Building Open Systems Using Edutella — Edutella framework and integration possibilities*

Peter Dolog

Learning Lab Lower Saxony
dolog@learninglab.de,
http://www.learninglab.de/~dolog

Abstract. In this report I reflect the current state of Edutella provider and consumer interface to provide description for other applications or systems how to become a peer in Edutella network, and how to query for metadata distributed in Edutella network.

1 Introduction

Internet as an open environment provides us with the opportunity to share and reuse resources already available. Heterogeneity of users and resources in the web stresses the importance of customized (personalized) delivery of the resources.

Edutella [5, 6] framework allow us to connect the resource providers and provide their metadata for queriing to other connected peers.

Edutella allow us to connect also systems which provide a metadata about learners and users and thus to allow to query the provision of learner features to other systems as well.

We proposed an architecture in [2], which takes availability of the user model in the network into account and utilize it for adaptation purposes or personalisation services. The architecture is depicted in the figure 1.

Circles represents simple providers without reasoning capabilities. Rectangles represent peers, which are able to perform programs. Multiple rectangle symbols represent metadata. Learning resources are provided through resource provider peers. Resources can be referenced by courses which represent simple learning services. Courses can be personalized by adaptation services provided by personalization peers. Courses and resources can be recommended by recommendation services or can be filtered by filtering services. Personal learning assistants support learners or user to use the network.

We discussed possibilities how to adapt course provision which is composed from resources described by metadata in RDF bindings of LOM in [1].

This paper explaines how to integrate external systems (content or resource providers) with Edutella framework, introduce QEL language used for query

* This report was done in the context of ELENA project (http://www.elena-project.org) and part of this report have appeared in several ELENA project deliverables.

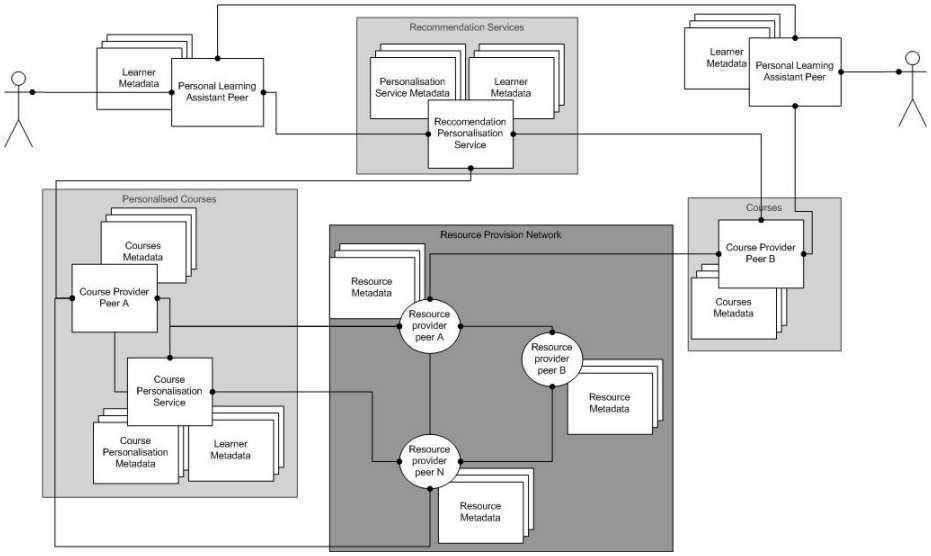

Fig. 1. ELENA architecture for personalization services. The personalisation services might be generic adaptive functionalities provided and described in common language, e.g. first order logic (see [4] for details). The generic parsonalisation services can be then reused in several cources and/or queries. The example of such generic personalisation service would be recommendation of particular course fragment based on its prerequisites what can be defined independently from topics and fragments available in the course.

Edutella and introduce interfaces for provision and query services provided ba edutella infrastructure.

The rest of the paper is structured as folows. Section 2 describes several points of view to components which are available in the network. Section 3 briefly introduce edutella framework and query language used in Edutella. Section 5 privide several possibilities how to integrate existing systems into Edutella network to provide their resources. Section 6 provides a description about interfaces which are needed to implement in Systems which want to take a part in the Edutella network.

2 Components in network

There are four main views of artefacts in a smart learning space discussed in this document:

- Software components
- Metadata components
- Resource/learning object components
- Systems

Software components realize access and provision of resource and metadata artefacts. These artefacts can be described from two points of view:

- Which technical services they provide (provision, learning, evaluation)
- Which kind of artefact they provide (resource/content, metadata, educational services)

The metadata and resource components are subjects of authoring and are changing. It means that they are related to a specific provider of software component, which supports the authoring and managing of these resources, services and metadata about them.

The systems point of view considers which systems as metadata and resource providers are connected to the ELENA network.

3 Edutella Framework

3.1 The JXTA P2P Framework

JXTA is an Open Source project [3] (or visit web page: http://www.jxta.org/) 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, see Figure 2, reproduced from [3]). 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. Therefore JXTA is a very useful framework for prototyping and developing P2P applications.

This layered approach fits very nicely into our application scenarios defined for Edutella: Edutella Services (In the future described in web service languages like DAML-S or WSDL, etc.) complement the Jxta Service Layer, building upon the JXTA Core Layer, and Edutella Applications/Front-ends (Services are also part of 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.

