EDUTELLA: Searching and Annotating Resources within an RDF-based P2P Network

Wolfgang Nejdl, Boris Wolf
L3S and Knowledge Based Systems
University of Hannover
30167 Hannover, Germany
{nejdl,wolf}@kbs.uni-hannover.de

Steffen Staab, Julien Tane
L3S and Institute AIFB
76128 Karlsruhe, Germany
{sst,jta}@aifb.uni-karlsruhe.de

ABSTRACT

P2P applications for searching and exchanging information over the Web have become increasingly popular. This has lead to a number of (usually thematically) focused communities, which allow efficient searching within such communities, and which use specific metadata sets to specify the resources stored within the P2P network. By concentrating on domain and application specific formats for metadata and query languages, however, current P2P networks appear to be fragmenting into non-interoperable niche markets. This contribution describes the open source project Edutella which builds upon metadata standards defined for the WWW and aims to provide an RDF-based metadata infrastructure for P2P applications, building on the recently announced JXTA Framework. We describe one basic service (query) and an Edutella application (annotation) within this network, both being built on a common query language exchange format, and specify the main architecture and APIs of the Edutella P2P network.

1. BACKGROUND

The advantage of the WWW is that it constitutes a predominantly decentral paradigm storing information resources in hypertext like structures. Searching in the WWW, however, typically follows a client-server model, viz. browser vs. search engine [16], inheriting the corresponding benefits and pitfalls. To name some problems, search engines cover only a decreasing percentage of the information available on the Web and their content is often not up to date because of the time required for crawling of the Web.

In contrast, information resources in P2P networks are stored on numerous peers waiting to be queried for these resources. The querying of peer-to-peer networks allows the comprehensive retrieval of up-to-date resources stored at relevant sites. But in order to achieve this, it requires a query mechanism using some description of the resources managed by these peers.

While in the server/client-based environment of the World Wide Web metadata are useful and important, for *Peer-to-Peer (P2P)* environments that come without underlying hypertext structures metadata are absolutely crucial. Such metadata are easy to pro-

vide for specialized cases, but non-trivial for general applications. The core concern of our research therefore is to develop a general infrastructure for combining metadata with P2P networks.

In the context of educational resources for example, which we are currently focusing on, P2P-based approaches are more flexible than centralized approaches like Client-Server computing, with several advantages for the participating institutions. As content providers in a P2P network they do not loose control over their learning resources but still provide them for use within the network. As content consumers, both teachers and students, benefit from having access not only to a local repository, but to a whole network, using queries over the metadata distributed within the network to retrieve required resources.

Recent P2P applications have been very successful for special cases like exchanging audio files. However, retrieving MP3 coded audio files using title and author does not need complex query languages nor complex metadata, so special purpose formats for these P2P applications have been sufficient. Metadata in Gnutella are limited to a file name and a path. This is fine for queries looking for the song "Madonna - Like a Virgin", but cannot be extended to something like "Introduction to Algebra - Lecture 23". For educational resources, queries are more complex and have to build upon standards like IEEE-LOM/IMS [2] metadata with up to 100 metadata entries, which might even be complemented by domain specific extensions.

Furthermore, by concentrating on domain specific formats, current P2P implementations appear to be fragmenting into niche markets instead of developing unifying mechanisms for future P2P applications.

In order to facilitate interoperability and reusability of educational resources, we need to provide an infrastructure flexible enough to accommodate complex and varying metadata sets, and avoid creating another special purpose application suitable only for a specific application area which is outdated as soon as metadata requirements and definitions change. The Edutella infrastructure [4] therefore builds on the W3C metadata standard RDF(S) [1], and uses a standard query model suitable for this formalism, based on Datalog, to exchange queries throughout the Edutella network.

For the local user, the Edutella network transparently provides access to distributed information resources, and different clients/peers can be used to access, retrieve and update these resources. The service and the peer that we will describe in more detail in this paper are querying and annotating resources distributed in the Edutella P2P network, respectively.

