Managing Metadata in Open Learning Repositories and P2P Networks

Von der Fakultät für Elektrotechnik und Informatik der
Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur/Doktor-Ingenieurin
genehmigte Dissertation

von Dipl.-Inform. Hadhami Dhraief

geboren am
12. Juli 1971 in Tunis, Tunesien

2005

1. Referent: Prof. Dr. techn. Wolfgang Nejdl
2. Referent: Prof. Dr.-Ing. Klaus Jobmann
Tag der Promotion: 1.11.2005

For Taha Emin & Mourad

“Gebildet ist, wer weiss, wo er findet, was er nicht weiss.” Georg Simmel

Zusammenfassung

"Now, miraculously, we have the Web. For the documents in our lives, everything is simple and smooth. But for data, we are still pre-Web."(Tim Berners-Lee, Business Model for the Semantic Web)

Die erfolgreiche Verwendung und Wiederverwendung, die Suche und die Bearbeitung von Daten, hängen von der effektiven Definition, Verwendung und Verwaltung der Metadaten ab.

Der erste Teil dieser Thesis untersucht Probleme in Bezug auf Lernmetadaten, die als "A und O" jeder Anwendung im Bereich E-Learning betrachtet werden können. Genauer gesagt, wir untersuchen Lernmetadaten im Kontext eines "lokalen" "Open Learning Repository" (OLR). Dabei wird der pädagogische Background in der Bearbeitung von Metadaten hervorgehoben, indem Metadatenstandards und die Strukturierung von Lernmaterialen diskutiert werden. Wir haben, u.a., das Fehlen der Adressierung von Lernprozessen und instruktionalen Theorien in dem "Learning Object Metadata" Standard (LOM) gezeigt, dann haben wir eine Erweiterung von LOM, basierend auf der Einführung einer abstrakten Ebene und der Idee von instruktionalen Rollen, vorgestellt. Wir haben ebenfalls mehrere Kurse basierend auf verschiedenen instruktionalen Theorien strukturiert. Unsere OLRs können als Framework und Testumgebung betrachtet werden, wo Metadatenmodellierungssprachen, Lernmetadatenstandards und Metadatenverwaltung innerhalb eines interdisziplinären Teams vorgestellt und diskutiert werden.

Im zweiten Teil dieser Arbeit werden die Probleme in Bezug auf Lernmetadaten, insbesondere die Metadatenverwaltung, zu den Problemen in Bezug auf Metadaten, die allgemein zur Annotation von Webressourcen verwendet werden, verallgemeinert. Wir haben außerdem die Metadatenverwaltung von der lokalen Umgebung der OLRs zu der verteilten Umgebung der P2P-Netzwerke erweitert. Die OLRs spielen dabei die Rolle der Metadatenanbieter. Obwohl ziemlich viele Datenbanktechniken in dem Kontext von P2P-Netzen wieder angewendet werden können, stellt die P2P-Infrastruktur für die Metadatenverwaltung zusätzliche Herausforderungen, die auf die offene und dynamische Natur dieser Netzwerke zurückzuführen sind, dar. Die Hauptaufgabe hier ist das Ermöglichen einer effizienten, dynamischen und verteilten Anfragebearbeitung. Wir präsentieren hierfür unsere Super-Peer-Topologie und Schema-Wissende, mit Statistiken angereicherte verteilte Routingindexe und zeigen, wie diese Indexe die Verteilung und das dynamische Expandieren der Anfragepläne ermöglichen. Wir präsentieren darauf eine Reihe von Transformationsregeln zur Optimierung von Anfrageplänen und diskutieren verschiedene Optimierungsstrategien im Detail. Zusätzlich zu der Optimierung der verteilten komplexen Anfragebearbeitung untersuchten wir Strategien für semantisches Caching in P2P-Netzwerken, um die Antwortzeiten zu optimieren und das Netz zu entlasten.

Schlagwörter: Metadaten, Peer-to-Peer Netzwerke, Abfragenoptimierung

Abstract

"Now, miraculously, we have the Web. For the documents in our lives, everything is simple and smooth. But for data, we are still pre-Web."(Tim Berners-Lee, Business Model for the Semantic Web)

The successful use and re-use, search, and operation of data, depends on the effective definition, use and management of metadata.

The first part of this thesis considers the issues related to learning metadata, which are the nuts and bolts of any application in the field of e-learning. More precisely we investigate learning metadata issues in the context of a "local" open learning repository (OLR for short). Thereby, we stress the pedagogical background in handling metadata, discussing metadata standards, and structuring learning materials. We demonstrate, inter alia, the lack of addressing learning processes and instructional theories in the learning object metadata standard (LOM). Then, we propose an extension of LOM based on the introduction of an abstraction layer and the notion of instructional roles. We also structure several courses based on different instructional models.

Our open learning repositories can be considered as a framework and a testbed where metadata modeling languages, learning metadata standards, and metadata management are presented and discussed within an interdisciplinary team.

In the second part, we generalize the learning metadata issues, particularly metadata management, to issues related to the broadly used metadata that annotate any resource on the Web. We also expand the metadata management from the local environment of open learning repositories to the distributed environment of peer-to-peer networks. The open learning repositories play then the role of special peers, the *metadata providers*, in the P2P network. Unfortunately, although quite a few database techniques can be re-used in the P2P context, P2P metadata management infrastructures pose additional challenges caused by the open and dynamic nature of these networks. The main task here is to enable an efficient dynamic distributed query processing. For this purpose, we briefly present our super-peer based topology and schema-aware distributed routing indices extended with suitable statistics. Then, we show how these indices facilitate the distribution and dynamic expansion of query plans. After that, we propose a set of transformation rules to optimize query plans and discuss different optimization strategies in detail. In addition to the optimization of complex distributed query processing, we also investigate semantic caching strategies for P2P networks, in order to optimize the query response time and reduce the network load.

Keywords: Metadata, peer-to-peer networks, query optimization

Contents

List of Abbreviations

- ADL - Advanced Distributed Learning
- AICC - Aviation Industry CBT Committee
- ASL - Apache Software Licence
- CBT - Computing Based Training
- CSCL - Computer supported collaborative learning
- CSS - Cascading Style Sheets
- CVS - Concurrent Versions System
- DBMS - Database Management System
- DC - Dublin Core
- DCMES - Dublin Core Metadata Element Set
- DCMI - Dublin Core Metadata Initiative
- DOM - Document Object Model
- GPL - General Public License
- IDE - Integrated Development Environment
- IM - Instructional Model
- IMS - Instructional Management System
- JDBC - Java Database Connectivity
- J2EE - Java 2 Platform Enterprise Edition
- LOM - Learning Object Metadata
- LMML - Learning Material Markup Language
- LTSC - Learning Technology Standards Committee
- MARC - Machine Readable Cataloging

- MCR - Maximal Contained Rewriting
- MPL - Mozilla Public License
- NISO - National Information Standards Organization
- ODBC - Open DataBase Connectivity
- OLR - Open Learning Repository
- PBL - Problem Based Learning
- P2P - Peer to Peer
- PHP - Hypertext Preprocessor
- POM - Project Object Model
- QEP - Query Evaluation Plan
- RDF - Resource Description Framework
- RDFS - RDF Schema
- SCORM - Sharable Content Object Reference Model
- SQC - Sematic Query Caching
- UML - Unified Modeling Language
- URI - Uniform Resource Identifier
- URL - Uniform Resource Locator
- W3C - World Wide Web Consortium
- WAR - Web Application Archive
- XML - eXtensible Markup Language
- XMLC - XML Compiler

Chapter 1

Introduction

In almost any article, discussion or project in the e-learning world, we hit on the terms “Learning objects”, “Learning object metadata” and “learning object repositories”. These terms are mostly freely defined and used without any consensus about their meaning. Various problems have been raised in association with these terms and their technologies. Moreover, many problems have been caused by the broad and varied dimensions of learning, which should make the base in the design of e-learning environments as well as in the definition of learning metadata standards. In general, a lack of addressing instructional theories and learning processes in the field of e-learning can be asserted. This is primarily due to the absence of interdisciplinary work. Roughly speaking, there is a high focus on the “e”, that is on the technologies used in e-learning. Little attention is given to the “learning” itself.

In this thesis, we focus on learning metadata instead of the learning objects. In doing so, we do not consider, for instance, the issues in connection with learning objects such as definition, creation, quality, granularity, administration and management. Our work can be considered as framework and testbed where metadata modeling languages, learning metadata standards, metadata management, and search technologies are presented and discussed within an interdisciplinary team. We are convinced that the interdisciplinary work is a mandatory premise in the field of e-learning.

The first part of this thesis investigate the issues related to learning metadata in the context of a “local” open learning repository. Thereby, we stress the pedagogical background in handling metadata, discussing metadata standards and structuring learning materials. Thus we inter alia investigate the lack of addressing learning processes and instructional theories in the learning object metadata standard (LOM), propose an extension of LOM based on the introduction of the notion of

instructional roles, and also structure different courses based on different instructional models.

In the second part, we generalize the learning metadata issues, particularly metadata management, to issues related to the broadly used metadata that annotate any resource on the Web. We also expand the metadata management from the local environment of open learning repositories to the distributed environment of peer-to-peer networks. The open learning repositories play then the role of special peers, the *metadata providers*, in the P2P network. Unfortunately, although quite a few database techniques can be reused in the P2P context, P2P metadata management infrastructures pose additional challenges caused by the open and dynamic nature of these networks. In P2P networks, we can assume neither global knowledge about metadata distribution, nor the suitableness of static topologies and static query plans for these networks. Unlike in traditional distributed database systems, we cannot assume complete information schema and allocation schema instances but rather work with distributed schema information which can only direct query processing tasks from one node to one or more neighboring nodes. The main task here is to enable an efficient dynamic distributed query processing. Thereby, we assume schema-based P2P networks and complex query processing, where queries need data from more than one peer in order to be executed. In addition to the optimization of complex distributed query processing, we also investigate semantic caching strategies for P2P networks, in order to optimize the query response time and reduce the network load.

1.1 Contribution of this Work

This thesis addresses the following major open issues:

- *Metadata Management in Open Learning Repositories*
  - *Metadata modeling*: We compare the metadata modeling languages RDF/RDFS - the lingua franca of the semantic Web and O-Telos - a deductive object-oriented conceptual modeling language - and analyze their similarities and differences, based on our long experience in modeling and meta modeling of open learning repositories. This comparison provides a better understanding of the strengths and the weaknesses of RDF and its modeling capabilities.
  - *Learning Metadata Standards*: We discuss the Learning Object Metadata Standard LOM and show the lack of description facilities of different instructional roles, which are available for or can be played by a learning object within a course.We extend previous work which has

tried to extend LOM with didactic metadata by introducing an additional abstraction layer to LOM which explicitly takes different instructional theories into account.

- *Open learning repositories*: We propose three generations of open learning repositories which serve as a testbed for the investigated issues on learning metadata, learning metadata standards and learning metadata management. Thereby, we investigate how these open learning repositories can be used to enhance teaching and learning. Thus, we implement and evaluate different instructional models.

- *Metadata Management in P2P Networks*

  - *Schema-based P2P Infrastructure*: We present schema-aware distributed routing indices extended with suitable statistics for our schema-based P2P infrastructure, Edutella, and show how these indices facilitate the distribution and dynamic expansion of query plans.
  - *Query processing in schema-based P2P networks*: We propose a set of transformation rules to optimize query plans and discuss different optimization strategies in detail, enabling efficient distributed query processing in a schema-based P2P network.
  - *Semantic Caching*: We propose a semantic caching component for our schema-based P2P-network based on well-known algorithms for answering queries using materialized views.

1.2 Thesis Overview

This thesis is outlined as follows: Chapter 2 will introduce briefly metadata and present some metadata modeling languages. Thereby, we will compare the metadata modeling languages RDF/RDFS and O-Telos and analyze their similarities and differences, based on our long experience in modeling and meta modeling of open learning repositories. Then, we will focus on metadata for e-learning and introduce standardization in learning. Thereby, we will present the well established Learning Object Metadata Standard (LOM), discuss the insufficiency of LOM in describing learning objects in the context of different instructional theories, and propose an extension of LOM.

In chapter 3, we will bring in what we call open learning repositories and the ideas behind it. We will then present our first prototype OLR2 and describe the underlying architecture and technologies and the user interfaces. We will close this chapter with an evaluation of this first prototype.

Chapter 4 introduces the second generation of open learning repositories, called OLR3. The focus in this chapter will be on the implemented different instructional theories resulting from the work of an interdisciplinary team of computer scientists and pedagogues. This chapter addresses the problem of how an open learning repository can we used to enhance teaching and learning.

In chapter 5 we will present the youngest generation of our open learning repositories developed based on a re-engineering of OLR3. The focus of this prototype is the technical infrastructure. Starting from an evaluation of OLR3, standard components and tools (e.g. the open source object-relational database management system *PostgreSQL*, the project build tool *Maven*, a persistence layer using *Apache Torque*, etc.) were introduced to improve earlier implementation decisions and to release OLR3 as an open source project. We will then put OLR4 in the context of the distributed environment of peer-to-peer networks in Edutella. OLR4 plays then the role of metadata provider as a peer which aims at storing, querying, exchanging and processing any kind of metadata modelled in RDF.

In the context of this thesis, we have been also investigating how to perform the search of metadata for more efficiency in the retrieval of learning materials. Chapter 6 will address the problem and complexity of query processing in the highly distributed, dynamic and open environment of P2P networks, especially in schema-based super-peer-networks. For this purpose, we will first describe briefly our super-peer based topology and schema-aware distributed routing indices extended with suitable statistics. Then, we will explain how this information is extracted and updated. Second, we will show how these indices facilitate the distribution and dynamic expansion of query plans. Third, we will propose a set of transformation rules to optimize query plans and discuss different optimization strategies in detail, enabling efficient distributed query processing in a schema-based P2P network.

Chapter 7 will go on with the challenge of improving the query processing and information retrieval in Edutella. We introduce therefore the well-known problem of answering queries using materialized views and present the most known and efficient algorithms which handle this problem. Based on one of these algorithms, namely the MiniCon-Algorithm, we define and describe semantic caching, which has been implemented in the context of Edutella.

The author has published most of her research results on conferences and in publications over the last years:

- Excerpts from chapter 2 were first published through
  - *Open Learning Repositories and Metadata Modeling*
    In cooperation with W. Nejdl, B. Wolf and M. Wolpers
    International Semantic Web Working Symposium (SWWS) July 30 - August 1, 2001 Stanford University, California, USA.
  - *O-Telos-RDF: A Resource Description Format with Enhanced Meta-Modeling Functionalities based on O-Telos*
    In cooperation with W. Nejdl and M. Wolpers
    Workshop on Knowledge Markup and Semantic Annotation at the First International Conference on Knowledge Capture (K-CAP'2001), Oct. 21 - 23, 2001, Victoria, B.C., Canada.
  - *How are Learning Objects Used in Learning Processes? Instructional Roles of Learning Objects in LOM*
    In cooperation with H. Allert and W. Nejdl
    ED-MEDIA 2002, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Denver Colorado, United States, June 24-29, 2002.
  - *Meta-Level Category Role in Metadata Standards for eLearning. Instructional Roles and Instructional Qualities of Learning Objects*
    In cooperation with H. Allert and W. Nejdl
    COSIGN 2002 - The 2nd International Conference on COMPUTATIONAL SEMIOTICS FOR GAMES AND NEW MEDIA. Augsburg (Germany), 2nd September - 4th September, 2002.
- Excerpts from chapter 3 were first published through
  - *Open Learning Repositories and Metadata Modeling*
    In cooperation with W. Nejdl, B. Wolf and M. Wolpers
    International Semantic Web Working Symposium (SWWS) July 30 - August 1, 2001 Stanford University, California, USA.
  - *Building up AI Resources as an AI Testbed*
    In cooperation with W. Nejdl and B.Wolf
    IJCAI'01 Workshop on Effective Interactive AI Resources August 5th, 2001, Edmonton, Canada.

- Excerpts from chapter 4 were first published through
  - *Instructional Models and Scenarios for an Open Learning Repository - Instructional Design and Metadata*
    In cooperation with H.Allert, T. Kunze, W. Nejdl and C. Richter
    E-Learn 2002: World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education (formerly the WebNet Conference). Montreal, Canada, October 15-19, 2002.
  - *Contextualized Models and Metadata for Learning Repositories*
    In cooperation with H. Allert, W. Nejdl and C. Richter
    In Online Education Using Learning Objects. By Rory McGreal, University of Athabasca, Canada, May 2004.
- Excerpts from chapter 6 were first published through
  - *Distributed Queries and Query Optimization in Schema-Based P2P-Systems*
    In cooperation with I.Brunkhorst, A. Kemper, W. Nejdl and C. Wiesner
    International Workshop On Databases, Information Systems and Peer-to-Peer Computing September 7-8, 2003 Humboldt University, Berlin, Germany Collocated with VLDB 2003.
  - *Processing and Optimization of Complex Queries in Schema-based P2P-Networks*
    In cooperation with A. Kemper, W. Nejdl and C. Wiesner
    Proceedings of the 2nd International Workshop On Databases, Information Systems and Peer-to-Peer Computing, Toronto, Canada, September 2004. In conjunction with the 30th International Conference on Very Large Data Bases.

Chapter 2

Metadata

2.1 Introduction

Why Metadata? Metadata are generally defined as data about data. This is a very broad definition which can be found in almost every document about metadata. The term “metadata” is interpreted differently in different communities. In content management and information retrieval metadata can be understood as ”any assertion about information resources”. In the context of Web search metadata is defined as “*machine understandable information for the web”* [9]. In the traditional library community metadata schemas and standards (e.g. MARC 21) are commonly used to describe both electronic and traditional resources.

Wether in the Web context or in the traditional context the main purpose of metadata is to facilitate the web search and enhance the quality of search results.

In our context (Web), the metadata usage is becoming an imperative. The Web is growing exponentially. It is impossible to determine the exact size of the web - The Inktomi WebMap announced on 18 January 2000 that the Web surpasses one billion unique indexable documents (found in [52]). Information search in this enormous hodgepodge of data on the Web is comparable to looking for a needle in a haystack. Many a speaks from ”the ill Web”. The Web is suffering from the lack of metadata infrastructure which was added several years (as from 1997) after the Web emergence. The following quote from Ted Nelson (the coiner of the term “hypertext”) at the Hypertext97 conference can be considered as the estimation of the hypertext community of the Web.

“The reaction of the hypertext research community to the World Wide Web is like finding out that you have a fully grown child. And it’s a delinquent.” [51]

This quote could reflect the opinion of the creator of the Web, Tim Berners-

Lee, who is working since 1998 [1] with many others especially within the W3C in order find remedies to the current Web and to build the tomorrow's Web, called *Semantic Web*. The Semantic Web is not a new Web, it just[2] extends the Web with a metadata infrastructure.

"*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*" – Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001.

Categorizing Metadata Based on the purpose and functions of metadata there are attempts to categorize metadata. For instance, the NISO (National Information Standards Organization, USA) [123] defines three types of metadata:

1. *Descriptive Metadata*: For search purposes (most used and known).
2. *Structural Metadata*: Describes the composition/relationships among objects.
3. *Administrative Metadata*: Provides information to help manage a resource (e.g. when a resource was created).

The Getty Standards and Digital Resource Management Program (USA) [8] defines in its "Metadata Introduction" [53] 5 metadata categories: *Administrative*, *Descriptive*, *Preservation*, *Technical*, and *Use*.

In the context of this thesis (learning repositories, learning metadata), we will focus on *descriptive* and *structure* metadata.

- *Descriptive Metadata*: We call them also *annotation metadata*. This type of metadata is used to describe learning objects (see Section 2.3). It includes elements such as title, keywords, author, etc.
- *Structure Metadata*: Provides information such as the composition of a learning object(e.g. how a course is built based on course units).

In the following, we will first present in Section 2.2 the metadata modeling languages RDF/RDFS and O-Telos and analyze their similarities and differences, based on our long experience in the field of metadata modeling. Then, we will introduce standardization in learning in Section 2.3. Thereby, we will present the well established Learning Object Metadata Standard (LOM), discuss the insufficiency of LOM in describing learning objects in the context of different instructional theories and propose an extension of LOM.

[1] Tim Berners-Lee wrote the Road map for the Semantic Web in 1998.

[2] "just" does not mean that the purpose is simple or easy to realize. It is rather a big challenge.

2.2 Metadata Modeling

2.2.1 RDF/RDFS

RDF

RDF [126], the Resource Description Framework, is the recommended standard of the World Wide Web Consortium (W3C) to model metadata about Web resources. RDF has a simple data model which is its fundamental syntax. This model is independent of any specific syntax serialization. The basic concept in RDF is the *RDF Graph Model*. The *RDF graph* is a collection of *triples* (statements) describing resources. An *RDF triple* consists of: *Subject*, *Object*, and *Predicate* (a.k.a the *property* of the triple).

Subject, *Object* and *Predicate* are resources identified by a *URI* (*Uniform Resource Identifier*). A *URI* is the more general form of the well-known Web identifier for Web pages, the so called *URL* (*Uniform Resource Locator*). The targets in an RDF Graph may be constant values (called *literals*) represented by character strings instead of resources. The following example is taken from a simplified version of the OLR2-schema (see Chapter 3). The lecture material of a lecture consists of lecture units. A lecture unit is composed of theory units. All units/elements are annotated (for instance title, description, etc.).

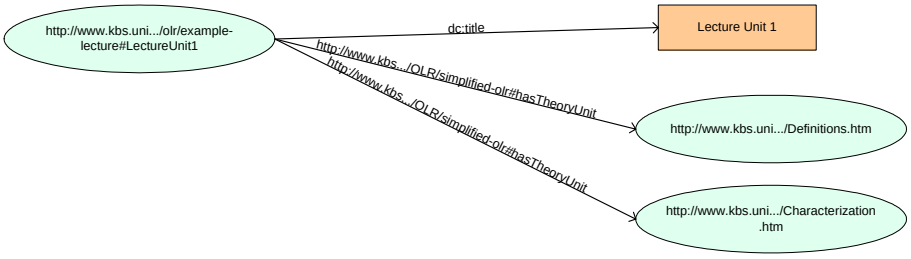


Figure 2.1: A Simple RDF Graph Describing “Lecture Unit 1”

The effectiveness of graphs for representing information is a matter of fact in the field of computer science. The *RDF Graph Model* allows many statements to be aggregated so that machine agents can apply the well-tested graph search techniques to find out the data. This representation however, is impracticable to exchange RDF descriptions. This is why RDF leverages the *eXtensible Markup Language* (XML) as a common syntax for the exchange and processing of metadata. The following example represents the XML-serialization of the RDF graph in Figure 2.1:

```
...
<rdf:Description ID="LectureUnit1">
  <s:title> Lecture Unit 1</s:title>
  <s:hasTheoryUnit rdf:resource="http://.../Definitions.htm" />
  <s:hasTheoryUnit rdf:resource="http://.../Characterization.htm" />
 </rdf:Description>
 ...
```

RDF Schema

RDF predicates (*rdf:property*) as presented above are not just attributes of resources (they would correspond in this case to attribute-value in relational databases for instance), they present rather relationships between resources. *RDF however, provides no mechanisms for describing these properties, nor does it provide any mechanisms for describing the relationships between these properties and other resources* [19]. For this purpose the W3C developed the *RDF vocabulary description language, RDF Schema* (RDFS for short). RDF Schema defines classes and properties that may be used to describe classes, properties and other resources. Each community (for instance Dublin Core [40], IMS [3]) defines and uses its own vocabulary to describe resources. RDFS does not stipulate semantics, i.e. vocabulary, for each community, but provides the mechanism that may be used to define this vocabulary. RDF's vocabulary description language, RDF Schema, is a semantic extension (as defined in [120]) of RDF.

At first glance, the RDF/RDFS concepts class (*rdfs:Class*) and property (*rdf:Property*) seem to be similar to those in object oriented programming languages. However, RDF/RDFS differs from the traditional object oriented design. It is namely property-centric. Properties in RDF are described using the *rdfs:domain* and *rdfs:range* mechanisms in terms of the classes to which they apply. *"For example, we could define the eg:author property to have a domain of eg:Document and a range of eg:Person, whereas a classical object oriented system might typically define a class eg:Book with an attribute called eg:author of type eg:Person."* [19] This property-centric characteristic of RDF/RDFS enables defining additional properties (attributes, vocabulary) without the need to re-define the corresponding class descriptions. Thus, anyone could define his own vocabulary to describe existing resources on the Web.

2.2.2 O-Telos

O-Telos is a deductive object-oriented conceptual modeling language very suitable for modeling and meta-modeling tasks. It has been implemented in the ConceptBase database system [75]. Its object-oriented constructs like object, class, metaclass, etc. are expressed using a frame syntax. Each frame declares an object

by stating its name, the classes it subclasses, the classes it instantiates, and the attributes it declares or instantiates. Frames are declared using predefined classes: *Individual* containing all individuals as instances, *Attribute* containing all attributes as instances, *Class* containing all classes as instances, *String*, *Integer*, etc. The use of the predefined classes is defined by a set of axioms to insure referential integrity, correct instantiation and inheritance. The following example is taken from a simplified version of the OLR2 schema. It is used to illustrate the O-Telos language: The lecture material of a course consists of course units, which group the specific elements. All units/elements can be annotated according to Dublin Core, i.e., they have a name and a description etc. The model of the above example declares the following O-Telos frames, which define the two classes course and course unit as well as a Dublin Core class, and the corresponding attributes:

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

The frame *Course* declares a class named *Course* consisting of arbitrarily many units. A unit is declared by the frame *CourseUnit*, and groups *TheoryUnits*, *Examples*, etc. Both are subclasses of *DC_Unit*, stating that they can have a title and a description, both of type String. The next frames declare the individuals, e.g. a course unit with the title “Lecture Unit 1”, the description “Introduction to Intelligent Agents”. This resource belongs to the course “Introduction to AI” which is an introductory course in Artificial Intelligence. Additionally, this resource belongs to another course “AI 2” which is an advanced course in Artificial Intelligence.

```
Individual IntroAILecture in Course with
  title
    t1 : "Introduction to AI"
  description
    d1 : "Introductory course in AI"
end Individual AdvancedAILecture in Course with
  title
    t1 : "AI 2"
  description
    d1 : "Advanced course in AI"
end Individual IntroAILectureUnit1 in CourseUnit with
```

```
  title
    t1 : "Lecture Unit 1"
  description
    d1 : "Introduction to Intelligent Agents"
  theory_unit
    tu1: "http://www.kbs.uni-hannover.de/.../Definitions.htm";
    tu2: "http://www.kbs.uni-hannover.de/.../Characterisation.htm"
  parent_course
    c1 : IntroAILecture;
    c2 : AdvancedAILecture
end
```

The frame *IntroAILectureUnit1* shows how the declared attributes *title*, *description*, *theory_unit* and *parent_course* are instantiated. The *theory_unit* and *parent_course* attributes show that O-Telos attributes usually are multi-valued. The frames are translated to sets of propositions which can be stored in a deductive database (for instance the ConceptBase database).

The definition of O-Telos propositions is a relation P(oid,x,l,y) with oid being the identifier, x being the *source*, l being the *label* and y being the *destination*. Consequently P(oid,x,l,y) states a relationship called l with ID oid from object x to object y. O-Telos defines specific interpretations for four predefined types of propositions:

1. *Object declaration* P(oid,oid,l,oid) declares an object named l.
2. *Instance relationship* P(oid,x,*instanceof,y) states that x is an instance of y.
3. *Inheritance relationship* P(oid,x,*isa,y) states that x is a specialization of y.
4. *Ordinary attributes* P(oid,x,l,y) says that x has an attribute named l with value y.

2.2.3 Comparing RDF/RDFS to O-Telos

Motivation

As noted above, RDF/RDFS is a simple but quite powerful modeling language to annotate WWW resources with semantical information. RDFS enables on the one hand the simple construction of conceptual models of sets of WWW resources, on the other hand it has been designed as a quite flexible representation language for these conceptual models. Unfortunately, the RDF Schema Specification [13] fails to give simple, yet formal explanations of RDFS concepts, which causes a lot of confusion when one really tries to use all RDFS possibilities. RDFS tries to be as self-expressible as possible, which leads to several properties playing dual

roles both as primitive constructs and as specific instances of RDF/RDFS properties (*rdfs:domain*, *rfds:range*, *rdfs:subClassOf*, and *rdf:type*), see also the detailed discussion in [108], where these properties are both defined in the RDF or RDFS-Schema and are used to define those schemas at the same time. Moreover, the self-expressibility of RDFS falls short of fulfilling its promise for meta-modeling, because of the constraints of the underlying triple model: Only a three level modeling hierarchy is possible (*rdfs:class*, specific classes as instances of *rdfs:class*, and instances of classes). Another drawback of RDFS is its poor support of the reification of statements. An object identifier must be assigned explicitly to each statement that is to be reified. This has to be done by adding explicit statements about the subject, predicate and object of the specific statement. Building on our previous work on modeling and meta-modeling in learning repositories (hyperbooks) [151], [107], we will compare RDF/RDFS modeling and annotation with the conceptual modeling language O-Telos, which has been strictly axiomatized in [74], based on the formalization of Telos (see e.g. [101]).

As for reification, O-Telos, being based on 4-tuples instead of triples, assigns a unique object identifier to each statement, which can be used to directly reference that statement. The comparison of RDF/RDFS with O-Telos and the discussion of possible mappings from RDF to O-Telos and back are useful for the metadata exchange between our O-Telos- and RDF-Hyperbook Systems. This will also shed light on some advantages and disadvantages of the design decisions of RDFS. In [103], we formalized an RDF variant, we call *O-Telos-RDF* based on the O-Telos model, which allows annotation in a way very similar to RDF, but extends RDFS with enhanced reification and meta-modeling capabilities.

