Distributed information management with XML and Web services[*]

Serge Abiteboul[**]

INRIA-Futurs, LRI and Xyleme
Serge.Abiteboul @inria.fr

Abstract. XML and Web services are revolutioning the automatic management of distributed information, somewhat in the same way HTML, Web browser and search engines modified human access to world wide information. To illustrate, we present Active XML that is based on embedding Web service calls inside XML documents. We mention two particular applications that take advantage of this new technology and novel research issues in this setting.
This paper is based primarily on research at INRIA-Futurs in the Gemo Group, around XML and Web services (in particular, the Xyleme, Active XML and Spin projects).

1 Introduction

The field of distributed data management has centered for many years around the relational model. More recently, the Web has simplified a world wide (or intranet) publication of data based on HTML (the backbone of the Web) and data access using Web browsers, search engines and query forms. However, because of the inconvenience of a document model (HTML is a model of document and not a data model) and limitations of the core HTTP protocol, the management of distributed information remains cumbersome. The situation is today dramatically improving with the introduction of XML and Web services. The Extensible Markup Language, XML [32], is a self-describing semi-structured data model that is becoming the standard format for data exchange over the Web. Web services [37] provide an infrastructure for distributed computing at large, independently of any platform, system or programming language. Together, they provide the appropriate framework for distributed management of information.

From a technical viewpoint, there is nothing really new in XML and Web services. XML is a tree data model much simpler than many of its ancestors such as SGML. Web services may be viewed as a simplified version of Corba. Together, they are bringing an important breakthrough to distributed data management simply because they propose Web solutions that can be easily deployed and

[*] The research was partly funded by EC Project DBGlobe (IST 2001-32645), the RNTL Project e.dot and the French ACI MDP2P

[**] Gemo Team, PCRI Saclay, a joint lab between CNRS, Ecole Polytechnique, INRIA and Université Paris-Sud

used independently of the nature of the machine, the operating system and the application languages. XML and Web services do not solve any open problem but they pave the way for a new generation of systems and generate a mine of new problems to solve.

We first describe some key aspects of XML and Web services (Section 2).

In Section 3, we argue for Active XML [7], AXML in short, that consists in embedding calls to Web services in XML documents. AXML documents provide extensional information as well as intensional one, i.e., means to obtain more data. Intensional information is specified by calls to Web services. By calling the service one can obtain up to date information. AXML also provides control of the service calls both from the client side (pull) or from the server side (push).

We illustrate issues in distributed data management and the use of AXML through two particular applications. In Section 4, we consider content warehouses, i.e., warehouses of non-numerical data, in a Peer-to-Peer environment. In Section 5, we consider the management of personal data.

This paper is not meant as a survey of distributed data management but more modestly, as a survey of some works of the author around the use of XML and Web services for distributed data management. The author wants to thank all the people with whom he worked on these projects and in particular B. Amann, O. Benjelloun, S. Cluet, G. Cobéna, I. Manolescu, T. Milo, A. Milo, B. Nguyen, M.-C. Rousset and J. Widom.

2 XML and Web services

XML is a new exchange format promoted by the W3C [36] and the industry. An XML document may be viewed as a labeled ordered tree. An example of an XML document is given in Figure 1. We will see in the next section that this document is also an Active XML document. 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. Observe that unlike HTML, XML does not provide any information about the document presentation. (This is typically provided externally using a style sheet.)

In an XML document, the nodes are labeled. The document may be typed with a declaration given in a language called XML schema [33]. If such a document typing is provided, the labeling provides typing information for pieces of information. For instance, the typing may request a *movie* element to consist of a *title*, zero or more *author*s and *reviews*. Some software that knows about the type of this document will easily extract information from it. In that sense, the labels provide both a type and semantics to pieces of information. The typing proposed by XML schema is very flexible. In some sense, XML marries the document world with the database world and can be used to represent documents, but also structured or semistructured data [2]. For instance, it allows to describe a relational database as well as an HTML page.

An essential difference between a data standard such as XML and a document standard such as HTML is the presence of structure that enables the use of queries beyond keyword search. One can query XML documents using query languages such as XPath or XQuery.