Query Service. The Edutella query service is the most basic service within the Edutella network. Peers register queries they may be asked through the query service (i.e. by specifying sup-

ported metadata schemas (e.g. "this peer provides metadata according to the LOM 6.1 or DCMI standards") or by specifying individual properties or even values for these properties (e.g. "this peer provides metadata of the form dc_title(X,Y)" or "this peer provides metadata of the form dc_title(X,'Artificial Intelligence')"). Queries are sent through the Edutella network to the subset of peers who have registered with the service to be interested in this kind of query. The resulting RDF models are sent back to the requesting peer.

Edutella Annotation. In order to be able to meet the requirements of being applicable in a wide range of application scenarios, the Edutella annotation tool must be independent from a particular domain. For instance, it may not just offer annotation for IEEE LOM, but rather it must support a wide range of semantic definitions as it is possible in RDF schema. In order to approach this objective, we investigate two orthogonal dimensions for metadata creation based on which we may emulate all annotation schemes we know of.

Building on the JXTA P2P Framework. JXTA is an Open Source project [5] supported and managed by Sun Microsystems. In essence, JXTA is a set of XML based protocols to cover typical P2P functionality. It provides a Java binding offering a layered approach for creating P2P applications (core, services, applications). In addition to remote service access (such as offered by SOAP), JXTA provides additional P2P protocols and services, including peer discovery, peer groups, peer pipes, and peer monitors.

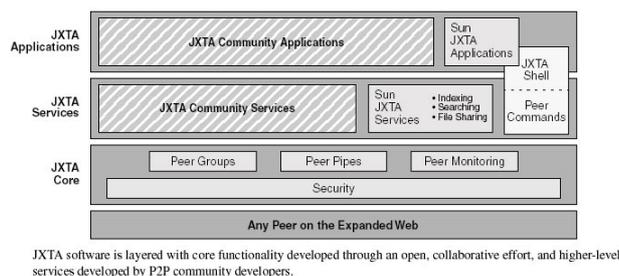Figure 1, reproduced from [5], specifies the different layers within the JXTA architecture.

JXTA software is layered with core functionality developed through an open, collaborative effort, and higher-level services developed by P2P community developers.

Figure 1: JXTA Layers

JXTA provides a layered architecture that fits very nicely into the Edutella application scenarios:

- Edutella Services (described in web service languages like DAML-S or WSDL, etc.) complement the JXTA Service Layer, building upon the JXTA Core Layer[1], and

- Edutella Peers live on the Application Layer, using the functionality provided by these Edutella services as well as possibly other JXTA services.

On the Edutella Service layer, we define data exchange formats and protocols (how to exchange queries, query results and other metadata between Edutella Peers), as well as APIs for advanced functionality in a library-like manner. Applications like repositories, annotation tools or GUI interfaces connected to and accessing the Edutella network are implemented on the application layer.

---

[1] A previous prototype from our group, implemented this summer to gain experiences with the JXTA and JXTAsearch framework, extended the JXTAsearch service (the prototype and our experiences with it are described in [13]), but building directly on the JXTA Core services makes a more flexible design possible.

In section 2, we discuss the Edutella query service and the common data model (ECDM), which provides the basis for the Edutella query exchange language and format implementing distributed queries over the Edutella network, as well as the basic Edutella API for query and registration/distribution peers. Section 3 discusses the annotation application that connects to the Edutella network in order to exploit existing metadata as well as create and provide new metadata to the network.

2. EDUTELLA QUERY SERVICE

2.1 The Query Mechanism

The Edutella Query Service is intended to be a standardized query exchange mechanism for RDF metadata stored in distributed RDF repositories and is meant to serve as both query interface for individual RDF repositories located at single Edutella peers as well as query interface for distributed queries spanning multiple RDF repositories. An RDF repository (or knowledge base) consists of RDF statements (or facts) and describes metadata according to arbitrary RDFS schemas.

