# Usage of metadata

Von der Fakultät für Elektrotechnik und Informatik der
Universität Hannover
zur Erlangung des Grades

DOKTOR DER NATURWISSENSCHAFTEN

Dr. rer. nat.

genehmigte Dissertation

von
Dipl.-Math. Jan O. Brase

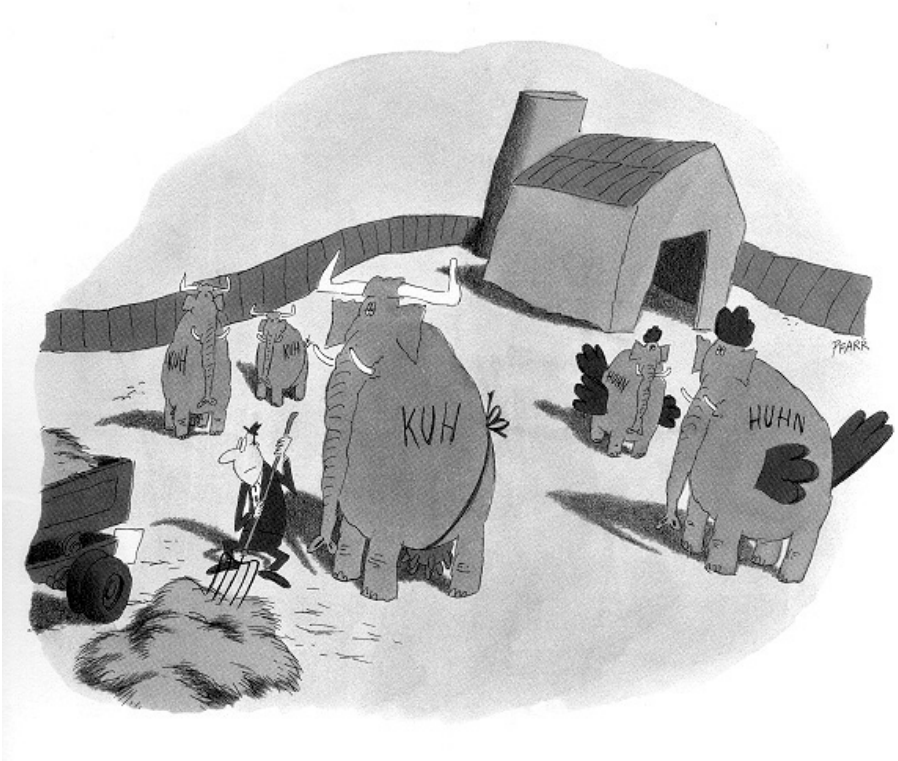
geboren am
20. November 1970 in Hannover

2005

Referent: Prof. Dr. Wolfgang Nejdl
Koreferent: Prof. Dr. Udo Lipeck
Tag der Promotion: 02. September 2005

Für Jonathan O. Brase

# Usage of metadata




Ferien auf dem Bauernhof
B. Pfarr, 1999

# Zusammenfassung

*Schlagwörter: Metadaten, Wissensverarbeitung, Internet-Technologien*

Metadaten bedeutet Daten über Daten, strukturierte Daten, die die Eigenschaften einer Ressource beschreiben. Der Griechische Ausdruck ”Meta” steht für eine Art von höherer Ordnung oder eine weitere Ebene. Der Begriff Metadaten war schon lange im Gebrauch bevor das Internet unsere Definition von Information so fundamental geändert hat, aber er wurde noch nie so aktiv verwendet wie heutzutage. In den letzten 5 Jahren hat sich unser Umgang mit elektronischen Ressourcen fundamental geändert. Insbesondere die Annotierung von Ressourcen mit Metadaten ist dabei immer wichtiger geworden.

Diese Arbeit will zuerst die bekannten Metadatenschemas *Dublin Core* (DC) und *Learning objects metadata* (LOM) vorstellen, sowie technische Methoden diese Schemas zu verwenden. Grundlage unser Arbeit war die Idee des Semantic Web, einer Erweiterung des bestehenden Internets, die auf der Verwendung von Metadaten aufbaut. Die Basis der Semantic Web Architektur sind die technischen Standards *Extensible Markup Language* (XML) und *Resource Description Framework* (RDF) die verwendet werden um Online-Ressourcen mit Metadaten zu versehen. Die Verwendung des Schemas LOM mit dem technischen Standard RDF erlaubt das *RDF binding von LOM*.

In dieser Arbeit wird diskutiert werden, wie Metadaten im Bereich eLearning verwendet werden können um vollständige Online-Kurse mit allen Informationen zu beschreiben die nötig sind, um auf diese Kurse in Lern-Netzwerken zugreifen zu können. Metadaten werden auch verwendet um den Inhalt von Lernressourcen zu klassifizieren und somit Suchanfragen nach inhaltlich verwandten Ressourcen innerhalb von Lern-Repositories oder P2P Netzwerken zu stellen. Durch die Verwendung von speziellen Inferenzregeln ist es außerdem möglich, implizite Metadateninformationen aus expliziten Kursbeschreibung zu extrahieren.

Eine weitere große Anwendung ist die Verwendung von Metadaten bei Kontextbasierter Informationsbereitstellung. Im Gegensatz zu den eLearning-Szenarios werden die Metadaten hier für implizite Informationsbereitstellung genutzt.

Zum Abschluss der Arbeit wir ein aktuelles Projekt vor dem Hintergrund Digitaler Bibliotheken vorgestellt.

# Abstract

*Keywords: Metadata, Knowledge management, Internet technologies*

Metadata means data about data, structured data which describes the characteristics of a resource. The term ”meta” derives from the Greek word denoting a nature of higher order or more fundamental kind. The term metadata was used long before the internet fundamentally changed our definition of information resources, but never before it has been so actively used as nowadays. The last five years have seen dramatic changes in our dealing with electronic resources. Especially the Annotation of resources with metadata has become more and more important.

This thesis will first introduce the common metadata schemes *Dublin Core* (DC) and *Learning objects metadata* (LOM)as well as technical ways of using them. Background for this work is the idea of the Semantic Web, an extension of the existing internet, which foundation is the usage of metadata. Basis of the Semantic Web architecture are the technical standards *Extensible Markup Language* (XML) and *Resource Description Framework* (RDF) that are used to annotate online resources with metadata. The use of the schema LOM with the technical standard RDF is made possible through the *RDF binding of LOM*.

This thesis will discuss how metadata can be used in the area of eLearning to annotate complete online courses with all information necessary to exchange these courses in learning networks. Metadata can also be used to classify content of learning resources and therefore enable queries for content-related resources inside learning repositories or P2P networks. With the usage of inference rules it is further possible to extract implicit metadata information from explicit course descriptions.

Another main application area is the usage of metadata in context based information provision. Contrary to the eLearning scenarios we have no longer only explicit, but now also implicit information provision.

At the end of this thesis we will briefy introduce usage of metadata in the background of digital libraries.

# Contents

# Abkürzungsverzeichnis

- *ACM* - Association for Computing Machinery
- *ACM CCS* - Association for Computing Machinery Computer Classification system
- *CISTI* - Canada Institute for Scientific and technical information
- *coData* - Committee on Data for Science and Technology
- *CSS* - Cascading Style Sheets
- *DC* - Dublin Core
- *DCMI* - Dublin Core Metadata Initiative
- *DCTerms* - Dublin Core Qualifiers
- *DEF* - The danish national research database
- *DFG* - German research foundation
- *DFKI* - German research centre for artificle intelligence
- *DL* - Description logic
- *DOI* - Digital object identifier
- *DTD* - Document Type Definition
- *ECDM* - Edutella Common Data Model
- *HTML* - Hypertext Markup Language
- *IDF* - International DOI foundation
- *IEEE* - Institute Of Electric And Electronic Engineers
- *IETF* - Internet Engineering Task Force
- *JDBC* - Java Database Connectivity

- *Loc* - Library of congress
- *LOM* - Learning Objects Metadata
- *LTSC* - Learning Technology Standards Committee
- *MAC* - Media Access Controll
- *MARC* - Machine Readable Cataloging
- *OLR* - Open Learning Repositoriy
- *OOP* - Object Oriented Programming
- *OWL* - Web Ontology Language
- *P2P* - Peer-to-Peer
- *PICA* - Project for Integrated Catalogue Automation
- *RDF* - Resource Description Framework
- *RDFS* - Resource Description Framework Schema
- *SWEBOK* - Software Engineering Body of Knowledge
- *SweLL* - Swedish Learning Lab
- *TCP/IP* - Transmission Control Protocol / Internet Protocol
- *TIB* - German National Library of Science and Technology (Technische Informationsbibliothek)
- *URI* - Uniform Resource Identifier
- *URL* - Uniform Resource Localizator
- *URN* - Uniform Resource Names
- *W3C* - World Wide Web Consortium
- *WWW* - World Wide Web
- *XML* - Extensible markup language
- *XSLT* - eXtensible Stylesheet Language Transformation

# Chapter 1

# Introduction

Metadata means data about data, structured data which describes the characteristics of a resource. The term ”meta” derives from the Greek word denoting a nature of higher order or more fundamental kind. The term metadata was used long before the internet fundamentally changed our definition of information resources, but never before it has been so actively used as nowadays. Tim Berners-Lee, one of the founder’s of the *World Wide Web* (WWW) and director of the *World Wide Web Consortiums* (W3C) [97] gave the definition:
*”Metadata is machine understandable information about web resources or other things”* (see [10])

Metadata provides us with basis information about a resource, like information about the author, the title or the date of publication. Thus it shares many similarities to the cataloguing that always took place in libraries, museums and archives. Like cataloguing of documents is based on standards, the effective use of metadata needs to be built on standards, too. The complexity of rules for standardization in the library community however cannot be transferred to the vast, chaotic and unordered amount of resources in electronic repositories like the WWW, as we see in chapter 2.

The term metadata therefore includes also the search for new attempts to describe resources and information, optimised for an effective and inexpensive use in electronic repositories.

A metadata record consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values. An example of a simple metadata record would be:

**Title**: Ontologies for eLearning
**Creator**: W. Nejdl
**Creator**: J. Brase

Metadata shall help us to achieve better search results when searching for resources in the *World Wide Web* (WWW). Why shouldn't the present internet search engines be not good enough? The problem relates to the underlying nature of the WWW. In the early 1990s, "surfing" the WWW was popularised in the mass media. These days, the concept of browsing the Web is little used. The Web has become a two-edged sword. While it is now very easy to publish information, it is becoming more difficult to find relevant information. For outsiders and casual users, much of the useful material is difficult to locate and therefore is effectively unavailable.

At the global level, internet search engines were developed to search across multiple Web sites. Unfortunately, these search engines have not been the panacea that some people had hoped for. Every search engine will give you good results some of the time and bad results some of the time. This is what information scientists term "high recall" and "low precision". The high recall refers to the well known (and frustrating) experience of using an Internet search engine and receiving thousands of "hits". It is popularly known as information overload. The low precision refers to not being able to locate the most useful documents. The search engine companies do not view the high hit rates as a problem. Indeed, they market their products on the basis of their coverage of the Web, not in the precision of the search results.

How does metadata solve the problem? A more formal definition of metadata offers an idea:
*Metadata is data associated with objects which relieves their potential users of having full advance knowledge of their existence or characteristics.* (see [29])

Information resources must be made visible in a way that allows people to tell whether the resources are likely to be useful to them. Metadata is a systematic method for describing resources and thereby improving access to them. If a resource is worth making available, then it is worth describing it with metadata, so as to maximise the ability to locate it.

Metadata provides the essential link between the information creator and the information user.

This thesis will discuss applications from different areas in which the usage of metadata is a matter of course nowadays.

In chapter 2 the two major standards for metadata are introduced: *Dublin Core* (DC) and *Learning objects metadata* (LOM). The author has attended work-meetings of the LOM definition group and has written the first German version of LOM in this context. As the library context is the classical usage field for metadata, we will in this chapter also present the most popular catalogue schemes in the library community today, as well as the usage of metadata for online library catalogues.

Chapter 3 introduces the principles of the Semantic Web, an extension of the existing internet, which foundation is the usage of metadata. Basis of the Semantic Web architecture are the technical standards *XML* and *RDF* that are used to annotate online resources with metadata. The usage of the metadata standard LOM with the technical standard RDF is made possible through the *RDF binding of LOM* that we will discuss in more detail in this chapter, as the author was a member of the group creating this binding.

Chapter 4 describes now concrete usage scenarios for metadata in eLearning on the basis of the standards introduced in the last chapter. In the context of various projects the author has developed a metadata scheme for the annotation of complete online courses with LOM-RDF-metadata, as well as different approaches and techniques for content classification. In the context of an electronic learning repository we will discuss how related resources can automatically be displayed using our methods. For easy annotation the author has also developed a set of inference rules, which we introduce in the background of a RDF based P2P network, where these rules can be used to gain richer result sets. The tow last chapters present usage of our metadata-technologies in two further scenarios:

Chapter 5 discusses the usage of metadata in a context based semantic web environment: Based on metadata a user will receive individual information on his handheld PC while visiting the research centre L3S. This scenario has been realised in a bachelor thesis supervised by the author.

In chapter 6 finally we will introduce an actual project that uses metadata for the first time to register scientific primary data in a library catalogue. This project is still ongoing and technically advised by the author.

# Chapter 2

# Metadata and Cataloguing schemes

## 2.1 Metadata schemes

In order to understand the intricacies of using metadata, it is necessary to understand the concept of metadata "metamodels". These are the conceptual schemas we use to describe our metadata models. Each metadata scheme will usually have the following characteristics:

1. a limited number of elements
2. the name of each element
3. the meaning of each element

**Dublin Core:** One of the most common metadata schemes on the web today is the *Dublin Core Schema* (DC) [1] by the *Dublin Core Metadata Initiative* (DCMI). The DCMI [37] now is an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources.

Each Dublin Core record is defined using a set of 15 elements from the ISO/IEC 11179 standard for the description of data elements, including for example: Title, Identifier, Language and Comment. To annotate the author of a learning resource DC suggests for example to use the element *creator*.

[1] The name Dublin Core is due to the fact that the first version of the set was written on a napkin in a pub in Dublin, Ohio

The complete list of elements can be found in table 2.1 The metadata elements fall into three groups which roughly indicate the class or scope of information stored in them:

1. Elements related mainly to the content of the resource: *title, subject, description, source, language, relation, coverage*
2. Elements related mainly to the resource when viewed as intellectual property: *creator, publisher, contributor, rights*
3. Elements related mainly to the physical manifestation of the resource: *date, type, format, identifier*

| **Elemnet** | **Description** |
|---|---|
| *Title* | A name given to the resource |
| *Creator* | An entity primarily responsible for making the content of the resource |
| *Subject* | A topic of the content of the resource |
| *Description* | An account of the content of the resource |
| *Publisher* | An entity responsible for making the resource available |
| *Contributor* | An entity responsible for making contributions to the content of the resource |
| *Date* | A date of an event in the lifecycle of the resource |
| *Type* | The nature or genre of the content of the resource |
| *Format* | The physical or digital manifestation of the resource |
| *Identifier* | An unambiguous reference to the resource within a given context |
| *Source* | A Reference to a resource from which the present resource is derived |
| *Language* | A language of the intellectual content of the resource |
| *Relation* | A reference to a related resource |
| *Coverage* | The extent or scope of the content of the resource |
| *Rights* | Information about rights held in and over the resource |

Table 2.1: Dublin Core

**Dublin Core Qualifiers:** Whereas ”Simple Dublin Core” uses only the elements from the Dublin Core metadata set as element-value-pairs, ”Qualified Dublin Core” employs additional qualifiers to further refine the meaning of a resource. The DCMI recommends a set of qualifiers called *Dublin Core Qualifiers* (DCterms), which include for example Created, Issued or Moified as some of the alternative qualifiers to refine the Date element (see table 2.2 for details). Note that in DCterms, there is a 16th DC element *audience*. From a structural point of view, DCterms elements are not subclasses of DC elements, but an extension on the same level. Furthermore DCterms offer different schemes to classify the value of an entry. A data entry could for example have the date scheme *dcterms:Period* if it includes a time intervall or *dcterms:W3CDTF* if it follows the W3C Encoding rules for dates and times, a profile based on ISO 8601. For a complete description, we refer the reader to [37].

The *metamodel* for Dublin Core defines the semantics of the DC elements and their qualifiers, such as: “An element is a property of the resource being described”, “An element refinement is a property of a resource that shares the meaning of a particular DCMI element but with narrower semantics”,

Table 2.2: Qualified Dublin Core (DCterms)

| **DC element** | **DCterms element** |
|---|---|
| Title | *Alternative* |
| Creator | |
| Subject | |
| Description | *TableOfContents*<br>*Abstract* |
| Publisher | |
| Contributor | |
| Date | *Created*<br>*Valid*<br>*Available*<br>*Issued*<br>*Modified*<br>*DateCopyrighted*<br>*DateSubmitted*<br>*DateAccepted* |
| Type | |
| Format | *Extent*<br>*Medium* |
| Identifier | *BibliographicCitation* |
| Source | |
| Language | |
| Relation | *IsVersionOf*<br>*HasVersion*<br>*IsReplacedBy*<br>*Replaces*<br>*IsRequiredBy*<br>*Requires*<br>*IsPartOf*<br>*HasPart*<br>*IsReferencedBy*<br>*References*<br>*IsFormatOf*<br>*HasFormat*<br>*ConformsTo* |
| Coverage | *Spatial*<br>*Temporal* |
| Rights | *AccessRights*<br>License |
| Audience | *Mediator*<br>*Education Level* |

**Learning objects metadata:** Since Dublin Core is designed for metadata for any kind of (digital) resource, it pays no heed to the specific needs we encounter in describing learning resources. The *Learning Objects Metadata Standard* (LOM) [66] by the *Learning Technology Standards Committee*(LTSC) of the *Institute Of Electric And Electronic Engineers* (IEEE) was therefore established as an extension of Dublin Core. Work on the LOM schema has started in 1998, the latest draft version was 6.4. Now that the standard has been accepted, it is officially LOM 1.0. Each learning object can be described using a set of more than 70 Data elements grouped into categories. The LOM 1.0 Base Schema consists of nine such categories:

1. General The General category groups the general information that describes the learning object as a whole.
2. Lifecycle The Lifecycle category groups the features related to the history and current state of this learning object and those who have affected this learning object during its evolution.
3. Meta-Metadata The Meta-Metadata category groups information about the metadata instance itself (rather than the learning object that the metadata instance describes).
4. Technical The Technical category groups the technical requirements and technical characteristics of the learning object.
5. Educational The Educational category groups the educational and pedagogic characteristics of the learning object.
6. Rights The Rights category groups the intellectual property rights and conditions of use for the learning object.
7. Relation The Relation category groups features that define the relationship between the learning object and other related learning objects.
8. Annotation The Annotation category provides comments on the educational use of the learning object and provides information on when and by whom the comments were created.
9. Classification The Classification category describes this learning object in relation to a particular classification system.

Collectively, these categories form the LOM 1.0 Base Schema. The Classification category may be used to provide certain types of extensions to

> the LOM 1.0 Base Schema, as any classification system can be referenced. Categories group data elements. The LOM data model is a hierarchy of data elements, including aggregate data elements and simple data elements (leaf nodes of the hierarchy). In the LOM 1.0 Base Schema, only leaf nodes have individual values defined through their associated value space and datatype. Aggregates in the LOM 1.0 Base Schema do not have individual values. All data elements are optional: this means that a conforming LOM instance may include values for any data element defined in the Base Schema. Since LOM was developed to be used for any kind of learning resource, LOM users soon find out that they do not really need to use all 70 elements.

The dependencies between the three standards DC, DCterms and LOM will be discussed in chapter 3 when we will discuss the technical realization of metadata in more detail.

All metadata elements used in the different scenarios in the next chapters can be derived from these three standards.

## 2.2 Digital libraries

### 2.2.1 Catalogue data

The promise of digital information organization implies the possibility of disseminating materials and information far beyond what has ever been imagined. One could view the area of *digital libraries* having two major facets:

1. The creation of cataloguing information (*catalogue data*) to store the information about resources in digital format.

2. Accomplishing this task with methods that offer effectively handle quantities of data exponentially larger than libraries have ever done. A key issue impacting the wide dissemination of digital information is the scalability of providing information (*metadata*) to structure and enable searching, navigation, and presentation of online documents and to enable searching, discovery, and retrieval of information.

1999 saw the discussion that many libraries in the United States and in Europe had not yet catalogued all of their holdings, and may not have all of these records in machine-readable format (see [100]). A shortage of staff, sometimes bad transitions from manual to automated cataloguing work flows, and the ”information explosion,” have created backlogs in cataloguing departments that most institutions do not publicize. The discipline of cataloguing has devised methods and policies to describe physical artefacts such as books, periodicals, microforms, sound recordings, and maps. These descriptions are largely based on the physical ”container” in which the information resides, and thus are considered format-based description. Intellectual description, that is, data about the subject of the information in the ”container” and a classification number reflecting subject analysis, is also created by cataloguers.

A discussion of granularity, or the level at which an item is described, is a conceptual key for understanding digital information organization. *Item level* cataloguing is probably most familiar to users of online library catalogues, who try to find monographs and multimedia materials. That is, one cataloguing record is made for one work. Archives and special collections often catalogue at the *collection level*, insofar as it is not feasible to individually describe every letter in a huge archive or assign meaningful classification numbers to millions of photographs. With indexed journal articles, the ”item” to be catalogued might be the title of the

journal along with an accounting of the individual issues, or holdings. Article-level indexing information gives further description of the intellectual content of "pieces" of each issue of the journal. Conversely, one catalogue entry might exist only at the title level of the serial publication without the more in-depth indexing information. Clearly, the article-level indexing provides greater access and description; it is also more expensive and labour-intensive to create and maintain. The topic of granularity of description is important because the creation of cataloguing data is one of the more expensive aspects of traditional library methods of providing access to materials. We will not go further into detail here, for more on bibliographic description, we refer to [5].

The two most popular catalogue schemes in the worldwide library community today are:

**MARC:** In march 1967 the *Library of congress* (Loc) started to store machine-readable catalogue information on tape in a format called MARC II (MARC is the acronym for *Machine-Readable Cataloguing*). The background for this format were the "Anglo-American Cataloguing Rules" (see [5])) from 1967. MARC II evolved to USMARC in the 1980s and to MARC21 in the late 1990s. The Marc fields have 3 digits, and are divided in 10 blocks:

- **0xx** *Identification and control fields* - language, shelf mark, classification ,etc.
- **1xx** *Main entry fields* - people and organisations involved, conference names, etc.
- **2xx** *Title and title-related fields*
- **3xx** *Physical description etc. fields* - dimension, playing time, physical medium ,etc.
- **4xx** *Series statement fields* - details of parts, details of parents
- **5xx** *Note fields* - footnotes
- **6xx** *Subject access fields* - keywords
- **7xx** *Added entry and linking entry fields* - further authors, different titles, etc.
- **8xx** *Series added entry fields* - physical and electronic location ,etc.
- **9xx** *Local data* - not standardized

Altogether MARC21 has around 330 fields. The field groups 1 to 8 follow precisely the order in the original catalogue cards of the Loc, therefore the first and second author for example (100 and 700) are relatively far away. This makes it easier for the cataloguer, but these inabilities to chance old concepts for the new online-world is typical for the situation of the libraries today. For a complete overview of MARC 21 we refer to [67].

**PICA:** In the year 1967 the Dutch royal library and six Dutch universities started with a pilot project for the recording of catalogue data. PICA, an acronym for *Project for Integrated Catalogue Automation* is based on MARC II, PICA is divided into the external format PICA3 and the internal representation format PICA+. PICA3 fields have 4 digits and are divided in the following groups:

**0xxx** *Control information* - date, time, internal number

**1xxx** *Coded entries* - language, country, physical form, etc.

**2xxx** *Identification* - ISBN, ISSN, etc.

**30xx** *Person names* - authors, editor, etc.

**31xx** *Organisation names* - all involved organisations

**32xx** *Series titles* - details of parts, details of parents

**4xxx** *Title description* - title information including footnotes

**5xxx** *Classification* - content classification following different schemes

**6xxx** *Local data* - internal classification

**7xxx** *Shelf marks*

**8xxx** *Internal identifiers*

**9xxx** *Summary*- summary of the content

Contrary to MARC21 the fields have a logical order and the whole structure is more oriented to the needs of online repositories, yet PICA nowadays has 1300 different fields.

For a complete overview we refer to [76].

**ORBIT:** In 1998 the *Danish National Research Database* (DEF) ([27]) introduced their own catalogue format ORBIT. This scheme identifies 5 types of objects:

1. Persons
2. Organisations
3. Projects
4. Events
5. Documents

Every instance of this 5 objects is described by a list of attributes.
Every instance can furthermore have relations with other objects or instances (see fig. 2.1). The number of relations depends on the type of object.
An article from a conference would be characterised:

**Document** : The article itself with relations to:

- **Person** - The authors
- **Organisation** - Their institutes
- **Project** - The respective projects
- **Event** - The conference
- **Document** - The proceedings

The 25 possible types of relations are strictly defined by their main object and its related auxiliary object (from-to). Each relation is described by the role of the auxiliary object within the main object. Examples are:

- Person to document: *authorOf*
- Organisation to person: *director*
- Document to event: *presented at*

These structure allows quick finding of all relations between resources. ORBIT is XML based (see chapter 3). For a complete description we refer to [74].