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 hype around Web services comes from e-commerce, one of their main current uses is for the management of distributed information. If XML provides the data model, Web services provide the adequate abstraction level to describe the various actors in data management such as databases, wrappers or mediators and the communications between them.

Web services in fact consist of an array of emerging standards. To find the desired service, one can query yellow-pages using UDDI [31] (Universal Discovery Description and Integration). Then to understand how to obtain the information, one use in particular WSDL [38] (Web Service Definition Language), something like Corba's IDL for the Web. One can then get the information with SOAP [30] (the Simple Object Access Protocol), an XML based lightweight protocol for exchange of information. Of course life is more complicated, so one also often has to sequence operations (see Web Services Choreography [39]) and consider issues such as access rights, privacy policy, encryption, payment, transactions, etc.

XML and Web services are nothing really new from a technical viewpoint. However, they form a nice environment to recycle old ideas in a trendy environment. Furthermore, they lead to an El-Dorado in terms of new problems and challenges for computer science research.

First, the trees are ordered and the typing more flexible. Tree automata [10], a technology rarely used in the context of data management, is a most appropriate tool for XML typing and query processing. Automata theory is perhaps enriching here with a new dimension the topic of descriptive complexity [16] that combines logic (to specify queries) and complexity (the resource required to evaluate queries).

Next, the distribution of data and computation opens new avenues at the frontier between data management (PODS-SIGMOD [26] like) and distributed computing (PODC [25] like). The distributed computation of a query may require opening connections between various query processors and cooperating towards answering the query, perhaps having partial computations move between different systems. Indeed, the context of the Web, a P2P environment with autonomous data sources, is changing dramatically the query evaluation problem:

- In classical distributed databases, we know about the structure of data in particular sources and the semantics of their content. On the Web, we typically discover new sources that we need to exploit. For instance, to integrate data sources [18], we need to understand their structure, possibly restructure them (tree rewriting) and build semantic bridges between their information (ontology mediation).

– In classical data management, we control data updates using technology such as transaction management or triggers. On the Web, data sources are autonomous and such control is unavailable. We may have to use services to synchronize copies of some data. We may also have to subscribe to change notifications. Often, the only solution is to poll regularly the data source to learn about changes. In most cases, we have to live with a lower consistency level.

Last but not least, we have to deal with the scale of the Web. The most spectacular applications of distributed data management are perhaps in that respect, Web search engines, e.g., Google, and Web look-up services, e.g., Kazaa. In both cases, the difficulty is the scaling to possibly billions of documents and millions of servers. This is leading to new algorithms. This also requires rethinking classical notions such as complexity and computability. For instance, how would one characterize the complexity of a query requiring a crawl of the entire Web? Linear? What would be the consistency of the result when a large number of the pages that have been visited no longer exist or have changed since our last visit.

3 Active XML

To illustrate the power of combining XML and Web services, we briefly describe Active XML that consists in embedding Web service calls in XML documents. This section is based on works in the context of the Active XML project [7].

In Active XML (AXML for short), parts of the data are given explicitly, while other parts consist of calls to Web services that generate more data. AXML is based on a P2P architecture. Each AXML peer acts as a client by activating Web service calls embedded in its documents. It also acts a server by supporting Web services corresponding to queries or updates over its repository of documents.

AXML is an XML dialect. For instance, the document in Figure 1 is an AXML document. The `sc` elements are used to denote embedded service calls. In this document, reviews are obtained from the `cine.com` Web site. Information about more Hitchcock movies may be obtained from the `allocine.com` site.

The data obtained by a call to a Web service may be viewed as intensional (it is originally not present). It may also be viewed as dynamic, in the sense of dynamic Web pages. The same call possibly returns different, up-to-date documents when called at different times. When a service call is activated, the data it returns is inserted in the document. Therefore, documents evolve in time as a consequence of service call activations. Of particular importance is thus the decision to activate a particular service call. In cases, this activation is decided by the peer hosting the document. For instance, a peer may decide to call a service only when the data it provides is requested by a user; the same peer may choose to refresh the data returned by another call on a periodic basis, say weekly. In other cases, the Web service provider may decide to send updates to the client, for instance because the latter registered to a subscription-based, continuous service.

