Interoperability for Peer-to-Peer Networks: Opening P2P to the rest of the World

Ingo Brunkhorst and Daniel Olmedilla

L3S Research Center and University of Hanover, Hanover, Germany
{brunkhorst,olmedilla}@l3s.de

Abstract. Due to the information growth, distributed environments are offered as a feasible and scalable solution. Peer-to-Peer (P2P) networks have become one of the most important and used distributed environments inside (and outside) the e-learning community. They bring many advantages such as high flexibility for peers to dynamically join or leave the network, scalability, autonomy and high resilience against peer failures. However, every single one of them typically uses an interface specifically developed for that network, and it requires every peer to implement it in order to join. This is leading to increased development costs for potentially new participants of the network, and usually makes different P2P networks unable to interact with other systems and environments, isolating the network as a whole. In this paper, we report on a solution based on a proxy-based architecture and semantic mappings in order to allow the sharing of content between the set of peers inside a P2P network and other systems outside the network. Furthermore, we present an open-source implementation of the modules described in the paper.

1 Introduction

The World Wide Web has become a well accepted medium for communication among people for private, academic and business affairs. As a consequence, the amount of digital material that is sent along and stored in the network increases rapidly. Obviously, learning is not unaffected by this trend, and the amount of Learning Objects (LO's henceforth) in school, academic and business environments continues to grow rapidly. As a consequence of this evolution, the focus shifts to new questions, like for example "Where shall the LO's be stored?", "Who manages them?" or "Are they easily findable?".

Dynamic sharing of information from end user machines was prohibitively costly in the past due to the lack of storage capacity and network bandwidth available for most desktop computers. As a consequence, networks of computers were mostly reduced to set of interconnected, powerful servers. In this configuration, it is relatively simple to know which servers are available and which information is available where to whom. This is also the typical architecture in business coalitions where several companies share their assets within a network of partners.

On the other hand, with the boom of web-based file-sharing services (e.g., Napster, Gnutella, Morpheus), peer-to-peer (P2P for brevity) networks have become more relevant. The advantages of the P2P approach include: high flexibility for peers to join or leave the network dynamically, scalability (recently it was shown that for really large networks, a hybrid solution with super-peers scales better [13]), autonomy (as peers do not relinquish control over their resources) and high resilience against peer failures. The main disadvantages are that P2P networks require constant management, as peers join and leave continuously, thereby producing an extra load on the network and slowing down response time during search. The use of specific interfaces for each network usually means, as no standards exist yet, that only peers implementing such interfaces can query for, or provide content in the network.

Obviously, peers must implement specific P2P network interfaces in order to join them, which is typically a different one for each network. This means an extra effort for systems willing to connect, since such systems need to develop a different interface for each network they want to join. This barrier makes P2P networks unable to interact with other systems and environments. However, in many cases such other systems already have a (possibly standardized) interface that could be adapted by re-using connecting components. In this paper, we report on a solution for interoperability in the Edutella P2P network [14] in order to allow the sharing of content from peers within the network with other systems and environments outside the network. Our approach is based on a proxy-based architecture as well as on modules that provide semantic mappings capabilities, thus allowing the communication between a P2P network and any other system. In addition, we provide open-source implementations of the components described in order to encourage re-usability as well as a mapping editor in order to ease the tasks for administrators and developers.

The paper builds on previous work of the authors [17] and is organized as follows: Section 2 starts with an introduction of Edutella, the P2P network we based our work on. The general requirements for interoperability of systems and the assumptions we made for our work are described in Section 3. We continue in Section 4 with a description of our proxy based architecture followed by an introduction of the module with semantic mappings in Section 5. An introduction to an editor developed to help users on the task of specifying such mappings is provided in Section 6. Finally, Section 7 describes related work followed by Section 8 with the conclusions of the paper and a discussion of further work.

2 Edutella

Often, learning object providers do not want to abandon control over their resources to a third party, not even among the members of a coalition. The same concern about abandoning control also often applies to individuals, who may not want to give away their content to any centralized repository. In order to deal with this issue, distributed environments have shown to be a feasible solution for interconnection, integration and access to large amounts of information. P2P networks are one example of the impact the distribution of information might have in the sharing of information. In such networks, peers can offer various ser-

vices to the user ranging from search and delivery of content, to personalization and security services. In addition, they contribute to the solution of managing the information growth, and allow every learning resource provider to offer its information without loosing control over it.