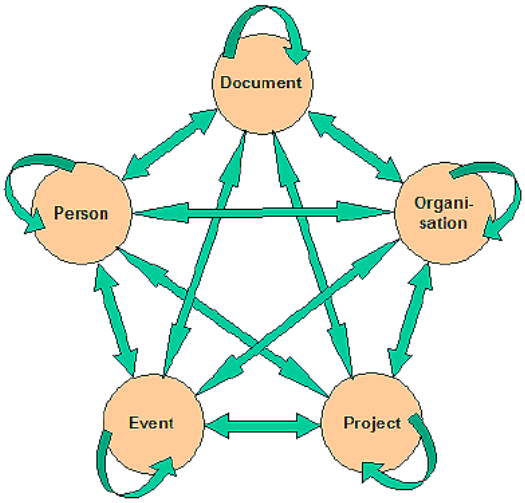



Figure 2.1: Relations in the ORBIT metadata scheme

### 2.2.2 Metadata

On the surface, provision of metadata to accompany digital objects does not seem difficult. Roughly speaking, many people think that all that must be done is to take existing cataloguing information, convert it to the appropriate format, and link it to the digital resources. The process is not that simple due to several factors. First of all, the conversion of a physical artefact implies not just putting information into a new format but the concomitant goal to display the information in a logical way. To do that, information in addition to the content must be produced or extracted to enable the structure and display of the data. If existing schemes for classification and indexing are used, human intellectual capital is necessary at some point in the process to apply thesaurus terms and enable other access points (catalogue). As mentioned in the last section, metadata in the digital library context therefore is used to structure and enable searching, navigation, and presentation of online documents. Briefly the cataloguing and metadata schemes in use in the most important digital libraries worldwide are as follows:

Table 2.3: Metadata elements for the online search at the LoC

| **Element** | **DC-analogy** |
|---|---|
| *Keyword* | dc:subject |
| *Title* | dc:title |
| *LLCN-ISBN-ISSN* | dc:identifier |
| *Name: Personal name* | dc:creator or dc:contributor |
| *Subject all* | dc:subject |
| *LC control number (LLCN)* | dc:identifier |
| *International standard numbers* | dc:identifier |
| *Series* | dcterms:isPartOf |
| *Publication information* | dc:publisher |
| *Subject: authorized* | dc:subject |
| *Name: All* | dc:creator or dc:contributor |
| *Name: corporate/meeting* | dc:contributor |
| *Title: uniform title* | dc:title |
| *Notes* | dc:description |
| *Subject geographic: all* | dc:subject |
| *Contents notes* | dc:description |
| *Credits/performers* | dc:contributor |
| *ISSN* | dc:identifier |

**Library of congress (LoC):** The catalogue system of the LoC is of course MARC21. The online catalogue can be searched using 18 metadata elements, the elements *keyword, title, standard number, name, publication information, subject, notes and credits/performers* would have direct DC or DCterms correspondence. 10 more elements are only variations of these 7. The element *series* might be expressed with a *dcterms:isPartOf* The search allows a Boolean combination of two elements and the choice *"all of these"*, *"any of these"* or *"as a phrase"*, if more than one term is searched for. The complete list of elements can be found in table 2.3.

Table 2.4: Metadata elements for the online search at the BL

| **Element** | **DC-analogy** |
|---|---|
| *Author* | dc:creator |
| *Title* | dc:title |
| *Publication year* | dcterms:issued |
| *Publisher* | dc:publisher |
| *Place of Publication* | none |
| *Subject* | dc:subject |
| *ISBN* | dc:identifier |
| *ISBN* | dc:identifier |
| *Uniform title* | dc:title |
| *Format* | dc:format |
| *Subject heading (Library of congress)* | dc:subject |
| *Notes* | dc:description |
| *Content notes* | dc:subject |
| *Shelf mark* | dc:identifier |
| *Other/superseded Shelf mark* | dc:identifier |
| *Type/characteristic* | dc:type |
| *report number* | dc:identifier |
| *Technical report number* | dc:identifier |
| *System number* | dc:identifier |

**British Library:** In June 2004 the British library had changed their catalogue system to MARC21 The online catalogue [23] can be searched using 19 metadata elements. The elements *author, title, publication year, publisher, subject, ISBN, ISSN, format, subject heading, notes and Type/characteristic* have a direct DC or DCterms analogy. *Uniform title* is a variation of *title*, all of the other elements except *PublicationPlace* can be seen as identifiers of some sort. All elements and their analogies are shown in table 2.4. The search can furthermore be limited to *language*, *year* and *format*.

Table 2.5: Metadata elements for the online search at CISTI catalogue

| **Element** | **DC-analogy** |
|---|---|
| *Author* | dc:creator |
| *Title* | dc:title |
| *Subject* | dc:subject |
| *Note* | dc:description |
| *Year* | dcterms:created<br>or dcterms:extends |
| *Catalogue subset* | dc:subject |
| *Location* | none |
| *Language* | dc:language |
| *Conference publication* | none |
| *Type of publication* | dc:type |
| *Publisher* | dc:publisher |

**Canada Institute for Scientific and technical information (CISTI):** CISTI's collection ([24]) is one of the largest in North America. Nearly 500 new items are received every day. It includes published information from around the world in all areas of physical and life sciences, engineering, technology and health sciences.

The CISTI collection includes: over 50,000 different serial titles, over 11,000 currently received serials, over 600,000 books, conference proceedings and technical reports etc.

The collection is also based on MARC21, the 11 metadata elements for the online-search are displayed in table 2.5. Again, we find the elements *author, title, subject, note, year, type of publication, publisher and language* that offer a direct DC or DCterms analogy. The element *Catalogue subset* is used to identify resources inside CISTI‘s cataloguing system and has therefore an analogon in *dc:subject*. *Location* and *Conference Publication* have no DC anologies.

Table 2.6: Metadata elements for the online search at the DEF

| **Element** | **DC-analogy** |
|---|---|
| *Author/person* | dc:creator |
| *Journal/Serial Title* | dc:title |
| *Organisation* | dc:publisher |
| *Number* | dc:identifier |

.

**The Danish national research database (DEF):** The Danish National Research Database presents an overall picture of research in progress and published Danish research. The database has been established by the Ministry of Science, Technology and Innovation, and is today a part of Denmark's Electronic Research Library. The day to day running of the database is maintained by the Project Management of the database.

Since 1988, when collecting and storing information about Danish research results and research in progress began, the database has grown to include more than 150.000 research references.

The Danish National Research Database is based on information delivered from universities, institutions of higher education, Government research institutes, research councils and other public institutions carrying out research. The number of database suppliers continuously grows. Consult the updated list of data suppliers.

The internal format is the ORBIT format, metadata elements for the online search are only 4 elements that have direct anaologies to DC, and are displayed in table 2.6.

We identify that most search elements have more or less DC anologies. Only the place of a publication, or the physical location of a resource is difficult to express with DC. It is however obvious that a direct mapping to DC or DCterms could only work, if we would express explicit rules for the usage of the attributes and if possible provide exact vocabularies for certain elements. We are currently working on explicit use of Dublin Core metadata elements to identify a set of query attributes in cooperation with the *German national library of science and technology* (TIB).

# Chapter 3

# Technical realization of metadata

In this chapter we will introduce the principles of the Semantic Web, an extension of the existing internet, which foundation is the usage of metadata. Basis of the Semantic Web architecture are the technical standards *XML* and *RDF* that are used to annotate online resources with metadata. The usage of the metadata standard LOM with the technical standard RDF is made possible through the *RDF binding of LOM* that we will discuss in more detail in this chapter.

## 3.1 The Semantic Web

Already in September 1998 Tim Berners-Lee published the *Semantic Web Road map* [12], where he describes the necessary steps that would lead from the WWW towards a web, in which machines are able to extract and understand the information available, the semantic web. The actual birth of the semantic web nevertheless is often defined in context of the article *The Semantic Web* [11] by Tim Berners-Lee, James Hendler und Ora Lassila, where the most popular definition for the semantic web was published for the first time:

> *The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*

This extension has already proceeded. The W3C [97] defines the architecture of the Semantic Web as a layer model (see fig: 3.1). The development of the semantic web takes place step by step building one layer on top of the other. For every step standards must be defined, enabling a global means of data exchange like the standard TCP/IP model.

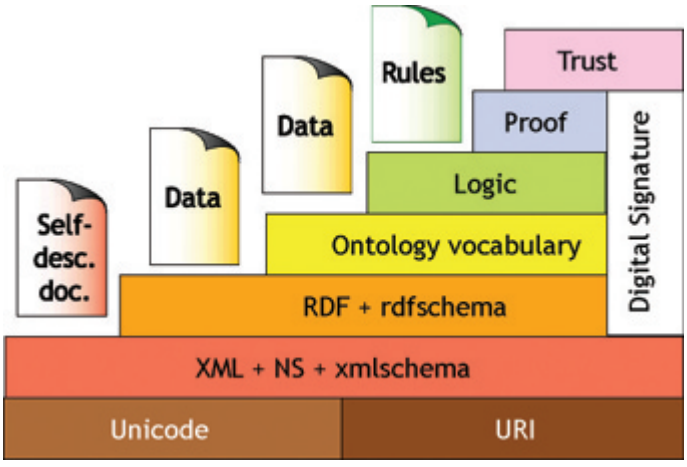



Figure 3.1: The Semantic Web Tower

XML is the basis of the *Semantic Web Towers*. XML is ideal to create structured documents that use *Unicode* as text and *Uniform Resource Identifier* (URIs) to identify other web resources. RDF is expressed using XML syntax.

In this chapter we will introduce the this first three layers of this tower and will notice one major principle of the semantic web: the *downwards compatibility*: All layers use constructs of the lower layers, enabling programs designed for one layer, to understand the information of lower layers.

### 3.1.1 HTML

In the present WWW the *Hypertext Markup Language* (HTML) is established as the standard format for web resources. A web browser can interpret the HTML-Tags and present the HTML-page. One could therefore include metadata simply in the HTML code of the page (We will not go into details of the HTML-language here):

```
<html>
<body>
<h1> Ontologies for eLearning </h1>
<h2> Authors:</h2>
<h3> Wolfgang Nejdl (Prof.)<br/>
Jan Brase (Researcher) </h3>
</body>
</html>
```

Humans can easily understand the content of this resource, since HTML pages are displayed in a browser in a very structured way. Following Berners Lee definition of metadata in chapter 2 , this information should nevertheless be machine-readable. A *Web Application*, trying to retrieve the authors of this resource would encounter heavy difficulties. There is no explicit information about who the authors are. The problem is that HTML documents do not contain structural information that is information about pieces of the document and their relationships.

### 3.1.2 XML

For machine readability another standard should be used instead of HTML, the *Extensible Markup Language* (XML), which was derived from a document description language called SGML (an international standard for structured documents). In 1996, discussions began which focused on how to define a markup language with the power and extensibility of SGML but with the simplicity of HTML. The *World Wide Web Consortium* (W3C) decided to sponsor a group of SGML experts including members from Sun. They skipped all of the non-essential, unused, cryptic parts of SGML, leading to a 26 pages specification of XML (see [95]) opposed to the 500+ pages of the SGML specification. By mid

1997 Microsoft had launched the Channel Definition Format (CDF) as one of the first real-world applications of XML. Finally, in 1998, the W3C approved Version 1.0 of the XML specification.

Our example from above could be written as an XML resource as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE authorsOfresource SYSTEM "authors.dtd">
<authorsOfresource name="Ontologies for eLearning">
    <member name="Wolfgang Nejdl" function="Professor"/>
    <member name="Jan Brase" function="Researcher"/>
</authorsOfresource>
```

As a HTML document, this XML document is structured with tags, but it is far more accessible to machines because every piece of information is described. If two applications however want to communicate about this resource, we would have to ensure that they use the same vocabulary. It is therefore necessary to define all the elements and attribute names that may be used. In the first line of our XML example a *Document Type Definition* (DTD) is declared for the XML-resource. The DTD *authors.dtd* contains all structured information and could look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ATTLIST authorsOfresource name CDATA #REQUIRED>
<!ELEMENT authorsOfresource (member+)>
<!ATTLIST member
     name CDATA #REQUIRED
     function CDATA #REQUIRED
>
<!ELEMENT member EMPTY>
```

The meaning of this DTD is as follows:

**Line 02:** Every XML file following this DTD must contain at least one element *authorsOfresource* of string type with the attribute *name*. (CDATA stands for string, # REQUIRED implies the appearance of the attribute).

**Line 03:** The element *authorsOfresource* contains the element *member*, appearing one or more times (The cardinality is expressed in (member+)

**Line 04-06:** The element *member* must have the attributes *name* and *function* of type string (CDATA).

**Line 07:** Otherwise the element *name* is empty.

If we use XML together with DTD documents we could annotate our resources with machine-readable, syntactically correct metadata. For a complete description of XML, we refer to ([95]). If we would also like to model the semantic structure of our metadata standard, we have to go beyond XML to the next important standard:

### 3.1.3 RDF

The *Resource Description Framework* (RDF) is a data model to describe web resources using so called ”RDF-statements”.

RDF is the result of a number of metadata communities bringing together their needs to provide a robust and flexible architecture for supporting metadata on the web. While the development of RDF as a general metadata framework, and as such, a simple knowledge representation mechanism for the web, was heavily inspired by the W3C since 1997, no one individual or organization invented RDF. RDF is a collaborative design effort. Several W3C Member companies were contributing intellectual resources. It is drawing upon the XML design as well as proposals submitted by Microsoft and Netscape. Other metadata efforts, such as the Dublin Core have also influenced the design of the RDF. In 1999 the RDF Model and Syntax Specification was released as a W3C Recommendation (see [90]).

RDF statements are just triples consisting of a subject, a property and an object, where the object can be a resource or an atomic value called *literal*. A subject is a resource, a ”thing” referenced by an URL (By theory any kind of URI could be used, but in practice it are mostly URLs). A property is a special kind of resource, it describes relationships between resources, but as a resource, the property must also be identified via a URL. If we would for example want to state that a resource at http://www.xyz.com/resource.html, has the author ”nejdl”, we would first have to identify an URL for our property (we could find it at the web page of Dublin Core, for example) and then could use the following triple:

```
Subject: http://www.xyz.com/resource.html
Property: http://purl.org/dc/elements/1.1#creator
Object:"nejdl"
```

An abstract data model needs a concrete syntax of course, and RDF has been given a syntax in XML, enabling the *downwards compatibility* of the semantic

web tower in fig. 3.1, as every RDF file is also a valid XML-file. Using the XML syntax proposed by the W3C the statement above can be written as:

```
<rdf:RDF
 xmlns:rdf="http://www.w3.org/2000/01/rdf-syntax#"
 xmlns:dc="http://purl.org/dc/elements/1.1#">

 <rdf:Description about=http://www.xyz.com/resource.html>
    <dc:creator>nejdl</dc:creator>
 </rdf:Description>

</rdf:RDF>
```

Note that the definition of namespaces in line 02-03 allows us to use tags for properties that are identified with URLs in an elegant way. We will not discuss the details of RDF here, we refer the reader to [90] instead.

Further RDF constructs include:

**rdf:Statement** This element, together with its child elements *rdf:subject*, *rdf:object* and *rdf:predicate*, allows us to make statements about statements. RDF allows this, using a *reification mechanism*. The RDF statement above could be reified as:

```
<rdf:Statement rdf:about="StatementAboutResource.html">
<rdf:subject rdf:resource="http://www.xyz.com/resource.html">
<rdf:predicate rdf:resource="&dc;name"/>
<rdf:object> nejdl </rdf:object>
</rdf:Statement>
```

*Rdf:subject*, *rdf:object* and *rdf:predicate* now allow us to access parts of this statement. Note that for better readability we assume, that in this example *http://purl.org/dc/elements/1.1#* is not only refined as a namespace *dc*, but also as a XML entity *&dc;*. An example for XML entity definition can be found in section 4.2.4

**Container elements** The elements *rdf:Alt*, *rdf:Seq* and *rdf:Bag* are used if an object of a statement is a list. They indicate whether the list is unordered *rdf:Alt*, ordered *rdf:Seq* or the elements are alternatives for each other *rdf:Bag*.

RDF is domain-independent in that no assumptions about a particular domain of use are made. To define the vocabulary used in a RDF data model, one can define their own terminology in a language called *RDF schema* (RDFS).

### 3.1.4 RDFS

In contrast to XML schemas, which are used for validation of XML records, RDF schema offers a vocabulary based on RDF to describe Web resources as classes with certain attributes, and therefore allowing hierarchies and restrictions. Like in *Object Oriented Programming (OOP)* concrete application areas can be modelled by defining classes with certain attributes. An instance of a class can only have the attributes the class allows for its instances. Furthermore ranges can be defined for properties. If a property is of type *Boolean*, the values for this property can only be *true* or *false*. Finally hierarchies can be modelled by defining certain classes as subclasses of other classes. There is however a big difference to classes, inheritance and properties in OOP. In OOP an object class defines the properties that apply to an existing class. To add new properties to a class means to modify that class. RDFS however is property centred: Properties are defined globally, as they are not included in class definitions. It is possible to define new properties that apply to an existing class without changing that class.

An example of the most important RDFS constructs would be:

- **Classes** - All classes are instances of **rdfs:Class**
  - **rdfs:Resource** - Everything described with RDF is a resource and of this type
  - **rdfs:Class** - This is the super class of all resources that are classes
  - **rdfs:Literal** - Values of properties that are not referring to other resources
  - **rdf:Property** - Properties that are defined in RDFS are always instances of **rdf:Property**
- **Properties** - All properties are instances of **rdf:Property**
  - **Properties, defining relations**:
    - **rdfs:subClassOf** - The Triple (A, rdfs:subClassOf, B) implies that A and B are classes, and A is a subclass of B.
    - **rdfs:subPropertyOf** - The Triple (A, rdfs:subPropertyOf, B) implies that A and B are properties, and A is a subproperty of B.
    - **rdf:type** - The Triple (A, rdf:type, B) implies that B is an instance of **rdfs:Class** and A is an instance of B
  - **Properties, defining restrictions**:

    - **rdfs:range** - The Triple (P, rdfs:range, C) implies that P is a property and C is a class. Furthermore P can only have values that are instances of C.
    - **rdfs:domain** - The Triple (P, rdfs:domain, C) implies that P is a property and C is a class. Furthermore only instances of C can have the property P.
  - **Utility properties**:
    - **rdfs:label** - As identifiers of resources often have no meaning for humans, **rdfs:label** can be used to give a meaningful name to a resource.
    - **rdfs:comment** - To include descriptions of a resource.
    - **rdfs:isDefinedBy** - To refer to a file, where a certain resource is defined

These basis concepts of RDFS are defined using RDF/XML. Therefore RDFS is just a predefined vocabulary based on RDF/XML. So every RDFS document is also a legal RDF document. Below you can see the definition of the property *rdfs:subClassOf* following W3C recommendation (see [22]):

```
01 ...
02 ...
03 <rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#subClassOf">
04   <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
05   <rdfs:label>subClassOf</rdfs:label>
06   <rdfs:comment>The subject is a subclass of a class.</rdfs:comment>
07   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
08   <rdfs:domain rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
09 </rdf:Property>
10
11 <rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Class">
12   <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
13   <rdfs:label>Class</rdfs:label>
14   <rdfs:comment>The class of classes.</rdfs:comment>
15   <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
16 </rdfs:Class>
17
18 <rdfs:Class rdf:about="http://www.w3.org/2000/01/rdf-schema#Resource">
19   <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
20   <rdfs:label>Resource</rdfs:label>
21   <rdfs:comment>The class resource, everything.</rdfs:comment>
22 </rdfs:Class>
23
24 <rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#domain">
25   <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
26   <rdfs:label>domain</rdfs:label>
27   <rdfs:comment>A domain of the subject property.</rdfs:comment>
28  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
```

```
29   <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
30 </rdf:Property>
31
32 <rdf:Property rdf:about="http://www.w3.org/2000/01/rdf-schema#range">
33   <rdfs:isDefinedBy rdf:resource="http://www.w3.org/2000/01/rdf-schema#"/>
34   <rdfs:label>range</rdfs:label>
35   <rdfs:comment>A range of the subject property.</rdfs:comment>
36   <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
37   <rdfs:domain rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/>
38 </rdf:Property>
39 ...
40 ...
```

Comments on this definition:

**subClassOf:** To define *rdfs:subClassOf* a resource of the type (*rdf:type*) *rdf:Property* is defined in line 03 with the range and domain *Class*. Line 04-06 describe the resource with some *utility properties*. Domain and range of this resource are *Class*

**Class:** The resource *rdfs:Class* is of its own type (line 11), and is a subclass of the resource *rdfs:Resource* (line 15), which by definition of the property *subClassOf*, has to be as well of the type *rdfs:Class*, as domian and range of the property are *rdfs:Class*.

**Resource:** The resource *rdfs:Resource* has only an ID and apart from some *utility properties* only the information that it is of type *rdfs:Class* (line 19).

**domain and range:** Both resources are of the type *rdfs:Property* and have also *rdfs:Property* as domain (line 30 and 38).

We will present examples of how to use such restrictions in the next section, for a complete description of RDF and RDFS, we refer to ([65] and [22]).

### 3.1.5 Bindings

As mentioned before, to enrich the usage of a standard metadata schema one should define the structure of the metadata elements and their relationships with one another, This is done by creating a ”binding”, a formal language representation for the metadata language. The namespace ”dc:” in our RDF example refers to an URL containing a RDF schema that describes the structure of the metadata elements of dublin core, dc:creator in this case. This schema is called the RDF-binding for DC. If we want to describe a resource with RDF and use LOM, we

write triples like the one above in a RDF file and refer to the RDF binding of the LOM elements we use. A final metadata description is just a set of these triples. The use of namespaces makes it a part of a global network of information, where anyone has the capability of adding metadata to any resource, using standardized or specialized schemas describing these metadata. This modularity of the architecture leads to naturally reusable constructs. The LOM RDF binding is directly compatible with Dublin Core RDF binding, and therefore Dublin Core elements are used directly instead of defining new LOM elements for these DC properties. This of course helps a lot to enhance the interoperability between resources that are annotated with DC and others that are annotated with LOM. In the latest draft for the RDF binding of LOM ([71]) for example, about 80 percent of the LOM elements are defined using DC and DCterms. We will discuss this binding in more detail in the next section.

## 3.2 The Metadata binding of LOM

In June 2002 the *Institute Of Electric And Electronic Engineers* (IEEE) approved the first version of the Learning Object Metadata (LOM) standard. LOM is gradually becoming the reference standard of choice for educational systems managing learning objects of many kinds.

The LOM data model standard, or IEEE LTSC 1484.12.1, is only the first part of a multi-part standard. This first part contains an abstract model of the descriptors, or elements that are used to describe learning objects, and does not deal with the technical realization of these elements.

Work is currently underway within the *Learning Technology Standards Committee* (LTSC) of the IEEE to produce standards for three bindings of the LOM abstract data model:

- ISO/IEC 11404 (P1484.12.2), an abstract language for specifying datatypes,
- XML (P1484.12.3), and
- RDF (P1484.12.4)

we will not discuss the first method here.

XML Bindings define an exchange format for metadata. The metadata might be contained in a database and an XML representation is usually generated on demand, for export to other tools and environments. Thus, an XML metadata record is a self-contained entity with a well-defined hierarchical structure, and there is seldom a natural way to reuse other metadata standards (or specific fields from other standards). Examples for the XML binding can be found on the homepage of the IMS consortium ([57]). An excerpt from a XML binding for LOM could look as follows (This DTD defines the syntax for the LOM category *lifecycle*)

```
<!ELEMENT lom (general?, lifecycle?, metametadata?, technical?, educational?,
                                rights?, relation*, annotation*, classification*)>

<!ELEMENT lifecycle (version?, status?, contribute*)>
<!ELEMENT version (langstring?)>
<!ELEMENT status (vocabulary?)>
<!ELEMENT contribute (role?, centity*, date?)>
<!ELEMENT role (vocabulary?)>
<!-- centity: The is the ENTITY element.  The word ENTITY is reserved within XML,
so the name has been changed to centity to stand for "Contributing Entity".   -->
<!ELEMENT centity (vcard?)>
<!ELEMENT vcard ANY>
<!ELEMENT date (datestructure?)>
<!ELEMENT datestructure (datetime?, description?)>
<!ELEMENT datetime (#PCDATA)>
```

We will discuss in the following the details of the RDF binding of LOM, which has been developed by a group led by Michael Nilsson from *Swedish Learning Lab* (SweLL), with input from our group and input from the Viennese colleagues from the UNIVERSAL project [51]. Excerpts from this section were first presented in [16]

This work was initiated in 2000 within the context of the IMS Global Learning Consortium [57], and a first draft was released as an appendix to version 1.2 of their popular metadata standard [58], which was based on earlier drafts of the LOM standard. The effort was subsequently transferred to LTSC, and the current drafts can be found at [59] and [71] .

### 3.2.1 Using RDF for Metadata as Compared to XML

There are significant differences in the metadata modelling approaches used in the LOM XML binding and in the LOM RDF binding, resulting from both the differences in the design of the respective frameworks and their different typical usage scenarios.

#### Metadata metamodels

Compared to the DC metamodel, which has no hierarchie, LOM uses a completely different metamodel.

The descriptors are organized in a tree-like structure under the nine categories. This tree makes it possible to organize the information in a consistent way, grouping information into related pieces.

However, it can be easily seen that this metamodel is not compatible with the DC metamodel. As a simple example, the *2.3.3 Date* element is not a property of the resource being described, but can be seen to be a property of the *Contribution* it belongs to. Similarly, the elements in the *Meta-metadata* category are not properties of the resource being described, but of the metadata document itself.

The metamodel used by LOM is thus not compatible with the metamodel used by Dublin Core. When does this matter? Binding LOM to RDF is the obvious example in this context, as the metamodel of RDF is based on a property-value model and not containment. In general, it leads to difficulties when trying to combine terms from two metadata standards into the same system. When the metamodels are compatible, such a combination or mapping can be realized by simply translating the metamodel constructs. If the metamodels are incompatible, the translation must be done on an idiosyncratic, element-by-element basis.

This metamodel incompatibility is the main source of the challenges in binding LOM to RDF.

### Semantic modelling

In an XML binding such as the LOM XML binding, the structure of the XML instance is the result of choosing the most convenient syntax, creating the element hierarchy that best matches the structure of the LOM elements (see example above). The XML metamodel is also containment-based, and is therefore easily adapted to LOM.

Where XML data has no other semantics than just a tree, RDF data has the semantics of an object-oriented system, and can therefore be viewed as objects having properties that relate them to other objects. The *type* of an object or of a property defines its interpretation, and is thus not simply a syntactic marker.

In the XML binding of LOM each LOM element is represented by an XML element. In RDF, the semantics of each LOM element decides its representation:
If it is a property applying to a resource? - use an RDF property.
If it is a resource having properties? - use an object with a specific type.
If it is just a container with no object or property semantics (Such as *General*)? - consider using a namespace for the contained properties and object types.
And the choices matter, as those constructs have fundamentally different semantics, i.e., they will be processed differently by applications. All of these constructs are used in the current binding draft.

Thus, a considerable amount of effort is needed to extract the desired semantic quality of each LOM element in order to be able to represent it appropriately. If this reinterpretation is not done, you risk losing not only clarity for the human consumer, but you risk more serious damage to the usefulness of the model. Much of the effort that has gone into the LOM RDF binding has focused on creating such a well-formed (machine-interpretable) semantics of the model.

We therefore expect to see much richer structures on many levels in an RDF representation than in the corresponding XML binding instance. The RDF binding thus adds semantics to the LOM data model, in that it adds interpretations to the elements that are not explicit in the LOM data model.