A key aspect of the approach is that AXML peers exchange AXML documents, i.e., document with embedded service calls. Let us highlight an essential difference between the exchange of regular XML data and that of AXML data. In frameworks such as Sun's JSP or PHP, dynamic data is supported by programming constructs embedded inside documents. Upon request, all the code is evaluated and replaced by its result to obtain a regular, fully materialized HTML or XML document. But since Active XML documents embed calls to Web services, one does not need to materialize all the service calls before sending some data. Instead, a more flexible data exchange paradigm is possible, where the sender sends an XML document with embedded service calls (namely, an AXML document) and gives the receiver the freedom to materialize the data if and when needed.

To conclude this section, we briefly mention two issues to illustrate the novel problems that are raised by the approach:

To call or not to call: Suppose someone asks for the information we have about the Vertigo movie. We may choose to call `cine.com` to obtain the reviews or not before sending the data. The decision may be guided by considerations such as performance, cost, access rights, security, etc. Now if we choose to activate the service call, it may return a document with embedded service calls and we have to decide whether to activate those or not, and so on, recursively. The solution we implemented is based on a negotiation between the peers to determine the type of data to exchange. This leads to complex automata manipulation [21]. Indeed, the general problem has deep connections with alternating tree automata, i.e., tree automata alternating universal and existential states [27].

Lazy service calls: As mentioned above, it is possible in AXML to specify that a call is activated only when the data it returns may be needed, e.g., to answer a query. A difficulty is then to decide whether a particular call is needed or not. For instance, if someone asks for information about the actors of "the 39 steps" of Hitchcock, we need to call `allocine.com` to get more movies of this director. Furthermore, if this service is sophisticated enough, we may be able to ask only for information about that particular movie ("push" the selection to the source). Some surprising connections between this problem and optimization techniques for deductive database and logic programming are exhibited in [6].

4 P2P Content Warehousing

In the next two sections, we mention two Web applications based on distributed information management, P2P content warehousing in this section and personal data management in the next. The content of this section is based on works in the Xyleme project [35] for content warehousing and on Spin [5] and MDP2P [20] for the P2P aspect.

Content warehouse Distributed enterprises and more generally communities centered around some common interest produce and need to share large volumes of data, e.g., reports, emails, contact addresses, documentation, contracts, Web sites. It is becoming more and more difficult to manage this information and critical to be able to find the desired data in a timely manner. This suggests organizing this information in *content warehouses* [4].

The goal of a warehouse [28] is to provide an integrated access to heterogeneous, autonomous, distributed sources of information. Queries are typically evaluated without access to the original sources. The focus here is on a warehouse of *content*, i,e., "qualitative information" and documents in various formats, rather than OLAP (on-line analytical processing) warehouses that are more concerned with "quantitative information", that are typically organized in relations.

A content warehouse supports the construction, enrichment, monitoring and maintenance of large repositories of information with methods to access, analyze and annotate this information. Clearly, XML is the obvious candidate to represent the warehouse information, and in particular the meta-data about the warehouse documents, although some other formats such as pdf, html or doc, have also to be handled. Furthermore, since most systems will soon support Web service interfaces, Web services provide the natural infrastructure for communications in such a warehouse.

Peer to peer systems There are many possible definitions of P2P systems. We mean here that a large and varying number of computers cooperate to solve particular tasks (here warehousing tasks) without any centralized authority. In other words, the challenge is to build an efficient, robust, scalable system based on (typically) inexpensive, unreliable, computers distributed on a wide area network. In the context of distributed data management, P2P is becoming more and more popular. See, for instance, [9, 24, 14, 11, 1, 12].

The implementation of a content warehouse in a P2P setting may be motivated by a number of factors. For instance, from an economic viewpoint, a P2P system allows to share the costs and reduce them by taking advantage of existing infrastructures. Also, in a setting of a content warehouse with a very large volume of data and many users, we can take advantage of the distribution to improve performance.

