The MIND Architecture for Heterogeneous Multimedia Federated Digital Libraries

Henrik Nottelmann and Norbert Fuhr

Institute of Informatics and Interactive Systems, University of Duisburg-Essen,
47048 Duisburg, Germany, {nottelmann,fuhr}@uni-duisburg.de

Abstract. In this paper we describe the architecture of the MIND system for federating multimedia digital libraries. MIND integrates heterogeneous, multimedia non-co-operating digital libraries and gives the user the impression of a single coherent system. The architecture consists of a single mediator and one proxy (composed of several proxy components) for every connected library. These specialised, distributed components are connected via SOAP. This architecture with clearly defined responsibilities perfectly fits the needs of a distributed research project, and allows for integrating different platforms and programming languages.

1 Introduction

Today, people have routine access to a huge number of heterogeneous and distributed digital libraries. To satisfy an information need, three different steps have to be performed:

1. Relevant libraries have to be selected (“resource selection”).
2. The information need has to be reformulated for every library w. r. t. its schema (“schema mapping”) and query syntax.
3. The results from the selected libraries have to merged (“data fusion”).

This is an ineffective manual task for which accurate tools are desirable.

MIND (being developed in an EU project) is an end-to-end solution for federated digital libraries which covers all these issues. We started from information retrieval approaches which focus on retrieval quality, but mostly only consider monomedial and homogeneous sources. We extended these approaches for dealing with different kinds of media (text, facts, images and transcripts of speech recognition) as well as handling heterogeneous libraries (e.g. with different schemas). Another innovation is that MIND also considers non-co-operating libraries which only provide the query interface.

2 The MIND architecture

The goal of the MIND project was to develop methods and a prototype for integrating heterogeneous multimedia digital libraries (DLs). Some of these DLs might co-operate with the project, but most of them won’t. Heterogeneity appears in different forms: query languages (e.g. proprietary languages, SQL, XQuery, XIRQL), communication protocols (HTTP GET/POST, Z39.50, SOAP), document models (relational, DOM) and the logical and semantic document structure (defined by schemas).

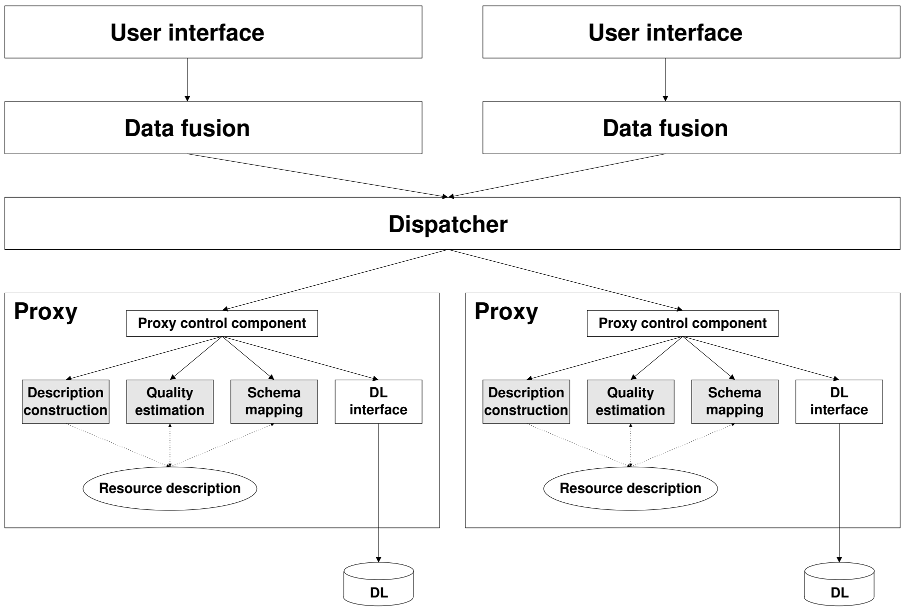

Fig. 1. MIND basic architecture

The MIND architecture reflects these three characteristics (heterogeneity, multimedia, non-co-operating DLs). It follows the standard structure with one mediator (called "dispatcher" in MIND) and wrappers ("proxies") for every library. The proxies extend the functionality of the libraries; they deal with the heterogeneity problem, and they give the dispatcher the required information not provided by the libraries. The only requirement for the DLs is that we can issue queries to it and receive documents, and that a DL client can specifiy the number of documents to be returned.

