# Web services and data integration

Serge Abiteboul
INRIA

Omar Benjelloun
INRIA

Tova Milo
Tel Aviv U. and INRIA

## Abstract

*The developments of XML [4] and Web services [3] are changing radically the art of data integration. We briefly describe Web services and consider some of their impact on data integration. We argue that XML and Web services provide the proper infrastructure for data integration at the Web scale. This is illustrated by some work going on at INRIA on Active XML, that is XML extended by allowing the embedding of calls to Web services.*

## 1 Introduction

Data integration has been a very hot topic both for research and industry for many years. In this paper, we argue that XML and Web services provide the proper infrastructure for data integration at the Web scale. First XML provides a nice mediation model, i.e., a lingua franqua, or more precisely a syntax, that most pieces of software can (or will soon) understand. Second, Web services provide the adequate abstraction level to describe the various actors in data integration such as wrappers or mediators and the communications between them. We illustrate this thesis with an on-going software project at INRIA, namely Active XML, that is based on the embedding of Web service calls inside XML documents.

What is the information that may be found on the Web and is candidate for integration:

- HTML still forms the backbone of the Web, but popular document formats such as DOC, PDF, BS are massively present.
- A new Web exchange format, namely XML [4], promoted by the w3c [9] and the industry, is radically changing the Web. XML marries, in some sense, the document world with the database world [2]. XML can be used to represent documents, but also structured or semistructured data [2].
- Multimedia formats for images, sound, video are crucial components of the Web that we will not discuss here.
- A lot of Web data is typically retrieved as a result of queries, in particular queries to database systems. This means that data (query parameters) has to be fed to obtain more data. The term deep Web or hidden Web are used for such data typically accessed via forms, scripts or using a new standard for Web services, SOAP [3].

In this paper, we consider a uniform infrastructure for data integration based on two main standards:

1. A standard of data exchange, XML, a mark-up language for tree-structured data.
2. A standards for Web services, namely SOAP. (Or more precisely, as we will see a family of standards.)

As mentioned above, data is found on the Web in very heterogeneous forms. We will assume that appropriate wrappers have been developed to transform other kinds of formats to XML. Indeed, XML is in the spirit of data integration models that have been proposed and the fact that it is a standard makes it a natural choice for data integration.

Similarly, for accesses to databases or in general to information services (the deep Web), we will assume that such accesses are wrapped inside SOAP Web services. SOAP services provide the proper standard for such accesses: they provide a uniform protocol independent of the communication protocol, the operating system and the application language.

We will ignore here the critical issue of the wrapping of data to XML and of the wrapping of operations of the data to SOAP. This will allow us to live in a more unified world with XML as the unique data model and Web services as the unique distributed model of computation. Beside ignoring the internal logic of the wrappers, we will not be concerned with the logic of integration, e.g., is it in a warehousing or mediator approach, is it based on local-as-view or global-as-view. Thus, we will be able to focus on the infrastructure for data integration independently of the specific technicalities of the integration application that is considered. Particular kinds of wrappers and various kinds of integrations may then be used within our framework just by selecting some specific Web services with the appropriate logic.

The paper is organized as follows. We assume some knowledge of XML since this is now a well-accepted standard. We present Web services. We then consider how data integration can benefit from XML and Web services. Finally, we mention a particular research project at INRIA, namely Active XML, that is based on embedding Web service calls inside Active XML documents.

## 2 Web services in short

Web Services (successors of Corba and DCOM) are one of the main steps in the evolution of the Web. They allow active objects to be placed on Web sites providing distributed services to potential clients. Although most of the noise comes from e-commerce, one of their main current uses is for the management of distributed information. Distributed database systems always suffered from platform and software incompatibilities. Web services are not inventing anything new, but they are bringing an important breakthrough to distributed data management simply because they propose Web solutions that can be easily deployed independently of the nature of the machine, the operating system and the application languages.

In short, the general setting benefits from a variety of standards around Web services:

**Find the service/information** First one can query some yellow-pages to find the provider of some information. This is achieved for instance with a standard, UDDI [8], Universal Discovery Description and Integration. UDDI is a specification for distributed Web-based information registries of Web Services. UDDI is also a publicly accessible set of implementations.

**Understand how to obtain it** This is achieved in particular using WSDL [6], the Web Service Definition Language, a language for describing Web service interfaces, something like Corba's IDL for the Web. WSDL is an XML format for describing network services based on operations and messages. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define a functionality of a service.