The combination of the concept of warehouse (a centralized access to information) and of a P2P architecture (by definition stressing distribution) may seem a bit confusing, so let us try to articulate it more precisely. The data sources are heterogeneous, autonomous and distributed. The warehouse presents a unique entry point to this information, i.e., the warehouse is *logically* homogeneous and centralized. A P2P warehouse is an implementation of the concept of warehouse that is based on distributed peers. So, the warehouse is *physically* distributed over heterogeneous and autonomous machines. Note that a higher level of trust may possibly be achieved between the warehouse peers than between the original data sources.

From a technical viewpoint, the main issue is *distributed data management*, by no means a novel issue [22]. However, in a P2P environment, the context and in particular, the absence of central authority and the number of peers (from hundreds to possibly millions), change the rules of the game. Typically, problems that have long been studied take a different flavor. In particular, the following issues need to be addressed: (i) Information and service discovery (ii) Web crawling, (iii) document ranking (for queries) (iv) P2P mediation (integrating independent ontologies, e.g., [15, 13, 17]), (v) change monitoring.

In an on-going project called Spin (for set of pages of interest), we have worked on a centralized content warehouse. An XML repository is used to store the data and in particular the meta-data about the documents of the warehouse. Clearly, the management of meta-data is here an essential component, as advocated in Tim Berners-Lee's view of the Semantic Web [8]. All the processing is achieved via Web services, e.g., Web crawling, classification, tagging. A content warehouse in a particular domain may be specified using some graphical user interface. This specification is compiled into AXML documents where information to be acquired by the warehouse is described by Web service calls as well as the processing to be performed on this data. We are currently working on turning Spin into a P2P system.

5 Personal data

The second Web application we consider here is distributed personal data management. This section is influenced by works in the DbGlobe project [23].

The management of personal data is becoming an important issue [19]. We typically handle more and more personal data in a variety of formats, distributed on a number of devices. In particular, we handle emails, addresses, bookmarks, documents, pictures, music, movies, bank accounts, etc. For instance, the information we have on a particular friend may be stored on a pda (address book, agenda), in phone directories (mobile or fix), in car repository (GPS coordinates, maps), in Web pages (her Web site), in pictures and movies, etc. Furthermore, lots of small personal data sources will soon be available in our environment, e.g., our house or our car.

All this personal data should be viewed as a distributed database. Indeed, to manage such data, we need most of the functionalities we mentioned for content warehouses; e.g., we need to query it or archive it. Furthermore, we need to maintain it up to date and in particular, we need to synchronize automatically various replicas of the same information.

Because of the standards they provide for distributed information management, XML and Web services form the appropriate infrastructure for doing this. Our thesis is that Active XML is the right articulation between the two to build more flexible systems for distributed personal data management.

One particular aspect of personal data management is that some of the devices are mobile, e.g., pda, car, phone. This has direct impact on aspects such as availability (a pda may be off line) or performance (a mobile phone has little storage and limited bandwidth). Also, some functionalities may depend on the location. For instance, to print a document, we need to find a nearby printer; and to select a restaurant, we care only for nearby places.

To mention another related application we are considering, consider the customization of a personal computer. It is typically is a cumbersome task that overwhelms most computer users. However, when considered from an abstract viewpoint, this is a simple task of data management. A particular person first needs access to its personal data. We already mentioned how this could be achieved. Then, the person is used to a particular environment and needs some specific software. If the specific needs of that person are described in an (A)XML document, we believe that the task of customizing the system to her could be handled automatically by the system. In particular, specific software the person uses could be obtained via Web services and maintained as well using Web service-based subscriptions.

Web services and XML only facilitate the exchange of information. The hard problems remain such as developing user friendly interfaces in this setting or automatically integrating and maintaining this information (including change control).

6 Conclusion

We have briefly discussed XML and Web services. We have illustrated how these could be used to facilitate the management of information distributed over a network such as the Web using Active XML and two applications, namely P2P content warehousing and personal data management. As already mentioned, there is a mine of new problems to study.