To enable a peer to participate in its network, Edutella uses wrappers based on both a common datamodel and a common query exchange format. For communication within the Edutella network the wrapper translates the local data model into the Edutella common data model (ECDM) and vice versa, and connects to the Edutella Network using the JXTA P2P primitives, transmitting the queries based on the ECDM in RDF/XML form. In order to describe and handle different query capabilities of a particular peer, we define several RDF-QEL-i exchange language levels with increasing expressiveness: Currently we have defined language levels RDF-QEL-1, -2, -3, -4 and -5 (see [11]). The most simple language (RDF-QEL-1, purely conjunctive queries) can be expressed as un-reified RDF graph, the more complex ones are more expressive than RDF itself, and therefore have to be expressed using reified RDF statements (e.g. RDF-QEL-3 covers relational algebra, RDF-QEL-4 incorporates Datalog). However, all language levels can be represented through the same internal ECDM data model.
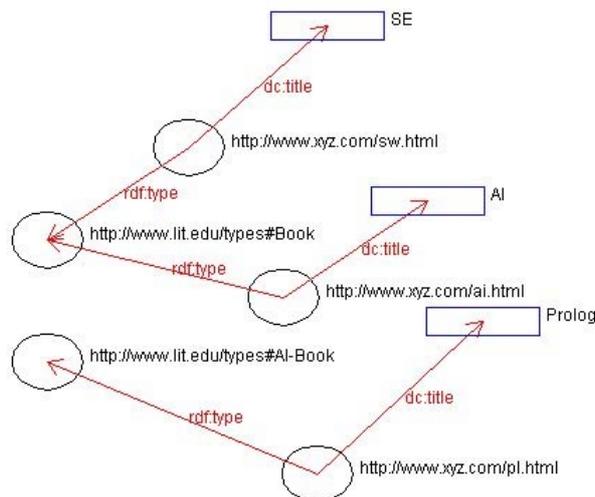

Figure 2: Knowledge Base as RDF Graph

The example presented throughout our paper, we will use a simple RDF knowledge base and a simple query on the knowledge base

depicted in Figure 2. Evaluating the query (plain English)

> "Return all resources that are a book having the title 'AI' or that are an AI book."

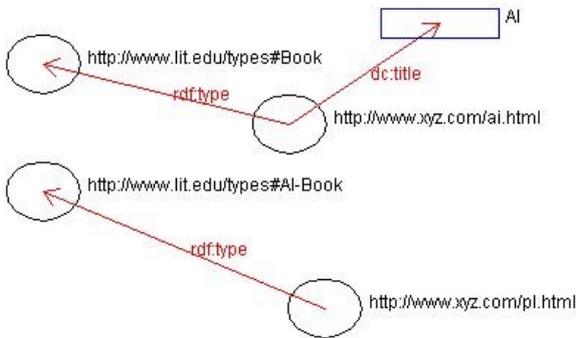we get the query results shown in Figure 3, represented as an RDF-graph.

Figure 3: Query Results as RDF Graph

2.2 Edutella Common Data Model (ECDM)

2.2.1 Basic Semantics

As common query and datamodel, Edutella peers use Datalog, a non-procedural query language based on Horn clauses without function symbols. A Datalog program can be expressed as a set of rules/implications (where each rule consists of one positive literal in the consequent of the rule (the head), and one or more negative literals in the antecedent of the rule (the body)), a set of facts (single positive literals) and the actual query literals (a rule without head, i.e. one or more negative literals). Literals are predicate expressions describing relations between any combination of variables and constants such as title(http://www.xyz.com/book.html, 'Artificial Intelligence'). Disjunction in a query is expressed by a set of rules with identical head. A Datalog query then is a conjunction of query literals plus a possibly empty set of rules.

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

In RDF any statement is considered to be an assertion. We can view an RDF repository as a set of ground assertions either using binary predicates as shown above, or as ternary statements "s(S,P,O)", if we include the predicate as an additional argument. In the following query, we use the binary predicate notation.

```
aibook(X) :- title(X, 'AI'), type(X, Book).
aibook(X) :- type(X, AI-Book).
?- aibook(X).
```

As our query is a disjunction of two conjunctive subqueries, its Datalog representation is composed of two rules with identical heads. The literals in the rules' bodies directly reflect RDF statements with their subjects being the variable X and their objects being bound to constant values such as 'AI'. Literals used in the

head of rules denote derived predicates. In our example, the query expression "aibook(X)" asks for all bindings of X, which conform to the given Datalog rules and the knowledge base to be queried (cf. below for results).

