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. Building repositories for e-learning is an iterative process and course content and course structure are always changing. We realized the necessity to separate content from structure of a given course during the conception of our first e-learning repository, which we called KBS-Hyperbook, several years ago at our institute. This system has been built around a conceptual model for structure and contents of the domain, which is expressed in the O-Telos conceptual modelling language. To ease exchange of metadata between such repositories, the Open Learning Repository (OLR), an e-learning repository we built during the last year to experiment with various features useful for such repositories, has been developed using RDF/RDFS as modelling language.
In the first part of this paper, we describe the OLR system in more detail, and show how it uses RDF/RDFS as its underlying modelling language to express information about the learning objects contained in the repository, as well as information about the relationships between these learning objects. Based on our experience in meta-modelling using different modelling languages, we will in the second part of this paper discuss RDF/RDFS and O-Telos modelling in more depth and will analyse similarities and differences of these two modelling languages.

Keywords
Meta-modelling, RDF/RDFS, conceptual modelling, hypermedia, learning repositories.

1 The Open Learning Repository

1.1 Motivation

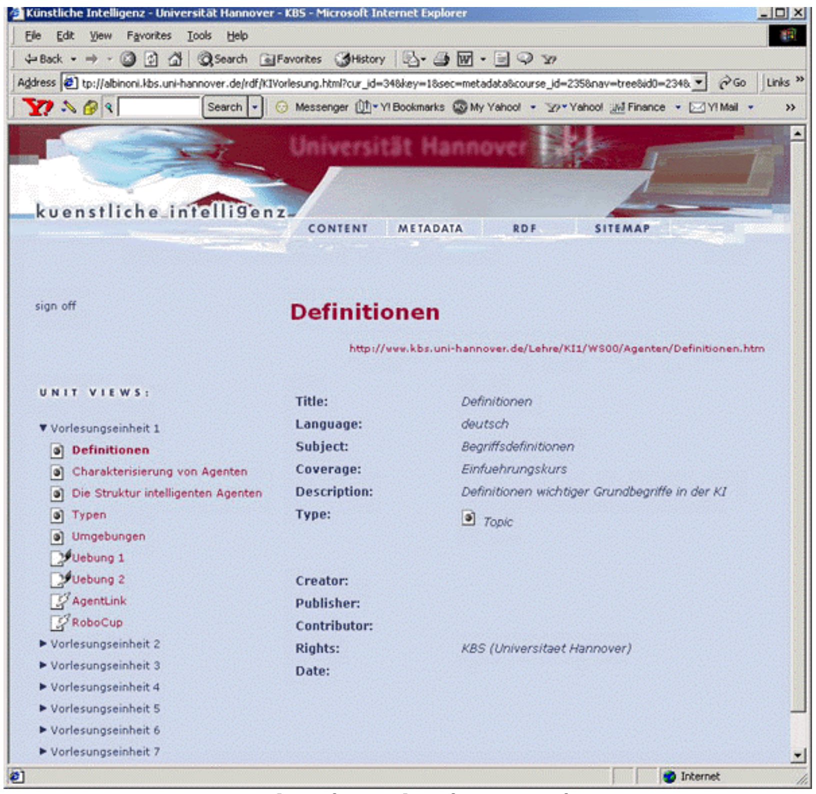
Our Open Learning Repositories aim at metadata-based course portals, which structure and connect modularised course materials over the Web. The modular content can be distributed anywhere on the internet, and is integrated by explicit metadata information in order to build courses and connected sets of learning materials. Modules can be reused for other courses and in other contexts, leading to a course portal which integrates modules from different sources and authors. Semantic annotation is necessary for authors to help them choose modules and to connect them into course structures.

We use a relational database to store all metadata, but store no content in the database itself. The stored metadata represent information about the structure and the access paths within a particular course, the URLs as identifiers for 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 currently 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) [1] metadata from arbitrary RDF schemas. However, we have chosen not to implement a one-size-fits-all approach, and follow a customisable approach, implementing different interfaces together with their schemas and metadata for different courses using a common infrastructure. Initial loading for a specific course is done by importing an RDF metadata file (using XML syntax) based on this course's RDFS [2] 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 [3] set. We are currently moving these metadata to the LOM standard, using the recently developed LOM-RDF-binding.

The web interface for navigating the course follows 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


Künstliche Intelligenz - Universität Hannover - KBS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address tp://albinoni.kbs.uni-hannover.de/rdf/KIVorlesung.html?cur_id=34&key=1&sec=metadata&course_id=235&nav=tree&id0=234&
Go Links
Search Messenger Y! Bookmarks My Yahoo! Yahoo! Finance Y! Mail
Universität Hannover
kuenstliche intelligenz
CONTENT METADATA RDF SITEMAP
sign off
Internet

Definitionen

http://www.kbs.uni-hannover.de/Lehre/KI1/WS00/Agenten/Definitionen.htm

UNIT VIEWS:

- ▼ Vorlesungseinheit 1
  - Definitionen
  - Charakterisierung von Agenten
  - Die Struktur intelligenten Agenten
  - Typen
  - Umgebungen
  - Uebung 1
  - Uebung 2
  - AgentLink
  - RoboCup
- ▶ Vorlesungseinheit 2
- ▶ Vorlesungseinheit 3
- ▶ Vorlesungseinheit 4
- ▶ Vorlesungseinheit 5
- ▶ Vorlesungseinheit 6
- ▶ Vorlesungseinheit 7

Title:	*Definitionen*
Language:	*deutsch*
Subject:	*Begriffsdefinitionen*
Coverage:	*Einfuehrungskurs*
Description:	*Definitionen wichtiger Grundbegriffe in der KI*
Type:	*Topic*
Creator:	
Publisher:	
Contributor:	
Rights:	*KBS (Universitaet Hannover)*
Date:	

Figure 1: 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 (figure 5). 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 (figure 1).

