# MISTEL - AN APPROACH TO ACCESS MULTIPLE RESOURCES

Thomas Bopp, Thorsten Hampel, Robert Hinn
*University of Paderborn*
*Fuerstenallee 11, D-33102 Paderborn, Germany*

Jan Pawlowski, Christian Prpitsch
*University of Duisburg-Essen*
*Universitaetsstr. 9, D-45141 Essen, Germany*

Keywords: Multiple sources, digital library, federated search, web service.

Abstract: Digital documents are widely spread around the web in information systems of all kinds. The approach described in this paper is to unify the access to documents and connect applications to share, search and publish documents in a standardised way. The sample implementation uses web services to integrate knowledge management, a learning management system, and a digital library. We propose a procedure to access resources in heterogenous repositories by negotiating capabilities before sending queries. The result is a unified service model for accessing resources, which can easily be supported by different repositories and clients.

## 1 INTRODUCTION

Every scientist or teacher at a university usually distributes some resources to colleagues or students. Teachers usually provide learning materials in a wide range: From single sheets of paper to books. These resources can be published in two different ways: As printed paper collections or by electronic systems. In former times, the process of getting a book has been straight forward. Entering a library, looking for the right book, and searching for a specific chapter. Obviously, this is a quite time consuming task: Students need to visit a library far away only to read one single article. This is partially solved by the internet and electronic libraries, but up to now different system classes, from digital libraries, or applications supporting the structuring of courses, to cooperative knowledge management are not coupled in any way.

The goal of our research project "'Mistel"' is to unify the access to materials and bring together knowledge organization, learning environments, libraries, and planning systems. To accomplish this task, three applications (one from each system class) are interconnected: The sTeam system (Hampel and Keil-Slawik, 2002) provides a platform for knowledge organization and serves as a learning environment for the students. For planning of lectures, ELM (Pawlowski, 2001) has been selected. Miless (Gollan et al., 1999) represents an electronic library. Beside the systems functionality, these particular systems have been chosen as they are all available on an open-source basis. The goal of the project is on one hand to demonstrate the interoperability of various systems on a webservice basis and on the other hand to find transferable webservice infrastructures. Hence, the defined webservice infrastructures are not limited to these systems.

In this article, the term *resource* is used as a collection of every kind of electronic output an author produces. Even though the term resource is mainly used to describe scientific articles and learning materials, it is not restricted to any format (MIME-type) nor to a special subject. Therefore, even video streams and audio files are called resources. The system providing and storing those resources is called a *repository*.

Our approach combines planning system with knowledge organization and digital library. In order to identify the required services, we created one scenario for each of the system classes:

The planning system is used to plan, design and create new courses/lectures. A digital library is useful to search for resources, which can be included in the course (for example as a link). If a new course is created, it can be published into the library as a SCORM (Dodds, 2004) file or uploaded to the knowledge management system and conducted there. Existing courses within the knowledge management system can be reviewed. Comments by the learners can be subsequently used to improve the course.

From the knowledge management point of view, users can find resources in the digital library and include them into their work spaces. They can be arranged with existing resources, annotated and exchanged with other learners. New resources, which



Bopp T., Hampel T., Hinn R., Pawlowski J. and Prpitsch C. (2006).
MISTEL - AN APPROACH TO ACCESS MULTIPLE RESOURCES.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - DISI*, pages 319-322
DOI: 10.5220/0002456203190322

have been cooperatively created previously and discussed with other users, are published into the library. Apart from that, the knowledge management serves as a platform to conduct the courses and lectures created by the planning system.

In this paper, we describe the process of searching and retrieving resources from digital repositories.

## 2 DISTRIBUTED REPOSITORIES

The very first step to get information about some subject is to describe it. The simplest way to do this is to choose a few keywords and assign them to a resource. More complex metadata is used by the project *ARIADNE GLOBE* to access repositories (Simon et al., 2005a). Unfortunately, the set of metadata has to match between all participants and there are just too many standards. Two well known specifications are Learning Object Metadata (LOM) (IEEE, 2002) and Dublin Core (DC) (DCMI, 2005). Both are widely used by two different groups of systems: Digital libraries use DC and Learning Object Repositories (LORs) use LOM. There are other projects connecting just one group of systems like LORs (Simon et al., 2005b) or connecting some systems in a peer-to-peer way (Nejdl et al., 2002). The Open Archives Intiative (Agnew, 2003) provides a widely used standard to access libraries, but it is also restricted to the metadata set DC and therefore cannot to be used in connection with E-Learning and LOM.

Our goal is to access different repositories with unified interfaces based on web services. A client's architecture needs to be independent from a repository's way of organizing resources. Since it is not intended to change the internals of a system, these standards have to be implemented in addition to existing interfaces. Moreover, we believe it is essential to create simple solutions which are easy to implement. It should be appealing to developers to implement the interfaces and gain access to other repositories.

Web services can be published in parallel to existing sites and services over http(s). The technology *web service* with its standard SOAP protocol (Gudgin et al., 2003) is a good basis to develop unified services, because it is a generally accepted standard, which is independent from programming languages and operating systems.