Educational Context Every single university usually already has a large pool of educational resources distributed over its institutions. These are under control of the single entities or individuals, and it is unlikely that these entities will give up their control, which explains why all approaches for the distribution of educational media based on central repositories have failed so far. Furthermore, setting up and maintaining central servers is costly. The costs are hardly justifiable, since a server distributing educational material would not directly

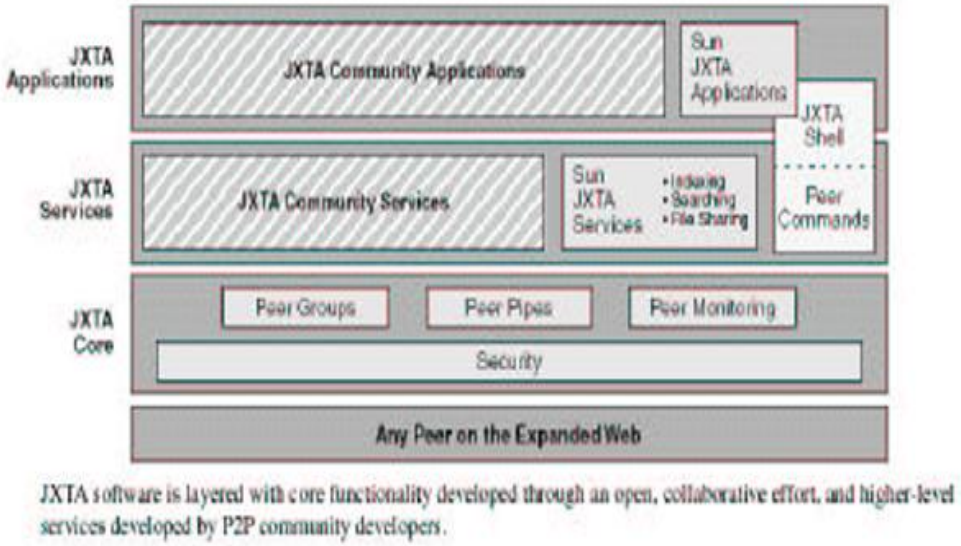

Fig. 2. JXTA layers

benefit the sponsoring university. We believe that, in order to really facilitate the exchange of educational media, approaches based on metadata-enhanced peer-to-peer (P2P) networks are necessary.

In a typical P2P-based e-learning scenario, each university acts not only as a content provider but also as a content consumer, including local annotation of resources produced at other sites. As content provider in a P2P network they will not lose their control over their learning resources but still provide them for use within the network. As a content consumer 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. P2P networks have already been quite successful for exchanging data in heterogeneous environments, and have been brought into focus with services like Napster and Gnutella, providing access to distributed resources like MP3 coded audio data. However, pure Napster and Gnutella like approaches are not suitable for the exchange of educational media. For example, the metadata in Gnutella is limited to a file name and a path. While this might work for files with titles like Madonna - Like a Virgin, it certainly does not work for Introduction to Algebra - Lecture 23. Furthermore, these special purpose services lead to fragmented communities, which use special purpose clients to access their service. The educational domain is in need of a much richer metadata markup of resources, a markup that is often highly domain and resource type specific. In order to facilitate interoperability and reusability of educational resources, we need to build a

system supporting a wide range of such resources. This places high demands on the interchange protocols and metadata schemata used in such a system, as well as on the overall technical structure. Also, we do not want to create yet another special purpose solution which is outdated as soon as metadata requirements and definitions change.

Our metadata based peer-to-peer system therefore has to be able to integrate heterogeneous peers (using different repositories, query languages and functionalities) as well as different kinds of metadata schemas. We find common grounds in the essential assumption that all resources maintained in the Edutella network can be described in RDF, and all functionality in the Edutella network is mediated through RDF statements and queries on them. For the local user, the Edutella network transparently provides access to distributed information resources, and different clients/peers can be used to access these resources. Each peer will be required to offer a number of basic services and may offer additional advanced services.

3.2 Edutella Services

Edutella connects highly heterogeneous peers (heterogeneous in their uptime, performance, storage size, functionality, number of users etc.). However, each Edutella peer can make its metadata information available as a set of RDF statements. Our goal is to make the distributed nature of the individual RDF peers connected to the Edutella network completely transparent by specifying and implementing a set of Edutella services. Each peer will be characterized by the set of services it offers.

Query Service Peers register the queries they may be asked through the query service (i.e., by specifying supported 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 statements are sent back to the requesting peer.

Edutella Replication This service is complementing local storage by replicating data in additional peers to achieve data persistence / availability and workload balancing while maintaining data integrity and consistency. Since Edutella is mainly concerned with metadata, replication of metadata is our initial focus. Replication of data might be an additional possibility (though this complicates synchronization of updates). This service is not implemented yet.

Edutella Mapping, Mediation, Clustering While groups of peers will usually agree on using a common schema (e.g., SCORM or IMS/LOM for educational resources), extensions or variations might be needed in some locations. The Edutella Mapping service will be able to manage mappings between different schemata and use these mappings to translate queries over one schema X to queries over another schema Y. Mapping services will also provide interoperation between RDF- and XML-based repositories. Mediation services actively mediate access between different services, clustering services use semantic information to set up semantic routing and semantic clusters. These services are not implemented yet.

