A Simple Query Interface for Interoperable Learning Repositories

Bernd Simon*), David Massart±), Frans van Assche±), Stefaan Ternier+), Erik Duval+), Stefan Brantner#), Daniel Olmedilla†), Zoltán Miklós*)

*) Vienna University of Economics and Business Administration, Austria, {bsimon, zmiklos}@wu-wien.ac.at
±) European Schoolnet, Belgium, {david.massart, frans.van.assche}@eun.org
+) Katholieke Universiteit Leuven, Belgium, {erik.duval, stefaan.ternier}@cs.kuleuven.ac.be
#) BearingPoint Infonova GmbH, Austria, stefan.brantner@bearingpoint.com
†) Learning Lab Lower Saxony (L3S), Germany, olmedilla@l3s.de

ABSTRACT

In order to achieve interoperability among learning repositories, implementers require a common communication framework for querying. This paper proposes a set of methods referred to as Simple Query Interface (SQI) as a universal interoperability layer for educational networks. The methods proposed can be used by a source for configuring and submitting queries to a target system and retrieving results from it. The SQI interface can be implemented in a synchronous or an asynchronous manner. SQI abstracts from query languages and metadata schemas. SQI has been evaluated by several prototype implementations demonstrating its universal applicability, and is on the way to being standardized in the CEN/ISSS Learning Technologies Workshop. The latest developments of SQI can be followed at http://www.prolearn-project.org/lori/.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval - Search process, H3.7 [Information Storage and Retrieval]: Digital Libraries - Systems issues, H.3.5 [Information Storage and Retrieval]: Online Information Services - Web-based services

General Terms

Management, Design, Standardization, Languages

Keywords

Interoperability, Application Program Interface, Learning Repositories, Querying, Web Services

1. INTRODUCTION

The Web puts a huge number of learning resources within reach of anyone with Internet access. However, many valuable resources are difficult to find in an efficient manner, because valuable resources are hidden in the closed and proprietary worlds of learning (content) management systems, streaming media servers and online collaboration tools.

Such systems are commonly referred to as learning object repositories being part of an educational web. Learning object repositories hold information on learning objects (i.e., metadata), in order to describe educational artefacts such as courses, online tutorials, lecture notes, electronic textbooks, tutoring sessions, quizzes, etc.

In this paper we propose a common query interface as one part of the solution for exploring the hidden educational web. The notion 'hidden web' refers to the web, which is hidden behind proprietary search interfaces and authentication mechanisms [14]. This proprietary world of interfaces, leads to a lack of interoperability. Interoperability can be defined as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [9]. To a user, the lack of interoperability, for example, means [16]:

- Applications and their data are isolated
- Redundant data entry is required.

In order to achieve interoperability on the educational web a common semantic model is required. The semantic model – also referred to as 'ontology' – should specify the properties of the learning resources accessible within the repository [20]. Each declaration of a learning resource property constitutes an ontological commitment to use the defined term in interactions with the repository.

Additionally, interoperable learning object repositories are based on common protocols, which define the interactions between repositories.

To achieve interoperability, different kinds of protocols can be used. The Learning Object Repository Interoperability Framework presented in Figure 1 distinguishes between core services and application services. Core services are needed, for example, to agree on a common procedure for uniquely identifying learning objects. Other core services are related with authenticating users and repositories, or with creating and managing sessions for interaction between applications.

Typical applications that make learning repositories interoperable are, for example, the indexing service, the harvesting service or the query service. The indexing service, as a kind of replication service, allows repository A to "push" learning object metadata to repository B. It supports distributed maintenance of metadata through insert, delete or update operations. The harvesting service is a service, where repository A "pulls" metadata from a repository B. The query service allows repository A to search repository B for suitable learning resources, so the metadata transferred matches a specific query. A contracting service assigns access rights to a learning object stored at a remote repository. The delivery service interacts with the repository where the learning resource is stored and delivers an electronic learning resource to the end user.

Application services make use of core services. For example, the core service session management might be required for the query service.