For content developers we implemented an enhanced web interface which allows the developer to manipulate metadata through HTML forms (figure 1). The OLR system translates all user input into suitable SQL update and insert statements hence avoiding to confront the user with having to understand XML/RDF notation. To evaluate OLR usage, the system tracks all user behaviour in the database, including which course elements are accessed and when, which updates are made and by whom. We are using this information to evaluate different navigation schemes and different types of course units.

1.3 OLR-Architecture

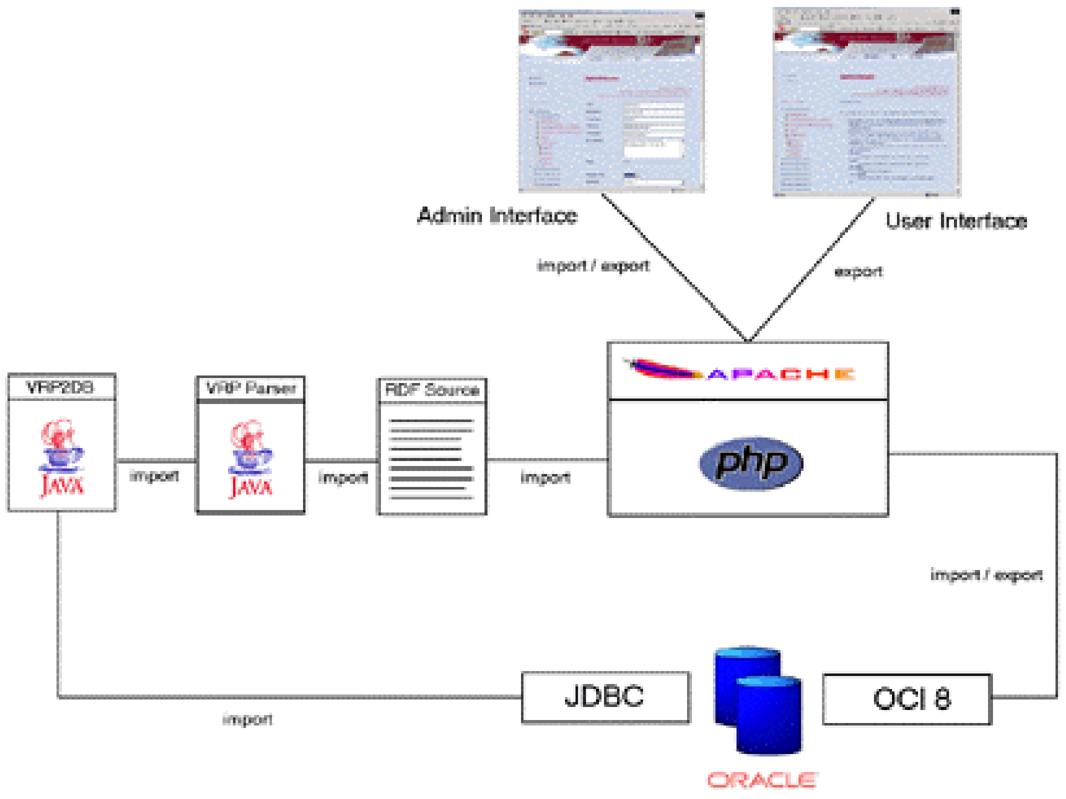
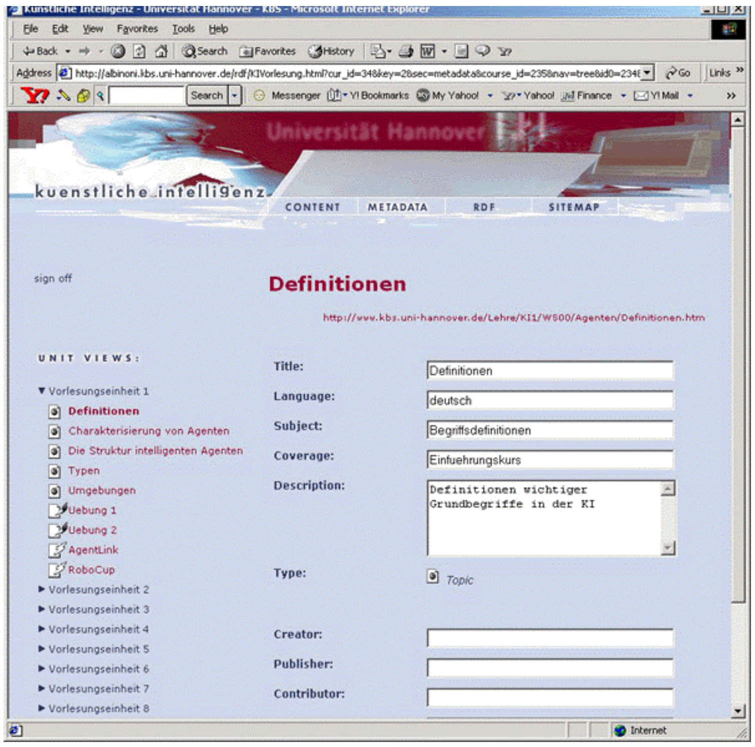


Figure 2: OLR Architecture

The OLR architecture is shown in figure 2. The system is based on a 3-tier architecture. As front end any state-of-the-art web browser may be used (IE5, NS4). The mid-tier is a combination of Apache Web server and PHP4 module. The backend holds an Oracle 8i database and can physically be the same machine as the one running the Web server.

Whenever the user selects a link or button Apache delegates the client request to the PHP module executing the appropriate PHP script. In most cases this script will need to interact with the database since it stores all RDF metadata. For communication with Oracle PHP uses its built-in OCI8 interface. The PHP script evaluates the data returned by Oracle and dynamically creates a HTML page which in turn is sent back to the client browser initially requesting the page.