### Metadata Frameworks: Documents vs. statements

The fundamental unit in RDF is the *statement*, that expresses the value of one property of one resource. Such statements can be arbitrarily combined, separated

and recombined.

Thus, the metadata for one resource need not be contained in a single RDF document. Translations might be administrated separately, and different categories of metadata might be separated. This dramatically strengthens the incentive both to reuse identical structures that are used repeatedly, as well as to create decentralized descriptions of resources. Both of these phenomena naturally lead to a fundamentally different approach to metadata modelling than that found in XML-based metadata. While XML describes the structure of a complete metadata instance, RDF describes the structure of single metadata statement. The RDF binding must therefore be designed one element at a time.

As a consequence of this, we cannot expect the RDF binding to fulfil the same purpose as the XML binding. The XML binding defines an exchange format for metadata. The metadata might be contained in a database and an XML representation generated on demand, for export to other tools and environments. Thus, an XML metadata record is a self-contained entity with a well-defined structure. In RDF, the metadata for a resource is not always self-contained, but rather forms part of a global network of information, where anyone has the capability of adding any kind of metadata to any resource.

### Semantic and Structural Extensions

Another aspect is that of compatibility. In the XML binding of LOM, there is no standard way to reuse other metadata standards. The reason for this is the monolithic nature of an XML document – there is no canonical way of combining information from two documents into one.

The statement-centric design of RDF leads to naturally reusable constructs. Metadata elements can be extended both structurally (by adding more information), or semantically (by adding refinements of elements). This binding has been designed to be directly compatible with Dublin Core (including the DC Qualifiers, DC Type and DC Education vocabularies) and with the vCard RDF binding [71]. However, this compatibly comes at the price of modelling freedom – some modelling restrictions are imposed on us. Fortunately, much of this compatibility comes for free when using the RDF metamodel.

Finally, as RDF is intended to be processed by software, and in many cases software with no explicit knowledge of LOM, it is important to use explicit data typing, i.e. self-describing data. This will be seen below in the representation of languages and dates, which are strings tagged with their encoding scheme. Thus, a goal of this binding has been to define a set of RDF constructs that facilitates

introduction of LOM metadata into the semantic web in the most semantically complete and useful way.

### 3.2.2 The RDF binding of LOM

We will now turn to discussing some of the main features of the LOM RDF binding. The binding makes use of RDF schema to express some of the semantics of the RDF constructs. There is no need to explain in detail the binding of each and every LOM element, as that is covered by the binding draft itself.

However, there are a number of modelling constructs that are of more general interest, and we will now discuss them. For details on these constructs and the rest of the binding the binding draft can be seen in [71].

#### Relationship to Dublin Core

Some of the LOM elements are semantically similar to Dublin Core elements, and Appendix B of the LOM standard contains a translation between these elements and the corresponding Dublin Core elements.

Our RDF representation of LOM relies heavily on the Dublin Core metadata element set ([39], and its representation in RDF. LOM elements are modelled in a way similar to the representation of Dublin Core Qualifiers, give in [38] in RDF. Where applicable, LOM elements are described as *rdfs:subClassOf* or *rdfs:subPropertyOf* the corresponding DC/DCterms elements. In this sense, parts of LOM can be viewed as proper extensions to qualified Dublin Core (see fig. 3.2).

Our RDF representation of LOM is therefore almost fully Dublin Core RDF compatible, in the sense that most Dublin Core metadata constructed according to this binding can be directly understood by Dublin Core-aware software. Most of the elements of the LOM Dublin Core mapping (in Appendix B of [66]) are compatibly represented, allowing the use of all the Dublin Core constructs in a way compatible with the DC RDF binding [38] and this binding. It is, however, not always possible to map a pure Dublin Core construct (constructed without reference to this binding) to a LOM element without adding information, as LOM requires a more specific structure in many elements. The guiding principle has instead been that using the *dumb-down* algorithm described in [38] on LOM metadata should result in useful Dublin Core metadata. This algorithm essentially removes all qualifying properties, leaving only the literal value of a property, which then corresponds to unqualified Dublin Core. This binding has been designed with

this algorithm in mind. This way, software can produce unqualified Dublin Core meta-data from LOM RDF meta-data in a straightforward and standardized way. Of course, if the software understands qualified Dublin Core, no ”dumb-down” is necessary.

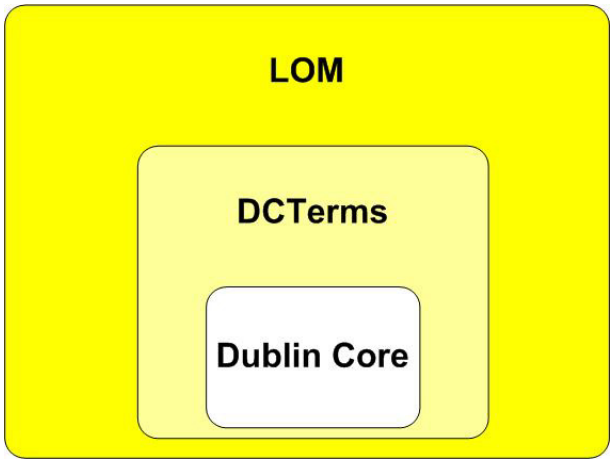



Figure 3.2: The relationship from LOM to Dublin Core

### Langstring

In the LOM standard, many of the entries are either of the Data type *Langstring* or *Vocabulary*. The first one can be easily realized in RDF. When encoding a string in a specific language, we use the language tag for RDF literal. In the XML serialization, this corresponds to the xml:lang attribute, as described in [91] and [38]. An example of a language-tagged string follows:

```
<rdf:Description rdf:about="http://www.test.com/">
  <dc:title xml:lang="en">A test</dc:title>
</rdf:Description>
```

Here ”en” is a language code conforming to RCF1766 (see [61]). In order to encode strings in several languages, which is needed for the LangString construct, we use the rdf:Alt construct:

```
<rdf:Description rdf:about="http://www.test.com/">
  <dc:title>
   <rdf:Alt>
    <rdf:li xml:lang="en">A test</rdf:li>
```

```
    <rdf:li xml:lang="de">Ein Test</rdf:li>
    <rdf:li xml:lang="se">En test</rdf:li>
   </rdf:Alt>
  </dc:title>
</rdf:Description>
```

This technique allows us to separate the original title from translations, as the first title is the default (according to the semantics of rdf:Alt). It also allows Dublin Core-only RDF parsers to understand what the title is, via the ”dumb-down” algorithm. Finally, it allows us to add translations in separate RDF documents. A necessary prerequisite for this is that rdf:Alt instances are given a URI so that it can be referenced.

### Vocabularies

Vocabularies are represented in several different ways in this binding. The fundamental idea is that the (source, value) construct in LOM is best represented in RDF using the (namespace, value) construct that is naturally contained in a resource URI in RDF. Thus, vocabulary values are resources, and the source of a vocabulary is implicit in the URI of a resource.

This binding provides RDF resources for all the restricted vocabulary terms defined in LOM. These resources can be used directly as values of the corresponding property, for example:

```
<lom-life:status rdf:resource=
  "http://ltsc.ieee.org/2002/09/lom-lifecycle#Draft"/>
```

These resources are in turn described in the LOM RDF schemas, which give them their official label and description. In the case of the *Draft* term, it is described as being of type *Status*, as are all the terms in the LOM *Status* vocabulary.

Users of the binding are free to define their own RDF resources for use as values in vocabularies, for example:

```
<lom-life:status rdf:resource=
   "http://www.myVoc.org/vocab#ReleaseCandidate"/>
```

In the RDF schema describing this vocabulary, the *ReleaseCandidate* resource would also be described as an instance of the *Status* class. In this way, extending the LOM vocabularies is as simple as defining new instances of the relevant RDF schema classes.

Thus, vocabularies will need to be explicitly translated to RDF. This convention leads to some difficulties when interfacing with the XML binding, where vocabularies are not explicitly defined in this way. Further development in this area will be necessary.

### Using vocabularies for Properties

In several cases, the LOM vocabulary item is not to be used as the object of an RDF Statement, but rather as the predicate in the statement. This is the case with element *7.1 Relation.Kind*. An example could look like:

```
<dcterms:hasPart rdf:resource=
   "http://www.ieee.org/someContent.html"/>
```

Here the Relation is of Relation.Kind *hasPart*. LOM defines twelve terms for this vocabulary, and each of them corresponds to a separate property. Defining new vocabularies for this element is as simple as for the "Status" example above. The only difference is that in this case, instead of defining new instances of the "Status" class, one would need to define new sub-properties of the property *dc:relation*. This new property is an RDF resource, and thus the same remarks apply: explicit translation of vocabularies to RDF is necessary, the terms can be described in an RDF schema, and care must be taken when interfacing with the XML binding.

### Element encodings

There are many places in the LOM standard where string literals that are not intended to be human-language text are used as values, such as dates, whole numbers or ranges of numbers, or language tags. When encoding such values, the LOM RDF binding takes the approach of tagging the value with a data type. The construction for dates, for example, looks like:

```
<dc:date>
  <dcterms:W3CDTF>
    <rdf:value>1999-03-05</rdf:value>
  </dcterms:W3CDTF>
</dc:date>
```

This construct is used to indicate that the string "1999-03-05" is encoded using the W3C Date and Time format. As mentioned before Dublin Core Qualifiers define several useful element encodings such as W3CDTF for dates and RFC1766 for language tags. In some other cases, the LOM RDF binding defines new data types for similar fields. Using this technique, the RDF data becomes self-describing in a very useful way.

### Using Metametadata

Generally, the metametadata category is obsolete in RDF, as RDF itself comes with good support for metametadata. Two ways to describe such information are provided by RDF, and both rely on reusing the usual metadata properties from LOM and Dublin Core. These properties are applied to either:

- the URI representing the RDF document containing the metadata
- a set of RDF statements (using the RDF reification mechanism)

### Classifications

This is the most complex category of all in LOM. Instead of describing the full path to the describing "taxon" element in each metadata instance, the RDF binding allows taxonomies to be described separately from each metadata instance. The idea is to represent a hierarchical taxonomy separately, and then point into nodes in this hierarchy when classifying resources. At the same time, it is possible to reuse the subject classifications from Dublin Core Qualifiers. Using this will then look like:

```
<dc:subject rdf:resource=
  "http://www.myVocabulary.com/taxonomy#A01.047"/>
```

In this example, the value is an element in a subject classification. This "taxon" can be described in a separate RDF document, and annotated using ordinary RDF metadata. In chapter 4.2 you can find detailed information on the use of taxonomies for the classifications category.

### VCards

Another common LOM value type is the VCard, which is used in several places to describe a person or other entity. In the XML binding, VCards are inserted literally, without XML markup. In the RDF binding, the VCard is made into a resource, with properties such as *vCard:FN, vCard:ORG* being used to describe the VCard properties of that entity. The RDF properties are taken from the VCard RDF binding ([92]). Describing the entity that made an annotation in LOM could for example look like

```
<lom-ann:entity>
  <lom-base:Entity>
    <vCard:FN>Anna Eriksson</vCard:FN>
    <vCard:ORG>ABC Inc.</vCard:ORG>
  </lom-base:Entity>
</lom-ann:entity>
```

### 3.2.3 Example

In contrast to the XML binding which tries to cover the whole structure of the LOM categories, the RDF binding offers explicit suggestion which RDF constructs to use for different LOM elements. At the beginning of this section, we have seen the LOM category *lifecycle* in XML notation. This category in the RDF binding can be seen in table 3.1.

In the next chapter, when we will show how to use the LOM-RDF binding to annotate learning ressources with metadata, we will see a detailed example for the use of LOM *2.3. Contribute*

| **LOM element** | **Usage guidelines** |
|---|---|
| 2.1 Version | Use *lom-life:version*, pointing to a textual description. |
| 2.2 Status | Use *lom-life:status*.<br>As values, use instances of *lom-life:StatusType*.<br>For LOM restricted vocabulary, use the values:<br>*lom-life:Draft,lom-life:Final,lom-life:Revised*<br>or *lom-life:Unavailable*. |
| 2.3 Contribute | There are three possibilities:<br><br>1. Use *dc:publisher* pointing to the contributing entity, in VCard format. The role is implicitly the same as the LOM ”publisher” role, and the date implicitly the date contained in *dcterms:issued* or, if this is missing, *dc:date*.<br><br>2. Use *dc:creator* pointing to the contributing entity, in VCard format. The role is implicitly the same as the LOM ”author” role, and the date implicitly the date contained in *dcterms:issued* or, if this is missing, *dc:date*.<br><br>3. Use an *rdfs:subPropertyOf dc:contributor* with *rdf:value* pointing to the contributing entity, in VCard format, and *dc:date* pointing to the date of the contribution. The resource should e of type *lom-life:Contribution*.<br><br>The contributing entity should be a resource of type *lom:Entity*. The *dc:date*, *dcterms:issued* or *dcterms:created*should point to an instance of *dcterms:W3CDTF*. |

Table 3.1: The LOM category lifecycle from the RDF binding guide


Figure 3.3: The RDF-Query model for LOM category 4.Technical

## 3.3 The LOM RDF Query model

So the current LOM RDF binding specification is a large table with plain English definitions how a RDF expression of LOM looks like. To formally describe and visualize this binding we have chosen to use a combination of RDF Schema with a Query Model. A Query Model is a single RDF record containing all LOM RDF elements in reified triples. The RDF Schema is used to define classes, properties, and whenever possible, range and domain restrictions on some of these properties. While this LOM RDF Schema restrictions are of global character, the LOM Query Model specifies them in the specific context of a LOM record. For building the LOM Query model, we used the technique of reifications, which are RDF constructs referring to statements rather than expressing them. The LOM Query Model is in effect a query built with a set of reified statements where regular resources or literals sometimes are replaced with variables. By repeating variables in several reified statements a tree is formed. This tree is a generic mirror of how a full LOM record would look like and hence is very suitable as a visualization

of the LOM RDF binding (see fig. 3.3). The LOM Query Model was constructed with the help of the general purpose RDF editing and visualization tool Conzilla [72]. For more information about Query models we refer to [75].

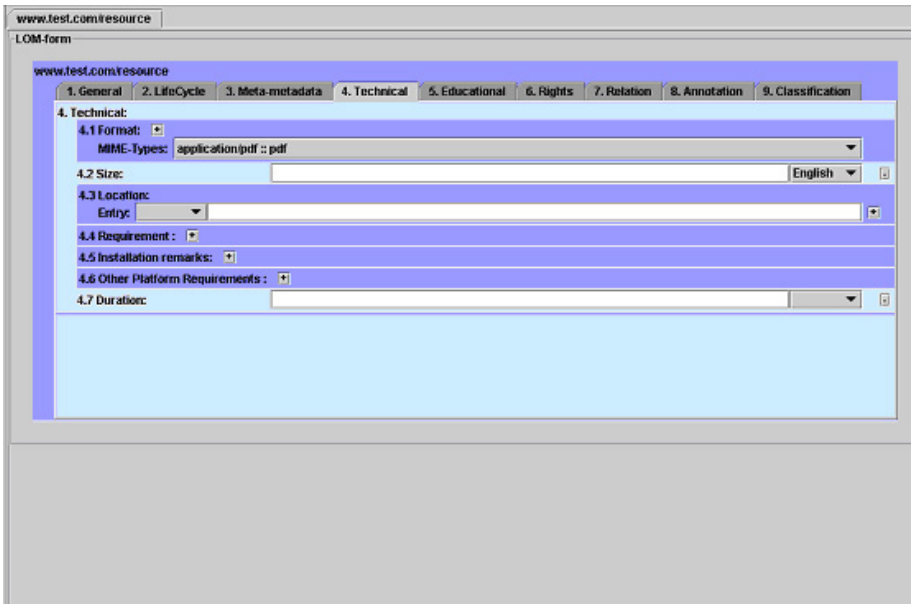



Figure 3.4: The LOM Category 4.Technical in the RDF Editor

Furthermore the LOM Query Model allows us to construct tools for validation, querying, presentation or editing of LOM records.

Based on the LOM RDF Query model, we can create various LOM editors via a complementary Form Model (which specifies how the Query Model should be presented in a form). The LOM Query model can be loaded into a RDF based editor and he can extract all informationen about the different elements, their domains or ranges. The full LOM editor in fig. 3.4 makes use of the entire LOM Query Model. Based on the LOM Query Model it expects a resource, a literal or even a vocabulary entry. Provided with a URL for vocabularies, we can present the different choices in a drop down menu, as for the MIME types in 4.1 Format in our example. Alternative editors using e.g. a subset of LOM can be created from the same Query Model. Additionally, the vocabularies used in LOM can easily be extended or changed by including them in the Form Model. The screenshot in fig. 3.4 shows the editor SHAME, build at the Swedish learning lab. For more details about this editor we refer to [75].

# Chapter 4

# Metadata in eLearning context

ELearning, or online learning, stands for all forms of Internet-enabled and/or computer supported learning. It refers to the use of computer and computer network technologies to create, deliver, manage and support learning, usually independent of specific locations or times. eLearning can involve complete online courses, where all aspects of learning, from learner enrolment to tuition and support take place online. At the other end of the eLearning spectrum, these elements may well take place in a face to face situation, with only the learning resources available on the internet. Accessibility of learning resources is accompanied by the need to annotate the resources with rich, standardized and widely used metadata.

This chapter gives you an overview over the use of metadata in eLearning as well as about innovative approaches and techniques we developed for enhanced eLearning scenarios: First we introduce the LOM-RDF metadata set we have used to annotate complete courses in the context of eLearning. We give you an overview of classification schemes as well as different approaches and techniques for content classification. In the context of an electronic learning repository we will discuss how related resources can automatically be displayed using our methods. For easy annotation the author has also developed a set of inference rules, which we introduce in the background of a RDF based P2P network, where these rules can be used to gain richer result sets.

## 4.1 Annotating complete courses

### 4.1.1 Elements for distributed computer science education

The ULI project (University teaching network for computer science) was funded by the German government, and established an exchange of course material, courses and certificates in the area of computer science. 11 German universities with 18 different professors had agreed to exchange their courses and allowed students from one university to attend courses at another university, using advanced eLearning technologies. For more information about the project, we refer to ([86]). To make this exchange of courses possible, we started to use metadata to annotate the learning materials in complete online courses.

Though the courses usually differ in the kind and amount of learning materials they use, their use of learning resources is surprisingly homogeneous. The average course is divided in 6 to 7 units or knowledge modules which themselves can be split into 3 to 7 learning resources. This leads to an average number of about 35 learning resources per course, with a learning resource being the slides of the lecture, a video or any other set of pages dealing with one subject.

For annotating these resources, we defined a best-practice subset of 17 elements which is summarized in the following table, using the categories defined in LOM. This subset was first published in [17]. To technically realize the annotation of courses with metadata we used the RDF binding of LOM as introduced in the last chapter.

It turned out that these 17 elements were enough to annotate and query our resources, and represent a compromise between more abstract and more detailed annotation sets. The annotations of one whole course could be included in a single RDF file. All RDF-triples were then imported into a relational database, to customize the display of the resources described and to query for specific learning resources. Fig. 4.1 shows you the placement of this subset among the standards introduced in chapter 2.

This LOM subset was used in various finished projects and is still in use for metadata annotation in the learning environments of the following projects:

- The EU IST project ELENA ([41])
- Work package 5 of the EU Network of Excellence PROLEARN ([80])

Table 4.1: Our LOM subset

| **LOM-Catgory** | **Element** | **Technical realization** |
|---|---|---|
| 1. General | 1.2 Title | **dc:title** as langstring |
| | 1.3 Language | **dc:language** in RFC1766 format |
| | 1.4 Description | **dc:description** as langstring |
| 2.Lifecycle | 2.3 Contribute | **dc:creator** with a **lom:entity** and the author in vCard format ”name surname” **dcterms:created** with the date in W3C format |
| 4.Technical | 4.1 Format | **dc:format** in MIME-Type |
| 5.Educational | 5.2 Learning Resource Type | **rdf:type** this element links to a self-defined vocabulary |
| 6.Rights | 6.3 Description | **dc:rights** if there are no copyright restrictions this element is blank, otherwise it holds the owner‘s name |
| 7. Relation | | **dcterms:hasFormat** **dcterms:isFormatOf** **dcterms:hasPart** **dcterms:isPartOf** **dcterms:hasVersion** **dcterms:isVersionOf** **dcterms:requires** **dcterms:isRequiredBy** |
| 9.Classification | | **dc:subject** for content classification. This element links to an entry in a hierarchical scheme, that is an instance of **lom_cls:Taxonomy** (see next section) |

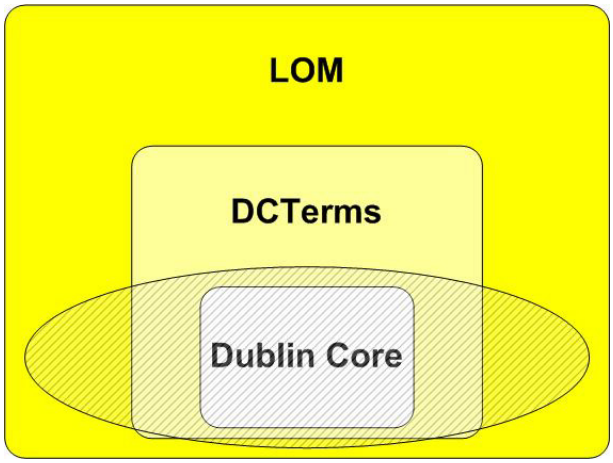



Figure 4.1: Our LOM subset

### 4.1.2 An example course for software engineering

In our system a whole course is represented by a single rdf-file. In the following you see a simple example of a course consisting of one lecture-unit with the slides and the streaming video of the lecture as learning resources, containing description, title, rights, language and a classification entry. The course is described in a RDF file with the metadata elements from our LOM-subset, following the guidelines from the RDF binding of LOM, discussed in section 3.2

Note that for better readability the vocabularies for the resource type and content classification are defined as entities in line 03-05

```
01 <?xml version='1.0' encoding='ISO-8859-1'?>
02 <!DOCTYPE rdf:RDF [
03 <!ENTITY type "http://telemann.kbs.uni-hannover.de:3333/olr/olr_v9#">
04  <!ENTITY SWEBOK "http://www.kbs.uni-hannover.de/ULI/SWT_Ontologie.rdf#">
05 ]>
06
07 <rdf:RDF
08 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
09 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
10 xmlns:dc="http://purl.org/dc/elements/1.1/"
11 xmlns:dcterms="http://purl.org/dc/terms/"
12 xmlns:lom_cls="http://www.imsproject.org/rdf/imsmd_classificationv1p2#"
13 xmlns:lom="http://ltsc.ieee.org/2002/09/lom-base#"
14 xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#">
15
16 <rdf:Description rdf:ID="SELecture">
17 <rdf:type rdf:resource="&type;Course"/>
18 <dc:title>
```

```
19  <rdf:Alt>
20     <rdf:li xml:lang="en">SE Lecture in winter 2001</rdf:li>
21     <rdf:li xml:lang="de">ST Vorlesung Winter 2001<rdf:li>
22 </rdf:Alt>
23 </dc:title>
24 <dc:creator>
25  <lom:entity>
26     <vCard:FN>Friedrich Steimann</vCard:FN>
27  </lom:entity>
28 </dc:creator>
29 <dcterms:created>
30  <dcterms:W3CDTF>
31     <rdf:value>2001-09-15</rdf:value>
32 </dcterms:W3CDTF>
33 </dcterms:created>
34 <dcterms:hasPart>
35 <rdf:Seq>
36      <rdf:li rdf:resource="#Unit1"/>
37  </rdf:Seq>
38 </dcterms:hasPart>
39 </rdf:Description>
40
41 <rdf:Description rdf:ID="Unit1">
42 <rdf:type rdf:resource="&type;Unit"/>
43 <dc:title>
44  <rdf:Alt>
45      <rdf:li xml:lang="en">1. Lecture unit 22.10.2001</rdf:li>
46      <rdf:li xml:lang="de">1. Stunde 22.10.2001</rdf:li>
47  </rdf:Alt>
48 </dc:title>
49 <dc:description>
50  <rdf:Alt>
51      <rdf:li xml:lang="en">Introduction to the course</rdf:li>
52      <rdf:li xml:lang="de">Einleitung</rdf:li>
53  </rdf:Alt>
54 </dc:description>
55 <dcterms:isPartOf rdf:resource="#SELecture"/>
56 <dcterms:hasPart>
57  <rdf:Seq>
58      <rdf:li rdf:resource="http://www.kbs.uni-hannover.de/Lehre/
                                                    SWT1/S1T1.pdf"/>
59      <rdf:li rdf:resource="http://www.mml.uni-hannover.de/
                               meta/kbs.cgi?SWT1WS01S01T1.rm"/>
60 </rdf:Seq>
61 </dcterms:hasPart>
62 </rdf:Description>
63
64 <rdf:Description rdf:about="http://www.kbs.uni-hannover.de/Lehre/
                                                    SWT1/S1T1.pdf">
65 <rdf:type rdf:resource="&type;Resource"/>
66 <dc:title>
67  <rdf:Alt>
68      <rdf:li xml:lang="en">Slides for the first lecture</rdf:li>
69      <rdf:li xml:lang="de">Folien der ersten Stunde</rdf:li>
70  </rdf:Alt>
71 </dc:title>
72 <dc:description>
```

```
73  <rdf:Alt>
74      <rdf:li xml:lang="en">Overview of the discipline. And a brief
                    introduction to the Function-Point-Method</rdf:li>
75      <rdf:li xml:lang="de">Ueberblick ber das Fach. Kurze
                    Einfuehrung in die Function-Point-Methode</rdf:li>
76  </rdf:Alt>
77 </dc:description>
78 <dc:rights> balzert< /dc:rights>
79 <dc:language>
80 <dcterms:RFC1766>
81     <rdf:value>en</rdf:value>
82  </dcterms:RFC1766>
83 <dc:subject rdf:resource="&SWEBOK;MeasSWDevelopment"/>
84 <dc:subject rdf:resource="&SWEBOK;FuncDesMeasures"/>
85 <dcterms:hasFormat rdf:resource="http://www.mml.uni-hannover.de/
                                  meta/kbs.cgi?SWT1WS01S01T1.rm"/>
86 <dcterms:requires rdf:resource="http://www.kbs.uni-hannover.de/
                                        Lehre/SWT1/S3T2.pdf"/>
87 <dcterms:isPartOf rdf:resource="#Unit1"/>
88 <dc:format>
89   <dcterms:IMT>
90       <rdf:value>application/pdf</rdf:value>
91   </dc:terms:IMT>
92 </dc:format>
93 </rdf:Description>
94
95 </rdf:RDF>
```

Line 16-39 define the structure of the complete course, Line 41-62 of the unit, using rdf:Sequence to link to the child resources. Also between these description tags stands all metadata concerning the whole course respectively the unit. Between line 64 and 93 is the metadata concerning PDF-slides as one of the learning resources.

