Towards Federated Search Based on Web Services

Jens Graupmann, Michael Biwer, Patrick Zimmer
University of Saarland, Germany
Department of Computer Science
P.O. Box 151150, D-66041 Saarbrücken
E-mail:graupman@cs.uni-sb.de

Abstract: Some emerging trends in the recent development of the WWW can be observed. These trends are technical, like Web Services, as well as semantic, like the integration of ontologies. We propose an architecture for a new kind of federated search system, which takes these aspects and new developments into account. A major challenge in this context is to cope with portals and the data sources behind the portals, the so-called "Deep Web". One component of the proposed architecture is the service mediator, which generates wrapper classes and additional files to make portals accessible as Web Services. Other components are an onotology server, which provides Web Service based access to different ontologies and an XML Filter server that converts different source formats to XML. This loosely coupled architecture supports federated search on semistructured data and the evaluation of semantic join operations.

1 Introduction

1.1 Motivation

Some emerging trends in the recent development of the WWW can be observed. These trends are technical, like Web Services, as well as semantic, like the integration of ontologies. They are the foundation of the Next Generation Web. We will point out some aspects of these developments and propose as an application example an architecture for a new kind of federated search system, which takes these aspects and new developments into account. One of the differences to other systems is that we do not want to do any kind of "schema integration". We don't even know the data format of some of our data sources in advance. We only want to find data objects (HTML, XML, PDF documents etc.) satisfying our search conditions as well as possible. First we will introduce some of the emerging technologies.

1.1.1 The "Deep Web" (Portals)

A lot of information in the WWW is stored in data sources, mostly relational databases, which are connected to some server application logic, which dynamically generates Web pages. These information repositories are hosted by so-called *Web Portals*. This highly

dynamic content leads to problems with today's prevalent crawler based search engines. The main issues are:

1. Information is not accessible by crawlers, because it is generated as a response to a HTML form submission.
2. Links between pages change frequently. For example the content of a category "daily news" changes daily and is regularly moved to the "archive" category.

To avoid these problems we wrap the portal search engines themselves into Web Services and use them in the fashion of a meta search engine.

1.1.2 The "Semantic Web" (Ontologies)

One important aspect needed to improve the accuracy of search engines is the inclusion of ontological information, not only to analyze Web content, but also to interpret and expand user queries. The "Next Generation Web" should be able to provide this information by itself. It has to provide a kind of "semantic interoperability" [DMHF00]. An approach to provide means formalizing this is RDF/RDF-Schema [RDF00], a language to describe Web resources and the corresponding schema containing (weak) ontological information. A more expressive language for this purpose is DAML+OIL [DAML02], which seems to become a de-facto standard.

1.1.3 The "Automated Web" (Web Services)

Web pages in general are generated by humans for humans. This is of course not surprising, but if we want to use tools to efficiently support us, we have to utilize technologies, which allow computer programs to interact efficiently by using an appropriate data format, as well as an appropriate infrastructure. In many projects such infrastructures and data exchange formats were invented, but only recently these technologies are becoming widely accepted. These technologies include, of course, XML as data exchange format and Web Services as infrastructure. SOAP is the Protocol on top of HTTP, the services themselves are described in the Web Service Description Language WSDL and registered in the Universal Description, Discovery and Integration (UDDI) Registry [UDDI01].

1.2 Related Work

Several recent approaches have dealt with the integration of heterogeneous data sources. One of them is the TSIMMIS project [CHW91] . Its architecture is based on data source wrappers, which are generated semi automatically. The wrappers, called Translators, convert the data sources into a common data format. The Carnot[CHW91] project was one of the first projects dealing with a federated search architecture integrating ontologies. In this project a general purpose ontology is used. The OBSERVER Project [MKSI96] integrates

multiple ontologies and considers problems, arising in the context of the combination of the different ontologies (mapping, inconsistencies etc.). Every node in the system has a component, called Ontology Server, which is responsible for the translation of the query expressed in terms of a user ontology (the ontology, the user has chosen before formulating the query) into queries understood by the underlying data repositories. This translation is done by utilization of mapping information defined for every data repository. In [BNL98] the topic of "Join Processing in Web Databases" is addressed. In this work first a database consisting of Web pages and links is built. Based on these materialized views a join can be computed on the link structure. This means that "Web tuples" from one table are connected to the tuples of another table, when pointing to the same URL. In contrast W3QL [KOSH98] considers the WWW itself as a large database and uses an SQL like query language for query processing.