Simple Mapping of RDF to O-Telos

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

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

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

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

The example shows that the *rdf:type* property is mapped to the O-Telos relationship *in* (*instanceof*). Other property declarations such as *dc:title*, dc:description, etc. are mapped to the respective O-Telos attributes. Both representations require the declarations of the objects/classes *Unit/olr_unit* and the course *AILecture*.

Enhancing the simple mapping (descriptions and aggregations)

A more in-depth examination of the RDF Model and Syntax Specification and our modeling of a given course shows that we can distinguish two types of general classes in RDF. The first type, we call *aggregation classes*, are classes whose instances group/aggregate other instances. An *aggregation class* is defined in RDF using the following statement:

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

These *aggregation classes* may include additional attributes for their aggregates. As shown in the above example these types of classes can directly mapped to O-Telos constructs. The second type of the general classes in RDF, we call *annotation classes*, are classes whose instances are assigned to web pages directly (see also the discussion in [108]). In RDF an *annotation class* is defined using the following statement:

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

These *annotation classes* define attributes to describe the assigned web pages. *Annotation classes* can be used in various RDF schemas to declare attributes on the same resource (referenced by its URI). Thus, annotation objects can be mapped to O-Telos constructs only if there is no other annotation object stating some attribute about the same resource. Because the O-Telos object takes the URI as its unique ID and all other attributes are referenced as above. In general this cannot be assured, as RDF, in contrast to (the frame syntax of) O-Telos, is a property centric language,

where properties about a given resource can be declared in different locations. To reflect this modularity, we need a different approach for mapping RDF annotation classes to O-Telos. As mentioned, resources which are described by *RDF annotation classes* have no ID property. They are just groupings of attributes, as the following example shows:

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

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

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

Looking at the example, we again realize the RDF property-centric approach, i.e. properties are the basic RDF constructs while classes etc. are just an add on to define *rdfs:domain* and rdfs:range constraints of these properties. The advantage of the property-centric approach is that properties can be assigned to web sites in a modular way. Furthermore it is semantically unimportant whether all properties are instantiated at once. As a result properties are always multi-valued , i.e. the expression (rdf:description about=“...”) for a specific web page can be used repeatedly in an RDF file (possibly in several RDF files!). A disadvantage of this modularity is of course that we cannot define single-valued attributes in RDF. For instance, it is not possible to define a property with a single value to represent the size of a resource. This, by the way, makes it difficult, if not impossible, to watch for violations of the single value property of *rdfs:range*. Several people

can define different (in this case inconsistent) RDF-Statements for the size of the resource which leads to inconsistent information about the resource. In contrast, although attributes are basically multi-valued in O-Telos, they can be constrained to be single valued by O-Telos constraints. Using the frame syntax of O-Telos, modularity like in RDF is not possible, as definitions and instances in O-Telos are class-centric and not property-centric. So, in O-Telos it is not possible to use e.g. "http:///Agents/Definitions.htm" as ID for two instances. In order to declare several O-Telos objects about the same resource, it is necessary to introduce an additional attribute "about" holding the URI of the resource. This additional attribute however, introduces an additional identifier which is not necessary in the tuple representation. Using this workaround, different O-Telos objects describing a resource have their own IDs as required by the O-Telos axioms, but can describe the same resource. A similar approach has to be used in XML Schema, by the way. The previous RDF example of the resource "http:///Agents/Definitions.htm" is declared in O-Telos by the following single frame:

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

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

```
Individual AgentDefinition1 in TheoryUnit with
  about
    a : "http:///Agents/Definitions.htm"
  language
    l : "German"
  coverage
    c : "Introductory course"
rights
    r : "KBS"
end Individual AgentDefinition2 in TheoryUnit with
  about
    a : "http:///Agents/Definitions.htm"
  parentUnit
    pu : LectureUnit1
end
```

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

Sequences and Reification in RDF and O-Telos

Let us look briefly at sequencing and reification in RDF and O-Telos. As an example we use the following RDF declaration of the resource LectureUnit1 which we will translate to O-Telos. LectureUnit1 defines a sequence for values of the olr:theoryUnit property. Its RDF declaration is given below:

```
 <rdf:RDF xml:lang="en"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:olr="http://.../rdf/olr_schema_5#"
   xmlns:dc="http://purl.org/dc/elements/1.0#">
   <rdf:Description ID="LectureUnit1">
     <rdf:type resource="http:///rdf/olr_schema_5#Unit"/>
     <dc:title>Lecture Unit 1</dc:title>
     <dc:description>Introduction to intelligent agents</dc:description>
     <olr:parentCourse rdf:resource=''#AILecture''/>
     <olr:theoryUnit>
       <rdf:Seq>
         <rdf:li rdf:resource="http://.../Agents/Definitions.htm"/>
         <rdf:li rdf:resource="http://.../Agents/Characterisation.htm"/>
         <rdf:li rdf:resource="http://.../Agenten/Structur.htm"/>
       </rdf:Seq>
     </olr:theoryUnit>
   </rdf:Description>
</rdf:RDF>
```

In this example, the order of resources of the property *olr:theoryUnit* is defined by the container object RDF sequence (*rdf:Seq*). This order is used for the visualization of the course hierarchy. While it is a convenient way to represent sequences, it is conceptually questionable, as *rdf:seq* is used as range of *olr:theoryUnit*, instead of the more explicit ranges describing the specific type of the child resource (like *theoryUnit*, or, for other properties, *example*, *slide*, etc.). O-Telos does not define such a construct for stating sequences, but represents sequences implicitly by the order of attribute statements in the O-Telos frames. Of course it is not insured that each implementation of O-Telos interprets the frames in the same way so that the attribute order (the sequence) might vary from one implementation to another. If we want to state our RDF example without using RDF sequence, but still represent sequences, we could use an attribute ordinal for the RDF-statements representing the sequence of the property values. These statements then look like: (oid ,ordinal,i), with i:integer and oid:ID is the ID of a statement (s,p,o) with s:subject,

p:predicate and o:object. In other words, we need the possibility to make statements about statements, e.g. by referring to the IDs of statements in statements. Unfortunately, RDF statements do not have IDs. Instead we have to introduce higher-order statements which are a special kind of statements about statements: (s,p,o,t) with s:subject, p:predicate, o:object and t:type. Applied to our example this could be written as follows:

```
<olr:Unit rdf:ID="LectureUnit1"/> <rdf:Description>
  <rdf:subject resource="#LectureUnit1" />
  <rdf:predicate resource="http://.../#theoryUnit" />
  <rdf:object rdf:resource="http://.../Agents/Definitions.htm"/>
  <rdf:type resource="http://.../22-rdf-syntax-ns#Statement"/>
  <olr:ordinalNo>1</olr:ordinalNo>
</rdf:Description> <rdf:Description>
  <rdf:subject resource="#LectureUnit1" />
  <rdf:predicate resource="http://.../#theoryUnit" />
  <rdf:object rdf:resource="http://.../Agents/Characterisation.htm"/>
  <rdf:type resource="http://.../22-rdf-syntax-ns#Statement"/>
  <olr:ordinalNo>2</olr:ordinalNo>
</rdf:Description>
```

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

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

In [106] we show how to use this idea in an extended variant of RDF (O-Telos-RDF), which easily allows reifications of arbitrary statements by referencing statement IDs. Of course, introducing unique ids for property statements in RDF is not possible globally. Still, locally at one site, this is possible, and the site prefix can make these ids unique worldwide (which is the approach we propose in [106]).

2.3 Metadata Standards for E-Learning Applications

2.3.1 Metadata Standards

We distinguish two major metadata standards being used in the e-Learning realm: the Dublin Core (DC) and Learning Object Metadata (LOM) standards. Dublin Core has been developed for general purposes and not specially for e-Learning content, but it is used in a wide range of education contexts for resource discovery. LOM is a much more complex standard that has been developed specifically for the e-learning context. Both LOM and DCMI have been leaders in the formation of metadata specification for the Web. The Learning Technology Standards Committee Learning Objects Metadata (LTSC-LOM) and the Dublin Core Metadata Initiative (DCMI) announced on December 2000 their joint commitment to develop interoperable metadata for learning, education and training. This cooperation aim at having a common approach to educational metadata, *"which is crucial to further speed up adoption of metadata technologies. That in turn is the first, crucial step on the long road to open learning infrastructures"* [12].

DC The Dublin Core (DC) metadata standard developed by the Dublin Core Metadata Initiative (DCMI) [40] is a simple element set for describing resources (electronic documents or a "real" physical objects). The Dublin Core Metadata Element Set (DCMES) [11] was the first metadata standard deliverable out of the DCMI. DCMES provides a semantic vocabulary for describing the "core" information properties, such as "Title" and "Creator" and "Description". It includes fifteen elements. We distinguish two level in the DC metadata standard: Simple and Qualified. The "Simple Dublin Core" uses no qualifiers. Its elements are simple attribute-value pairs without any "qualifiers" (such as encoding schemes, enumerated lists of values, or other processing clues) to provide more detailed information about a resource. The "Qualified Dublin Core" employs additional qualifiers to further refine the meaning of a resource. *"One use for such qualifiers are to indicate if a metadata value is a compound or structured value, rather than just a string"* [16]. For further detailed information on DC and the use of qualifiers see the DC usage guide [16].

DCMES can be represented in many syntax formats. In the context of this thesis, the encoding for the DCMES in the Extensible Markup Language (XML) [10] using simple RDF is the most interesting. The basic idea is that DC Elements correspond to RDF properties. The details of expressing the full DC (both simple and qualified) in RDF/XML can be found in [85] [39].

LOM The Learning Object Metadata standard (LOM for short) [2] defines a structure for interoperable descriptions of learning objects. It aims at facilitating search, management and (re)use of learning objects by authors of online-courses, teachers and learners. Learning Objects (LOs), oftentimes called "reusable learning objects (RLOs)", are defined by many authors as *"digital entities deliverable over the internet"* [150]. In the LOM draft standard[3] however, a learning object is defined as "*an entity, digital or non-digital, that may be used for learning, education or training.*" [15]. In other words, learning objects are content units that can be re-assembled (thus reused) to create new courses possibly in different learning contexts and based on different instructional theories.

LOM has a strong relation to the already presented Dublin Core. The mapping of the 15 DC data elements to some LOM data elements, which can be found on the appendix B of the draft standard [15], proves that LOM somehow includes DC.

The LOM basic schema consists of nine categories as described in the draft standard for learning object metadata [15]:

- The *General* category groups the general information that describes the learning object as a whole.
- The *Lifecycle* category groups the features related to the history and current state of a learning object and those who have affected this learning object during its evolution.
- The *Meta-metadata* category groups information about the metadata itself.
- The *Technical* category groups the technical requirements and technical characteristics of the learning object.
- The *Educational* category groups the educational and pedagogic characteristics of the learning object.
- The *Rights* category groups the intellectual property rights and conditions of use for the learning object.
- The *Relation* category groups features that define the relationship between the learning object and other related learning objects.
- 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.

[3] IEEE approved in June 2002 the first version of the LOM standard.

- The *Classification* category describes this learning object in relation to a particular classification system.

Each category is a grouping of data elements (attributes) describing a learning object. For instance, the *General* category includes data elements such as title, description and keywords. For some datatypes the draft standard defined Vocabularies, i.e., a recommended list of appropriate values. In the *Educational* category we find, for instance, the data element *Learning Resource Type* which may have as value *exercise* or *lecture* or *exam*, etc., defined in the value space for this data element.

2.3.2 Standardization in Learning

In this section, we discuss the context in which standardization in Learning takes place. Standardization has to face a context of diversity. A wide variety of diverse instructional models, learning theories, instructional principles, and paradigms guide the design of learning environments both explicitly and implicitly. Standardization should also address controversial goals and assumptions on learning. We will state some contrasts:

Effective teaching as well as theory of change Several initiatives focus on effective learning as SCORM (ADL) does: *As new instructional technologies emerge, they provide opportunities for universally accessible and effective life-long learning* [17]. But Janneck states in [71] controversial trends in initiatives of improving teaching and learning: Whereas public and politics postulate more effective learning, discourses in educational science focus on qualitative change in learning culture . Learner centered approaches exist in parallel to instructionalist and teacher centered approaches. Distribution and teaching of knowledge exist in parallel to facilitating collaborative co-construction of knowledge and peer-tutoring (e.g. in CSCL).

Learning objective: Knowledge as well as competencies Most computer based learning environments are restricted to the teaching of knowledge. They merely focus on organizing and structuring units of information (knowledge objects)and on the ”right” curriculum and life-long learning on-demand. Developing new tools and learning environments, however, might also enable learners to acquire new skills and social competencies [72]. While some learning environments imply learners who are self-organized, others aim at imparting the skill of self-organization.

Valuable diversity Standardization has to address any scenario based on diverse requirements and assumptions on learning. Therefore, the context for Standardization in Learning is well characterized by Lyotards comprehension of science which is explained by Beck in [29]: *There is need to emphasize in a postmodern manner the conflicting diversity of models, the competition of paradigms, and the impossibility of integrative and finally valid solutions. The failure of integrating theories is specified a characteristics of postmodernism*. Standards that aim at instructional neutrality must fail from the point of view of the science of philosophy. They risk to address a narrowed perspective on learning. Also the formation of a pedagogical meta-model [86] is not what we intend. We want to open the view on learning in standardization. and propose an approach of Instructional Roles in Learning Metadata Standards, which supports the idea of explicitly modeling and annotating different paradigms, models, and principles in learning.

2.4 What Is Missing in LOM? Instructional Roles of Learning Objects

2.4.1 Motivation

To allow the reuse of learning objects, various standards have been developed that describe learning objects, their relationships, etc. The IEEE LOM draft standard for learning object metadata [15] specifies a variety of bibliographic and technical properties of learning objects, as well as different relationships between learning objects. It makes the search and reuse of learning objects based on these metadata possible. However, even though the LOM draft includes an educational category, the standard does not specify, which instructional roles are available for or can be played by a learning object within a course. As curriculum programs do, LOM concentrates on what should be taught and when, rather than how. Obviously, a standard for learning objects metadata should not specify how to teach, but it should definitively be able to provide information about how to specify pedagogical aspects of learning objects.

A recent paper by Schulmeister emphasized this point: “While potential students of distance learning courses can search for price, author or subject of courses, they cannot search for certain criteria which may be as important: For instance, how can one tell whether Law school students of the Cyberversity of European Law are advised better than the students of Capella University, or whether one can participate in independent work groups at Athabasca University which are not available at ESC Pau. All of these are still open issues” [132]. Schulmeister claims that students can not elect on the basis of standards like AICC, SCORM, and IMS,

as instructional principles of online courses are not addressed so far. We add some questions students might have: “Which learning processes are supported?”, “Is communication among learners embedded in the program?”, “Is computer supported collaboration included?”, “Does it meet my preferred learning style?”.

This difficulty is caused by the fact, that LOM attributes specify properties only at a very basic abstraction level. LOM specifies annotations for content and activities (Learning Resource Type Vocabulary: e.g. Exercise, Simulation, Questionnaire, Figure, Table, Narrative Text). LOM does not support metadata about instructional models and instructional theory, even though authors are implicitly or explicitly using specific instructional theories. Moreover, it does not support information about the use of learning objects in learning processes, which are a central concern in instructional design.

Specifying author and title is of course easier than specifying instructional information, but the question, whether LOM can be extended to implement the specification of instructional metadata related to instructional models and instructional theory, is an urgent one for a standard defining metadata for learning objects. Our approach extends previous work which has tried to extend LOM with didactic metadata. Meder established a detailed ontology for instructional design (“Didaktische Ontologie” [96]), but he only differentiated existing LOM attributes and introduced additional types and corresponding vocabulary to specify the types of learning objects (detailed KnowledgeTypes, types of CommunicationsMedia, Transaction/Assigment, and CommunicativeContribution) and of hierarchical and associative relations linking these learning objects (MatterOfFactRelations are subdivided into HierarchyRelations and RefersRelations). This additional vocabulary is highly structured. But authors must be familiar with the use of this vocabulary in different educational contexts. No support is given based on the corresponding instructional theories or learning processes. We extend this and similar approaches, by introducing an additional abstraction layer to LOM which explicitly takes different instructional theories into account.

We also investigate which additional specifications for learning object metadata related to instructional criteria are useful, and how these metadata can be specified and grouped, based on the corresponding instructional theories. In the following, we will give an introduction to the current LOM standard (IEEE LOM Working Draft 6.1 [15], 2001). Then, we will discuss the abstraction levels of pedagogical dimensions. Finally, we will introduce a concept of instructional roles in modeling.

2.4.2 Current LOM Model

As described above, the LOM basic schema consists of nine categories. Each category is a grouping of data elements describing a learning object. This basic schema

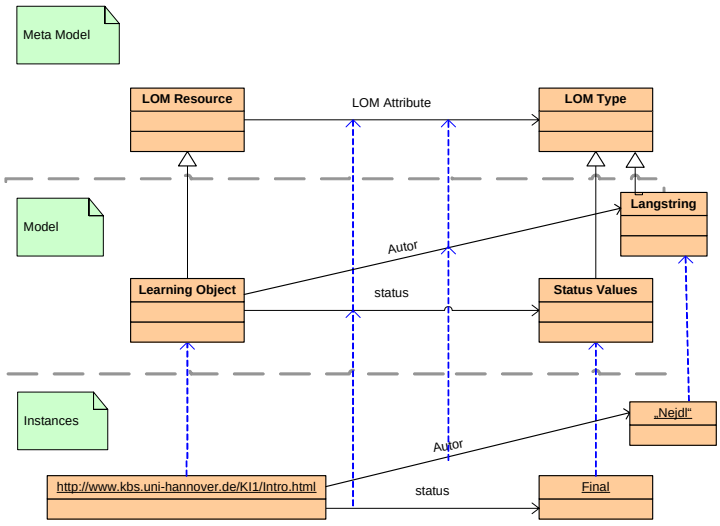


Figure 2.2: Current LOM Model

as described in the specification does not show the common meta-model of LOM. Based on the Draft Standard for Learning Object Metadata (LOM for short) [15], we have modelled the LOM basic schema using the Unified Modeling Language (UML).

As shown in Figure 2.2, the LOM Meta-Model simply consists of two types/classes: the LOM resource and the LOM type, linked by LOM attributes (LOM data elements). In the model layer, we find only the learning object, the attributes describing the learning object and the datatypes for those attributes. In the LOM specification two types of data are defined:

- *Langstring*, which represents one or more character strings.
- The second type is just a set of values for a given LOM data element (attribute), called *value space*. For instance the data element named *structure* in the category *General*, underlying the organizational structure of a given learning object has 8 values in its value space (*Collection, Mixed, Linear, Hierarchical, Networked, Branched, Parcelled, Atomic*). This corresponds to the usual enumeration types present in many programming languages.

The descriptions of LOM are context-independent and static classifications. This approach is nor appropriate for many didactic aspects: For example to annotate collaborative learning the type resp. the vocabulary “collaborative” can hardly be added to a single category. “Collaborative learning” is an instructional principle which affects and shifts the entire environment: The role of teacher and learner (Intended End User Role (LOM 5.5)), activities, Interactivity Type (LOM

5.1), Context (LOM5.6), Typical Learning Time (LOM 5.9), purpose, organizational framework, and many more. The current LOM model does neither provide concepts for modeling instructional models, instructional principles, nor specify epistemological approaches .

2.4.3 Pedagogical Background

We claim that LOM does not integrate pedagogical dimensions. That's why we have to first define what we mean by pedagogical dimensions. Thereby, we present a top-down-model in which pedagogical dimensions are embedded in different layers of abstraction - according to educational and cognitive sciences. In this model, LOM only addresses the bottom layer which is the most basic.

Pedagogical Dimensions - Abstraction Layers

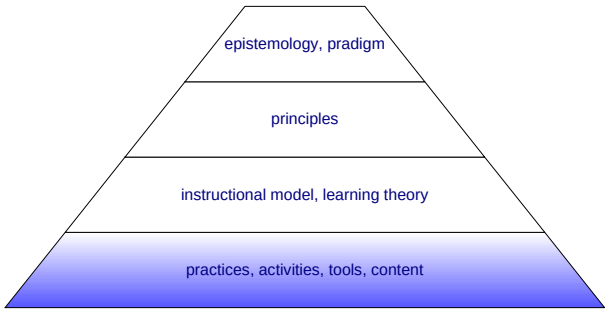


Figure 2.3: Top-Down-Model

4th (highest) Layer of Abstraction: Epistemology, Paradigm The highest level of abstraction addresses, either implicitly or explicitly, the broad orientation concerning epistemology resp. theory of cognition. This layer is often referred to as a paradigm or as a way of teaching, learning, thinking and designing. Behaviorism, cognitivism, constructivism are major approaches. Papert for example shows two main approaches, which contrasts learning with teaching: He distinguishes constructionism from instructionism [119]. These terms also address this layer of abstraction. In practice instructional design is often a mixture of different paradigms.

3rd Layer of Abstraction: (Instructional) Principles One or more instructional principles can be derived from epistemology. Merrill refers to this layer as "a set of underlying principles". Examples are:

- Anchoring new concepts into the learner's already existing cognitive structure, which will make the new concepts recallable [28].
- Problem solving, which will make knowledge transferable.

In the literature as well as in practice, we often find fixed terms which include some well-agreed principles (e.g. Problem Based Learning (PBL), Situated Learning, Communities of Practice (CoP), Case Based Learning, Collaborative Learning).

2nd Layer of Abstraction: Instructional Models, Learning Theories According to Merrill, "principles are implemented by a program" and, "a program is based on principles" [98]. Instructional models and theories, as well as communication theories, are guidelines or sets of strategies. Models often structure learning processes into several phases, make learning cycles explicit, and organize learning processes in a specific way. Learning processes are also not addressed by LOM.

1st Layer of Abstraction: Content, Practices, Activities Content, practices, sets of activities, scenarios, and curriculum programs that assemble content located at this layer. This is about what is actually done and to be learned as well as which resources are actually used. This the only layer addressed by the LOM Metadata Schema.

Discussion of the Top-Down-model

A lack of shared terminology in the domain of instructional design and educational science can be stated. The term "theory" for example is used at different levels. Some learning theories are worked out as conceptual frameworks and are close to the highest level of abstraction [28] [83], other theories are nearby the most concrete layer and therefore tend to be models (McCarthy 1996, 4-MAT [98]). Pedagogy as well as instructional design are both ill-structured domains. Historical as well as cultural background is relevant in forming terminology as well.

The described top-down-model shows that any decision that is made at a higher level of abstraction affects any other more basic level. Choosing the PBL principles results in shifting for instance learning cycle, roles and actual activities. Epistemological approach, instructional principles, as well as learning processes and phases are not addressed by LOM. When including pedagogical dimensions, we must provide for continuous change, trends, different cultural backgrounds, ongoing social development, educational traditions, and even individual believes in a special approach. So, is it feasible to fit pedagogical dimensions into standards?

The presented top-down approach follows the German tradition of teaching as a reflective practice [67]. Klafki in 1985 re-innovates and reflects the term "Bildung"

which was central to Wilhelm von Humboldt's Theory of Bildung in the epoch 1770 to 1830, the late Enlightenment ("'Bildung' as a central concept in pedagogic reflection" [83]. Klafki emphasizes the significance of classical theories of Bildung for a contemporary concept of Allgemeinbildung [84].

The top-down-model presented above is derived from the German tradition of education. But there is an immense need for implementing a model in LOM that reflects both, American as well as European, and hopefully also other traditions in the field of pedagogy and instructional design. American education is influenced by curriculum theory and curriculum traditions and is stamped by a different cultural background [66]. Curriculum theory concentrates on understanding the overall educational significance of the curriculum.

However analogies between different traditions can be stated: Merrill in "First Principles of Instruction" presents various instructional theories and underlying principles. His approach seems to be comparable to our perspective which is presented by the German authors (Klafki, Kösel). *"A practice is a specific instructional activity. A program is an approach consisting of a set of prescribed practices. A principle is a relationship that is always true under appropriate conditions regardless of program or practice. Practices always implement or fail to implement underlying principles whether they are specified or not. Instructional approaches may facilitate the implementation of one or more instructional principles.*" [98].

The Meta-Model we present takes into account, not only the most basic level, but also higher level of learning models and theories. It faces the need for dynamic classification and includes the role-concept.

2.4.4 Extended LOM Instructional Roles in LOM

Using LOM metadata we cannot specify the instructional aptitude of a learning object. Is a learning object suitable to be used in a scenario of collaborative learning or in a scenario enabling problem solving?

Metadata should be useable to specify instructional aptitude at any level of abstraction as explained in the previous chapter: models, theories, principles, and even epistemology. We will demonstrate and substantiate this by two use-cases, elaborating the level of instructional models and learning theories in the following.

Learning Sequences - Learning Processes

In separating content from structure, learning objects are decontextualized. In order to advice the recontextualization of content for learning, learning objects should be integrated in learning strategies, learning processes or sequences (such as case

studies). We already stated that learning processes are not addressed by LOM. In our meta-model, learning objects are integrated into different learning cycles supporting processes which are derived from different instructional models. In the role-concept of our meta-model, learning theories and instructional models represent context. Instructional models define instructional phases within a learning cycle. Merrill stresses the importance of phases in learning cycles and states common instructional phases in PBL [98]: *Many instructional models design environments which involve students in distinct phases of learning. Each model determines a set of specific phases. Each phase is part of a learning cycle and involves important, often implicit components of effective instruction. Many current instructional models suggest that the most effective learning environments are those that are problem-based and involve the students in four distinct phases learning:*

1. *Activation of prior experience,*
2. *Demonstration of skills,*
3. *Application of skills,*
4. *Integration of these skills into real-world activities.*

In our meta-model any phase of learning represents a specific instructional role. Learning objects (types) fill different instructional roles within learning processes or learning cycles, which are set up by different learning theories or instructional models. One and the same learning object may fill different instructional roles defined by different instructional models and learning theories or derived from various instructional principles.

Our concept of roles stringently and clearly distinguishes the natural types of learning objects (media type, tools e.g.) from their instructional role/purpose [58]. The Teachware-specific Meta-Model in Learning Material Markup Language LMML [139] defines Motivation as well as Example, Exercise, Question, Table, List, Multimedia and others as instances of ContentObject. But from the perspective of instructional design Table, List and Multimedia elements are media types or different types of illustration which are contained in the curriculum. These types fill, for instance, the role Example or Motivation. Categories of LOM do not address a main task of instructional design, namely the support of learning processes respectively cycles of learning. There are different ways for modeling learning sequences. The selection of a learning sequence is based on instructional principles and is epistemology focused. Learning theories and instructional models suggest to involve the students in distinct phases of learning. The top-down-model can be mapped to the concept of types (class) and roles as shown in Figure 2.4.

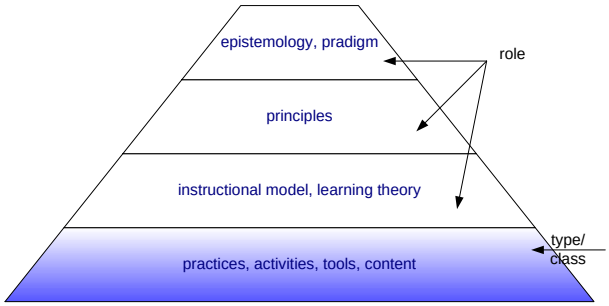


Figure 2.4: The Model Distinguishes Roles from Type/Class

Roles for Modeling

In the following, we will specify the role concept and the underlying reasons for using it. Learning theories serve as context. They design learning processes and strategies, which are structured along phases and in learning cycles. In each phase/learning cycle learning objects fill different roles. The same learning object may fill different instructional roles defined by different instructional models and learning theories.

What is a Role? There is a wide choice and diversity of definitions of the role concept in literature. We will not examine more closely the different meanings and uses of the role concept. In [134] there is an elaboration of this concept. We will focus more on the relevant definitions for our purpose. Role is defined as follows in the Encyclopedia Britannica *" A role is a comprehensive pattern of behavior that is socially recognized, providing a means of identifying and placing individuals in a societyRole expectations include both actions and qualities: a teacher may be expected not only to deliver lectures, assign homework, and prepare examinations but also to be dedicated, concerned, honest, and responsible. Individuals usually occupy several positions, which may or may not be compatible with one another: one person may be husband, father, artist, and patient, with each role entailing certain obligations, duties, privileges, and rights vis--vis other persons."* [18]

The notion of role stems originally from the theater. Steimann distinguishes the specification of the characteristics of a role-player from the player himself *"a role is a kind of protocol specification, that determines the behavior and characteristics of a role player, but not the role player himself."* [134].

Roles can be taken dynamically in contrast to types/classes. Types/classes, which are the fundamental concepts in the object-oriented modeling, are static

structures. That is, an instance of a class once and forever belongs to the that class - it can not change without loosing its identity. An HTML-page may fill different roles in the same instructional model or in two different instructional models.

Instructional Roles Let's map these definitions to the educational jargon, the individual can be mapped to a given learning object and the society to an epistemological approach. A learning object may play different roles within different instructional models and even within the same instructional model. For instance, the role of a given learning object provides a means of identifying and placing learning objects in a learning sequence. The role to recontextualize the content for learning and to place the learning objects in the right place in the learning process.

This distinction between class and role is only on the semantical level. On the syntax level we do not differentiate between these concepts. In fact, in the practice of object-oriented modeling class is used for both of them.

