Active XML and Data Activation* (Extended Abstract)

Serge Abiteboul[1], Susan Davidson[2], Tova Milo[3]

[1] INRIA-Futurs and LRI-Universit Paris-Sud
[2] U. of Pennsylvania
[3] Tel Aviv University

1 Introduction

The field of distributed data management [13] has centered for many years around the relational model. More recently, the Web has made the world wide and intranet publication of data much simpler, by relying on HTML, Web browsers, plain-text search engines and query forms. However, the management of distributed information remained cumbersome. The situation is dramatically improving today with the introduction of XML [18] and Web services [21]. Together, these two standards provide an infrastructure for distributed computing at large, independent of any platform, system or programming language, i.e., the appropriate framework for distributed management of information.

We discuss here Active XML (AXML, for short), a declarative framework that harnesses these emerging standards for the integration and management of distributed data. An AXML document is an XML document where some of the data is given explicitly, while other portions are given only intensionally by means of embedded calls to Web services. By calling the services, one can obtain up-to-date information. In particular, AXML provides control over the activation of service calls both from the client side (pull) or from the server side (push).

It should be noted that the idea of mixing data and code is not new, e.g., stored procedures in relational systems [15], method calls in object-oriented databases [10], and queries in scripting languages such as PHP. The novelty is that since both XML and Web services are standards, AXML documents can be universally understood, and therefore can be universally *exchanged*.

In the present paper, we focus on the idea of "activating" portions of static data, e.g., transforming an XML document into an Active XML document. As we will see, a wide range of XML sub-documents are candidates for being activated. Furthermore, one may want to activate data for a number of different motivations. In some cases, it may be to introduce new functionalities, e.g., to provide monitoring over the data; in other cases, it may be for performance reasons, e.g., to lower the cost of refresh. Activation may be performed either manually (by the application designer) or automatically (e.g., by a Web server).

* The research was partly funded by the RNTL Project e.dot, the French ACI MDP2P, the Fulbright Program, and Hitachi.

We conclude this introduction by a brief discussion of XML and Web services. AXML is considered in Section 2 and data activation in Section 3.

```
<directory>
  <movies>
    <director>Hitchcock</director>
    <sc service="movies@allocine.com" >Hitchcock</sc>
    <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>
  </movies>
</directory>
```

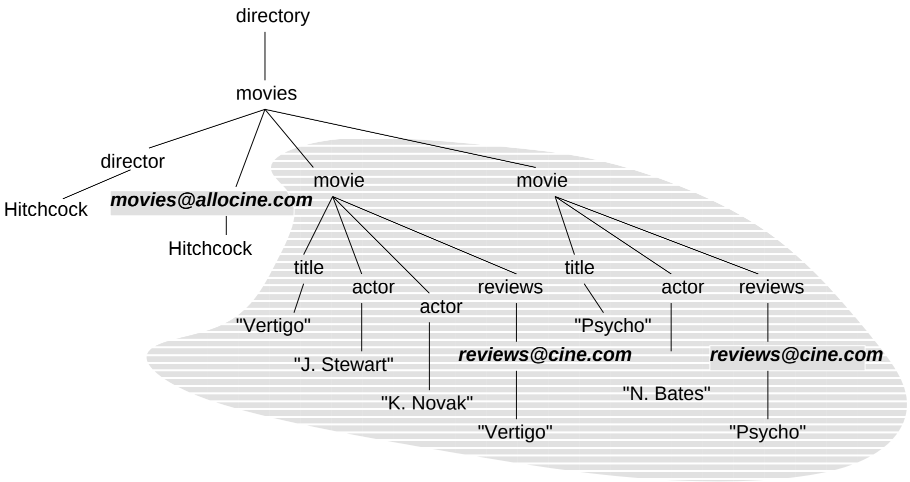

Fig. 1. An AXML document and its tree representation

XML and Web services. XML is a new semistructured data exchange format [3] promoted by the Word-Wide-Web Consortium and widely adopted by industry. An XML document can be viewed as a labeled ordered tree, as seen in the example[1] of Figure 1. Unlike HTML, XML does not provide any information about the document presentation. This is typically provided externally using a CSS or XSL style-sheet. XML documents may be typed, e.g., using XML

[1] We will see in the next section that this XML document is also an AXML document.

