unreadable declarations.

In contrast, the RDF example of the resource "LectureUnit1" is declared without sequences as follows:

```
<rdf:Description ID="LectureUnit1">
  <rdf:type resource="http://.../olr_schema_6#Unit"/>
  <dc:title>Lecture Unit 1</dc:title>
  <dc:description>Introduction to intelligent agents</dc:description>
  <olr:parentCourse rdf:resource="#AIIecture"/>
  <olr:child rdf:resource="http://.../Agents/Definitions.htm"/>
  <olr:child rdf:resource="http://.../Agents/Characterisation.htm"/>
  <olr:child rdf:resource="http://.../Agents/Structur.htm"/>
  <olr:child rdf:resource="http://.../Agents/Types.htm"/>
</rdf:Description>
```

Using the O-Telos frame syntax the above example is stated below in O-Telos:

```
Individual LectureUnit1 in olr_unit with
  dc_title
    t1: "Lecture Unit 1"
  dc_description
    d1: "Introduction to intelligent agents"
  olr_parent_course
    pc1: AIIecture
  olr_child
    c1: "http://.../Definitions.htm";
    c2: "http://.../Characterisation.htm";
    c3: "http://.../Structur.htm";
    c4: "http://.../Types.htm"
end
```

The individual "LectureUnit1" is declared as instance of the class olr_unit because of the rdf:type property in the RDF declaration. In order to instantiate the properties dc:title, dc:description, olr_parent_course and olr_child the class olr_unit has to define these attributes. Note that the class olr_course has to be declared also in order to have a valid class declaration according to O-Telos.. So far we have discussed how RDF declarations can be mapped to O-Telos declarations. We will now compare O-Telos and RDF metadata on the tuple level

2.6 Comparing RDF and O-Telos on the tuple level

Let us now compare the declarations of RDF and O-Telos by comparing their triple respective quadruple representations. As mentioned before RDF declarations can be represented as triples. A RDF triple has the definition:

$\langle s, p, o \rangle$ with s:subject, p:predicate and o:object, reading: there is a property p from subject s to object o.

O-Telos declarations can be represented by quadruples which we call propositions. In general the propositions are defined as a relationship:

$P(oid, x, l, y)$ with oid:objectID, x:source, l:label, y:destination reading: it exists an object with oid stating a relationship called l from object x to object y.

In order to get comparable triples and propositions we translate the OLR schema to O-Telos. The comparison of the RDF triples and the O-Telos propositions of the OLR schema shows that the O-Telos propositions are basically equal to the RDF triples. Interestingly, usual RDF triples are expressed by two O-Telos propositions for each triple.

The following example declaration about the resource "http://.../Agents/Characterisation.htm" shows how we translated RDF declaration to O-Telos and that propositions and triples are equal.

```
<rdf:Description about="http://.../Agents/Characterisation.htm">
  <rdf:type resource="http://.../rdf/olr#Topic"/>
  <dc:title>Characterisation of agents</dc:title>
  <olr:parentUnit rdf:resource="#LectureUnit1"/>
```

</rdf:Description>

The RDF declarations defines three properties to the resource "http://.../Agents/Characterisation.htm". The rdf:type property defines the resource as of type olr#Topic while dc:title states the name of the resource and olr:parentUnit defines the resource LectureUnit1 as parentUnit.

Nr.	Subject	Predicate	Object
1	http://.../Agents/Characterisation.htm	http://.../22-rdf-syntax-ns#type	http://.../rdf/olr#Topic
2	http://.../Agents/Characterisation.htm	http://purl.org/dc/elements/1.0#title	Characterisation of agents
3	http://.../Agents/Characterisation.htm	http://.../rdf/olr_schema_7#parentUnit	online:#LectureUnit1

The above table states the three RDF triples of the declaration of "http://.../Agents/Characterisation.htm". The triples show explicitly that all three properties belong to the resource, while the predicates state name and the accompanying namespace of the properties.

The O-Telos frame representation of the example is introduced below, we will use the URI as the ID for the corresponding individual.

```

Individual "http://.../Agents/Characterisation.htm" in Topic with
  dc_title
    t1 : "Characterisation of agents"
  parentUnit
    pu1 : LectureUnit1
end

```

The frame declares the object "http://.../Agents/Characterisation.htm" as instance of class Topic similarly to the rdf:type property of the RDF declaration. The other two attributes dc_title and parentUnit correspond to the respective properties.

oid	source	Label	Destination
#1	#1	„http://.../Agents/Characterisation.htm“	#1
#2	#1	*instanceof	#Topic
#3	#1	t1	"Characterisation of agents"
#4	#3	*instanceof	#dc_title
#5	#1	pu1	#LectureUnit1
#6	#5	*instanceof	#parentUnit

The above table states the O-Telos propositions of the declaration of "<http://.../Agenten/Charakterisierung.htm>". The proposition oid #1 represents the object "http://?" while proposition #2 states that the object is instance of class #Topic. The proposition #3 declares that the object from #1 has an attribute t1 with value "Characterisation of agents" while proposition #4 declares the attribute t1 from #3 as instance of #dc_title. The proposition #5 declares that the object from #1 has an attribute pu as a reference to #LectureUnit1 while proposition #6 declares the attribute from #5 as instance of #parentUnit.