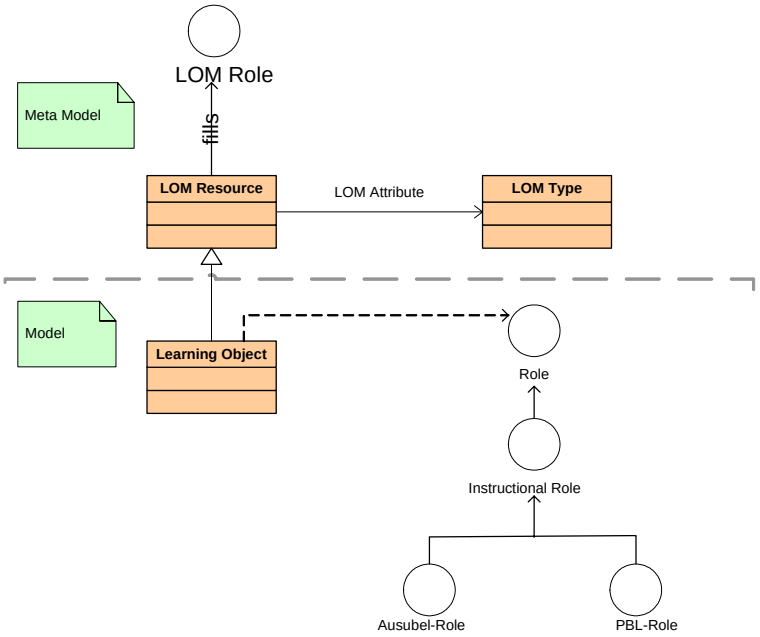


Figure 2.5: Extended LOM Model

Figure 2.5 shows the LOM Meta-Model extended with the role concept. We defined a new class "Instructional Role" as a subclass of the general class "Role". We will focus on two instructional models/principles: The Ausubel instructional

model and the PBL instructional model. Thus, we defined two subclasses “Roles according to Ausubel” and “Roles according to PBL” for the roles according to the two instructional models. Instructional roles according to Ausubel and to PBL are implemented in the Open Learning Repository OLR3 and described in detail in chapter 4For each further instructional model we can define new subclasses of the class “Instructional Role”. Learning models as well as instructional principles and paradigms are used as context of roles. In fact, metadata sets according to specific roles are to be set up and agreed upon by communities of practice and scientific communities in regarding fields. Standardization initiatives may moderate these processes. These processes will not be easy as standardization itself is hard work. But in advance, LOM could be reduced to less attributes. We suggest to remove the Category “Educational” and some others and address instructional and educational information at the level of specific roles. Models carefully describe the instructional function of each phase within a learning process. We refer to phase plus specified function as instructional role. Used in such a way, the instructional role is compatible with the idea of the role concept: a type must have certain characterizing predicates [58], qualities, attribute, or requirements in order to be able to fill a certain role. Characterizing predicates, attributes and requirements are matchable with the concept of instructional aptitude explained in the beginning of this chapter. We suggest to name instructional aptitudes “instructional qualities”.

2.4.5 Summary

Motivated by the lack of instructional information in the current LOM standard, we presented a short analysis of this deficiency and showed how the concept of instructional (didactic) roles can be used to extend the current LOM standard to include this missing information. The important advantage of this approach, in contrast to a standard class-oriented approach[4], is its ability to deal with dynamic modeling and instantiation. Integrating Instructional Roles in standards must ground on broader and agreed sets of attributes which address different instructional principles, learning theories, and paradigms. In Chapter 4 we will present two use cases for better understanding of the concept of Instructional Roles. These use-cases implement the instructional principle Expository Teaching and the Problem Based Learning by specific models.

[4]The traditional class-oriented approach is suitable for the static attributes currently used in LOM

Chapter 3

OLR2

3.1 Introduction

What is a Repository? The term is from the Latin repositorium, a vessel or chamber in which things can be placed, and it can mean a place where things are collected [6]. In the field of computer science, repositories are more than databases. They are built on top of a (or several) database(s) and offer additionally a set of application services. In [129] repositories are defined as *systems that handle metadata... serve as catalogues for data and application models*.

Open Learning Repositories Our open learning repositories (OLRs for short) hold collections of learning metadata, that is metadata about learning objects. They aim at providing mechanisms to store, discover, and exchange learning metadata. Moreover, our open learning repositories serve as testbed for the investigated issues on learning metadata, learning metadata standards, and learning metadata management. As Figure 3.1 shows, there are three basic pillars in our framework:

- *Learning*: This includes instructional theories and models, learning standardization, etc.
- *Metadata*: Metadata modeling languages and standards.
- *Repository*: This represents the used technologies such as the database management system, the application server, etc.

The principle of openness is related to these three parts of our framework. In the context of learning, the learning metadata standards should be open to any instructional theory or model, as discussed in Chapter 2. As we will show later (see Chapter 4), the OLR metadata schema has also to be open to any teaching/learning

strategy, learning materials, etc. The openness in the metadata context consists at using open web standards such as RDF/RDFS and open metadata standards such as DC and LOM. We aim also at using only open source technologies and standards, e.g. Apache Torque, Enhydra, PostgreSQL. The openness related to the repository part of our framework is realized in our youngest prototype (see Chapter 5).

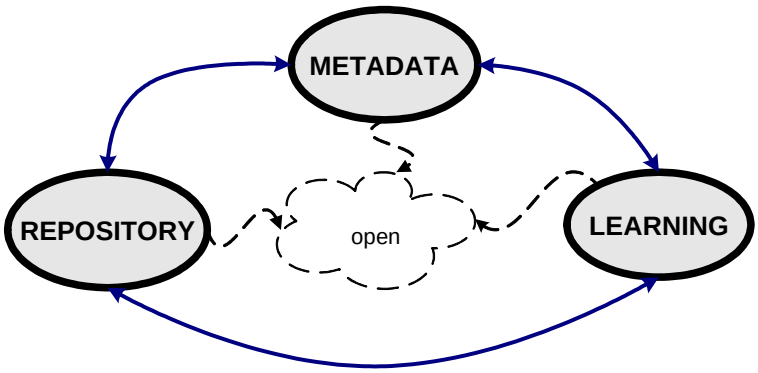


Figure 3.1: Open Learning Metadata Repository

The Ideas behind Building repositories for e-learning is an iterative process, since course content and course structure are always changing. We realized the necessity to separate content from structure of a given course during the conception of our first e-learning repository, which we called *KBS-Hyperbook*, several years ago at our institute. This system has been built around a conceptual model for structure and contents of the domain, which is expressed in the *O-Telos* conceptual modelling language. Since mid 2001, we have been working on a new generation of e-learning repositories, we called *open learning repositories*.

Our open learning repositories aim at metadata-based course portals, which structure and connect modularized course materials over the Web. The modular content can be distributed anywhere on the internet, and is integrated by explicit metadata information, in order to build courses and connected sets of learning materials. Modules[1] can be reused for other courses and in other contexts, leading to a course portal which integrates modules from different sources and authors. Semantic annotation is necessary for authors to help them choose modules and to connect these modules into course structures. We use a relational database to store all metadata, but store no content in the database itself. The stored metadata represent information about the structure and the access paths within a particular course,

[1]These modules can be considered as reusable learning objects (RLO). However, we do not deal with the problem of defining learning objects. This problem goes beyond the scope of this work.

the URLs as identifiers for single elements (modules, courslets, course units, subunits, etc.), and other useful metadata about the content itself (e.g. Dublin Core or IEEE LOM metadata).

In the rest of this chapter, we will focus on the first generation of open learning metadata repositories, OLR2. Thereby, we will first describe the technologies used in OLR2 and present storing RDF metadata in a relational database. Then, we will present the different web interfaces for browsing and manipulating the metadata. Finally, a scenario-based evaluation of OLR2 is presented. The second and third generation will be respectively presented in Chapter 4 and Chapter 5.

3.2 Technologies and Architecture

The OLR2 repository[2] can store RDF (metadata) from arbitrary RDF schemas. However, we have chosen not to implement a one-size-fits-all approach, and followed a customizable approach, implementing different interfaces together with their schemas and metadata for different courses using a common infrastructure. Initial loading of a specific course is done by importing an RDF metadata file (using XML syntax) based on this course's RDFS schema [41]. Our Artificial Intelligence course prototype uses a simple schema describing the course structure (units, subunits, elements, and arbitrary links between these elements) and simple cataloguing of its elements using the Dublin Core metadata set.

The web interface for navigating the course follows a multi-view approach. A user visiting the course has a choice between three different navigation schemes. The first one is a hierarchical tree-like navigation directly reflecting the course structure stored in the database. A visitor may open and close units and subunits to display the elements/pages of the logical document. The second view provides a trail navigation where the user has the possibility to move forward and backward on a trail. Third, we experimented with a semantic net or context net navigation. In this approach, the user can view units in different contexts, navigation is implemented as a kind of fish-eye where the current unit located in the center surrounded by related units and contexts. All navigation elements are created dynamically on demand. In addition to displaying course content, OLR2 provides different ways of reviewing the stored course metadata. Either the system displays metadata in a nicely formatted way suitable for a human reader (see Figure 3.3, or it generates the corresponding RDF source in XML notation. For content developers, we implemented an enhanced Web interface which allows the developer to manipulate

[2] OLR2 does not indicate a second generation of OLRs. We experimented in the early stages with storing RDF in a relational database and called the resulting technical infrastructure OLR1. OLR1 is not considered as the first generation of OLRs.

metadata through HTML forms. The OLR2 system translates all user input into suitable SQL update and insert the resulting statements into the database. In doing so, the user has not to understand complex XML/RDF notation. For the purpose of evaluation, the system tracks all user behavior in the database, including which course elements are accessed and when, which updates are made and by whom. This information can be used to evaluate different navigation schemes and different types of course units.

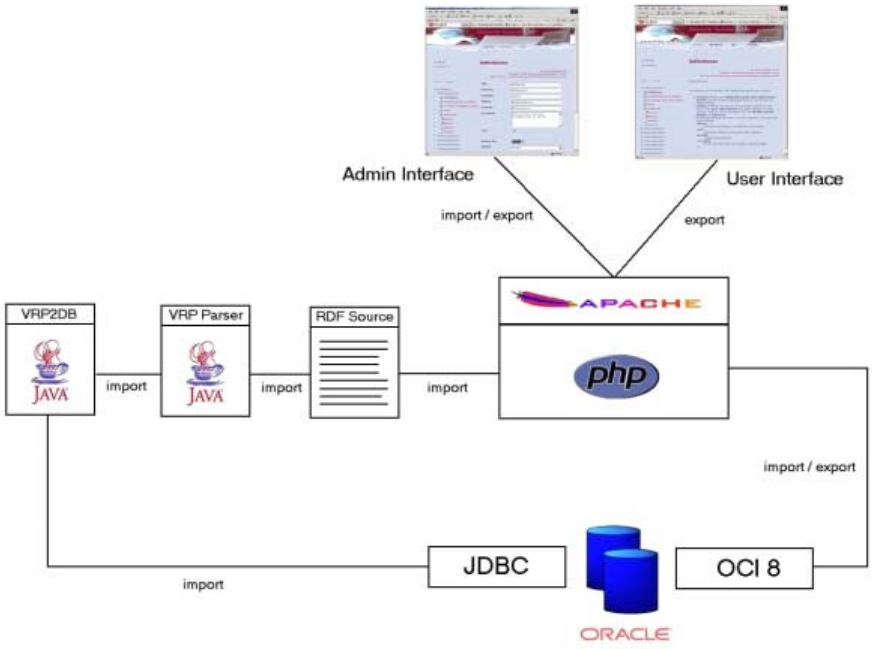


Figure 3.2: OLR2 Architecture

3.2.1 OLR2 Architecture

The OLR2 architecture is shown in Figure 3.2. The system is based on a 3-tier architecture. As front end any state-of-the-art web browser may be used. The mid-tier is a combination of Apache Web server and PHP4 module. The back-end holds an Oracle-8i database and can physically be on the same machine as the one running the Web server. Whenever the user selects a link or button, Apache delegates the client request to the PHP module which the executes the appropriate PHP script. In most cases this script will need to interact with the database, since it stores all RDF metadata. For communication with Oracle, PHP uses its built-in

OCI8-interface. The PHP script evaluates the data returned by Oracle and dynamically creates an HTML page, which in turn is sent back to the client browser. In addition, the web interface allows the upload of raw RDF source code (XML syntax) to be stored in temporary files within the server's file system. A shell script runs the VRP parser [87] against these RDF metadata. The generated triples build then the input of a Java application using the JDBC interface, which imports all statements into the database.

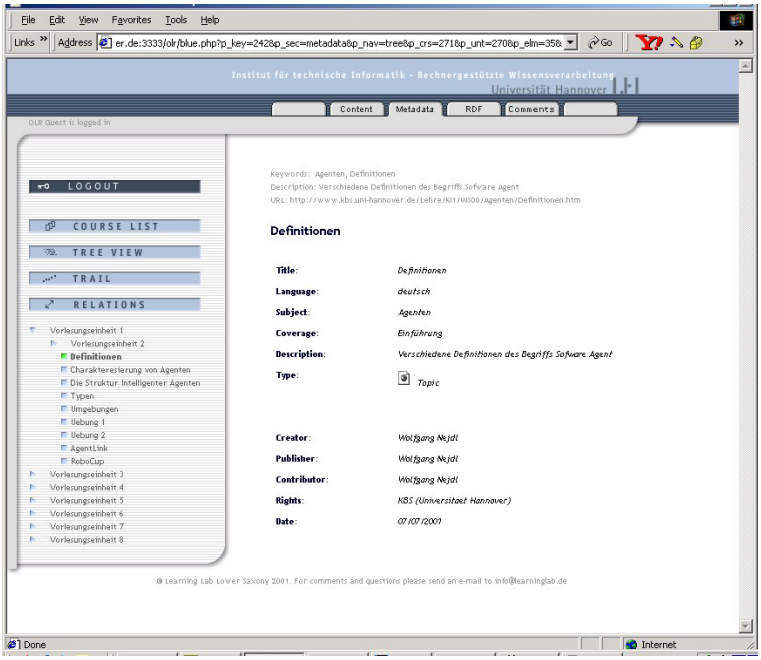


Figure 3.3: Display of Metadata for a Specific Resource

3.2.2 RDF Annotation

The OLR2 system stores virtually all courses' metadata as RDF metadata. In web based learning and teaching, the trend is to encode learning materials with meaningful and machine understandable metadata in order to facilitate modular and reusable content repositories. One of the practical uses of RDF, as it has been described by W3C, is in Web sitemaps. "*The RDF schema specification provides a mechanism for defining the vocabulary needed for this kind of application*" [14]. Thus, with RDF, we can describe in our application, how modules, course units, courslets are related to each other or which examples or exercises belong to a course unit, RDF metadata used in this way are called *structural* or *relational meta-*

data. Another practical use of RDF is the description of web pages/units, which is mandatory to build a course based on modular content, distributed over different sites. To standardize these kinds of descriptions, initiatives like IMS and IEEE LOM specify schemas suitable for learning objects. We have been involved as well in the German LOM version as in a LOM-RDF-binding [4]. The primary target of RDF is to provide a standardized way of creating and using such specialized metadata schemas to describe resources on the Web. Some of the goals the W3C aims to reach using RDF are:

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

As we described in Chapter 2, everything in RDF is expressed through statements, which are triples consisting of subject, predicate and object (corresponding to instantiated binary predicates). Expressing the sentence ”Smith is the author of the HTML document that can be found at the URL “http://www.xyz.com/somedoc.html” for example is done by a statement, where “http://www.xyz.com/somedoc.html” is the subject of our statement, its predicate is “author” (which is a property in RDF terminology) and its object is the literal “Smith”. Another possibility would be to use a resource (with an URL) as the object of such a statement, like “http://www.xyz.com/smith.html”, assuming we want to use this URL as identifier for the person Smith. This simple example reveals the basic building blocks of any RDF statement: resources and literals. Anything that can be reached by a URL is a resource whereas a literal is a simple character string. Subjects and predicates always need to be resources while an object may be either resource or literal. In addition, predicates normally are properties described by an RDF schema. The RDF specification does not insist on any implementation of the statement concept in particular. It introduces a graph representation suitable for the human reader and an XML-encoding of that graph suitable for XML based parsers. The XML encoding is probably the most popular RDF representation. To create self-defined predicates like “author” in our example, one needs to create an RDF schema. Like RDF metadata these RDF schemas consist of statements and hence can be expressed utilizing the same XML syntax or any other representation. With RDF Schema resources can be modelled as classes and predicates as

properties. Thus, it is possible to constrain the type of a predicate's range and domain. For instance, we can say that the predicate author may only point to resources that are instances of a class Person and may only be applied to resources being instances of a class Book. Since we decided to utilize RDF in OLR2 for both the annotation of content as well as the description of course structures we developed an RDF schema for this purpose. Our implementation focuses on the cataloguing/annotation and on the logical document features of RDF. An OLR2 course is a Logical Document. Cataloguing is used to store element information (e.g. title, author). Each course consists of a number of units that contain elements and further subunits. Each element represents any kind of Internet resource accessible through a known URL. For the first version of our introductory course on Artificial Intelligence, we defined five types of basic elements: *Topics*, *examples*, *slides*, *exercises* and *further references*. This choice reflects the typical building blocks of a lecture on an abstract level. If necessary, further element types can be incorporated easily to satisfy other users' needs (we used, for instance, additional elements in our Software Engineering course). The basic building blocks (units and elements) are linked together in a tree-like structure that represents a course. Each element is described by metadata. The vocabulary describing each element is basically the Dublin Core Metadata set. We used RDF sequences to link elements to units and units to courses. This is necessary because the order of the course elements is essential. The disadvantage of this was, that in the RDF Schema (RDF Schema Specification, March 2000) it was not possible to constrain the type of container elements. The RDF schema for OLR2 can be found in [41].

3.2.3 Database Schema - Storing RDF in a Relational Database

In essence, everything in RDF is expressed through statements: simple triples composed of resources, namespaces and literals - no matter how complex the RDF schema behind might be. XML syntax is the standard approach for hiding RDF in HTML pages. This approach always requires a parser to analyze the meta-information and it conflicts with one of RDF's key concepts where a group of RDF statements makes propositions about several distributed resources linking them together to one Logical Document. In contrast, using triples directly makes it easy to store RDF metadata in a relational database. In Doing so, we are able to create a repository for metadata managed at one central location by using relational database technology. This approach separates the metadata from the content it describes. SQL queries are used to extract the relevant RDF statements. An obvious advantage of storing RDF in a relational database is performance: An SQL query, which selects a couple of statements, can be much faster than parsing an RDF document in XML representation to retrieve the same results. Especially when a lot

of similar queries are executed, the database query optimizer and cashing mechanisms can speed up requests considerably. When looking at large numbers of statements, compact storage is another plus of the database approach: Within a set of RDF metadata a lot of literals could occur more than once. Namespaces are a good example for this characteristic: Every resource name is preceded by a namespace and these namespaces are often similar or identical. To avoid redundancy, these namespaces are kept in a separate table and are referenced by their respective IDs. For our OLR2 system, we modified the McBride schema, which is one of several suggestions presented on [97] and also discussed within the RDF community. The OLR2 system is based on the Oracle-8i database, but any standard relational database would be suitable. The main table in our database is RDF_STATEMENT. This table represents the relationship between the three parts of a statement consisting of *resource* (stored in RDF_RESOURCE), *predicate* (also stored in RDF_RESOURCE) and *object* (stored in either RDF_RESOURCE or RDF_LITERAL). Therefore RDF_STATEMENT contains three main attributes: *subject*, *predicate* and *object*. These attributes are references to the resource and the literal table. Since the object can either be a resource or a literal, we use two attributes for *object*: OBJ_RESOURCE and OBJ_LITERAL. The Open Learning Repository OLR2 is a repository to integrate, manipulate and annotate more than one course. Thus, we need to store large amounts of statements for every course. For this purpose, we utilize the table RDF_MODEL. Each model corresponds to one course.

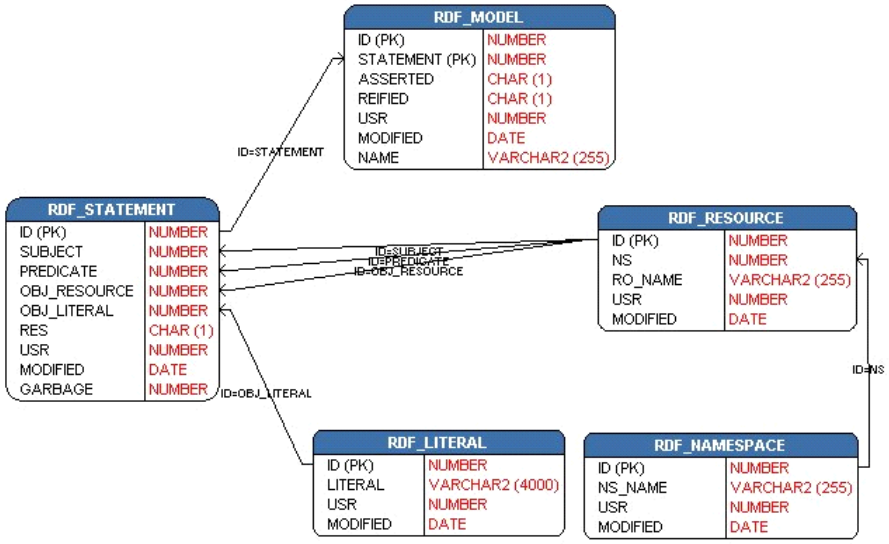


Figure 3.4: OLR2 Database Schema

Distinctions to the McBride schema Because OLR2 is used in a learning context, we established different user groups with different roles and rights. Every group may have a specific view on courses and metadata. Hence, we defineed a table RDF_USER for user administration which is connected to the other tables via the attribute USR. We also add the attribute MODIFIED representing the last modification date. In OLR2, all dynamic content is created based on SQL queries stored in the table SQL_QUERY together with a short description to facilitate the reuse of such queries and to support the PHP interface. From developers' perspective this greatly enhances reusability and maintainability of the underlying PHP source code. In order to evaluate the different visualizations and navigation possibilities in OLR, we define a table RDF_TRACK to record the user behavior while accessing course elements (which resources have been visited, in which order, how often, in which view). The basic OLR2 database schema is shown in Figure 3.4.

3.3 Interfaces

The Web interface for browsing and manipulating courses in OLR2 needs to be highly dynamic, since it needs to take into account the current state of the database. For this reason, all HTML code is generated on demand by PHP scripts. To control the complexity of the system, the PHP scripts are organized in several layers. Structure and purpose of the different layers are briefly outlined below. Database access with PHP is straightforward: PHP already comes with a built-in API for communicating with an Oracle database through the standard OCI8-interface. We designed a number of SQL queries to suit the special needs of the OLR2 system. These queries are stored in a database table, SQL_QUERY, with a unique ID, a query name and a short text describing the query's purpose. This approach greatly enhances maintainability and transparency of the system. Queries may contain parameters like resource-IDs specified in brackets. Core of the code for running SQL queries is the PHP-class *RDFStatement*. Its constructor requires the query name and eventually a number of parameters. *RDFStatement* then executes the query and transfers all results into a PHP array. All database specific code is hidden behind the public interface of this class. On top of the *RDFStatement*-class, we have developed the OLR2-API - a growing number of PHP functions like *getResourceTitle(resource_id)* that take some resource-ID as in-parameter and retrieve all statements about the specified resource for a specific property. These *getResourceXXX()*-functions utilize the *RDFStatement*-class. Note that database primary keys (usually integer values) serve as in-parameters to identify resources rather than a combination of namespace and literal which tends to be long strings. This is extremely useful for our web interface, since it keeps track of all state in-

formation (e.g. current course, unit or element) by URL parameters. The OLR2 API is accompanied by a number of other APIs such as an API for user and session management and an API for import and export of RDF source in XML syntax. The next layer consists of a number of basic building blocks - PHP script fragments calling API functions and performing the HTML markup of the returned results. For instance, there are PHP blocks for creating the different navigation elements or for displaying content or metadata of a course element. The final abstraction layer is represented by templates. In essence, templates are HTML files composed by dynamically putting together the basic building blocks. Most templates follow the same structure with a navigation element on the left, a content area on the right and above that a header section displaying title and essential metadata (see Figure 3.5. The templates also verify user access rights.

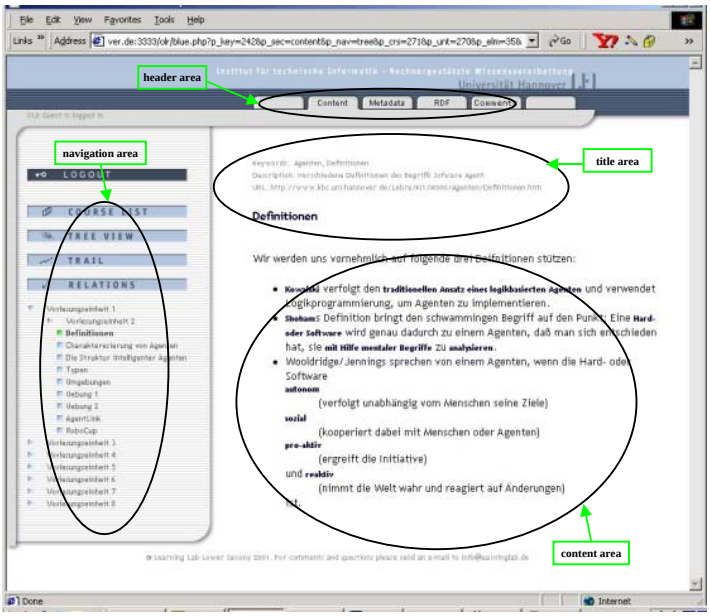


Figure 3.5: OLR Sample Template

The structure of the templates directly supports our multi-view vision. If, for instance, we want to use a trail instead of a hierarchical navigation, only one line of code needs to be changed in the appropriate template to replace the inclusion of the hierarchy-block by the trail-block. Although all RDF data are stored in the database tables, the content contributor's web interface allows the direct import and export of RDF source in XML syntax. After inserting XML code describing an OLR

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

3.4 Evaluation of OLR2

The focus in the first generation of our open learning repositories, OLR2, was the technical infrastructure. So the main questions were, how to store the metadata, which metadata modeling languages should we use (O-Telos, XML or RDF) etc. Even though we have implemented different navigation interfaces (classic hierarchical tree-like navigation, a trail navigation and semantic (context) net navigation) and realized different views on a given course, we admit the lack of pedagogical background during the design phase of OLR2.

The OLR2 system has been used in the context of two courses, one in artificial intelligence and one in software engineering. At the end of the semester, a scenario-based evaluation of OLR2 has been carried out. In this evaluation survey 20 students, which had attended an introductory course on artificial intelligence, were interviewed using a semi structured guideline. The lecture notes of this course on artificial intelligence had been provided to the students using OLR2. The aim of the survey was to find out how to improve the lecture notes on the one hand, and the OLR2 on the other hand. Based on these interviews, a “scenario-of-actual-use” was written. In addition, the designers, teachers and developers of the lecture notes and the OLR2 were asked to write “scenarios-of-intended-use” where they should express their ideas of how the material should be used by the students. The comparison of the “scenario-of-actual-use” and the “scenarios-of-intended-use” revealed that the OLR2 had been used in a very restricted way by the students. Much of the

functionality implemented in the OLR2 had not been used or has been seen as not useful for the students to reach their objectives. The survey is described in more detail in a technical report [128]. One reasonable explanation of the mismatch between the use made by the students and the functionality provided by the OLR2 is that the learning activities of the students are oriented to objectives determined by the course and its embedding in the organizational structure of an university and characteristics of the learner.

One consequence of this result is that we should not only implement different views on learning materials, but also have to provide help-files, which address the learners' reflection of learning strategies. By doing so we face the fact that choosing from different learning modules requires reflection on ones own activities of learning.

The meaning of contextual conditions for the design of the next OLR generation Another consequence of the evaluation mentioned above is that in the further design process of the next OLR generation general conditions and characteristics of the learner should be addressed more explicitly. With the term general conditions, we refer to characteristics of the educational setting in which the learning material is embedded. The most important characteristic of the educational setting that may influence the use of a certain learning material are the implicit and explicit learning objectives. While in many educational settings some learning objectives as the acquisition of domain specific knowledge are named explicitly, other possible learning objectives as the acquisition of social and scientific competencies stay more ore less implicit. It seems reasonable that these learning objectives propagated by an educational setting have an impact on the selection of the students individual learning activities.

A clear description of the implicit and explicit learning objectives within a given educational setting and its possible influences on students behavior will be very important to make comprehensible design decisions. Only if possible conflicts between the learning objectives within a given educational setting and the learning objectives of the developed learning material can be anticipated, it will be possible to decide if the learning material should be adapted to the educational setting or vice versa. This problem will be even more important if the learning material is constructed on a special educational model with explicit learning objectives and a certain conception of the learner. We assume that focussed changes in learning can only be fruitful in interaction with organizational changes (lecture and exams) - it will not change by simply offering a new e-learning system to the students. In other words, we assume that the use of a learning environment as OLR2 could only be understood adequately as a part of more complex and longer lasting students

learning activities. The educational setting with its implicit and explicit learning objectives and their possible impacts on the students behavior will constitute the context, which is addressed by scenarios written by all stakeholders. Scenarios of use appear to be an adequate way to describe the possible impacts of the context, because they force the writer to think about the concrete behavior of the user.