In addition the web interface allows to upload raw RDF source code in XML syntax to be stored in temporary files within the server's file system. A shell script then runs the VRP parser [4] against these RDF metadata. The generated triples are input to a Java application using the JDBC interface which imports all statements into the database.


Kunstliche Intelligenz - Universitat Hannover - KBS - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://albinoni.kbs.uni-hannover.de/rdf/KIVorlesung.html?cur_id=34&key=2&sec=metadata&course_id=235&nav=tree&id0=234
Go Links
Search Messenger Y! Bookmarks My Yahoo! Yahoo! Finance Y! Mail
Universität Hannover
kuenstliche intelligenz
CONTENT
METADATA
RDF
SITEMAP
sign off
Internet

Definitionen

http://www.kbs.uni-hannover.de/Lehre/KI1/WS00/Agenten/Definitionen.htm

UNIT VIEWS:

- Vorlesungseinheit 1
  - Definitionen
  - Charakterisierung von Agenten
  - Die Struktur intelligenten Agenten
  - Typen
  - Umgebungen
  - Uebung 1
  - Uebung 2
  - AgentLink
  - RoboCup
- Vorlesungseinheit 2
- Vorlesungseinheit 3
- Vorlesungseinheit 4
- Vorlesungseinheit 5
- Vorlesungseinheit 6
- Vorlesungseinheit 7
- Vorlesungseinheit 8

Title:	Definitionen
Language:	deutsch
Subject:	Begriffsdefinitionen
Coverage:	Einfuehrungskurs
Description:	Definitionen wichtiger Grundbegriffe in der KI
Type:	Topic
Creator:	
Publisher:	
Contributor:	

Figure 3: 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 metadata in order to facilitate modular and reusable content repositories.

One of the practical uses of RDF, as it has been described by W3C, is in Web sitemaps. "*The RDF schema specification provides a mechanism for defining the vocabulary needed for this kind of application*" [5].

Thus, with RDF, we can describe for our application, how modules, course units, courselets are related to each other or which examples or exercises belong to a course unit, RDF metadata used in this way are called structural or relational metadata. Another practical use of RDF is the description of web pages/units, which is mandatory to build a course based on modular content, distributed over different sites. To standardize these kinds of descriptions, initiatives like IMS and IEEE LOM specify schemas suitable for learning objects, and we have been involved in the German LOM version as well as in a LOM-RDF-binding suitable for these learning objects.

RDF (Resource Description Framework) is supported by a growing Web community. The primary target of RDF is to provide a standardized way of creating and using such specialized metadata schemas to describe resources on the Web.

Some of the goals the W3C aims to reach using RDF are:

- Resource Discovery to improve the results of Search Engines.
- Cataloguing to describe content and its relationships at a particular Web.
- Interoperability and Knowledge Sharing for information exchange between different applications, Software Agents etc.
- Logical Document: Several pieces of content physically distributed over the Internet build one single Logical Document, where RDF is the glue holding these resources together.

Everything in RDF is expressed through statements, which are triples consisting of subject, predicate and object (corresponding to instantiated binary predicates). Expressing the sentence “Smith is the author of the HTML document that can be found at the URL “http://www.xyz.com/somedoc.html”” for example is done by a statement, where “http://www.xyz.com/somedoc.html” is the subject of our statement, its predicate is “author” (which is a property in RDF terminology) and its object is the literal “Smith”. Another possibility would be to use a resource (with an URL) as the object of such a statement, like “http://www.xyz.com/smith.html”, assuming we want to use this URL as identifier for the person Smith.

This simple example reveals the basic building blocks of any RDF statement: resources and literals. Anything that can be reached by a URL is a resource whereas a literal is a simple character string. Subjects and predicates always need to be resources while an object may be either resource or literal. In addition predicates normally are properties described by an RDF schema.

The RDF specification does not insist on any implementation of the statement concept in particular. It introduces a graph representation suitable for the human reader and an XML-encoding of that graph suitable for XML based parsers. The XML encoding is probably the most popular RDF representation.

To create self-defined predicates like “author” in our example, one needs to create an RDF schema. Like RDF metadata these RDF schemas consist of statements and hence can be expressed utilizing the same XML syntax or any other representation.

With RDF Schema resources can be modelled as classes and predicates as properties. Thus it is possible to constrain the type of a predicate’s range and domain. For example we can say that the predicate *author* may only point to resources that are instances of a class *Person* and may only be applied to resources being instances of a class *Book*.

Since we decided to utilize RDF in OLR for both the annotation of content as well as the description of course structures we developed an RDF schema for this purpose. Our implementation focuses on the cataloguing/annotation and on the logical document features of RDF. An OLR course is a Logical Document and cataloguing is used to store element information (e.g. title, author).

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 by metadata. The vocabulary describing each element is basically the Dublin Core Metadata set.

We currently use RDF sequences to link elements to units and units to courses. This is necessary because the order of the course elements is essential. The disadvantage of this is, that in the current version of RDF Schema it is not possible to constrain the type of container elements. In the second part of the paper we include several examples, which use stronger typing constraints instead of RDF sequences.

Database Schema

In essence, everything in RDF is expressed through statements: simple triples composed of resources, namespaces and literals - no matter how complex the RDF schema behind might be. XML syntax is the standard approach for hiding RDF in HTML pages it describes. This approach always requires a parser to analyse the meta-information and it conflicts with one of RDF’s key concepts where a group of RDF statements makes propositions about several distributed resources linking them together to one Logical Document.