The Edutella P2P network [14] was developed with these principles as main design requirements. Edutella is a schema-based P2P network for an open world scenario in which LO's are freely offered (at not charge) and everybody is able to join (no agreement with an existing member of the network is required). It has various service facilities implemented, like for example query or publishing/subscription. Schema-based means that peers interchange RDF meta-data (data about data) among each other but not the resources themselves, that is, they interchange information about e.g. title, description, language and authors of a resource. This information can be queried using the QEL query language [15] (based on Datalog). Metadata interchange and search services provide the basic infrastructure needed to retrieve information about resources and services.

3 Requirements and Assumptions

It is important to note that in this paper we only consider the sharing of metadata about LO's. While this metadata is typically available, the learning object itself might not. Therefore, we do not deal with negotiations for the actual use of LO's by users here.

Admittedly, providing transparent access to all available repositories would be easy if all players would use the same metadata profile, query language, storage layer and communication protocol. However, this is not going to happen in the very near future due to the lack of a standard and the proprietary solutions already adopted by most of them.

In the following, we explain what requirements LO's repositories must satisfy in order to achieve interoperability and which are the assumptions within our network.

Common Communication Protocol and Interface. Repositories provide different access methods and interfaces over, among others, Web Services, different Remote Procedure Call methods, HTTP forms or even other appropriate solutions. In order to be able to communicate to each other, it is needed that they agree on a common protocol and a common interface. In this paper, we built on the methods specified in the Simple Query Interface [22] initiative (SQI for brevity), a rapidly maturing standard, using its Web Service binding.

Common Query Language. At the lower levels of data management, metadata is stored in different kinds of repositories, such as relational databases, RDF repositories, file systems, XML stores, etc. On top of this lower level, repositories expose their content through different search and query languages. Some examples are SQL, XQuery, QEL or CQL. In our system we have several wrappers implemented in order to provide access to the most common repositories (relational databases, RDF repositories, RDF files, etc.). Common to all of them is that the wrapper receives a query in QEL and transforms it into the local query language.

Common Metadata Profile. Although IEEE LOM [1] is becoming a standard for e-learning metadata, many repositories are based on specific profiles that may include extensions and specific value spaces. This means that a mapping needs to be provided [12]. This need even increases when content exchanged covers several domains. There are then two possibilities here: either each system maps its schema to a second system schema (in which case we reach semantic interoperability by means of pair of mappings [2, 8] or a common global schema is provided and both systems must map into that common schema. Later in the paper, a section on semantic mappings provides a longer explanation and describes a module we developed which allows both approaches.

It is important to notice that although we assume the configuration described above it could be perfectly possible to use a different query language than QEL, a different communication protocol than Web Services and a different interface than SQI though our implementations currently do not support it.

4 Proxying Interoperability

P2P networks are dynamic networks where peers can act as server and client indistinctly and peers might freely join and leave the network over the time. Obviously, peers must implement the specific P2P network protocol in order to connect to it. Consumers and providers try to implement standard interfaces in order to maximize the implementation costs and effort. If they want to access or expose content in a P2P network, this requires the additional extra effort of implementing the specific network interface (one for each different P2P network to be accessed). This barrier makes P2P networks unable to interact with each other or with other systems and environments.

In order to solve this problem, we based our solution in proxies that are used to connect peers in a P2P network with the "outside" world. These proxies bridge two systems with different capabilities by means of implementing the protocol and/or interface supported by each system respectively. This way, a proxy is able to forward requests and responses from one system to another.

Nowadays, systems try to provide their services/resources via standard interfaces like SQI [22] or OKI [25]. In our case, we have implemented proxies able to bridge the proprietary[1] JXTA/Edutella protocol and interface into a the Web Service binding of the Simple Query Interface.

Taking the P2P network as a reference, there are two different desirable scenarios [18]:

1. **An external consumer/client wants to query content in the P2P network.** For example, let us suppose that we would like to offer the content of a P2P network via Web Services and/or in a web site. The first solution would be to make the (web) server join the P2P network. However, the load

[1] Here we use the term "proprietary" to emphasize that this protocol is not standard for P2P networks but it does not mean it is not open. In fact, JXTA/Edutella is open-source and anyone can use it easily

of the server would increase considerably and even some problems could arise in case the server wants to provide content from more than one network (it would need to join all of them). A cleaner solution (and the one we follow in this paper) is to forward the query from the server to the P2P network by means of proxies and retrieve the answer with the same mechanism.