Scenarios as a medium for communication Beside the fact that scenarios seem to be an adequate way of describing the use of a certain artefact on different levels of abstraction and to highlight different perspectives on one task, they furthermore serve as a medium for communication between all involved stakeholders. A collective base for communication is very necessary because of the interdisciplinary design team involved in the design of the next generation of OLR, OLR3, and the integrated learning modules that consists of instructional designers, computer scientists and teaching staff. On the one hand, in the development of any kind of educational material many different variables like the learning content, the underlying pedagogical models, characteristics of the learner, contextual conditions and technological requirements have to be taken into consideration [140]. On the other hand, it is unrealistic to recommend that every stakeholder has expertise in all the addressed topics. While the cooperation of an interdisciplinary design team allows addressing all the relevant topics it keeps still necessary to find a way to communicate the different ideas, concerns, etc. within the design team. Scenarios seem to provide such a collective base for communication within the project. The necessity to formulate the possible consequences of certain variables, theories, etc. on the concrete behavior of an actor shall assure that the necessary information to make decisions on the design, can be communicated in the whole team. An adequate medium for communication is also an important condition to understand made design decisions at a later point of time and to receive starting point for further evaluation procedures.

The importance of well-structured learning materials Good structured learning materials are very important for learning and teaching, but appropriate and well structured learning materials for e-learning are absolutely crucial. During the development of OLR2 and specifically, its usage in the context of two lectures “Introduction to Artificial Intelligence” and “Software Engineering I”, we asserted that our existing course materials can not be taken over without enhancement, some modifications, and restructuring. Thus, new learning materials for the course ”Introduction to Artificial Intelligence” had to be produced and already existing materials had to be restructured and annotated.

Chapter 4

OLR3

4.1 Motivation

As mentioned in the previous chapter, we realized that there was a lack of pedagogical background in the first generation of our OLRs. Thus, we decided to design and implement a new prototype of OLRs, which we called OLR3. In a team of computer scientists and pedagogues we focused on, how we can use an open learning repository to enhance teaching and learning. For this purpose, we have implemented different instructional models. The results of the evaluation of the first generation of our open learning repositories, OLR2, motivated us to address the general conditions that influence the use of an e-learning system in a given educational setting more explicitly in the further design process of the OLR3. The multidisciplinarity of the project team made it necessary to find a medium for communication that allows to communicate domain specific ideas and theories beyond the different areas of expertise. As described in the evaluation report [128] of OLR2, we recommend a scenario-based design to handle this task. In this chapter, we will first give an overview on the different instructional models implemented in the second OLR generation. Then, we will describe briefly the architecture of OLR3. Finally, we will present the different functionalities and web interfaces in OLR3 and give a short comparison with other course editors.

4.2 Instructional Models and Scenarios in OLR3

Learning management systems and learning environments as well as standards for metadata in the field of learning, often implement or address specific learning and teaching principles, theories and models. These learning theories are addressed either implicit or explicit. This means that neither teachers nor learners can choose a

way of learning which is suitable best to their preferred learning strategy or learning style. Choosing from different learning principles or models of course requires reflection on ones own activities of learning or teaching. In livelong learning, learners have to be able to decide on learning needs and choosing a preferred way of learning themselves:

"Dieses lebenslange Lernen soll und kann jedoch nicht lebenslang durch formale Verpflichtung und Kontrolle erzwungen werden, sondern es muss eine freie Entscheidung jedes mndigen Menschen sein, den eigenen Lernbedarf und die Lernform selbst zu bestimmen. In Zukunft wird es vermehrt Mglichkeiten geben, selbstbestimmt - im Sinne einer Selbststeuerung - zu lernen, unabhngig von brokratischen Bedingungen, vorgeschriebenen Curricula und institutionellen Organisationsformen" [125].

Modeling different spaces or models means modeling different views on learning material. We designed OLR3 open to different learning and teaching strategies. In OLR3, two different spaces are modelling different aspects/views: The Instructional Model, and the Structural and Cooperative Model. In order to face the different needs of novices and experts, we offer different navigational structures: guiding sequences (supporting learning processes) as well as links to related learning objects (supporting exploration).

4.2.1 Instructional Model

There is no unique instructional ontology but different approaches, theories and models. In the Instructional Model, instructional phases and relations are specified. They are derived from different instructional models and learning theories. Furthermore learning processes are modelled along different instructional principles (for instance PBL and case-based). Existing metadata standards do not support learning processes [63]. But as we discussed in chapter 2, a learning object may play different roles in the learning process either within one Instructional Model or in different Instructional Models. In doing so, we subscribe the view of Meder, who stresses that learning objects should be defined in six dimensions [96]. One of these dimensions defines the position of an learning object within the process of knowledge acquisition. Up to now, we implemented two different instructional models derived from different principles and paradigms: Expository Teaching by Ausubel (cognitivism) and a Problem-Based-Model (situated approach).

Learning Sequence and Phases according to Ausubel Ausubel's theory is a cognitive learning theory. "*The model of cognitive organization proposed for the learning and retention of meaningful materials assumes the existence of a cognitive structure that is hierarchically organized.*" [28]. Ausubels theory deals with how

learners learn large amounts of meaningful material from textual presentations. According to Ausubel, learning is well organized by superordinational, representational, and combinatorial processes that occur during the reception of information.

"A primary process in learning is subsumption in which new material is related to relevant ideas in the existing cognitive structure on a substantive, non-verbatim basis. Cognitive structures represent the residue of all learning experiences; forgetting occurs because certain details get integrated and lose their individual identity." [80]

Roles according to Ausubel According to Ausubel a learning sequence consists of four learning phases: *Advance organizer*, *Progressive differentiation*, *Practice*, and *Integrating (Ausubels Expository Teaching)*. Each Phase is mapped to a specific didactic function. A given learning object plays a precise role within a learning phase. A learning object can play different roles within a given learning sequence. For instance, a text or video-file can be used in the phase "advance organizer" as motivation for the learner and also be used in the phase "practice" as an "apply practice element" in another course (see Figure 4.1). Table 4.1 describes the instructional purposes of the different learning object roles according to Ausbel's subsumption theory.

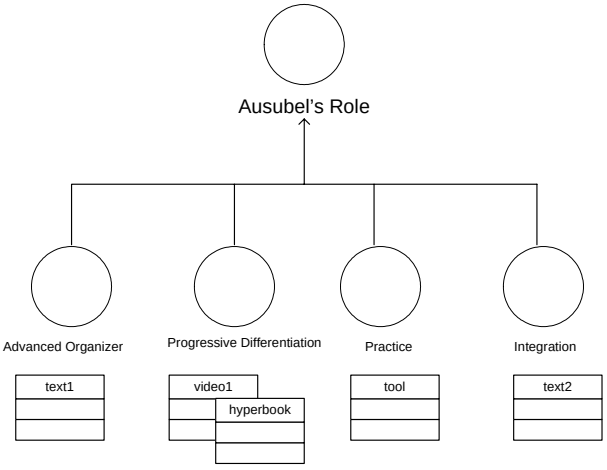


Figure 4.1: Roles According to Ausubel

Problem Based Learning Many instructional models are concerned with Problem Based Learning (PBL for short). Most of them refer to various approaches of constructivism. For standardization it may be useful to identify and model common phases which are existent in many models. Referring to Merrill these are [98]:

Role	Instructional Purpose
Advance Organizer	Prepare integration of knowledge and develop learning goals. Subsuming bridge between new learning materials and existing related ideas. Present introductory materials that help students relating new ideas to existing knowledge schemes. Ask questions like: What do you want to find out? What steps do you need to get there? What do you already know?
Progressive Differentiation	Presenting learning contents in structured forms. Every unit prepares the transfer of differentiated concepts.
Practice	Repeatedly use the learned content in order to reinforce it.
Integrating and Connecting	Connecting the newly learned content with other knowledge/examples and contexts.

Table 4.1: Roles According to Ausubel

Role	Instructional Purpose
Goal Description	Problem to be solved. Set ultimate goal.
Specify Criteria	Criteria your solution should meet. What aspects do you want to focus on? How do you know that you reached your goals?
Background Knowledge	Sample and share knowledge. Information retrieval.
Generate Ideas	Draft provisional hypothesis. Generate and develop solution. Compare different solutions.
Implement Solution	Implement and compare different solutions.
Reflect	Evaluate solution, reflect product, reflect process.
Generalize	Conceptualize, integrate, and generalize your knowledge (i.e. move from example to theory).

Table 4.2: Roles According to PBL

Activation of prior experience, *demonstration of skills*, *application of skills*, and *integration of these skills into real world activities*. Use-cases can be derived from specific instructional models and do exactly fit into local instructional practice. Our use-case is derived from the "Konzept der mehrperspektivischen Technikdidaktik" (concept of multi-perspective instruction in the education of engineering) [130] and an instructional model based on this concept [149].

Roles According to PBL Table 4.2 describes the different roles a learning object may play according to PBL. A learning object can also play different roles within different learning sequences modelled according to different instructional models. For instance, a text presenting a theory can be used in the phase "advance organizer" (roles according to Ausubel) as well as in the phase "Generalize" (roles according to PBL).

4.2.2 Structural and Cooperative Model

Compatible to learning processes specified in the Instructional Model, but also independent from specific theories, methodical and structural relations as well as cooperation-supporting relations and learner activities are specified. Examples for methodical and structural relations are for instance “theory generalize example” or “exercise applies theory”. In order to model these relations, each learning object must be categorized methodically. In this model also cooperative relations and learner activities are modelled. Examples for learner activities are for instance “add keyword”, “add relation”, “add comment”, etc. Cooperative relations are modelled to guide students cooperative work.

4.3 Technologies and Architecture

The main focus of the follower prototype of OLR2 is the design, implementation and test of various instructional models, in order to enhance learning and teaching using open learning repositories. However, for this purpose important modifications and enhancement of the used technologies and architecture to face the new complex requirement were inevitable.

Our 2nd generation of open learning repositories, OLR3, is implemented in Java and works as a JavaServlet, running on an Enhydra [47] Application Server (open source software). One of the most important advantages of the Enhydra Application Server is the absolute separation between design (HTML pages) and the Java code, which facilitates the system maintenance. The application server is connected to an Oracle Database via JDBC, which is used to store the metadata entered by course authors and students. OLR3 uses the same basic database schema as OLR2 with narrow additions. The basic ideas of OLR2 have been taken over by OLR3. We distinguish in OLR3 also two types of metadata: *structure metadata* and *annotation metadata*. *Structure metadata* represent information about the structure of a given course or course unit, the navigation path within a course or a course unit, and the relations between different learning objects. *Annotation metadata* represent the information about the content itself. The LOM metadata standard is used to annotate the learning materials in OLR3. Of course, not the complete LOM is used, but rather a LOM subset. All metadata are defined in RDF and RDFS. The RDF schemes, needed for either the annotation or the structure of learning materials can come from anywhere in the internet. The database does only hold the metadata from RDF files, that are prepared to act as a data source for the courses metadata.

The central part of the system is a storage called “StatementPool”. It holds all metadata that is known to the system at runtime. When an author starts working

on a course, the pool is filled with the already existing data about that course from the database, and all statements from the used RDF schemes. Any referenced RDF schema will be parsed using the SiRPAC RDF parser [7], whereas imported RDF files are parsed by a VRP RDF parser [87], which provides semantical checks against given RDF schema rules. Different schemas enable different "metadata profile applications"[1]. Every author can build his own metadata application profile in OLR3.

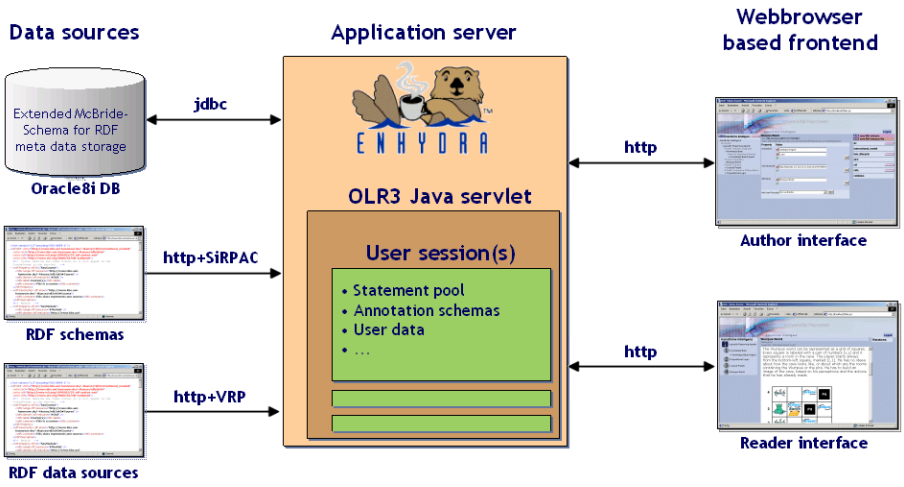


Figure 4.2: OLR3 Architecture

OLR Open Schema The principal disadvantage of OLR2 is that the structure metadata schema is hard implemented. That is, all courses designed in OLR2 have the same structure (course, course units, etc). Defining new structure schemas implies an enormous implementation work. Since we aim at realizing different instructional models, and consequently defining different structure schemas according to different instructional theories, we designed and implemented a new schema for OLR3. To make a long story short, OLR3 takes over the basic ideas of OLR2 (metadata based e-learning portal, RDF, database schema), but it was complectly re-implemented. The new OLR3 schema[2] is very flexible. Merely an RDF-class (*olr3:course*) is preassigned as root for all course structure schemas (Instructional Models).

[1] Application profile: an assemblage of metadata elements selected from one or more metadata schemas and combined in a compound schema [44].

[2] can be found on `http://www.kbs.uni-hannover.de/~hdhraief/OLR/rdf_schema/OLR3/rdf/`

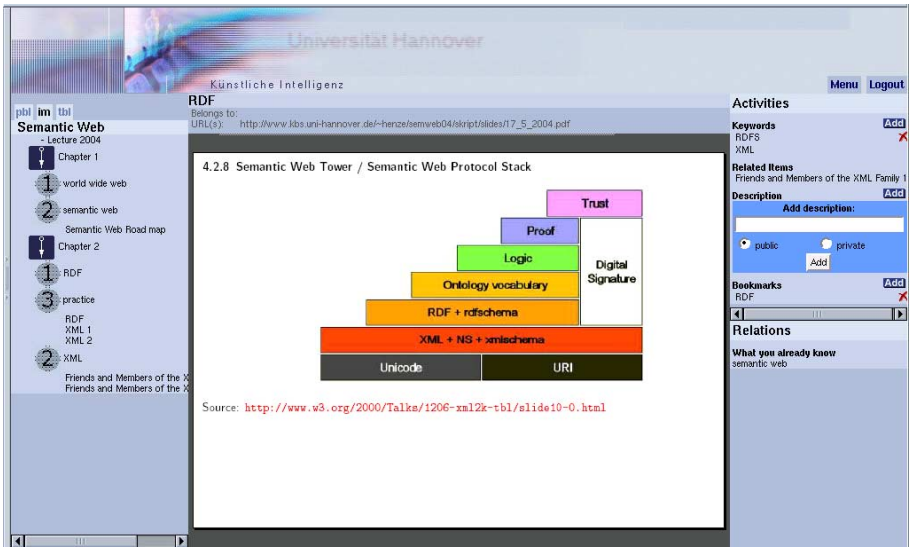


Figure 4.3: OLR3 Reader Interface: IM Structure

4.4 OLR3 Interfaces

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

4.4.1 OLR3 Learner Web Interface

Learner of a given course can navigate through an existing course structure, displayed as a tree and extended by additional, metadata-defined images for better understanding. We implemented the instructional models, presented in Section 4.2, in our OLR3 prototype. The Ausubel model has been used for structuring the courses “Semantic Web” and “Technologies for the Internet I”. The PBL model is implemented, but not used so far. In order to structure courses using PBL, the modification of existent content, and even the creation of new content, which fits to the different learning phases, is essential.

Within the navigation tree, the user may select a single course element, whose content will be shown in the middle of the screen. A specific engine prepares and filters the content, and displays it based on a stored layout. The reader interface also offers the reader the possibility of making minor additions to the metadata

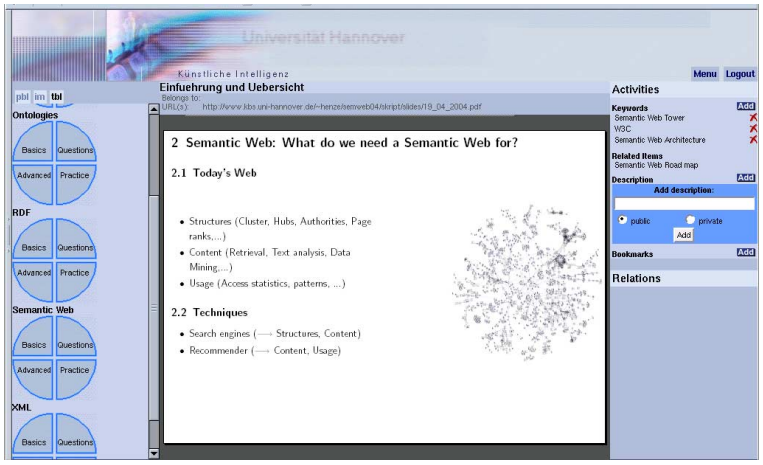


Figure 4.4: OLR Reader Interface: TBL Structure

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. Theses functions implement the relations and activities modelled in the Structural and Cooperative Model, which have been introduced in Section 4.2. The figures 4.3 and 4.4 show on the right side examples of those activities and relations.

TBL schema In addition to the implemented instructional models (Ausubel’s subsumption theory and PBL), we have defined and implemented a new course structure: TBL - Topic Based Learning - schema. TBL corresponds to our traditional course structuring. The TBL schema consists of 4 phases:

- *Basics*: Includes the basic information related to the topic. These basics are usually presented by the teachers.
- *Advanced*: Here are, for instance, further reading materials for interested students and a profound knowledge about the topic comprised.
- *Practice*: Includes exercises.
- *Questions*: For instance FAQ or former exam questions.

In the TBL reader view, these four main structure elements are displayed as pieces of a pie. This view of content reflects that the succession of the phases is irrelevant.

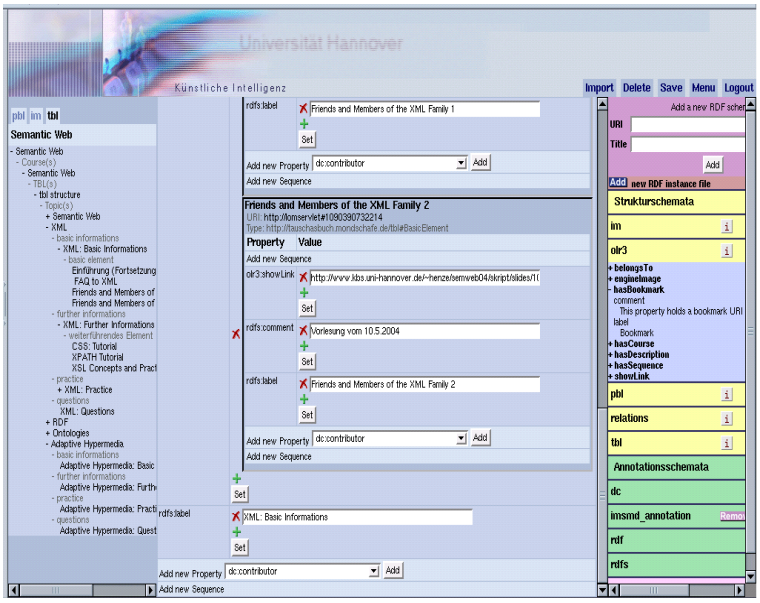


Figure 4.5: OLR Author Interface

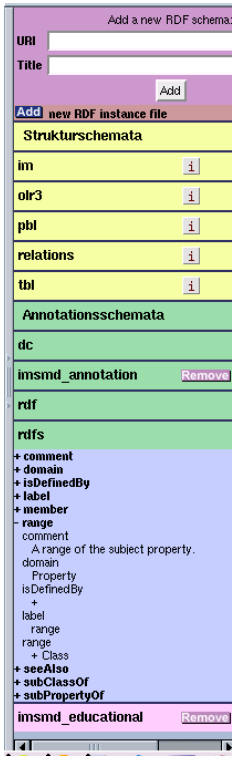
4.4.2 OLR3 Author Interface

The second interface is the RDF metadata editor which is intended for course authors, who can navigate through the structure tree of a course and select any subelement. The author interface consists of three parts as shown in Figure 4.5:

- *The course viewer* (on the left of the screen): The different implemented structure schemas (IM (Ausubel), PBL, TBL) are displayed like an “index card”. The course viewer displays all instances for a selected RDF-schema in a tree manner. Each instance is represented by a node, beginning by the course class (root). The root is, as mentioned before, the sole hard implemented element. The rest of the elements and the tree structure are entirely driven form the RDF-schema. The subelements of a given class can be expanded by clicking on the plus near the instance name. All the items in the tree can be selected for further annotation, which takes place in the middle part.

- *Content area* (in the middle of the screen): It displays the editable properties (attributes) of a selected item on the course viewer. The author can also add new annotation or structure properties to a selected item. For this purpose, the editor contains a list of valid properties. “Valid” means that the prop-

(a) Course Viewer



(b) Structured Schema Viewer

erty either has no *rdfs:domain* definition or an *rdfs:domain* definition that refers to the origin class of the selected item. In doing so, OLR3 ensures the construction of valid statements.

- *Structural schema viewer*, also called *toolbar* (on the right of the screen): It displays all system-bound RDF-schemas and provides the possibility of expanding any included schema to show the included properties. The properties can also be expanded to show for instance their *rdfs:domain* or rdfs:range. For a user friendly display the *structure* and *annotation schemas* are distinguished on the toolbar. Additionally, the *structure schemas* are linked to web pages with information about their structure and their usage. The author can also bind RDF schemes (e.g. DC, DCQ, LOM) from anywhere in the Internet to extend the set of available properties for annotation, or unbind RDF schemes that are not needed anymore.

4.4.3 Comparison with other Course Editors

It is, as a matter of principle, difficult to establish a straightforward comparison of OLR3 to other "tools" or software, since it is a framework, where the metadata management and the use of educational metadata are investigated in parallel. Hence, the following comparison is only related to the OLR3 RDF metadata browser/viewer (author interface), which is only a part of the OLR3 framework and does not include, for instance, the modeling and use of different instructional models in OLR3. There is a good deal of editors for the semantic web language RDF. All of them have a common purpose, namely to simplify the writing of RDF schemas, and consequently to facilitate and encourage the annotation of resources on the Web. In the following, we compare OLR3 metadata editor with three established RDF schema editors.

CREAM and ONT-O-MAT CREAM - CREAting Metadata for the Semantic Web - [62] developed at the university of Karlsruhe has a very similar schmea-driven metadata approach. However, it is, in contrast to OLR3, not a web application, so that an installation of the software is necessary. Another difference between the ONT-O-MAT implementation of CREAM and the OLR3 metadata editor is that OLR3 stores all the data locally in a relational database and offers no features for the use of distributed data (e.g. on different peers). CREAM against it already work with distributed data.

Conzilla Conzilla [109], the concept browser developed by the CID in Sweden, is an interesting editor for LOM. However, it not designed to enable the inclusion and use of other metadata schemas, especially course structure schemas. Moreover, Conzilla is not used to display course content.

K-med Course Editor The K-med project [65] was developed at the universtity of Darmstadt, Germany. It uses LOM to annotate and structure the course. The system, however, does not enable to add new course structures or annotation schemas.

Chapter 5

OLR4: OLR3 Reengineered

5.1 Motivation

The design and implementation of the first and second generation of OLRs have been realized over three years within different projects, with different focuses, and by several developers. As described in Chapter 4, OLR3 uses an application server, different web application frameworks and a database. Different modules of OLR3 are coded in different programming languages: HTML, XML, Java, SQL. A complex software project running in the open environment of the web accrued. Software developers, who have/want to extend and/or to optimize OLR3 have to deal above all with different complex tools, technologies and concepts. This implies a considerable loss of time. The upgrading, maintenance and optimization of the system became more and more complex. Thus, we recognized the necessity of a reengineering and a progressive refinement of OLR3, in order to enable the entrance of future researchers, designers and developers with a low overhead in the project. The main focus of the third generation of OLRs is to realize the openness principle related to the repository, i.e. the used technologies. We aimed at introducing standard components and tools to improve earlier implementation decisions and to release OLR3 as an open source project. Additionally, the web user interface has been also optimized.

This chapter is organized as follows. In the next section, we describe the problems in the project infrastructure on different application tiers and propose solutions and modifications to solve them. In Section 5.3, we introduce systems and services, which address the concern of the durability of online resources and describe how these systems are related to our OLRs. We will also briefly describe the role of OLRs in a P2P network.

5.2 Technologies and Architecture

In this section we will describe the problems we detected at the different tiers of OLR3 and propose solutions, which have been implemented during the reengineering of OLR3.

5.2.1 Back-End Tier

Problems Related to the Database in Former Prototypes

Database Vendor Lock-In The first and second generation of OLRs use an Oracle database and further Oracle specific functions for string concatenation, date conversion and conditional expressions in three database views. In order to realize our openness principle, we decided to move from the commercial database product Oracle to an open source database like PostgreSQL [121] or MySQL [102].

Database Connection The OLR3 system connects to the Oracle database using a JDBC over ODBC connection. That means, the Oracle client libraries need to be installed on the computer running OLR3 (server). Additionally, the ODBC connection to the database has been configured on the OLR server. This setup is complicated compared to connecting to the Oracle database using a pure Java JDBC driver. That would prevent the installation of the Oracle client software and the configuration of ODBC services.

OLR3 Persistence In OLR3, two Java packages are responsible for SQL database queries and object persistence. The first package *olr3Servlet.data* contains one class for (almost) each database table. Instances of these classes represent entries in the corresponding tables (data objects). The single attributes can be read and written using get and set methods on the data objects. The package *olr3Servlet.db* contains again a class for each database table, and thus for each class in *olr3Servlet.data*. The classes of the second package perform queries on the database (access classes), populate the data objects with the query results, or save the state of data objects to the database tables. The populated data objects are used in the application layer to populate the HTML pages with the attributes of the data objects. In general this is a sufficient layered design of the OLR3 persistence and concerns are well separated. However, it is still tedious to propagate a database model change (e.g. a new table or column) up to the persistence classes. In case of a new table, we would have to write a new data class, a new access class, the needed query, and data object population code. After adding new classes, we also would have to test the new code and fix introduced bugs. It would save a lot of

time, if the data classes and database access classes would be generated, and if the access classes already would provide basic methods to query, to retrieve, and to save data objects.

SQL Queries If one deletes an RDF course in the OLR3 system, many RDF statements, resources, namespaces, and literals must be deleted. In OLR3, each of these objects is deleted using a separate SQL query, which is sent to the database. This is expensive, since the database is often running on a different host than the OLR3 system, which implies that each single query and its result have to travel through the network. Concerning the processing of queries, which have to travel through the network, the runtime complexity for deleting an RDF model in the database is $O(n)$, where n is the number of objects to be deleted. For instance, an RDF model consisting of 1000 RDF statements is connected to at least two resources and a literal or three resources. The resources are connected to namespaces. This implies, $1000+3000+3000 = 7000$ queries are executed (in the worst case) to delete the model from database. Some of the delete queries are actually executed in separate database transactions, which means the deletion of a model is not isolated 100%. In case of an error, during the deletion of database entries, the data can potentially become inconsistent, since the hole transaction - deleting the model - can not be reverted completely.

Solutions

Removing Database Vendor Lock-In The vendor lock-in is mainly caused by the OLR3 database views and the usage of the Oracle specific SQL language constructs, e.g. the conditional string concatenation. To avoid the vendor lock-in, we first replaced the views in the database with SQL queries in the Java database access layer, then replaced the conditional string concatenation with equivalent Java language constructs in the database access layer. After that, we provided an SQL script, which creates the database tables independent of the database vendor of the target database. The script can be generated automatically (See Section 5.2.1). This way, the OLR3 system does not need any views in the database. In doing so, it becomes easier to create the OLR3 database schema in different relational databases.

An alternative to this solution is to use inline views, which means a select statement in the from clause in an SQL query. But inline views are not supported in all open database management systems (e.g. MySQL).

Postgresql We opted for the open object-relational database management system PostgreSQL as alternative to Oracle. Once the script to create the OLR database

schema has been generated, only the JDBC database driver has to be changed in an OLR property file. Afterwards the ANT build script, described in paragraph in 5.2.4, can be used to create the tables and data in the configured target database. For an in-depth description OLR4 database setup and configuration, see the installation and reference documentation [112, 113].

Using a Third Party Persistence Layer A third party persistence layer could save a lot of development time, when the OLR database schema is changed or extended. In addition to the time savings, a third party persistence layer lowers the learning curve for developers knowing the used persistence layer product. Future developers do not have to deal with project specific persistence code. Several open source and commercial Java persistence layers are available.

We decided to use Apache Torque [143] as persistence layer for OLR, because it provides tools to smoothly integrate and reuse the existing OLR3 database schema step by step. Hibernate [64] is very popular due to its transparent persistence approach and it is used more often than Torque. Transparent persistence means, that no database model needs to be specified, no source code is generated, and no tables need to be created. However, Hibernate is not suitable for our purpose, since the Edutella OLR4 interface (presented in Section 5.3) requires knowledge about the database schema.

Torque There are several ways to integrate Torque into a Java project. In the following, we first give a short overview of the functionality of Torque. Then, we describe the steps required to integrate Torque into OLR3.

- In the first step, a so called *database schema file* [144] is required. The *database schema file* is an XML file describing the database schema using markups to define tables, columns, constraints, keys, etc. Instead of writing such a schema file for OLR3 in the first step, we used a Torque tool called *generator* [142]. The *generator* generates a schema file based on the metadata of an existing database using JDBC to connect to the target database. The current version of the OLR database schema file is located in the OLR source code repository [111]. The *Torque generator* is also used to create the access and data classes and the Java object model.