In the following we give you a brief overview of the subset of LOM, we use to annotate our courses in software engineering.

**rdf:type** (line 17, 42, 65): *rdf:type* links to entries in a self-defined vocabulary, describing different resource types. The type *course* or *unit* stands for a self-created structure in contrast with a learning resource, which is physically present somewhere in the internet.

**rdf:ID** (line 16, 41): Each structure element like the course or a unit, that can not be physically located via its URI, needs a unique ID to be referenced by inside the database.

**dc:title** (line 18ff, 43ff, 66ff): The title of the resource will be displayed in the navigation tree. Often the title is displayed as the result of a query. There-

fore it is very useful to choose a title that is short, but rich and understandable.

**dc:creator, dcterms:created** (line 24ff, 29ff): These elements to describe author and time are usually only used once in the course metadata description and then inherited to every unit and sub-unit inside the system (see section 4.5.

**dc:description** (line 49ff, 72ff): Contains a short description that can be displayed, when opening the resource.

**dcterms:isPartOf, dcterms:hasPart** (line 34ff, 55, 56ff, 87): This elements are used to define the structure of a course.

**dc:rights** (line 78): Some parts of the lecture ”Software Engineering I” are based on a lecture by Prof. Balzert from the university of Bochum. We use this element to cope with any problems of copyright by identifying resources from other authors.

**dc:language** (line 79ff): A useful element for querying resources to make sure that you will be able to understand the language of the learning resources you get as a result.

**dcterms:hasFormat** (line 85f): As the lectures are represented by a pdf-file with the slides and a streaming video, we use dcterms:hasFormat and dcterms:isFormatOf to link them with each other. We can then receive better search results by eliminating resources that are only different technical formats of an already known search-result.

**dcterms:requires** (line 86f): This metadata instance from the *Relation* category is needed for adaptive version of the script. It shall prevent readers from starting with chapters that build up on other chapters the reader has not yet looked at.

**dc:format** (line 88ff) This element is used to ensure that the resources are displayed in the best possible way. For *pdf* or *avi*, a new window will open, including a small navigation bar for ”Play”, ”Pause” and ”Rewind” in case of *avi*.

**dc:subject** (line 83f): This element is used to classify the content of the learning resource. We will discuss it in detail later in section 4.2.

## 4.2 Content Classification with dc:subject

General metadata annotation is useful, but not enough for finding specific resources. Another important metadata task is classifying the content of a learning resource. It is obvious that self-defined keywords can only be a first solution to this problem. To provide for better search results, keywords used should be part of larger hierarchical classification scheme, in order to specify both sub- and supertopics. Defining a private scheme for a specific field unfortunately works only in the closed micro world of a single university. To be more general, we therefore decided to use internationally already accepted classification systems. In the following we will introduce three different solutions we came up with. Excerpts from this section were first published in [18].

### 4.2.1 One scheme - ACM CCS

The *ACM Computer Classification system* (ACM CCS) ([2]) has been used by the *Association for Computer Machinery* ACM since several decades to classify scientific publications in the field of computer science. On the basic level, we find 11 nodes that split up in two more levels. Part of the classification hierarchy is reproduced in the following overview.

- A. General Literature
- B. Hardware
- C. Computer Systems Organization
- D. Software
  - D.0 GENERAL
  - D.1 PROGRAMMING TECHNIQUES
  - D.2 SOFTWARE ENGINEERING
  - D.3 PROGRAMMING LANGUAGES
  - D.4 OPERATING SYSTEMS
  - D.m MISCELLANEOUS
- E. Data
- F. Theory of Computation
- G. Mathematics of Computing
- H. Information Systems
- I. Computing Methodologies

  - I.0 GENERAL
  - I.1 SYMBOLIC AND ALGEBRAIC MANIPULATION
  - I.2 ARTIFICIAL INTELLIGENCE
    - I.2.0 General
    - I.2.1 Applications and Expert Systems
    - I.2.2 Automatic Programming
    - I.2.3 Deduction and Theorem Proving
    - I.2.4 Knowledge Representation Formalisms and Methods
    - I.2.5 Programming Languages and Software
    - I.2.6 Learning
    - I.2.7 Natural Language Processing
    - I.2.8 Problem Solving, Control Methods, and Search
    - I.2.9 Robotics
    - I.2.10 Vision and Scene Understanding
    - I.2.11 Distributed Artificial Intelligence
    - I.2.m Miscellaneous
  - I.3 COMPUTER GRAPHICS
  - I.4 IMAGE PROCESSING AND COMPUTER VISION
  - I.5 PATTERN RECOGNITION
  - I.6 SIMULATION AND MODELING
  - I.7 DOCUMENT AND TEXT PROCESSING
  - I.m MISCELLANEOUS
- J. Computer Applications
- K. Computing Milieux

The classification has a fourth level containing unordered keywords, thus including about 1600 entries on all four levels. For our use of the ACM CCS, we also numbered the keyword lists in the fourth level to receive unique IDs like: *B.1.1.2* for the keyword ”Micro programmed logic arrays” that is accessible via the taxon path: *Hardware(B)/CONTROL STRUCTURES AND MICROPROGRAMMING(B.1)/Control Design Styles(B.1.1)*.

The complete ACM classification RDF-file can be found at [13].

In the context of our projects this classification turned out to fit very well, because it covers the whole field of computer science, just as the different courses cover the whole discipline. Typically a course received approximately 5 classification entries from the ACM CCS, and one entry per chapter was a typical distribution. Therefore classification with ACM CCS is excellent for the exchange of complete knowledge modules. If we look for a taxonomy that allows us to annotate different submodules and small, single learning resources, we have two other possibilities: extending the ACM CCS, or looking for another classification system.

### 4.2.2 More details - extending ACM CCS

In an article for the AI magazine in the mid 80's D. Waltz suggested an extension of the node I.2 Artificial Intelligence of the ACM CCS [98]. He refined the keywords in the fourth level as nodes for two more levels, gaining about 100 more entries focussing on the field of artificial intelligence. As an example, the keyword entry "games" in the node I.2.1 Applications and Expert Systems was extended to:

- I.2.1 Applications and Expert Systems
  - I.2.1.0 Cartography
  - I.2.1.1 Games
    - I.2.1.1.0 Chess
    - I.2.1.1.1 Checkers
    - I.2.1.1.2 Backgammon
    - I.2.1.1.3 Biding Games
    - I.2.1.1.4 Wagering Games
    - I.2.1.1.5 War Games
    - I.2.1.1.6 Games, Other
  - I.2.1.2 Industrial automation
  - I.2.1.3 Law
  - I.2.1.4 Medicine and science
  - I.2.1.5 Natural language interfaces
  - I.2.1.6 Office automation

As we had labelled the keywords of the fourth level for the use of the classical ACM CCS as well, it was easy for us to adapt his suggestions to the modern (1998) version of the ACM CCS, leading to a quite detailed classification scheme to classify our learning resources in the discipline of Artificial Intelligence. The complete RDF-file can be found at [14].

### 4.2.3 One scheme for a specific sub-discipline - the SWEBOK

For our course in software engineering we used a different classification scheme, the Guide to the *Software Engineering Body of Knowledge* (SWEBOK). The SWEBOK has been developed in context of an IEEE/ACM working group. On their webpage [54] the working group states their goal as follows: "The purpose of this guide is to provide a consensually-validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body

of Knowledge supporting that discipline." Almost 500 software engineering professionals from 41 countries have hierarchically structured the field of software engineering in 10 Knowledge Areas and almost 300 topics, based on their number of publications. This therefore represents a very nice example for a consensually derived classification scheme in a specific community. The main knowledge areas are:

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

The complete RDF-file can be found at [15]. To use the SWEBOK for our courses in the context of exchanging learning resources with other peers, it was important to define mappings between these knowledge areas and the ACM CCS. These mappings are for example:
*D.2.1 Requirements/Specifications* maps with *Software requirements*
*D.2.2 Design Tools and Techniques* maps with *Software engineering tools and methods / Software tools / Software design tools* and of course the whole *Software design* (since *D.2.10 Design* no longer used as of January 1998)
*D.2.9 Management* maps to *Software configuration management* and *Software engineering management*, same as *K.6.3 Software Management*

So we finally had what we needed: One scheme to use in a world wide context of exchanging learning resources in the field of computer science, plus two specialized schemes to classify the content of our own lectures, detailed enough to differentiate between the content of single learning resources, but mapping perfectly to the global scheme, if other peers want to access the resources. The ACM CCS scheme, extended with the subdisciplines from section 4.2.2 is used again in chapter 5 in another context.

### 4.2.4 Machine readable classification schemes

As in mentioned in chapter 3.2 the attribut *dc:subject* points to a node in a classification scheme. Therefore our different classification schemes had to be written in machine readable format. By definition from the RDF binding of LOM all instances have to be subclasses of lom_cls_Taxonomy, ordered with the element lom_cls:taxon, where lom_cls:rootTaxon denotes any root element in this taxonomy. An short excerpt from the ACM-classification can be seen below. *Dcterms:isVersionOf* is used to model relationships inside the classification scheme. Note that the entity ACM, defined in line 04 is the classification scheme itself.

```
<?xml version='1.0' encoding='ISO-8859-1'?>

<!DOCTYPE rdf:RDF [
<!ENTITY ACM "http://www.kbs.uni-hannover.de/Uli/ACM_CCS.rdf#">
]>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://dublincore.org/2000/03/13/dcq#"
xmlns:lom_cls="http://www.imsproject.org/rdf/imsmd_classificationv1p2#"
xmlns:lang="http://www.kbs.uni-hannover.de/Uli/lang.rdf#">


<dcterms:SubjectScheme rdf:about="&ACM;ACMClassification">
<rdfs:label>The ACM Computing Classification System (1998)</rdfs:label>
</dcterms:SubjectScheme>

<lom_cls:Taxonomy rdf:about="&ACM;">
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="en">Ontology for Computer Science</rdf:li>
    <rdf:li xml:lang="de">Fachontologie fuer Informatik</rdf:li>
    <rdf:li xml:lang="sv">Ontologi foer datavetenskap</rdf:li>
  </rdf:Alt>
</dc:title>

<lom_cls:rootTaxon>
<ACM:ACMClassification rdf:about="&ACM;B">
<rdf:value>B</rdf:value>
<dc:title xml:lang="en">Hardware</dc:title>

    <lom_cls:taxon>
    <ACM:ACMClassification rdf:about="&ACM;B.1">
    <rdf:value>B.1</rdf:value>
    <dc:title xml:lang="en">CONTROL STRUCTURES AND MICROPROGRAMMING</dc:title>
    <dcterms:isVersionOf rdf:resource="&ACM;D.3.2"/>

        <lom_cls:taxon>
        <ACM:ACMClassification rdf:about="&ACM;B.1.1">
        <rdf:value>B.1.1</rdf:value>
```

```
        <dc:title xml:lang="en">Control Design Styles</dc:title>

            <lom_cls:taxon>
            <ACM:ACMClassification rdf:about="&ACM;B.1.1.0">
            <rdf:value>B.1.1.0</rdf:value>
            <dc:title xml:lang="en">Hardwired control</dc:title>
            </ACM:ACMClassification>
            </lom_cls:taxon>

        </ACM:ACMClassification>
        </lom_cls:taxon>

    </ACM:ACMClassification>
    </lom_cls:taxon>

</ACM:ACMClassification>
</lom_cls:rootTaxon>

</lom_cls:Taxonomy>
</rdf:RDF>
```

To classify a learnig ressource with the content B.1.1.0 for example, the element in the course description would be:

```
<dc:subject rdf:resource="&ACM;B.1.1.0">
```

The complete ACM classification RDF-file can be found at [13].

### 4.2.5 Content classification in the semantic-web tower

In the context of the semantic-web tower, as pictured in fig. 3.1, our content classification systems would be located in the the *ontology vocabulary layer*. Ontologies provide a shared and common understanding of a domain that can be communicated between people and application systems like agents. They are developed to facilitate knowledge sharing and reuse [47]. In the simplest case, an ontology describes a hierarchy of concepts related by relationships [50], as in our example. In a more sophisticated ontology, suitable axioms are added in order to express other relationships between concepts and to constrain their intended interpretation. Allthough RDFS is suitable to model a simple ontology like our classification system, we cannot expect the RDFS constructs hierarchies and restrictions to be rich enough to create more sophisticated ontologies (see [6]). We will return to this problem in chapter 5.

## 4.3 Open learning repository (OLR)

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

For further information about the OLR we refer to [34].

### 4.3.1 Architecture

The basic architecture of the different OLR versions is the same. The first two versions were php-based. Our newest Open Learning Repository, the OLR3 system however, that we use for our artificial intelligence course, is implemented in Java and works as a JavaServlet, running on an Enhydra [42] Application Server (open source software). It is connected to an Oracle Database via JDBC, which is used to store the metadata entered by course authors and students. RDF schemes, needed for either the annotation of metadata or the import of externally prepared metadata, can come from anywhere in the internet.

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

Any referenced RDF schema will be parsed using the SiRPAC RDF parser [83], whereas imported RDF files are parsed by a VRP RDF parser [87] , which provides semantically checks against given RDF schema rules.

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

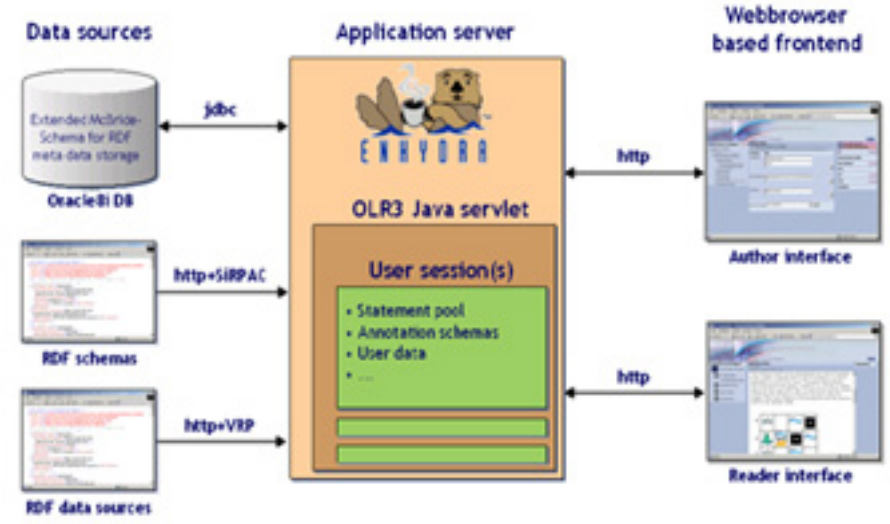



Figure 4.2: The architecture of the OLR3 system

### 4.3.2 Reader Interface Layout

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

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

### 4.3.3 Querying the course metadata

We assume that we can access the metadata RDF triples with a view STATEMENT(Subject,Predicate,Object).Let us also consider that every resource has an ID that was automatically generated, when the RDF file was stored in the database. We

will now have look at how the metadata is queried in more detail. The following was first published in [19]. Let us return to our software engineering course, and the way it was represented in the OLR2.

The complete metadata of the course an the complete SWEBOK Taxonomy can be found in the database. For example:

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

(Note: For easier understanding, we assume, that the complex structures of *dc:title* or *dc:language* are already ”dumbed-down” (see section 3.2.2) in the database)

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

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

SELECT **Object** FROM STATEMENTS WHERE **Subject**=’[577]’ AND **Predicate**=’dc:subject’

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

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

We decided to display the entries as hyperlinks, leading to a brief description of the certain subject, that this resource was described as belonging to by the entry (see fig. 4.3).

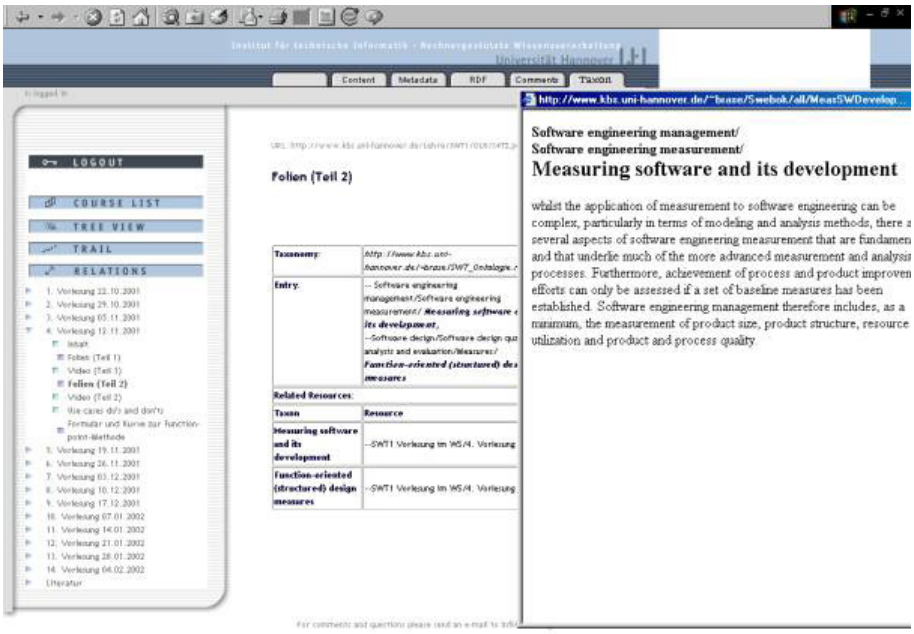



Figure 4.3: Taxon entries for a specific resource in the OLR2

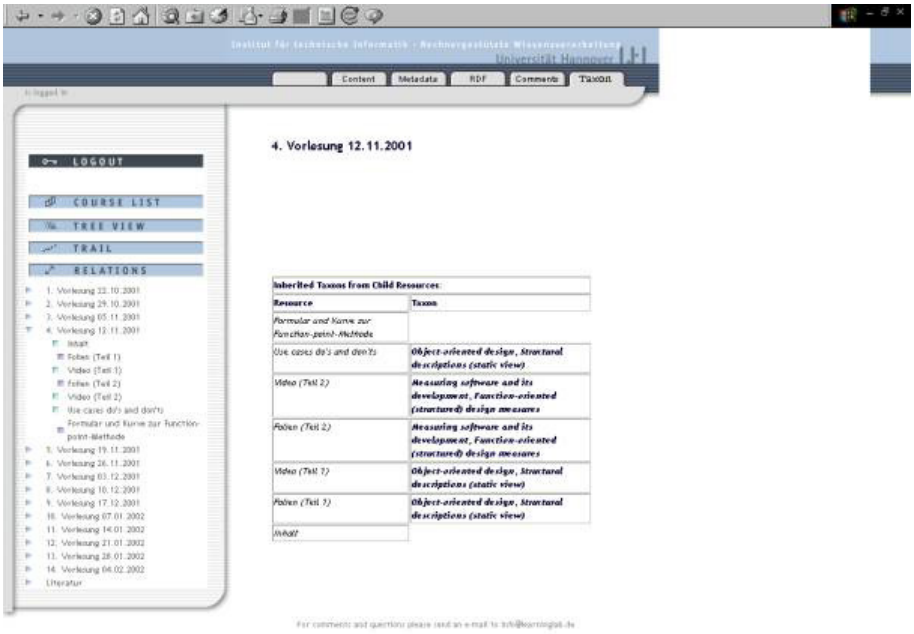



Figure 4.4: Related resources for a specific resource in the OLR2

### Related Resources

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

SELECT **Subject** FROM STATEMENTS WHERE **Predicate**='dc:subject'
AND **Object**='http://www.kbs.uni-hannover.de/ULI/SWT_Ontologie.rdf#MeasSWDevelopment')

and displayed as a hyperlink to the resource (see fig. 4.4).

### Lecture classification

Since a course is represented by a RDF file, a single learning resource can be annotated via its RDF description. The whole course, or a single lecture exists

only inside the schema. Of course it has a description that we could annotate, but since we want to be as flexible as possible in creating different views of different lectures from the same learning resources, lectures and courses inherit their taxon-entries from their child-units. If the user chooses the layer ”taxon” while he is not in a learning resource, but looking at the structure of a unit, the sub-units of this unit are retrieved via:

SELECT The sub-units names FROM STATEMENTS WHERE **Subject** IN
(SELECT **Object** FROM STATEMENTS WHERE **Predicate** =’dcterms:hasPart’
AND **Subject**=The ID of the current unit)

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

### 4.3.4 Related work

Even though the number of digital libraries and Learning repositories have started to increase rapidly, we haven‘t yet encountered a similar approach in classifying resources than our use of the LOM Standard together with a machine readable rdf-file as a classification scheme. Whereas the use of metadate to annotate resource is widely accepted and common, the traditional way of annotating the content of a resource is by using keywords, and query these keywords later via a full text search.

Even Portal the Digital library of the ACM [78] itself uses the ACM CSS Classification for a simple keyword annotation of the resources, the same as for example the SMETE Digital Library [84].

We decided to develop OLR3 on the basis of regular web-browsers, to give every student the opportunity to access our courses without further software installation. This is a different approach from document viewers like CREAM from the University of Karlsruhe [55], although the schema-driven metadata approach is very similar.

A similar approach was developed in the K-Med project by the university of Darmstadt [56]. The courses presented in their editor are also only represented by metadata-schemes, using LOM metadata. The system however is focused on

a XML version of LOM and therefore lacks our possibility to include new, self-created metadata elements in the course schema, when using RDF-based LOM.

Conzilla the Concept-Browser, developed by the CID at the KTH Stockholm [72]and briefly menioned in chapter 6.3.1 is a very interesting metadata-focused tool, with an editor using the LOM standard, which is very similar to our approach. It is however no course editor. Its main goal is to present complete fields of science and their concepts.

A simple editor for LOM RDF metadata was also developed at the CID. In chapter 6.3.1 we have briefly introduced the SHAME editor [75]. Based on RDF Query models, this editor allows creation and editing of any LOM RDF record. It has been the first editor to use the RDF model of LOM.

## 4.4 Metadata for a P2P network

### 4.4.1 Decentralized P2P networks

As discussed before, exchanging learning resources is one of the greatest advantages in eLearning and eTeaching. A system for the exchange of learning resources however has to build on the fact that most universities or departments have already established their own way of storing their learning resources and do so locally in almost all cases. Since all these institutions are not interested in losing this independence by giving their learning resources away to central ”knowledge pools” the best way to establish such an exchange system is by building up a *peer-to-peer* (P2P) network.

### 4.4.2 Edutella infrastructure

The Edutella project (see [40] ,[70] and [69]) addresses these shortcomings by building on RDF as the basis for a so-called schema-based P2P network. The project is a multi-staged effort to scope, specify, architect and implement an RDF-based metadata infrastructure for P2P-networks based on the recently announced JXTA framework by SUN Microsystems [49]. The initial Edutella services are

**Query Service:** Standardized query and retrieval of RDF metadata.

**Replication Service:** Providing data persistence / availability and workload balancing while maintaining data integrity and consistency.

**Mapping Service:** Translating between different metadata vocabularies to enable interoperability between different peers.

**Mediation Service:** Define views that join data from different metadata sources and reconcile conflicting and overlapping information.

**Annotation Service:** Annotate materials stored anywhere within the Edutella Network.

### 4.4.3 Edutella query service

The Edutella infrastructure uses our metadata subset defined in section 4.1 as one standard to annotate learning resources (Other used standards include ADL

SCORM [3], IMS. etc.). Unlike the OLR infrastructure we still need a standardized query language to cope with the different solutions each peer may have found to structure and store its learning resources. The Edutella Query Service is a standardized query exchange mechanism for RDF metadata stored in distributed RDF repositories and serves both as query interface for individual RDF repositories located at single Edutella peers as well as query interface for distributed queries spanning multiple RDF repositories (storing RDF statements based on arbitrary RDFS schemata).

One of the main purposes is to abstract from various possible RDF storage layer query languages (e.g., SQL) and from different user level query languages (e.g., RQL, TRIPLE): The Edutella Query Exchange Language and the Edutella Common Data Model provide the syntax and semantics for an overall standard query interface across heterogeneous peer repositories for any kind of RDF metadata. The Edutella network uses the query exchange language family RDF-QEL (based on Datalog semantics and subsets thereof) as standardized query exchange language format which is transmitted in an RDF/XML-format.

We will start with a simple RDF knowledge base with the following RDF XML Serialization:

```
<lib:Book about='http://www.xyz.com/sw.html'>
    <dc:title>Software Engineering</dc:title>
</lib:Book>

<lib:Book about='http://www.xyz.com/ia.html'>
    <dc:title>Intelligent Agents</dc:title>
    <dc:subject
        rdf:resource='http://www.kbs.uni-hannover.de/Uli/ACM_CCS#I.2' />
</lib:Book>

<lib:Book about='http://www.xyz.com/ai.html'>
    <dc:title>Artificial Intelligence</dc:title>
</lib:Book>

<lib:AI-Book about='http://www.xyz.com/pl.html'>
    <dc:title>Prolog</dc:title>
</lib:AI-Book>
```

To simplify the query, we assume that the book on intelligent agents is annotated with the ACM CCS node I.2 ARTIFICIAL INTELLIGENCE. Otherwise, we could easily query for super topics of the entry by using the *lom_cls:taxon* element and the fact that the classification scheme is also part of the knowledge base. We will show you a small example of this handling of subtopics later.

Edutella peers can be highly heterogeneous in terms of the functionality (i.e., services) they offer. A simple peer has RDF storage capability only. The peer has

some kind of local storage for RDF triples (e.g., a relational database) as well as some kind of local query language (e.g., SQL). In addition the peer might offer more complex services such as annotation, mediation or mapping.

**Edutella Common Data Model (ECDM)**

The ECDM is based on Datalog, which is a well-known non-procedural query language based on Horn clauses without function symbols. A Datalog program can be expressed as a set of rules/implications (where each rule consists of one positive literal in the consequent of the rule (the head), and one or more negative literals in the antecedent of the rule (the body)), a set of facts (single positive literals) and the actual query literals (a rule without head, i.e., one or more negative literals). Literals are predicates expressions describing relations between any combination of variables and constants such as *title(http://www.xyz.com/book.html, 'Artificial Intelligence')*. Disjunction is expressed as a set of rules with identical head. Additionally, we can use negation as failure in the antecedent of a rule, with the semantics that such a literal cannot be proved from the knowledge base. A Datalog query then is a conjunction of query literals plus a possibly empty set of rules [82].