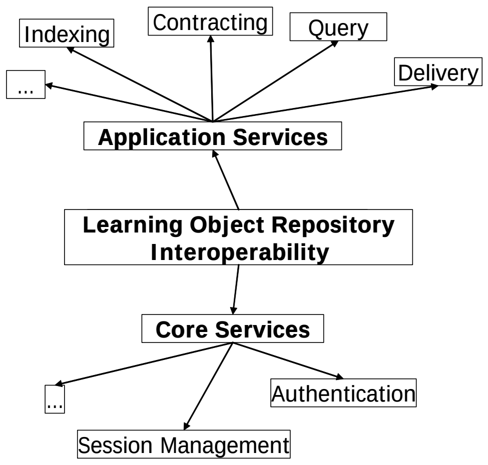
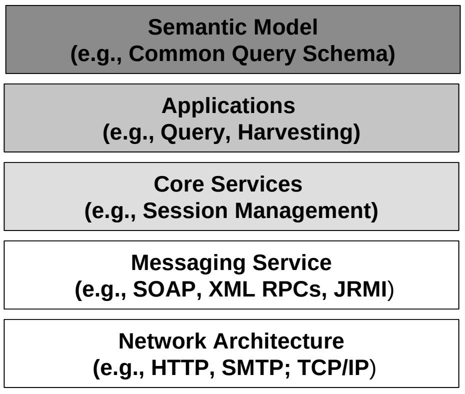


Figure 1. Interoperability Framework

Both, core and application services, require a common messaging infrastructure, which enables repositories to interact. XML RPC, Java RMI, and WSDL/SOAP are examples of such messaging services. A messaging service is based on a common network infrastructure and lower level protocols such as TCP/IP, HTTP, etc. Figure 2 depicts the various layers of Learning Object Repository Interoperability as described above.



Semantic Model (e.g., Common Query Schema)
Applications (e.g., Query, Harvesting)
Core Services (e.g., Session Management)
Messaging Service (e.g., SOAP, XML RPCs, JRMI)
Network Architecture (e.g., HTTP, SMTP; TCP/IP)

Figure 2. Interoperability Stack

Learning Management Systems (LMS), Learning Content Management Systems (LCMS), Knowledge Pools, or Brokerage Platforms are the kind of information technology the interfaces proposed herein are designed for. Within its focus on the query interface, this paper targets architects of educational networks, managers of learning resource repositories, stakeholders in learning object re-use, as well as researches in web services and system interoperability. However, although we refer to learning repositories interoperability with a special focus on learning object metadata, this query interface can also be used within other domains and application scenarios [12].

The remainder of this paper is structured as follows: While Section 2 is devoted to specification of the Query Service including Authentication and Session Management, Section 3 presents implementation of the API. Section 4 reviews related work. The paper concludes with a discussion of the status quo and outlines future work on the interface specification.

2. Simple Query Interface

An Application Program Interface (API) for query services needs to specify a number of methods a repository can make available in order to receive and answer queries from other applications.

To distinguish the requestor from the answering system in our scenarios, the term “source” is introduced in order to label a system which issues a search (the source of the query). The term “target” labels the system which is queried (the target of the query). Alternatively, the “source” can also be referred to as “requestor” and “target” as “provider”.

Metadata can be stored using different means, such as file-based repositories, (distributed) relational databases, XML repositories, or RDF tool kits, which use different query languages constituting a heterogeneous environment. In order to make learning repositories interoperable, not only a common interface needs to be defined, but also a common query language together with a common results format for learning object descriptions needs to be agreed on. Interoperability aspects such as common query schema, results format are part of the semantic model of an educational network (see Figure 2). This research focuses on the transport mechanisms required for querying, issues related to the semantic model are not within the scope of this paper.

The query service is used to send a query in the common query language to the target. Next, the query results, represented in the common results format, are transported to the source. On the implementation level, wrappers may need to be built to convert a query from a common query language X to a local query language Y and transform the query and the query results from a proprietary format to a common one and vice-versa.

Figure 3 illustrates an exchange process, where Learning Repository A (the source) submits a query to Learning Repository B (the target). It is assumed that both systems have agreed upon a common query language beforehand. The concepts used in the query statement are part of a common (query) schema. At Repository B, the interface component might need to transfer the query from the common query language to the local one. Also some mappings from the common to the proprietary schema might be required before submitting the search. This task is performed by a wrapper component. Once the search has yielded results, the results set is forwarded to the source, formatted according to a common results format.

The collaborative effort of combining highly heterogeneous repositories has led to the following requirements:

- The API needs to be neutral in terms of results format, query schema and query language: The repositories connecting can be of highly heterogeneous nature: therefore, no assumptions about these components of interoperability stack can be made.
- The API needs to support synchronous and asyn-chronous queries in order to allow the application of the API in heterogeneous use cases.
- The API needs to support, both, a stateful and a stateless implementation.
- The API shall be based on a session management concept in order to separate authentication issues from query management.