2. **An external provider wants to offer content to the P2P network.** We assume that providers that have already implemented a standard interface will not be happy spending more time and money in developing the specific interface(s) of the network(s) they want to join. In contrary, they would like to reuse the one they have which would also ease its administration (as only one interface needs to be maintained).

According to these two scenarios, there are two different types of proxies with different functionality. The former scenario requires the so-called "consumer proxy" and the latter the so-called "provider proxy" (names are assigned according to the role they play). A consumer proxy acts as a mediator between an external client that wants to query the network and the P2P network itself. A provider proxy acts as a mediator in order to provide the content of an external provider into the P2P network.

4.1 Consumer Proxy

As described above in scenario 1, in some cases it is needed to be able to query a P2P network without the need of joining it. A consumer proxy is a peer which is part of the P2P network (and therefore it is able to send queries to and receive the answers from it) and which is also able to receive requests and send responses using a different protocol and interface. This way, an external client is able to query the P2P network through the proxy.

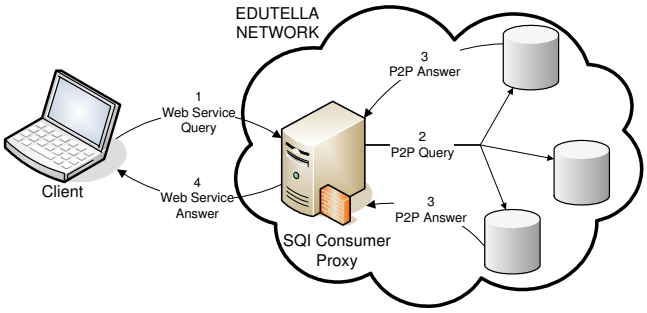

Fig. 1. Consumer Proxy

In our implementation, a consumer proxy mediates between the Edutella/JXTA and the SQI protocol. As depicted in figure 1, it is responsible for

1. Receiving queries from external clients via SQI
2. Forwarding the query to the Edutella network using the JXTA/Edutella interface

3. Collecting the results sent from peers within the network using the JXTA/Edutella interface
4. Forwarding those results to the requester system via SQI

This simple mechanism allows any system to query the content of the Edutella P2P network without needing to implement its specific interface. In addition, the proxy can return the results to the client application

Asynchronously. The results are sent to the client as soon as they arrive to the proxy. This is the typical mechanism in distributed environments as not all the results are generated at once but they must be gathered from the different systems in the network.

Synchronously. The results are gathered at the proxy and sent in a single message to the client. Although this is not the intuitive way for a distributed environment it could be desirable in some scenarios (e.g., in mobile devices we do not want our device to receive a new message every time a new result arrives to the proxy but better ask for new results in a proactive manner).

4.2 Provider Proxy

In order to fulfill our scenario 2, a second type of proxy has been developed. This provider proxy is a peer connected to the P2P network which also is able to send requests and receive responses by means of a different protocol and interface. Therefore, it is able to forward queries to external providers and receive their answers providing their content to the network.

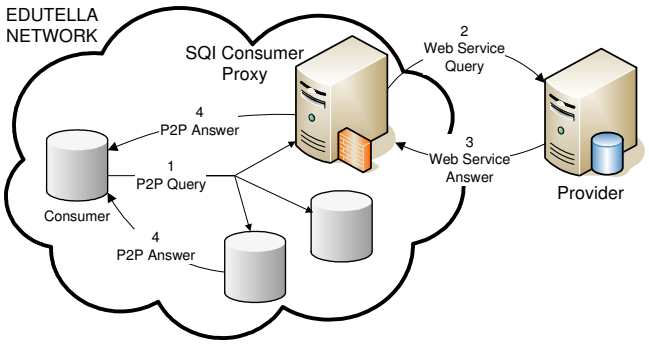

Fig. 2. Provider Proxy

As in the the case of consumer proxies, our provider proxy mediates between the Edutella/JXTA and the SQI protocol. As depicted in figure 2, it is responsible for

1. Receiving queries from peers in the network using the JXTA/Edutella interface
2. Forwarding them to the external provider via SQI
3. Receiving the results from the external provider via SQI

4. Sending them back to the peer that sent the query using the JXTA/Edutella interface

Due to the asynchronous nature of a P2P network, it is possible for the provider proxy to receive the results from the external provider in a synchronous (e.g., in case the external provider is a relational database) or asynchronous (e.g., if the external provider is another distributed environment) way.