Technically, the whole communication of a query is made within the web service by XML fragments. Communication is grouped into three parts: The first one is the agreement-part, like a web-browser and server checking the presence of common encryption standards. Second, the query is build based upon the selected repositories' capabilities. This has been adopted from the http-protocol. The last step is to start the request and retrieve the resources' descriptions. After this step, the query is done and the user can select resources to either download or get a link of as described below.

First, the client queries some basic parameters (supported MIME-types, protocols to transfer data, possible permissions with description). Then, the client can check for at least one matching item in the first and second category. If this fails, no further communication is possible. After this check, other basic parameters are queried. They describe the server's method to store resources (hierarchical or flat). A server can use catalogues like the common Dewey Decimal classification[1] in libraries for assigning resources to catalogue entries. It is also possible to assign one resource to multiple catalogue entries and/or catalogues. A client is able to retrieve these classifications and metadata sets from the server. With respect to slow connections and large classifications it is also possible not to retrieve a whole classification at once but to navigate through it. At this point, a classification is treated like a hierarchical organization in a repository. Both arrangements contain a tree with resources connected to the nodes. Figure 1 illustrates

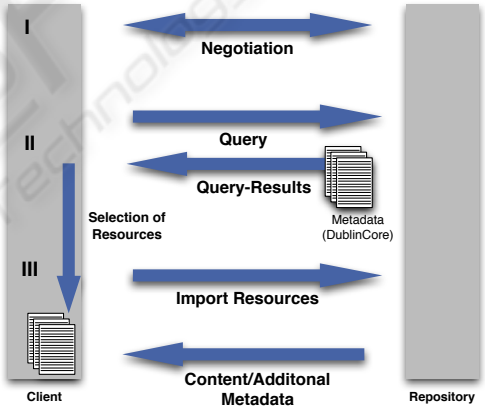


Figure 1: Importing Resources.

the communication steps between client and repository. Before any query can be submitted the two parties need to negotiate about the metadata, catalogues and supported protocols. Then the query is submitted and processed by the repository. Finally, resources are selected and imported from the repository to the client.

## 3 COMPOSING A QUERY

Our initial motivation is the connection of different system classes via a web service infrastructure. This

[1] http://www.oclc.org/dewey

infrastructure consists of different web services allowing the retrieval (search) and storage of different resources between connected applications. The retrieval of resources is either based on a hierarchical structure or on metadata sets. Every repository uses at least one set of metadata. In a very simple case this set consists of the fields author and resource title.

In the prior step, there were queries on supported metadata sets. This information is necessary to the client to generate a set of search fields. If several repositories are to be queried at the same time, they have to support an intersection of metadata sets or at least some common fields. The retrieval via metadata sets is done by putting a value to a field and adding a compare operator, e.g. `title = "How to create a query"`. This creates one search item. The connection of several items is done by AND, OR and NOT.

Hierarchical structures are used if resources are organized in a file-system or attached to classifications. Navigation in the context of this approach is done by moving from one node to a subnode and looking for attached resources and subnodes. In our approach, this is done by just one method returning the node's inventory. In case of querying many repositories at the same time, it is necessary to use the same tree in all repositories, e.g. common classifications. The client has to distinguish the resources by repository as described later.

After this step, a query consisting of query-items and connected by operators AND, OR and NOT is created. It is very simple to transform a query into SQL using XSL-transformation because many repositories use relational databases and their mathematical set theory. A similar solution of creating a query is used by SRU/SRW, a standard for encapsulating Z39.50 into XML[2] with the specification of web services. It does not consist of pure XML but expresses queries in a non XML language. This is not transformable by XSLT and therefore not part of our approach. The SQI approach defines negotiation on a common query language. It does not support several hetereogenous repositories in one query.

## 4 PERFORMING A QUERY

The created query is sent to different repositories and they return a subset of metadata defined by the user. The problem of disjunctive sets of metadata will not be discussed in this article. The client collects all responses and presents them to the user.

One very important field is the ID of a found resource. If many repositories are queried at the same time, the client has to keep the assignment to his server. This problem can be solved by adding the system's identifier (e.g. URL) to the resource's ID. There are two ways to do so: The first is to put the system's ID into the metadata (e.g. `<providedby> www.someurl.org </providedby>`). The second is to tell the repository to produce a unique ID (e.g. `<ID>www.someurl.org?1234</ID>`). The second solution conforms with metadata standards using an alphanumeric string as identifier. It is also easy to implement on the side of the repository, because it just adds the own system-ID to the ID.

The first idea of fetching the results was only to fetch resource-IDs and get the metadata in the next step. However, this solution proved to be quite slow, because of the protocol's overhead (XML, SOAP envelope, HTTP connection) and the parsing steps for XML. Too much time and bandwidth is wasted when fetching a number of resource metadata sequentially. However, fetching all metadata of all resources at the same time accelerated the query a little, but still wasting lots of bandwidth and repository resources. Inspecting the log files of a repository, we found that most users tend to select only a small subset of the result anyway. Therefore, the best solution in many cases is to fetch a small set of metadata for all resources.