2.2.2 ECDM Datamodel and Queries

Figure 4 visualizes the ECDM, as implemented in our current prototype, as UML diagram. Our Java binding relies on JXTA [5] and makes extensive use of the Stanford RDF API [10]. The implementation of all classes shown in figure 4 is found in the Java package `net.jxta.edutella.util.datamodel`. All classes whose names start with RDF represent standard RDF concepts and correspond to their equivalent counterparts within the Stanford RDF API. These are `RDFReifiedStatement`, `RDFNode`, `RDFResource`, `RDFLiteral` and `RDFModel`.

Queries are represented by `EduQuery` which aggregates an arbitrary number of rules (`EduRule`) and query literals (`EduLiteral`). `EduLiterals` are either `RDFReifiedStatements` (binary predicates / ternary statement literals, corresponding to reified RDF statements), `EduStatementLiterals` (non-ternary statement literals, which cannot be expressed as ordinary RDF statements), or `EduConditionLiterals` (a condition expression on variables such as $X > 5$).

Technically, it is sufficient to define a single instance of `EduLiteral` as query literal. However, by using a set of `EduLiteral` objects, all query literals together can be interpreted as the RDF result graph of the `EduQuery`, as long as the query literals are all instances of RDFReifiedStatement.

An `EduRule` consists of an `EduStatementLiteral` as its head and an arbitrary number of `EduLiterals` as its body. `EduStatementLiterals` can occur within a rule's body as well to allow reuse of other rules and recursion.[2]

`EduVariable` objects are ordinary RDF resources with the super class `RDFResource`. Being of type `EduVariable` however marks a resource to be a variable. An additional attribute allows to specify the label of a variable. Variables may occur in all places where `RDFResources` are allowed: As subject, predicate or object within `RDFReifiedStatements` as well as arguments of `EduStatementLiterals` or `EduConditionLiterals`. The class `EduVariableBinding` introduces a further extension to `EduVariable` by providing an actual value for a variable. Variable values can be either `RDFResource` or `RDFLiteral` objects.

Besides the ECDM data model the Java binding also provides a package `net.jxta.edutella.util` which contains classes for importing queries provided in various languages into the internal ECDM model or in turn exporting queries from their internal representation into different syntaxes. The current prototype includes the classes `SQL` (export of SQL queries), `Datalog`, `RDFQEL1` and `RDFQEL3` (all of them supporting import and export of queries). Any peer can plug in additional classes here to support further query languages (see [11]).