- In the second step, the generated code and the persistence layer have to be integrated into OLR3. For this purpose, a properties-file called *Torque.properties*, which defines database connection and runtime parameters, has to be provided. The OLR4 object model classes are located in

the Java package *olr.om* and can be viewed in the OLR code cross reference [110]. A more detailed introduction to Torque and its usage in OLR4 with some examples can be found in the OLR database reference documentation [112].

- The last step is to use the data and access classes from the package *olr.om* instead of the classes from *olr3Servlet.data* and *olr3Servlet.db* in the OLR4 application.

The integration of new Torque access (persistence) classes turned out to be complicated. Therefore, we had to replace all code executing SQL queries and all result set operations by simple method calls to the persistence objects generated by Torque. Fortunately, it was possible to start with some isolated classes/tables like *rdf_user* and *rdf_group*. After testing the integration of those classes, we continued integrating the more difficult classes like *rdf_statement*. Currently, all the old packages *olr3Servlet.data* and *olr3Servlet.db* have completely been replaced by *olr.om*. OLR code, which implements the access to the database, data manipulation and query processing, is pushed down to the third party persistence layer.

Query Optimization Deleting a model and all related entities entry by entry is slow and can be done with queries like *delete from ... where id in (x,y,z)*, which would reduce the number of queries to the number of related tables. Thus, it would reduce the runtime complexity from $O(n)$ to $O(1)$, where n is the number of entries being deleted from database. After this query optimization, we only execute one query to delete all statements and six queries to delete the related resources, namespaces and literals. Six queries are needed and not three, because we need a lookup query for the ids of the resources, namespaces, and literals, which should be deleted, in order to ensure, that only entries, which are not referenced by other models, are deleted. Thus we need constant $1 + 6 = 7$ queries to delete a model from the database in all cases.

5.2.2 Middleware

Problems Related to the Architecture in OLR3 OLR3 uses the Enhydra Application Server 3.1, which is not 100% J2EE compliant. Hence, it is difficult to deploy the OLR3 application to J2EE compliant application servers. The term *Servlet* is used in some OLR3 Java package and class names. However, OLR3 is an *Enhydra Super Servlet Application* [47], and not a Servlet as implied by the names. That is, Enhydra itself provides a *super* Servlet, which loads the OLR3 application.

It would be interesting to package and deploy the OLR3 application together with the Enhydra Super Servlet as a J2EE compliant web application. In doing that,

we guarantee that users could then use any J2EE application server to run OLR4. The Enhydra Application Server had so far to be installed separately to run OLR3. The OLR3 start script starts the Enhydra Server, which loads the OLR3 application. Enhydra starts then Apache Tomcat - a J2EE Servlet and JSP engine. If we bundle the necessary Enhydra and Tomcat runtime classes and libraries with OLR4, the installation of the Enhydra Application Server becomes superfluously. OLR4 can be than started as a stand-alone web application, which runs in a dedicated Tomcat instance, or it can be deployed as a J2EE web application into any J2EE compliant application server.

OLR4 Target Architecture Figure 5.1 shows the target architecture of OLR4. This architecture is more open and flexible than the former one. The OLR4 application can be deployed into any J2EE compliant application server and no special database product is needed.

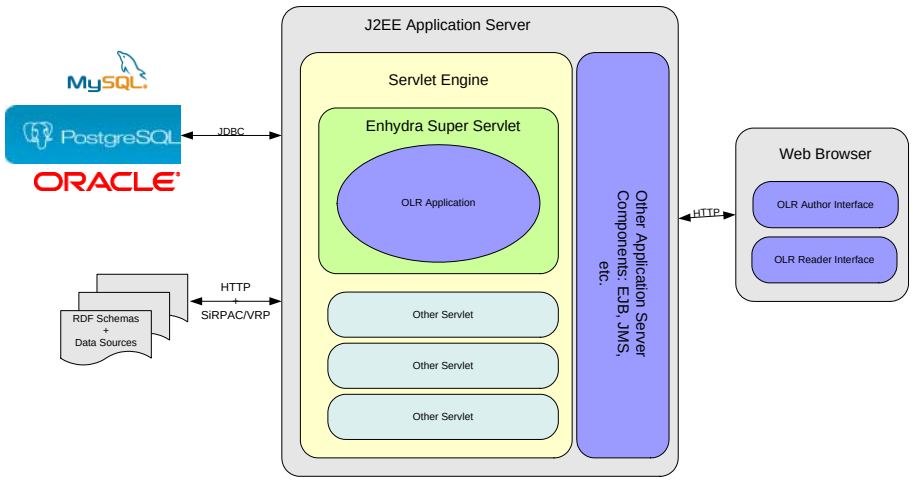


Figure 5.1: OLR4 Architecture

To include the Enhydra runtime libraries and classes in OLR4, we defined the appropriate Enhydra libraries and further libraries as dependencies in OLR4's Maven project descriptor (see subsection 5.2.4). Maven downloads these dependencies into the local library repository during build time. Additionally, the Maven build script for OLR4 was extended to include these libraries in the OLR4 distribution archives. Thus, the installation of the Enhydra Application Server is not required anymore and the installation of OLR3 becomes easier.

In order to deploy OLR4 as a J2EE application, we created a so called *Web Application Archive* (WAR) file [146], which contains all needed byte code, libraries and other resources as well as an *Web Application Archive descriptor*. The *Web Application Archive descriptor* file, *web.xml*, defines the Servlet class, the Servlet parameters, etc. In order to package the WAR file, we simply have to communicate the location of the web.xml to the Maven WAR plug-in [147]. Henceforth, when we invoke the WAR plug-in, the hole OLR4 application, including all dependencies, is packaged as a J2EE web application. The plug-in can be invoked by executing "*maven war*" in the OLR4 source root directory, which will create a file called *olr.war*. For further installation instructions and information about the WAR file, see the OLR4 installation documentation [117].

5.2.3 Application and Presentation Layer

Figure 5.2 shows the interaction between the presentation, the application and the database access layer. This architecture follows the model view controller pattern and has turned out to be very flexible and maintainable.

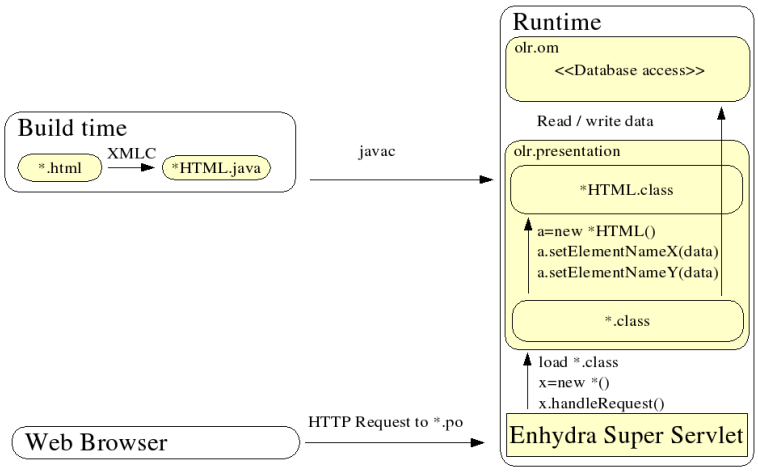


Figure 5.2: OLR Presentation Flow

OLR4 provides a HTML web user interface (UI for short) to author and read courses. The single screens of the UI are defined in various HTML pages located in the directory *OLR_HOME/src/resources*.

During a user session these screens are populated by the application layer with data from the underlying database. Thus, the application layer needs to access and modify the internal structure of the HTML pages.

OLR4 uses Enhydra XMLC to generate a Java class, *XMLC object* for each HTML page. These classes are used by the application layer to alter the internal structure of the HTML page at runtime. The OLR4 build system (Maven) invokes the Enhydra XML compiler (XMLC) and places all generated Java sources in the directory *OLR_HOME/target/src/xmlc*. The classes and files of the generated sources have the same names as the HTML pages, but with the extension ".java".

The only deal between the presentation and application layer is to use identifier and class attributes for HTML elements, which should be accessible via the XMLC objects. The XMLC objects will then provide methods to access and alter the HTML structure using the DOM (Document Object Model) API [35] as well as element access methods generated by XMLC.

Following the above implementation pattern, it is possible to design the web UI independent of the application layer as long as the identifier (id) and class attributes are defined.

Problems Related to HTML Style and Structure

A typical OLR3 HTML source file looks as follows:

```
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <style type="text/css">
<!-- .text_standard {  font-family: Arial, Helvetica, sans-serif;
font-size: 10px} .text_gross {  font-family: Arial, Helvetica,
sans-serif; font-size: 12px;} a { text-decoration:none;} -->
    </style>

    <script type="text/javascript">
<!-- function RefreshHeader() {
  Frame=eval("parent.Header");
    if (Frame != null)
    Frame.location.href = "Header.po";
}
//-->
  </script>

  </head>

  <body onLoad="RefreshHeader()" bgcolor="#acb8d5" text="#000000"
      link="#000000" vlink="#000000" alink="#666666" leftmargin="0"
      topmargin="0" marginwidth="0"     marginheight="0">
    <table border="0" width="100%" height="100%" summary="">
      <tr>
        <td class="text_gross" align="center" valign="middle">
          <font color="#FF0000"><b><span id="ErrorText"> </span></b></font>
          <form action="Login.po" method="post" id="loginform">
            ... more HTML tags ...
```

```
          </form>
          </td>
        </tr>
      </table>
    </body>
</html>
```

The above HTML source and all the other HTML sources in OLR3 show the following problems.

1. CSS (Cascading Style Sheets) definitions are not separated from the HTML source files. Changing the style is a tedious job, since about 30 files (the number is still growing) need to be changed.

2. Some styles are defined within the HTML tags as attributes like

   ```
   <body bgcolor="#acb8d5" text="#000000" ... >
   ```

   or style tags are used like

   ```
   <font color="#FF0000">
   ```

   This implies that style changes must be propagated to all HTML files.

3. Redundant Javascript code in the HTML source files, also cause maintenance overhead, if the Javascript code changes.

Solutions

Separating Style and Javascript from HTML Markup All CSS definitions and Javascript code should be first moved to a central CSS or Javascript source file, which is referenced by all appropriate HTML files. Then, HTML style or script code can be changed in a single file, which reduces maintenance overhead. Style information in the form of HTML attributes or tags should be replaced by appropriate CSS definitions in order to centralize the style and its maintenance. Separating the HTML markup from the style information would enable to change the look and feel of the OLR4 application in a very flexible and fast way. [37] gives an example of how flexible CSS based styles are.

The HTML source code shown above would then look as follows.

```
<html>
  <head>
    <title></title>
```

```
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link rel="stylesheet" href="style/olr.css" type="text/css" />
    <script src="script/olr.js" type="text/javascript">
  </head>

  <body>
    <table summary="">
      <tr>
        <td class="text_gross" >
          <span id="ErrorText"> </span>
          <form action="Login.po" method="post" id="loginform">
            ... more HTML tags ...
          </form>
          </td>
      </tr>
    </table>
  </body>
</html>
```

The separation of style and javascript source code from HTML is not implemented, so far, in OLR4. This can be considered as part of future work.

5.2.4 Project Infrastructure and Build System

As mentioned above, one of our aims by optimizing the software project OLR3 is to release it as an open source project. For this purpose, an optimization of the project infrastructure was indispensable. The OLR3 prototype was developed without using a code version control system, bug tracker or any mailing lists. The necessity of an optimization of the project infrastructure emerged at the beginning of the summer term 2004. OLR3 have had evaluated by two groups of students in the context of the software project[1]. Each group was formed of 4 students, which became also implementation tasks to optimize the detected problems and weak points in OLR3. We were confronted with the problem of installing OLR3 on several machines. The students desired also to install the OLR3 application at home. There was also the problem of the code version control, since the students have had to work in teams. In the following, we will present the different technologies used in the optimization process of the project infrastructure as well as the new project build system and describe their use and integration in the new prototype OLR4.

Code Version Control Developing complex extensive applications inside teams of several developers require the control of all files made over time. Otherwise getting the current version becomes very cumbersome, if the developers just exchange

[1]The software project is for computer science students at the university of Hanover. It is valuated with 9 credit points.

the files over the internet or copy them from one computer to another. Even small projects like the software project at the university, which takes place in almost all computer science studies, need a version control system.

A version control system maintains an organized set of all the versions of issued files. It allows developers to go back to previous versions of their files, in order e.g. to view the changes between them. A version control keeps a historically accurate and retrievable log of a file's revisions. It can be considered as "project-wide time machine" [141], i.e., developers can dial in a date and get any version of the project. In the current prototype of OLR we use the Concurrent Versions System (CVS for short). *CVS is widely used in both open source and proprietary software development projects, and is generally considered to be the best freely available, full-featured version control tool* [1].

Bug Tracker A bug tracking system is an efficient tool to manage, document, and record bugs in the software. Each bug should be connected to a test, which proofs whether a bug is fixed or not. This would also guarantee that already fixed bugs do not reenter the system (regression).

OLR at Sourceforge Sourceforge [133] offers reliable software project infrastructure at no costs. Sourceforge presumes the licensing of the software of the hosted project under one of the many open source software licenses such as the GNU General Public License (GPL) [54] [57], the Mozilla Public License (MPL) [100], the Apache Software License 2.0 (ASL) [27], etc. We decided to use the Apache Software License 2.0 for OLR. The OLR project was accepted to be hosted at Sourceforge. Thus, we could use the offered CVS version control system, the mailing lists, the bug tracker, the file release system, and the web space for the OLR project website [114].

Project Build System

Problems OLR3 does not dispose of any automated software build or configuration system. It is therefore complex to get all the sources compiled, as well as to run the system. Furthermore, OLR3 uses an XML-to-Java compiler (XMLC) [152], which must be executed before the compilation of the Java source files. The XMLC compiler itself must be configured within the used IDE. There are no SQL scripts to create the OLR3 database tables and no scripts to insert demo course data into an OLR3 database. It is difficult to get the right database setup for the OLR3 application. Thus, it is difficult to install and to run the OLR3 system in an acceptable time frame. The complicated and barely documented OLR3 software setup is a huge barrier to enter the project.

Most of successful open source software projects succeeded, because developers around the world were and are able to obtain, compile, configure and setup the software within ten minutes, so called *up-and-running-in-ten-minutes*. In the absence of build automation developers have to do primitive, repetitive, tedious, and irrelevant (for the problem domain) project build tasks.

The OLR4 build system should perform build and configuration tasks, like invoking the XMLC compiler, compiling Java sources, generating Java source code documentation and other project documentation in HTML, generating Eclipse project files, creating database tables, inserting basic demo course data into the database, and configuring the OLR3 runtime environment.

In the following, we will present two open source build automation tools, which are often used in open source and industrial Java projects.

ANT and Maven We decided to use Apache Maven to implement the new OLR build system and Apache ANT to perform database setup tasks. In the following we will briefly describe ANT and Maven and show how these tools are used in the OLR project. For an in-depth description of these build tools see [25, 93, 94]. The OLR4 project website [114] provides detailed information on how to build OLR4 with Maven.

Apache ANT is a Java based project build automation tool similar to GNU's *make*. It can be configured via an XML build script file and extended by adding custom Java classes called ANT *tasks*. A large number of so called *core tasks* [26] is available with ANT. These *core tasks* perform actions like copying, moving, and renaming of files, compiling Java sources, invoking *javadoc* generation, searching and replacing in files, opening database connections and many more. These tasks are invoked in a so called *ANT build script*. It is unfortunately common practice to write ANT build scripts performing the required tasks for each project depending on the project's directory layout and structure. However, it is obvious that using a standardized directory and project structure would enable to reuse already existing *ANT build scripts* in many projects. This is the main vision of Apache Maven.

Maven is a Java project management and project comprehension tool. Maven is based on the concept of a project object model (POM). All artifacts produced by Maven are a result of consulting a well defined model for your project. Builds, documentation, source metrics, and source cross-references are all controlled by your POM. [95].

In other words, Maven uses a central XML based project descriptor [92], which contains project information about infrastructure resources for instance version control and bug tracking, web space resources, where files and documentation should be uploaded, version information, branch information, mailing list

addresses, library dependencies, software license and copyright information, contributors, and some project specific information concerning the build.

Maven is based on ANT, i.e. it also offers all features provided by ANT. Since Maven assumes that all projects have the same project directory layout, it can execute the provided build scripts for all projects, which are setup and described with Maven. Further advantages of using Maven are:

- Maven generates and uploads a website containing extracted information from the Maven project descriptor, additional documentation provided by the specific project, code metrics, change log information from CVS, javadoc and an HTML code cross reference.
- Maven resolves and downloads third party library dependencies defined in the project descriptor.
- Maven stores all downloaded dependency libraries in a local central library repository and shares them over multiple Maven projects. This reduces file redundancies caused by third party libraries used in multiple projects.
- Maven builds, packages and deploys hole software releases with one command.
- Maven generates project files for different Java IDEs like Eclipse [45] or IntelliJ IDEA [68].
- Maven automatically executes JUnit [77] test.
- Maven supports the integration of persistence layers like Apache Torque [143] or Hibernate [64] into the build process, i.e. it can be used to generate Java classes forming an automated persistence layer.

It was easy to transform OLR3 into a Maven project. For this purpose, we first created an empty Maven project [50] and moved OLR3 files from the old directory structure to the one generated with Maven. Then, we customized the Maven project descriptor to fit the new OLR4 project (e.g. the library dependencies, infrastructure resources, etc). The OLR Maven project descriptor can be viewed in OLR's code repository [111].

In addition to the functionalities provided by Maven, the XMLC compiler has to be invoked to generate the HTML Java stubs [2] and the database setup has to be integrated. Similar to the ANT *tasks*, Maven uses the term *goal*, which is a kind of *ANT script snippet*. For OLR3 we extended the Maven build script to invoke the XMLC compiler before the compilation of the Java source files is executed.

[2] used by OLR3 to dynamically generate HTML pages.

Documentation The OLR Maven project contains installation [115] and reference [116] documentation which makes it possible to *get started with OLR in 10 minutes*. The documentation is part of the OLR project website [114] generated by Maven.

Database Setup Automation As mentioned above, OLR3 provides no setup procedures or SQL scripts to create the OLR3 database tables and to create basic course data. The scripts to create tables and data are discussed in 5.2.1. In the following we show how the database scripts are integrated into the OLR build automation. Since we use Maven, it seems obvious to extend the OLR Maven build script with the ANT SQL task in order to execute the scripts in a database. But we decided to use a plain ANT build script to perform the SQL tasks, because this build script is also shipped to the end user in the OLR binary version. This build script enables the possible automated creation of the database tables and initial course data. The ANT build script reads the database settings defined in a properties file and uses the ANT SQL task to execute an SQL script, which creates the needed database tables. A further SQL script is then executed to insert sample course and bootstrap data into the database. Furthermore, the ANT SQL task opens a JDBC [73] connection to the database. The ANT script and the database properties file can be viewed in OLR's code repository [111]. The database setup is also documented on the OLR project website [115].

5.3 Outlook

Our Open Learning Repositories link to learning resources on the Internet using URLs, which implies the problem of the durability of the courses in our OLRs. As soon as an URL changes - even if it is not the nature of URLs to change - courses referencing to these resources will break. The durability of courses depends on the durability of resources the courses link to. In the following, we will introduce systems and services, which address the concern of the durability of online resources and describe how these systems are related to the OLRs environment. We will also briefly describe the role of OLRs in a peer-to-peer network.

Resource Handle System Handle.net offers services and infrastructure to assign so called handles to resources. The handles are URLs, which will never change and are, like IP addresses, resolved on request to the real URL. *"The Handle System is a comprehensive system for assigning, managing, and resolving persistent identifiers, known as "handles" for digital objects and other resources on the Internet. Handles can be used as Uniform Resource Names (URNs). The Handle*

System enables to store handles of digital resources and resolve those handles into the information necessary to locate and access the resources. This associated information can be changed as needed to reflect the current state of the identified resource without changing the handle, allowing the name of the item to persist over changes of location and other state information." [61, from the introduction to the Handle System]

The Handle System is based on RFC 3651[3] [127, Handle System Namespace and Service Definition].

Durable Resource Repositories If all important resource repositories on the Internet would assign handles as described above to their resources, the durability of those resources and of OLR courses could be increased. DSpace is a digital repository, which uses the Handle System to conserve digital resources. "*DSpace is a groundbreaking digital library system to capture, store, index, preserve, and redistribute the intellectual output of a university's research faculty in digital formats.Developed jointly by MIT Libraries and Hewlett-Packard (HP), DSpace is now freely available to research institutions world-wide as an open source system that can be customized and extended.*" [42, DSpace website].

Figure 5.3 shows a possible OLRs environment, which supports durable online courses based on the Handle System and DSpace repositories. The picture also implies that handles are also assigned to URLs for courses and other resources provided by an Open Learning Repository.

Open Learning Repositories in P2P-Networks Our Open Learning Repositories describe structure and content of courses using RDF metadata. Learning resources, however, are not included directly into the courses. They are just referenced by the courses using hyperlinks. If an author wants to describe or create a course using a system like OLR4, a main task is to search and find high quality learning resources on the web. Resources, OLR4 courses link to, are often hosted and distributed over universities (and other institutions) all over the world. The metadata providers and consumers can be considered as peers within a highly dynamic and distributed peer-to-peer network. In this environment, our Open Learning Repositories may play the role of potential metadata providers.

Parallel to our research on the Open Learning Repositories, we work closely with Research Center L3S [5] on a P2P project called Edutella [104]. The Edutella project provides P2P services and infrastructure for peers to provide and to query

[3]The Requests for Comments (RFC) document series is a set of technical and organizational notes about the Internet (originally the ARPANET), beginning in 1969

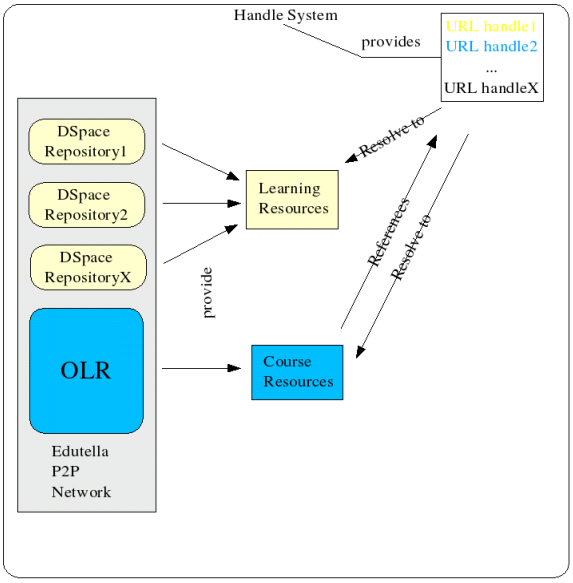


Figure 5.3: OLR4 Environment for Durable Courses

RDF metadata shared in the network. In this Edutella P2P network, OLR3 and OLR4 play the role of metadata provider-peers.

In order to make this metadata available for queries and exchange in an Edutella P2P network, the Open Learning Repositories need to provide an interface for the Edutella P2P services to retrieve and process queries on the stored RDF metadata. This interface based on the McBride database schema, which was used by OLR2, OLR3 and also in the current OLR4 system, is already implemented. The implementation of the Edutella interface is a simple mapping of Edutella QEL queries to SQL queries. The implementation assumes that a special database view, the *edutella_view* for the McBride schema exists. This view is part of the OLR4 SQL sources. The OLR Edutella provider can be started as a separate service. Further details about the implementation of the OLR's interface to Edutella can be found on the Edutella homepage [46].

Chapter 6

Processing and Optimization of Complex Queries in Schema-Based P2P Networks

6.1 Introduction

We have investigated, so far, the management and use of metadata, especially learning (educational) metadata, in the context of our "local" Open Learning Repositories. Thereby, we discussed metadata modeling languages as well as learning metadata standards from an interdisciplinary perspective. Then, the Open Learning Repositories - the framework and testbed for our discussions and proposed approaches and ideas - with their various focuses have been presented. At the end of Chapter 5, we briefly introduced the potential integration of our OLRs in the P2P infrastructure Edutella, where they may play the role of metadata providers.

In the following, we will generalize the learning metadata issues, particularly metadata management and search, from the local OLRs to the distributed environment of P2P networks. Moreover, we will consider the broadly used (not only educational) metadata that annotate any resource on the Web. Our focus in this second part of our work is the optimization of the metadata management and search.

Peer-to-Peer infrastructures are emerging as one of the important data management infrastructures in the World Wide Web. So far, however, most work has focused on simple P2P networks, which do tackle efficient query distribution to a large set of peers, but assume that each query can be answered completely at each peer. For queries that need data from more than one peer to be executed this is clearly insufficient. Unfortunately, though quite a few database techniques can be re-used in the P2P context, P2P data management infrastructures pose additional

challenges caused by the dynamic nature of these networks. In P2P networks, we can assume neither global knowledge about data distribution, nor the suitability of static topologies and static query plans for these networks. Unlike in traditional distributed database systems, we cannot assume complete information schema and allocation schema instances, but rather work with distributed schema information, which can only direct query processing tasks from one node to one or more neighboring nodes.

P2P computing provides a very efficient way of storing and accessing distributed resources, as shown by the success of music file sharing networks, such as Gnutella, where simple attributes are used to describe the resources. A lot of effort has been put into refining topologies and query routing functionalities of these networks. A new breed of P2P applications inspired from earlier systems like Napster and Gnutella has more efficient infrastructures such as the ones based on distributed hash tables. Less effort has been put into extending the representation and query functionalities offered by such networks. Projects exploring more expressive P2P infrastructures [20, 30, 59, 104] have only slowly started the move toward schema-based P2P networks.

In the Edutella project [46, 104] we have been exploring several issues arising in that context, in order to design and implement a schema-based P2P infrastructure for the Semantic Web. Edutella relies on the W3C metadata standards RDF and RDF Schema (RDFS) to describe distributed resources, and uses basic P2P primitives provided as part of the JXTA framework [56].

Related Work Although distributed query optimization and execution are well known issues investigated in database research, distributed query processing in schema-based P2P networks is novel. Middleware systems, e.g., Garlic [76], have been used to overcome the heterogeneity faced when data are dispersed across different data sources. In [89] central mapping information of all participating is used to provide access to distributed data sources. [118] introduces so called mutant query plans, which encapsulate partially evaluated query plans and data. Loss of pipelining during execution limits the general applicability for distributed query processing, and no user-defined operators are supported. AmbientDB [31] executes SQL queries over a P2P network. The approach is based on distributed hash tables and does not take into account user-defined operators.

Very recent work of Stuckenschmidt et al. [138] exploits schema paths for optimizing queries on distributed RDF repositories. Their approach constructs the overall query plan in a mediator-like manner and uses replicated schema paths (which serve as a global allocation schema of the data) to determine which portions of the query plan can be pushed to the data sources. The approach does not handle

the case that individual portions of the pushed query plan can be further distributed. In a highly distributed environment like a P2P network it is, however, a scalability concern to assume global knowledge of the allocation schema. For example, the update behavior of the join indices will be a problem in such an environment, as new data sources with new RDF properties joining the network will lead to an enormous growth of all join indices and huge transfer costs. Our approach addresses in particular load balancing strategies during query plan generation and mechanisms for the dynamic placement of operators. Their query processing facilities are limited to joins and selections. User-defined operators are not considered but needed in case multiple resources contribute data to the same property, which potentially leads to an enormous explosion of the search space.

Motivation To enable dynamic, extensible, and distributed query processing in schema-based P2P networks, where we need to evaluate complex queries requiring data from several peers and where both standard query operators and user-defined code can be executed nearby the data, we have to distribute query processing to the (super-)peers. Since each peer in a P2P network usually has varying resources available, e.g., regarding bandwidth or processing power, exploiting the different capabilities in a P2P network can lead to an efficient network architecture, where a small subset of peers, called super-peers [153], takes over specific responsibilities for peer aggregation, query routing, and mediation.

In such an architecture, super-peers can, on the one hand, provide query processing capabilities, and on the other hand functionality for the management of index structures and for query optimization. Super-peer based P2P infrastructures are usually based on a two-phase routing architecture, which first routes queries in the super-peer backbone, and then distributes them to the peers connected to the super-peers. Our routing mechanism is based on two distributed routing indices storing information to route within the super-peer backbone and between super-peers and their respective peers [105]. The query processors at the super-peers can be dynamically extended by special-purpose query operators that are shipped to the query processor as part of the query plan. In this way, query evaluation plans (QEPs for short) with user-defined code, e.g., selection predicates, compression functions, join predicates, etc., can be pushed from the client to the (super-) peers where they are executed.

Furthermore, super-peers have to provide an optimizer for dynamically generating good query plans from the queries they receive. We utilize these distributed query processing capabilities at the super-peers and distribute the user's query to the corresponding super-peers. This distribution process is guided by the (dynamic) distributed routing indices, which correspond to the (static) data allocation schema

in traditional distributed DBMSs. However, as the index is dynamic and itself distributed over the super-peers, static query optimization as used in distributed DBMSs is not possible. Query optimization must be therefore be dynamic and based on the data allocation schema known at each super-peer.

In this chapter, we first describe briefly our super-peer based topology and schema-aware distributed routing indices extended with suitable statistics and describe how this information is extracted and updated. Second, we show how these indices facilitate the distribution and dynamic expansion of query plans. Third, we propose a set of transformation rules to optimize query plans and discuss different optimization strategies in detail, enabling efficient distributed query processing in a schema-based P2P network.

6.2 Distributed Routing Indices

Efficient query routing is one of the corner stones of advanced P2P systems. We rely on a super-peer topology with "schema-aware" routing indices.