**Get and use it** We can then directly obtain the information or more interestingly use it in our (mediator or warehouse) system using SOAP, the Simple Object Access Protocol. SOAP is an XML based lightweight protocol for exchange of information in a distributed environment. In particular, it allows to specify the (XML) types of arguments and service results. SOAP can be used, in particular, in combination with HTTP.

Of course life is more complicated, so one will probably have to sequence operations (see Web Service Flow Language) and consider issues such as privacy policy, cost, etc. (see Web Service Endpoint Language).

To conclude this section, one should observe that the last versions of SOAP are still rather limited, e.g., they do not support the notion of session. But it seems clear that these limitations will be removed in the future.

## 3 Data integration

When we consider the activity of data integration, we find ingredients that are similar to those of Web services:

1. Find the data sources of interest in some directories;
2. Find the descriptions of the sources;
3. If needed, find wrappers for these sources;
4. Find a mediator/warehouse or construct one – this is the core of the data integration technology;
5. Run a query.

Note that standard approaches to data integration often ignore the first two points. They assume that a set of sources is given and that their sorts and semantics are known. To achieve the grail of automatic data integration over the Web, we do need to consider these two issues.

The various functionalities above may be provided by software that may run on different machines. Consider a data wrapper that translates from one local XML dialect for chemical products to a standard XML dialect such as the Chemical Markup Language. Since it translates from XML to XML, it is natural to see such a wrapper as a Web service. Now, consider some data wrapper that manages some internal non XML format. We may see it as receiving XML data (the arguments of a query) and also producing XML data, the result that consists of data translated from its internal format to XML. So, we can see it also as a Web service. For the other partners in the integration system, it is clear that we can insist that they be Web services. So, Web services can be thereby used as the common ground (together with XML) for data integration.

To illustrate, consider Figure 1. We have included in the schema two services for finding the sources (that could be using UDDI) and their descriptions (that could be in WSDL). From a logical viewpoint, the mediator integrates data coming from 3 sources. First, source1 exports some XML and the mediator can directly maps (wrapper1) the schema of source1 to the mediation schema, say ChemicalML. The mediator uses some Web service wrapper2 to adapt the output of source2 to the mediation schema. Finally, the last source knows about ChemicalML and does itself the wrapping.

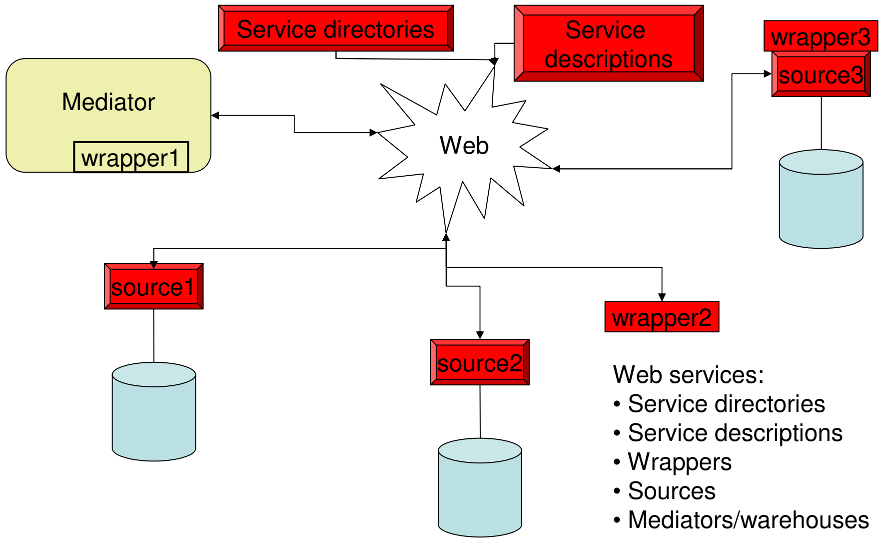


**Figure 1. Data integration with Web services**

## 4 Active XML

We illustrate briefly how this may be put to work using Active XML (AXML in short), a language and a system developed at INRIA. A description of AXML may be found in [1].

The underlying model is based on *Active XML* documents, i.e., XML documents possibly embedding calls to Web services. More precisely, in AXML documents, some XML elements are interpreted as calls to Web services. An *AXML peer* first consists of a repository that stores some AXML documents. Embedded Web service calls in the documents thus are typically used to acquire more data or to refresh data. An AXML peer is a client of Web services, AXML services or standard SOAP services. The documents in the repository are active in that some of the Web service calls may be automatically activated (in the style of triggers in active databases). An AXML peer may also act as a server: Web services operations of the peer may be defined by means of queries (e.g., XQuery) on AXML documents and exported. Being AXML data themselves, the arguments and results of service calls may also contain service calls allowing for distributed query processing over the Web.