In contrast, using triples directly makes it easy to store RDF metadata in a relational database. Doing so enables us to create a repository for metadata managed at one central location using relational database technology. This approach separates the metadata from the content it describes. SQL queries are used to extract the relevant RDF statements.

An obvious advantage of storing RDF in a relational database is performance: A SQL query selecting a couple of statements can be much faster than parsing an RDF document in XML representation to retrieve the same results. Especially when a lot of similar queries are executed the database’s query optimiser and cashing mechanisms can speed things up considerably. When looking at large numbers of statements compact storage is another plus for the database approach: Within a set of RDF metadata a lot of literals tend to occur more than once. Namespaces are a good example for this characteristic: Every resource name is preceded by a namespace and often these namespaces are similar or identical. Being kept in a separate table, each namespace needs to be stored in the database only once. For multiple usage any namespace only needs to be referenced by its ID.

For our OLR server, we modified the McBride schema, which is one of several suggestion presented on the RDF/DB Page from Sergey Melnik [6] , also discussed within the RDF community. The OLR system is based on the Oracle 8i database, but any standard relational database would be suitable.

The main table in our database is RDF_STATEMENT. This table represents the relationship between the three parts of a statement consisting of RESOURCE (stored in RDF_RESOURCE), PREDICATE (also stored in RDF_RESOURCE) and OBJECT (stored in either RDF_RESOURCE or RDF_LITERAL). Therefore RDF_STATEMENT contains three main attributes: SUBJECT, PREDICATE and OBJECT. These attributes are references to the resource and the literal table. Since the object can either be 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 establish different user groups with different roles and rights. Every group may have a specific view on courses and metadata.

Hence we define a table RDF_USER 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 all dynamic content is created based on 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 4.

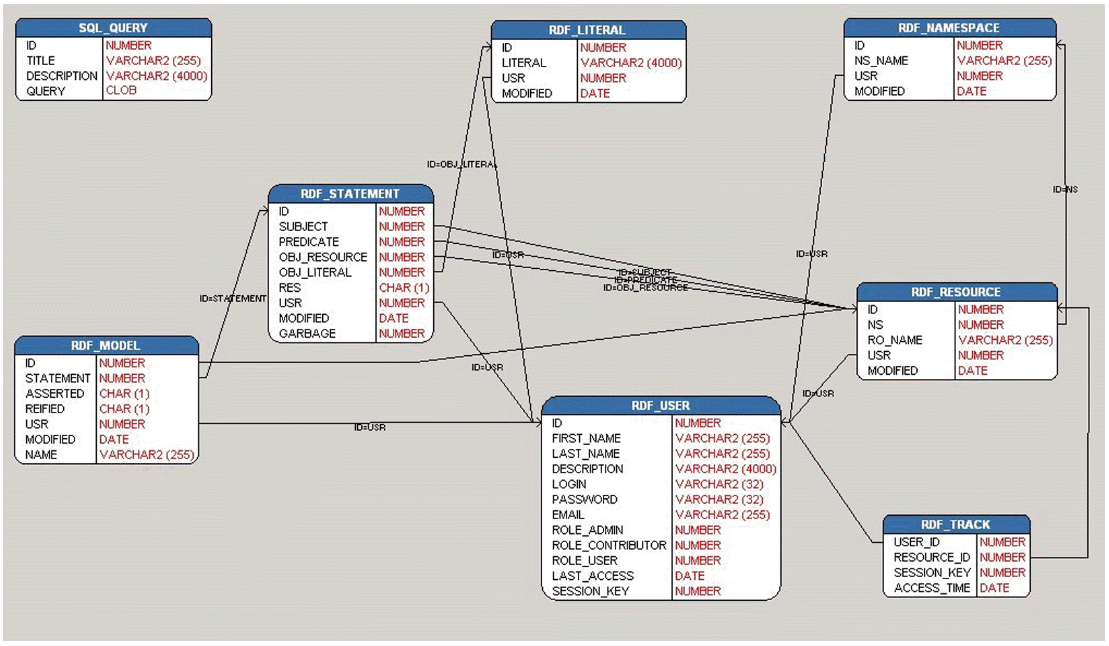


Figure 4: Database Schema

1.4.4 Architecture and Features of the OLR Web Interface

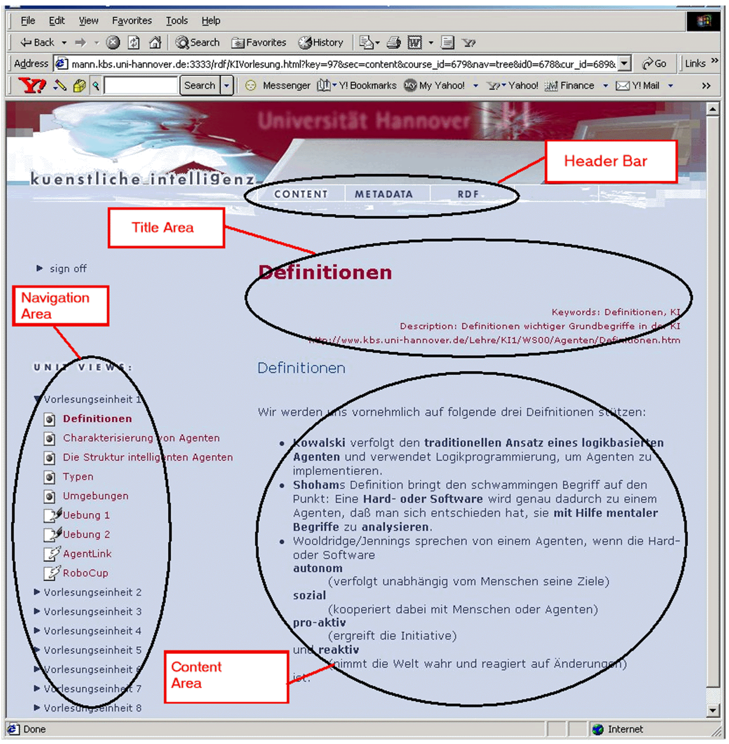
The Web interface for browsing and manipulating OLR courses needs to be highly dynamic since it needs to take into account the current state of the database. For this reason all HTML code is generated on demand by PHP scripts. To control the complexity of the system the PHP scripts are organized in several layers. Structure and purpose of the different layers are briefly outlined below.