With this architecture, the dispatcher "only" has to deal with co-operating proxies. On top of the dispatcher, a "data fuser" merges the results together. The basic components of this architecture are depicted in Fig. 1.

The proxies consists of several sub-components (the grey components in Fig. 1 are media-specific):

Proxy control component: This component is called by the dispatcher (actually, it is the only one used by the dispatcher). The major job of this component is to call the other components, and to combine their results.

Description construction: This media-specific component is used for creating resource descriptions. Resource descriptions store textual descriptions of the corresponding library, e.g. its schema and statistical metadata (e.g. average term indexing weights).

Quality estimation: This media-specific component is responsible for estimating the number of relevant documents in the corresponding DL w. r. t. its media type. This is a sub-task of the resource selection problem (see Sec. 6).

Schema mapping: This media-specific component translates between the user schema and the DL schema: Queries have to be transformed from the user schema into the DL schema, and documents have to be transformed back from the DL schema into the user schema (see Sec. 4).

DL interface: The interface component is the connection from the proxy to the underlying library. Thus, this component provides a uniform access API.

Using external resource descriptions allows for reusing standard implementations of components (despite the DL interface, which has to be re-implemented for every proxy). Nevertheless, this approach is flexible so that different proxy implementations can be used as well.

We decided to develop a distributed system, where the components can reside on different machines. Communication is done via SOAP (see Sec. 3). This has several advantages over a monolithic architecture:

1. Components can be developed easily by the responsible partner, and integration is quite easy, without the problem of maintaining the sources and compiled programs on different sites.
2. Components can run on a dedicated computer which provides required resources (computation power, additional software or data sources like a thesaurus).
3. Different hardware, operating systems and programming languages can be used.
4. Load balancing and replication can improve the system scalability and reliability.

The disadvantage is that communication is more complex than in a homogeneous and centralised environment.

Concluding, we set up an architecture which shifted the main functionality into the "smart" proxies which extend the underlying DLs. The proxies also solve the heterogeneity problem; the dispatcher only sees a collection of homogeneous, co-operating libraries which adhere to a standard protocol (i.e., the MIND protocol).

3 Communication with SOAP

The hierarchical MIND communication structure can be implemented with simple procedure calls in a centralised environment. But as the MIND components are distributed, this solution has to be extended to remote procedure calls (RPC).

When the development of the prototype started in 2001, we decided to use SOAP[1]. In contrast to other solutions like XML-RPC[2], the W3C language SOAP now is the standard for connecting components (so called "web services").

SOAP messages are encoded in XML and typically transmitted via HTTP. The SOAP service has to process the request XML document and to send back a "response" XML document. The major advantage of this text-based SOAP is that it improves interoperability between different platforms and languages. In MIND, SOAP components are implemented in C++, Perl and Java.

The XML request document contains the envelope (the root element) and the body element (which contains the payload as its immediate child element). The payload

[1] http://www.w3.org/TR/SOAP/

[2] http://www.xmlrpc.com/

(body) element is identified by a fully qualified name, where the component name is the namespace URI and the procedure name is the local name. In MIND, the component name has the form "urn:NAME", e.g. "urn:dispatcher.Dispatcher" for the dispatcher or "urn:proxy-estimation-text.Google" for the text-specific quality estimation component of the Google proxy. Coding the service name into the XML message allows for attaching several components to one URL (i.e., to one HTTP server).