Data with embedded calls to operations is an old idea. It exists in relational databases; it is at the core of object databases [5] and has already been considered in the context of semistructured data in, e.g., [7]. This very simple idea is the essence of Active XML. We will see that the combination of XML and Web services (e.g., in AXML) provides a uniform vision of a very wide range of heterogeneous sources provided by Web services. It also allows to capture different scenarios of data integration.

AXML documents may use standard Web services that take XML data as arguments and return XML data. Beyond such simple Web services, AXML data may refer to more complex services, which really allows to take fully advantage of the combination of XML and Web service technologies:

**Controlling call activation:** An essential issue in this framework is when to actually invoke each particular service call. Based on the specification, the system may choose to do so regularly (e.g., weekly), when the information is needed (as in mediator systems), or beforehand (as in data warehouses). By allowing declarative control of service calls activation, the language allows to combine mediation and warehousing in a very flexible manner.

**Stream services:** these are services such as subscriptions that entail a possibly permanent relationship between the caller and the service. Such services provide the caller with a (possibly infinite) stream of answers/notifications and can be used, in particular, for change management. (Such features are not yet supported by the current version of SOAP, so they have to be handled on top of SOAP.)

**Services exchanging AXML:** Standard Web services typically exchange "plain" XML data. In contrast, we allow Web services to exchange AXML data that may contain calls to other services. This allows, for instance, for delegating portions of a computation to other sites, i.e., to distribute computations. (Recall that AXML uses an XML syntax, so there is no difficulty in exchanging AXML documents using SOAP.)

## 5 Simple scenario

In this section, we use a simplified syntax for AXML. The exact syntax (an XML dialect) would unnecessarily complicate the presentation. Suppose that we are at a site local.com and we want to integrate information about a particular chemical product "Pb". First, suppose that the sources that provide the relevant information are not known a-priori. A first phase is needed to discover them. This may be a call to a service directory such as:

```
<sc name="ChemicalDirectory"
    server"wci.com">
    <topic>Pb</topic>
    <format>ChemicalML</format>
    </sc>
```

The directory service may also be used then to find automatically wrappers. This means that the directory service knows the main exchange formats (for chemical products) and some codes (Web services) to translate from one to another.

Once this search has been performed, the system disposes of several sources and of some wrappers between them:

```
<source name="s1"
   server="m1.co" operation="getchemical"
   <preprocessing>
   <wrapper server="local.com"
            operation="translate">
   <arg><a from="H2ML" to="CML"></arg>
   </wrapper>
   </preprocessing>
</source>
<source name="s2">...</source>
<source name="s3">...</source>
```

Then the sources are queried, say in a warehousing mode, and the data document is enriched with their (wrapped) result. The language AXML is used to specify how to obtain information (by querying the directories or data sources when they are now), how to enrich the information that is obtained (e.g., by wrapping it or by monitoring its changes in a warehouse mode), and how to integrate it.

**Acknowledgment** We would like to thank all the people who worked on AXML and in particular, Bernd Amann, Vikas Bansal, Angela Bonifati, Gregory Cobena, Khai (Fred) Dang Ngoc, Ioana Manolescu, Benjamin Nguyen, Roger Weber.

## References

[1] S. Abiteboul, O. Benjelloun, and T. Milo. Towards a flexible model for data and service integration, preliminary report,. In *Internat. Workshop on Foundations of Models and Languages for Data and Objects*, 2001.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, San Francisco, California, 1999.

[3] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, and D. Winer. Simple Object Access Protocol (SOAP) 1.1. W3C Note, May 2000. http://www.w3.org/TR/SOAP.

[4] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, October 2000. http://www.w3.org/TR/REC-xml.

[5] R. G. G. Cattell, editor. *The Object Database Standard: ODMG-93*. Morgan Kaufmann, San Mateo, California, 1994.

[6] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Definition Language (WSDL) version 1.1. W3C Note, March 2001. http://www.w3.org/TR/wsdl.

[7] O. I. Structured and S. Data. T. Lahiri and S. Abiteboul and J. Widom. In *Proc. International Workshop on Database Programming Languages*, 1999.

[8] Universal Description, Discovery, and Integration of Business for the Web (UDDI). http://www.uddi.org.

[9] The World Wide Web Consortium (W3C). http://www.w3.org.