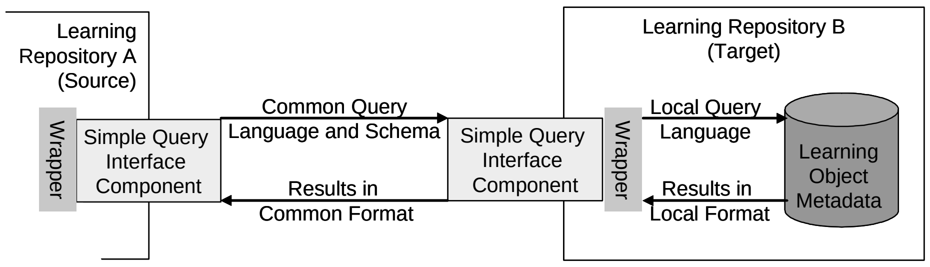

Learning Repository A (Source)
Wrapper
Simple Query Interface Component
Common Query Language and Schema
Results in Common Format
Learning Repository B (Target)
Simple Query Interface Component
Wrapper
Local Query Language
Results in Local Format
Learning Object Metadata

Figure 3. Communication between two Repositories

In addition, the design of the API itself is based on the following design principles:

- Command-Query Separation Principle,
- Simple Command Set and Extensibility.

Since one design objective of the API is to keep the specification simple and easy to implement, the API is labeled Simple Query Interface (SQI).

The following sub-sections describe each of the above mentioned design principles in more detail.

2.1 Query Language and Results Format

In order to make use of SQI to implement full query functionality, the API needs to be complemented with agreements about:

- the set of attributes and vocabularies that can be used in the query,
- the query language and its representation,
- the representation of list of learning objects that satisfy the query, and
- the representation of individual metadata instances on learning objects.

SQI is agnostic on these issues: Any agreement between two or more repositories is valid for SQI. Such agreements can, for example, be expressed by XML schemas or RDF schemas.

Although SQI does not directly contribute to overcome the differences of the various paradigms in metadata management (Z39.50, XML-based approaches, RDF community), it aims to become an independent specification for all open educational repositories.

2.2 Synchronous and Asynchronous Queries

SQI can be deployed in two different scenarios: In the synchronous scenario, the target returns the query results to the source. Results retrieval is therefore initiated by the source through the submission of the query and through other methods allowing the source to access the query results.

In the asynchronous scenario, results transmission is target-initiated. Whenever a significant amount of matching results is found, these results are forwarded to the source by the target. To support this communication the source must implement a results listener. The source must be able to uniquely identify a query sent to a particular target (even if the same query is sent to multiple targets). Otherwise the source is not able to distinguish the search results retrieved from various targets and/or queries previously submitted to a target.

Please note that the asynchronous query mode does not require an asynchronous handling on the messaging layer. It can also be implemented by two synchronous functions at the source and the target, respectively.

A query interface operated in synchronous mode can perform multiple queries per session (even simultaneously). In case of an asynchronously operated query interface, the source provides a query ID that allows it to link incoming results to a submitted query (the source might query many targets and each target might answer to a query by returning more than one result to the source). Multiple queries can also be active within a session in asynchronous query mode.

2.3 Session Management

The application interfaces make abstraction from authentication and access control issues. However, there is a need to authenticate the source in order to allow a target, for example, to link query policies to a source repository. For instance:

- Repository A is allowed to query Repository B without any limitations,
- Repository C is only allowed to retrieve 1000 query results per day from Repository D at a maximum.

Ideally, authentication is performed only once for a series of interactions. To accomplish this, a session token needs to be returned after successful authentication that can be used to identify the system in the subsequent communication.

Session management needs to be understood as a higher-layer management of configuration settings and authentication. The session ID serves as a mandatory element in the application interfaces in order to identify the requestor/source in all query commands.

Therefore, the SQI is based on a simple session management concept. A session has to be established before any further communication can take place. This specification separates query management and processing from authentication (and query policy management).

In case of a synchronously operated query interface, the source establishes a session at the target and uses the Session ID, which it obtained from the target, to identify itself during communication. Authentication does not need to be based on credentials, since also anonymous sessions can be created.

The specification introduces an incomplete list of possible means for establishing a session for the communication between two systems.