5 Semantic Mappings

In previous sections, we have described some of the basics for interoperability, namely common protocol and interfaces (or the use of proxies as presented in previous section) and common query language (or the use of appropriate wrappers). Although these elements ensure that two systems are able to talk to each other they do not guarantee that they will be able to understand each other due to the possible use of different schemas/ontologies.

Nowadays, there is a big effort on standardization of domain ontologies. For example, Dublin Core [4] is intended as standard for cross-domain information resource description and LOM [1] describes attributes required to fully/adequately describe a Learning Object. Unfortunately, still many proprietary schemas are used in each domain (e.g., database schemas within companies). For example, Dublin Core suggests using the attribute "creator" to describe the responsible person of making or writing a resource. While many repositories probably follow this suggestion when annotating their resources, others might use e.g. their own attribute "author" instead. In order to bring interoperability among them, data integration in the form of semantic mappings is needed. In this context, a semantic mapping is a transformation from one data model to another data model according to a set of rules (mappings).

In a distributed network we can distinguish among several integration possibilities:

- If no virtual and unified schema is assumed in the network, systems within the network must provide pairs of mappings between each two systems. Subsequently, the distributed network can be seen as a directed graph in which each arrow represents an available mapping from one node to another. After that, they can be applied transitively in order to infer new mappings which were not explicitly defined. This is specially useful in P2P networks as it is usually not possible to enforce a unique and common schema. Authors in [2, 8] study this approach and provide algorithms to estimate the correctness of the inferred mappings.
- If a virtual and unified schema is assumed, there are two approaches for providing integration between the global schema and local schemas at the sources:
  - **Global As View (GAV)** [9]. In this approach, the global schema is expressed in terms of the data sources (an example is depicted in figure 3).
  - **Local As View (LAV)** [26]. In this approach, each source is defined as a view over the global schema. This way, the global schema is specified independently from the sources (an example is depicted in figure 4).

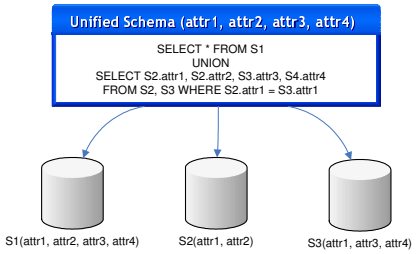

Fig. 3. Global As View Approach

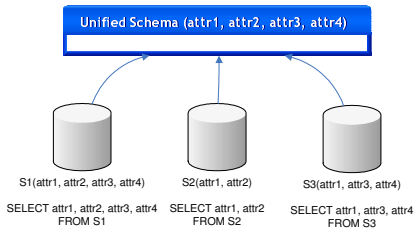

Fig. 4. Local As View Approach

A discussion of both GAV and LAV is provided in [11] as well as an introduction to "query rewriting" mechanisms. Query rewriting is the process in which a query expressed in the global schema is reformulated into another query according to a set of mappings [23]. This is the mechanism we have used in the mappings module we describe subsequently.

5.1 Query Rewriting Module

In order to provide semantic interoperability in our network, we have developed a module which transforms a query q_1 into a query q_2 according to the set of mappings specified. This module is intended to work on pairs of mappings without a unified schema, or in GAV or LAV integration approaches.

QEL, the language we use in our network, is based on datalog. In addition to standard datalog constructs, QEL includes some built-in predicates. Taking into account that in our network only metadata (in RDF) is queried and exchanged, the most important one is

$$qel : s(Subject, Predicate, Object)$$

which according to the QEL specification [15] "is true if Subject and Predicate are anonymous or non-anonymous RDF resources, and Object is a non-anonymous or anonymous RDF resource or an RDF Literal and the triple Resource Predicate Object exists in the RDF data". For example, a query like

$$?- qel : s(X, dc : title, 'Artificial Intelligence').$$

will return all the resources which title is "Artificial Intelligence". Other useful built-in predicates are *qel:like(X,Y)* ("used to determine whether an RDF literal or URI contains a string as a substring"), *qel:lessThan(X,Y)* and *qel:greaterThan(X,Y)* which are used to compare two RDF literals.