The management of structured and centralized data was made feasible by a sound foundation based on the development of relational database theory with deep connections to descriptive complexity. For the management of semistructured and distributed information, we are now at a stage where a field is building and a formal foundation is still in infancy. The development of this foundation is a main challenge for the researchers in the field.

References

1. K. Aberer, P-Grid: A Self-Organizing Access Structure for P2P Information Systems, CoopIS, 179-194, 2001.
2. S. Abiteboul, P. Buneman, and D. Suciu, Data on the Web, Morgan Kaufmann Publishers, 2000.
3. S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, T. Milo, Active XML Documents with Distribution and Replication, ACM SIGMOD, 2003.
4. S. Abiteboul, S. Cluet, G. Ferran and A. Milo, Xyleme Content Warehouse, Xyleme whitepaper, 2003.
5. S. Abiteboul, G. Cobena, B. Nguyen, A. Poggi, Construction and Maintenance of a Set of Pages of Interest (SPIN), Conference on Bases de Donnees Avancees, 2002.

6. S. Abiteboul and T. Milo, Web Services meet Datalog, 2003, submitted.
7. The AXML project, INRIA, http://www-rocq.inria.fr/verso/Gemo/Projects/axml.
8. T. Berners Lee, O. Lassila, and R. R. Swick, The semantic Web, Scientic American, vol. 5, 2001.
9. R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreutz, S. Seltzsam, K. Stocker, ObjectGlobe: Ubiquitous query processing on the Internet, The VLDB Journal, 10:48, 2001.
10. Tata, Tree Automata Techniques and Applications, H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tommasi, www.grappa.univ-lille3.fr/tata/
11. F. M. Cuenca-Acuna, C. Peery, R. P. Martin, T. D. Nguyen, PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, Department of Computer Science, Rutgers University, 2002.
12. T. Grabs, K. Böhm, H.-J. Schek, Scalable Distributed Query and Update Service Implementations for XML Document Elements, IEEE RIDE Int. Workshop on Document Management for Data Intensive Business and Scientific Applications, 2001.
13. F. Goasdoue and M.-C. Rousset, Querying Distributed Data through Distributed Ontologies: a Simple but Scalable Approach, 2003.
14. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, D. Suciu, What can databases do for peer-to-peer? WebDB Workshop on Databases and the Web, 2001.
15. A. Y. Halevy, Z. G. Ives, D. Suciu, I. Tatarinov, Schema Mediation in Peer Data Management Systems, ICDE, 2003.
16. N. Immerman, Descriptive Complexity, Springer 1998.
17. A. Kementsietsidis, M. Arenas, and R. Miller, Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues, ACM SIGMOD, 2003.
18. M. Lenzerini, Data Integration, A Theoretical Perspective, ACM PODS 20'02, Madison, Winsconsin, USA, 2002.
19. The Lowell Database Research Self-Assessment Meeting, 2003. research.microsoft.com/ gray/lowell/
20. Massive Data in Peer-to-Peer (MDP2P), a research project funded by the ACI Masses de Donnees of the French Ministry of Research, www.sciences.univ-nantes.fr/irin/ATLAS/MDP2P/
21. T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, F. Dang Ngoc, Exchanging Intensional XML Data, ACM SIGMOD, 2003.
22. M.T. Özsszu, and P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, 1999.
23. E. Pitoura, S. Abiteboul, D. Pfoser, G. Samaras, M. Vazirgiannis, et al, DBGlobe: a service-oriented P2P system for global computing, SIGMOD Record 32(3): 77-82 (2003), DBGlobe is an IST research project funded by the European Community.
24. The Piazza Project, U. Washington, data.cs.washington.edu/p2p/piazza/
25. PODC, ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing.
26. PODS, ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems;
SIGMOD, ACM SIGMOD Conference on the Management of Data.
27. Active Context-Free Games, L. Segoufin, A. Muscholl and T. Schwentick, 2003 (submitted)
28. J. Widom, Research Problems in Data Warehousing, In Proceedings of the 4th Int'l Conference on Information and Knowledge Management (CIKM), 1995.