Database access with PHP is straightforward: It already comes with a built-in API for communicating with an Oracle database through the standard OCI8 interface. We designed a number of SQL queries to suit the special needs of the OLR system. These queries are stored in a database table SQL_QUERY itself along with a unique ID, a query name and a short text describing the query's purpose. This approach greatly enhances maintainability and transparency of the system. Queries may contain parameters like resource IDs specified in brackets.

Core of the code for running SQL queries is the PHP class *RDFStatement*. Its constructor requires the query name and eventually a number of parameters. *RDFStatement* then executes the query and transfers all results into a PHP array. All database specific code is hidden behind the public interface of this class.

On top of the *RDFStatement* class we develop the OLR API - a growing number of PHP functions like *getResourceTitle(resource_id)* that take some resource ID as in-parameter and retrieve all statements about the specified resource for a specific property. These *getResourceXXX()* - functions utilize the *RDFStatement* class. Note that database primary keys (usually integer values) serve as in-parameters to identify resources rather than a combination of namespace and literal which tends to be long strings. This is extremely useful for our web interface since it keeps track of all state information (e.g. current course, unit or element) by URL parameters. The OLR API is accompanied by a number of other APIs such as an API for user and session management and an API for import and export of RDF source in XML syntax.

The next layer consists of a number of basic building blocks – PHP script fragments calling API functions and performing the HTML markup of the returned results. For instance there are PHP blocks for creating the different navigation elements or for displaying content or metadata of a course element. The final abstraction layer is represented by templates. In essence templates are HTML files composed by dynamically putting together the basic building blocks. Most templates follow the same structure with a navigation element on the left, a content area on the right and above that a header section displaying title and essential metadata. The templates also verify user access rights.



Definitionen

Keywords: Definitionen, KI
Description: Definitionen wichtiger Grundbegriffe in der KI
http://www.kbs.uni-hannover.de/Lehre/KI1/WS00/Agenten/Definitionen.htm

Definitionen

Wir werden uns vornehmlich auf folgende drei Deifnitionen stützen:

- **Kowalski** verfolgt den **traditionellen Ansatz eines logikbasierten Agenten** und verwendet Logikprogrammierung, um Agenten zu implementieren.
- **Shoham**s Definition bringt den schwammingen Begriff auf den Punkt: Eine **Hard- oder Software** wird genau dadurch zu einem Agenten, daß man sich entschieden hat, sie **mit Hilfe mentaler Begriffe** zu **analysieren**.
- Wooldridge/Jennings sprechen von einem Agenten, wenn die Hard- oder Software
  autonom
  (verfolgt unabhängig vom Menschen seine Ziele)
  sozial
  (kooperiert dabei mit Menschen oder Agenten)
  pro-aktiv
  (ergreift die Initiative)
  und **reaktiv**
  (nimmt die Welt wahr und reagiert auf Änderungen)
  ist.

Figure 5: OLR sample template

The structure of the templates directly supports our multi-view vision. If for example you want to use a trail instead of a hierarchical navigation only one line of code needs to be changed in the appropriate template to replace inclusion of the hierarchy-block by the trail-block.

Though all RDF data are stored in the database tables the content contributor's web interface allows direct import and export of RDF source in XML syntax. After inserting XML code describing an OLR course by copy&paste into an HTML form its content is uploaded to the server, analysed by the VRP parser and then imported into the database by a java application through the Oracle JDBC interface. A newly imported course then appears in the list of available courses. Without any knowledge of RDF or the specifics of the underlying OLR schema another content contributor then has the opportunity to modify the course through clearly arranged HTML forms. The system allows to create, modify or delete course elements and units. This is one conceptual advantage of the combination database plus web interface over the standard approach of hiding some static XML RDF within an HTML file: A authorized subgroup has the opportunity to dynamically change and extend the content of the repository through an intuitive interface and the database always keeps track of who did what and at which time modifications where issued. In addition it is possible to export XML RDF metadata on any level of granularity: One can export the XML RDF for a single course element, a unit including all its elements and subunits or a complete course. This feature is beneficial for reuse when creating new courses and supports metadata processing by other RDF XML compatible applications.

2 Comparing RDF/RDFS to the O-Telos modelling language

2.1 Motivation

As noted above, RDF is a simple but quite powerful modelling language to annotate WWW resources with semantical information. RDFS enables the simple construction of conceptual models of sets of WWW resources, and on the other had has been designed as a quite flexible representation language for these conceptual models. Unfortunately, the RDF Schema Specification [2] fails to give simple, yet formal explanations of RDFS concepts, which causes a lot of confusion when one really tries to use all RDFS possibilities. RDFS tries to be as self-expressible as possible, which leads to several properties playing dual roles both as primitive constructs and as specific instances of RDF/RDFS properties (*rdfs:domain*, *rfds:range*, *rdfs:subClassOf*, and *rdf:type*, see also the detailed discussion in [7]), where these properties are both defined in the RDF or RDFS-Schema and are used to define those schemas at the same time. On the other hand, the self-expressibility of RDFS falls short of fulfilling its promise for meta-modelling, because of the constraints of the underlying triple model, only a three level modelling hierarchy is possible (*rdfs:class*, specific classes as instances of *rdfs:class*, and instances of classes).