After the query has been executed and the result has been received, the user chooses a subset from the resources of the result set. To be able to select the appropriate resources the common metadata is displayed for each resource. As described above this is already part of the result. Additional information (more detailed metadata) can be retrieved by another call for each resource separately.

## 5 ACCESSING RESOURCES

There are several solutions to access a resource. One is to directly transfer the complete resource to the client. This could be done by putting the resource's content and metadata into the web service response. By using BASE64-encoding it is also possible to transfer binary data. However, this solution has two disadvantages: A SOAP-message (Gudgin et al., 2003) being an XML-resource has to be completely transferred and then unpacked before any action can be taken (there is no streaming). This leads to memory problems with large resources on the client side and on the repository side while creating the message. The second disadvantage concerns the MIME-type of the resource. If the client for some reason cannot handle the resource, it has to be saved on disk and an external application has to be called. Those disadvantages are solved by not transferring the content within a SOAP message. There are many suitable protocols

[2] http://www.loc.gov/z3950/agency/zing/srw

on the internet like http(s), ftp or rsync. Those protocols are well known, generally approved, and do not include much overhead. Our approach is not to send resources through SOAP, but to submit a link where the resource can be retrieved.

To create a flexible solution the client and the repository negotiate on the protocols to use (see figure 1). Usually http(s) covers all needs, but in special cases this approach is flexible enough to arrange connections using unknown protocols. This negotiation can only be done after transferring the metadata because it depends on the MIME-type. If the resource is a streaming source, it makes no sense to download it. If a web browser has to handle streaming sources, the browser detects the MIME-type and uses a dedicated application to handle it. This workflow can be done with our approach as well: The client has to select an application and pass the link to it. Links are necessary in the context of E-Learning. A link can be put into a learning resource, shipped to learners and are always needed, if a result set of resources is to be stored. The only problem is that links can change over time (link consistence). This has to be solved by the repositories. For example, in libraries new resources can be added, but not exchanged or modified. The link has to be created carefully because future versions of the repository (e.g. new software version, other technology) must not change the link.

# 6 CONCLUSION

The described approach shows one solution for accessing different repositories. It does not matter what kind of organisation a repository uses nor what MIME-type the resources are. At the same time, the client's architecture is independent from the composition of the repository. This approach can also be used to access any system providing metadata or hierarchical structures.

With the establishment of web services as open and standardised interfaces, our architecture represents the key technology to offer unified access to different repositories. The interfaces need to be as simple as possible in order to bring together as many systems as possible. At the same time, some sort of negotiation is required, because different systems support varying protocols.

In a service oriented architecture (SOA) there is the idea of sharing services, found in centralized registries. Transferred to our approach it is possible to use indexing services of a foreign system. Systems can be reduced to their core features and therefore kept small in quantity but with huge quality.

# ACKNOWLEDGEMENTS

The project is funded by the *Deutsche Forschungsgemeinschaft* (DFG).

# REFERENCES

Agnew, G. (2003). Developing a metadata strategy. *Cataloging & Classification Quaterly*, 34(3):31–46.

DCMI (2005). Dcmi metadata terms. webpage, Dublin Core Metadata Initiative, http://dublincore.org/documents/dcmi-terms.

Dodds, P. (2004). Scorm 2004 overview. Technical report, Advanced Distributed Learning (ADL).

Gollan, H., Luetzenkirchen, F., and Nastoll, D. (1999). Miless - a learning and teaching server for multi-media documents. In Cooperman, G., Jessen, E., and Michler, G., editors, *Lecture Notes in Control and Information Sciences: Workshop on Wide Area Networks and High Performance Computing*. Springer, London.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. F. (2003). Soap version 1.2. Technical report, World Wide Web Consortium (W3C).

Hampel, T. and Keil-Slawik, R. (2002). steam: Structuring information in a team – distributed knowledge management in cooperative learning environments. *Journal of Educational Resources in Computing*, 1(2):1–27.

IEEE (2002). Draft standard for learning object metadata. Technical report, Institute of Electrical and Electronical Engineers Standards Department.

Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., and Risch, T. (2002). Edutella: A p2p networking infrastructure based on rdf. In *World Wide Web 2002*, Honolulu, Hawaii, USA.

Pawlowski, J. M. (2001). *Das Essener-Lern-Modell (ELM): Ein Vorgehensmodell zur Entwicklung computergestützter Lernumgebungen*. PhD thesis, University of Essen.

Simon, B., Massart, D., van Assche, F., Ternier, S., Duval, E., Brantner, S., Olmedilla, D., and Miklos, Z. (2005a). A simple query interface for interoperable learning repositories. In Simon, B., Olmedilla, D., and Saito, N., editors, *Proceedings of the 1st Workshop on Interoperability of Web-based Educational Systems*, pages 11–18, Chiba, Japan. CEUR.

Simon, B., Massart, D., van Assche, F., Ternier, S., Duval, E., Brantner, S., Olmedilla, D., and Miklos, Z. (2005b). A simple query interface for interoperable learning repositories. In Simon, B., Olmedilla, D., and Saito, N., editors, *Proceedings of the 1st Workshop on Interoperability of Web-based Educational Systems*, pages 11–18, Chiba, Japan. CEUR.