---

[2] Note, that as input format we can even allow arbitrary first order logic formulas in the body of rules, which then can be transformed into a set of rules using the Lloyd-Topor transformation [8].
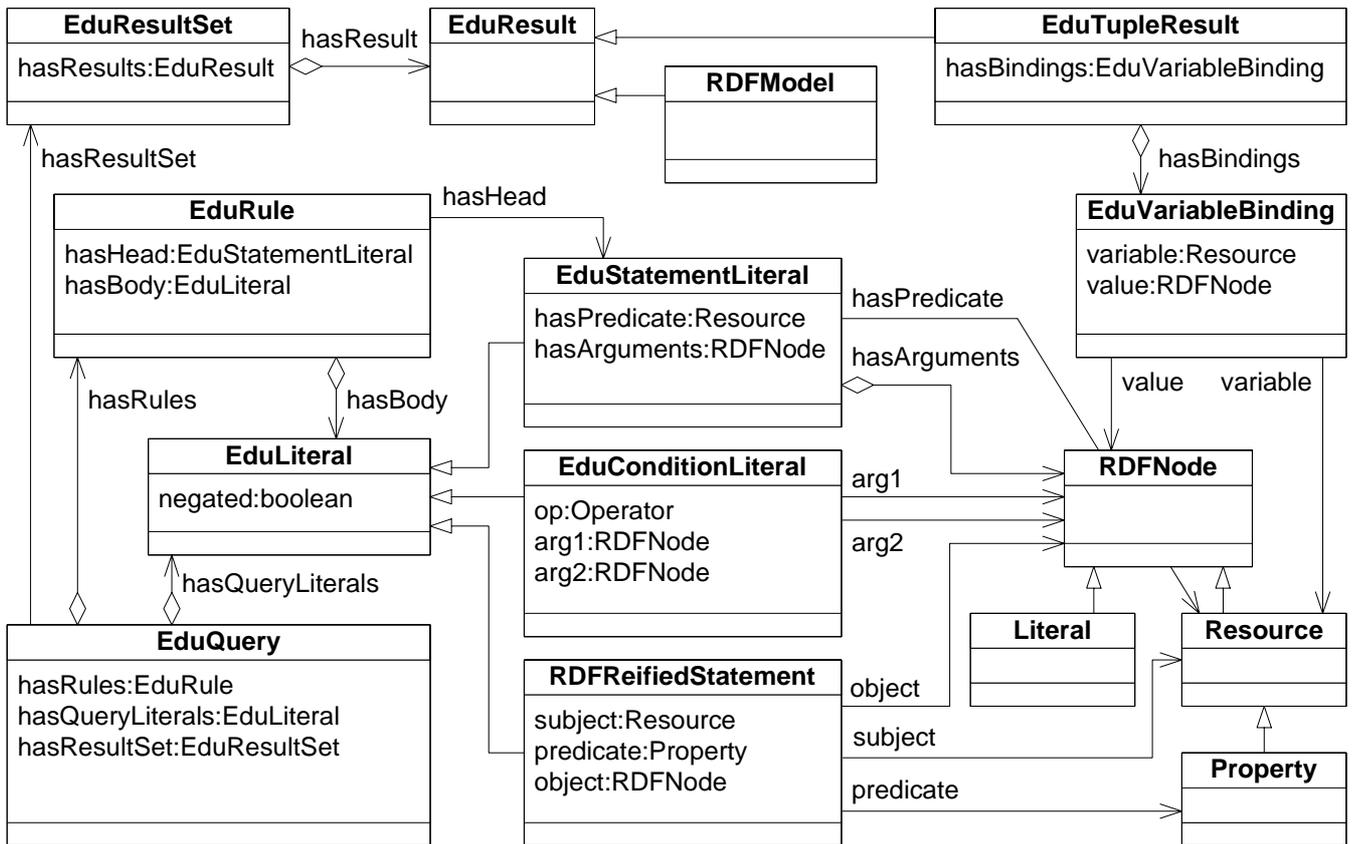
Figure 4: Edutella Common Data Model (ECDM)

2.2.3 Query Results

As a default, we represent query results as a set of tuples of variables with their bindings serialized in XML/RDF-format, as specified in Figure 4, which follows closely the convention of returning substitutions for variables occuring in queries to logic programs.

Another possibility, which has been explored recently in Web related languages focusing on querying semistructured data, is the ability to create objects as query results.

In the simple case of RDF-QEL-1, we can return as answer objects the graph representing the RDF-QEL-1 query itself with all Edutella specific statements removed and all variables instantiated. The results can be interpreted as the relevant sub graph of the RDF graph we are running our queries against (see Figure 3). When we use general RDF-QEL-i queries, we assume the structure of the answer graph to be defined by the subset of binary query literals. Note, that all variables used in the query literals are assumed to be existentially quantified, so if they are not instantiated during the query evaluation, they are represented as anonymous nodes in the RDF answer graph.