The HyperCuP Topology

Super-peers are arranged in the HyperCuP topology. The HyperCuP algorithm as described in [131] is capable of organizing super-peers of a P2P network into a recursive graph structure called a hypercube that stems from the family of Cayley graphs. Super-peers join the HyperCuP topology by asking any of the already integrated super-peers which then carries out the super-peer integration protocol. No central maintenance is necessary for changing the HyperCuP structure. The basic HyperCuP topology enables efficient and non-redundant query broadcasts. For broadcasts, each node can be seen as the root of a specific spanning tree through the P2P network. Peers connect to the super-peers in a star-like fashion. Figure 6.1 shows an example super-peer based P2P network.

Routing Indices

Our super-peers [105] employ routing indices which explicitly acknowledge the semantic heterogeneity of schema-based P2P networks, and therefore include schema information as well as other possible index information. The indices are local in the sense that all index entries only refer to direct neighbors (peers and super-peers). Network connections among the super-peers form the super-peer backbone that is responsible for message routing and integration/mediation of metadata.

Our super-peer network implements a routing mechanism based on two indices storing information to route within the P2P backbone and between super-peers and

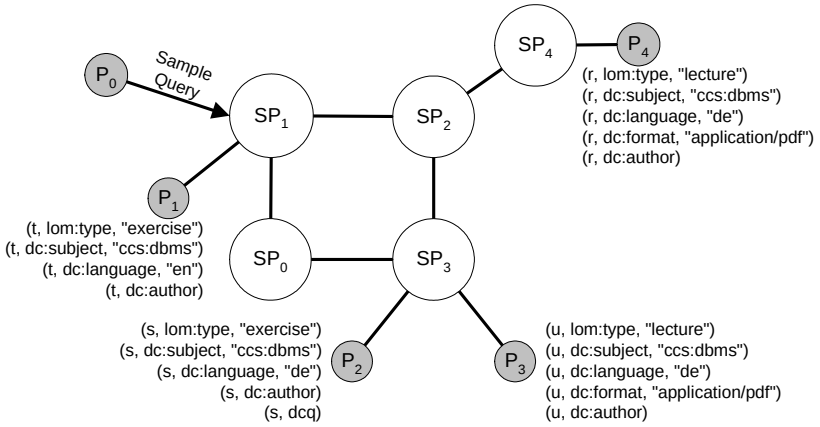


Figure 6.1: Routing Example Network

their respective peers.

Super-Peer/Peer Routing Indices The super-peer/peer routing indices (SP/P indices) contain information about each peer connected to the super-peer, including schema and attribute information from the peers. On registration the peer provides this information to its super-peer. In contrast to other approaches (Gnutella, CAN [124]), our indices do not refer to individual content elements but to peers (as in CHORD [137]). The indices can contain information about peers at different granularities: schemas, schema properties, property value ranges and individual property values. Details are described in [105]. Using indices with different granularities enables us to state queries at different levels of accuracy.

Super-Peer/Super-Peer Routing Indices In order to avoid backbone broadcasting, we use super-peer/super-peer routing indices (SP/SP indices) to forward queries among the super-peers. These SP/SP indices are essentially extracts and summaries from all local SP/P indices maintained in the super-peers. Similar to the SP/P indices they contain schema information at different granularities, but refer to the super-peers' neighbors in the super-peer backbone. Queries are forwarded to super-peer neighbors based on the SP/SP indices, and sent to connected peers based on the SP/P indices.

Update of the SP/P Index An update of the SP/P index of a given super-peer occurs, when a new peer registers, a peer leaves, or the metadata information of a registered peer changes.

Peers connecting to a super-peer have to register their metadata information at this super-peer thus providing the necessary schema information for constructing

the SP/P and SP/SP routing indices. During registration an XML registration message encapsulates a metadata-based description of the peer properties. A peer must register at least one schema (e.g., the DC Element Set or the LOM Standard) with a set of properties (possibly with additional information) or with information about specific property values.

If a peer leaves the super-peer, all references to this peer have to be removed from the SP/P index of the respective super-peer. The same applies if a peer fails to re-register periodically. In the case of a peer joining the network or re-registering, its respective metadata/schema information are matched against the SP/P entries of the respective super-peer. If the SP/P routing index already contains the peers' metadata, only a reference to the peer is stored in the index otherwise the respective metadata with references to the peer are added to the index. The following algorithm formalizes this procedure:

We define S as a set of schema elements[1]: $S = \{s_i \| i = 1...n\}$. The super-peer SP_x already stores a set S_x of schema elements in its SP/P index. The SP/P index of a super peer SP_x can be considered as a mapping $s_i \mapsto \{P_j \| j = 1...m\}$. A new peer P_y registers at the super peer SP_x with a set S_y of schema elements.

1. If $S_y \subseteq S_x$, then add P_y to the list of peers at each $s_i \in S_y$.
2. Else if $S_y \setminus S_x = \{s_n, ..., s_m\} \neq \emptyset$, then update the SP/P index by adding new rows $s_n \mapsto P_y, ..., s_m \mapsto P_y$.

Update of the SP/SP Index Let us first consider how to update the SP/SP indices in the backbone, when one of them has been modified. We assume, that SP/P modifications are collected for some period and trigger the update process for SP/SP indices periodically, if necessary. Super-peers in the network are organized into a HyperCuP topology, which implicitly defines each super-peer as root of a spanning tree. Query routing takes place along the spanning trees (restricted by the SP/SP indices), so the update of SP/SP indices has to be done in the reverse direction. For these updates, again each super-peer acts as the root of a spanning tree (in the "backward direction"), as shown in Figure 6.2 for the super-peer G. In this example we have a simple (complete) cube, which has three dimensions (0,1,2), such that every node has 3 neighbors. In order to update the SP/SP indices after an update of the SP/P index of the super-peer SP_x we build the spanning tree of the super-peer SP_x as follows: SP_x sends the update message to all its neighbors, tagging it with the edge label (dimension) on which the message was sent. Super-peers, receiving the message, update their SP/SP index accordingly and forward the update message, but only to those super-peers tagged with lower edge labels. Furthermore,

[1] A complete schema, e.g. dc, is also considered as schema element

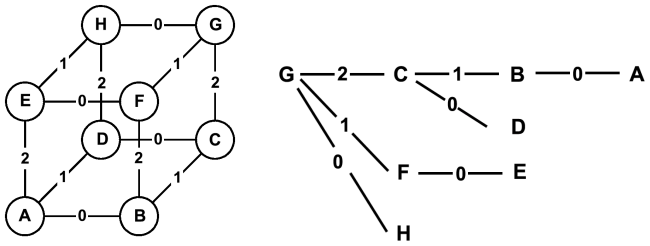


Figure 6.2: HyperCup Topology and Spanning Tree Example

whenever a message does not change the SP/SP index at a receiving super-peer SP_y, forwarding stops. The update is done as follows:

- For all $s_i \in S_x \cap S_y$ add dimension of SP_x to the list of dimensions at row s_i if this dimension does not exist.
- For all $s_i \in S_x \setminus S_y$ add a new row $s_i \mapsto dimension(SP_x)$

Adding a new super-peer is a bit more complicated. For a new super-peer, the HyperCuP protocol takes care of identifying new neighbors as discussed in [131]. In this process one of the super-peers is "responsible" for integrating the new super-peer. In most cases the new super-peer will fill a "vacant" position in the hypercube, which has temporarily been administered by the responsible super-peer. In this process, this super-peer, who has been holding an additional SP/SP and SP/P index for the vacant position, transfers these indices to the new super-peer. If the new super-peer opens a new dimension, it has to take over some peers from the old super-peer, and the SP/SP index has to be split. The neighboring super-peers have to update their indices accordingly, by exchanging the responsible super-peer with the new super-peer on the appropriate dimension. Beyond the immediate neighbors, no further update is necessary. The HyperCuP protocol also takes care of super-peers leaving the backbone. We usually assume that the leaving super-peer coordinates this operation, and specifically asks appropriate super-peers that will administer its position afterwards. In this process the administering super-peers take over the SP/SP and SP/P indices of the leaving super-peer, and the neighbors of the leaving super-peer as well as of the administering ones have to update their SP/SP indices. Again, no update is required beyond the immediate neighbors. Peers of the leaving super-peer reconnect to the administering super-peer.

In the case of unexpected link failure its neighbors determine the "closest" (regarding smallest hop distance) super-peer. This super-peer then coordinates the administration of the open position with the same procedure as described above. Peers of the failing super-peer have to reconnect at some other super-peer, possibly triggering further SP/SP update messages.

Statistics in the Routing Indices

The routing indices, as described so far, enable the efficient routing of queries. Nevertheless, additional information (statistics, physical parameters of the network, etc.) both in the SP/P and the SP/SP routing indices are necessary to enhance the optimization process and enable the choice of the best query execution plan. As mentioned in the introduction, we aim at using approved techniques and methods in databases, particularly from distributed database systems. The most important parameters for query optimization in this context are number and size of the stored documents at the different peers. This information is provided by the peers during the registration process. The following piece of the RDF-Schema *P*eerDescription shows the definition of the property elementCount, used for the documents count at a given peer, at the property-value level.

```
(...) <rdf:Property rdf:ID="elementCount">
    <rdfs:isDefinedBy rdf:resource=
    "http://www.learninglab.de/.../PeerDescription#"/>
    <rdfs:label>elementCount</rdfs:label>
    <rdfs:comment>An integer that specifies how often an element has occurred.
    Used in conjunction with hasPropertyValues.</rdfs:comment>
    <rdfs:range rdf:resource="http://www.w3c.org/.../rdf_schema#Literal"/>
    <rdfs:domain rdf:resource="#Peer"/>
</rdf:Property/>
(...)
```

If we register documents only at the property value level, we can derive the information for the property level by accumulating the number and size of documents for each property. Multi-valued properties like dc:author complicate this aggregation. Histograms [69] can help to obtain more precise estimates. We assume that the registration occurs at property level, property value level, and property value range level. The schema level can be considered as meta-level, which can be used to answer general queries (e.g. “Which standards are used to annotate documents at Peer P_x?”). Thus, the information about the number and size of documents are not relevant at this level. Table 6.1 shows the SP/P routing index of super-peer SP_1 including statistics at different granularities. In the following, we will restrict the discussion on the size (si) and the number (n) of available documents. However, it is easily possible to add further useful statistics such as minimum, maximum, and average values and the total number of documents at each peer. If a peer P_y (re-)registers or leaves a given super-peer SP_x with a schema element set including document statistics $S_y(s_1(n_1, si_1), \ldots, s_m(n_m, si_m))$, an update of the SP/P and the SP/SP indices is needed. The algorithm for building and updating the SP/P routing indices, described before, remains unmodified. The peers simply register including their statistics information in addition to the schema elements. The up-

Granularity	**SP/P Index of SP_1**		
Schema	dc		P_1, P_0
	lom		P_1, SP_3
Property	dc:subject		P_1 [13], P_0 [16]
	dc:language		P_1 [15]
	lom:type		P_1 [10]
Property Value	lom:type	"exercise"	P_1 [10]
	dc:language	"de"	P_0 [15]

Table 6.1: SP/P Index of SP_1 at Different Granularities

date information of the SP/SP indices propagated via messages however must be extended as follows:

1. SP_x derives the total number and size of the documents (and potentially further statistics) registered by the peers for each schema element $s_i \in S_y$ and sends these statistics combined with s_i to its neighbors in its spanning tree.

2. Any other super-peer in the spanning tree of SP_x updates its SP/SP index and derives the total number and size of the documents in its SP/SP index at each $s_i \in S_y$ and forwards the data to its neighbors.

6.3 Query Processing

Using the indices described in the previous section we can now describe how query plan generation and distribution proceeds in our P2P network.

6.3.1 Distributed Plan Generation

In contrast to traditional distributed query optimization approaches, we cannot generate the query plan statically at one single host. Therefore, we have to generate an abstract query plan at a super-peer which is partially executed locally and where we push other parts of the query plan to its neighbors. The plan generation at each super-peer, therefore, involves five major steps as depicted in Figure 6.3 and is described in details in [34].

First, the received query (stated in our SQL dialect) is parsed and transformed into an internal representation which is a decomposition of the query into its building blocks. Then, the local indices are consulted to determine the location of the required resources. For this purpose we have to distinguish between resource directions (RDs) and physical resources (PRs). Users specify the desired information by giving properties and property-values restricting logical resources (LRs). These

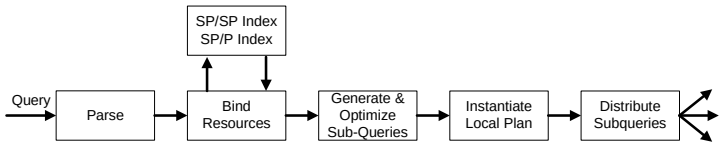


Figure 6.3: Plan Generation at a Super-Peer

LRs are bound to RDs resp. PRs where all levels of granularity of the indices have to be considered. Multiple RDs and PRs can contribute data for the same LR. Based on the bindings, a local query plan is generated. As super-peers have a very limited view of the whole P2P network (only the neighbors are known), it is obvious that no comprehensive static plan in the traditional sense can be produced. Therefore, we determine which sub-plans have to be delegated to the neighboring (super-)peers. The remaining parts constitute the input to the local plan. To perform cost based optimization, the optimizer uses statistics of the input data, the network topology, and the hosts. The optimizer may collect and use response times, transfer rates, and even result sizes from previous query executions. Finally, the local query plan is instantiated at the super-peer, all user-defined code is loaded, the communication path to the super-peer which uses this part of the query plan as input is established, and the remaining sub-queries are distributed to the corresponding super-peers, where they are processed further.

6.3.2 Query Optimization

Let us now describe some of the details involved in the optimization process at a super-peer. We employ a transformation-based optimizer starting with an initial query plan. The optimizer applies equivalence transformations and determines the cost of the generated alternatives using a cost model. In contrast to bottom-up approaches employed in traditional dynamic programming based optimization we can stop at any time with a complete and valid query plan. In our implementation we use iterative improvement to enumerate plan alternatives. Superior techniques as shown in [135] are applicable.

In the following, we present the set of the most important transformation rules, focusing on the ones relevant to processing joins and unions within the P2P context. Further rules can be added easily. Furthermore, we extend conventional cost models taking the special requirements of P2P query processing into account. During the optimization process we employ heuristics that favor query plans with few sub-plans as this leads to more robust distributed query execution. A huge number of wide spread sub-plans accessing the same documents would be more error-prone and often inefficient to execute. Our decision also implies, that less messages are exchanged between the (super-) peers and less data is transferred.

The Initial Query Plan

The initial (canonical) query plan accesses only logical resources and is constructed in the following way: Use all join predicates and join the logical resources. If logical resources could not be joined due to a lack of join predicates, the Cartesian product includes them into the query plan. Thereafter, all remaining selection predicates and user-defined filters are applied on top of the query plan. Finally, the result is submitted to the client.

The Transformation Rules

The initial query plan is optimized top-down using a transformation-based optimizer. In such an approach we apply a set of transformation rules to the query plan and generate alternatives, which are then ranked using our cost model. The best (local) query plan is executed. Transformation rules are represented as

$$\frac{\{\text{inputQEP}\} \quad [\text{condition/action}]}{\{\text{outputQEP}\}}$$

where one input query plan is transformed into an output query plan. The condition/action part may be omitted. We assume that the transformations are executed at host H_L. If H_L is a super-peer, we have access to the local routing indices *SPP* and *SPSP*.

Basic Transformation Rules We can express the *Bind Resources* step explained in the previous subsection as the following *Binding Transformation*:

$$\frac{\{LR\} \quad [PR_j@P_j \in \textit{match}(\textit{SPP}),\, RD_k@SP_k \in \textit{match}(\textit{SPSP})]}{\left\{\bigcup_j PR_j@P_j \cup \bigcup_k RD_k@SP_k\right\}}$$

The function *match* consults the local indices and determines the location of the matching resources. The LRs are bound to RDs, if a corresponding data source is found in the SP/SP index. Using the SP/P index, LRs are bound to PRs, i.e., the URIs of registered resources. Multiple RDs and PRs can contribute data for the same LR. This is expressed by the union of RDs and PRs. $PR_j@P_j$ denotes that the j-th bound PR belongs to the corresponding LR and references a resource at peer P_j. A similar argument applies for the RDs.

Applying the following two transformations to a query plan pushes selections and user-defined filters down towards the data sources. This enables us to reduce the amount of transferred data early.

$$\frac{\{\sigma(A\ op\ B)\}}{\{\sigma(A)\ op\ \sigma(B)\}} \qquad \frac{\{\sigma(op(A))\}}{\{op(\sigma(A))\}}$$

Here, A and B are arbitrary sub-plans.

The next two rules apply the associative and commutative laws to unions, joins, and Cartesian products.

$$\frac{\{(A\ op\ B)\ op\ C\}\ \ [op \in \{\cup, \bowtie, \times\}]}{\{A\ op\ (B\ op\ C)\}} \qquad \frac{\{A\ op\ B\}\ \ [op \in \{\cup, \bowtie, \times\}]}{\{B\ op\ A\}}$$

Again, A, B, and C denote arbitrary sub-plans.

Finally, each operator is annotated with the host where it is to be executed. This is done bottom up from the leaves of the operator tree which constitute PRs and RDs. The annotations of the leaves are given by the first transformation rule. An operator can be executed on host H_L, if all its inputs are computed at H_L.

$$\frac{\{A@H_1\ op\ B@H_2\}\ \ [H_1 \neq H_2]}{\{A@H_1\ op@H_L\ B@H_2\}} \quad \frac{\{A@H_1\ op\ B@H_2\}\ \ [H_1 = H_2]}{\{A@H_1\ op@H_1\ B@H_1\}} \quad \frac{\{op(A@H_1)\}}{\{op@H_1(A@H_1)\}}$$

A and B are sub-plans and $op@H_1$ indicates that the operator op is executed at host H_1. This rule enables us to execute mobile code at remote hosts, e.g., to push selective filter predicates, complex join predicates, or compression functions to the data sources.

The plans generated by the rules so far typically have one union operator for each logical resource. The degree of parallelism can be increased and distributed computing resources can be utilized better if operators are distributed over the P2P network.

Optimization Strategy: Union of Joins As shown above, several PRs and RDs can contribute data for the same LRs. The simplest way for incorporating the data for such an LR would be to union all the accessed physical resources before any other operation is considered for that LR. This would be done by the binding transformation. This naive strategy would produce good plans in some cases, but query optimization would be limited and possibly better plans might never be considered. Thus, several alternatives for the naive query plan must be considered by applying equivalence transformations. To increase the degree of distribution, the query plan can be transformed using the following transformation which turns the join of unions into a union of joins:

$$\frac{\{(A_1 \cup \ldots \cup A_n) \bowtie (B_1 \cup \ldots \cup B_m)\}}{\{(A_1 \bowtie (B_1 \cup \ldots \cup B_m)) \cup \ldots \cup (A_n \bowtie (B_1 \cup \ldots \cup B_m))\}}$$

If many RDs and PRs are bound to LRs and when this rule is applied recursively in

combination with the associative and commutative laws the number of plans which have to be considered during query optimization is huge. [32] has derived a lower bound for the number of alternatives when joining two LRs, consisting of n_1 and n_2 bound resources:

$$\begin{aligned} \text{UJ}(n_1, n_2) &= \sum_{j=1}^{n_1} \left(\left\{ \begin{matrix} n_1 \\ j \end{matrix} \right\} \text{bell}(n_2)^j \right) \\ &+ \sum_{j=1}^{n_2} \left(\left\{ \begin{matrix} n_2 \\ j \end{matrix} \right\} \text{bell}(n_1)^j \right) - \text{bell}(n_1)\text{bell}(n_2) \end{aligned}$$

In this definition $\left\{ \begin{matrix} m \\ k \end{matrix} \right\}$ denotes the Stirling number of the second kind which represents the number of ways a set with m elements can be partitioned into k disjoint, non-empty subsets. The term bell(m) denotes the Bell number which represents the number of ways a set with n elements can be partitioned into disjoint, non-empty subsets. The definition of *UJ* follows the construction of a query plan starting from its canonical form. First we have to select a LR constituting of different bindings. Each such binding has to be joined with an expression which is equivalent to the other LR. All these expressions are counted by the call to the function for the Bell numbers. At the end we have to consider duplicate QEPs which are generated when for every appearance of a LR in a QEP the same partitioning is selected. If the same partitionings are selected, the order in which the LRs are used in the construction of a QEP does not matter anymore. Therefore, the last term of the definition of *UJ* includes the number of QEPs with that property. Table 6.2 gives an impression of the search space explosion, the generated plan may have a huge number of sub-queries.

configuration	number of plans
$UJ([2,2])$	8
$UJ([3,3])$	385
$UJ([4,4])$	144705
$UJ([5,5])$	913749304

Table 6.2: Explosion of the Search Space

Optimizing by Collecting Resources A very promising heuristics in a distributed environment is to collect as many bindings of one LR as possible at one host. To implement this strategy, the optimizer determines one “collecting host” to collect all data of one logical resource. Other hosts are informed to send all data to

$$\frac{\left\{\bigcup_j PR_j@P_j \cup \bigcup_k RD_k@SP_k\right\} \quad \left[H_C \in \bigcup_j P_j \cup \bigcup_k SP_k\right]}{\left\{CR(LR)@H_C \cup \bigcup_j^{H_C \neq H_j} CollectSend(H_C, LR)@P_j \cup \bigcup_k^{H_C \neq H_k} CollectSend(H_C, LR)@SP_k\right\}}$$

(a) Collecting Host Selection

$$\frac{\{CollectSend(H_C, LR)\} \quad \begin{bmatrix} PR_j@P_j \in \textit{match}(\textit{SPP}), \\ RD_k@SP_k \in \textit{match}(\textit{SPSP}) \end{bmatrix}}{\{CollectSend(H_C, LR)@P_j, \ldots, CollectSend(H_C, LR)@SP_k\}}$$

(b) Propagate CollectSend

$$\frac{\{CollectSend(H_C, LR)\} \quad \begin{bmatrix} PR_j@P_j \in \textit{match}(\textit{SPP}), \\ RD_k@SP_k \in \textit{match}(\textit{SPSP}) \end{bmatrix}}{\left\{Send(H_C, \bigcup_j PR^j@P_j \cup \bigcup_k RD_k@SP_k)@H_L\right\}}$$

(c) Execute CollectSend

$$\frac{\{CR(LR)\} \quad \begin{bmatrix} PR_j@P_j \in \textit{match}(\textit{SPP}), \\ RD_k@SP_k \in \textit{match}(\textit{SPSP}) \end{bmatrix}}{\left\{Receive@H_L \cup \bigcup_j PR_j@P_j \cup \bigcup_k RD_k@SP_k\right\}}$$

(d) Execute Collect Resource At Host

$$\frac{\{CR(LR)\} \quad \begin{bmatrix} PR_j@P_j \in \textit{match}(\textit{SPP}), RD_k@SP_k \in \textit{match}(\textit{SPSP}), \\ H_C \in \bigcup_j P_j \cup \bigcup_k SP_k, setForward(LR, H_C) \end{bmatrix}}{\left\{\begin{array}{l} CR(LR)@H_C \cup \bigcup_j^{H_C \neq H_j} CollectSend(H_C, LR)@P_j \\ \qquad \cup \bigcup_k^{H_C \neq H_k} CollectSend(H_C, LR)@SP_k \end{array}\right\}}$$

(e) Forward Collect Resource

Figure 6.4: Transformation Rules for the "Collect Resources" Strategy

the collecting host (in the following this is done by the CollectSend Operator). In contrast to the canonical query plan this collecting host is determined dynamically and may change during query execution, i.e., we can place the resource-collecting union at an arbitrary (super-) peer. In well clustered networks it is useful to place the collecting union operator nearby the majority of the data and to ship only a few resources.

To include this strategy in our query optimization, we introduce Collect Resources (CRs) which can be used in the previous rules like bound resources. Additionally, we propose the following five transformation rules (shown in Figure 6.4):

- First, the collecting host H_C is selected from the set of all referenced neighbors (taken from the PRs and RDs) (Figure 6.4(a)). Then, we replace all bound resources, i.e., PRs and RDs, of the input plan with a collect resource which is executed at H_C and CollectSend operators are pushed to the other neighbors. These CollectSend operators ship all data of the LR to the col-

lecting host H_C.

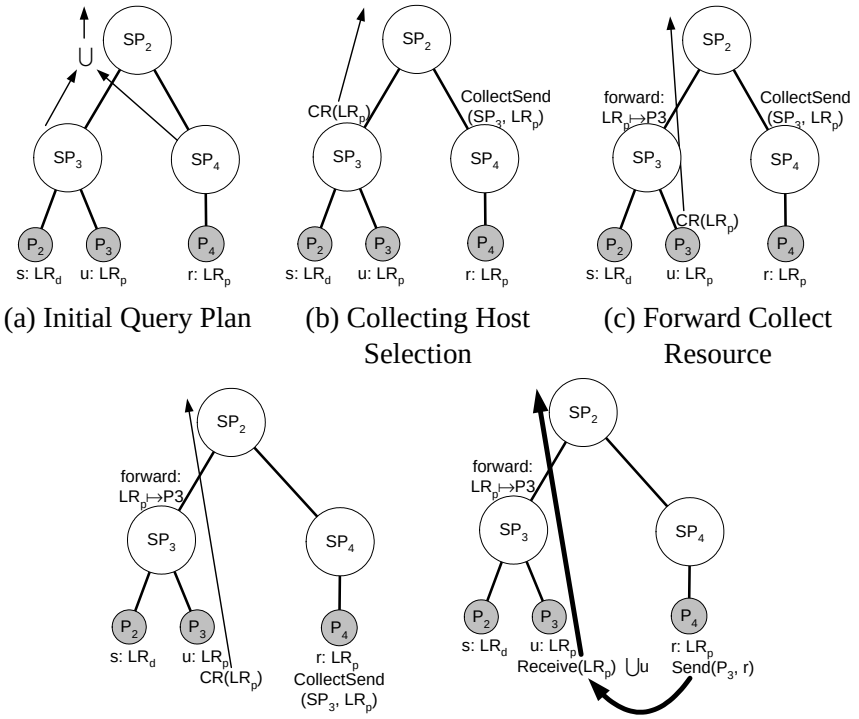
- When a CollectSend operator is received by a host, it can be propagated to all its matching neighbors (Figure 6.4(b)) which are determined from the local indices. The plan is split into multiple parts which are distributed broadcast-like to the neighbors.
- Hosts can also execute the CollectSend operator (Figure 6.4(c)). This is treated as a binding transformation where results are sent back to the collecting host.
- A collecting host can execute the CR operator by accepting resources belonging to the given LR (Figure 6.4(d)). The results are sent from sub-plans built by the latter two transformations. Additionally, resources are bound using the local indices.
- Finally, the CR operator can also be forwarded to a neighbor (Figure 6.4(e)). This means that first, we choose the new collecting host H_C from the neighbors and set an appropriate forward. The CR is pushed to H_C and all matching neighbors are instructed to send their data for LR to H_C. During query instantiation a CollectSend operator follows the forwards and creates a proper Send operator with the actual collecting host as target. Thus, results are sent directly to the correct host.

Figure 6.5 illustrates the rules querying resources of LR_p, i.e., the documents r and u. Starting at SP_2 as the local host with the initial query plan (Figure 6.5(a)), SP_3 is selected as collecting host of LR_p (Figure 6.5(b)) and a CollectSend informs SP_4 to send all documents regarding LR_p to SP_3. SP_3 decides to forward the CR to P_3 where the results are sent directly back to the initial caller (bypassing SP_3 and SP_2) (Figure 6.5(c)). SP_4, on its part, propagates the CollectSend operator to P_4 (Figure 6.5(d)). Finally, P_4 finds out by considering SP_3 to send the local resource r to P_3 and P_3 executes the CR operator and returns u and the received document r (Figure 6.5(e)).

Splitting and Distributing the Query Plan

Valid query plans must be completely annotated and all resources must be bound. The best query plan is split into a local plan and multiple remote query plans. The remote plans are shipped to the referenced hosts[2] where the optimization process

[2]Note, that these are always neighboring hosts.



(a) Initial Query Plan (b) Collecting Host Selection (c) Forward Collect Resource

(d) Propagate CollectSend (e) Execute Collect Resources and CollectSend

Figure 6.5: Example Applications of "Collect Resources" Accessing LR_p
(Thin lines demonstrate the query plan during instantiation, bold lines show the flow of results.)

continues on the smaller query plans. The local query plan is instantiated and combines the results of the remote query plans.

Algorithm 1 splits (in DFS manner) a QEP into the local plan Q_L and the remote plans. The remote plans are stored in the mapping Q_R from the host where to execute the remaining query parts onto the query plan itself. One remote host may execute multiple sub-plans. The recursive function is called with the top-level operator of the query plan. Then the child operators are examined. If a child is executed at the same host, i.e., the local host, the function is called recursively. Otherwise, this is the root of a remote sub-plan and a Send operator is put on top of the sub-plan including the child operator. The remote sub-plan is separated from the local plan and a Receive operator at the local host is responsible for the connection to the remote plan.

Algorithm 1 Splitting the Query Plan

```
1: Q_L = Q
2: Q_R = ∅
3: function splitPlan(op)
4:   for all childOp ∈ op.children do
5:     if childOp.host == op.host then
6:       splitPlan(childOp)
7:     else
8:       Q_R.put(childOp.host, Send(op.host, childOp))
9:       replace(Q_L, childOp, Receive(childOp.host))
10:    end if
11:  end for
12: end function
```

The Cost Model

Some of the parameters used for our cost model are stored within the local SP/P and SP/SP indices as described in Section 6.2, others are determined periodically by the runtime environment. In our distributed query processing environment we are interested in the plan with the lowest response time. Such a response time cost model was devised in [49] and explicitly takes parallelism and pipelining into account.