Another drawback of RDFS is its poor support of the reification of statements. An object identifier must be assigned explicitly to each statement that is to be reified. This has to be done by adding explicit statements about the subject, predicate and object of the specific statement.

Building on our previous work on open learning repositories [8], [9], we will in this second part of the paper compare RDF/RDFS modelling and annotation with the conceptual modelling language O-Telos, which has been strictly axiomatized in [10], based on the formalization of Telos (see e.g. [15]). As a conceptual modelling language, O-Telos is used in various contexts to describe and formalize conceptual models [9], [11], [12], [13]. As for reification, O-Telos, being based on 4-tuples instead of triples, assigns a unique object identifier to each statement, which can be used to directly reference that statement.

In this paper, we will compare RDF/RDFS with O-Telos, and discuss possible mappings from RDF to O-Telos and back, which is useful in our context (making it possible to exchange metadata between our O-Telos- and RDF-Hyperbook Systems), and also sheds light on some advantages and disadvantages of the design decisions of RDFS. In [16] we formalize an RDF variant we call O-Telos-RDF based on the O-Telos model, which allows annotation in a way very similar to RDF, but extends RDFS with enhanced reification and meta-modelling capabilities.

2.2 An introduction to O-Telos

O-Telos is a deductive object-oriented conceptual modelling language very suitable for modelling and meta-modelling tasks. It has been implemented in the ConceptBase database system [12]. Its object-oriented constructs like object, class, meta-class, etc. are expressed using a frame syntax. Each frame declares an object by stating its name, the classes it subclasses, the classes it instantiates and the attributes it declares or instantiates.

Frames are declared using predefined classes: Individual containing all individuals as instances, Attribute containing all attributes as instances, Class containing all classes as instances, String, Integer, etc. The use of the predefined classes is defined by a set of axioms to insure referential integrity, correct instantiation and inheritance.

The following example is taken from a simplified version of the OLR schema. It is used to illustrate the O-Telos language:

The lecture material of a course consists of course units, which group the specific elements. All units/elements can be annotated according to Dublin Core, i.e. they have a name and a description etc..

The model of the above example declares the following O-Telos frames, which define the two classes course and course unit as well as a Dublin Core Class, and the corresponding attributes:

```
Class DC_Unit with
   attribute
      about: URL;
      title: String;
      description: String
end
Class Course isA DC_Unit with
   attribute
      units: CourseUnit
end
Class CourseUnit isA DC_Unit with
   attribute
      parent_course : Course;
      theory_unit: TheoryUnit;
      example_unit: Example
end
```

The frame *Course* declares a class named *Course* consisting of arbitrarily many units. A unit is declared by the frame *CourseUnit*, and groups *TheoryUnits*, *Examples*, etc. Both are subclasses of *DC_Unit*, stating that they can have a *title* and a *description*, both of type *String*.

The next frames declare the individuals, e.g. a course unit with the title “Lecture Unit 1”, the description “Introduction to Intelligent Agents”. This resource belongs to the course “Introduction to AI 1” which is an introductory course in Artificial Intelligence. Additionally, this resource belongs to another course “AI 2” which is an advanced course in Artificial Intelligence.

```
Individual IntroAILecture in Course with
  title
    t1 : "Introduction to AI 1"
  description
    d1 : "Introductory course in AI"
end
Individual AdvancedAILecture in Course with
  title
    t1 : "AI 2"
  description
    d1 : "Advanced course in AI"
end
Individual IntroAILectureUnit1 in CourseUnit with
  title
    t1 : "Lecture Unit 1"
  description
    d1 : "Introduction to Intelligent Agents"
  theory_unit
    tu1: "http://www.kbs.uni-hannover.de/.../Definitions.htm";
    tu2: "http://www.kbs.uni-hannover.de/.../Characterisation.htm"
  parent_course
    c1 : IntroAILecture;
    c2 : AdvancedAILecture
end
```

The frame *IntroAILectureUnit1* shows how the declared attributes *title*, *description, theory_unit* and *parent_course* are instantiated. The *theory_unit* and *parent_course* attributes show that O-Telos attributes usually are multi-valued.

The frames are translated to sets of propositions which can be stored e.g. 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 defines specific 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 P(oid,x,l,y) represents ordinary attributes: x has an attribute named l with value y.

2.3 Simple mapping of RDF to O-Telos

Let us now construct a simple mapping from RDF to O-Telos and vice versa. We will recognize, that both languages are based on very similar ideas for their basic representation.

We start with a simple RDF 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="#AILecture"/>
   <olr:theoryUnit rdf:resource="http://.../Agents/Definitions.htm"/>
   <olr:theoryUnit rdf:resource="http://.../Agents/Characterisation.htm"/>
   <olr:theoryUnit rdf:resource="http://.../Agents/Structure.htm"/>
```

```
  <olr:theoryUnit rdf:resource="http://.../Agents/Types.htm"/>
</rdf:Description>
```

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

```
Individual LectureUnit1 in CourseUnit with
   dc_title
     t1: "Lecture Unit 1"
   dc_description
     d1: "Introduction to intelligent agents"
   parent_course
     pc1: AILecture
   theory_unit
     tu1: "http://www.kbs.uni-hannover.de/.../Definitions.htm";
     tu2: "http://www.kbs.uni-hannover.de/.../Characterisation.htm";
     tu3: "http://www.kbs.uni-hannover.de/.../Structure.htm";
     tu4: "http://www.kbs.uni-hannover.de/.../Types.htm"
end
```

The 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. Both representations require the declarations of the objects/classes *Unit/olr_unit* and the course *AILecture*.

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 sometimes include additional 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* (see also the discussion in [7]). In RDF an *annotation cla*ss 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 objects can be mapped to O-Telos constructs only if there is no other annotation object stating some attribute about the same resource. Because the O-Telos object takes the URI as its unique ID and all other attributes are referenced as above. 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 mentioned, resources which are described by the RDF declaration <rdf:Description about="http://..."> have no ID property. They are just groupings of attributes, as the following example shows:

```
<rdf:Description about="http://.../Agents/Definitions.htm">
  <rdf:type resource="http://.../rdf/olr#TheoryUnit"/>
    <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".