Datalog queries easily map to relations and relational query languages like relational algebra or SQL. In terms of relational algebra, Datalog is capable of expressing selection, union, join and projection and hence is a relationally complete query language. Additional features include transitive closure and other recursive definitions.

The example knowledge base in Datalog reads

```
title('http://www.xyz.com/ai.html','Artificial Intelligence').
type('http://www.xyz.com/ai.html',Book).
title('http://www.xyz.com/ia.html','Intelligent Agents')
subject('http://www.xyz.com/ia.html','http://www.kbs.uni-hannover.de/Uli/ACM_CCS#I.2').
type('http://www.xyz.com/ia.html,Book').
title('http://www.xyz.com/sw.html','Software Engineering').
type('http://www.xyz.com/sw.html',Book).
title('http://www.xyz.com/pl.html','Prolog').
type('http://www.xyz.com/pl.html',AI-Book).
```

Each RDF repository can be viewed as a set of ground assertions either using binary predicates as shown above, or as ternary statements "s(S,P,O)", if we include the predicate as an additional argument. In the following examples, we use the binary surface representation.

We will have a closer look on the evaluating of the following query (plain English)

> "Return all resources that are a book having the title "Artificial Intelligence" or have content, that is a subtopic of "Artificial Intelligence" or that are an AI book."

**Example Query in (binary) Datalog notation.**

```
aibook(X) :- title(X, 'Artificial Intelligence'), type(X, Book).
aibook(X) :- type(X, AI-Book).
aibook(X) :- subject(X,'http://www.kbs.uni-hannover.de/Uli/ACM_CCS#I.2').

?- aibook(X).
```

Since our query is a disjunction of three (purely conjunctive) subqueries, its Datalog representation is composed of three rules with identical heads. The literals in the rules' bodies directly reflect RDF statements with their subjects being the variable X and their objects being bound to constant values such as 'Artificial Intelligence'. Literals used in the head of rules denote derived predicates (not necessarily binary ones). The query expression "aibook(X)" asks for all bindings of X, which conform to the given Datalog rules and our knowledge base, with the results:

```
aibook('http://www.xyz.com/ai.html')
aibook('http://www.xyz.com/ia.html')
aibook('http://www.xyz.com/pl.html')
```

It is of course easy to extend the examples to subtopics of 'Artificial Intelligence'. We would then extend our query with the search for super topics, by defining the new concept 'content' and then replacing the last subquery with:

```
aibook(X):- content(X,"http://www.kbs.uni-hannover.de/Uli/ACM_CCS#I.2").

content(X,Y):- subject(X,Y).
content(X,Y):- subject(X,Z),
                    subtopic(Z,Y).
subtopic(Z,Y):- taxon(Y,Z).
subtopic(Z,Y):- taxon(Y,W),
                    taxon(W,Z).
```

For a complete over view of Edutella, we refer to [40]

## 4.5 Inferring Metadata

### 4.5.1 Problem description

The motivation in describing the courses with metadata was to achieve better retrieval results when searching for educational content and to allow more precise queries. Annotating complete courses, consisting of a set of related resources, it is obvious that several metadata fields are implicit for certain resources in that they can be easily derived from the fields of other resources. For example some metadata elements are simply inverse elements to other ones: The qualified relationship *dcterms:hasPart* between two resources implies the inverse relationship *dcterms:isPartOf* where RDF subject and object are interchanged (i.e. the directed RDF arc between both is reversed).With the help of a set of logical rules, which can be processed by an inference engine, all these implicit metadata elements or RDF statements can be created automatically from the existing ones and added to get complete annotations.

Another point to notice is that the specifications for the LOM data model and even of the LOM RDF binding are mainly on the syntactical level, but leave out important semantical information. What is needed here are axioms (we can use the inference rules mentioned above as integrity constraints) which provide a formal basis for a more precise description of the usage of all LOM elements. One example is the is again the *dcterms:PartOf* relationship between two resources.

In our definition this relationship describes the hierarchical structure in terms of course modules (see fig. 4.5). However, one could use an *dcterms:isPartOf* qualifier in terms of a temporal relationship (see fig. 4.6) The usage of this relationship changes the axioms for this element dramatically. Adding axioms is therefore an important means for creating a shared semantical basis for metadata elements usage and thus clarifying how to use the LOM metadata elements. By that the exchangeability of LOM metadata records between different applications is increased.

We start from a comprehensive though not complete description of learning resources by metadata elements. These metadata elements not only use elements to express keywords and creator information but also provide a specific model of our course in terms of (hierarchical) structure. By processing the axioms not only the metadata annotations can be completed by implicit metadata as described above. Additionally these axioms help checking the semantical consistency with regards to the intended interpretation.

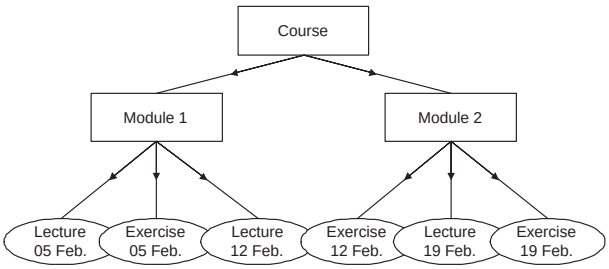



Figure 4.5: Hierarchical Structure of a Course Defined via *dcterms:hasPart*

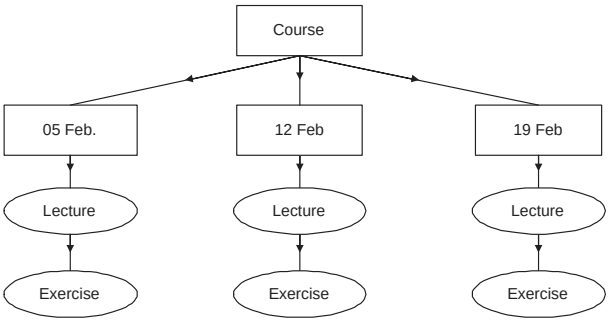



Figure 4.6: Temporal Structure of a Course Defined via *dcterms:hasPart*

### 4.5.2 Inference rules and Axioms for a Formal Description of LOM

The rules will be described using first-order logic annotation for simplicity, allthough the logic behind these rules is sometimes not pure first-order logic. Other languages like Prolog or TRIPLE can be used similarly. The inference rules were first published in [20]. A complete table of the LOM elements extended with their inference rules can be found in the Appendix.

#### Rules

In the following rules, R is used as an abbreviation for learning resources. The metadata elements from DCterms that define relations between resources as *dcterms:hasPart, dcterms:hasVersion,* etc. play an important role in the annotation, because most of our inference rules are especially useful when relations between learning resources are taken into account. In the following, element, element1, are being used as a placeholder since many of the rules work for different elements.

**Inverse Elements:** The most basic rule describes the fact that some elements have inverse elements. If there is a dcterms:hasPart relationship between two resources R1 and R2, than there has to be also a dcterms:isPartOf relationship between R2 and R1. Inverse elements are marked in the LOM table with **inverse(element$_1$, element$_2$)** The rule is defined in first-order logic as:

$$\begin{gathered}\forall R_1, R_2, element_1, element_2 : \\ (element_2(R_2, R_1) \wedge inverse(element_1, element_2)) \\ \Rightarrow element_1(R_1, R_2).\end{gathered}$$

$$\begin{gathered}\forall element_1, element_2 : inverse(element_1, element_2) \\ \Longleftrightarrow\ inverse(element_2, element_1).\end{gathered}$$

**Transitive Elements:** Transitivity occurs with the elements *dcterms:hasPart* and *dcterms:isPartOf*. If a resource R1 includes a part R2 and R2 in turn includes a part R3, then it can be inferred that R1 includes part R3. Transitive elements are marked in the LOM table with **transitive(element)**. The rule defined in first-order logic is:

$$\begin{gathered}\forall R_1, R_2, R_3, element : \\ element(R_1, R_2) \wedge element(R_2, R_3) \wedge transitive(element)\end{gathered}$$

$$\Rightarrow element(R_1, R_3).$$

**Inheritance:** Predicates can be inherited along certain elements. As the element *dcterms:hasPart* and *dcterms:isPartOf* are used to structure a course, a lot of predicates like *1.3 Language, 1.5 Keyword*, etc. can be inherited from a lecture unit to the whole lecture. Predicates that are inherited in such a way along a certain element are marked in our extended LOM table with **inheritance_along(element,predicate)**), the rule is expressed as:

$$\forall R_1, R_i, element, predicate, value :$$
$$element(R_1, R_i) \wedge predicate(R_i, value) \wedge inheritance\_along(element, predicate)$$
$$\Rightarrow predicate(R_1, value).$$

A special situation occurs for the predicate *7.1 Relation Kind* where the metadata instance *dcterms:requires* is used, to describe the background knowledge for a learning resource. The value of this predicate is only inherited along a *dcterms:hasPart*, for example, if the learning resource providing the background knowledge is not also connected via *dcterms:hasPart*. Predicates that are inherited in such a way along a certain element are marked in ou extended LOM table with **outwardInheritance_ along(element,predicate)**, the rule is expressed as:

$$\forall R_1, R_i, R_j (j \neq i), element, predicate :$$
$$element(R_1, R_i) \wedge predicate(R_i, R_j) \wedge \not element(R_1, R_j) \wedge$$
$$outwardInheritance\_along(element, predicate)$$
$$\Rightarrow predicate(R_1, R_j).$$

Some inverse relationships like *dcterms:hasFormat* and *dcterms:isFormatOf* are so strong that every predicate value from a resource is inherited to its related resources. Predicates that are inherited in such a way along a certain element are marked in the extended LOM table with
**inverseInheritance_along(A,P).**, the rule is expressed as:

$$\forall R_1, R_i, element_1, element_2, predicate, value:$$
$$(element_1(R_1, R_i) \vee element_2(R_1, R_i)) \wedge predicate(R_i, value_i) \wedge$$
$$inverse(element_1, element_2) \wedge$$
$$(inverseInheritance\_along(element_1, predicate) \vee$$
$$inverseInheritance\_along(element_2, predicate))$$
$$\Rightarrow predicate(R_1, value_i).$$

### Aggregation

The following rules leave the scope of first-order logic, and appear only in very special situations. Allthough they each appear only for one LOM element and no elements from our used subset, we present them here for reasons of completeness. Sometimes the value of a predicate is the sum of values of predicates from other resources. For example, if a resource is separated in different parts via *dcterms:hasPart*, the size of the resource as defined with the predicate *4.2 Size* is the sum of the parts' sizes. Predicates that are added in such a way along a certain element are marked in the extended LOM table with **summation_along(element,predicate).**, the rule is expressed as:

$$\forall R_1, R_i, element, predicate, value_i:$$
$$element(R_1, R_i) \wedge predicate(R_i, value_i) \wedge$$
$$sum\_along(element, predicate)$$
$$sum = \sum_i value_i$$
$$\Rightarrow predicate(R_1, sum).$$

For Boolean values, the summation corresponds to a Boolean OR. This is used for example, when a resource is divided into several parts and we want to determine whether the whole resource is copyrighted or not, based on the copyright annotations of its parts. In this case, we use *dcterms:hasPart* and infer the copyright status based on the values of the different parts (values "true" or none for the

predicate *6.2. Copyright and other restrictions*). Predicates that are aggregated in such a way along a certain element are marked in the extended LOM table with **booleanOR_ along(element,predicate).**, the rule is expressed as:

$$\begin{gathered}\forall R_1, R_i, element, predicate, value_i : \\ element(R_1, R_i) \wedge predicate(R_i, value_i) \wedge \\ booleanOR\_along(element, predicate) \\ sum = \bigvee\nolimits_i value_i \\ \Rightarrow predicate(R_1, sum).\end{gathered}$$

The "aggregation" level of a resource (see, e.g., predicate *1.8 Aggregation level*) is a value from 1 to 5. Since a collection of level 1 resources is defined by the LOM standard to have level 2 as value, the value of this predicate for a resource that has certain child resources identified via *dcterms:hasPart* can be defined as the maximum value of the child resources plus 1. Predicates that are aggregated in such a way along a certain element are marked in the extended LOM table with **maxSummation_ along(element,predicate).**, the rule is expressed as:

$$\begin{gathered}\forall R_1, R_i, element, predicate, value_i : \\ element(R_1, R_i) \wedge predicate(R_i, value_i) \wedge \\ maxSummation\_along(element, predicate) \\ sum = max\{value_i\} + 1 \\ \Rightarrow predicate(R_1, sum).\end{gathered}$$

### Inference Rules for Content Classification

In section 4.2 we have described how to annotate the content of a resource. If this semantic structure can also be accessed by an inference engine, we can formulate the following rule to infer that a resource that covers a topic also covers all subtopics.

$$\forall R, content_1, content_2 :$$
$$dc\_subject(R, content_1) \wedge lom\_cls\_taxon(content_1, content_2)$$
$$\Rightarrow dc_subject(R, content_2).$$

Our best-practice subset of 17 elements, extended by the rules we can use for this element can be seen in table 4.2. In the appendix you will find the complete table of LOM elements, expanded with their inference rules.

### 4.5.3 Inferring over Metadata with Prolog

Metadata can easily be transferred to Prolog. The Metadata-statement:

```
The author of a resource http://www.xyz.com is "Peter Smith"
```

Can be stored as

```
rdf(dc_creator,http://www.xyz.com, "Peter Smith")
```

(if we use the Dublin Core element for authors).

A inheritance along *dcterms:hasPart* inference rule for the element author (*dc:creator*) can than be written in Prolog as:

```
rdf(dc_creator,Resource2,Value):-
rdf(dc_creator,Resource1,Value),
rdf(dcterms_hasPart,Resource1,Resource2),
inheritance_along(dcterms_hasPart,dc_creator).
```

To use our inference rules, we wrote an inference machine in Prolog. The more complex ”Aggregation”-inference rules, were of course unable to define in Prolog, but as they don’t apply to our subset of LOM, we could do without. We developed an RDF-Prolog-Parser, based on Minerva, a ISO-13211-1 Prolog compiler and executive hosted in Java. Using this inference machine we were able to create new expanded RDF files for each course. Extended with the implicit information about the course material, querying this files enhanced our query results, when searching for example in the context of the Edutella project.

Table 4.2: The subset with inference rules

| **LOM category** | **Metadata name** | **used element** | **inference rules** |
|---|---|---|---|
| 1. General | 1.2 Title | **dc:title** | none |
| | 1.3 Language | **dc:language** | inheritance_along (**dcterms:hasPart**,Language). |
| | 1.4 Description | **dc:description** | inheritance_along (**dcterms:hasPart**,Description). |
| 2. Lifecycle | 2.3 Contribute | **dc:creator** with a **lom:entity** and the author in vCard format dcterms:created with the date in W3C format | inheritance_along (**dcterms:hasPart**,Entity). inheritance_along (**dcterms:hasPart**,Date). |
| 4.Technical | 4.1 Format | **dc:format** | inheritance_along (**dcterms:hasPart**,Format). |
| 5.Educational | 5.2 Learning Resource Type | **rdf:type** | inheritance_along (**dcterms:hasPart**,Type). |
| 6. Rights | 6.3 Description | **dc:rights** | inheritance_along (**dcterms:hasPart**,Entry). |
| 7. Relation | | **dcterms:hasFormat** **dcterms:isFormatOf** **dcterms:hasPart** **dcterms:isPartOf** **dcterms:hasVersion** **dcterms:isVersionOf** **dcterms:requires** **dcterms:isRequiredBy** | inverse(**dcterms:hasFormat**, **dcterms:isFormatOf**). inverse(**dcterms:hasPart**, **dcterms:isPartOf**). inverse(**dcterms:requires**, **dcterms:isRequiredBy**). inverse(**dcterms:hasVersion**, **dcterms:isVersionOf**). outwardInheritance_along (**dcterms:hasPart**, **dcterms:requires**). inverseInheritance_along (**dcterms:hasFormat**, **dcterms:requires**). OutwardInheritance_along (**dcterms:hasVersion**, **dcterms:requires**). transitive(**dcterms:hasPart**). transitive(**dcterms:isPartOf**). |
| 9. Classification | | **dc:subject** for content classification. This element links to an entry in a hierarchical ontology, that is an instance of **lom_cls:Taxonomy** (see next section) | inheritance_along (**dcterms:hasPart**,**Entry**). inverseInheritance_along (**dcterms:hasFormat**,**Entry**). inheritance_along (**dcterms:hasVersion**,**Entry**). |

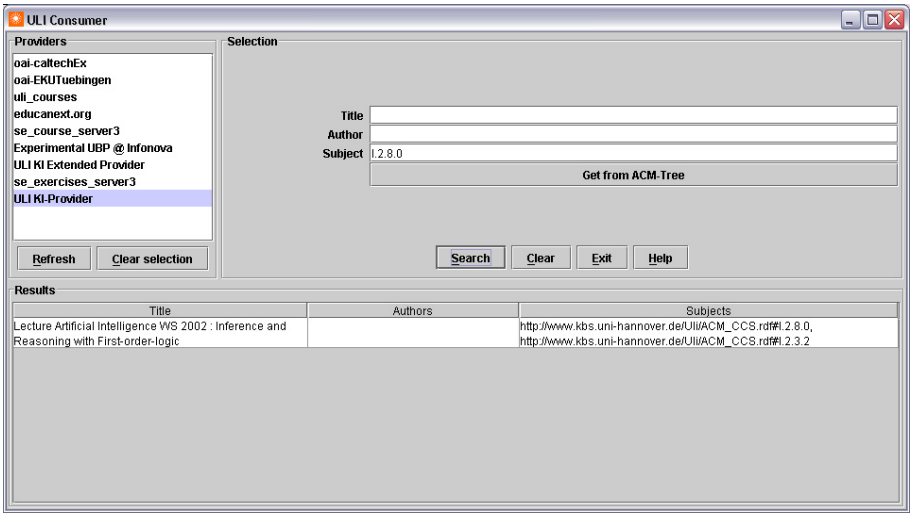



Figure 4.7: Query Results on Not-inferred Data

### 4.5.4 Enriched query results

Fig. 4.7 shows a Edutella query result set based on a course in artificial intelligence. The search in this example was for the metadata entry *dc:subject: I.2.8.0* (standing for *I. Computing Methodologies / ARTIFICIAL INTELLIGENCE / Problem Solving, Control Methods, and Search / Backtracking* in the ACM classification). The result is a single learning resource with is no author annotated because there is only one author information for the complete course.

Querying the RDF files that were extended using our inference rules, the file-based provider offers a different result set shown in fig. 4.8.

Not only the learning resource is a result, but also the unit and the course it belongs to, because the content information is inherited upward, following the fact that if a learning resource in a course has the subject ”Backtracking”, the complete course has the subject ”Backtracking”. Also the resource has inherited the author information from the course.

### 4.5.5 Inference rules in the semantic web tower

In the context of the semantic-web tower, as pictured in fig. 3.1, our infering of the metadata with PROLOG would be located in the the *logic layer*. It is however very unefficient to translate our RDF-descriptions to PROLOG and re-translate

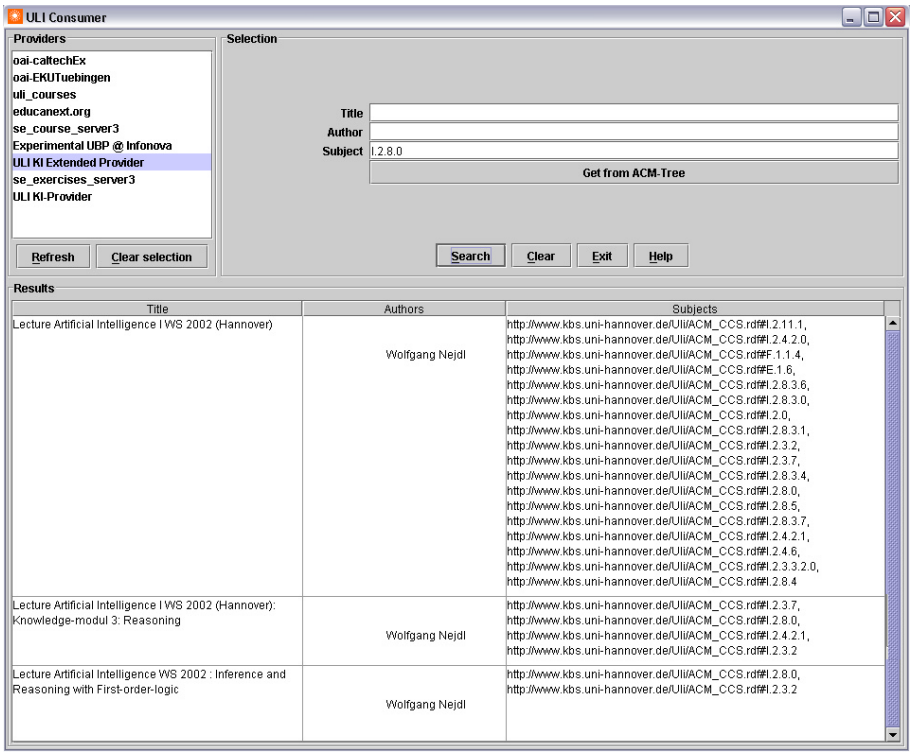



Figure 4.8: Query Results on Inferred Data

the infered metadata back to RDF, not to mention, that we completely lose our downwards-compatibility. In chapter 5 we will introduce an ontology language, that offers the basis for a more sophisticated *logic layer*.

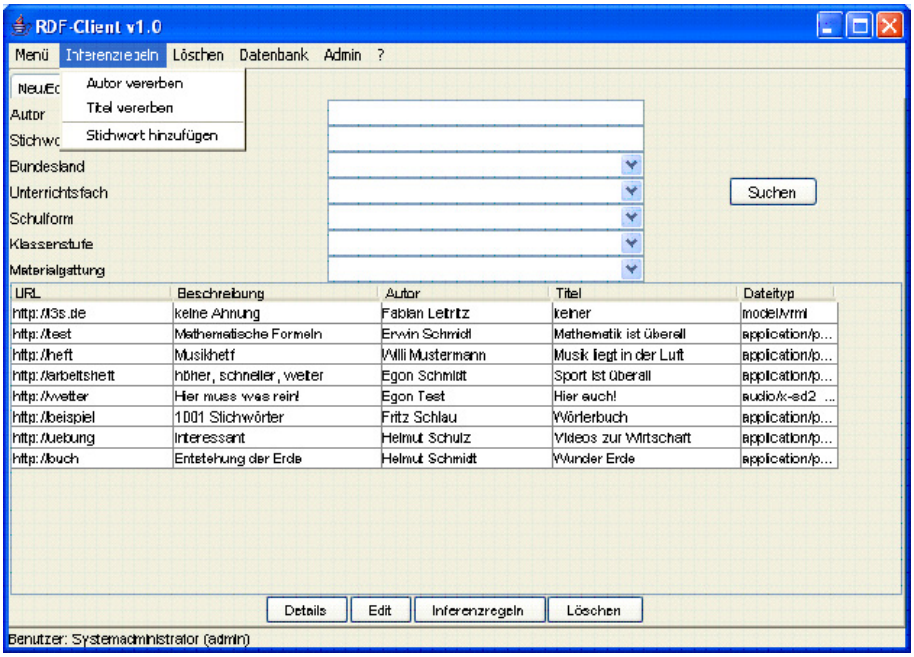



Figure 4.9: A LOM editor using inference rules

### 4.5.6 A LOM editor with inference rules

In the last years, our theoretical work concerning inference rules has become the basis for new and better metadata editors based on LOM. In the context of a software project at the university of Hannover, the SHAME editor, briefly introduced in chapter 3 has been extended with inference rules for user friendly annotation. The extended editor is currently used at Cornelssen, a German publisher, to annotate their offer of books and other resources. A screenshot can be seen in fig. 4.9.

In this editor the inference rules allow the user to inherit *author* and *title* information via the *hasPart*-relationship or to simple add keywords to all *isPartOf*-resources. The inference rules work directly with the RDF-models of the resources.

## 4.6 Conclusion

In our work with metadata for eLearning we have tried to follow the architecture of the Semantic Web as closely as possible. The usage of RDF for our LOM-subset has proven itself effective and simple. The usage of RDF to model our content classification hierarchies has worked well, too. If we would model these ontologies again however we would use new ontology languages that came up in the last years, and that we successfully used in other projects, as described for example in chapter 5.

Unfortunately we also never found the time to include all our different techniques and theories in one course metadata editor for P2P infrastructures.

I theory, such an editor would be able to create and edit RDF course descriptions following our LOM-subset discussed in section 4.1.

The values for the metadata elements would be inserted in a simple user interface as displayed in fig. 4.10. It would include a visual representation of the ACM-Classification tree to easily drag and drop content classification in the metadata description.

Finally the interface would need the user to only fill in the explicit information definitely needed for the course structure. All elements that could be inferred would be automatically included in the RDF-file when finishing your annotation. For example the user would only need to insert one *dc:creator* for the course, to let the system inherit this author for all parts, or the user would only have to express one direction of an inverse element-pair like *dcterms:isPartOf,decterms:hasPart*. Our first tests have shown that we would approximately need half the metadata elements filled in explicitly, where the other half could be inferred.

The editor was planned to be also part of any Edutella server to upload the course description directly into the P2P network.

Work on such an editor had begun in 2003 but was never finished because of lack of time and money.