In the ECDM, `EduResult` is an abstract super class for different forms of query result representations. Results may be either represented as tuples (`EduTupleResult` objects aggregating an arbitrary number of `EduVariableBindings`) or as RDF graphs (`RDFModel` objects). In terms of relational algebra each `EduResult` object can be interpreted as one row in the result set of a relational database query. Each `EduResult` object corresponds to one match for a query. `EduResultSet` objects aggregate an arbitrary number of `EduResult` objects and repre-

sent a complete result set for an Edutella query. The individual results may be either `EduTupleResult` or `RDFModel` objects but they are all required to have the same type. When executing a query all query literals are evaluated using the necessary rules. After query execution a `EduQuery` object references an appropriate `EduResultSet` object pointing to all query results.

Classes within `net.jxta.edutella.util` also allow the import and export of query results in various other formats. Currently implemented are the classes `SQL` (for import of results provided as JDBC `ResultSets`) and `GraphViz` (for export of graph description files in GraphViz format allowing to use the free GraphViz tool to visualize query results).

2.3 Registration Service and Mediators

The *wrapper-mediator* approach introduced in [17], divides the functionality of a data integration system into two kinds of subsystems. The *wrappers* provide access to the data in the data sources using a *common data model* (CDM) and a *common query language*. The *mediators* provide coherent views of the data in the data sources by performing semantic reconciliation of the CDM data representations provided by the wrappers. Both common data model (ECDM) and common query language for the Edutella network have been defined in this paper.

Our simple "wrapping" mediators (see Figure 5) distribute queries to the appropriate peer with the restriction that queries can be answered completely by one Edutella peer. Complex 'integrating' mediators are discussed in [11].

Registration of peer query capabilities is based on (instantiated) property statements and schema information, basically telling the
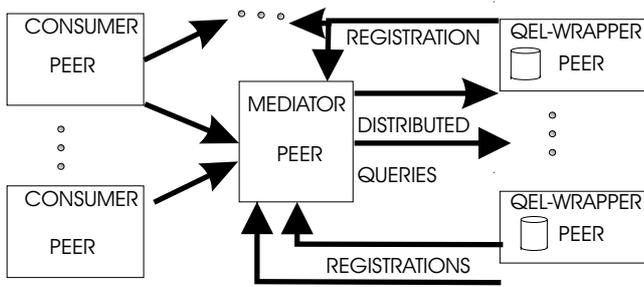
Figure 5: Query Mediator Wrapper

network, which kind of schema the peer uses, with some possible value constraints. These registration messages have the same syntax as RDF-QEL-1 queries, which are sent from the peer to the registration / query distribution hub. Additionally, the peer announces to the hub, which query level it can handle (RDF-QEL-1, RDF-QEL-2, etc.) Whenever the hub receives queries, it uses these registrations to forward queries to the appropriate peers, merges the results, and sends them back as one result set.

The packages `net.jxta.edutella.peer`, `net.jxta.edutella.provider`, `net.jxta.edutella.hub`, `net.jxta.edutella.consumer` contain interfaces to handle the distributed query mechanisms

Possible other registration methods would include specific term hierarchies which can be used as property value. A simple version could be **registerPropertyValue()**.

The query message contains not only the query itself but also information about query and result type (e.g. QEL-1, QEL-3 for queries and RDFModel, EduTupleResult for results). The returned message contains the original query in addition to its results.

3. EDUTELLA ANNOTATION

3.1 RDF(S) Annotation in a Nutshell

In order to easily provide metadata for a particular document, the annotation service provides a document viewer. Currently, the document viewer may display HTML pages, an extension for PDF documents is underway.

Furthermore, the annotation service provides a browser for RDF schema. This means that a corresponding definition, e.g. Dublin Core in RDFS[3] is loaded into the annotation tool and may be browsed. Fields for annotation are displayed according to the schema definition and may either be filled by typing or by marking and dragging information from the document viewer.

Thereby, annotations and fields for annotations may take quite a number of different guises. In our context an annotation is a set of instantiations attached to an HTML document. We distinguish (i) instantiations of RDFS classes, (ii) instantiated properties from one class instance to a datatype instance, and (iii) instantiated properties from one class instance to another class instance. Class instances have unique URIs. Instantiations may be attached to particular markups in the HTML documents, viz. URIs and attribute values may appear as strings in the HTML text.