Schema [19], and may be queried using query languages such as XPath or XQuery [20]. Web services consist of an array of emerging standards. For instance, to find a desired service, one can query a UDDI [17] directory. To understand how to interact with the service, one relies on WSDL [22], something like Corba's IDL. One can then access the service using SOAP [16], an XML-based lightweight protocol for the exchange of information.

2 Active XML

In this section, we briefly describe AXML. Details may be found in [7].

In AXML, parts of the data are given explicitly, while other parts consist of calls to Web services that generate more data. The approach is based on the exchange of AXML documents in a peer-to-peer setting, where each AXML peer manages a (possibly persistent) set of AXML documents. Each peer acts as a client, by activating Web service calls embedded in its documents, and also acts as a server, by providing AXML documents and supporting Web services that correspond to queries or updates over the documents it manages. The activation of calls can be controlled to occur periodically, in reaction to some particular event (as with database triggers), or in a *lazy* way, whenever it may contribute data to the answer of a query.

AXML is an XML dialect, as illustrated by the document in Figure 1. (Note that the syntax is simplified in the example for purposes of presentation.) The `sc` elements are used to denote embedded service calls. Here, reviews are obtained from `cine.com`, and information about more Hitchcock movies may be obtained from `allocine.com`. The data obtained by a call to a Web service may be viewed as intensional, as it is not originally present. It may also be viewed as dynamic, since the same service call possibly returns different data when called at different times. When a service call is activated, the data returned is inserted in the document that contains the call. Therefore, documents evolve in time as a consequence of call activations. Of particular importance is thus the decision to activate a particular service call.

The AXML project has been going on for several years. We conclude this section by mentioning some results around AXML.

A system is now available as open source [7]. In [12], a technique to decide whether or not calls should be activated based on typing is introduced. The general problem has deep connections with tree automata [11] and alternating automata, i.e., automata alternating between universal and existential states [14]. Optimization issues in the context of AXML are presented in [1]. In [4], a framework for managing distribution and replication in the context of AXML is considered. Foundations of AXML are studied in [2].

3 Data activation

Consider some piece of information, e.g., a Paris guide. It contains some relatively static information such as a list and descriptions of museums. It also contains more dynamic information such as a (typically short) list of hotels that are not full and a list of temporary exhibits. Each time the guide is downloaded, some useful refreshes are performed, but some unnecessary ones are performed as well. By activating the document, one can adapt to the particular needs of the application. In some sense, one splits the information into smaller pieces so that the application is able to "guard" each piece of information with specific policies.

Data activation is, in some sense, the opposite of data materialization. The model we use, AXML, mixes extensional and intensional data, so it captures both aspects. For instance, consider again the activation of the list of exhibits. We should now see this list as the result of a particular guarded Web service. We can attach to this Web service a policy to guide its usage, e.g., the Web service is activated only when needed and the result is considered as valid for 3 days. Thus the information is activated at the provider side and is materialized and cached at the client side. This approach therefore provides much more elaborate control of caching than that provided by current Web server technology.

What is activation good for? A main motivation for activation is the control of changes. For instance, the guarding service may be used for sophisticate caching, for archiving, for monitoring information or for highlighting recent changes to a user. The guard may also be used for enforcing concurrency control, access control or other security based mechanisms such as the encryption of the data. Guarding services may also be used to enrich the data in many ways such as summarization, classification, introducing semantic tags, or reorganizing the data.

Another important use of activation is for improving the performance of exchanges. Consider a Web server that exports a large document. When the document changes, we need to transfer the entire document to refresh it. If portions of it are activated, we can have a more selective refresh strategy. Furthermore, the entire document may be guarded by a service capable of transmitting *deltas*, i.e. able to transmit the changes instead of retransmitting the entire document.