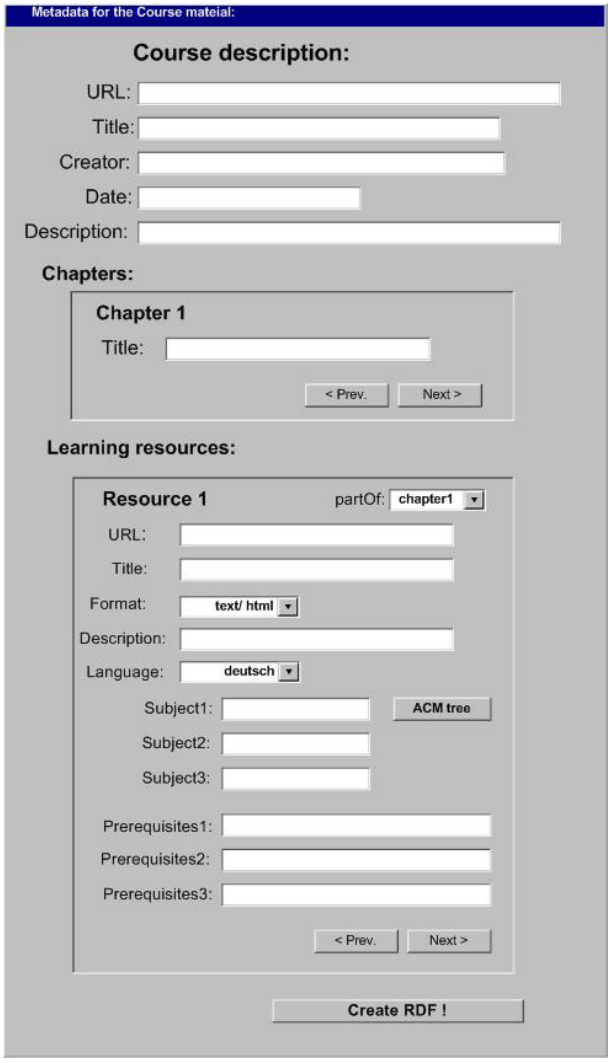



Figure 4.10: The user-interface design for an ideal course metadata editor

# Chapter 5

# Metadata in a context based environment

This chapter discusses the usage of metadata in a context based semantic web environment: Based on metadata a user will receive individual information on his handheld PC while visiting the research center L3S. Allthough the quality of metadata is more or less the same as in our eLearning-scenarios discussed in the last chapter, the type of query processing is completely difficult in this scenario. We have no longer only explicit, but now also implicit information provision.

## 5.1 Background

### 5.1.1 The semantic web ontology layer

As we have discussed in the last chapter, the expressiveness of RDF and RDFS is limited to model web objects and their relationships toward each other, as are defined in the *ontology vocabulary layer* of the semantic-web tower in fig. 3.1. An ideal ontology language should provide the following concepte:

- *Local validity or properties*
- *Disjunctive classes*
- *Boolean junctions of classes*
- *Cardinal restrictions*
- *Characteristics of properties* like transitivity, inversity, etc.

The *Web Ontology Language (OWL)* [88] is such an extension of RDFS to model Ontologies.

### 5.1.2 Web Ontology Language (OWL)

In 2001 research groups from the United States and Europe had already identified the need for a more powerful ontology modelling language. This led to the definition of DAML+OIL (see [28]) (the name being a join of the American language proposal DAML-ONT and the European language OIL). DAML+OIL was taken as the foundation for the *W3C Ontology Working Group* in defining OWL. OWL provides the concepts defined above.

As powerful expressiveness and effective reasoning are often contrary to each other, the W3C has defined three sublanguages of OWL (see [89]):

**OWL Full** OWL Full has the maximum expressiveness, allowing all language constructs of OWL and the syntactical liberties of RDF. Because of the complexity of OWL full Ontologies modelled with these language might include facts that are not determinable. Therefore reasoning possibilities in OWL Full Ontologies are limited.

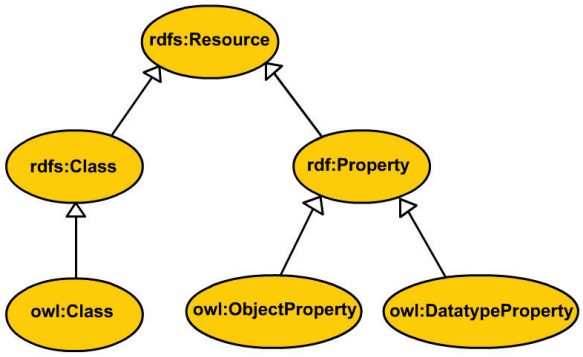



Figure 5.1: Subclass relationships between core concepts of OWL, RDF and RDFS

**OWL DL** OWL Description Logic is a part of OWL Full. As a *Description logic* (DL) it is fully determinable. DLs are logic-based knowledge representation formalism for modelling a domain in terms of concepts, roles and individuals, they are subsets of first-order logic (see [8]). OWL DL uses all constructs of OWL Full but defines restrictions for some constructs. A class for example may be subclass of different classes, but may be no instances of a class. This follows the DL principle of separating the terminology and instance description. OWL DL is not fully compatible with RDF.

**OWL Lite** OWL Lite is a further limitation of OWL DL, restricting all constructs that have cardinal restrictions.

The following rules describe the compatibility (see [89]):

- *Every legal OWL Lite ontology is a legal OWL DL ontology.*
- *Every legal OWL DL ontology is a legal OWL Full ontology.*
- *Every valid OWL Lite conclusion is a valid OWL DL conclusion.*
- *Every valid OWL DL conclusion is a valid OWL Full conclusion.*

The principle of the downward compatibility is only kept in OWL Full. Fig. 5.1 illustrates the subclass relationships between core concepts of OWL, RDF and RDFS.

We will discuss examples of OWL Ontologies in section 5.3.

### 5.1.3 Cooltown

The *Cooltown Program* [53], a program of HP Labs, acts in accordance to the definition of the Semantic Web as aforementioned. For several years, HP Labs has been working at the intersection of nomadicity, appliances, networking, and the web. This, as labelled on their homepage, "vision of the future" is cooltown - a vision of a technology future where people, places, and things are first class citizens of the connected world, wired and wireless - a place where e-services meet the physical world, where humans are mobile, devices and services are federated and context-aware. Many projects are a part of Cooltown all based on the *Cooltown Idea*, which can be summarized as:

1. People in Cooltown are mobile
2. Devices and services are federated and context-aware
3. Everything - people, places and things - has a web presence
4. Web-based appliances and e-services give the people what they need when and where they need it for work, play, life.

An example Cooltown project is a scenario, in which visitors of *Lasar Segall museum* in Sao Paolo, Brazil, are equipped with mobile devices (see [36]).
We realized a similar scenario:

### 5.1.4 Scenario

Visitors of the *L3S Research Center* [64] are enabled to make a self-guided tour by equipping them with a Pocket PC, where they can at the beginning of the tour choose their research interests to create their personal user profile. On their tour they are provided with context-aware information about researcher projects and knowledge about the respective domain. The information depends on the visitor's position and on his personal interests, based on their user profile. A fully functional prototype of the system has been implemented during a bachelor thesis in 2004. In section 5.2 we will briefly introduce the technical background of this scenario, while section 5.3 offers a description of the metadata used in this context. We will finally explain the functionality of the system by a example in section 5.4.

### 5.1.5 Related projects

#### Explore

Explore [45] is a project of the German *Fraunhofer Institutes*[45]. Goal of the project is to develop a mobile, interactive and context-sensitive system for games and guided tours in museums and expeditions, especially for children. The users receive information, small lectures our questions on their mobile phones, based on their location.

#### LISTEN

*LISTEN* [46] is another museum project by the *Fraunhofer Institute*, in which the visitor is only equipped with a special headphone. Moving through the expedition, a unique collection of sounds, spoken text and music is created. While standing in front of an artwork the user receives detailed background information about the piece.

#### Weltkulturerbe Völklinger Hütte

The company *eyeled*[44] has already realized a number of visitor information systems. The world heritage *Völklinger Hütte*[96], a mining museum, is just one example. Visitor receive information on a pocket PC, based on their location inside the open air museum.

#### SmartKom

SmartKom [33] is a project of the *German research centre for artificial intelligence* (DFKI) in Kaiserslautern and ten other partners. The goal of SmartKom is the development of a self-explaining, user-adaptive interface for the interaction of humans and technology in dialog. The SmartKom system coordinates 14 different applications with over 50 different functionalities., including a car navigation system, television or information services. SmartKom is independend from the application and offers an identical user interface. The SmartKom system uses Ontologies to enable a context-aware collaboration of the different services.

All these related projects only make small use of semantic web technologies and metadata, unlike our scenario, which offers a unique combination of semantic

web technologies with user adaptivity and precise user localization.

## 5.2 Architecture

A general structure of the context-aware information providing system is shown in fig. 5.2.

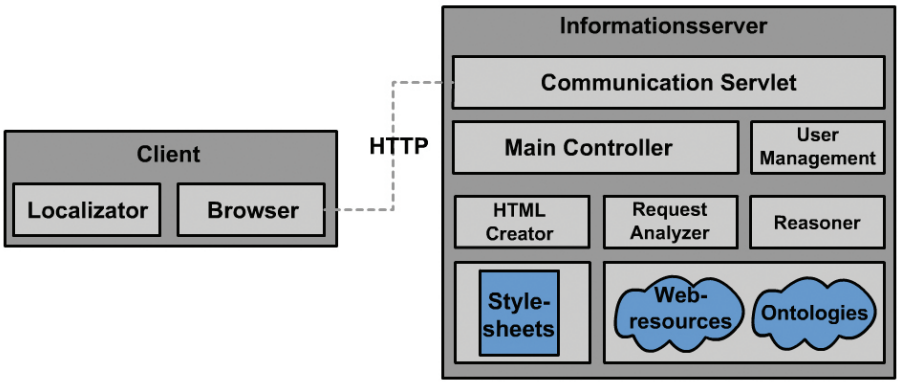



Figure 5.2: Architecture of the context-aware information providing system

The system is divided in a server- and a client-architecture, whereas the client-application is installed on the Pocket-PC and the information server is running on a usual web server. Programs are communicating using the *Hypertext Transport protocol* (HTTP). The client is the active part in the communication, requesting the information server to send data (*pull* technology). The client-application has two components: A *localizator*, constantly computing the visitors position and a *browser* displaying web pages and enabling the visitor to navigate through these pages. A screenshot of the client-application ca be seen in fig. 5.3.

The main components of the server-application are the *main controller*- controlling the complete program and the *communication servlet*- the interface to the client-application. The main controller uses these four modules:

- *User management*: Managing the user profiles as well as logging all activities of the current user.
- *Reasoner*: This module offers various tasks to find suitable information, depending on the user profile and on the information available via Ontologies

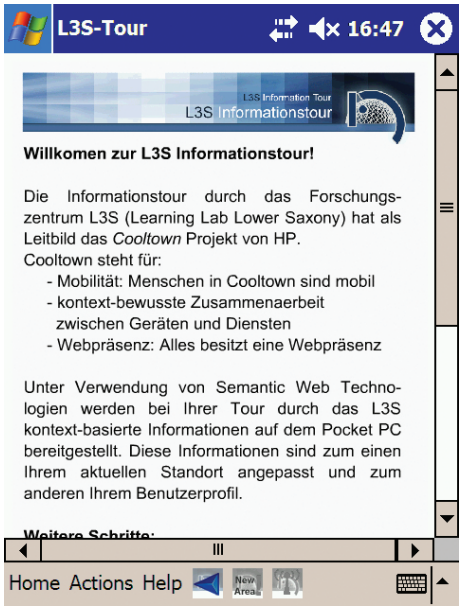



Figure 5.3: A screenshot of the client-application

or web resources.

- *Request analyzer*: Analysing the user's request and defining the tasks for the main controller.
- *HTML creator*: The suitable information identifies by the Reasoner is transformed into HTML documents using prefabricated stylesheets.

We will not go into much technical details of the system here, the next section will only analyse the usage of metadata in the system.

## 5.3 Application metadata

### 5.3.1 Localizator

The actual position is computed using the signal strength of the *access points* in the research center. In the upper part of fig. 5.4 you can see the position of these access points. The red circles indicate furthermore the intensity of the signals.

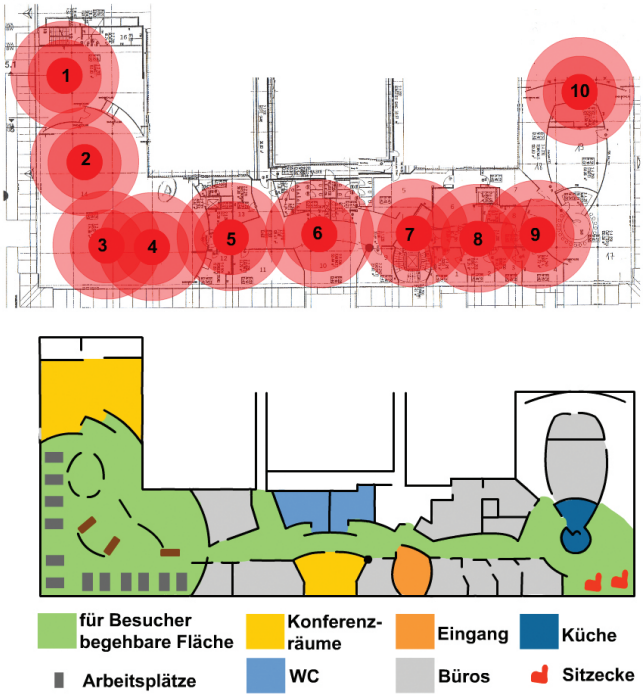



Figure 5.4: Map of the research center L3S

To compute the position, one has to initialize the system with reference measures first. At defined places the signal strengths are measured and protocolled in a XML model, representing the environment:

```
<environment name="l3s">
 ...
 <area name="Name des Referenzpunktes"
        url="http://server/info.html?action=new-area&amp;area=areaID">
    <accesspoint mac="d9-7a" max="-70" min="-85" avg="-78"/>
    <accesspoint mac="1b-7c" max="-60" min="-75" avg="-68"/>
```

```
    <accesspoint mac="73-2"  max="-80" min="-90" avg="-83"/>
    <accesspoint mac="7f-20" max="-55" min="-68" avg="-62"/>
    <accesspoint mac="83-80" max="-58" min="-74" avg="-64"/>
    <accesspoint mac="63-d0" max="-60" min="-75" avg="-67"/>
 </area>
 ...
</environment>
```

The structure of this XML model is defined via the following DTD:

```
<!ELEMENT environment (area+)>
<!ATTLIST environment name NMTOKEN #REQUIRED>

 <!ELEMENT area (accesspoint+)>
 <!ATTLIST area
       name NMTOKEN #REQUIRED
       url  ID #REQUIRED>

 <!ELEMENT accesspoint EMPTY>
 <!ATTLIST accesspoint
       mac NMTOKEN #REQUIRED
       avg CDATA #REQUIRED
       max CDATA #REQUIRED
       min CDATA #REQUIRED>
```

An *environment* therefore consists of multiple *areas*. Each area has an identifier (an URL), and for each area we have a set of reference measures related to *access points*. The access points are identified via the last digits of their MAC-address. The signal strength is stored with the maximum, minimum and average value. This XML document is loaded from the information server when starting the client application.

On a regular basis the localizator is scanning the signal strength of the MAC-addresses to identify the present location. This is done with a simple classification algorithm: *The nearest neighbour*. This algorithm is choosing the area which has the greatest similarity to the actual data. The similarity is defined with the *Euclidean distance*:

$$f(a,b) = \sqrt{\sum_{i=1}^{n}(|a_i| - |b_i|)^2}$$

$a$ and $b$ are vectors, containing the signal strengths of the access points, where:

$$\begin{aligned} a_i &= SignalStrength(AP_i, position(a), t(a)) \\ &und \\ b_i &= SignalStrength(AP_i, position(b), t(b)) \end{aligned}$$

Therefore $a_i$ is the strength of the access points $AP_i$, which is measured in the area $a$ at a certain time $t(a)$ (accordingly for $b_i$).

Used for position localization via wireless LAN this algorithm is often called *nearest neighbour in signal space* (see [9] and [79])

### 5.3.2 Information server

The information server differentiates between web resources and Ontologies, where Web resources are XML documents, including information and metadata.

#### Web resources

During the tour of the L3S, information is provided about people, projects and other things. Most of this information is included in web resources, XML documents that are transformed into a web page. The basic structure of such a web resource is the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<webpage title="ELAN"
    logo="http://www.l3s.de/data/webresources/elan.jpg"
    logoWidth="400" logoHeight="80">
   <!-- Metadata of the page-->
   <metadata>
     <id>elan</id>
     <filename>elan.xml</filename>
     <hasAuthor>
       http://www.l3s.de/ontologies/ResearcherOntology.owl#abelFabian
     </hasAuthor>
     <lastModified>10.08.2004 - 11:14</lastModified>
     <typeOfSite>project</typeOfSite>
     <hasChild>elan_services1</hasChild>
     <hasChild>elan_services2</hasChild>
     <possibleStylesheet>standard.xsl</possibleStylesheet>
     <keyword>E_Learning</keyword>
     <keyword>Lehre</keyword>
   </metadata>

   <!--Content of the page-->
   <home title="Welcome to ELAN" menuItem="welcome" typeOfPage="home">
     ELAN, the eLearning Academic ...
   </home>
   <details title="Ziele 2007" menuItem="details" typeOfPage="details" number="1">
     We want to integrate ...
   </details>
   <details title="ELAN-Netzpilot" menuItem="details" typeOfPage="details" number="2">
     The netpilot Hannover/Braunschweig ...
   </details>
</webpage>
```

The structure of the metadata section is defined in a Stylesheet. We will have a closer look on the properties *keyword* and *hasAuthor*, because they map to built-in Ontologies.

### 5.3.3 ACM CCS ontology

The *keyword* entry maps to the ACM CCS ontology. The ACM CCS ontology is based on the ACM CCS classification scheme as described in section 4.2. Every subject from the scheme is now a class (*owl:class*). The structure of the discipline is defined with *rdfs:subClassOf* instead of *lom_ cls:taxon*. This classification scheme is also used to define the visitors research interests in the user profile, therefore each class also has the Boolean property *asInterestSelectable*. Only subjects with a true value can be selected by the visitor in the beginning of the tour (see fig. 5.5).

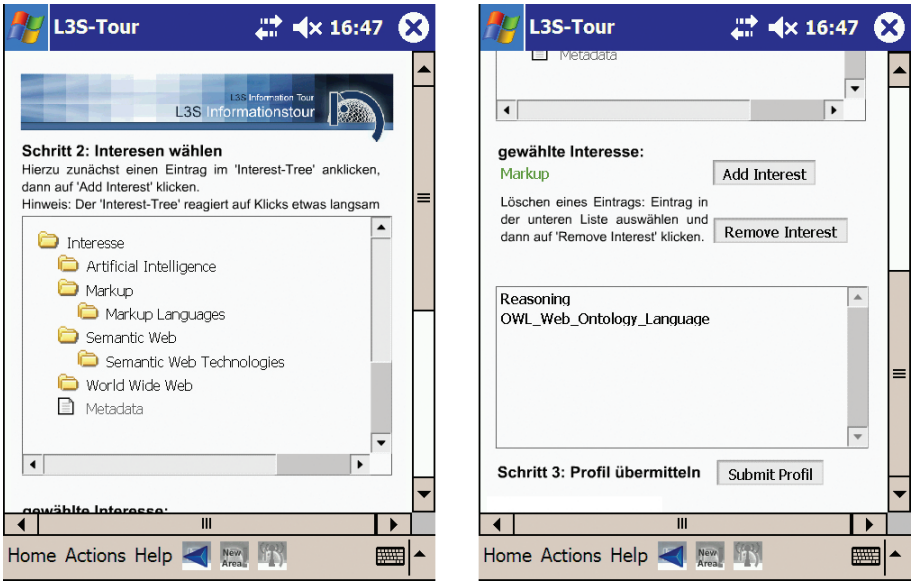



Figure 5.5: Definition or research interests, based on the ACM CCS

### 5.3.4 Researcher ontology

The *hasAuthor* entry maps to a resource in the *researcher-ontology*. The *researcher ontology* is based on an ontology developed in the context of the project

*OntoWeb* (see [73]). Its class hierarchy can be seen in fig. 5.6.

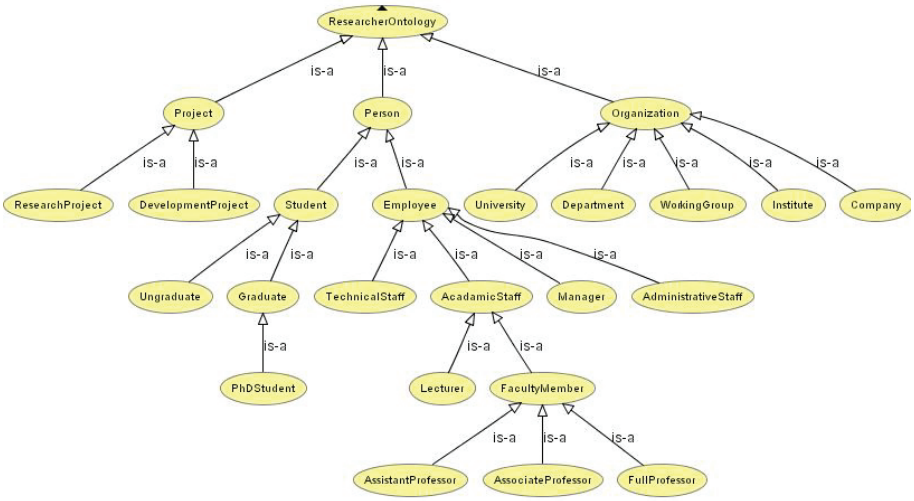



Figure 5.6: The researcher ontology

The main properties are:

**DatatypeProperties** All classes have the properties *name* and *webpage*. Organisations also have the properties *country* and *short name*. For people more properties like *e-mail*, *phoneNumber* and *currentEmployment* are defined. The property *pictureURL* allows linking pictures of the respective person.

**ObjectProperties** There are two major object properties: *involvedIn* and *hasMember*, which are inverse to each other and have specializing subproperties.

The properties *involvedIn* and *hasMember* are defined as follows:

```
<owl:ObjectProperty rdf:ID="involvedIn">
   <rdfs:domain>
     <owl:Class>
       <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#Person" />
         <owl:Class rdf:about="#Organization" />
       </owl:unionOf>
     </owl:Class>
   </rdfs:domain>
   <rdfs:range>
     <owl:Class>
```