RDF-QEL-i Language Levels In the definition of the Edutella query exchange language, several important design criteria have been formulated:

- Standard Semantics of query exchange language, as well as a sound RDF serialization. Simple and standard semantics of the query exchangelanguage is important, as transformations to and from this language have to be performed within the Edutella peer wrappers, which have to preserve the semantics of the query in the original query language. Additionally, a sound encoding of the queries in RDF to be shipped around between Edutella peers has to be provided.
- Expressiveness of the language. We want to interface with both simple graph based query engines as well as SQL query engines and even with inference engines. It is important that the language allows expressing simple queries in a form that simple query providers can directly use, while allowing for advanced peers to fully use their expressiveness.
- Adaptability to different formalisms. The query language has to be neutral to different representation semantics, it should be able to use any predicates with predefined semantics (like rdfs:subclassOf), but not have their semantics built in, in order to be applicable to different semantic formalisms used in the Edutella peers. It should be as easily connected to simple RDFS repositories as to relational databases or object-relation ones, and inference systems, which all have different base semantics and capabilities.
- Transformability of the query language. The basic query exchange language model must be easy to translate into many different query languages (both for importing and exporting), allowing easy implementation of Edutella peer wrappers.

Edutella follows a layered approach for defining the query exchange language. Currently we have defined language levels RDF-QEL-1, -2, -3, -4 and -5, differing in expressivity. The simplest language (RDF-QEL-1) can be expressed as unreified RDF graph, the more complex ones are more expressive than RDF itself and therefore have to be expressed using reified RDF statements. All language levels can be represented through the same internal data model [5].

Currently, Edutella providers support RDF-QEL-3 language.

RDF-QEL-1. The RDF-QEL-1 syntax design is driven by its simplicity and readability: Following a QBE (Query By Example) paradigm queries are represented using ordinary RDF graphs having exactly the same structure as the answer graph, with additional annotations to denote variables and constraints on them. Any RDF graph query can be interpreted as a logical (conjunctive) formula that is to be proven from a knowledge base.

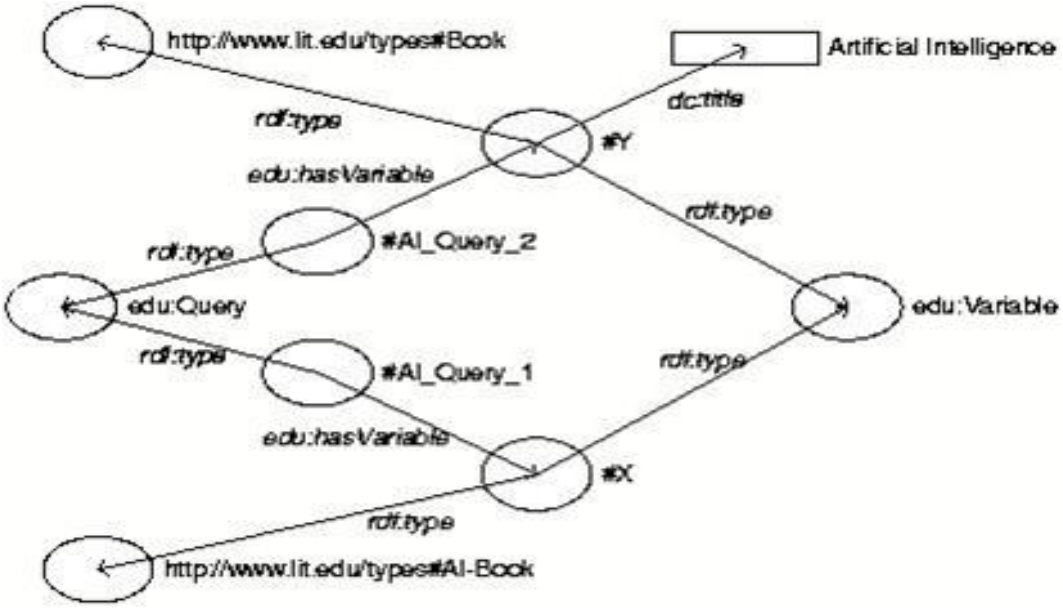

Fig. 3. Example Query in RDF-QEL-1, Unreified Format

Since disjunction cannot be expressed in RDF-QEL-1 our example query has to be split into two separate sub queries (Figure 3).

```
<edu:QEL1Query rdf:ID="AI_Query_1">
<edu:hasVariable rdf:resource="#X"/>
</edu:QEL1Query>

<edu:Variable rdf:ID="X" rdfs:label="X">
  <rdf:type rdf:resource="http://www.lit.edu/types#AIBook"/>
</edu:Variable>

<edu:QEL1Query rdf:ID="AI_Query_2">
  <edu:hasVariable rdf:resource="#Y"/>
</edu:QEL1Query>

<edu:Variable rdf:ID="Y" rdfs:label="X">
```

```
  <rdf:type rdf:resource="http://www.lit.edu/types#Book"/>
  <dc:title>Artificial Intelligence</dc:title>
</edu:Variable>
```