For instance, one may decide, (i), to create an identifier for a person by instantiating HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/WBS/SHA/#HANDSCHUH from the class DC:CREATOR and for a course HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/LEHRVERANSTALTUNGEN/

WINTER/EBIZ+INTELLIGENTWEB/#COURSE from the class SWRC:SEMINAR. (ii), one may instantiate the attributes of the first identifier by names like "Siegfried Handschuh" or "Siggi". (iii), one may relate instances, e.g. the first with the second identifier by the property SWRC:TEACHES.

These types of instantiations may be considered one dimension in the metadata creation process. Another, orthogonal, dimension is defined by the way annotations are created, used and maintained:

1. *Unlinked facts* fill fields of the schema. There is no correspondence to the given document that is recognizable by the machine.

2. *Quotations* are excerpts from the document. E.g. a name like "Tim Berners-Lee" may appear in the document and also fill a field of the RDF schema description.

3. *References* are pointers to parts of the document. We use XPointer to select parts of the document. E.g. one may assert that a particular cell of a HTML table contains the name of the president of the U.S.A. — and in the right context one might expect that it is updated if it changes.

By the combination of these two dimensions (and the corresponding implications) we may emulate the metadata structure of all the different annotation tools that we currently know of (cf. [6] for a longer list of free and commercial tools).

3.2 Architecture

The Edutella annotation service is composed of the Edutella Peer structure and the KAON tool-suite [4] [9] incorporating the OntoMat Plugin Framework[5] and Annotation application [6] (cf. Figure 6).
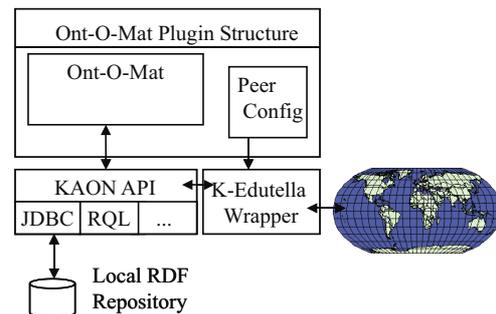

Figure 6: Ont-O-Mat as Edutella Peer

KAON is a Semantic Web tool suite originally created in isolation of Edutella. The OntoMat Framework is part of this tool suite and provides a java-based plug in structure which allows for loading services dynamically. One such service is Ont-O-Mat, which constitutes an annotation tool in the sense described above. Ont-O-Mat uses the KAON API to query for RDF schema definitions in order to build up its ontology browser. It queries for instances, attributes and relationships in order to let its users explore the current state of the knowledge base, e.g. in order to directly relate HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/WBS/SHA/#HANDSCHUH with HTTP://WWW.AIFB.UNI-KARLSRUHE.DE/LEHRVERANSTALTUNGEN/ WINTER/EBIZ+INTELLIGENTWEB/#COURSE.

The KAON API hides the actual implementation of the repository and the query language used. For instance, it allows to connect

---

[3] http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/.

[4] http://kaon.semanticweb.org/

[5] http://annotation.semanticweb.org/

to a KAON RDF repository via a simple JDBC connection or to a RQL-based repository. The repository is used in two ways. First, it stores already available metadata and serves them to Ont-O-Mat via KAON API in order to allow for coherent metadata. This way, the chance is increased that there is only *one* identifier for the person named "Siegfried Handschuh" at Institute AIFB. Our experiences have shown that without such service several identifiers for single persons are created. Second, it stores metadata created by Ont-O-Mat.

Furthermore, we provide an Edutella Wrapper for KAON (K-Edutella Wrapper), which — like the corresponding Peer Configurator GUI — is a KAON plugin. The task of the K-Edutella Wrapper is to wrap the KAON-API for QEL and vice versa. The K-Edutella Wrapper calls JXTA lower levels for services like registration, pipes, etc. in order to connect to the outside world. Thus, from the point of view of the Ont-O-Mat user, he has a tool that may directly connect the Edutella network in order to query metadata from other peers or provide metadata from his repository.