```
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Organization" />
        <owl:Class rdf:about="#Project" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#hasMember" />
  </owl:inverseOf>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="hasMember">
  <rdfs:domain>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Project" />
        <owl:Class rdf:about="#Organization" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:domain>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Person" />
        <owl:Class rdf:about="#Organization" />
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#involvedIn" />
  </owl:inverseOf>
</owl:ObjectProperty>
```

The researcher ontology at the moment includes 150 resources.

### 5.3.5 Environment ontology

To enable the application to display context based information based an the users position, the *environment ontology* specifies what can be found in which area. Fig. 5.7 illustrates the classes and properties of this ontology: Every *environment* consists of (*hasArea*) different *areas*. To every environment (or area) *persons* are related to (*hasInhabitant*). The property *hasInhabitant* maps to resources in the researcher ontology, enabling relationships to projects, people are involved in. The property *explicitLink* allows relations to web resources.

### 5.3.6 Reasoning

The Reasoner offers functionalities that are needed to extract and analyse the information from the web resources and Ontologies. Three tasks can be identified:

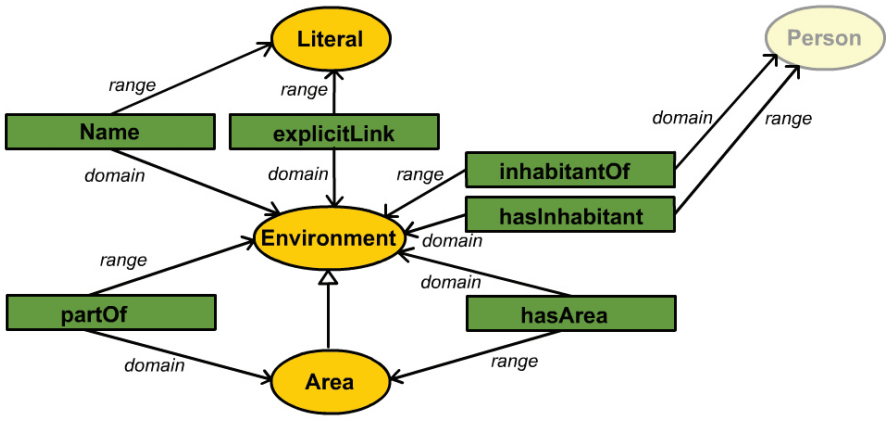



Figure 5.7: The environment ontology

Extracting information from web resources, extract and infer information from Ontologies and matching interests from different resources. Based on the information provided in the web resources and Ontologies, these tasks were trivial to implement. A complete description of the full system and all of its functionalities can be found in [1].

## 5.4 Example

The following example shall help for a better understanding of the functionality of the system:
Tim, currently finishing his bachelor-thesis, is visiting the research center L3S. He receives a Pocket PC and is asked to define his research interests from a drop down menu, based on the ACM CCS (see fig. 5.5). Tim chooses ”Reasoning” While walking through the L3S, he constantly receives information about people working close to his positions and the projects they work in. These projects are always ordered according to his research interests, placing the project with the most background in ”Reasoning” on top position (see fig. 5.8).

If Tim chooses to follow a hyperlink for one of the projects, a webpage will be displayed. The webpage will not be the standard project homepage, but a combination of web resources (see section 5.3.2), displaying Tim only the aspects of a project that match with his research interests (see fig. 5.9).


Figure 5.8: Ranked information about projects

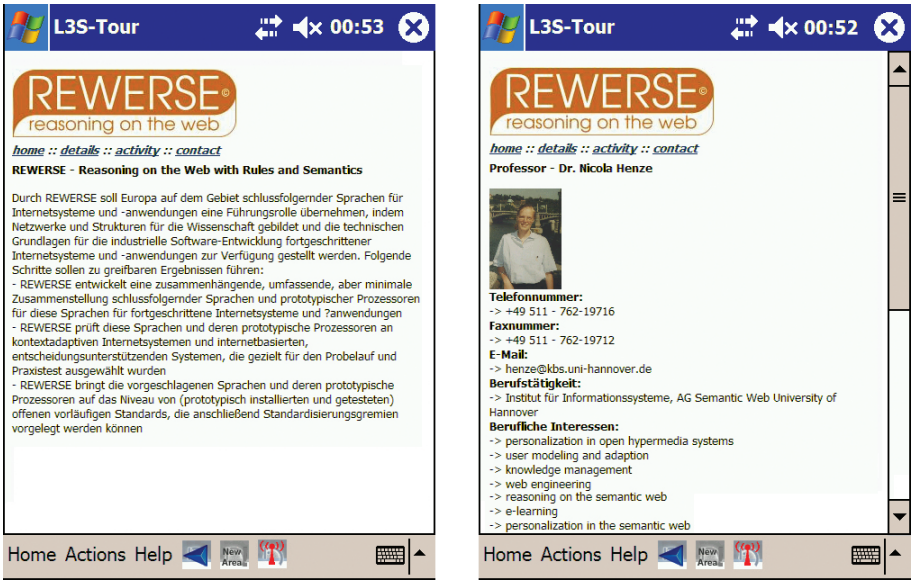



Figure 5.9: Displayed web pages - a combination of web resources

# Chapter 6

# Metadata for scientific primary data

In this chapter we will introduce a current project that uses metadata for the first time to register scientific primary data.

## 6.1 Problem description

In principle, scientists are prepared to provide data, but for the time being it is unusual to appreciate the necessary extra work for processing, context documentation and quality assurance. The classical mode of distributing scientific results is their publication in professional journals. These articles in journals are recorded in the ”citation index”. The index is used for a performance evaluation of scientists. Data publications have not been taken into account until now.

Project data is widely spread among research institutes and is collected and governed by scientists. Due to the lack of acknowledgement of this extra work, project data is often poorly documented, therefore badly accessible and not maintainable over long time periods. Large amounts of data are unused as they are only known and accessible to a small group of scientists.

Lately discussion about falsification in scientific results, resulted in the introduction of new rules of good scientific practice in the German scientific institutions. The rules also include guidelines for data access. Primary data of a publication has to be stored and made accessible for at least 10 years to allow a verification of the results.

In existing scientific journals, there is no room for repeating data work, like use of existing methods to complete a data basis and by this making it usable for later scientific applications. Repeating data work is no original scientific effort but is necessary as support of science.

## 6.2 The project ”Publication and Citation of Scientific Primary Data”

### 6.2.1 Background

On an initiative from a working group from the *Committee on Data for Science and Technology* (coData) [26], the *German research foundation* (DFG) [31] has started the project *Publication and Citation of Scientific Primary Data* as part of the program *Information-infrastructure of network-based scientific-cooperation and digital publication* in 2004. Starting with the field of earth science the *German national library of science and technology* (TIB) is established as a registration agency for scientific primary data. The data is still stored at the local research institutions, where the responsibility for valuating and maintaining of the data still lies. The project was first introduced in [21]

### 6.2.2 Describing, Citing and searching for primary data

In addition to the local data preparation the research institutions transmit the URL where the data can be accessed to the TIB, together with a XML-file containing all relevant metadata.

The TIB is saving this information about the primary data and awards the primary data with a *Digital object identifier* (DOI) as unique identifier for registration (See section 6.3.1 for details). Any scientist working with this data is now able to cite the data in his work by its DOI. By this, scientific primary data is not exclusively understood as part of a scientific publication, but has its own identity. All information about the data is now accessible through the online library catalogue of the TIB. The entry is displayed with all relevant metadata and persistent identifiers as links to access the dataset itself (see fig. 6.1).

If a scientist reads a publication where the registered data is used, he might be interested in analysing the data under different aspects. After gaining permission to do so by the research institution maintaining the data, he can cite the data in his own publications using its DOI, referring to the uniqueness and own identity of the original data.

If furthermore a scientist is interested in certain data, he can use the online library catalogue of the TIB to search for scientific primary data. A metadata search might result in a certain data set the scientist might want to use for his own publications. Resolving the DOI gives him access to the data, to find it sufficient

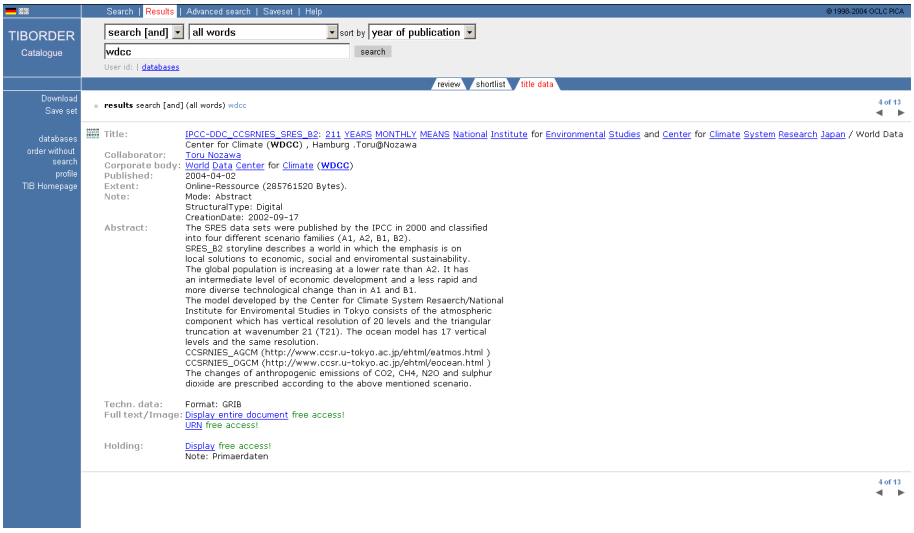



Figure 6.1: A dataset as a query result in the library catalogue

or not. The metadata also reveals the copyright holders of the data. Gaining permission to use this data by the research institution maintaining the data, he can also cite the data in his own publications using its DOI.

## 6.3 Technical aspects

### 6.3.1 Identifiers

**DOI** To register the data, the TIB awards it with a DOI as a unique identifier. *Digital Object Identifier* (DOI) is a system for identifying content objects in the digital environment. DOIs are names assigned to any entity for use on digital networks. They are used to provide current information, including where they (or information about them) can be found on the Internet. Information about a digital object may change over time, including where to find it, but its DOI will not change.

The DOI system provides a framework for persistent identification, managing intellectual content, managing metadata, linking customers with content suppliers, facilitating electronic commerce, and enabling automated management of media. DOIs can be used for any form of management of any data, whether commercial or non-commercial.

The system is managed by the *International DOI foundation* (IDF), an open membership consortium including both commercial and non-commercial partners, and has recently been accepted for standardisation within ISO. Several million DOIs have been assigned by DOI Registration Agencies in the US, Australasia, and Europe.

Using DOIs as identifiers makes managing intellectual property in a networked environment much easier and more convenient, and allows the construction of automated services and transactions. For more information, we refer to [35].

The TIB has become a member of the IDF in 2003 and serves as the official Registration agency for scientific primary data. A DOI consists of two parts: a prefix and a suffix. For scientific primary data a DOI looks like this:

```
10.1594/WDCC/EH4_OPYC_SRES_A2
```

*10.1594*(Prefix) stands for the TIB as the registration agency who awarded this DOI. *WDCC* stands for the respective research institution. In our example the *World Data Center for Climate* (WDCC) and the rest is the internal name of the Data at the research institution.

This DOI can be resolved (and the data can be cited) in every web browser worldwide using the *Handle system* from the *Cooperation for National Research Initiatives (CNRI)*. The Handle system is a free java based comprehensive system for assigning, managing, and resolving persistent identifiers, known as "handles," for digital objects and other resources on the Internet (for more information see [52]).

**URN** Another common identifier in the publication world is the *URN (Uniform resource name)*. URNs are intended to serve as persistent, location-independent, resource identifiers and are designed to make it easy to map other namespaces (which share the properties of URNs) into URN-space. Contrary to the DOI system, URNs are not central supervised, although the *Internet Engineering Task Force* (IETF) ([60]) is responsibly for the assignment of URN namespaces. We will not go into much detail about the differences between URN and DOI, we refer to [77] for this discussion. Although we have decided to use DOIs for our registration, every registered dataset is also awarded by a unique URN. The TIB has registered the URN namespace *URN:tib* at the IETF. For best interoperability each awarded URN follows our DOI structure. For our example above the URN would look like:

```
URN:tib:10.1594/WDCC/EH4_OPYC_SRES_A2
```

To resolve this URNs the TIB has started a cooperation with the *German Library (DDB)* (see [30]) in Frankfurt. In the project *Epicur* (see [43]) the DDB has started to register online dissertations with unique URNs. The DDB will also register our URNs for scientific primary data and provide resolving of the URNs through the DDB infrastructure. There is however no metadata connected with this URNs.

### 6.3.2 Metadata schema

The main reason we have decided to use DOIs for our registration is the possibility to create so called *DOI Application profiles (AP)*. APs, are abstractions used to group DOIs into sets in which all DOIs of the given set, or AP, share a metadata schema. We therefore designed a set of metadata elements to describe our scientific primary data. Whenever possible, we have tried to use Dublin Core (DC) (see [37]) equivalent metadata elements.

The metadata scheme can be found in table 6.1

Table 6.1: Metadata for scientific primary data

| **Attribute** | **DC-mapping** |
|---|---|
| 1. DOI | none |
| 2. identifier | *dc:identifier* |
| 3. creator | *dc:creator* |
| 4. publisher | *dc:publisher* |
| 5. title | *dc:title* |
| 6. language | *dc:language* |
| 7. StructuralType | none |
| 8. mode | none |
| 9. resourceType | none |
| 10. registrationAgency | none |
| 11. issueDate | none |
| 12. issueNumber | none |
| 13. creationDate | *dcterms:created* |
| 14. publicationDate | *dc:date* |
| 15. description | *dc:description* |
| 16. publicationPlace | none |
| 17. size | *dcterms:extend* |
| 17.1 value | |
| 17.2 unit | |
| 18. format | *dc:format* |
| 19. edition | none |
| 20. relatedDOIs | *dc:source* (and others) |
| 20.1 relatedDOI | |
| 20.2 relationType | |

The elements are defined as follows:

1. **DOI** A DOI that identifies a resource.

2. **identifier** Any alphanumeric string which is unique within its domain of issue: For example, an ID from a legacy scheme or from the internal database of the resource's publisher. An Identifier is to be declared if one exists: some resources may have no identifier other than a DOI.

3. **creator** The main researchers involved working on the data, or the author s of the publication in order.

4. **publisher** The institution which submitted the work

5. **title** A name or title by which a resource is known.

6. **language** Primary language of the resource, if not English

7. **structuralType** The primary structural type of a resource. Fixed value: *Digital*

8. **mode** The principal sensory mode(s) in which a resource is intended to be perceived. Fixed value: *abstract*.

9. **resourceType** The general type of a resource. Fixed value: *dataset*.

10. **registrationAgency** The DOI Registration Agency responsible for issuing the metadata description. Fixed value: **10.1594** (TIB).

11. **issueDate** The Date on which the metadata description was made.

12. **issueNumber** The sequence number of this Declaration in the series metadata descriptions for this DOI.

13. **creationDate** Principal Date the work was created. E.g. the end of the measure for data or the finishing of the writing for publications.

14. **publicationDate** Date the work was released for publication.

15. **description** All additional information that does not fit in any of the other categories.

16. **publicationPlace** Place the resource has been published.

**17. size** Size of the resource, divided into:

- **17.1 value** number
- **17.1 unit** amount of data, number of pages, byte size etc..

**18. format** Technical format of the resource (MIME-Type).

**19. edition** Edition number of the resource (If the primary data set has changed the edition number increases).

**20. relatedDOIs** divided into:

- **20.1 relatedDOI** The DOI of the related resource
- **20.1 relationType** vocabulary: If the resource is a dataset, the relation to the work that uses this data should be: *isCitedBy*, If the resource is a new edition of an old dataset: *isNewVersionOf*, etc..

The elements *3,4,5,6,14,16,17,19* are obligatory for the citing of electronic media (ISO 690-2), the elements *7-12* give technical information and are required from the DOI metadata Kernel. Some of them have in our case default values, given in bold fonts.

This metadata set represents the smallest agreable subset between the interests of the TIB, the IDF and three different research institutes with three different own cataloguing schemes. Therefore it provides some basic information about the data, sufficient for citing, but not sufficient for comfortable metadata queries via the library catalogue, as all information about the content of the data and its technical format is only included in fulltext in the description. We are currently working on extending the set for this issues, based on initiatives like *Learning Objects Metadata (LOM)* by the *LTSC/IEEE* (see [66]) or the *CLRC Scientific Metadata Model* (see [25]).

Due to the expected huge amount of datasets that need to be registered, we have decided to distinguish between *citable datasets* on the collection level and *core datasets* on the item level. Core datasets receive their identifiers, but no metadata, as they are not included in the library catalogue. Only citable datasets, usually collections of, or publications from core dataset will be included in the catalogue.

## 6.4 Technical realization

To register the primary data, we have installed a web interface at the TIB. This interface receives XML files from the data providers, and starts the registration process:

- The DOI is registered via a java based transmission to the DOI foundation.
- For the URN registration a XML file has to be send by mail to the DDB.
- The metadata has to be transformed to PICA (see section 2.2.1) format and uploaded on a ftp server at the central library database.

This relationship is also displayed in fig. 6.2.

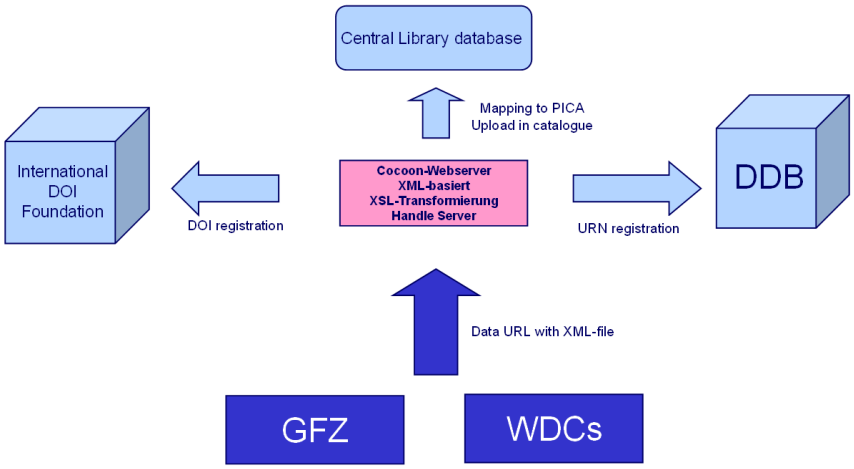



Figure 6.2: The architecture of the registration process

We have identified four different methods the system has to execute:

1. **CitationDOI** - For a citable dataset a DOI and an URN are registered
2. **DataDOI** - A core dataset only receives DOI and URN
3. **URLupdate** - If the URL of a dataset changes, this information has to be stored at the DDB for the URN and the IDF for the DOI resolution

4. **MetadataUpdate** - If any part of metadata changes for a citable dataset, a new PICA file has to be created.

To execute these different tasks, based on a single XML file, we have based the system on Apache Cocoon (see [7])

### 6.4.1 Cocoon

Cocoon is an XML publishing framework, it was founded in 1999 as an open source project under Apache Software Foundation. Cocoons offers the separation of content, style, logic and management functions in an XML content based web site (see fig. 6.3).

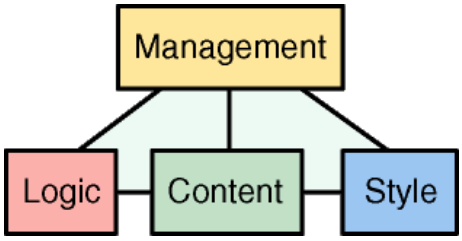



Figure 6.3: Cocoon: separation of content, style, logic and management functions

This separation allows us to easily change the parts of the architecture or the appearance of the system. Since it is initialised by the retrieval of a XML-file, send to the system by the research institutes, every registration starts a XML based pipeline process (see fig. 6.4).

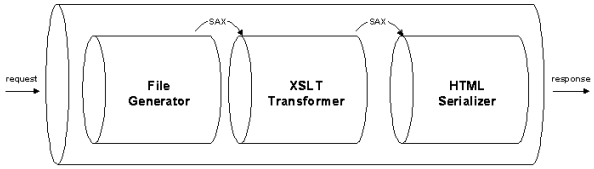



Figure 6.4: Cocoon: pipeline processing

All transaction is based on XML and XSLT files.

### 6.4.2 XSLT

The *eXtensible Stylesheet Language Transformation* (XSLT) is a language for transforming XML documents into other XML documents. The origins of XSL are in *Cascading Style Sheets* (CSS), where a ”stylesheet” is used to add formatting to an HTML file. The syntax to use a stylesheet in XSLT is similar to the syntax in CSS.

XSLT stylesheets have a very different function than CSS stylesheets, however. CSS allows you to define the colours, backgrounds, and font-types for an HTML web page. XSLT allows you to transform an XML file into an HTML file or another text-based format.

The development of XSLT progressed in several stages as people learned more about requirements for the language.

**XML Query Language** Since XML allows people to define their own markup tags for documents, transforming one XML document into another became a common requirement. Also, since browsers can’t display XML documents directly, it was necessary to transform XML into HTML to allow browsers to display it on a web page.

To meet these needs, Microsoft, Texcel, and webMethods submitted a proposal in September 1998 to the W3C called the XML Query Language or XQL. Part of that proposal was the use of the XSL pattern language as the basis for a general query mechanism for XML documents.

**eXtensible Stylesheet for Transformation** In May of 1999, the W3C decided to unify all the research that had been going on in ”a common core semantic model for querying” and one result was introduction of the eXtensible Stylesheet Language for Transformation or XSLT.

**XPath** During the development of XSLT, another member of the XML family, known as XPointer was defined. XPointer takes the idea of anchor tags to a new level. Both XPointer and XSLT needed a way to point to various parts of a document. XSLT needed it to select the part of the document that would be transformed and XPointer for linking two documents. The solution was to provide a common syntax and semantics that both XSLT and XPointer could use. This new subset was called XPath. Although XPath is a subset of XSLT, it can also be used it on its own.

Using XPath, we can specify the locations of document structures or data in an XML document, and then process the information using XSLT. In practice, it

can be difficult to determine where XSLT stops and where XPath starts, but they were developed as two different standards in the W3C. When we are working with XSLT, the context for a query is the node in the source XML document currently being processed, ”/resource” in our case. XPATH then offers various commands, based on the structure of XML files, for example:

**value-of select** =*”tag”* allows displaying the value of a certain *tag* from the XML file.

**for-each select** =*”tag”* allows the structure of loops based on the number of appearances of a certain *tag*.

**choose** allows *if-then* structures.

For a complete description of XSLT we refer to [94].

### Converting XML to PICA

As you can see from fig. 6.2 our system also includes a translation from XML-files to the PICA format. Some example XSLT commands are:

**Simple tag values** Some PICA entries can easily be derived from XML entries, or combination of XML-entries: *4000 title* is the combination of the metadata attributes title, publisher, publicationPlace and creator.

```
4000 <xsl:value-of select="/resource/title"/>/
     <xsl:value-of select="/resource/publisher"/>,
     <xsl:value-of select="/resource/publicationPlace"/>.
     <xsl:value-of select="/resource/creator"/>
```

**Loops** For every author (instances of the attribute creator) there is a new ordered PICA entry:

```
<xsl:for-each select="/resource/creator">
30<xsl:value-of select="position()+10"/>
   <xsl:value-of select="."/>
</xsl:for-each>
```

**If-then structures** If the attribute relationType has the value ”isCompiledBy”, than the related DOI has to appear in the PICA category *4227 Compilation*, otherwise it appears in *4201 Footnote*.

```
<xsl:for-each select="/resource/relatedDOIs">
<xsl:choose>
<xsl:when test="relationType='isCompiledBy'">
4227 <xsl:value-of select="relatedDOI"/>
</xsl:when>
<xsl:otherwise>
4201 <xsl:value-of select="relationType"/>:<xsl:value-of select="relatedDOI"/>
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
```

The XSLT code of the complete transformation can be found in the appendix.

## 6.5 Status

Registration of scientific primary data has always been an important issue. With the new digital library techniques, it is finally made possible.

In cooperation with

- World Data Center for Climate (WDCC) (see [99])
- Geoforschungszentrum Potsdam (GFZ) (see [48])
- Alfred Wegener Institute (Marum/AWI) (see [4])
- Deutsches Klima Rechenzentrum (DKRZ) (see [32])
- Max Plank Institute for Meteorology (MPIM) (see [68])

the TIB now is the worlds first registration agency for primary data in the field of earth sciences.

The web service installed at the TIB is fully functional and running. We have registered 30 citable and 200 core datasets in 2004 so far. We expect an amount of approximately 150,000 datasets to be registered by the *TIB* until the end of 2005. The registration of primary data will be widened to other science fields in 2006.

# Chapter 7

# Conclusion and further work

The last five years have seen dramatic changes in our dealing with electronic resources. Especially the annotation of resources with metadata has become more and more important. We have used metadata in these years in three different areas:

**eLearning** In our work with metadata for eLearning we have tried to follow the architecture of the Semantic Web as closely as possible. The usage of RDF for our LOM-subset has proven itself effective and simple. The reuse of as many Dublin Core properties as possible for the RDF binding of LOM helped us to convince different communities to accept our subset. Storing the RDF triples in a central database, we were able to realize metadata based retrieval tasks even of a more complex kind with basic database operations. The usage of RDF to model our content classification hierarchies has worked well, too. If we would model these ontologies again however we would use OWL as we did in our later projects, as described in chapter 5. This would have enabled us to implement a logical layer for our inference rules conform to the definition of the semantic web tower. Instead we had realized the logic on top our metadata descriptions with Prolog and had to translate the metadata back and forth. Unfortunately we also never found the time to include all our different techniques and theories in one course metadata editor for P2P infrastructures as presented briefly in section 4.6. The realization of some of our ideas inside the SHAME-editor in use at Cornelssen however has proven most of our ideas practical.

**Context based information retrieval** For this scenario we could consider all Semantic Web principles. The system was completely implemented and is

running in test version at the *Forschungszentrum L3S*. As soon all ontologies would be filled with content it could actually be used to guide visitors.

**Digital libraries** As mentioned in chapter 2: The promise of digital information organization implies the possibility of disseminating materials and information far beyond what has ever been imagined. The challenges and opportunities that lie in the combination of the huge digital library archives with modern information retrieval and semantic web technologies are numberless. Work in this area as described has just begun and promises to become more and more important in the next years.

We believe that the usage of metadata will have its main importance in the control of the huge but ordered amounts of resources available at libraries. Hopefully the experiences we will gain in this task will be valid for the mastering of the even larger, chaotic and unordered amount of information in the WWW to finally achieve the semantic web that has been a dream for almost 7 years now.

Another important usage scenario is the registration of scientific data, as described in chapter 6: Registration of scientific primary data has always been an important issue. In the scope of the project presented in that chapter it is finally made possible. We believe that this project has the power to fundamentally change the world of scientific publication in the next years. Therefore we included it in our thesis to present what enormous possibilities can arise in the *usage of metadata*.

# Appendix A

## The Learning Objects Metadata Standard Schema LOM - extended with inference rules

| Nr | Name | Value space | Inference Rules |
|---|---|---|---|
| 1 | **General** | | |
| 1.1 | Identifier | - | none |
| 1.2 | Title | - | none |
| 1.3 | Language | LanguageID = Langcode | inheritance_along_ **dcterms:hasPart**(Language). |
| 1.4 | Description | - | inheritance_along_ **dcterms:hasPart**(Description). |
| 1.5 | Keyword | - | inheritance_along_ **dcterms:hasPart**(Keyword). inverseInheritance_along_ **dcterms:format**(Keyword). inheritance_along_ **dcterms:hasVersion**(Keyword). |
| 1.6 | Coverage | - | inheritance_along_ **dcterms:hasPart**(Coverage). inverseInheritance_along_ **dcterms:format**(Coverage). inheritance_along_ **dcterms:hasVersion**(Coverage). |
| 1.7 | Structure | atomic, collection, networked, hierarchical linear | none, should be defined by the author |
| 1.8 | Aggregation Level | 1,2,3,4 | maxSummation_along_ **dcterms:hasPart**(AggregationLevel). |
| 2 | **Life Cycle** | | |
| 2.1 | Version | - | none |
| 2.2 | Status | draft, final revised, unavailable | inheritance_along_ **dcterms:hasPart**(Status). |
| 2.3 | *Contribute* | | |
| 2.3.1 | Role | author, publisher, unknown, ... | inheritance_along_ **dcterms:hasPart**(Role). |
| 2.3.2 | Entity | vCard | inheritance_along_ **dcterms:hasPart**(Entity). |
| 2.3.3 | Date | Datatype: DateTime | inheritance_along_ **dcterms:hasPart**(Date). |
| 3 | **Meta-Metadata** | | |
| 3.1 | Identifier | - | none |
| 3.3 | *Contribute* | | |
| 3.2.1 | Role | creator, validator | inheritance_along_ **dcterms:hasPart**(Role). |
| 3.2.2 | Entity | vCard | inheritance_along_ **dcterms:hasPart**(Entity). |
| 3.2.3 | Date | Datatype: DateTime | inheritance_along_ **dcterms:hasPart**(Role). |
| 3.3 | Metadata Schema | Repertoire of ISO/IEC 10646-1:2000 | inheritance_along_ **dcterms:hasPart**(Metadata Schema). |
| 3.4 | Language | see 1.3 | inheritance_along_ **dcterms:hasPart**(Language). |
| 4 | **Technical** | | |
| 4.1 | Format | MIME types | inheritance_along_ **dcterms:hasPart**(Format). |
| 4.2 | Size | ISO/IEC 646:1991 | summation_along_ **dcterms:hasPart**(Size). |
| 4.3 | Location | Repertoire of ISO/IEC 10646-1:2000 | none |
| 4.4 | Requirement | see LOM-Standard | inheritance_along_ **dcterms:hasPart**(Requirement). |
| 4.5 | Installation Remarks | - | inheritance_along_ **dcterms:hasPart**(Installation Remarks). |
| 4.6 | Other Platform Requirements | - | inheritance_along_ **dcterms:hasPart**(Other Platform Requirements). |
| 4.7 | Duration | Datatype: Duration | summation_along_ **dcterms:hasPart**(Duration). |

| | | | |
|---|---|---|---|
| 5 | **Educational** | | |
| 5.1 | Interactivity Type | - | inheritance_along_ **dcterms:hasPart**(Interactivity Type). |
| 5.2 | Learning Resource Type | exercise, simulation, questionnaire, ... | inheritance_along_ **dcterms:hasPart**(Learning Resource Type). |
| 5.3 | Interactivity Level | low, medium, high, very high | none, should be defined by the author |
| 5.4 | Semantic Density | low, medium, high, very high | none, should be defined by the author |
| 5.5 | Intended End User Role | teacher, author, learner, manager | none |
| 5.6 | Context | school, higher education , training, other | none |
| 5.7 | Typical Age Range | - | none |
| 5.8 | Difficulty | easy, medium, difficult, very difficult | none |
| 5.9 | Typical Learning Time | Datatype: Duration | none |
| 5.10 | Description | - | none |
| 5.11 | Language | LanguageID = Langcode | none |
| 6 | **Rights** | | |
| 6.1 | Cost | yes,no | booleanOR_along_ **dcterms:hasPart**(lom-rights:cost). |
| 6.2 | Copyright and Other Restrictions | - | inheritance_along_ **dcterms:hasPart**(Copyright and Other Restrictions ). |
| 6.3 | Description | - | inheritance_along_ **dcterms:hasPart**(Description). |
| 7 | **Relation** | | |
| 7.1 | Kind | **dcterms:** hasPart,isPartOf requires, isRequiredBy hasVersion, isVersionOf hasFormat, isFormatOf references, isReferencedBy isBasedOn, isBasisFor | inverse(**dcterms:hasPart**, **dcterms:isPartOf**). inverse(**dcterms:requires**, **dcterms:isRequiredBy**). inverse(**dcterms:hasVersion**, **dcterms:isVersionOf**). inverse(**dcterms:hasFormat**, **dcterms:isFormatOf**). inverse(**dcterms:references**, **dcterms:isReferencedBy**). inverse(**dcterms:isBasedOn**, **dcterms:isBasisFor**). outwardInheritance_along_ **dcterms:hasPart**(**dcterms:requires**). inverseInheritance_along_ **dcterms:hasFormat**(**dcterms:requires**). OutwardInheritance_along_ **dcterms:hasVersion**(**dcterms:requires**). transitive(**dcterms:hasPart**). transitive(**dcterms:isPartOf**). |
| 8 | **Annotation** | | |
| 8.1 | Entity | vCard | inheritance_along_ **dcterms:hasPart**(Entity). |
| 8.2 | Date | Datatype: DateTime | inheritance_along_ **dcterms:hasPart**(Date). |
| 8.3 | Description | - | inheritance_along_ **dcterms:hasPart**(Description). |

| | | | |
|---|---|---|---|
| 9 | **Classification** | | |
| 9.1 | Purpose | discipline, idea, prerequisite, etc. see the LOM Standard | inheritance_along_ **dcterms:hasPart**(**Purpose**). inverseInheritance_along_ **dcterms:hasFormat**(**Purpose**). inheritance_along_ **dcterms:hasVersion**(**Purpose**). |
| 9.2 | *Taxon Path* | | |
| 9.2.1 | Source | Repertoire of ISO/IEC 10646-1:2000 | inheritance_along_ **dcterms:hasPart**(**Source**). inverseInheritance_along_ **dcterms:hasFormat**(**Source**). inheritance_along_ **dcterms:hasVersion**(**Source**). |
| 9.2.2 | *Taxon* | - | |
| 9.2.2.1 | Id | Repertoire of ISO/IEC 10646-1:2000 | inheritance_along_ **dcterms:hasPart**(**Id**). inverseInheritance_along_ **dcterms:hasFormat**(**Id**). inheritance_along_ **dcterms:hasVersion**(**Id**). |
| 9.2.2.2 | Entry | | inheritance_along_ **dcterms:hasPart**(**Entry**). inverseInheritance_along_ **dcterms:hasFormat**(**Entry**). inheritance_along_ **dcterms:hasVersion**(**Entry**). |
| 9.3 | Description | | inheritance_along_ **dcterms:hasPart**(**Description**). inverseInheritance_along_ **dcterms:hasFormat**(**Description**). inheritance_along_ **dcterms:hasVersion**(**Description**). |
| 9.4 | Keyword | | inheritance_along_ **dcterms:hasPart**(**Keyword**). inverseInheritance_along_ **dcterms:hasFormat**(**Keyword**). inheritance_along_ **dcterms:hasVersion**(**Keyword**). |

# Appendix B

## The XSL transformation from XML to PICA

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:template match="/">
<!-- XSL stylesheet to translata the XML metadata into PICA format -->
<!--****************************************************************** -->
0500 Oax <!--Default value-->
<!--****************************************************************** -->
1100 <xsl:value-of select="substring(resource/publicationDate,0, 5)"/>
          \$  <xsl:value-of select="resource/publicationDate"/>
<!--****************************************************************** -->
1110 o3 <<!--Default value (digital object-->
<!-- Other entries depend on the vocabulary -->
 <xsl:choose><xsl:when test="resource/resourceType=’Dataset’">
1110 d6 </xsl:when>
<xsl:when test="resource/resourceType=’PrimaryData’">
1110 d6</xsl:when>
<xsl:when test="resource/resourceType=’JournalArticle’">
1110 z6</xsl:when>
<xsl:when test="resource/resourceType=’NewspaperArticle’">
1110 z7</xsl:when>
<xsl:when test="resource/resourceType=’Image’">
1110 b5</xsl:when>
<xsl:when test="resource/resourceType=’Map’">
1110 ka</xsl:when>
<xsl:when test="resource/resourceType=’Article’">
1110 fd</xsl:when>
<xsl:when test="resource/resourceType=’Diagram’">
1110 df</xsl:when>
<xsl:otherwise><!-- If the resource Type cannot match -->
4201 ResourceType: <xsl:value-of select="resource/resourceType"/>
<!--, The type appears as footnote-->
</xsl:otherwise>
</xsl:choose>
<!--****************************************************************** -->
1500 /1<xsl:value-of select="resource/language"/>
<!--****************************************************************** -->
2199 <xsl:value-of select="resource/DOI"/>
<!-- DOI appears as unique identifier-->
<!--****************************************************************** -->
<xsl:for-each select="resource/creator">
<!--Loop over all creators-->
30<xsl:value-of select="position()+9"/><!--Numbering-->
<xsl:value-of select="."/>
</xsl:for-each>
<!--****************************************************************** -->
<xsl:for-each select="resource/publisher">
<!--Loop over all publishers -->
31<xsl:value-of select="position()+19"/><!--Numbering-->
        <xsl:value-of select="."/><!-- Verlegerk??rzel -->
</xsl:for-each>
<!--****************************************************************** -->
4000 <xsl:value-of select="resource/title"/>/
```

```
        <xsl:value-of select="/resource/publisher"/>:
        <xsl:value-of select="resource/publicationPlace"/>.
        <xsl:value-of select="/resource/creator"/>
<!-- Title is: title, publisher, place and author-->
<!--********************************************************************* -->
<xsl:if test="not(resource/edition='1')">
<!--The first edition is not mentioned -->
            <xsl:for-each select="resource/edition">
4020  <xsl:value-of select="../edition"/> ed.</xsl:for-each></xsl:if>
<!--********************************************************************* -->
4031 <xsl:value-of select="resource/publicationPlace"/> :
   <xsl:value-of select="/resource/publisher"/>
<!--********************************************************************* -->
<xsl:for-each select="resource/size">
<!--Loop over all possible formats-->
406<xsl:value-of select="position()-1"/> Online-Ressource
   (<xsl:value-of select="value"/> <xsl:value-of select="unit"/>)
</xsl:for-each>
<!--********************************************************************* -->
4083 &lt;1&gt;html = D http://dx.doi.org/<xsl:value-of select="resource/DOI"/>
4083 &lt;1&gt;html = G urn:nbn:de:tib-<xsl:value-of select="resource/DOI"/>
<!--********************************************************************* -->
<xsl:for-each select="resource/mode">
4201 Mode: <xsl:value-of select="."/>
</xsl:for-each>
<!--********************************************************************* -->
4201 StructuralType: <xsl:value-of select="resource/structuralType"/>
<!--********************************************************************* -->
<xsl:if test="resource/creationDate">
<!-- CreationDate is optional-->
4201 CreationDate: <xsl:value-of select="resource/creationDate"/>
</xsl:if>
<!--********************************************************************* -->
4207 <xsl:value-of select="resource/description"/>
<!--********************************************************************* -->
4238 Format: <xsl:value-of select="resource/format"/>
<!--********************************************************************* -->
<xsl:for-each select="resource/relatedDOIs">
<!--Related resources are either -->
<xsl:choose>
<xsl:when test="relationType='isCompiledBy'">
4227 <xsl:value-of select="relatedDOI"/>
<!--Compilations, or -->
</xsl:when>
<xsl:otherwise>
4201 <xsl:value-of select="relationType"/>:<xsl:value-of select="relatedDOI"/>
<!--  andere -->
</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
   <!--********************************************************************* -->
7001 z <!--Default value-->
4801 Primaerdaten <!--Default value-->
7133 &lt;1&gt;html = D http://dx.doi.org/<xsl:value-of select="resource/DOI"/>
    <!--********************************************************************* -->

</xsl:template>
```

```
</xsl:stylesheet>
```