The most important parameters of query optimization in traditional databases are number and size of intermediary results. The same applies to P2P query processing, where we utilize the number of documents and the overall/maximum/minimum size of the registered resources for estimating the costs of a query plan. Our cost model also considers physical properties of the network, e.g., the network bandwidth, the latency, and the number of hops to the neighbors. But it is also important to know CPU and memory load of the local (super-) peer and the neighbors, as especially the super-peers are in danger of being overloaded, when too many queries execute operators at the super-peers. This would slow down query execution, so the optimizer should be aware of the current load situation of the local super-peer and the neighboring (super-) peers and generate alternative query plans, e.g. by using the "Collect" strategy, which enables the query optimizer to place operators on low loaded hosts. For these reasons, we utilize load information as one important parameter for the optimizer's cost model. Load collectors are used to collect data for the optimizer's view of the load situation of all relevant resources at the neighboring hosts. We measure the average CPU and memory load on (super-) peers and send the current situation to the neighbors.

The optimizer's view of the load situation is updated at intervals of several seconds to prevent overloading the network. Using this information the optimizer at each (super-) peer can decide whether a sub-plan can be pushed to a neighbor, or—in the case of an overload—an alternative query plan would produce faster results.

Additionally, adapting the techniques presented in [136], our cost model can be extended to take the response time of "similar" queries, i.e., queries accessing the same index entries, into account.

6.4 Implementation

The discussed techniques for processing and optimizing complex queries in the highly dynamic and open environment of schema-based super-peer networks are already implemented in Edutella. The Edutella System [46] which constitutes an RDF-based metadata infrastructure for JXTA [78] is an open source project written in java.

The organization of the super-peer backbone in the HyperCup-topology occurs dynamically. The distributed routing indices are also built and updated dynamically based on the registration files of the peers/super-peers.

We distinguish between *metadata statistics* such as document count and file size and *network statistic parameters*. The *metadata statistics* are automatically extracted from the registration files and stored in the SP/P and SP/SP routing indices. The *network statistic parameters* can be extracted at a given super-peer in an active way (e.g. memory load) by asking the neighboring super-peers or in a passive way by storing for example the response time of a given super-peer or peer. The statistics are currently used during the plan generation. The complex query processing modules are included in the package *net.jxta.edutella.complexquery*. We also implemented a subpackage *net.jxta.edutella.complexquery.graph* for the visualization of the QEPs. The subpackage *net.jxta.edutella.complexquery.work* includes all classes needed for the execution of the QEP's different steps.

The complex query processing techniques are also implemented in QueryFlow [81, 82] which is based on ObjectGlobe [33], building upon earlier work by some of the authors on distributed query processing.

6.5 Conclusion

Peer-to-Peer data management infrastructures are emerging as one of the important infrastructures for data intensive networks on the World Wide Web. In this chapter, we have investigated query distribution and query optimization issues for

schema-based peer-to-peer networks, which use complex and possibly heterogeneous schemas for describing the data managed by the participating peers. Specifically, we have focussed on addressing one particularly severe shortcoming of current peer-to-peer networks, namely that they are unable to handle queries which need data from several peers to compute answers.

Comparing P2P data management networks to conventional distributed and federated database systems, we have identified specific additional challenges which make it impossible to apply distributed database query planning and optimization techniques in a straightforward way. We have, therefore, specified an innovative query routing and planning architecture based on distributed routing indices managed by a suitably connected set of super-peers, which makes distributed query processing available also in P2P data management networks. We have discussed how to use transformation-based techniques for incremental query optimization at each super-peer, and specified a set of transformation rules, relevant for processing joins and unions in such a network. These techniques allow us to place query operators next to data sources and utilize distributed computing resources more effectively.

Future work will concentrate on the further investigation of simulations and experiments to evaluate and extend our current set of transformation rules. We will also evaluate the use of additional statistics useful as input to our query plan generation more intensively.

Chapter 7

Semantic Caching in P2P Networks

7.1 Motivation

In chapter 6 we showed how our super-peer based network topology provides better scalability than pure peer-to-peer networks. Super-peers can provide, on the one hand, query processing capabilities, and on the other hand functionality for the management of index structures and for query optimization. Furthermore, our super-peers provide an optimizer for dynamically generating good query plans from the queries they receive. Additionally to all these already described approaches, we propose in this chapter our recent work on semantic caching in P2P networks, which aims also at providing better scalability and optimizing query evaluation.

Caching is nothing new. Moreover caching is basic in computer science. Most modern computer systems use this principle in a number of places for many reasons, such as to improve the performance of the main processor(s), to speed up disk accesses, reduce the response time, and so on. Traditional cache strategies hit on their limits in the highly dynamic and open environment of P2P networks. Recently, there has been a growing interest in the so called semantic caching approach, especially in complex information environments like heterogeneous distributed databases. semantic caching can be simply defined as caching of semantic descriptions to the cached data.

In the following, we will first give an overview of the primary related work and theories. Semantic caching is related to a good deal of theories and research topics (e.g. query containment problems, answering queries using views, conventional caching) giving the problem a notable complex entrance. Then, we will present

our first implementation of the semantic caching approach in Edutella. Our investigation on semantic caching in P2P networks is an ongoing work and still at the beginning. Thus, there is a large room for improvement and much further work.

7.2 Preliminaries

7.2.1 Caching

Traditional Caching - Client-Side Caching In simple terms data caching is storing data in memory for quick access. There are two types of conventional caching: *page caching* and *tuple caching* ([24], [48]).

- *Page Caching:* Is widely used in operative and database management systems. It assumes that queries posed at the client can be processed locally and be broken down to the level of request for individual pages (defined groups of tuples). If a requested page is not available in the client cache, the page request is forwarded to the server. This presumes that the data organization at the servers is known. This assumption is not given in schema-based P2P networks.

- *Tuple Caching:* The cache is maintained in terms of individual tuples, and therefore, it is more flexible than page caching. Tuple caching is used for instance to cache web pages at proxy servers.

Semantic Caching The key idea of semantic caching is to remember the queries in addition to the query results. Semantic caching uses dynamically defined groups of tuples. We discern two types of semantic caching [91] [38]:

- *Semantic Query Caching (SQC)*: Can be considered as "just" a more precise appellation of semantic caching. *"It is an answer set stored as a relational table, labelled by the wuery that resulted in the answer set"*. [55]. Godfrey and Gryz presented in 1999 a general logical formalism for SQC.

- *Semantic Region Caching*: Is an extension of the semantic query caching with *semantic regions* [38], i.e., grouping of semantically related tuples (see cache replacement issues). The cache is managed as a collection of *semantic regions*.

Cache Replacement Issues A cache replacement policy defines the items designated for replacement when additional space is needed in the cache. This policy

assigns values to all items in the cache. The items with the lowest value are replaced. We distinguish in the traditional systems between *temporal locality*[1] and spatial locality. A "hard and fast" policy does not exist. Each of the replacement policies has both advantages and disadvantages. The choice of a certain replacement policy depends on the use case. In the following we describe briefly the most known replacement strategies.

- *FIFO - First In First Out*: The items which has been longest in the cache are replaced. This strategy does not take into account how often the items were referenced, i.e., the cache use is ignored.
- *Random*: The items to be replaced are chosen at random. This strategy does not consider the use or age of the items. The advantages of this strategy are the lowest implementation effort, the lowest resources consumption and the neutrality of treatment of all cache items. The crucial disadvantage is the ignorance of the cache use.
- *LRU - Least Recently Used*: This strategy is based on the temporal locality principle, i.e., the items which have been longest not used are replaced. The main drawback of this policy is the risk that cache content obsolesce.
- *LFU - Last Frequently Used*: This policy replaces the cache items, which are the least used. The outcome of this is the risk that covered content would be never updated.

In addition to these replacement policies there are some enhancements to avoid the disadvantages. However, these enhancements could increase the implementation efforts and the resources consumption.

- *TTL - Time To Live*: Solves the problem of obsolete content by adding either a global TTL to all cache items or individual TTLs to each item.
- *Semantic Regions*: Group together semantically related cache items. Thereby each tuple in the cache is associated to exactly one *semantic region*. Semantic regions are dynamically built based on the posed queries [38] [55]. The dynamic definition and rearrangement of the semantic regions causes a considerable overhead, which is not in all use cases warrantable or profitable. Furthermore, portions of a semantic region could imperceptibly obsolesce, since only use statistics on the whole region are available.

[1] The property states that items that have been referenced recently are likely to be referenced again in the near future. [38]

Cache Misses Issues In the following, we describe the approaches for treatment of cache-misses. These issues are described in [38] and [55] in the context of client-server architecture where caching and query processing take place at the client. We assume in our context that query processing and semantic caching take place at the super-peers.

- *Remainder Queries*: If the query posed at a given super-peer can be split (rewritten) in "cache answerable" and "cache non-answerable" part, the "cache non-answerable", called *remainder query*, is forwarded to the other super-peers in the backbone. If the query can be completely be answered by the cache, then we obtain a null remainder query, i.e., no communication with other super-peers is needed. The advantage of this approach is the lowering of the network load. However, the query rewriting implies an increased complexity.

- *Faulting*: Faulting considers the query as a whole. That is, if the complete answering of the query fails, the query is forwarded. The processing of this approach is very simple, but it causes a higher network load.

7.2.2 Conjunctive Query Containment

Conjunctive queries are the most common form of queries. They are equivalent in expressive power to SPJ (Selection, Projection, Join) queries in relational algebra [145] [23]. A conjunctive query is a first-order formula of the form ($A_1 \wedge A_2 \wedge ... \wedge A_m \xrightarrow{f} C$), i.e., (*antecedent*$\rightarrow$ *consequent*). $C, A_1, A_2, ..., A_m$ are atomic formulas in the form $Q(t_1, t_2, ..., t_n)$, where Q is a relation (predicate) symbol and $t_i, 1 \leq i \leq n$ is a variable. The predicate symbol in the consequent does not appear in the antecedent (i.e., recursion is not allowed). Variables that appear in the consequent are called distinguished and must appear in the antecedent as well, while all others are called nondistinguished. Finally, a conjunctive query has an associated unary function f. A *view* is a conjunctive query with a unique head predicate.

Conjunctive Query Containment A query Q_1 is said to be contained in a query Q_2 if, in every database instance, the set of answers to Q_1 is a subset of the set of answers to Q_2. The *conjunctive query containment* is defined as follows: For every conjunctive query there is a unique (up to renaming of variables) minimal equivalent query, and it is obtained from the original query by "combining variables". The problem of conjunctive query containment and minimization is NP-complete [36]. However, the containment problem is not a "hard" problem in

practical issues (short queries, few pairs of subgoals with same predicate) [145]. Adding negated subgoals and/or arithmetic subgoals makes the decidement problem more complex.

7.2.3 Answering Queries Using Views

The problem of answering queries using views (a.k.a. rewriting queries using views) is to find efficient methods of answering a query using a set of previously materialized views over the database, rather than accessing the database relations [60]. This problem is relevant to many application areas like data integration, web caching and query optimization.

The problem is known to be NP-complete in [60], even for conjunctive queries without built-in atoms. The main source of complexity is the fact that there are an exponential number of candidate rewritings that need to be considered. However, there are algorithms which generate efficient maximally-contained rewritings (MCR) of a conjunctive query using a set of conjunctive views.

Bucket Algorithm The bucket algorithm [90] looks at how views can "cover" each of the query subgoals and each variable of the query. The algorithm proceeds in two steps. First, the algorithm computes a bucket for each subgoal in the query. In the second step, it considers all the possible combinations of views, one from each bucket, and check whether it is a semantically correct plan, or it can be made semantically correct, if additional built-in atoms are added to the plan. Finally, each plan is minimized by removing redundant subgoals. Here are the bucket-construction rules [145]:

- For each subgoal R_i in query Q, create a bucket B_i
- If view v_j contains a subgoal R_k unifiable with R_i, then put $v_j\theta$ in bucket B_i, where θ is $mgu(R_i, R_k)$[2]
- Try all combinations $v_1, v_2, ..., v_k$ of bodies for $Q' : ans \longleftarrow v_1, v_2, ..., v_k$, where v_i is in bucket B_i.
- See if you can create some substitutions in Q' to get $Q'' \subseteq Q$. If yes, then the substituted Q' is a maximal solution to Q.

The bucket-algorithm reduces in its first step the number of query rewritings considerably, and this is done by considering each query subgoal in isolation. This benefit, however, turns out to be the main inefficiency of the bucket algorithm. By

[2] for definitions of unification and mgu's, see [21], pp. 293-294

treating each query subgoal in isolation the bucket algorithm does not consider the interactions between view subgoals. This leads to irrelevant views in the buckets and the second step becomes very expensive. A detailed illustration of the limitation of this algorithm can be found in [122].

Inverse Rules Algorithm The key idea of the inverse-rules algorithm [43]is to *invert* the views definitions, and then use these inverted rules to answer the original query. Inverse rules compute tuples of the database relations from tuples of the views. A rewriting of the query using the views is simply the query itself composed with the inverse rules. It works even when the query is a recursive datalog program. An important limitation of inverse rules is that they do not deal with interpreted comparison predicates. A detailed description and evaluation of the inverse-rules algorithm can be found in [122].

MiniCon Algorithm The MiniCon algorithm [122] aims at the finding the maximally-contained rewriting (MCR)of a conjunctive query using a set of conjunctive views. It begins like the bucket algorithm, i.e., it maps in the first step each query subgoal to a view subgoal, and determine if there is a partial mapping from the query subgoal to the view subgoal. Once the partial mappings are found, the algorithm focuses on variables rather than on subgoals. The subgoal g of a query Q is mapped to a subgoal h of a view V according to the following rules:

The set of such query subgoals that have to be mapped to subgoals from one view (and the mapping information) is called a *MiniCon Description* (MCD). The mapping from query Q subgoals to view V subgoals is defined based on the following general rules: For every query variable X that is mapped to a view variable A:

- *Case 1*: X is distinguished (head variable), A is distinguished. OK.
  - A is exported, i.e. join with other views is possible.
- *Case 2*: X is nondistinguished, A is distinguished. OK. (see case 1)
- *Case 3*: X is distinguished, A is nondistinguished. Not OK.
  - X needs to be in the answer, but A is not exported.
- *Case 4*: X is nondistinguished, A is nondistinguished.
  - All the query subgoals using X must be able to be mapped to other subgoals in view V. Since A is not exported in V, it is impossible for V to join with other views.

The MCDs (generalized buckets) that only overlap on distinguished view variables are then combined. Given a query Q, a set of views V, and the set of MCDs C for Q over the views in V, the only combinations of MCDs that result in non-redundant rewritings of Q are of the form $C_1, C_2, ..., C_l$, where

- $Subgoals(Q) = Goals(C_1) \cup Goals(C_2) \cup ...$
- $For every i \neq j, Goals(C_i) \cap Goals(C_j) = \emptyset$.

The MiniCon algorithm evades the expensive containment checks (unlike the second step of the bucket algorithm) and consequently outperforms the bucket algorithm. An experimental comparison of the MiniCon [122] to the bucket and inverse-rules algorithms shows that it scales up well and outperforms the previous ones.

An other similar algorithm was developed parallel to the MiniCon algorithm: The *shared-variable-bucket algorithm* [99]. It is also an extension of the bucket algorithm and shares the same steps with the MiniCon algorithm.

The more complex problem of answering queries using views is investigated in [22]; this considered queries and views as conjunctive queries with arithmetic comparisons (CQACs) over dense orders. A novel algorithm, called *RewirteLSI-Query*, for generating maximally-contained rewritings for left-semi-interval (LSI) (or right-semi-interval (RSI)) queries using views with general arithmetic comparisons is presented. RewirteLSIQuery shares the basic steps of the MiniCon algorithm (or the Shared- Variable-Bucket algorithm). It is novel in dealing with nondistinguished view variables, and satisfying comparisons in the query.

7.3 Semantic Caching in Edutella

Edutella constitutes as a schema-based peer-to-peer network an interesting application area for semantic caching. In this milieu where we deal with a significant number of participant peers and super-peers, provider as well as consumer, the network resources become more precious and the avoidance of superfluous queries and messages crucial.

In order to plane the caches and to implement semantic caching in Edutella, it is expedient to look at the different peer types and their characteristics. We distinguish three types of peers in Edutella:

- *SuperPeers*: best network connection, highest availability, and high capacity (dedicated server).
- *ProviderPeers*: good network connection, high availability, and high capacity (dedicated server).

- *ConsumerPeers*: misconnection to good connection, temporal availability, and limited capacity.

Caching in Edutella should lower the network load and the response time (query optimization). For this purpose we have to make the following decisions:

- Which data have to be kept in the cache? Since we opted for semantic caching in Edutella, query descriptions and their respective results have to be cached.
- When to fill up the cache? The cache can be actively (by automatically generating queries), or passively (at the peers/super-peers posed queries and their result sets are extracted from the network traffic) filled up. We decided to implement the passive caching.
- Where to integrate caching in the Edutella infrastructure? The super-peers seem to be the ideal place for caching, in order to attend an optimal exploitation of caching. Due to their characteristics, the super-peers offer the best requirements for caching. Super-peers are, in addition to their high capacities and highest availability, responsible for query routing and query processing and dispose according to this of the posed queries and their results.
- What to do in the case the cache is full? In other words , which cache replacement policy should we use? For our first implementation, we decided to use the LRU replacement algorithm, as it is efficient [148] and simple to implement.

7.3.1 Semantic Caching Component in Edutella

It is quite simple to add new components to the Edutella framework, due to its modularity. The most relevant component for our purpose is the component concerned with the query processing. All tasks related to query processing, like the manipulation of query results, etc. take place at the query processing component. Query processing in Edutella is carried out at the super-peer level. We distinguish two types of queries: queries which are posed directly by a provider and those forwarded by a super-peer. Former queries initiate a *QueryServiceEvent*, the latter once a *RoutingServiceEvent*. As a matter of course, both events have to be proceeded, in order to get a complete query processing. Hence a *ServiceAdapter* intercepting and encapsulating the *QueryServiceEvent* and the *RoutingServiceEvent* into one

event is implemented. The *ServiceAdapter* is derived from a so called *AbstractRoutingAdapter*. It has also one own event, which is derived from a so called *AbstractEvent* and implemented by the *ProcessorEvent* interface.

Caching is integrated as a new component in the Edutella framework. Two classes, *CachingServiceAdapter* and *CachingProcessorImpl* provide the integration of caching in the query processing, i.e., all queries and query results have to pass through the caching component. The cache proofs, wether incoming queries can be answered by cache content. Queries, which can not be answered by the cache, are forwarded to the connected super-peers. The query results and the query descriptions are then stored in the cache.

CachingServiceAdapter After the incoming of a *QueryServiceEvent* or a *RoutingServiceEvent*, the *CachingServiceAdapter* generates a *FakeQueryReceiver* and a *FakeResultSender* and then encapsulates them together with the event to a new *CachingEvent*. The UML sequence diagram on Figure 7.1 shows this process.

CachingProcessor is responsible both for the processing of *CachingEvents* and for the coupling of the caching component. The *CachingProcessor* inherits from the *EdutellaProcessorImpl*, which implements the Edutella-processor interface. The incoming of a *CacheEvent* initiates the process(event) method of the caching processor, which then calls the local method *checkCache*. This method forwards the queries to the caching component. If the query can be answered by the cache, the results are then forwarded using *ResultSender*. This procedure is presented by the UML sequence diagram in Figure 7.2.

Caching in Edutella consists of two main components: The *CachingImpl* class, which is responsible for query processing, and the *Cache* class, which is responsible for the cache management including the cache replacement policy. The *Cache* class implements the *Interface Map*. A new view is added by *put(key, value)*, where *key* is the query and *value* the query value. The Map Interface enables a simple cache replacement, i.e. if a the whole cache management or the replacement policy has to be modified, the new class has only to implement the Map Interface. Both components are tied by well defined interfaces and could be changed independently from each other.

Answering Queries Using Semantic Caches We decided to use the MiniCon algorithm to answer incoming queries using the cached views and content, since

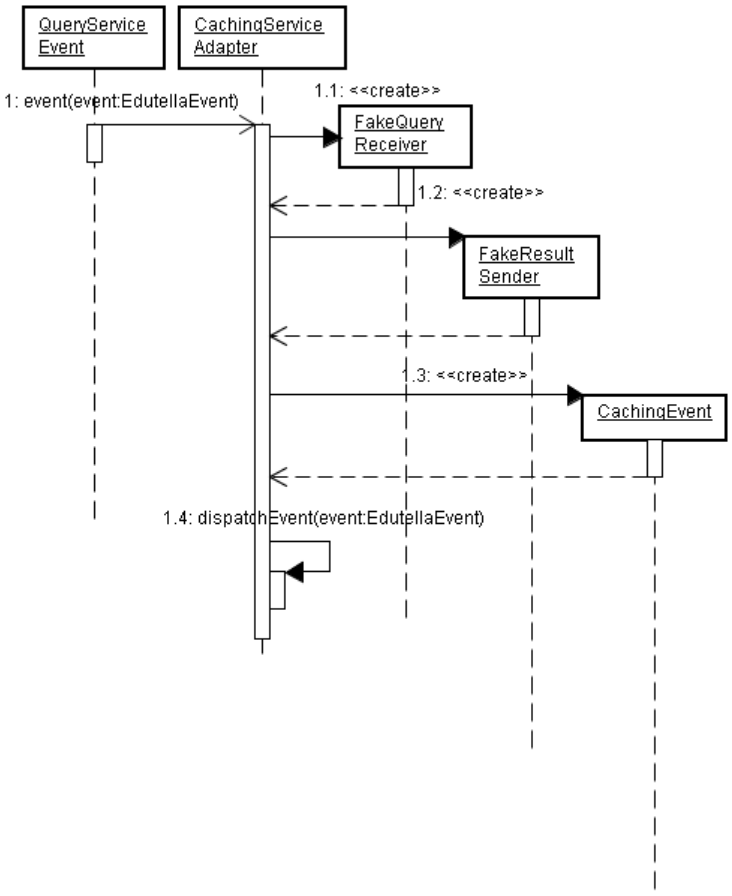


Figure 7.1: CachingServiceAdapter

it is one of the established well-scalable algorithms for answering queries using views. The incoming queries are rewritten using MiniCon based on the cached views.

The implementation of the MiniCon algorithms takes place in the *CachingImpl* class. It is, like the MiniCon algorithm, subdivided into two steps. The MCDs are initially generated, and then combined to build the optimal solutions. The MCDs are implemented as MCD-Objects containing the associated view V, the Mapping phi, the homomorphism h and a list of the covered query subgoals. Three methods are implemented for the generation of the MCDs:

- *formMCDs* has as input the query and the cache and as output a list of MCDs.
- *relevantLiteral* proofs the mapping between a query literal and a view literal

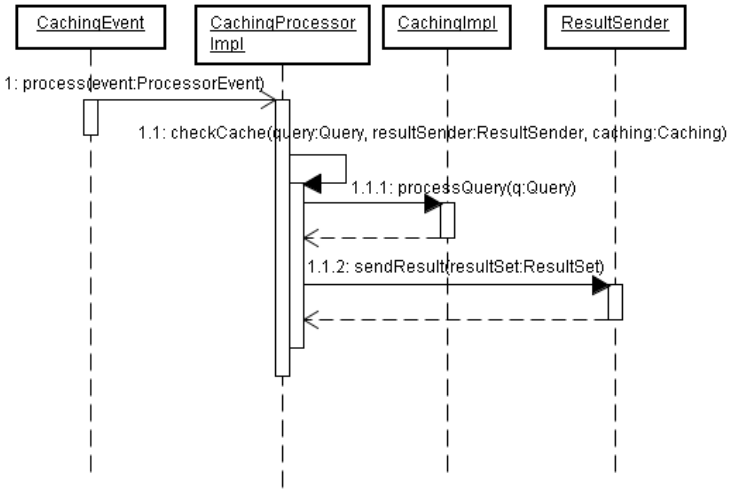


Figure 7.2: CachingProcessor

- *makeMCDs* uses the rules described in Section 7.2.3 to build the MCDs.

7.4 Related Work

As we already mentioned and briefly described, past research areas related to semantic caching include, inter alia, answering queries using views ([60], [43], [122], [145]), query containment problems ([36], [145]), and conventional caching ([24], [48]).

Semantic caching in client-server or multi-database systems has received growing interest recently. Dar et al. propose in [38] a model for client-side caching and replacement in a client-server database system and show the performance of the semantic caching approach compared to the traditional page caching and tuple caching. They mention the easy extension of their approaches to a multiple-server architecture or P2P network, but do not investigate the specific issues related to P2P networks (such as the placement of the cache data structures or the most appropriate cache replacement policy). Moreover, they only consider selection queries on single relations. Dealing with more complex queries is, however, certainly an important issue in the context of semantic caching, and in our approach, we do consider them. More specifically, we consider scalable algorithms for answering conjunctive queries (even with arithmetic comparisons[3]) using views. Godfrey and Gryz present in [55] a general formal framework for semantic caching.

[3] not implemented in our current prototype

For our approach, semantic caching approaches investigated in the context of Web caches are also interesting. The traditional approach of web caches consists of using dedicated machines for centralized web caching. In SQUIRREL [70] the centralized approach has been replaced by a decentralized P2P-approach. This approach, however, does not use semantic caches, or investigate the cache management at the peers. Lee and Chu present in [88] caching via query matching techniques for Web databases. They also only consider disjoint conjunctive predicates.

In the context of P2P systems, semantic caching has been, so far, not investigated. There is some work on caching in P2P networks, mainly associated with replication. Kangasharju and Ross present in [79] a set of distributed algorithms for replication and cache replacements policies (e.g. Top-K MFR and Top-K LRU) for P2P caches assuming an underlying DHT infrastructure. They show that Top-K MFR provides near optimal performance. Whether this cache replacement policy performs also in schema-based super-peer networks, remains to be investigated. Wierzbicki et al. compare in [148] some conventional cache replacement policies in the case of P2P traffic and show that the LRU policy performs very well.

7.5 Further Work

We reemphasize that this work represents only the preliminaries for further intensive investigation on semantic caching in P2P networks. We implemented a caching component which can be flexibly added to any peer, due to the modular Edutella architecture and to the possibility of integrating new components in the query processing. We currently use the semantic caching approach at the super-peers. However, semantic caching at the provider or consumer as well as a new dedicated caching-peer is absolutely conceivable. During the design and implementation process some new ideas accrued:

- *Semantic Regions as Replacement policy*: We use currently the LRU algorithm. The use of the complex semantic regions replacement policy is certainly a good alternative. A comparison of the two strategies with regard to the query response time is also a conceivable future task.
- *Update of Cache Content*: The cache management component could update the cache content by re-posing the stored queries. Here, consideration of the usage of statistics is necessary, in order to avoid superfluous network load (update of non-used content).
- *Invalidation of Cache Content*: It would be very advantageous, if provider-peers inform the cache management about content changes.

- *Caching-Peer*: The cache management and rewriting of queries requires a high capacity and availability of the caching peer. It would be helpful to dedicate specific (big and fast memory) super-peer(s) for caching. There is definitely a need to clarify in-depth, where to cache (i.e., at all super-peers in the backbone, or only at specific caching super-peers), and which queries to cache (e.g. all new incoming queries, frequently posed queries,etc.).

- *Complex Query Processing as Edutella Component*: The current complex query processing is not implemented as a component in Edutella. We use for the current semantic caching implementation for Edutella the simple query processing component. That is, we assume that a given query can be completely answered by a peer/super-peer. In the case of complex query processing (i.e., a query needs data from more than one peer to be executed), the queries are split and partial results are forwarded. This would imply an enhancement of the use and efficiency of semantic caching. In order to realize the integration of semantic caching in the complex query processing, we have to modify the complex query processing implementation and to integrate it as a new component in the Edutella architecture.

- *Experimental Evaluation*: It is obvious that experiments to examine the efficiency of semantic caching approach in Edutella, are extremely important. They are virtually the way to find out, whether semantic caching is profitable or not. The main metrics to use, for this purpose, are response time and cache hit rates. It would be also interesting to compare semantic caching to replication approaches in P2P networks.

Chapter 8

Summary

The main purpose of metadata is to facilitate and improve the retrieval of information. In the Web context, metadata just become crucial. In view to the exponential rate of the Web's growth, data retrieval becomes like finding needles in an enormous haystack. The fact that metadata infrastructure was added middlingly late to the Web increases significantly the complexity of its use and management. The World Wide Web Consortium (W3C) has been working for a few years on the Semantic Web. This "future" Web should enhance the actual Web with "Semantics" by adding a metadata layer. In this thesis, we aim at contributing to the efforts towards the Semantic Web, especially those related to metadata modeling and management.

In the first part, we focused on learning (a.k.a. educational) metadata. Learning metadata are specific metadata including information that has particular relevance for learning purposes. E-learning is non-conceivable without learning metadata. We investigated, more precisely, learning metadata issues in the context of a "local" open learning repository (OLR for short). Thereby, we addressed the following main issues:

Metadata modeling We compared the metadata modeling languages RDF/RDFS - the Semantic Web standard - and O-Telos - a deductive object-oriented conceptual modeling language - and analyzed their similarities and differences, based on our long experience in modeling and meta modeling of open learning repositories. This comparison provides a better understanding of the strengths and the weaknesses of RDF and its modeling capabilities.

Learning Metadata Standards We discussed the Learning Object Metadata Standard LOM and showed the lack of description facilities of different instruc-

tional roles, which are available for, or can be played by a learning object within a course.We extended previous work, which has tried to extend LOM with didactic metadata, by introducing an additional abstraction layer to LOM which explicitly takes different instructional theories into account.