RDF-QEL-2. Extending RDF-QEL-1 with disjunction leads to RDF-QEL-2. As this language is no longer purely assertional, it cannot be expressed directly in RDF without talking about RDF triples in order to combine them logically. For this purpose, we utilize the RDF construct called reification. Reifying an RDF statement involves creating a model of the RDF triple in the form of an RDF resource of type Statement. This resource has as properties the subject, the predicate and the object of the modeled RDF triple. Such reified statements are the building blocks for each query and can, in RDF-QEL-2, linked together by an AND-OR tree. In RDF-QEL-2 the example query reads like

```
<edu:Variable rdf:about="#X" rdfs:label="X"/>
<edu:And rdf:about="#andbagID">
  <rdf:_2 rdf:resource="#st2"/>
  <rdf:_1 rdf:resource="#st3"/>
</edu:And>

<edu:Or rdf:about="#orbagID">
  <rdf:_1 rdf:resource="#andbagID"/>
  <rdf:_2 rdf:resource="#st1"/>
</edu:Or>

<edu:QueryStatement rdf:about="#st1">
  <rdf:subject rdf:resource="#X"/>
  <rdf:object rdf:resource="http://www.lit.edu/types#AIBook"/>
  <rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
</edu:QueryStatement>

<edu:QueryStatement rdf:about="#st2">
  <rdf:subject rdf:resource="#X"/>
  <rdf:object rdf:resource="http://www.lit.edu/types#Book"/>
  <rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
</edu:QueryStatement>

<edu:QueryStatement rdf:about="#st3">
  <rdf:object>Artificial Intelligence</rdf:object>
  <rdf:subject rdf:resource="#X"/>
  <rdf:predicate rdf:resource="http://purl.org/dc/elements/1.1/title"/>
</edu:QueryStatement>
```

The advantage of the RDF-QEL-2 form is that queries can easily be visualized using a query graph. The Conzilla query interface [7] is based on a subset of UML, using the UML specialization relationship for logical OR and the UML aggregation relationship for logical AND. As shown in figure A.2, our current prototype uses a graph-view, which is displayed as ordinary RDF with the exception that the triplets searched for (which are reified in RDF-QEL-i, where

$n > 1$) are displayed as dashed arrows indicating that they are searched for. The logical view is displayed as a parse tree. This is the logical combination of the primitive statements, showing which combinations that should be matched at the same time in order for the query to succeed. The connections between the different views are displayed by highlighting the corresponding parts.

Figure A.2 Edutella Graph Query Interface Queries can be stored and reused later, thus we can work with a library of queries that can be combined to new queries. Those queries can either be used as is or as templates, where sub-strings, numerical values, etc are filled in. Details of sub-queries can be suppressed by hiding them in detailed maps that can be presented hierarchically.

RDF-QEL-3. Going a step further, we might actually choose to skip RDF-QEL-2 in favor of RDF-QEL-3, which allows conjunction, disjunction and negation of literals. RDF-QEL-3 is essentially Datalog. Hence, the query is a set of Datalog rules, which can be encoded easily using reified statements (as for RDF-QEL-2), introducing additional constructs for negation and implication. As long as queries are non-recursive this approach is relationally complete. The example query expressed in RDF-QEL-3 resembles the internal Datalog model described above.

```
<edu:QEL3Query rdf:ID="AI_Book_Query">
<edu:hasQueryLiteral rdf:resource="st0"/>
<edu:hasRule rdf:resource="r1"/>
<edu:hasRule rdf:resource="r2"/>
</edu:QEL3Query>

<edu:Variable rdf:ID="X" rdfs:label="X"/>

<edu:Rule rdf:ID="r1">
<edu:hasHead rdf:resource="st0"/>
<edu:hasBody rdf:resource="st2"/>
<edu:hasBody rdf:resource="st3"/>
</edu:Rule>

<edu:Rule rdf:ID="r2">
<edu:hasHead rdf:resource="st0"/>
<edu:hasBody rdf:resource="st1"/>
</edu:Rule>

<edu:QueryStatement rdf:ID="st0">
<edu:predicate rdf:resource="aibook"/>
<edu:arguments>
<rdf:Seq>
<rdf:_1 rdf:resource="#X"/>
</rdf:Seq>
</edu:arguments>
</edu:QueryStatement>
```

```
<edu:QueryStatement rdf:ID="st1">
<rdf:subject rdf:resource="#X"/>
<rdf:object rdf:resource="http://www.lit.edu/types#AIBook"/>
<rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
</edu:QueryStatement>

<edu:QueryStatement rdf:ID="st2">
<rdf:subject rdf:resource="#X"/>
<rdf:object rdf:resource="http://www.lit.edu/types#Book"/>
<rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
</edu:QueryStatement>

<edu:QueryStatement rdf:ID="st3">
<rdf:object>Artificial Intelligence</rdf:object>
<rdf:subject rdf:resource="#X"/>
<rdf:predicate rdf:resource="http://purl.org/dc/elements/1.1/title"/>
</edu:QueryStatement>
```

Further RDF-QEL-i Levels.

- RDF-QEL-4: RDF-QEL-4 allows recursion to express transitive closure and linear recursive query definitions, compatible with the SQL3 capabilities. So a relational query engine with full conformance to the SQL3 standard will be able to support the RDF-QEL-4 query level.
- RDF-QEL-5: Further levels allow arbitrary recursive definitions in stratified or dynamically stratified Datalog, guaranteeing one single minimal model and thus unambiguous query results ([8]).

RDF-QEL-i-A: Support for the usual aggregation functions as defined by SQL2 (e.g. COUNT, AVG, MIN, MAX) will be denoted by appending "-A" to the query language level, i.e. RDF-QEL-1-A, RDF-QEL-2-A, etc. RDF-QEL-i-A includes these aggregation functions as edu:count, edu:avg, edu:min, etc. Additional "foreign" functions like edu:substring etc. to be used in conditions might be useful as well, but have not been included yet in RDF-QEL-i-A.