Once a session has been established, the source has the right to communicate with the target. In order to establish a session, a user name and password or any other credential may be required. The identification of a source repository can prevent candidate target repositories from opening up their systems to unknown partners, and enables query policies.

A session is valid until it is destroyed. Hence, it continues to be active after a query has been executed. Alternatively, a session times out when no communication takes place during e.g. 30 minutes. However, a session might be valid much longer than 30 minutes and sometimes might even require manual destruction.

The specification assumes the use of secure authentication, authorization, and encryption mechanisms such as those provided by state-of-the-art technology (e.g., SSL).

2.4 Stateful and Stateless Communication

Stateful and stateless are attributes that describe whether repositories are designed to keep track of one or more preceding events in a given sequence of interactions. Stateful means that the target repository keeps track of the state of interaction, for example, by storing the results of a previously submitted query in a cache. Stateless means that there is no record of previous interactions and that each interaction request can only be handled on the basis of the information that comes with it. The SQI specification allows implementers to opt for a stateful or a stateless approach.

2.5 Command-Query Separation Principle

SQI design is based on the "Command-Query Separation Principle". This principle states that every method should either be a command that performs an action, or a query that returns data to the caller, but not both. More formally, methods should return a value only if they are referentially transparent and hence cause no side-effects. This leads to a style of design that produces clearer and more understandable interfaces.

The Command-Query Separation (CQS) is a principle of object-oriented computer programming. It was devised by Bertrand Meyer a part of his work on the Eiffel programming language [21].

2.6 Simple Command Set and Extensibility

In order to make the interface easily extensible an approach, minimizing the number of parameters of the various methods rather than the number of methods is adopted. Variations of the interface (e.g., a separation between common query schema and common results format), can easily be introduced by adding a new function (e.g., setSupportedQuerySchema) while no change in the already implemented methods is needed. Hereby, backwards compatibility can be more easily maintained.

As a result, additional methods for setting query parameters like maximum duration and maximum number of returned search results were introduced. This design choice leads to simpler methods, but the number of interdependent methods is higher. However, default values can be used for many of these query parameter configuration methods.

2.7 Overview of SQI Methods

Table 1 provides an overview of the various methods that are described below from a workflow perspective. A detailed description of the methods is provided in the specification [15].

First, the source needs to create a connection with the target, for example by using createAnonymousSession. Once a session has been established, the query interface at the target awaits the submission of a search request. In addition, a number of methods allow for the configuration of the interface at the target. Query parameters such as

- the query language (setQueryLanguage),
- the number of results returned within one results set (setResultsSetSize),
- the maximum number of query results (setMaxQueryResults),
- the maximum duration of query execution (setMaxDuration),
- and the results format (setResultsFormat)

can be set with the respective methods. The parameters set via these methods remain valid throughout the whole session or until they are set otherwise. If none of the methods is used before the first query is submitted, default values are assumed. The specification provides default values for MaxQueryResults and MaxDuration, and ResultsSetSize.

Session Management
createSession
createAnonymousSession
destroySession
Query Parameter Configuration
setResultsFormat
setMaxQueryResults
setMaxDuration
Synchronous Query Interface
setResultsSetSize
synchronousQuery
getTotalResultsCount
Asynchronous Query Interface
asynchronousQuery
setSourceLocation
queryResultsListener

Table 1. SQI Methods

Next, the source submits a query, using either the asynchronousQuery or the synchronousQuery method. The query is then processed by the target and produces a set of records, referred to as results set. The query is expressed in a query language identified through a query parameter. In the query, reference to a common schema might be made. In synchronous mode the query results are directly returned by the synchronousQuery method. The getTotalResultsCount method returns the total number for matching metadata records found by the target operating. In case of an asynchronously operated query interface the queryResultsListener method is called by the target to forward the query results to the source.

In order to report abnormal situations (e.g., erroneous parameters or inability to carry out an operation), an SQIFault is provided, which can be thrown by all the SQI methods. A system of fault codes permits to document those abnormal situations.

3. Implementations

Since the first stable version of the specification was made available in March 2004 many learning repositories have taken advantage of SQI to connect them to the outside world. Under the auspices of the CEN/ISSS Learning Technologies Workshop the following projects took advantage of the SQI specification.

3.1 ARIADNE

The core of the ARIADNE Knowledge Pool System (KPS) [5] is a distributed network of Learning Object Repositories that replicate both (the publicly available subset of) content and metadata. On top of this core infrastructure, ARIADNE provides its members with a set of tools that are loosely coupled with the KPS [17]. Through these tools, the user community can transparently manage learning objects.