1.3 Contribution

Most recent approaches have not addressed the issue of uncertainty in the quality of the results and data sources. But even those approaches that have considered uncertainty, have not considered join operations on multiple "uncertain" data sources. They have also posed stringent demands on the integrated data sources to permit a unified view on them, mostly realized by schema integration. In this work we introduce a federated search architecture, which aims to seamlessly integrate existing data sources, especially (Deep) Web Portals based on emerging technologies like Web Services and Web Ontologies. Furthermore we will outline how to compute a similarity join operation on these uncertain results and point out some aspects that have to be considered. Additional components are introduced for ontological support and the (pre)processing of the different data formats in order to deal more effectively with various other heterogeneous data sources.

2 Architecture

2.1 "Automated Web" meets "Deep Web"

Towards a Federated Service Web

In this section we describe a federated search architecture based on Web Service technologies. Although these techniques are rooted in the B2B area for commercial transactions [CS02], they can be adapted for other purposes, too. Figure 1 depicts an application scenario. First a client submits a query to a special kind of meta search engine. This engine should be able to condense the query to a term pair consisting of one concept or class and a set of properties for one instance of this class. In the example shown in figure 1 the submitted query deals with "*book*"s about the topic "*repairing Audi cars*". This pair is submitted to the service search engine (1). The class term is used by the search engine

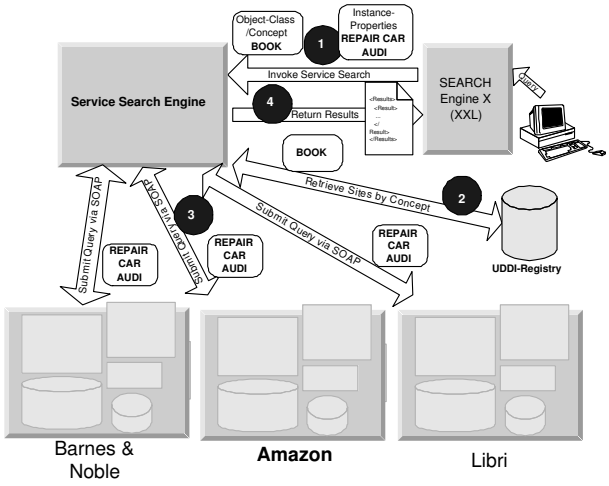


Figure 1: Web Service search

to determine relevant service providers (2). Once relevant services are found, the instance properties are submitted to these service providers (3) and results are received, combined and submitted back (4) to the first search engine.

Leveraging Web Services Infrastructure

Most existing Web Portals do not support Web Services, but they provide HTML form based access to an internal search engine. One way to integrate such servers is the generation of wrappers, which are hosted by a dedicated server application. Such an application, called *Service Mediator*, is responsible for the generation and invocation of wrapped services. If a client application gets a reference to a Web Service, hosted by the *Service Mediator,* it retrieves the corresponding WSDL description and generates an appropriate SOAP Message to invoke the service. The translation layer of the service mediator interprets the SOAP message, activates the corresponding wrapper classes and invokes the necessary methods to communicate via HTTP/HTML with the wrapped portal site to invoke the internal search engine of

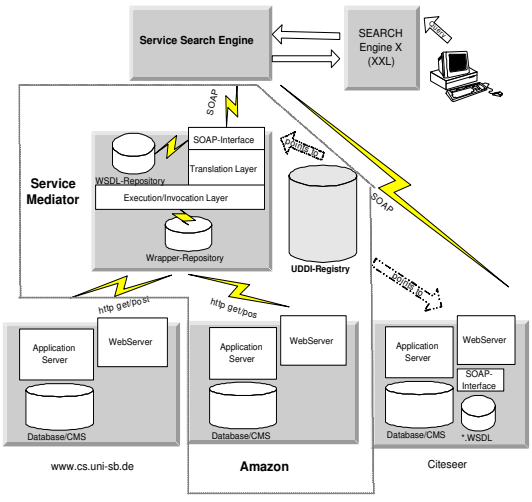


Figure 2: Service Mediator at runtime

the portal.