Open learning repositories We proposed three generations of open learning repositories (OLRs), which serve as testbed for the investigated issues on learning metadata, learning metadata standards and learning metadata management. Thereby, we investigated, within an interdisciplinary team, how these open learning repositories can be used to enhance teaching and learning. Thus, we have implemented and evaluated different instructional models. We use RDF - the lingua franca of the Semantic Web - as metadata modeling language in our OLRs. The OLRs have been implemented, with different focuses, during several years at our institute.

In the second part, we generalized the learning metadata issues, particularly metadata management, to issues related to the broadly used metadata that annotate any resource on the Web. We also expanded the metadata management from the local environment of open learning repositories to the distributed environment of peer-to-peer networks. The open learning repositories play then the role of special peers, the *metadata providers*, in the P2P network. Unfortunately, although quite a few database techniques can be re-used in the P2P context, P2P metadata management infrastructures pose additional challenges caused by the open and dynamic nature they exhibit. We investigated the following issues:

Schema-based P2P Infrastructure We presented our super-peer based topology and schema-aware distributed routing indices extended with suitable statistics for our schema-based P2P infrastructure, Edutella, and showed how these indices facilitate the distribution and dynamic expansion of query plans.

Query processing in schema-based P2P networks We proposed a set of transformation rules to optimize query plans and discussed different optimization strategies in detail, enabling efficient distributed query processing in a schema-based P2P network.

Semantic Caching We introduced a semantic caching component for our schema-based P2P-network based on well-known algorithms for answering queries using materialized views.

Bibliography

[1] About source code version control with CVS. https://www.cvshome.org/nonav/scdocs/ddCVS.html.

[2] IEEE Learning Technology Standards Committee. http://ltsc.ieee.org/wg12/.

[3] IMS Global Learning Consortium, Inc. http://www.imsglobal.org/.

[4] RDF Binding for LOM Metadata. http://kmr.nada.kth.se/el/ims/metadata.html.

[5] Research Center Learning Lab Lower Saxony (L3S). http://www.l3s.de.

[6] searchdatabase.com. http://searchdatabase.com.

[7] SiRPAC RDF Parser. http://www-db.stanford.edu/~melnik/rdf/api.html.

[8] The Getty Research Institute. Los Angeles, CA, USA. http://www.getty.edu/research/institute/.

[9] W3C - World Wide Web Consortium. http://www.w3c.org.

[10] Extensible markup language (xml) 1.0, February 1998. http://www.w3.org/TR/REC-xml.

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

[12] E-learning takes important step forward, December 2000. http://dublincore.org/news/pr-20001206.shtml.

[13] Resource Description Framework(RDF) Schema Specification 1.0. W3C Candidate Recommendation 27 March 2000. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, March 2000.

[14] Semantic Web Activity Statement, 2001. http://www.w3c.org/2001/sw/#activity.

[15] Draft Standard for Learning Object Metadata. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf, July 2002.

[16] DC Usage, 08 2003. `http://dublincore.org/documents/usageguide/`.

[17] Advanced Distributed Learning (ADL). Sharable Content Object Reference Model (SCORM), 2004. `http://www.adlnet.org/index.cfm?fuseaction=DownFile&libid=648&bc=false`.

[18] *Encyclopaedia Britannica*. Encyclopedia Britannica Inc., 2004.

[19] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. `http://www.w3.org/TR/2004/REC-rdf-schema-20040210/`, February 2004.

[20] K. Aberer and M. Hauswirth. Semantic gossiping. In *Database and Information Systems Research for Semantic Web and Enterprises, Invitational Workshop*, University of Georgia, Amicalola Falls and State Park, Georgia, April 2002.

[21] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, Reading, MA, USA, 1995.

[22] Foto Afrati, Chen Li, and Prasenjit Mitra. Answering queries using views with arithmetic comparisons. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 209–220. ACM Press, 2002.

[23] A. V. Aho, Y. Sagiv, and J. D. Ullman. Efficient optimization of a class of relational expressions. *ACM Trans. Database Syst.*, 4(4):435–454, 1979.

[24] Rafael Alonso, Daniel Barbara, and Hector Garcia-Molina. Data caching issues in an information retrieval system. *ACM Trans. Database Syst.*, 15(3):359–384, 1990.

[25] Apache ANT manual. `http://ant.apache.org/manual/index.html`.

[26] Apache ANT core tasks. `http://ant.apache.org/manual/coretasklist.html`.

[27] Apache Software License 2.0. `http://www.apache.org/licenses/LICENSE-2.0`.

[28] David P. Ausubel. *Educational Pychology. A Cognitive View.* New York, 1968.

[29] Christian Beck. *Ästhetisierung des Denkens. Zur Postmoderne-Rezeption der Pädagogik*. Bad Heilbrunn, 1993.

[30] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *Proceedings of the Fifth International Workshop on the Web and Databases*, Madison, Wisconsin, June 2002.

[31] P. Boncz and C. Treijtel. AmbientDB: Relational Query Processing over P2P Network. In *International Workshop on Databases, Information Systems and Peer-to-Peer Computing*, Berlin, Germany, September 2003.

[32] R. Braumandl. *Quality of Service and Query Processing in an Information Economy*. PhD thesis, Universität Passau, Fakultät für Mathematik und Informatik, D-94030 Passau, 2001. Universität Passau.

[33] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, and K. Stocker. ObjectGlobe: Ubiquitous query processing on the Internet. *The VLDB Journal: Special Issue on E-Services*, 10(3):48–71, August 2001.

[34] I. Brunkhorst, H. Dhraief, A. Kemper, W. Nejdl, and C. Wiesner. Distributed queries and query optimization in schema-baed p2p-systems. In *International Workshop On Databases, Information Systems and Peer-to-Peer Computing, VLDB 2003*, Berlin, Germany, September 2003.

[35] A. Buchmann, M. T. Özsu, M. Hornick, D. Georgakopoulos, and F. A. Manola. A transaction model for active distributed object systems. In A. K. Elmagarmid, editor, *Database Transaction Models For Advanced Applications*, The Morgan Kaufmann Series in Data Management Systems, pages 123–158. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1992.

[36] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 77–90. ACM Press, 1977.

[37] CSS Zen Garden - The Beauty in CSS Design. `http://csszengarden.com/`http://csszengarden.com/.

[38] Shaul Dar, Michael J. Franklin, Björn Jónsson, Divesh Srivastava, and Michael Tan. Semantic data caching and replacement. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 330–341. Morgan Kaufmann Publishers Inc., 1996.

[39] Eric Miller Dave Beckett and Dan Brickley. Expressing Simple Dublin Core in RDF/XML, July 2002. `http://dublincore.org/documents/dcmes-xml/`.

[40] Dublin Core Metadata Initiative. `http://dublincore.org/`.

[41] Hadhami Dhraief. OLR2 RDF Schema. `http://www.kbs.uni-hannover.de/~hdhraief/OLR/rdf_schema/OLR2/`, 2002.

[42] Digital repository. `http://www.dspace.org/`.

[43] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 109–116. ACM Press, 1997.

[44] S.Sutton E. Duval, W. Hodgins and S. L. Weibel. Metadata principles and practicalities, 2002. `http://www.dlib.org/dlib/april02/weibel/04weibel.html`.

[45] Eclipse project. `http://www.eclipse.org/`.

[46] The Edutella Project. http://edutella.jxta.org/, 2002.

[47] Getting started with Enhydra Application Server. `http://enhydra.objectweb.org/doc/5.1/getting\_started/pdf/getting-start%ed.pdf`.

[48] Michael J. Franklin, Michael J. Carey, and Miron Livny. Local disk caching for client-server database systems. In *Proceedings of the 19th International Conference on Very Large Data Bases*, pages 641–655. Morgan Kaufmann Publishers Inc., 1993.

[49] S. Ganguly, W. Hasan, and R. Krishnamurthy. Query optimization for parallel execution. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 9–18, San Diego, CA, USA, June 1992.

[50] Starting a new Maven project. `http://maven.apache.org/reference/user-guide.html\#Starting_a_New_Proje%ct`.

[51] Nick Gibbins, 1997. `http://www.ukoln.ac.uk/web-focus/events/conferences/www6/focus/hypertex%t97/gibbins/report.html`.

[52] T. Gill. Metadata and the World Wide Web. `http://www.getty.edu/research/conducting_research/standards/intrometada%ta/pdf/gill.pdf`, July 2000.

[53] Anne Gilliland-Swetland. Setting the stage, July 2000. `http://www.getty.edu/research/conducting_research/standards/intrometada%ta/2_articles/index.html`.

[54] GNU Website. `http://www.gnu.org/`.

[55] Parke Godfrey and Jarek Gryz. Answering queries by semantic caches. In *Proceedings of the 10th DEXA*, Florence, Italy, 1999.

[56] L. Gong. Project JXTA: A technology overview. Technical report, SUN Microsystems, April 2001. http://www.jxta.org/project/www/docs/TechOverview.pdf.

[57] GNU General Public License. `http://www.gnu.org/licenses/gpl.html`.

[58] Nicola Guarino. Concepts, attributes and arbitrary relations. *Data Knowledge Engineering*, 8:249–261, 1992.

[59] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, Budapest, Hungary, May 2003.

[60] Alon Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[61] Handle System, Introduction. `http://www.handle.net/introduction.html`.

[62] Siegfried Handschuh, Steffen Staab, and Alexander Maedche. Cream: creating relational metadata with a component-based, ontology-driven annotation framework. In *K-CAP 2001: Proceedings of the international conference on Knowledge capture*, pages 76–83. ACM Press, 2001.

[63] Wolfgang Nejdl Heidrun Allert, Hadhami Dhraief. How are learning objects used in learning processes? instructional roles of learning objects in lom. In *ED-MEDIA 2002, World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Denver Colorado, USA, 2002.

[64] Hibernate - Java persistence layer. `http://www.hibernate.org/`.

[65] Stefan Hoermann, Andreas Faatz, Oliver Merkel, Ansgar Hugo, and Ralf Steinmetz. Ein Kurseditor für modularisierte Lernressourcen auf der Basis von Learning Objects Metadata zur Erstellung von adaptierbaren Kursen. In *LLWA 01 - Tagungsband der GI-Workshopwoche "Lernen-Lehren-Wissen-Adaptivität"*, pages 315–323. Ralf Klinkenberg, Stefan Rueping, Andreas Fick, Nicola Henze, Christian Herzog, Ralf Molitor, Olaf Schroeder, October 2001.

[66] Stefan Hopmann and Kurt Riquarts. Starting a dialogue: A beginning conversation between didaktik and the curriculum traditions. In *Teaching as a Reflective Practice. The German Didaktik Tradition*. Lawrence Erlbaum Associates, Inc, 2000.

[67] Kurt Riquarts Ian Westbury, Stefan Hopmann, editor. *Teaching as a Reflective Practice. The German Didaktik Tradition*. Lawrence Erlbaum Associates, Inc, Mahwah, 2000.

[68] IDEA. `http://www.jetbrains.com/idea/`.

[69] Y. E. Ioannidis. The History of Histograms. In *Proc. of the Conf. on Very Large Data Bases (VLDB)*, pages 19–30, 2003.

[70] S. Iyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache, 2002.

[71] Michael Janneck. Themenzentrierte interaktion als gestaltungsrahmen fr community-systeme. In *GeNeMe 2001: Gemeinschaften in Neuen Medien Konferenzband*. 2001. `http://www.hyperkommunikation.ch/artikel/janneck_tzi.htm`.

[72] A. Janson. Moderation vr - moderations- und kreativitaetstechniken in virtuellen umgebungen., 2002. `http://web.uni-frankfurt.de/dz/neue_medien/standardisierung/janson_foli%en.pdf`.

[73] JDBC - Java Database Connectivity. `http://java.sun.com/products/jdbc/`.

[74] M. Jeusfeld. *Änderungskontrolle in deduktiven Objektbanken*. Infix-Verlag, St. Augustin, Deutschland, 1992.

[75] M. A. Jeusfeld, M. Jarke, H. W. Nissen, and M. Staudt. ConceptBase - Managing Conceptual Models about Information Systems. In P. Bernus, K. Mertins, and G. Schmidt, editors, *Handbook on Architectures of Informations Systems*. Springer Verlag, 1998.

[76] V. Josifovski, P. Schwarz, L. Haas, and E. Lin. Garlic: A New Flavor of Federated Query Processing for DB2. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, Madison, USA, June 2002.

[77] JUnit - Java unit test framework. `http://www.junit.org/`.

[78] Project JXTA Homepage. http://www.jxta.org/.

[79] Jussi Kangasharju and Keith W. Ross. Adaptive replication and replacement in p2p caching. `citeseer.ist.psu.edu/563593.html`.

[80] Greg Kearsley. Subsumption Theory (D. Ausubel). `http://tip.psychology.org/ausubel.html`.

[81] A. Kemper and C. Wiesner. HyperQueries: Dynamic Distributed Query Processing on the Internet. In *Proc. of the Conf. on Very Large Data Bases (VLDB)*, pages 551–560, Rom, Italy, September 2001.

[82] A. Kemper, C. Wiesner, and P. Winklhofer. Building Dynamic Market Places using HyperQueries. In *Proc. of the Intl. Conf. on Extending Database Technology (EDBT)*, volume Lecture Notes in computer Science (LNCS), pages 749–752, Prague, Czech Republic, March 2002. Springer Verlag.

[83] Wolfgang Klafki. *Neue Studien zur Bildungstheorie und Didaktik*. Weinheim, 1993.

[84] Wolfgang Klafki. The significance of classical theories of bildung for a contemporary concept of allgemeinbildung. In *Teaching as a Reflective Practice. The German Didaktik Tradition*. Lawrence Erlbaum Associates, Inc, Mahwah, 2000.

[85] Stefan Kokkelink and Roland Schwnzl. Expressing Qualified Dublin Core in RDF / XML, April 2002. `http://dublincore.org/documents/2002/04/14/dcq-rdf-xml/`.

[86] R. Koper. Modelling units of study from a pedagogical perspective. the pedagogical meta-model behind eml. Technical Report First Draft. Version 2, Open University of the Netherlands, 2001. `http://eml.ou.nl/introduction/docs/ped-metamodel.pdf`.

[87] K.Tolle. VRP RDF Parser. `http://www.ics.forth.gr/proj/isst/RDF`.

[88] Dongwon Lee and Wesley W. Chu. Towards intelligent semantic caching for web sources. *J. Intell. Inf. Syst.*, 17(1):23–45, 2001.

[89] A. Y. Levy, D. Srivastava, and T. Kirk. Data Model and Query Evaluation in Global Information Systems. *Journal of Intelligent Information Systems (JIIS)*, 5(2):121–143, 1995.

[90] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 251–262. Morgan Kaufmann Publishers Inc., 1996.

[91] Niki Pissinou Luo Li, Birgitta König-Ries and Kia Makki. Strategies for semantic caching. In *DEXA '01: Proceedings of the 12th International Conference on Database and Expert Systems Applications*, volume 2113 of *Lecture Notes in Computer Science*, pages 99–106. Springer, 2001.

[92] Apache Maven project descriptor. `http://maven.apache.org/reference/project-descriptor.html`.

[93] Getting started with Maven. `http://maven.apache.org/start/index.html`.

[94] Apache Maven reference documentation. `http://maven.apache.org/reference/index.html`.

[95] High level project build tool. `http://maven.apache.org/`http://maven.apache.org/.

[96] Norbert Meder. Didaktische ontologien. In *Globalisierung und Wissensorganisation: Neue Aspekte fr Wissen, Wissenschaft und Informationssysteme*. H. Peter Ohly, Gerhard Rahmstorf, Alexander Sigel, Wuerzburg, 2000.

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

[98] Davis Merrill. First principles of instruction. In *Educational Technology Research & Development*. 2001.

[99] Prasenjit Mitra. An algorithm for answering queries efficiently using views. In *ADC '01: Proceedings of the 12th Australasian conference on Database technologies*, pages 99–106. IEEE Computer Society, 2001.

[100] Mozilla Public License 1.1. `http://www.mozilla.org/MPL/MPL-1.1.html`.

[101] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: A language for representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4), 1990.

[102] MySQL. `http://www.mysql.org/`.

[103] W. Nejdl, H. Dhraief, and M. Wolpers. O-telos-rdf: a resource description format with enhanced meta-modeling functionalities based on o-telos, 2001.

[104] 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. In *Proceedings of the 11th International World Wide Web Conference*, Hawaii, USA, May 2002. http://edutella.jxta.org/reports/edutella-whitepaper.pdf.

[105] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Loser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *Proceedings of the International World Wide Web Conference*, Budapest, Hungary, May 2003. http://citeseer.nj.nec.com/nejdl02superpeerbased.html.

[106] Wolfgang Nejdl, Hadhami Dhraief, and Martin Wolpers. O-telos-rdf: An extension of rdf with enhanced meta-modeling and reification functionalities. `citeseer.ist.psu.edu/552801.html`.

[107] Wolfgang Nejdl and Nicola Henze. Adaptivity in the kbs hyperbook system. In *2nd Workshop on User Modeling and Adaptive Systems on the WWW*, Toronto,Canada, May 1999.

[108] Wolfgang Nejdl, Martin Wolpes, and Christian Capelle. The RDF schema specification revisited. In *Modellierung 2000*, St. Goar, Germany, April 2000. http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2000/modeling2000/wolpers.pdf.

[109] Mikael Nilsson and Matthias Palmr. Conzilla - towards a concept browser. Technical Report CID-53, TRITA-NA-D9911, Department of Numerical Analysis and Computing Science, KTH, Stockholm, 1999. http://kmr.nada.kth.se/papers/ConceptualBrowsing/cid_53.pdf.

[110] OLR Java code cross reference. `http://olr.sourceforge.net/xref/`.

[111] OLR code repository, 2004. `http://olr.sourceforge.net/cvs-usage.html`.

[112] OLR database reference. `http://olr.sourceforge.net/reference.html\#Database_and_object_model`.

[113] OLR database setup and configuration. `http://olr.sourceforge.net/install.html\#Setting_up_the_database`.

[114] OLR Project Website. `http://olr.sourceforge.net/`.

[115] OLR installation documentation. `http://olr.sourceforge.net/install.html`.

[116] OLR reference documentation. `http://olr.sourceforge.net/reference.html`.

[117] OLR install as web application. `http://olr.sourceforge.net/install.html\#Installing_as_web_application`.

[118] V. Papadimos and D. Maier. Distributed Query Processing and Catalogs for Peer-to-Peer Systems. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA, January 2003.

[119] Seymour Papert. Constructionism vs. Instructionism. `http://learning.media.mit.edu/`, 1980.

[120] Editor Patrick Hayes. RDF Semantics, February 2004. `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`.

[121] PostgreSQL. `http://www.postgresql.org/`.

[122] Rachel Pottinger and Alon Y. Levy. A scalable algorithm for answering queries using views. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 484–495. Morgan Kaufmann Publishers Inc., 2000.

[123] NISO Press. Understanding metadata, 2004. `http://www.niso.org/standards/resources/UnderstandingMetadata.pdf`.

[124] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press New York, NY, USA, 2001.

[125] Christina Rautenstrauch. *Tele-Tutoren - Qualifizierungsmerkmale einer neu entstehenden Profession*. Norbert Meder, Bielefeld, 2001.

[126] Resource description framework. `http://www.w3c.org/RDF`.

[127] RFC 3651, Handle System Namespace and Service Definition. `http://www.ietf.org/rfc/rfc3651.txt`.

[128] Christoph Richter. Bericht über die Interviews zum OLR-2 und KI-Skript.

[129] C. Meiler S. Jablonski, I. Petrov and U. Mayer. *Guide to Web Application and Platform Architectures*. Springer-Verlag, Berlin Heidelberg, 2004.

[130] Burkhard Sachs. Skizzen und Anmerkungen zur Didaktik eines mehrperspektivischen Technikunterrichts. Deutsches Institut für Fernstudien an der Universität Tübingen, Tübingen, 1979.

[131] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. HyperCuP—Hypercubes, Ontologies and Efficient Search on P2P Networks. In *International Workshop on Agents and Peer-to-Peer Computing*, Bologna, Italy, July 2002.

[132] Rolf Schulmeister. Szenarien netzbasierten lernens. In *Virtueller Campus: Szenarien - Strategien - Studium*, pages 16–38. Wagner, E./Kindt, M. (Hg.), New York/Mnchen/ Berlin, 2001.

[133] Open Source software development website. `http://www.sourceforge.net/`.

[134] Friedrich Steimann. *Modellierung mit Rollen*. Habilitationsschrift, University of Hanover, Germany, 2000.

[135] M. Steinbrunn, G. Moerkotte, and A. Kemper. Heuristic and randomized optimization for the join ordering problem. *The VLDB Journal*, 6(3):191–208, August 1997.

[136] M. Stillger, G. M. Lohman, V. Markl, and M. Kandil. LEO - DB2's LEarning Optimizer. In *Proc. of the Conf. on Very Large Data Bases (VLDB)*, pages 19–28, Rom, Italy, September 2001.

[137] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press New York, NY, USA, 2001.

[138] H. Stuckenschmidt, R. Vdovjak, G-J. Houben, and J. Broekstra. Index structures and algorithms for querying distributed rdf repositories. In *Proceedings of the 13th International World Wide Web Conference (WWW2004)*, New York, USA, May 2004.

[139] Christian Suess. Adaptive Knowledge Management: A Meta-Modeling Approach and its Binding to XML. In *GI-Workshop Grundlagen von Datenbanken*. In: H.-J. Klein (Ed.), 2000.

[140] Sigmar-Olaf Tergan. Grundlagen der Evaluation: ein Überblick. In *Qualitätsbeurteilung multimedialer Lern- und Informationssysteme - Evaluationsmethoden auf dem Prüfstand*. Peter Schenkel, Sigmar-Olaf Tergan, Alfred Lottmann, Nürnberg, 2000.

[141] Dave Thomas and Andy Hunt. *Pragmatic Version Control with CVS*. Sep 2003.

[142] Apache Torque Generator. `http://db.apache.org/torque/generator/`.

[143] Apache Torque - Java persistence layer. `http://db.apache.org/torque/`.

[144] Apache Torque - Schema reference. `http://db.apache.org/torque/schema-reference.html`.

[145] Jeffrey D. Ullman. The Bucket Algorithm. `http://www-db.stanford.edu/~ullman/cs345notes/slides01-12.pdf`.

[146] Web Application archive (WAR) howto. `http://jakarta.apache.org/tomcat/tomcat-5.0-doc/appdev/deployment.html`.

[147] Maven WAR plugin. `http://maven.apache.org/reference/plugins/war/`.

[148] Adam Wierzbicki, Nathaniel Leibowitz, Matei Ripeanu, and Rafal Wozniak. Cache Replacement Policies Revisited: The Case of P2P Traffic. `citeseer.ist.psu.edu/wierzbicki04cache.html`.

[149] G. Wiesenfarth. Zum technischen Handeln als Grundbegriff der Technikdidaktik. In *Zeitschrift fuer Technik im Unterricht*. 1992.

[150] David Wiley. Connecting learning objects to instructional design theory. The Instructional Use of Learning Objects, 2000. `http://www.reusability.org/read/`.

[151] W.Nejdl and M.Wolpers. KBS Hyperbook - A Data-Driven Information System on the Web. In *WWW8 Conference*, Toronto, May 1999.

[152] XML to Java compiler. `http://xmlc.objectweb.org/`.

[153] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer systems. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Viena, Austria, July 2002.

List of Figures

List of Tables

Publication List

2001

- W. Nejdl, H. Dhraief, B. Wolf. Building up AI Resources as an AI Testbed. IJCAI'01 Workshop on Effective Interactive AI Resources August 5th, 2001, Edmonton, Canada.
- H. Allert, H. Dhraief, W. Nejdl. Intelligent Online-Knowledge-Resources for Intentional Learning. Computer supported and instructional design of online knowledge resources. ED-MEDIA 2001 World Conference on Educational Multimedia, Hypermedia & Telecommunications, June 25-30, 2001, Tampere, Finland.
- H. Dhraief, W. Nejdl, B. Wolf, M. Wolpers. Open Learning Repositories and Metadata Modeling. International Semantic Web Working Symposium (SWWS), July 30 - August 1, 2001 Stanford University, California, USA.
- W. Nejdl, H. Dhraief and M. Wolpers. O-Telos-RDF: A Resource Description Format with Enhanced Meta-Modeling Functionalities based on O-Telos, Workshop on Knowledge Markup and Semantic Annotation at the First International Conference on Knowledge Capture (K-CAP'2001), Oct. 21 - 23, 2001, Victoria, B.C., Canada.

2002

- H. Allert, H. Dhraief, W. Nejdl. How are Learning Objects Used in Learning Processes? Instructional Roles of Learning Objects in LOM. ED-MEDIA 2002, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Denver Colorado, United States, June 24-29, 2002.
- H. Allert, H. Dhraief, W. Nejdl. Meta-Level Category Role in Metadata Standards for eLearning. Instructional Roles and Instructional Qualities of Learning Objects. COSIGN 2002 - The 2nd International Conference on COMPUTATIONAL SEMIOTICS FOR GAMES AND NEW MEDIA. Augsburg (Germany), 2nd September - 4th September, 2002.
- H. Allert, H. Dhraief, T. Kunze, W. Nejdl, C. Richter. Instructional Models and Scenarios for an Open Learning Repository - Instructional Design and Metadata. E-Learn 2002: World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education (formerly the WebNet Conference). Montreal, Canada, October 15-19, 2002.

- H. Allert, H. Dhraief, W. Nejdl. Intelligente Online Wissensbestände fr handlungsorientiertes Lernen. In: Neusel, Ayla; Poppenhusen, Margot. Universität Neu Denken. Leske + Budrich, 2002.

2003

- I. Brunkhorst, H. Dhraief, A. Kemper, W. Nejdl, C. Wiesner. Distributed Queries and Query Optimization in Schema-Based P2P-Systems. International Workshop On Databases, Information Systems and Peer-to-Peer Computing September 7-8, 2003 Humboldt University, Berlin, Germany Collocated with VLDB 2003.

- H. Dhraief, A. Kemper, W. Nejdl, C. Wiesner. Distributed Queries and Query Optimization in Schema-Based P2P-Systems. Technical Report, December 2003.

2004

- H. Allert, C. Richter, Christoph, H. Dhraief, W. Nejdl. Contextualized Models and Metadata for Learning Repositories. In: Rory McGreal (Ed.): Online Education Using Learning Objects. August 2004.

- H. Dhraief, A. Kemper, W. Nejdl, and C. Wiesner. Processing and optimization of complex queries in schema-based P2P-networks. In Proceedings of the 2nd International Workshop On Databases, Information Systems and Peer-to-Peer Computing, Toronto, Canada, September 2004. In conjunction with the 30th International Conference on Very Large Data Bases.

2005

- I. Brunkhorst and H. Dhraief. Semantic Caching in Schema-Based Peer-to-Peer Networks. In Proceedings of the 3rd International Workshop On Databases, Information Systems and Peer-to-Peer Computing, Trondheim, Norway, August 2005. In conjunction with 31st International Conference on Very Large Data Bases (VLDB 2005).

Curriculum Vitae

- *Name:* Hadhami Dhraief
- *Birth:* July 12, 1971, Tunis, Tunisia
- *Nationality:* Tunisian/German
- *Personal Data:* Married since February 12, 2001 with Mourad Gherib
- *Children:* Taha Emin Gherib (04.09.2002)

Education:

- *1976 - 1982* Primary School, Tunisia
- *1982 - 1989* Lycèe Carthage Prèsidence, Carthage, Tunisia
- *1989* Baccalaureat
- *1989 - 1990* Studienkolleg Hannover (German Course)
- *1990 - 1997* Computer Science student at the Technical University Carolo-Wilhelmina at Braunschweig
- *1997* Diploma (Dipl.-Inform.)
- *1999.09 - 2005.05* Ph.D. Candidate, University of Hannover, Germany

Experiences:

- *1993 - 1997* Student Assistant, Technical University Carolo-Wilhelmina at Braunschweig, Germnay
- *1998 - 1999* Software Engineer, Banque Finance International (BFI), Tunis, Tunisia
- *1999 - 2005* Research Assistant at the Institute of Information Systems - Knowledge Based Systems

Acknowledgments

First and foremost, I would like to thank my supervisor, Prof. Wolfgang Nejdl, for his guidance and support during the last five years. As great as his advice was to me in research, his comprehension and great support during an awkward period in my private life were just as important. Thank you for giving me the chance to finish my doctorate. I am grateful indeed.

I would also like to thank Franziska Pfeffer for her tender and loving care. Franziska, my fairy godmother, has always offered me an agreeable working condition and assisted me in good times and bad.

I would like to thank all my colleagues from the Institute of Information Systems at the University of Hannover for the pleasant working atmosphere.

This work could not have been achieved without the valuable collaboration of many students from the department of Computer Science at the University of Hannover: Boris Wolf, Tobias Kunze, Marc Luzof, Rolf Kulemann and Nils Müller. I would like to thank all of them.

I thank my friends, Chiraz El Borgi and Malek Selem, for proof-reading my thesis and their helpful comments on style and wording.

This work is the achievement of a long road, which began thanks the encouragement and support of my parents. They encouraged me to continue my studies in Germany, and assisted me during my long educational journey. They undergo with me all the ups and downs. Ebi & Mama, merci d'être toujours là pour moi.

To my husband, Mourad, and my son, Taha Emin (Mimou), I dedicate this thesis. Mourad, I can not thank you enough, without you, none of this could have been possible. Mimou, my sweeter adorable son, you show me every day the true values in life. I just adore you.

"Für große Gefühle sind Worte zu klein." (Der Bär im großen blauen Haus)