4 Providers

Following providers are available in Edutella:

- File-based metadata
- Relational database which supports nested queries
- ConceptBase

The file-based provider provides access to RDF files. This provider is able to handle several files. It constructs one RDF model in main memory from files. It is implemented by the JENA semantic web toolkit (see http://www.hpl.hp.com/semweb/jena-top.html for details). A relational database provider provides access to metadata, which are stored in a relational database. The metadata have to be stored in

one table. If they are stored in multiple tables, a view of RDF triples is needed. The provider assumes that the database supports nested queries. A ConceptBase provider provides access to ConceptBase. ConceptBase is a multi-user deductive object manager mainly intended for conceptual modelling and coordination in design environments. The system implements O-Telos, a dialect of Telos which amalgamates properties of deductive and object-oriented languages (see http://www-i5.informatik.rwth-aachen.de/CBdoc/cbflyer.html for details). Edutella providers accept RDF based queries in RDF-QEL-3. Until now, there is only a library of functions, which can be used in an external programming environment to access the functionality of an Edutella peer.

5 Scenarious for Integration

There are three possible scenarious for integration:

- Embedding Edutella query service and provider interface into an educational node
- Invoking Edutella query service from an educational node
- Implementing a wrapper for accessing the metadata repository of an educational node

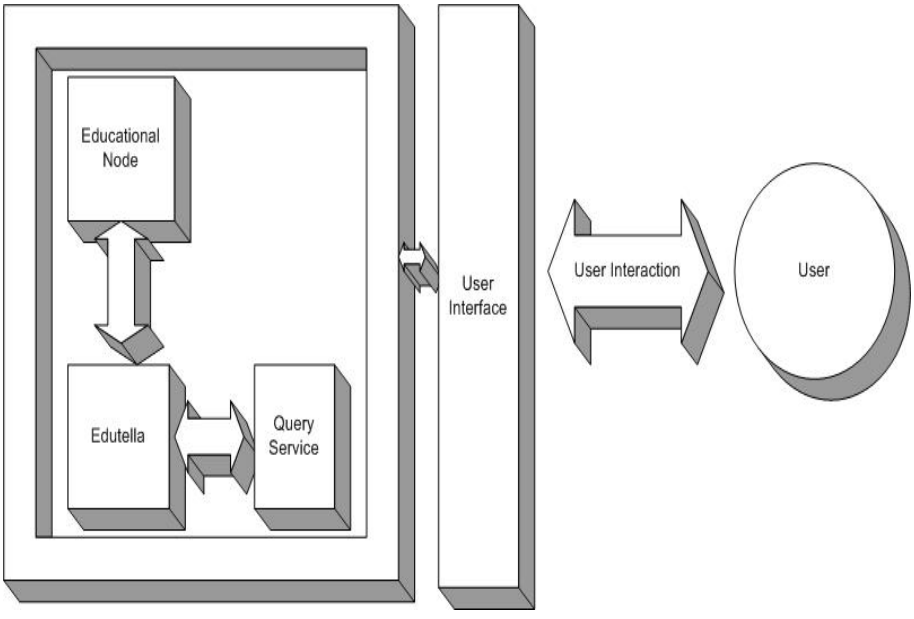

Fig. 4. Embedding Edutella query service and provider interface into an educational node

A schema for the first integration scenario *Embedding Edutella query service and provider interface into an educational node* is depicted in fig. 4. This scenario assumes that the program code of Edutella will be added to the program code of the educational node. The Edutella services thus extend services provided by the educational node. This scenario can be taken into account for educational nodes, which:

- are programmed in JAVA
- provide JAVA interfaces to their repositories and
- enable to customise their source codes.

As we discussed in the previous section, file-based and relational database providers are already implemented and can be considered as the most natural choice for accessing metadata repositories of an educational node. If the educational node provides a different type of metadata repository from those mentioned, a transformation procedure has to be implemented. The procedure transforms metadata from an educational node metadata repository schema to an appropriate schema for querying by Edutella.

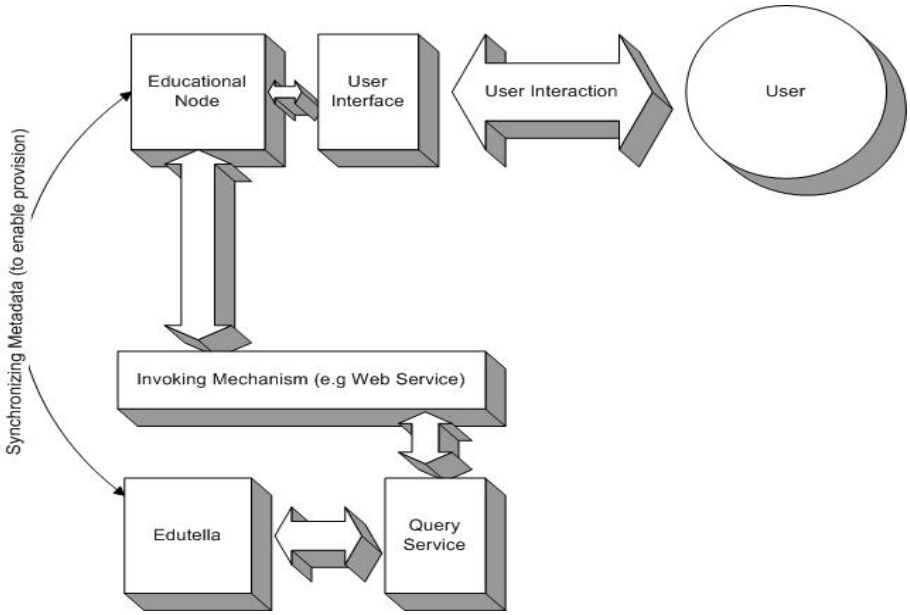

Fig. 5. Providing and querying in the Edutella network from an external educational node (invocation mechanism is not implemented yet)

Figure 5 depicts a schema for the second scenario *invoking Edutella query service from an educational node*. This scenario assumes a stand-alone Edutella peer and an educational node. The query service is provided through some kind

of invoking mechanism (e.g. web service). If educational node metadata provision is needed, synchronization between the metadata repository of the educational node and the metadata repository of the Edutella peer must be provided. This scenario can be considered for educational nodes, which:

- Are programmed in a different language than JAVA
- Do not allow to customize its program code or
- Do not allow for extending the program code.