Given this short introduction to the language, let us present the following simple query that we will use for our examples in the rest of the section:

$$@prefix\ qel : < http://www.edutella.org/qel\# > .$$
$$@prefix\ dc : < http://purl.org/dc/elements/1.1/ > .$$
$$@prefix\ lom : < http://ltsc.ieee.org/2002/09/lom-rights\# > .$$
$$?- qel : s(X, dc : title, Title),$$
$$\quad qel : s(X, dc : description, Description),$$
$$\quad qel : s(X, dc : creator, Creator),$$
$$\quad qel : s(X, lom : cost, Cost),$$
$$\quad qel : s(X, dc : subject, Subject).$$

This query retrieves all the resources with title, description, creator and subject attributes from Dublin Core and the cost from LOM. The first lines of the query with prefix "@" define the namespaces.

Given such a query, we identified the following requirements:

- The query specifies a property (in the paper we will use property and attribute indistinctly) that does not exist in the source but the source has an equivalent property which could be used instead of. For example, if one data source has its own schema where it uses the property "abstract" instead of the property "description" from the Dublin Core standard.
- The query specifies a property and one value according to a specific taxonomy and the source uses a different taxonomy. For example, if the query searches for resources with "dc:subject" following the ACM classification[3] and the data source does have "dc:subject" but it follows the Dutch Basic Classification [5].
- In general, if one of the attributes is not available at the data source, the whole query fails[2]. However, it might happen that although the source does not have explicitly such an attribute, all its resources would share the same value if it existed. For example, assume a repository where all the resources are offered for free. This repository does not have the property "lom:cost" because it is not needed. However, in case one query contains this attribute, the whole query would fail (even if the constraint in the query is "lom:cost = No" which is actually true though it is not annotated). In such a case, it is desirable to assign a default value to all the resources in the data source without having to explicitly annotate all the resources of the repository.

In order to satisfy these requirements we developed a module that performs two types of mappings and one extra transformation: property mapping, property-value mapping and default value transformation (see table 1 for the whole list of mappings and [16] for technical details).

5.2 Property Mapping

A property mapping specifies how one property in the query must be reformulated. When the mapping module receives a query that contains the triple $qel : s(X, p_1, Z)$ it rewrites it into $qel : s(X, p_2, Z)$. Using our example query and taken into account the requirement in which the source does not contain the property "dc:description" but "own:abstract" (where "own" stands for their local namespace), it is possible to define the following mapping:

$$(X, dc : description, Z) \leftarrow (X, own : abstract, Z)$$

This mapping is currently a 1-to-1 mapping, that is, there is only one triple at each side of the mapping (separated by the left arrow) but it is also possible to specify 1-to-2, 2-to-1 and 2-to-2 mappings (see table 1). For example, suppose the author in the source is encoded using the property full name from the vcard ontology [20]. In such a case, we need the following mapping

[2] Here we assume that only conjunctives queries are sent. Edutella and QEL support disjunctive queries but we will omit them here because of simplicity

Mapping type	Description
1-to-1 property mapping	$(R, p_1, O) \leftarrow (R, p_2, O)$
1-to-1 property-value-value mapping	$(R, p_1, v_1) \leftarrow (R, p_2, v_2)$
2-to-1 property mapping	$(R, p_1, O), (O, p_2, L) \leftarrow (R, p_3, L)$
2-to-1 property-value mapping	$(R, p_1, O), (O, p_2, v_1) \leftarrow (R, p_3, v_2)$
1-to-2 property mapping	$(R, p_1, L) \leftarrow (R, p_2, O), (O, p_3, L)$
1-to-2 property-value mapping	$(R, p_1, v_1) \leftarrow (R, p_2, O), (O, p_3, v_2)$
2-to-2 property mapping	$(R, p_1, O), (O, p_2, L) \leftarrow (R, p_3, O), (O, p_4, L)$
2-to-2 property-value mapping	$(R, p_1, O), (O, p_2, v_1) \leftarrow (R, p_3, O), (O, p_4, v_2)$
Default value	$(p \leftarrow v)$

Table 1. Types of Mappings

$$(X, dc:creator, Z) \leftarrow (X, dc:creator, Y), (Y, vcard:fn, Z)$$

in order to abstract from the internal representation at the source[3]

5.3 Property-Value Mapping

The mapping described above assumes that one property is completely mapped onto another one. However, mapping can be brought to the granularity of values. A property-value mapping applies only when a query contains not only a specific property but also a specific value for that property and then both of them map into other (possibly the same) property and value. For example, assume that our example query uses the ACM classification in the property "dc:subject" and our source does have the property "dc:subject" but annotated with the Dutch Basic Classification taxonomy. We could use several mappings of the form

$$(X, dc:subject,' Software/Programming\_Languages') \leftarrow (X, dc:subject,' Computer\_Science/Programming\_Languages')$$

to specify how the different values from the ACM taxonomy map into the Dutch Basic Classification.