O-Telos requires that the declaration of the attributes t1 and pu1 is included in the class Topic from which this object is an instance.

The comparison of the above example of the declaration of the resource "<http://.../Agenten/Charakterisierung.htm>" in RDF and O-Telos shows that they are equivalent. Especially the comparison of the RDF-triple representation with the O-Telos propositions emphasizes that RDF triples can simply be mapped to O-Telos propositions. The comparison reveals that two O-Telos propositions are used to represent one RDF triple, because in O-Telos, properties are explicitly instantiated using an additional tuple, while this information in RDF is contained in the property name, which includes the type as a prefix.

However, we have no inbuilt possibility to represent RDF namespace information in the O-Telos proposition, as it is built on the declaration of schema and metadata in one file. The introduction of an additional attribute "namespace" to the O-Telos attribute relationships can in some sense address this problem, e.g. by declaring a proposition #7 stating that #3 has namespace <http://purl.org/dc/elements/1.0#title> and proposition #8 stating that #5 has namespace http://.../rdf/olr_schema_7#parentUnit. However, RDF is definitely more suited to distributed declarations than O-Telos.

oid	source	Label	Destination
#7	#3	namespace	http://purl.org/dc/elements/1.0#title
#8	#5	namespace	http://.../rdf/olr_schema_7#parentUnit

3 Conclusion and future work

This paper discussed the use of RDF metadata in our open learning repository system OLR, as well as its underlying architecture. We are currently extending this system by different navigation schemes and are working on making it still easier to modify/extend metadata and metadata schemas in/with OLR. To support LOM metadata annotation of a large amount of (often hierarchically related) document pages, we will have to add some inferencing capabilities which for example allow (default) inheritance of LOM attributes along the LOM isPartOf relation. A further extension will be P2P exchange functionality between distributed OLR systems.

In the second part we have discussed RDF/RDFS modelling with the conceptual modelling language O-Telos, and analysed similarities as well as differences, in the course of this analysis discussing some design properties of RDF/RDFS from a conceptual modelling standpoint. Further work will be done on the applicability of the O-Telos query language to RDF-metadata.

This work has profited much from several discussions with our colleagues, and we want to thank especially Changtao Qu for his comments on several of the issues discussed in this paper.

4 References

[Ashrafuzzaman, 1996]

M. Ashrafuzzaman: Deductive Object-Oriented Database for Geographic Data Handling. Course project CMPT826, University of Saskatchewan, Canada, March 1996

[DC, 2001]

Dublin Core Initiative, 2001, <http://dublincore.org/>

[Gamper et.al., 1999]

Johan Gamper, Wolfgang Nejdl and Martin Wolpers: Combining Ontologies and Terminologies in Information Systems. *5th International Congress on Terminology and Knowledge Engineering*, Innsbruck, Austria, August 1999, <http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/1999/tke99/index.html>

[Jeusfeld, 1992]

M. Jeusfeld, *Änderungskontrolle in deduktiven Objektbanken*, Infix-Verlag 1992, St. Augustin, Germany

[Jeusfeld et.al., 1998 A]

M. A. Jeusfeld, M. Jarke, H. W. Nissen and M. Staudt, ConceptBase - Managing Conceptual Models about Information Systems, in *Handbook on Architectures of Informations Systems*, P. Bernus, K. Mertins and G. Schmidt (eds.), Springer Verlag 1998

[Jeusfeld et.al., 1998 B]

M.A. Jeusfeld, C. Quix, M. Jarke: Design and Analysis of Quality Information for Data Warehouses. in *Proc. 17th International Conference on Conceptual Modeling (ER'98)*, Singapore, Nov 16-19, 1998

[Melnik, 2000]

Sergey Melnik, Storing RDF in a relational database, 2000, <http://www-db.stanford.edu/~melnik/rdf/db.html>

[Nejdl. et.al., 2000]

Nejdl, M. Wolpers, C.Capelle, The RDF Schema Specification Revisited, *Modellierung 2000*, 5. - 7.4.2000, St Goar, Germany, <http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2000/modeling2000/wolpers.pdf>

[Nejdl and Wolpers, 1999]

Wolfgang Nejdl and Martin Wolpers: KBS Hyperbook - A Data-Driven Information System on the Web. *WWW8 Conference*, Toronto, May 1999, <http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/1999/www8/index.html>

[Saarela, 2001]

Janne Saarela, SiRPAC - Simple RDF Parser & Compiler World Wide Web Consortium (W3C), 2001, <http://www.w3.org/RDF/Implementations/SiRPAC/>

[VRP, 2001]

The Validating RDF Parser (VRP), Institute of Computer Science - Foundation of Research Technology Hellas - Greece (ICS-FORTH), 2001, <http://139.91.183.30:9090/RDF/VRP/index.html>

[W3C, 1999]

RDF Model and Syntax Specification, World Wide Web Consortium (W3C), February 1999, <http://www.w3.org/TR/REC-rdf-syntax/>

[W3C, 2000]

RDF Schema Specification, World Wide Web Consortium (W3C), March 2000, <http://www.w3.org/TR/2000/CR-rdf-schema-20000327/>

[W3C, 2001]

Semantic Web Activity Statement, World Wide Web Consortium (W3C), 2001, <http://www.w3.org/2001/sw/Activity>