For the mentioned synchronization mechanism, an appropriate time interval for synchronization has to be assigned. If the educational node metadata repository schema is different from RDF, the transformation procedure has to be provided together with synchronization. The up-to-dateness of metadata at the Edutella peer should also be considered regarding the appropriate time interval for and between synchronization.

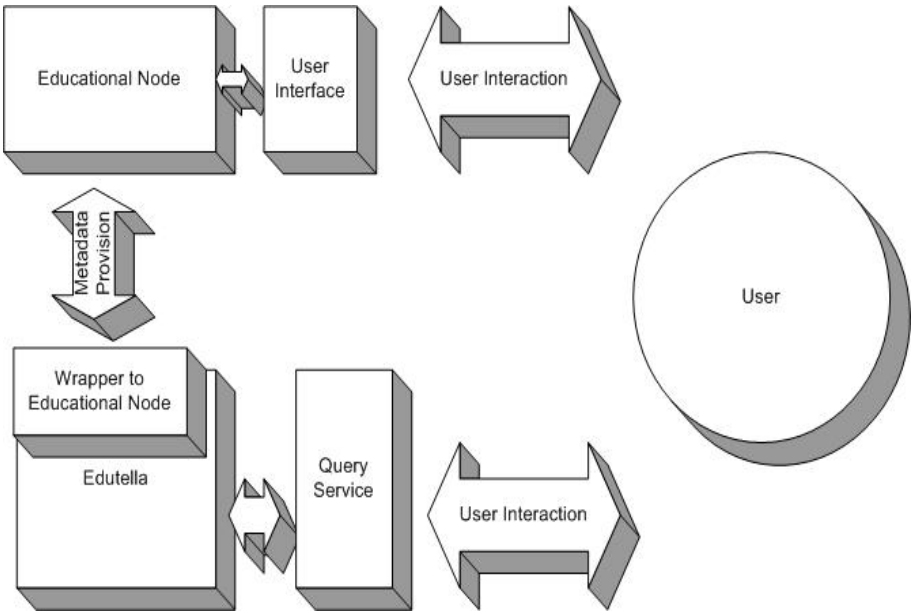

Fig. 6. Wrapper access to the educational node

Figure 6 depicts a schema for the third integration scenario *implementing a wrapper for accessing the metadata repository of an educational node.* This scenario assumes a stand-alone educational node and Edutella peer. The Edutella query service is provided to a user through an appropriate user interface. The metadata provision is realized through a wrapper to the metadata repository of the educational node. This scenario can be considered for educational nodes, which:

– Provide their metadata repository schema specification
– Provide metadata repository access programming interface appropriate for the JAVA programming language, or
– Do not allow to customize the educational node program code.

If the educational node metadata repository schema is different from RDF, a transformation procedure has to be part of the wrapper.

6 Interfaces

Figure 7 depicts a schema for one Educational Peer from the software component point of view.

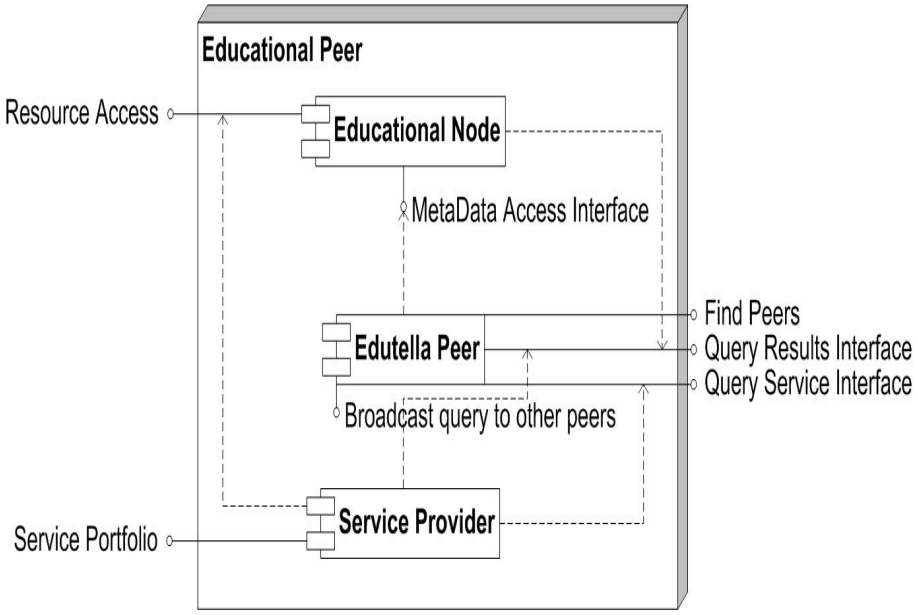

Fig. 7. High level component architecture of educational peer in smart learning space

There are three high level components depicted in fig. 7:

– Educational Node (EN) — any existing or possibly future system, which provides content and metadata,
– Edutella Peer Provider — standard Edutella provider, and
– Service Provider — optional (not implemented yet) component, which provides integrated services of Edutella peer and content management system.