To reduce administration overhead, we wanted a central mapping from component names to URLs, the "registry". So, only the URL of this additional SOAP service has to be stored on the client (e.g. in a configuration file). For replication, more than one URL can be specified for the same service. For improved efficiency, clients should (and do) cache the URLs retrieved from the registry, so that the client does not have to query the registry before every SOAP call. As the registry is only used for the SOAP communication, it is not included in the original architecture (see Fig. 1).

Most of the MIND code is written in Java, using the Apache SOAP implementation.[3] Here, a client calls a "router" servlet, which processes the incoming SOAP messages, converts the payload into Java objects (which specialised serialiser classes), calls the correct object (identified by the SOAP component name), and converts the result back into XML. For simplicity, we made the SOAP protocol transparent by introducing "stub" classes which implements the same interface as the component, but whose methods call the component's method via SOAP (as in RMI). This proved to be good, as then only the Java method in the stub has to be called, making the SOAP protocol transparent. This communication process is depicted in figure 2.

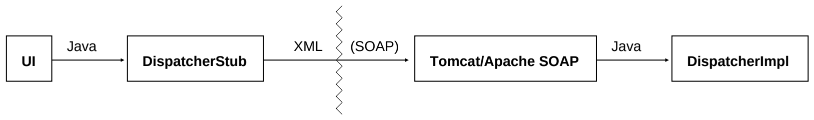

Fig. 2. MIND communication process in Java

As SOAP is an interoperability protocol, Java clients can call services written in another programming language, and vice versa.

The query-based sampling component (see section 5) is C++ software (from the Lemur[4] toolkit). Within the project, it has been extended so that it can be integrated in MIND. The query-based sampling component uses the gSOAP toolkit.[5]

4 Heterogeneous schemas

In this section, we describe and discuss the transformation of queries and documents between heterogeneous schemas.

[3] `http://xml.apache.org/soap/`
[4] `http://www-2.cs.cmu.edu/~lemur/`
[5] `http://www.cs.fsu.edu/~engelen/soap.html`

4.1 Query and document transformation

In heterogeneous digital libraries, each DL uses its own schema. Thus, queries have to be transformed from the user schema into the DL schema, and documents from the DL schema into the user schema.

In MIND, we employed a combined approach of DAML+OIL, probabilistic Datalog and XSLT [5]. The goal is to define schema mapping rules in a descriptive, textual way. A schema mapping rule specifies how a given attribute in one schema can be transformed into another attribute w. r. t. another schema; the attribute can be renamed, and the value can be modified. The latter is important for facts, where e.g. dates have to the transformed between different formats. Usually, text attributes are not modified (only renamed if necessary). MIND queries and documents are modelled in DAML+OIL [2], which has the power to become the standard ontology language. One disadvantage of DAML+OIL is that it lacks rules so far, thus DAML+OIL models are converted into probabilistic Datalog [4], and schema mapping rules are stated in this predicate logic language:

```
0.7 s2#author(D,V) :- s1#au(D,V).
0.3 s2#editor(D,V) :- s1#au(D,V).
```

This specifies that `au` has to be transformed into `author` with probability 0.7, and into `editor` with probability 0.3.

On the implementation level, the pDatalog rules are automatically converted into XSLT style-sheets. These style-sheets operate on XML serialisations of DAML+OIL models (of the underlying MIND queries and documents).

For images, this text based approach does not work, so the schema mappings are implemented manually in this case.

4.2 Implementation in MIND

The proxy transforms a user query into a DL query when necessary, thus hiding heterogeneity to the dispatcher. In other words, the dispatcher never notices that the proxy internally uses another schema. In a similar way, documents are transformed automatically from the DL schema into the user schema before they are returned to the dispatcher.