Figure 2 depicts a scenario with three portals. The two portals on the left do not support Web Services directly. Access via Web Services is provided by the Service Mediator. The portal on the right (Citeseer) is fully service enabled [1]. Therefore the Service Mediator is not needed to access its data. The generation of a wrapper for a specific Web site is initiated by a special SOAP message (Figure 3). This message contains the target portal's URL and additional information needed later for registration. In this generating mode, the system tries to parse the target Web site(1) and to detect form fields. Labels and internal names of the form fields are used as parameter names for the WSDL description. In many cases it might be possible to automatically generate a wrapper class for this site, but in complicated cases, especially if the target site uses script technologies manual interaction is probably indispensable. Furthermore the transformation of the result pages to XML can hardly be done automatically. For this purpose some frameworks have been developed, e.g. the W4F Toolkit[SA99], using a proprietary extraction and transformation language and the Andes Toolkit[Myl01] using XML Technologies, especially XSLT. After the generation of the wrapper classes and the corresponding WSDL description (2), the wrapped service is registered with a (UDDI-) registry (3) (This has not necessarily to be a UDDI Registry). The keywords and categories for the registration are submitted by the application, which initiated the wrapper generation. That means that this application has to do a preprocessing of the portal to extract meaningful keywords for categorization.

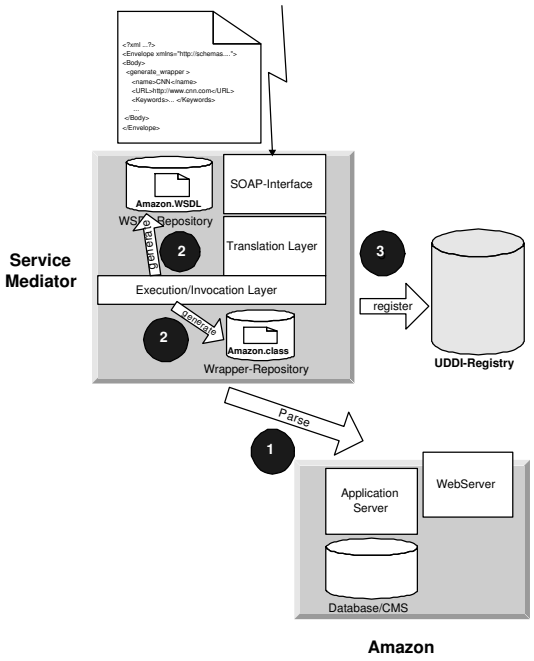


Figure 3: Service Mediator at generation time

2.2 "Semantic Web" meets "Deep Web"

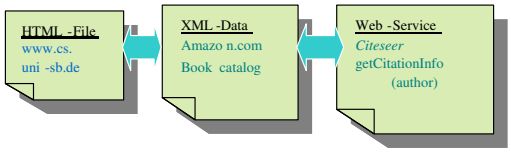
We want to enhance this architecture by the integration of "Semantic Web" technologies that are already in use in different application areas [FBDK02]. This means precisely that we integrate metadata especially ontologies into query formulation as well as query processing. The aspect of ontology based query formulation is addressed in the next section. From the architecture point of view we treat ontology sources like any other data source,

[1] Currently Citeseer does not offer such an interface. This is a fictitous example.

that we integrate with a Web Service interface. Currently we are implementing an ontology server which provides unified access to WordNet [Fel98], OpenCyc [OC02] and DAML+OIL [DAML02]. The returned results also contain scores to express the similarity of the returned descriptions to the original concept. These scores are needed for the computation of the overall similarity score of the returned values. The ontologies are connected to the server via a specific adapter which has to be implemented for every native ontology API.