EN will provide interface to its content and metadata. The MetaData Access interface is used by Edutella Peer component. This interface comprises the

metadata query interface and the metadata results interface. The SQL or file based access to metadata is preferred. If there is another interface, the wrapper should be implemented as an implementation of ProviderConnection interface. The Edutella Peer provides Find Peers interface, Query Results interface and Query Service interface. They can also be used outside of the educational peer. The Edutella Peer can stand as a metadata provider (can be queried) or consumer (consumes query results from broadcasted query to other peers Broadcast query to other peers interface). The Service Provider component can be developed, which will provide access to a service portfolio. This interface can also be used outside of the educational peer. The Service Provider component accesses the Query Service and Query Results interfaces to be able to compose the service portfolio. It also uses the Resource Access interface to be able to provide resources within the educational service. The integration scenarious discussed in previous section are possible realizations of usage relationships (dashed arrows).

6.1 Simple Edutella consumer and provider

This chapter describes a simple implementation of an Edutella consumer and provider. This is work in progress and thus subject to changes. The structural model of implementation for a simple Edutella provider is depicted in fig. 8.

An Edutella peer component provides a query service interface and query results interface. The query interface is used by the service provider, but can be exported directly to a user. The query interface can also be used by an LMS as user interface provider. In the current implementation of QueryService interface the query results are expressed in RDF. The generation of appropriate results for a user will be the aim of the user interface providers (LMS). Structural model of a simple implementation of the QueryService interface is depicted in fig. 9.

The interface for peers searching and for broadcasting query need not to be provided; it could be and it is now internal functionality of Edutella provider. Query results interface can be used by LMS and by service provider. Query results are provided in RDF. Apendix C: Interface specification for Edutella provider and consumer The ProviderConnection interface specification:

```
public interface ProviderConnection extends PooledConnection {
/**
 * returns a provider description
 *
 * @return human readable description of the provider
 */
    public String getDescription();

/**
 * calculates a result for a query.
 *
 * @param query the query to process
 * @return result set containing the result(s)
 */
```

```
    public EduResultSet executeQuery(EduQuery eduquery);// handle query

/**
 * initializes the connection.
 *
 */
public void init();

/**
 * closes the connection.
 *
 */
public void close();

/**
 * tests the connection.
 *
 * @return true, if connection is still usable
 */
public boolean validate();
}
```

The QueryService interface specification:

```
public interface QueryService {
/**
 * distributes a query to all discovered providers.
 * Results are sent to the ResultListener
 * provided by the caller.
 *
 * @param query the query to execute
 * @param listener results are sent to this ResultListener
 */
public void executeQuery(EduQuery query, ResultListener listener);


/**
 * distributes a query to a specific provider. Results are sent
 * to the ResultListener provided by the caller.
 *
 * @param query the query to execute
 * @param listener results are sent to this ResultListener
 * @param service information on the peer to be queried
 */
public void executeQuery(
EduQuery query,
ResultListener listener,
ServiceInfo service);

/**
```

```
 * cancels the query which was posed using this listener.
 * @param listener
 */
public void cancelQuery(ResultListener listener);

/**
 * returns a list of all available services
 *
 */
public Iterator getServices();

/**
 * checks if a specific service is still available
 */
public boolean isAvailable(ServiceInfo svc);

}
```

.

References

[1] Peter Dolog, Rita Gavriloaie, Wolfgang Nejdl, and Jan Brase. Integrating adaptive hypermedia techniques and open rdf-based environments. In *Proc. of 12th International World Wide Web Conference*, Budapest, Hungary, May 2003.

[2] Peter Dolog and Wolfgang Nejdl. Challenges and benefits of the semantic web for user modelling. In *In Proc. of AH2003 — Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, User Modelling Conference 2003*, Pittsburgh, PA, June 2003.

[3] L. Gong. Project jxta: A technology overview. Technical report, SUN Microsystems. http://www.jxta.org/project/www/docs/TechOverview.pdf. Accessed on October 25, 2002.

[4] Nicola Henze and Wolfgang Nejdl. Logically characterizing adaptive educational hypermedia systems. In *Proc. of the AH'2003 - Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, Budapest, Hungary, May 2003.

[5] 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 *In Proc. of 11th World Wide Web Conference*, Hawaii, USA, May 2002.

[6] Wolfgang Nejdl, Martin Wolpers, Wolf Siberski, Christoph Schmitz, Mario Schlosser, Ingo Brunkhorst, and Alexander Lser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *Proc. of 12th International World Wide Web Conference*, Budapest, Hungary, May 2003.

[7] Mikael Nilson and Matthias Palmer. Conzilla — towards a concept browser. Technical Report CID-53, TRITA-NA-D9911, Department of Numerical Analysis and Computing Science, KTH, Stockholm, 1999., 1999.

[8] R.C.Przymusinski. Every logic program has a natural stratification and an iterated least fixed-point model. In *Proceedings of the ACM Symposium on Principle of Database Systems (PODS)*, pages 11–21, 1998.