# Bibliography

[1] F. Abel *Kontextbasierte Informationsbereitstellung auf Basis des Semantic Web*
Bachelor Thesis 2004, University of Hannover

[2] ACM *The ACM Computing Classification System–1998 Version*
http://www.acm.org/class/1998/

[3] Technical Team ADLSCORM *Specification V1.2*
http://www.adlnet.org/Scorm/scorm.cfm

[4] Alfred Wegener Institute (Marum/AWI)
http://www.awi-bremerhaven.de

[5] Joint Steering Committee for Revision of AACR *Anglo-American cataloguing rules, second edition (AACR2)*
Revisions 1983.

[6] G. Antoniou and F. van Harmelem *A Semantic Web Primer*
MIT press 2004

[7] The Apache Cocoon project
http://cocoon.apache.org/

[8] F. Baader *The Description Logic Handbook*
ISBN: 0-52178-176-0

[9] P. Bahl and V.N. Padmanabhan *RADAR: An In-Building RF-based User Location and Tracking System*
Microsoft Research, 2000

[10] T. Berners-Lee *Design issue about Metadata architecture*
http://www.w3.org/DesignIssues/Metadata.html

[11] T. Berners-Lee, J. Hendler and O. Lassila *The semantic web*
Scientific American, Mai 2001

[12] T. Berners-Lee *Semantic Web Road map*
September 1998

[13] J. Brase *ACMCCS – The ACM Classification for Computer Science as a rdf-file*
http://www.kbs.uni-hannover.de/Uli/SWT_ Ontologie.rdf

[14] J. Brase *AIOnt – An extension of the ACM CCS for the field of Artificial Intelligence*
based on a paper by D.L. Waltz
http://www.kbs.uni-hannover.de/Uli/AI_ Ontologie.rdf

[15] J. Brase *mySWEBOK – A classification for Software enginering*
based on the SWEBOK
http://www.kbs.uni-hannover.de/Uli/SWT_ Ontologie.rdf

[16] J.Brase, M. Nilsson and M. Palmer *The LOM RDF binding - principles and implementation*
3rd Annual Ariadne Conference
Leuven, Belgium, November 2003

[17] J. Brase and W. Nejdl *Annotation for an open learning repository for computer science - case study and OLR3 editor*
from ”Annotation for the Semantic Web”, IOS-press 2003
ISBN 1-58603-345-X

[18] J. Brase and W. Nejdl *Ontologies for eLearning*
from ”Handbook on Ontologies”, Springer-Verlag 2003
ISBN 3-540-40834-7

[19] J. Brase, T. Kunze and W. Nejdl *Editing Learning objects metadata*
15th European conference on artifical intelligence (SAAKM workshop).
Lyon, France, July 2002

[20] J. Brase, M. Painter and W. Nejdl *Completion Axioms for learning object metadata - Towards a formal description of LOM*
3rd international conference on advanced learning technologies (ICALT).
Athens, Greece, July 2003

[21] J. Brase *Using digital library techniques - Registration of scientific primary data*
from ”Research and advanced technology for digital libraries - LNCS 3232”, Springer Verlag 2004
ISBN 3-540-23013-0

[22] D. Brickley and R. V. Guha *W3C Resource Description Framework (RDF) Schema Specification*
http://www.w3.org/TR/1998/WD-rdf-schema

[23] The integrated catalogue of the British library
http://catalogue.bl.uk/

[24] Canada Institute for Scientific and technical information
http://cat.cisti-icist.nrc-cnrc.gc.ca/search

[25] Council for the Central Laboratory of the Research Councils
*CLRC Scientific Metadata Model*
http://www-dienst.rl.ac.uk/library/2002/tr/dltr-2002001.pdf

[26] Committee on Data for Science and Technology (coData)
http://www.codata.org

[27] Danish National Research Database
http://www.forskningsdatabase.dk/

[28] DAML+OIL http://www.daml.org/2001/03/daml+0il-index.html

[29] L. Dempsey and R. Heery *Specification for resource description methods*
http://www.ukoln.ac.uk/metadata/desire/overview/

[30] Die Deutsche Bibliothek (German library)
http://www.ddb.de

[31] Deutsche Forschungsgemeinschaft (German research foundation)
http://www.dfg.de

[32] Deutsches Klima Rechenzentrum (DKRZ)
http://www.dkrz.de

[33] DFKI, W. Wahlster *SmartKom*
http://www.smartkom.org

[34] H. Dhraief, W. Nejdl, B. Wolf and M. Wolpers *Open Learning Repositories and Metadata Modeling*
International Semantic Web Working Symposium (SWWS)
Stanford, United States, July 2001

[35] International DOI foundation
http://www.doi.org

[36] M. Duan *An Introduction to Art, the Wireless Way*
mpulse, october, 2002

[37] The Dublin Core Metadata Initiative (DCMI)
http://dublincore.org/

[38] DCMI *Expressing Qualified Dublin Core in RDF / XML (Proposed Recommendation)*
http://dublincore.org/documents/2002/04/14/dcq-rdf-xml/

[39] DCMI *Dublin Core Metadata Element Set, Version 1.1: Reference Description*
http://dublincore.org/documents/1999/07/02/dces/

[40] The Edutella Project
http://edutella.jxta.org

[41] The EU IST project ELENA
http://www.elena-project.org/

[42] Enhydra Open Source Java/XML Application Server
http://enhydra.enhydra.org

[43] Project ”Enhancement of Persistent Identifier Services - Comprehensive Method for unequivocal Resource Identification”
http://www.persistent-identifier.de/

[44] Eyeled
http://www.eyeled.de

[45] Fraunhofer Institut, K. Heuwinkel *Explore - Gaming and Guiding System for Museum and Exhibition Environments*
http://www.isst.fhg.de/german/projekte/2004/explore.html

[46] Fraunhofer Institut, *LISTEN*
http://listen.imk.fraunhofer.de/d.html

[47] D. Fensel *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*
Springer, 2001

[48] Geoforschungszentrum Potsdam (GFZ)
http://www.gfz-potsdam.de

[49] L. Gong *Project JXTA: A Technology Overview*
http://www.jxta.org/project/www/docs/TechOverview.pdf

[50] N. Guarino *Formal Ontology in Information Systems*
FOIS'98, Trento, Italy.

[51] S. Guth, G. Neumann and B. Simon *UNIVERSAL - Design Spaces of Learning Media*
34th Hawaii International Conference on System Sciences
Maui, United States, January, 2001

[52] The Handle System
http://www.handle.net

[53] HP Labs Mobile Systems and Solutions *The Cooltown Program*
http://www.cooltown.com

[54] IEEE *The guide to the Software engeneering body of Knowledge*
http://www.swebok.org

[55] S. Handschuh, S. Staab and A. Maedche *CREAM - Creating relational metadata with a component-based, ontology-driven annotation framework*
Workshop on Knowledge Markup and Semantic Annotation at the First International Conference on Knowledge Capture (K-CAP'2001)
Victoria, Canada, 2001

[56] S. Hoermann, A. Faatz, et.al *Ein Kurseditor fr modularisierte Lernresourcen auf der Basis von LOM zur Erstellung von adaptierbaren Kursen*
LLWA 01 - GI-Workshopwoche "Lernen-Lehren-Wissen-Adaptivitt", 2002

[57] IMS Global Learning Consortium
http://www.imsproject.org

[58] IMS *Learning Resource Metadata Specification V1.2.1*
http://www.imsproject.org/metadata/index.html

[59] IMS *IMS Resource Description Framework(RDF) Bindings*
http://www.imsproject.org/rdf/

[60] The Internet Engineering Task Force (IETF) http://www.ietf.org

[61] IETF *RFC1766 - Tags for the Identification of Languages*
http://www.ietf.org/rfc/rfc1766.txt

[62] Jena A Semantic Web Framework for Java
http://jena.sourceforge.net/

[63] Joseki - Jena RDF Server
http://www.joseki.org/

[64] Forschungszentrum L3S
http://www.l3s.de

[65] O. Lassila and R. R. Swick *W3C Resource Description Framework (RDF) Model and Syntax Specification*
http://www.w3.org/TR/REC-rdf-syntax

[66] Learning Technology Standards Comittee of the IEEE: *Draft Standard for Learning Objects Metadata IEEE P1484.12.1/D6.4*12. June 2002)
http://ltsc.ieee.org/doc/wg12/LOM_ 1484_ 12_ 1_ v1_ Final_ Draft.pdf/

[67] The Marc 21 webpage
http://www.loc.gov/marc/

[68] Max Plank Institute for Meteorology (MPIM)
http://http://www.mpimet.mpg.de/

[69] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmr and T. Risch *EDUTELLA: a P2P Networking Infrastructure based on RDF*
11th International World Wide Web Conference
Honululu, United States, May 2002
http://edutella.jxta.org/reports/edutella-whitepaper.pdf

[70] W. Nejdl, B. Wolf, S. Staab and J. Tane *EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network*
Semantic Web Workshop, 11th International World Wide Web Conference
Honululu, United States, May 2002

[71] M. Nilsson. *The RDF binding of LOM*
May 2001
http://kmr.nada.kth.se/el/ims/metadata.html

[72] M. Nilsson and M. Palmer *Conzilla - Towards a Concept Browser*
(CID-53), KTH, 1999.

[73] OntoWeb *Ontology-based information exchange for knowledge management and electronic commerce*
http://www.ontoweb.org

[74] Center for Viden teknologi *ORBIT metadata format (v.1.3) - System administrator's guide*

[75] M. Palmer and H. Eriksson *SHAME - A RDF editor,*
(CID), KTH, 2003

[76] The PICA webpage
http://oclcpica.org/

[77] C. Plott and R. Ball *Mit Sicherheit zum Dokument - Die Identifizierung von Online-Publikationen*
B.I.T. journal 2004, 1:11-20

[78] *PORTAL to computing literature*
http://portal.acm.org/portal.cfm

[79] P. Prasithsangaree, P. Krishnamurthy and P.K. Chrysanthis *On indoor position location with wireless lans*
13th IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2002)
Lisbon, Portugal, September 2002.

[80] EU Network of Excellence PROLEARN
http://www.prolearn-project.org/

[81] A. Seaborne *A Programmer's Introduction to RDQL*
http://jena.sourceforge.net/
February 2004

[82] A. Silberschatz, H. F. Korth and S. Sudarshan *Database Systems Concepts*
McGraw-Hill Higher Education 2001, 4

[83] SiRPAC RDF Parser, Stanford
http://www-db.stanford.edu/ melnik/rdf/api.html

[84] *SMETE Digital Library* http://www.smete.org/

[85] L. Miller, A. Seaborne and A. Reggiori, A. , *Three Implementations of SquishQL, a Simple RDF Query Language*
April 2002

[86] The ULI-project
http://www.uli-campus.de

[87] K. Tolle*VRP RDF Parser*
ICS Forth, Greece
http://www.ics.forth.gr/proj/isst/RDF

[88] W3C *OWL Web Ontology Language Guide, W3C Recommendation*
http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[89] W3C *OWL Web Ontology Language Overview, W3C Recommendation*
http://www.w3.org/TR/2004/REC-owl-features-20040210/

[90] W3C *Resource Description Framework (RDF), W3C Recommendation*
http://www.w3.org/RDF/

[91] W3C *Refactoring RDF/XML Syntax (Working Draft)*
http://www.w3.org/TR/rdf-syntax-grammar/

[92] W3C *The vCard RDF binding*
http://www.w3.org/2001/vcard-rdf/3.0

[93] W3C *XML Path Language (XPath) Version 1.0, W3C Recommendation*
http://www.w3.org/TR/xpath

[94] W3C *XSL Transformations Version 1.0, W3C Recommendation*
http://www.w3.org/TR/xslt

[95] W3C *XML Schema Version 1.1, W3C Recommendation*
http://www.w3.org/XML/Schema

[96] Weltkulturerbe Völklinger Hütte
http://www.voelklinger-huette.org

[97] World Wide Web Consortium
http://www.w3.org/

[98] D. Waltz *Scientific datalink's artificial intelligence classification scheme*
The AI Magazine 1985, 6(1):58-63

[99] World data center climate (WDCC)
http://www.mad.zmaw.de/wdcc/

[100] WTEC Panel Report *Digital Information Organization in Japan*
http://www.wtec.org/loyola/digilibs/toc.htm

# List of Figures

# List of Tables

# Lebenslauf Jan Oliver Brase

**Persönliche Daten:**

- *Geburtsdatum:* 20.11.1970 in Hannover
- *Eltern:* Dr. Alfred Christian Oswald Brase, Arzt & Renate Irma Friederike Brase, geb. Schläger-Priesing, Lehrerin
- *Ehepartner:* Seit 13.4.2002 verheiratet mit Dr. Katja Brase, geb. Bäumer.
- *Kinder:* Seit 22.10.2003 gemeinsamer Sohn Jonathan Oswald Brase.

**Bildungsgang:**

- *1977 - 1981* Grundschule Am Mühlenweg, Hannover
- *1981 - 1983* Orientierungsstufe Isernhagen
- *1983 - 1990* Gymnasium Isernhagen
- *1990* Abitur
- *1992* 1 Semester Studium Generale, Universität Hannover
- *1992 - 1999* Studium der Mathematik mit Nebenfach Informatik, Universität Hannover
- *1999* Diplom mit Schwerpunkfach ”Zahlentheorie”
  Titel der Diplomarbeit: ”Nullstellen Dirichletscher L-Funktionen und ihrer Ableitungen”

**Beschäftigungen:**

- *1990 - 1992* Zivildienst in der Werner-Dicke-Schule für Körtperbehinderte Kinder, Hannover
- *1992 - 1994* Nebenjob in der EDV der Kassenärztlichen Vereinigung Niedersachsen
- *1994 - 1998* Wissenschaftliche Hilfskraft im Institut für Angewandte Mathematik, Universität Hannover

- *1999 - 2003* Wissenschaftliche Mitarbeiter am Institut für Informationssysteme - Fachgebiet Wissensbasierte Systeme, Universität Hannover
- *2004* - Angestellt am Forschungszentrum L3S, Universität Hannover

**Forschungsaufenthalt:** *2003* August - September am Center for User-oriented IT Design (CID) der KTH Stockholm, Schweden

# Vollständige Publikationsliste

Stand 08/2005

- *The LOM RDF binding - principles and implementation*
  In Zusammenarbeit mit M. Nilsson, M. Palmer
  3rd Annual Ariadne Conference, November 2003
  Leuven, Belgien

- *Annotation for an open learning repository for computer science - case study and OLR3 editor*
  In Zusammenarbeit mit W. Nejdl
  ”Annotation for the Semantic Web”, IOS-press 2003
  ISBN 1-58603-345-X

- *Ontologies for eLearning*
  In Zusammenarbeit mit W. Nejdl
  ”Handbook on Ontologies”, Springer-Verlag 2003
  ISBN 3-540-40834-7

- *Integrating Adaptive Hypermedia Techniques and Open RDF-based Environments*
  In Zusammenarbeit mit P. Dolog, R. Gavriolae, W. Nejdl
  12th international world wide web conference (WWW2003). May 2003
  Budapest, Ungarn

- *Editing Learning objects metadata*
  In Zusammenarbeit mit W. Nejdl, T. Kunze
  15th European conference on artifical intelligence (SAAKM workshop). July 2002
  Lyon, Frankreich

- *Schema Driven Input of RDF Metadata*
  In Zusammenarbeit mit W. Nejdl, T. Kunze
  1st international Semantic Web conference (ISWC). June 2002
  Sardinien, Italien

- *Infering Metadata for a Semantic Web Peer-to-Peer Environment*
  In Zusammenarbeit mit M. Painter
  Educational Technology and Society Journal, April 2004 issue (Volume 7 Issue 2).

- *Completion Axioms for learning object metadata - Towards a formal description of LOM*
  In Zusammenarbeit mit M. Painter, W. Nejdl
  *Best poster award* 3rd international conference on advanced learning technologies (ICALT) July 2003
  Athen, Griechenland

- *Using digital library techniques - Registration of scientific primary data*
  8th European Conference on Digital libraries, September 2004
  Bath, England
  & "Research and advanced technology for digital libraries" Springer LNCS 3232
  ISBN 3-540-23013-0

- *Neue Wege des Publizierens: Die Zitierfhigkeit wissenschaftlicher Primrdaten*
  Together with project STD-DOI
  11.IuK-Jahrestagung 2005. May 2005
  Bonn, Germany

- *Standards for the publication of scientific data by World Data Centres and the National Library of Science and Technology in Germany*
  Together with project STD-DOI
  Workshop on International Scientific Data, Standards and Digital Libraries at Joint conference for digital libraries (JCDL2005) June 7-11
  Denver, USA

- *Webservice infrastructure for the registration of scientific primary data*
  Together with U. Schindler and M. Diepenbroeck
  9th European Conference on digital libraries (ECDL 2005), September 2005
  Vienna, Austria
  & "Research and advanced technology for digital libraries" Springer LNCS 3652
  ISBN 3-540-28767-1

- *Data Centres and their role in Publication and Access to Data*
  Together with project STD-DOI & E.Paliouras
  Ensuring Long-term Preservation and Adding Value to Scientific and Technical data (PV 2005) 21-23 November 2005
  Edinburgh, UK

- *Getting the most out of COCOON: A XML-based webservice for a registration agency*
  Together with J. Hinzmann
  XML 2005, November 2005
  Atlanta, USA

# Danksagung

Mein allererster Dank gebührt Herrn Prof. Dr. Nejdl für das Vertrauen und die Unterstützung die er mir gegeben und die Wissenschaftliche Freiheit, die er mir gelassen hat. Außerdem danke ich Herrn Prof. Lipeck für die Zeit, die er sich genommen hat, seine konstruktive Kritik und für seine profunde Kenntnis der Promotionsordnung.

Ebenso möchte ich mich bei allen meinen Kollegen im KBS für ein angenehmes Arbeitsklima und die wissenschaftliche Zusammenarbeit bedanken. Ein besonderer Dank geht hierbei an Franziska Pfeffer.

Während meiner 5 jährigen Forschung im Bereich der Metadaten habe ich mit verschiedenen nationalen und internationalen Kollegen zusammengearbeitet. Besonders bedanken möchte ich mich bei Ambjörn Naeve, Mikael Nilsson und Matthias Palmer vom CID Stockholm für meinen schönen und erfolgreichen Forschungsaufenthalt in Schweden im Jahre 2003.

Bedanken möchte ich mich bei Fabian Abel und Jan Hinzmann, die mich als Bachelor-Arbeiter und studentische Hilfskraft bei meiner Arbeit unterstützt haben.

Ein besonders herzlicher Dank geht an meine Eltern und meine Schwester, die mich immer zur richtigen Zeit in meiner Arbeit unterstützt, gefördert und vor allem gefordert haben.

Katja, ich liebe Dich und unseren Sohn, und genieße jeden Moment des Lebens mit euch.