In the same way as the property mapping, it is possible to extend this 1-to-1 to 2-to-1, 1-to-2 and 2-to-2 mappings.

5.4 Default Value

Property and property-value mappings provide rules which define how source triples are reformulated into equivalent triples corresponding to the destination schema. The "default value" mapping works differently. The properties specified in default values do not exist in the source repository and therefore they must be removed (not just reformulated) in the new query. To our knowledge, default values have not been defined before in literature and therefore, we formalize the process in Appendix A.

Following this approach, when a query is received by our mapping module, if there exists in the query any occurrence of a property specified in the default

[3] Note that we use similar notation as in [26]. Therefore, the right side of the mapping rule is rewritten into the left side.

values, this occurrence is temporarily removed. This way, the query is sent to the local repository without that property (otherwise the query would fail) and a resultset is returned. However, this resultset still does not contain the default values that were requested (the properties previously removed) and therefore they must be added. Therefore, default values are added to each of the rows in the resultset returned by the repository. For example, following with our example query, suppose that our source repository does not have the property "lom:cost" but all the resources in the repository are free of charge. We can then define the following default value

$$(lom : cost \leftarrow' No')$$

This way, any triple in the query referring to the property "lom:cost" would be removed before the query is sent to the repository and added subsequently to the returned resultset together with the default value "No". In contrary, the query may specify that only elements which are not free of charge should be returned. In such a case, since it does not match the default value, the query is not executed and an empty resultset is returned.

6 Generating Mappings using an Editor

Creating the appropriate mappings for the query rewriting module can be a tedious task, especially when large schemas are involved. Yet another problem is that the rewriting engines are embedded in web services or applications, and therefore have to retrieve the list of mappings from a file or web resource.

In order to help the user to create these files, we have developed an application that can import RDF schemas, displays their properties and allows the user to create mappings by simply clicking on the available elements. Once created, the mappings can be stored on disk in different formats, to be copied into the configuration of the query rewriting engine. Alternatively, the editor can be used to load existing mapping definitions and allow their modification before writing them back to the storage media.

We developed the application as a plugin for the Rich Client Platform [21], using the Jena RDF framework for managing RDF data. The software itself is open source, and is available via CVS from the main Edutella repository[4] (as the rest of the components described in this paper). The component based approach of the RCP platform makes the editor easy extendable for additional functionality and allows the integration into other RCP tools, like the Eclipse Java Development Platform. The two main parts of the application are the Mapping Data Model and the User Interface.

6.1 Data Model

In the Editor application the mappings are represented as java objects (see figure 5). The Jena RDF API is providing access to the Properties of the involved schemas. The Mappings data Model consists of the classes representing the various forms of possible mappings, as well as an API to create new Mappings (Factory) and to store or load mappings (Parser).

[4] http://edutella.jxta.org/

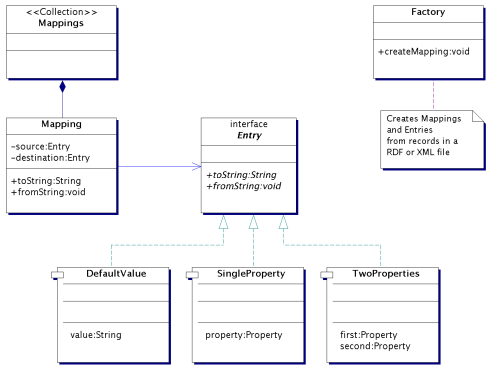

Fig. 5. Classes in the Editor Mapping Data Model

6.2 User Interface

The interface for the user was implemented as an Eclipse plugin, using the Eclipse Rich Client Platform. Therefore, it is easy to integrate the editor into other RCP applications, or to implement additional functionality for the mapping editor, e.g. assistants to support the mappings creation process by doing simple similarity comparisons based on the schema annotation.

After the application startup, a window asks for the location of a RDF schema to use as the source schema (see figure 6). A Schema source can be chosen from a list of common used schemas, or by supplying a URL pointing to the location of the RDF document defining the schema.

Once a schema is selected, the column on the left displays the source schema while the middle column shows the target schema. Figure 7 shows the application where the Dublin Core has been loaded as source schema (left side of the window) and the Friend-of-a-Friend schema [7] as target schema, the right side of mappings (in the middle column). Additionally, default values can be added to the second panel on the left side, below the source schema.