Currently, each node in this distributed network implements a relational metadata store, on top of which both a synchronous and an asynchronous SQI target are provided.

The synchronous target lowers the threshold for integrating a query API into a third party application that aims to provide access to the KPS. In this scenario, an application sends queries to one synchronous target and only downloads additional results when they are needed.

With the asynchronous target, interoperability with intermediary services is targeted. As these services usually distribute queries over a large number of repositories, asynchronous communication is more fault-tolerant. As all results are collected through one results listener, it is easier to manage and hence more convenient in this scenario.

In order to provide ARIADNE members access to other repositories, a federated search engine [18] has been developed. This engine offers a synchronous SQI interface to front-end applications. SILO, the ARIADNE search & indexation tool, uses this target e.g. to query a set of repositories. In the back-end, the federated search engine forwards the query to different SQI enabled repositories. Currently, searches are distributed into the following repositories: ARIADNE, EdNA Online, EducaNext, Merlot, Pond, RDN, SMETE, and VOCED. As all these repositories support different query languages, which do not always easily map into one another, we started with an approach where a least common denominator of all query languages was implemented. In this approach, the query only consists of search terms which are translated by each repository into a query it can process.

Currently, ARIADNE requires each partner to return a minimum of metadata fields, encoded as LOM XML: a URL, an identifier, a title and an identifier of the originating repository. The URL should resolve to the learning object. If access to the learning object is prohibited, the URL resolves to contact information. A repository identifier is necessary to give credits in a proper way to repository that yielded the results. Apart from the data elements mentioned above, all other LOM metadata fields are optional in the results.

3.2 CELEBRATE and ICLASS

The iClass adapter is a component of the Intelligent distributed Cognitive-based Learning System for Schools (iClass) [8]. It enables the end-users of "non-iClass" systems, such as the learning management systems and learning content management systems that are members of the Celebrate federation [19], to search and access iClass contents (i.e., metadata and learning objects). In iClass, metadata are stored in a peer-to-peer network of metadata repositories named "content server" and learning objects are stored in a peer-to-peer network of learning objects repositories named "content distribution system". Usually, obtaining a learning object is a two-step process:

1. Searching and evaluating metadata: Selecting a learning object that satisfies user needs on the basis of the description provided in the metadata;
2. Consuming the learning object: Getting the selected learning object at the location (usually a URL) provided in the metadata.

Using a standard and open interface is a strong requirement in order to enable as many learning systems as possible to search and access the iClass collections of learning objects. The simplicity of SQI, its ability to be used in combination with any query language and results format, and its asynchronous query mode make it a good candidate interface for searching the iClass content server.

In iClass, metadata provide an identifier of the learning object rather than its location. Actually, the adaptive and multimedia nature of the iClass learning objects combined with the peer-to-peer nature of the content distribution system makes it difficult to access learning objects directly. This is why an extra step is required to resolve the location of a learning object identified in the metadata. This "resolve-location" step is used to propagate a request for location from repository to repository until an instance of the requested learning object is found. The learning object is then moved to a streaming server close to the location of its requester and a URL from which the learning object can be consumed is returned by the content distribution system. Since this process has potentially a certain duration, the content distribution system answers to these requests asynchronously in order to ensure adequate performance of the caller, in this case the iClass adapter. It is only when a learning object is available at a given streaming server that its location is known and can be returned.

Since there does not exist an open interface for performing this step asynchronously and rather than create an ad hoc interface, it was decided to use SQI for this task as well by taking advantage of SQI independence in terms of query languages and results formats. This is achieved by adding a new "query language" (for requesting a location) and a new "results format" (for returning a location) to the list of languages and formats supported by the iClass adapter [12]. The new query language is named "`ICLASS-LO-ID`". A query in this ad hoc language consists of the requested learning object's identifier as found in the metadata. The results format is named "`URL`". A result in this format consists of a URL pointing to the requested learning object.

This solution permits the minimization of the cost of implementing a "resolve-learning-object-location" step for those learning object repositories that already use SQI for searching metadata. It is currently implemented as an extension of the SQI gateway of Celebrate.

3.3 ELENA's Smart Spaces for Learning

In order to achieve interoperability among heterogeneous educational systems, the ELENA project has implemented a novel infrastructure and software solution using various Semantic Web technologies. This infrastructure is built on the following corner stones:

1. A common API for querying, the Simple Query Interface (SQI) with a web-service based instantiation of the API,
2. A common semantic model for querying and results format presentation, instantiated in XML and RDF.
3. Re-usable components for integrating existing systems with a minimum effort based on query languages, such as QEL and XQuery.