4. CONCLUSION

Our prototype scenario features a set of (already existing) peers, which we have extended with the appropriate Edutella wrappers, and which connect to the Edutella framework with the functionalities *local and distributed queries* described in Section 2. The first prototype already contains the QEL query exchange mechanism, a simple mediatior and the wrapping of different repository peer types:

1. OLR (Open Learning Repository)[3] based peers using a subset of IMS/LOM metadata;

2. DbXML-based peers [14] as a prototype for an XML repository using a simple mapping service to translate from RDF-QEL-1 queries (conjunctive queries) to Xpath queries over the appropriate XML-LOM schema;

3. AMOS-II-based peers [15] with local repositories;

4. KAON-based peers [9] allowing remote annotation [6] using an RDF-based ontology format;

5. Concept-Base, a repository with full datalog capabilities [7].

Moreover the resulting environment will allow the design and integration of other tools which make use of metadata. In addition to the Ont-O-Mat, other applications such as Conzilla [12], which uses graphs to input queries and visualize results, will benefit from the Metadata enhanced Peer-to-Peer capabilities of Edutella. These first steps being done, a certain number of ameliorations are planned. In particular we will have to tackle mechanisms that provide replication of data implementing a a modification exchange language (MEL) and that resolve scalability issues like the selection of appropriate hubs for given queries.

Acknowledgements

5. REFERENCES

[1] Dan Brickley and R. V. Guha. W3C Resource Description Framework (RDF) Schema Specification. http://www.w3.org/TR/1998/WD-rdf-schema/, March 2000. W3C Candidate Recommendation.

[2] IEEE Learning Technology Standards Committee. IEEE LOM Working Draft 6.1. http://ltsc.ieee.org/wg12/index.html, April 2001.

[3] Hadhami Dhraief, Wolfgang Nejdl, Boris Wolf, and Martin Wolpers. Open learning repositories and metadata modeling. In *International Semantic Web Working Symposium (SWWS)*, Stanford, CA, July 2001.

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

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

[6] Siegfried Handschuh and Steffen Staab. Authoring and annotating Web pages in CREAM. In *Proceedings of WWW-2002*. ACM Press, October 2002.

[7] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, and S. Eherer. ConceptBase - a deductive object base for meta data management. *Journal on Intelligent Information Systems*, 4(2):167 – 192, 1995.

[8] J. W. Lloyd and R. W. Topor. Making prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[9] Alexander Maedche, Steffen Staab, Rudi Studer, York Sure, and Raphael Volz. Seal — Tying up information integration and Web site management by ontologies. *IEEE Data Engineering Bulletin*, March 2002. http://www.research.microsoft.com/research/db/debull/.

[10] Sergey Melnik. *RDF API Draft*, January 2001. http://www-db.stanford.edu/ melnik/rdf/api.html.

[11] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: A P2P networking infrastructure based on RDF. In *Proceedings of WWW-2002*. ACM Press, 2002.

[12] Mikael Nilsson and Matthias Palmér. 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.

[13] Changtao Qu and Wolfgang Nejdl. Exploring JXTAsearch for P2P learning resource discovery. Technical report, Learning Lab Lower Saxony, University of Hannover, November 2001.

[14] Changtao Qu and Wolfgang Nejdl. Towards interoperability and reusability of learning resources: A SCORM-conformant courseware for computer science education. Technical report, Learning Lab Lower Saxony, University of Hannover, October 2001.

[15] T. Risch and V. Josifovski. Distributed data integration by object-oriented mediator servers. *Concurrency and Computation: Practice and Experience*, 13(11):933 – 953, 2001.

[16] Steve Waterhouse, David Doolin, Gene Kan, and Yaroslav Faybishenko. Distributed search in p2p networks. *IEEE Internet Computing*, 6(1):68–72, January/February 2002. Special issue on Peer-to-Peer Networking.

[17] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38 – 49, 1992.