29. F. M. Cuenca-Acuna, C. Peery, R. P. Martin, T. D. Nguyen, Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities, Department of Computer Science, Rutgers University, 2002.
30. Simple Object Access Protocol (SOAP) by Apache. ws.apache.org/soap/
31. The Universal Description, Discovery and Integration (UDDI) protocol, www.uddi.org/
32. The Extensible Markup Language (XML), www.w3.org/XML/
33. XML Typing Language (XML Schema), /www.w3.org/XML/Schema
34. XML Query Language (XQuery), www.w3.org/XML/Query
35. Xyleme Web site, www.xyleme.com
36. The World Wide Web Consortium (W3C), www.w3.org/
37. W3C Web Services Activity, www.w3.org/2002/ws/
38. Web Services Description Language (WSDL), www.w3.org/TR/wsdl
39. W3C Web Services Choreography Working Group, www.w3.org/2002/ws/chor/

```
<directory>
  <movies>
    <director>Hitchcock</director>
    <movie> <title>Vertigo</title>
      <actor>J. Stewart</actor> <actor>K. Novak</actor>
      <reviews> <sc service=reviews@cine.com >Vertigo</sc></reviews>
    </movie>
    <movie> <title>Psycho</title>
      <actor>N. Bates</actor>
      <reviews> <sc service=reviews@cine.com >Psycho</sc></reviews>
    </movie> <sc service=movies@allocine.com >Hitchcock</sc>
  </movies>
</directory>
```

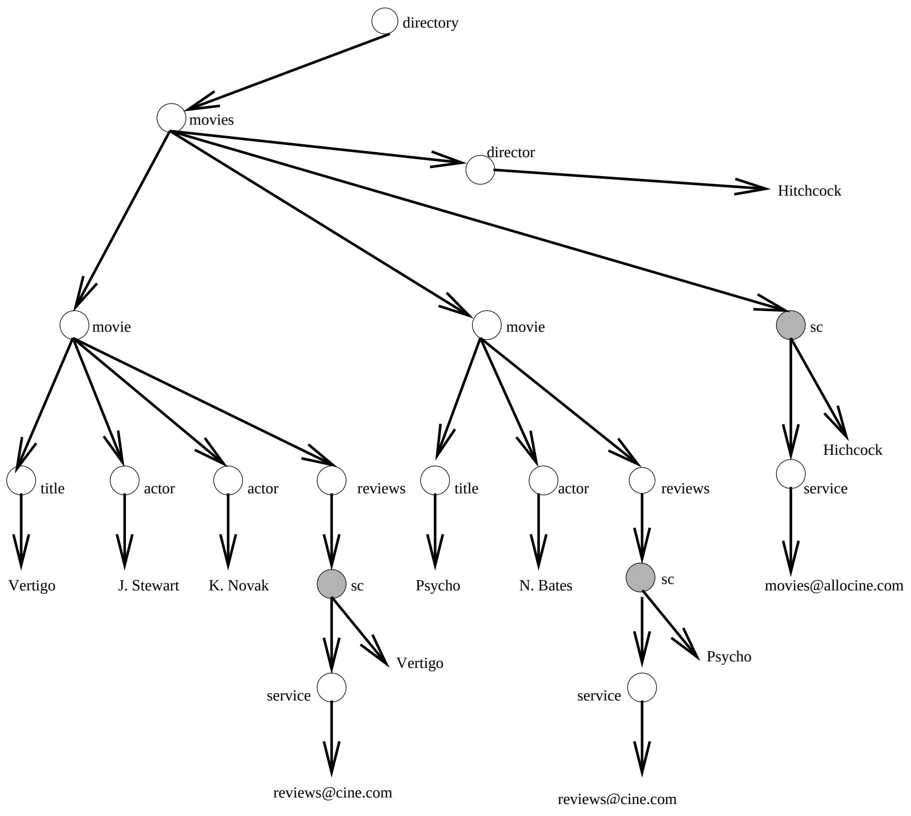

Fig. 1. Active XML document and its tree representation