2.3 Integration of Heterogeneous Data Sources

Due to the fact, that many information sources do not provide any metadata, we have to analyze the data and infer metadata to apply our ontologies to these data sources. One example for the generation of such metadata is the matching of concepts and attributes of an ontology to text fragments in a semistructured text document, e.g. the identification of book descriptions in an HTML page. In general automatic analysis is error prone. So we can not be sure about the correctness of our generated metadata. Even if the information source provides metadata, this often does not match the metadata our query was based on, e.g. one concept in the ontology, we used to pose our query does not match any concept of the ontology of the data source, but there may be many similar concepts. In consequence we have to deal with uncertainty and similarity, where results are ranked according to their "quality". To compute these values and subsequent to determine the best matches for the whole query, we have to build "virtual relations" which are views on our data objects, containing only the attributes stated in the query together with the corresponding similarity values. On the other hand if a data source is already metadata enabled, like the RDF+OIL enabled portal presented in [VDGA00], which also includes an interface for the RDF query language (RQL), the system can be integrated more easily. In the example shown below we have to compute a join between a HTML page and XML data.



Suppose the join attribute is a person name extracted from the Web page, which we want to connect to an author of a book contained in our XML book data. Furthermore we want to obtain citations of this author using a Web Service method provided by *Citeseer*. Other conceivable applications arise in corporate networks, where diverse information is distributed over relatively unstructured information sources e.g. folders containing spreadsheets, Word documents or Personal Information Manangement (PIM) applications. In this case the wrapper has to implement a whole subsystem to provide efficient access to these documents. In these cases on-demand retrieval is also crucial, because the freshness of the gathered information can be economically and legally important.

In conclusion we have to deal with the following dimensions of heterogeneity.

- **Structure**: We have to deal with unstructured data (plain text), semi structured data (like XML) and structured data (Database relations)
- **Metadata**: Data sources can contain no metadata, simple metadata (*Dublin core* metadata in HTML pages) and complex meta data (ontologies, RDF/S).
- **Ranking**: Some data sources return a list of unordered results. Other sources rank the results relatively or even with an absolute score representing their quality in respect to the query.
- **Query capabilities**: A data source does not have to provide any query capability at all (collection of text documents on a file system), but may provide a keyword based search or even an interface for a complex query language (SQL or XQuery).

To overcome the problem of structural similarity at least partly we are implementing a framework to convert different data formats to XML as an intermediate format. The XML Filter module of this framework not only converts HTML to XML, but also performs some preprocessing to support further steps. One of the aims of this HTML preprocessing is to make the hidden semantics in HTML documents more explicit, e.g. to convert the content of headings to XML Tags or extract the row and column labels of a HTML table. In Figure 4 all relevant components are shown.

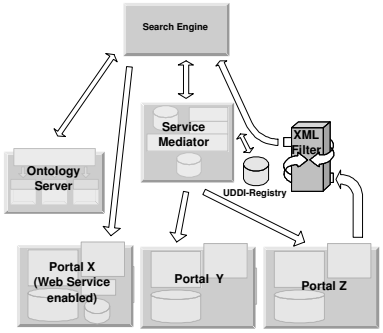


Figure 4: Architecture Overview

3 Prototype Implementation

We have implemented an initial prototype system and are in the process of extending its scope and capabilities. Currently the system automatically analyzes Web pages with HTML forms and generates the corresponding WSDL descriptions and wrapper classes for all forms on this page. The system is implemented in Java and can be invoked using a Web front-end via a Java servlet. The UDDI Registry we use is developed by HP [HP02]. For generation of the WSDL Description we use the UDDI4J package and the WSDL4J Package from IBM [IBM02]. For test purposes we have implemented another Servlet, which transform the WSDL form Description back into a HTML form via XSLT. The HTML frontend of the system allows as well the initiation of the generation, the registration at the registry with keywords, and the deletion of the Web Service as the search and invocation. Figure 5 shows a source HTML form (a), a fragment of the generated WSDL description (b) and the form rebuilt via XSLT (c). To get the rebuilt form, first the Web Service has to be retrieved from the UDDI Registry via the corresponding call. Next the WSDL description is transformed back by an XSLT script into a HTML form, similar to the original form. When this form is submitted the form data is sent back to the servlet, which parses the submitted data, builds the correct SOAP function call according