The goal of this infrastructure is the realization of a Smart Space for Learning that allows us to integrate heterogeneous educational nodes in a semantic network and provide 'smart' access technology for it [16]. In combination with process-support for learning goal definition, personalized search, and feedback tools the educational semantic network (the 'space') plays a crucial role for supporting corporate personnel development. The broad variety of learning resource types available allows us to significantly widen the scope of learning resource choices. Hereby, potential learners are not stuck with the course offerings of a particular provider or are restricted to a particular learning format, for example, a costly classroom-based course, but can expand their search to several types of learning formats, for example, books from Amazon, and providers. One driving force for the development of this feature has been an extensive requirements analyze, which has lead to the need of integrating resources of heterogeneous formats, in educational search tools [6].

For all connected systems we created a mapping to the common schema, which enabled us to issue queries against this schema. We expressed the common schema in RDF and used QEL as a query language.

So far, we have connected several systems to our network that can all be accessed by the personnel development portal HCD Online [4]. For all systems we had to create a mapping to establish the connection between the local metadata representation and our common schema. This was a challenging task, since these systems not only use different local schemas, but also differ how they represent the metadata.

The ULI Campus stores the metadata in RDF files. Academic and commercial learning (content) management systems (e.g. EducaNext, CLIX) or course databases (course catalog of the Vienna Executive Academy) often store the metadata in relational databases. They again used DBMSs from different vendors, in our case Oracle, Postgresql, MySQL, and Firebird, which cause difficulties in the way query results are encoded. Other systems store their metadata in XML files (Metzingen Continuing Education Center, EduSource educational network of Canada). A query translation technique, that translates QEL queries into corresponding XQuery queries was developed. Based on this translation technique we were able to integrate also other systems (LASON, Knowledgebay) using a native XML database (eXist). Again a different approach was required for integrating the media store of Amazon. Amazon offers a Web Services interface, so we had access to their rich metadata, stored in a proprietary format. We developed a query translation of QEL queries into Amazon search objects, which enabled a smooth integration of the available metadata.

We faced different kinds of challenges when integrating entire P2P networks (Edutella). While other systems usually give synchronous answers to queries, in case of Edutella we had to handle asynchronous answers from the network.

4. Related Work

OpenURL [3] as well as the Content Object Repository Discovery and Resolution Architecture (CORDRA) [2] are initiatives that investigate the "Identifying" problem. The work on SQI is "orthogonal" to this, in that queries and results can refer to identifiers of arbitrary nature.

Z39.50-International: Next Generation (ZING) covers a number of initiatives by Z39.50 implementers to make Z39.50 [11, 22] more broadly available and to make Z39.50 more attractive to information providers, developers, vendors, and users. SRW is the Search/Retrieve Web Service protocol, which is developed within ZING and aims to integrate access to various networked resources, and to promote interoperability between distributed databases, by providing a common utilization framework. SRW is a web-service-based protocol [23]. SRW takes advantage of CQL ("Common Query Language"), a powerful query language, which is a human-readable query.

SRW has many similarities with SQI, but also some differences. SRW is purely synchronous (source-initiated), i.e. query results are returned with the response. Additional query results can be retrieved later from the results set stored at the target for a pre-defined amount of time. SRU, the Search and Retrieve URL Service, is a companion service to SRW, the Search and Retrieve Web Service. Its primary difference is its access mechanism: SRU is a simple HTTP GET form of the service [1]. SRW encourages the use of Dublin Core, but is in general schema neutral (like SQI). SRW packs all the functionalities in a few methods and does not adhere to the "Command-Query separation principle". SRW does not provide hooks for authentication and access control nor is it based on a session management concept. It defines an Explain operation, allowing a client to easily discover the capabilities and facilities available at a particular server. SRW uses a rich set of XML-encoded application level diagnostics for reporting errors. SQI uses faults.

The purpose of the IMS Digital Repository Interoperability (DRI) Specification [10] is to provide recommendations for the interoperation of the most common repository functions. The DRI specification presents five core commands, i.e. search/expose, gather/expose, alert/expose, submit/store, and request/deliver, on a highly abstract level. The specification leaves many design choices for implementers. For example, while recommending Z39.50 (with its own query language) it also recommends XQuery as a query language. The query service does distinguish between asynchronous and synchronous query mode.