The proxy splits the query (the document, respectively) into media-specific fractions. Each fraction is forwarded to the corresponding media-specific schema mapping component. For text, facts and speech, the query (document) fraction is modelled in DAML+OIL and serialised in XML. Then, it is transformed into the targeted schema via XSLT. The resulting XML document is parsed, converted to new internal objects for the query (document), and returned to the proxy control component. There, the transformed fractions are recombined. For images, schema transformation with XSLT does not work, so it is performed in DL-specific program code.

4.3 Advantages and disadvantages of this approach

As images are handled in a way different to that of text, facts and speech, our distributed architecture with media-specific schema mapping components helps us in solving this problem.

However, we noticed that the differences for the three other media types are smaller than expected before. Thus, one schema mapping component for text, facts and speech would be sufficient and more efficient.

Our descriptive approach with probabilistic Datalog rules and the standard transformation language XSLT for processing queries and document allows for storing all library-specific information in textual files. Thus, only one standard implementation is required for all proxies.

5 Acquisition of resource descriptions

This section describes and discusses how metadata of a library ("resource descriptions") are acquired in MIND.

5.1 Query-based sampling

The second area where resource descriptions are used (besides the schema mapping task) is resource selection. For this, the resource descriptions have to contain data (e.g. average indexing weights) which can be used for estimating the retrieval quality of a DL. As MIND integrates non-co-operating digital libraries which only provide the query interface, the proxies cannot simply request resource descriptions from the DLs.

Thus, we employed the technique of query-based sampling [1]. An initial single-term query is sent to the library, and the top-ranked documents (typically four) are retrieved and added to the document sample. Then, a random term is chosen from the sample and used as the next query. This process is iterated until the sample converges, or a specified number of documents is retrieved. Finally, the sample can be used for extracting statistical metadata. So, with reasonably low costs (i.e., number of queries), an accurate resource description can be constructed from samples of, e.g., 300 documents.

Although originally developed for the CORI resource selection algorithm, it can be used for several other resource selection approaches (GlOSS, language models), including the decision-theoretic framework used in MIND.

During query-based sampling, it is easy to measure the computation and communication time of the underlying DL for that query, and to add this to the resource description. These measurements are used later for estimating the costs w. r. t. time in resource selection (see Sec. 6).

Query-based sampling is only possible for text. This means that the DL schema must contain at least one text attribute, so that this attribute can be used for sampling. Of course, the documents of the resulting sample documents can be used for deriving resource descriptions for all schema attributes.

5.2 Implementation in MIND

Query-based sampling in included in the Lemur toolkit developed by UMass and CMU and implemented in C++. For MIND, the query-based sampling part of Lemur has been extended by the project partner CMU so that it can sample MIND proxies (i.e., the query-based sampling part of Lemur can send SOAP calls to MIND proxies).

Query-based sampling is started with an external component, which issues a SOAP call to the corresponding proxy control component. This component—the entry point for all interaction with a proxy— calls a query-based-sampling component which contains the Lemur code. This component then interacts with the proxy again for sampling the underlying library.

5.3 Advantages and disadvantages of this approach

Query-based sampling in MIND is a good example for the advantages of using a distributed environment with a platform- and language-independent data exchange format like SOAP: We were able to integrate C++ software into a Java-based environment fairly easy. We were also able to handle calls in two directions: The proxy control component (written in Java) calls the query-based sampling code (C++), which then has to call iteratively the proxies for sampling the libraries.

The disadvantage is, of course, that we have to maintain and deploy another component. Also, the sampling time increases due to a larger communication overhead, but this does not matter for query-based sampling (which is done very infrequently).

6 Resource selection

In this section we briefly explain the MIND resource selection implementation.