Creating a Mapping is now an easy task. In the lower part of the window are fields for the supported types of mappings. These fields can be filled by selecting elements from the properties listed above them. Then, the mapping is created just pressing on the "Add" button. The new mapping is added to the list of mappings in the right panel. Finally, selecting the appropriate operation from the menu, the created mappings can stored or new schemas loaded.

In order to cover most of the usage scenarios in the early states of development, we first implemented in our editor 1-to-1 and 2-to-1 property mappings and default values. We plan to extend it to the full set of mappings. In addition, techniques for (semi)automatic generation of mappings as described in [19] may be implemented.

7 Related Work

In [18], the authors describe the two scenarios, consumer proxy and provider proxy, and implemented a translation from the JXTA protocols to Web Ser-

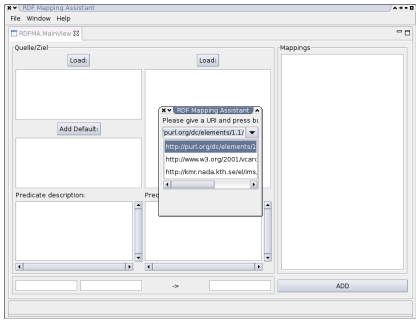

Fig. 6. Choosing a RDF Schema

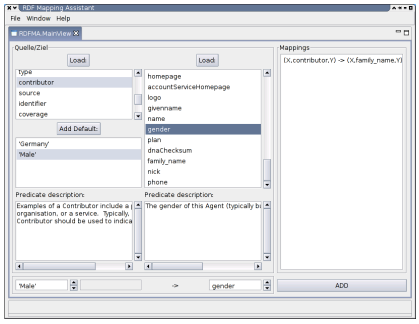

Fig. 7. Creating Mappings

vices and vice versa. In this paper, we enrich our proxies with the possibility of mapping query languages and schemas. In addition, [18] does not use any specific interface but wrap the java objects by means of serialization while in our approach we are using the SQI standard initiative.

[10] presents an interesting approach for interoperability of Learning Repositories. Authors briefly present an "ECL Gateway" which is similar to our idea of proxies. They implemented a translation between ECL and a P2P protocol. In our paper we extend this idea separating the two different scenarios, consuming and providing information, and describing in detail how proxies work and how mappings can be performed. Furthermore, we present our work on default values which, to our knowledge, had not been described yet in any paper. Lionshare [24] includes the possibility to perform federated searches in each of the peers. However, the integration is made at the peer level (users may choose whether to query the network or other federated repositories) thus not providing an interoperability solution at the network level.

There exists a large number of papers on ontology mapping, specially on the creation of such mappings in a (semi)automatic way [19] although due to space limitations we have only cited the most relevant ones. Future versions of our editor may include some of the existing algorithms to help administrators.

8 Conclusions and Further Work

In this paper, we showed by means of proxies and semantic mappings how it is possible to connect a P2P network like Edutella with other systems outside the network. These proxies provide the necessary mediation between the different protocols and interfaces and semantic mappings overcome the problem of schema heterogeneity. Both together allow external systems to query and provide content in the network avoiding the isolation of P2P networks from the rest of the world. We refer the reader to [17] for a detailed description of successfully interconnected systems using our components, like for example ARIADNE and the Media Library (Swedish Educational Radio and Television).

In this paper we have focused the interoperability problem on search. However, although this is of course the most important service, there are still some other issues that must be researched. One of the main topics we plan to research on in the future is distributed ranking algorithms. Currently, a lot of research

has been done around web ranking and merging of ranking lists (e.g., on meta-search engines). However, the former assumes that relationships of the form of links exist among resources in different repositories and the latter assumes that there exist overlapping in the content different repositories offer and rank. Unfortunately, this does not apply in a P2P network and the only existing measures are based on trust/reputation of the peers.

In addition, a challenge for the Local As View mappings described in this paper is how they would work in combination with Edutella Retrieval. Edutella Retrieval [6] is a recent addition in Edutella which allows information to be retrieved without requiring it explicitly in the query which would unnecessary eliminate valid matches. Since what is retrieved is not explicitly stated in the query it is difficult to detect which mappings to apply.