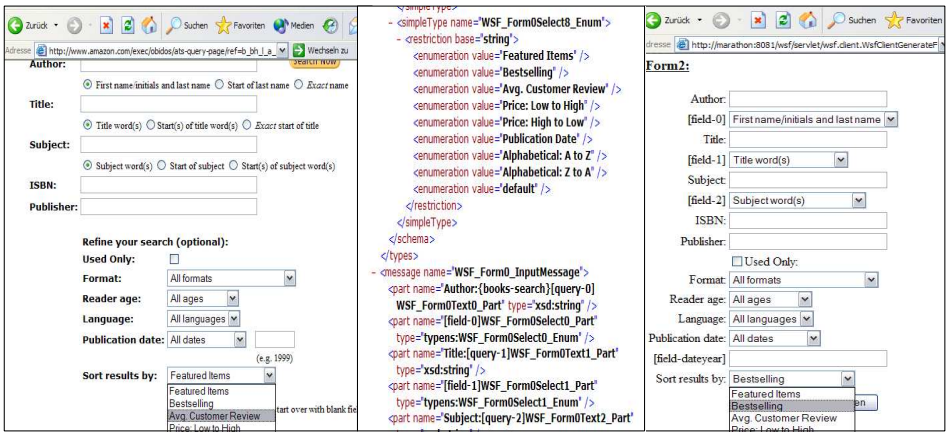


Figure 5: (a) Source Html form, (b) Generated WSDL file, (c) Rebuilt HTML form

to the WSDL description and invokes the corresponding Web Service. The wrapper then tries to send the function arguments to the HTML form in the correct format and returns the result URL to the Web Service caller. For forms using the HTTP POST Method the result is cached in a local directory and this URL is returned to the caller. The test client now simply displays the result URL. This is a very comfortable way to test the generated WSDL descriptions. Of course the search engine invokes the Web Service directly. The parser for the Web pages uses some simple heuristics to extract identifiers for form fields. Of course these can not cover every kind of HTML coding, but work fairly good on most forms. Nevertheless we are considering the use of more advanced machine learning techniques for this purpose (see, e.g., [BDS01]). This wrapper framework allows us to add new data sources almost automatically.

4 Towards Federated Search Based on Web Services and Ontologies

As a consequence of our intention to integrate many information sources with different query capabilities and data formats, we have to consider this different quality of data. A data source which provides XML files and corresponding XML schemas or RDF descriptions is easier to interpret than uncommented HTML pages, which have to be parsed and heuristically analyzed. So we have two main dimensions of uncertainty. The first is the confidence of the system to have recognized attributes in the data object correctly. For example if we want to extract the mileage of cars from car advertisements, one value for each returned attribute value pair represents the confidence of the system, that this text fragment really corresponds to a the representation of mileage. [EJX01] proposed four different facets of metadata for the computation of these values.

- Terminological relationships (synonyms, hyponyms etc.)

- Data value characteristics (the mileage of car is a number between one and one million)
- Target specific regular expression matches (if we're looking for the mileage of car in an advertisement we expect the term 'mile[s]' or 'km')
- The structure and sequence of elements.

We expand the terminological metadata facet to ontological metadata because we are interested in the number of recognized attributes of the assumed concept in the data object. The other dimension of uncertainty for this attribute value pair (recognized attribute plus value of this attribute) represents the domain dependent similarity to the value of the filter condition of the query. Based on these similarity values we want to compute a similarity join operation that combines uncertain information of different data sources. Queries are posed based on one or more ontologies. The query itself is formulated in an SQL-style query language, where relations are substituted by concepts of an ontology and schema attributes correspond to concept attributes.