6.1 Decision-theoretic resource selection

MIND employs the decision-theoretic framework for resource selection [7, 6]. The basic assumption is that we can assign specific retrieval expected costs (as the costs are unknown in advance) $EC_i(s_i, q)$ to each digital library DL_i when s_i documents are retrieved for query q. The term "costs" is used in a broad way and covers different sources:

Effectiveness: Probably most important, a user is interested in getting many relevant documents. Thus, the system has to estimate the expected number $E[r_i(s_i, q)]$ of relevant documents in the result set when s_i documents are retrieved from library DL_i for query q. In MIND, two approaches are used. Both are based on estimations of the probability $Pr(\text{rel}|q, d)$ that a document d is relevant to the query for the s top-ranked documents.

Time: This includes computation time at the library and communication time for delivering the result documents over the network. These costs are approximated by fitting a simple affine linear cost function by measured response times.

Money: If a DL charges for its usage, this has to be specified manually. In MIND, we only connect DLs which do not charge on a per-document basis.

A user can specify the importance of the different cost sources by user-defined cost parameters. Thus, a user can specify her own selection policy (e.g. cheap and fast results with a potentially smaller number of relevant documents). The expected costs $EC_i(s_i, q)$ then is the weighted sum of the costs of the different sources.

If the user specifies (together with her query) the total number n of documents which should be retrieved, the task then is to compute an optimum solution, i.e. a vector $s = (s_1, s_2, \ldots, s_m)^T$ with $|s| = \sum_{i=1}^{m} s_i = n$ which minimises the overall costs.

The optimum selection s can be computed by the algorithm presented in [3].

6.2 Implementation in MIND

In MIND, three components are involved in resource selection: the quality estimation component in the proxy, the proxy control component and the dispatcher.

Corresponding to the MIND philosophy of proxies extending non-co-operating DLs, costs $EC(s,q)$ are estimated by the proxy. The optimum selection—based on the costs of every proxy—can only be computed by the dispatcher.

For time and monetary costs, the MIND model uses simple approximations. These costs can be computed in the media-independent proxy control component. The situation is more complex for relevancy costs: Here, the probabilities of relevance $Pr(\mathrm{rel}|q,d)$ have to be estimated.

As this estimation is data-type- and media-specific, it is delegated to the media-specific quality estimation components. Thus, the query (already transformed from the user schema into the DL schema) is split into media-specific sub-queries. Each sub-query q_m is forwarded to the corresponding quality estimation component; the result is a list of n probabilities of relevance $Pr(\mathrm{rel}|q_m,d)$ of the n top-ranked documents (assumed that the user requested n documents) w. r. t. the sub-query. Of course, each estimator can choose its own method for deriving these probabilities, but all of them require data from the resource descriptions (e.g. average indexing weights). After all, they are combined in the proxy control component: the probabilities of relevance of the first entries in each line are combined, then the probabilities of relevances of the second entries, and so on.

6.3 Advantages and disadvantages of this approach

The MIND approach has a clear and natural separation of responsibilities: the media-specific component estimates the probabilities of relevance w. r. t. media-specific sub-queries, the proxy control component combines the probabilities and computes costs for the different cost sources, and the dispatcher computes—based on the cost estimation from all the proxies—an optimum selection.

In this decentralised approach of estimating retrieval quality, it is very easy to deal with heterogeneous schemas: Each quality estimation component only has to deal with one DL, and only sees queries in the DL schema. As the resource descriptions (and, thus, the statistical metadata required for resource selection) are stored on the proxy level, they are also separated from resource descriptions of other DLs, so heterogeneity of schemas is not a problem at all.

However, this MIND approach also has two disadvantages:

1. Each proxy has to be called (if it runs on another machine than the dispatcher, this means a SOAP call over the network), and each proxy has to query its own resource description. In the current MIND implementation, most of the resource

description data is stored in a relational database management system, as this makes it easier and more efficient when dealing with large data. However, this means that every proxy has to issue at least one SQL command to the database. Thus, resource selection in MIND is quite expensive.

2. The probabilities of relevance of the sub-queries have to be combined in the proxy control component. As each list only contains *n* documents, we combine the first probabilities, then the second and so on. However, the probabilities in the first ranks do not necessarily belong to the same (the first) document. For a precise combination, the probabilities of relevance and document ids for all documents in the sample are required; but this would lead to high communication costs in the proxy.