The EduSource project [7] aims to implement a holistic approach to building a network for learning repositories. As part of its communication protocol - referred to as the EduSource Communication Language (ECL) -, the IMS Digital Repository Specification was bound and implemented. A gateway for connecting between EduSource and the NSDL initiative, as well as a federated search connecting EduSource, EdNA and Smete serve as a first showcase.

OKI (Open Knowledge Initiative) is a development project for a flexible and open system to support on-line training on Internet [13]. OKI has issued specifications for a system architecture adapted to learning management functions. One of the main characteristics of the project is its commitment to the open source approach for software component development. OKI supplies specifications for a model of functional architecture and an API called Open Service Interface Definition (OSID). OKI OSID main aspects are:

- To supply specifications for a flexible and open source model of functional architecture
- Service Interface Definitions (SIDs) organize a hierarchy of packages, classes and agents and propose Java versions of these SIDs for use in Java-based systems and also as models for other object-oriented and service-based implementations.
- Components developed by OKI are compliant with specifications issued by IMS and ADL SCORM.

5. Limitations and Discussion

This paper presented the specification of the Simple Query Interface and the rationale behind its development. Although the effectiveness of the specification has been proven by several implementations, some issues still need to be further investigated.

Status Management: Methods supporting search status management could be added, for example, for cancellation of search, or query status reporting. This would allow a user at a source to cancel a search processed by a target. Similarly, query status reporting would enable a user at a source to be informed about the progress of a search processed by a target.

Explain Method and/or Capabilities Schema: No method for retrieving the capabilities of an SQI node is provided. One option here is an "explain" method. Such a method would return an SQI Profile Record that holds information on the query languages and results formats supported. Alternatively, a set of methods such as getSupportedQueryMode, getSupportedQueryLanguages, getSupportedResultsFormats, could be provided. Still another alternative is to use the SQI API itself to retrieve descriptions of the capabilities of a target (similar to the way that system tables can be queried in SQL databases). Hereby, the API could be used to answer questions like: Which query languages are supported? Which schemas are supported? Which query modes are supported? How many learning resources are available? In which format are results available? A schema describing these capabilities would be needed.

In order to be able to set all SQI parameters (queryLanguage, maxQueryResults, maxDuration, resultsFormat etc.) at once, without having to call the various individual methods separately, an additional setQueryParameters method could be introduced

SQI can be used for exchanging other things than learning resource metadata such as (language versions of) vocabularies, or evaluation data about training service providers, etc.

It would be important to find means for controlling ranking mechanisms when it comes to querying a set of targets. This would reduce the amount of data transfer, since metadata that is probably not of high user interest would not be transferred. At the same time, the quality of the results of such search would be significantly improved. Ranking mechanisms also need to be discussed in the light of privacy regulations and the capabilities of the query / retrieval semantics used on top of SQI.

6. Concluding Remarks

In order to achieve interoperability among learning repositories, implementers require a common communication framework for querying. This paper proposes a set of methods referred to as SQI as a universal interoperability layer for educational networks. At the time of writing 15 educational systems were registered at a preliminary SQI registry available at http://www.prolearn-project.org/lori/ and new implementations are ongoing.

The SQI case also shows how a standardization effort can go hand in hand with implementation work. While implementation feedback influences the standard development, only the umbrella of a standardization project can catalyze interoperability initiatives.

7. Acknowledgements

The SQI specification has been developed and financially supported under the auspices of the CEN/ISSS Workshop on Learning Technologies. This work is supported by European Commission via the IST projects CELEBRATE (http://celebrate.eun.org/), ELENA (http://www.elena-project.org/), ICLASS (http://www.iclass.info/) and PROLEARN (http://www.prolearn-project.org/). We acknowledge contributions and comments to the SQI specification from Christian Werner (Learning Lab Lower Saxony), Dan Rehak (Carnegie Mellon University), Griff Richards (Simon Fraser University), Gerhard Müller (IMC), Julien Tane (Universität Karlsruhe), Marek Hatala (Simon Fraser University), Matthew J. Dovey (Oxford University), Michel Arnaud (Université de Paris X Nanterre), Nikos Papazis (NCSR), Peter Dolog (Learning Lab Lower Saxony), Sascha Markus (IMC), Stefano Ceri (Politecnico Milano), Simos Retalis (University of Piraeus), and Teo van Veen (Koninklijke Bibliotheek).