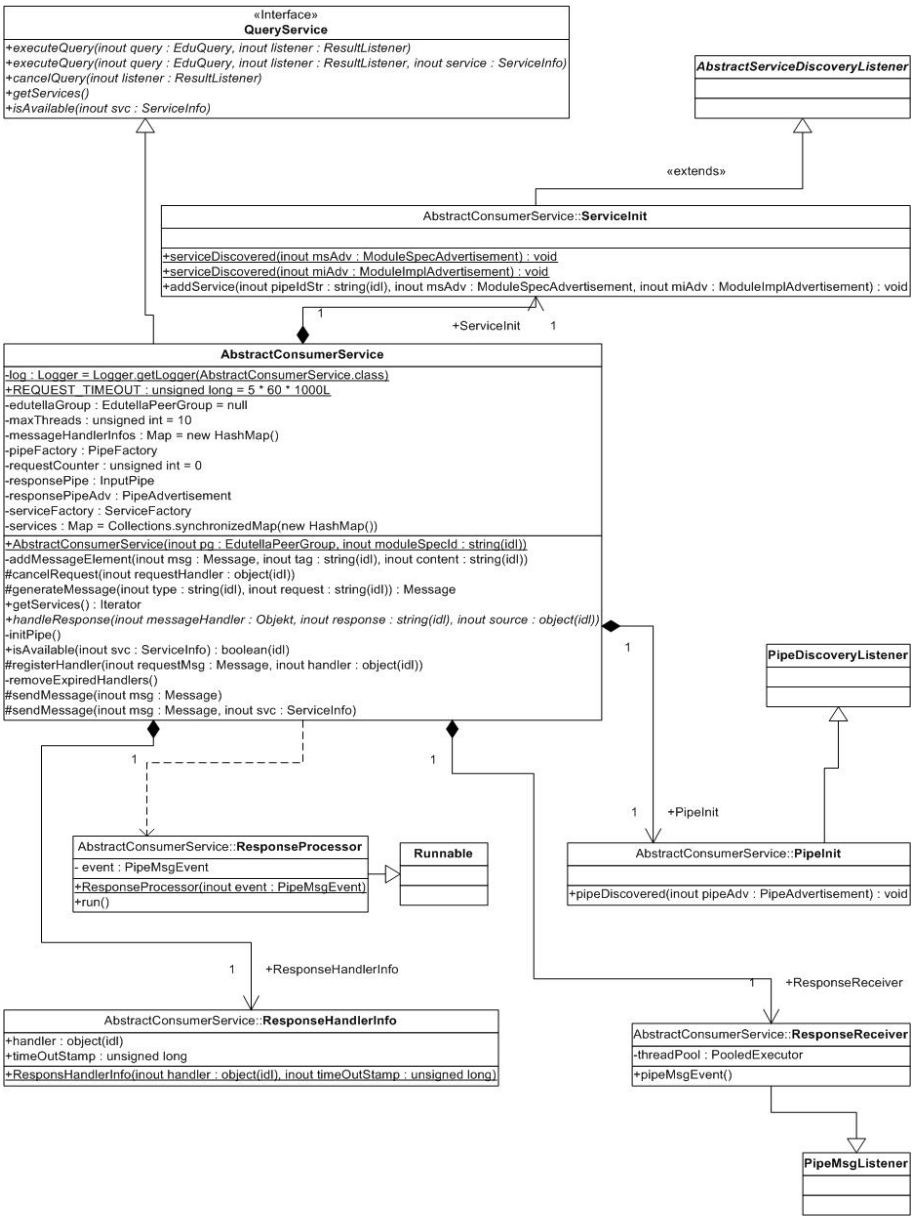

Fig. 8. Class diagram of simple implementation of QueryService interface

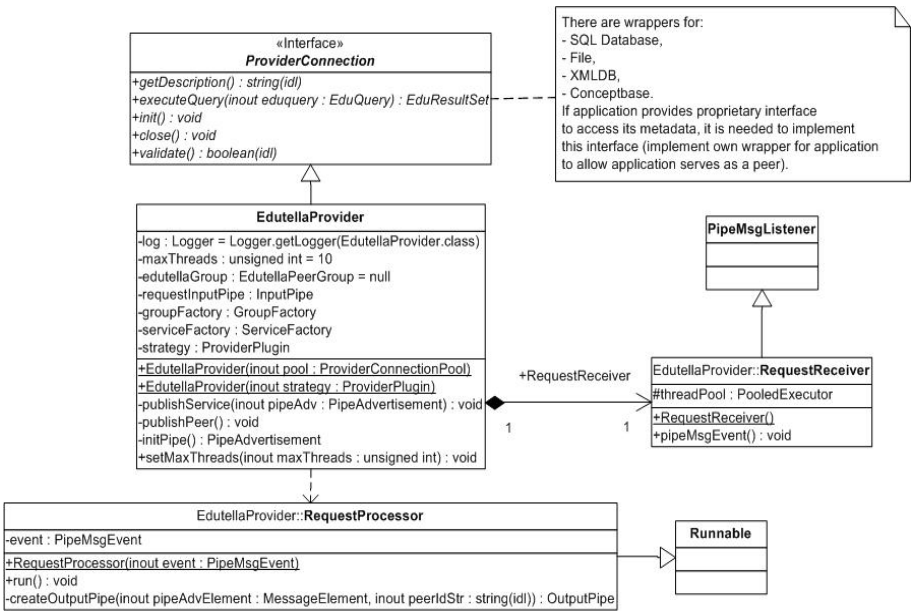

Fig. 9. Class diagram of implementation of ProviderConnection interface.