References

1. 1484.12.1 IEEE standard for learning object metadata. june 2002. http://ltsc.ieee.org/wg12.
2. Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. The chatty web: emergent semantics through gossiping. In *International World Wide Web Conferences*, Budapest, Hungary, may 2003.
3. ACM classification. http://www.acm.org/class/1998/overview.html.
4. Dublin core metadata innitiative (DCMI). http://dublincore.org/.
5. Dutch basic classification codes. http://www.kb.nl/vak/basis/bc98-en.html.
6. ERS: edutella retrieval service. http://edutella.jxta.org/spec/retrieval.html.
7. FOAF the friend of a friend (foaf) project. http://www.foaf-project.org/.
8. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE*, 2003.
9. Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal: Very Large Data Bases*, 10(4):270–294, 2001.
10. Marek Hatala, Griff Richards, Timmy Eap, and Jordan Willms. The interoperability of learning object repositories and services: Standards, implementations and lessons learned. In *13th World Wide Web Conference (WWW'04)*, New York, USA, May 2004.
11. Maurizio Lenzerini. Data integration: A theoretical perspective. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 233–246, Wisconsin, USA, jun 2002.
12. Jehad Najjar, Erik Duval, Stefaan Ternier, and Filip Neven. Towards interoperable learning object repositories: The ariadne experience. In *IADIS International Conference WWW/Internet*, Algarve, Portugal, nov 2003.
13. W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *12th International World Wide Web Conference (WWW'03)*, Budapest, Hungary, may 2003.
14. Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmer, and Tore Risch. Edutella: A P2P networking infrastructure based on RDF. In *11th International World Wide Web Conference (WWW'02)*, Hawaii, USA, jun 2002.
15. M. Nilsson and W. Siberski. RDF query exchange language (QEL) - concepts, semantics and RDF syntax, http://edutella.jxta.org/spec/qel.html, 2003.

16. Daniel Olmedilla. Working with edutella. technical report. http://www.l3s.de/ olmedilla/projects/edutella/edutella.pdf.
17. Daniel Olmedilla and Matthias Palmér. Interoperability for peer-to-peer networks: Opening p2p to the rest of the world. In *WWW Workshop on Interoperability of Web-Based Educational Systems*, volume 143 of *CEUR Workshop Proceedings*, Chiba, Japan, may 2005. Technical University of Aachen (RWTH).
18. C. Qu and W. Nejdl. Interacting edutella/JXTA peer-to-peer network with web services. In *2004 International Symposium on Applications and the Internet (SAINT 2004)*, Tokyo, Japan, jan 2004. IEEE Computer Society Press.
19. Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
20. Representing vCard objects in RDF/XML. http://www.w3.org/tr/vcard-rdf.
21. RCP rich client platform. http://www.eclipse.org/rcp/.
22. SQI: simple query interface. http://www.prolearn-project.org/lori/.
23. I. Tatarinov and A. Halevy. Efficient query reformulation in peer-data management systems. In *SIGMOD, 2004.*, 2004.
24. The lionshare project. http://lionshare.its.psu.edu/.
25. The open knowledge initiative (oki). http://www.okiproject.org/.
26. Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

A Default Value Formalization

Let T be the set of all triples of the form (R, P, O) such that R, P and O are resource, predicate and object respectively. In addition, let D be the set of default values such that $d = (P, V)$ where P is a property and V a literal.

Then, for all queries Q, define $Q \xrightarrow{D}_1 P$ iff

- $Q = (T_1, \ldots, T_{i-1}, T_i, T_{i+1}, \ldots, T_n)$
- $T_i = (R, p, O)$
- $U = \begin{cases} T_i & \text{if } \exists p, d | T_i = (R, p, O), d \in D, d = (p, V), \\ \emptyset & \text{otherwise.} \end{cases}$
- $P = Q \setminus U$

Finally, we denote with $\xrightarrow{D}$ the reflexive transitive closure of $\xrightarrow{D}_1$ and the unique result of rewriting of the query with default values by $Q_2 = removeDV(Q, D)$.

After this process, the resulting query Q_2 of this process is sent to the repository and a resultset S is received as an answer to the query.

Then, let U be the set of default values applied in the previous process and for all rows R in S, define $R \xrightarrow{U}_1 W$ iff

- $R = (V_1, \ldots, V_n)$
- $V_{n+1} | d = (P, V_{n+1}), d \in U$
- $W = R \cup V_{n+1}$

and finally we denote with $\xrightarrow{U}$ the reflexive transitive closure of $\xrightarrow{U}_1$ and the unique result of this operation as by $S_2 = addDV(S, U)$ where S_2 is the final resultset returned to the query.