8. References

[1] *SRU: Search and Retrieve URL Service*, vol. 2005: http://www.loc.gov/z3950/agency/zing/srw/sru.html.

[2] *CORDRA: Technical Introduction and Overview*: http://www.lsal.cmu.edu/lsal/expertise/projects/cordra/intro/intro-v1p00.html, 2004.

[3] *OpenURL*: http://library.caltech.edu/openurl/, 2004.

[4] S. Aguirre, S. Brantner, G. Huber, S. Markus, Z. Miklós, A. Mozo, D. Olmedilla, J. Salvachua, B. Simon, S. Sobernig, and T. Zillinger, "Corner Stones of Semantic Interoperability Demonstrated in a Smart Space for Learning," in *Poster Proceedings of the European Semantic Web Conference 2005*, S. Decker and H. Stuckenschmidt, Eds. Heraklion, Greece, 2005.

[5] E. Duval, E. Forte, K. Cardinaels, B. Verhoeven, R. Van Durm, K. Hendrikx, M. Wentland Forte, N. Ebel, M. Macowicz, K. Warkentyne, and F. Haenni, "The Ariadne Knowledge Pool System," *Communications of the ACM*, vol. 44, pp. 72-78, 2001.

[6] S. Gunnarsdottir, B. Kieslinger, T. Küchler, and B. Simon, "From e-Learning to Learning Management: Results of an International Survey," in *Proceedings of 4th International Conference on Knowledge Management*. Graz, Austria, 2004.

[7] M. Hatala, G. Richards, T. Eap, and J. Willms, "The Interoperability of Learning Object Repositories and

Services: Standards, Implementations and Lessons Learned," in *Proceedings of the 13th World Wide Web Conference*. New York City, USA, 2004.

[8] iClass, *Intelligent Cognitive-based Learning System for Schools: http://www.iclass.info/*, 2004.

[9] IEEE, *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: IEEE, 1990.

[10] IMS, *IMS Digital Repositories Interoperability - Core Functions Information Model*: http://www.imsglobal.org/digitalrepositories/driv1p0/imsdri_infov1p0.html, 2003.

[11] C. A. Lynch, "The Z39.50 Information Retrieval Standard - Part I: A Strategic View of Its Past, Present and Future," *D-Lib Magazine*, 1997.

[12] D. Massart, "Accessing Learning Contents Using a Simple Query Interface Adapter," in *Proceedings of the ProLearn-iClass Thematic Workshop on Learning Objects in Context*. Louvain, Belgium, 2005.

[13] OKI, *Open Knowledge Initiative*: http://web.mit.edu/oki/, 2004.

[14] S. Raghavan and H. Garcia-Molina, "Crawling the Hidden Web," in *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, 2001.

[15] *Simple Query Interface Specification*. http://www.prolearn-project.org/lori/, Version 1.0 Beta, 2004-04-13, 2005.

[16] B. Simon, S. Retalis, and S. Brantner, "Building Interoperability among Learning Content Management Systems," in *Proceedings of the 12th World Wide Web Conference*. Budapest, 2003.

[17] S. Ternier and E. Duval, "Web services for the ARIADNE Knowledge Pool System," in *Proceedings of 3rd Annual Ariadne Conference*. Leuven, Belgium, 2003.

[18] S. Ternier, D. Olmedilla, and E. Duval, "Peer-to-Peer versus Federated Search: Towards more Interoperable Learning Object Repositories," in *Proceedings of ED-MEDIA 2005*, 2005.

[19] F. van Assche and D. Massart, "Federation and brokerage of learning objects and their metadata," in *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies*, Kinshuk, C. K. Looi, E. Sultinen, D. Sampson, I. Aedo, L. Uden, and E. Kähkönen, Eds. Joensuu, Finland: IEEE Computer Society, 2004, pp. 316-320.

[20] G. Wiederhold, P. Wegner, and S. Ceri, "Toward Megaprogramming," *Communications of the ACM*, vol. 35, pp. 89-99, 1992.

[21] wordIQ.com, *Definition of Command-Query Separation*: http://www.wordiq.com/definition/Command-Query_Separation, 2005.

[22] Z39.50, "Z39.50: Part 1 - An Overview," *Biblio Tech Review*, 2001.

[23] ZING, *Search/Retrieve Web Service (SRW)*: http://lcweb.loc.gov/z3950/agency/zing/srw/, 2004.