Table 1. RDF-Triples of the single declaration about "http://.../Agents/-Definitions.htm"

Nr.	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#TheoryUnit
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

Table 1 shows the RDF-triples representing the RDF declaration of properties to the resource "http://.../Agents/Definitions.htm". The triples are generated by the SIRPAC [14] parser.

The above example can also be expressed in two separate RDF declarations about the resource "http://.../Agents/Definitions.htm". Both declarations assign values to attributes but represent two different grouping objects.

```
<rdf:Description about="http://.../Agents/Definitions.htm">
  <rdf:type resource="http://.../rdf/olr#TheoryUnit"/>
    <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#TheoryUnit"/>
    <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>
```

Table 2. RDF-Triples of one the multiple declarations about "http://.../Agents/-Definitions.htm"

Nr.	**Subject**	**Predicate**	**Object**
1	http://.../Agents/Definitions.htm	http://www.w3.org/1999/02/22-rdf-syntax-s#type	http://.../rdf/olr#TheoryUnit
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 A
4	http://.../Agents/Definitions.htm	http://purl.org/dc/elements/1.0#subject	Definitions

The number of triples = 4

Table 3. RDF-Triples of another of the multiple declarations about "http://.../Agents/-Definitions.htm"

Nr.	**Subject**	**Predicate**	**Object**
1	http://.../Agents/Definitions.htm	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://.../rdf/olr#TheoryUnit
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.

The above two tables Table 2 and Table 3 contain the RDF triples for the two separate RDF declarations of attributes to "http://.../Agents/Definitions.htm". By comparing the different triple sets that describe the example above we recognize that both declarations (compare Table 1 with the Tables 2 and 3) are identical which they have to be according to the RDF's specification. Looking at the example, we again realize the RDF property-centric approach, i.e. properties are the basic RDF constructs while classes etc. are just an add on to define *rdfs:domain* and *rdfs:range* constraints of these properties.

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

A disadvantage of this modularity is of course that we cannot define single-valued attributes in RDF. For instance, it is not possible to define a property with a single value to represent the size of a resource. This, by the way, makes it difficult, if not impossible, to watch for violations of the single value property of *rdfs:range*. 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, although attributes are basically multi-valued in O-Telos, too, they can be constrained to be single valued by O-Telos constraints.

Using the frame syntax of O-Telos, modularity like in RDF is not possible, as definitions and instances in O-Telos are class-centric and not property-centric. So, in O-Telos it is not possible to use e.g. "http://…/Agents/Definitions.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, which however introduces an additional identifier which is not necessary in the tuple representation. Using this workaround, different O-Telos objects describing a resource have their own IDs as required by the O-Telos axioms but can describe the same resource. A similar approach has to be used in XML Schema, by the way.

The previous RDF example of the resource “http://…/Agents/Definitions.htm” is declared in O-Telos by the following single frame:

```
Individual AgentDefinition1 in TheoryUnit with
  about
    a : "http://…/Agents/Definitions.htm"
  language
    l : "german"
  coverage
    c : "Introductory course"
  rights
    r : "KBS"
  parent_unit
    pu : LectureUnit1
end
```

In order to represent the above object *AgentDefinition1* by two frames an explicit *about*-attribute is used in the frames below. The instances *AgentDefinition1* and *AgentDefinition2* have different identifiers while they hold the same reference in their about-attribute to “http://…/Agents/Definitions.htm”.

```
Individual AgentDefinition1 in TheoryUnit with
  about
    a : "http://…/Agents/Definitions.htm"
  language
    l : "german"
  coverage
    c : "Introductory course"
rights
    r : "KBS"
end
Individual AgentDefinition2 in TheoryUnit with
  about
    a : "http://…/Agents/Definitions.htm"
  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 possible to declare objects about the same resource in different models.

2.5 Sequences and Reification in RDF and O-Telos

Let us look briefly at sequencing and reification in RDF and O-Telos. 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:theoryUnit* 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="#AILecture"/>
    <olr:theoryUnit>
      <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:theoryUnit>
  </rdf:Description>
</rdf:RDF>
```

In this example the order of resources of the property *olr:theoryUnit* is defined by 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:theoryUnit*, instead of the more explicit ranges describing the specific type of the child resource (like *theoryUnit,* or, for other properties, *example*, *slide*, etc. which we use in OLR).

O-Telos does not define such a construct for stating sequences, but represents sequences implicitly by the order of attribute statements in the O-Telos frames. Of course it is not insured that each implementation of O-Telos interprets the frames in the same way so that the attribute order (the sequence) might vary from one implementation 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. Unfortunately, RDF statements do not have IDs. Instead we have to introduce higher-order statements which are a special 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"/>
<rdf:Description>
  <rdf:subject resource="#LectureUnit1" />
  <rdf:predicate resource="http://.../#theoryUnit" />
  <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://.../#theoryUnit " />
  <rdf:object rdf:resource="http://.../Agents/Characterisation.htm"/>
  <rdf:type resource="http://.../22-rdf-syntax-ns#Statement"/>
  <olr:ordinalNo>2</olr:ordinalNo>
</rdf:Description>
```

Of course the disadvantage is the lost simplicity of the model and a rather complex und unreadable declaration. In O-Telos, specifying properties for other properties can be handled more directly, as all property statements have their own unique identifier, and thus can be directly annotated with additional attributes like in