7 Alternative architectures

Obviously, the MIND architecture is a compromise between various requirements of different types. Five other architectures could be considered as well:

1. A monolithic, centralised architecture without distributed components is faster (as there is no communication time), but does not allow for replication. Furthermore, it does not fit the needs of a distributed research project where each partner works on separate research problems.
2. The MIND architecture could live with less components. E.g., the description construction component could be fused with the quality estimation component, as they are strongly related. In addition, the text-, fact- and speech-specific schema mappings components work exactly the same and thus could be fused. Quality estimation for text differs from estimation for facts. However, for an increased estimation quality, estimating and communicating probabilities of relevance for all documents in the DL would be useful. Thus, one could also fuse these three media-specific quality estimation components and deal with the differences inside this component.
3. Resource selection could be much faster if the costs were estimated in the dispatcher. Then, the relational database used in MIND could be queried only once, and there would be no need for calling each associated proxy. Both would lead to faster resource selection. The disadvantage is that it is more difficult to deal with heterogeneous schemas: one would either transform the sample documents to the standard schema and then create the resource description (which makes it difficult and expensive to change the standard schema), or the resource descriptions for all attributes have to be stored in one huge database table (which would decrease processing time inside the database).
4. Communication in MIND is quite expensive due to the XML-based SOAP protocol. The client has to encode the internal objects into XML, the XML record (which is much larger than the internal representation) has to be transmitted over the network, and the service component has to parse the XML structure. Other native protocols on a binary basis (e.g. CORBA) would be more efficient, but it more difficult to achieve interoperability between different platforms and programming languages. In addition, CORBA cannot be operated over a firewall, which was used in one of the project partner's department. Another solution would be that the components operate directly on the SOAP XML tree, without converting it to internal

objects. However, this would lead to a more expensive programming work, which is not suitable for a prototype system which is developed in parallel with scientific research.

5. Nowadays, peer-to-peer systems become popular. A peer-to-peer architecture is more flexible than the MIND communication structure, removes control from any single organisation, and makes it easier for new service providers to join the federation. We will deal with this kind of architecture in a forthcoming project.

8 Conclusion and outlook

In this paper we described the MIND architecture for heterogeneous multimedia federated digital libraries. MIND follows a highly distributed approach with components for every distinct task. Communication is done via SOAP, which easily allows for integrating different platforms and programming languages. This architecture perfectly fits the needs for a distributed project where each partner works on a different part of the overall research problem, and where development is done in parallel to scientific research.

In a latter project stage, it turned out that some components could be fused together, but time constraints did not allow for doing so. Also, the overall system slows down due to communication time. Thus, our architecture is a little bit oversized in the MIND context. In a production system, the system could easily be adapted.

For similar projects, however, our architecture has the big advantage that is flexible and extensible. So, new functionality can be added quite easily and in a natural way, either in one of the existing components or in new ones.

In later versions of the system, we will downsize the architecture to the level actually required. In a new project, we will extend our approach to peer-to-peer nets.

References

[1] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[2] D. Connolly, F. v. Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. DAML+OIL (march 2001) reference description. Technical report, World Wide Web Consortium, 2001. `http://www.w3.org/TR/daml+oil-reference`.

[3] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[4] N. Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110, 2000.

[5] H. Nottelmann and N. Fuhr. Combining DAML+OIL, XSLT and probabilistic logics for uncertain schema mappings in MIND. In *European Conference on Digital Libraries (ECDL 2003)*. Springer, 2003.

[6] H. Nottelmann and N. Fuhr. Decision-theoretic resource selection for different data types in MIND. In *Proceedings ACM SIGIR 2003 Workshop on Distributed Information Retrieval*, New York, 2003. ACM.

[7] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of the 26st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, 2003. ACM.