5 Conclusion

We proposed an architecture for a federated search system, which integrates existing information sources, especially Web Portals but also other heterogeneous data sources utilizing state-of-the-art technologies. Most other approaches pose very strict (often only implicitly mentioned) requirements on the data sources, whereas our approach can handle very different data sources. From the outset, we considered different kinds of uncertainty in our architecture. This also allows us to compute a similarity join operation, which connects information from different heterogeneous data sources in a meaningful way [Fag96]. Although our approach integrates onotologies, we can include data sources without ontological support directly, like Web Services in the style of simple Web accessible methods. This flexibility mainly roots in the very loose coupling of the components that also allows dynamic reconfiguration (e.g. addition of new data sources) of the system at runtime.

References

[BNL98] S. S. Bhowmick, W. K. Ng, E.-P. Lim: Join Processing in Web Databases. 9th International Conference on Database and Expert Systems Applications (DEXA'98), 1998

[CGHI94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom: The TSIMMIS Project: Integration of Heterogeneous Information Sources. 16th Meeting of the Information Processing Society of Japan (IPSJ'94), 1994

[CHW91] C. Collet, M. N. Huhns, W.-M. Shen: Resource Integration Using a Large Knowledge Base in Carnot. IEEE Computer 24(12): 55-62, 1991

[CS02] F. Casati, M.-C. Shan: Models and Languages for Describing and Discovering E-Services, SIGMOD Conference 2001 (Tutorial), 2001

[DAML02] The Darpa Agent Markup Language, http://www.daml.org

[DMHF00] S. Decker, S. Melnik, F. v. Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, I. Horrocks: The Semantic Web: The Roles of XML and RDF. IEEE Internet Computing 4(5): 63-74, 2000

[EJX01] D. W. Embley, D. Jackman, L. Xu: Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. Workshop on Information Integration on the Web (WIIW 2001), 2001

[Fag96] R. Fagin: Combining Fuzzy Information from Multiple Systems. ACM Symposium on Principles of Database Systems (PODS'96), 1996

[FBDK02] D. Fensel, C. Bussler, Y. Ding, V. Kartseva, M. Klein, M. Korotkiy, B. Omelayenko, and R. Siebes: Semantic Web Application Areas. 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002), 2002

[HP02] HP total e-server; HP Web Services developer edition, http://www.bluestone.com/

[IBM02] UDDI4J, http://oss.software.ibm.com/developerworks/projects/uddi4j

[KS96] V. Kashyap, A. P. Sheth: Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach. VLDB Journal 5(4): 276-304, 1996

[KOSH98] D. Konopnicki, O. Shmueli: Information Gathering in the World-Wide Web: The W3QL Query Language and the W3QS System. TODS 23(4): 369-410, 1998

[MKSI96] E. Mena, V. Kashyap, A. P. Sheth, A. Illarramendi: OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. International Conference on Cooperative Information Systems (CoopIS 96), 1996

[Myl01] J. Myllymaki: Effective Web data extraction with standard XML technologies. International World Wide Web Conference (WWW 2001), 2001

[OC02] OpenCyc, The Open Source version of Cyc technology, http://www.opencyc.org

[RDF00] Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/RDF/

[SA99] A. Sahuguet, F. Azavant: Looking at the Web through XML Glasses. International Conference on Cooperative Information Systems (CoopIS 1999), 1999

[UDDI01] Universal Description, Discovery and Integration (UDDI), Specification 2.0. http://www.uddi.org

[VDGA00] V. Christophides, D. Plexousakis, G. Karvounarakis & S. Alexaki, Declarative Languages for Querying Portal Catalogs, 1st "DELOS"Workshop on "Information Seeking, Searching and Querying in Digital Libraries", 2000.

[BDS01] V. Borkar, K. Deshmukh,S. Sarawagi: Automatic Segmentation of Text into Structured Records. SIGMOD Conference (SIGMOD 2001), 2001

[Fel98] C. Fellbaum (Editor): WordNet: An Electronic Lexical Database, MIT Press, 1998