```
Attribute LectureUnit1!tu1 in CourseUnit!theoryUnit with
  ordinalNo
    o : 1
end
Attribute LectureUnit1!tu2 in CourseUnit!theoryUnit with
  ordinalNo
    o : 2
end
```

In [16] we show how to use this idea in an extended variant of RDF (O-Telos-RDF), which easily allows reifications of arbitrary statements by referencing statement IDs. Of course, introducing unique ids for property statements in RDF is not possible globally. Still, locally at one site, this is possible, and the site prefix can make these ids unique worldwide (which is the approach we propose in [16]).

2.6 Comparing RDF and O-Telos on the Tuple Level

As mentioned before RDF declarations can be represented as triples. A RDF triple has the definition:

<s,p,o> 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 are called propositions. In general propositions represent relationships:

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

So, O-Telos propositions include an explicit ID, while RDF triples do not. Using this ID, instantiation of properties is handled differently (explicitly in O-Telos, and implicitly in RDF), which results in a marked difference in the meta-modelling capabilities of RDF (rather restricted) and O-Telos (unrestricted meta-modelling hierarchies possible). Usually, each RDF triple is expressed by two O-Telos propositions, where the instantiation of a property is an own statement in O-Telos, but is handled implicitly (by directly using the predicate name) in RDF. In general, O-Telos propositions, which have a unique id, are much better suited for reification than RDF triples.

The following RDF declarations define three properties for the resource “http://.../Agents/Characterisation.htm”. The *rdf:type* property defines the resource as of type *olr#TheoryUnit* while *dc:title* states the name of the resource and *olr:parentUnit* defines the resource *LectureUnit1* as *parentUnit*. The triple representation shows three triples corresponding to this declaration.

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

Table 4. RDF-Triples of the declaration about "http://.../Agents/Characterisation.htm"

Nr.	Subject	Predicate	Object
1	http://.../Agents/Characterisation.htm	http://…/22-rdf-syntax-ns#type	http://.../rdf/olr#TheoryUnit
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

Table 4 states these three RDF triples. They show explicitly that all three properties belong to the resource, and the predicates (second argument) directly state name and the accompanying namespace of the properties.

The O-Telos frame declares the object "http://.../Agents/Characterisation.htm" as instance of (the keyword "in" in the frame) class *TheoryUnit* similarly to the *rdf:type* property of the RDF declaration. The other two attributes *dc_title* and *parentUnit* correspond to the respective properties.

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

However, the O-Telos propositions show more detail than the corresponding RDF triples:

Table 5. O-Telos propositions of the declaration of "http://.../Agenten/Characterisation.htm"

oid	source	Label	destination
#1	#1	"http://.../Agents/Characterisation.htm"	#1
#2	#1	*instanceof	#TheoryUnit
#3	#1	T1	"Characterisation of agents"
#4	#3	*instanceof	#dc_title
#5	#1	pu1	#LectureUnit1
#6	#5	*instanceof	#parentUnit

Table 5 shows the O-Telos propositions of our example. Proposition *#1* explicitly represents the object "http://…//Agents/Characterisation.htm" while proposition *#2* states that this object is instance of class *TheoryUnit*. 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*. Proposition *#5* declares that the object from *#1* has an attribute *pu1* as a reference to *#LectureUnit1* while proposition *#6* declares the attribute from *#5* as instance of *#parentUnit*. O-Telos also requires that the declaration of the attributes *t1* and *pu1* is included in the class *TheoryUnit* from which this object is an instance.

We have no direct possibility to represent RDF namespace information in our O-Telos propositions, as O-Telos relies on the declaration of schema and metadata in one file. In [16] however we specify statement IDs for O-Telos-RDF (which are invisible in O-Telos), that include namespace information in a way similar to RDF/RDFS.

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. An further extension will be P2P exchange functionality between distributed OLR systems.

In the second part of this paper we have compared RDF/RDFS with the conceptual modelling language O-Telos and discussed some mappings, which hopefully shed some light on the advantages and disadvantages of RDFS design decisions. We have continued this work in another report, which defines an RDF-variant called O-Telos-RDF with extended reification and meta-modelling capabilities. Further interesting work includes a comparison of the O-Telos query language (as implemented in Conceptbase) for RDF and O-Telos-RDF.

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.

References

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

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

[3] Dublin Core Initiative, 2001, http://dublincore.org/

[4] Karsten Tolle, Analyzing and Parsing RDF, Master's Thesis, Institute of Computer Engineering - University of Hannover in cooperation with the Institute of Computer Science - Foundation of Research Technology Hellas - Greece (ICS-FORTH), 2001, http://www.kbs.uni-hannover.de/Arbeiten/Diplomarbeiten-/00/tolle/AuPRDF.pdf

[5] Semantic Web Activity Statement, World Wide We Consortium (W3C), 2001, http://www.w3.org/2001/sw/Activity

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

[7] 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

[8] 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

[9] Wolfgang Nejdl and Nicola Henze: Adaptivity in the KBS Hyperbook System. 2nd Workshop on User Modeling and Adaptive Systems on the WWW, May 1999, Toronto, Canada, http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/1999/Henze.html

[10] M. Jeusfeld, Änderungskontrolle in deduktiven Objektbanken, Infix-Verlag 1992, St. Augustin, Germany

[11] 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

[12] 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

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

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

[15] J. Mylopoulos, A. Borgida, M. Jarke and M. Koubarakis, Telos: A langugage for representing knowledge about information systems, ACM Transaction on Information Systems, 8.4, 1990

[16] W. Nejdl, H. Dhraief and M. Wolpers, O-Telos-RDF: A Resource Description Format with Enhanced Meta-Modelling Functionalities based on O-Telos, Technical Report, Inst. f. Technische Informatik, Uni. Hannover, Germany