Finally, a large use of a data activation mechanism has to do with views. The XML format presents the disadvantage of somehow mixing the physical and logical levels. The unit of transfer of information is the entire XML document, which is as well the proposed logical unit to view the data. Activation allows more flexibility. A collection of pages may now be viewed as a document, and conversely, one or more elements in a document be viewed as a new document.

What may be activated? To set the scope of data activation, let us consider what types of data are candidates to be activated:

XML The focus is on Web exchange, so XML is an obvious candidate.

HTML Since HTML can be turned easily in XHTML, a particular XML idiom, HTML is also an obvious candidate.

Doc In general, any document, e.g., MS-Word or PDF, which presents some structure is a potential candidate; but in practice, this would require adapting the AXML framework to such formats.

Set A collection of documents is also a natural candidate because it is easy to "wrap" such a collection into an AXML document.

Query Some virtual XML data such as the result of an XML query or of some Web service.

Let us consider in more detail query activation since it may seem somewhat odd to "activate a query", i.e., some already intensional information. Why would one replace a query call by a call to some generic Web service in charge of guarding the query? We have found a number of advantages to do so. For instance, the same query may be used in various places. By activating it, we can organize the sharing of resources and for instance, we can better control the caching of results. Furthermore, suppose that the query is parameterized, e.g., Report($userName), and produces a report customized for this particular user. We can then factor out the part of the query that is common to all users and, instead of reevaluating everything at each request, evaluate only the part that is specific to the particular one.

Who activates data? Data activation may be decided and realized manually or automatically. Consider for instance the tourist guide. The publisher of the guide knows the application and may choose which portion of the guide to activate as well as the activation policies, e.g., the refresh policy for each portion of the document. On the other hand, an intelligent Web server may detect that users download regularly the entire guide and decide to turn it into active data, so that the download would be much faster, say using a *diff* algorithm. Both the guide designer or the Web server may decide (perhaps based on user requests) to propose a subscription mechanism to obtain the list of new exhibits on a regular basis.

Choosing what to activate. As mentioned earlier, the choice to activate some data may be guided by statistics about its usage, change rates, and performance requirements. From the point of view of the designer of the document (e.g., the Web site designer), one could imagine the development of a theory in the spirit of design theory for relational databases. Consider for instance a directory of projects with people assigned to projects. It certainly makes sense to activate each project because users may be selectively interested in some projects. On the other hand if people have addresses with street numbers, city and zip-codes, it probably does not make sense to activate the zip-code because it is unlikely to change separately from the rest of the address.

4 Conclusions

Active XML is a new model of distributed data management which rests on emerging Web standards (XML and Web services). We mentioned a number of performance improvements that may be obtained by data activation, e.g., for Web servers. More work remains to be done do develop optimization techniques that take advantage of these new opportunities. As with any data model, the question of design must be addressed. Recent work has considered design in the context of an XML document [8, 9, 5, 6], and has focussed on the problem of redundancy and update anomalies. However, with Active XML the question is changed: We now have the option of activating portions of the static data. Data activation opens up many different design opportunities, and is an exciting new area of research.

References

1. S. Abiteboul, O. Benjelloun, B. Cautis, I. Manolescu, T. Milo, N. Preda, Lazy Query Evaluation for Active XML, In Proc. of ACM SIGMOD 2004.
2. S. Abiteboul, O. Benjelloun, T. Milo, Positive Active XML, In Proc. of ACM PODS, 2004.
3. S. Abiteboul, P. Buneman, D. Suciu, Data on the Web, Morgan Kaufmann, 2000.
4. S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu, T. Milo, Active XML Documents with Distribution and Replication, In Proc. of ACM SIGMOD, 2003.
5. M. Arenas, Leonid Libkin, A normal form for XML documents, Proceedings of the 21th Symposium on Principles of Database Systems (PODS), 2002.
6. M. Arenas, L. Libkin, An Information-Theoretic Approach to Normal Forms for Relational and XML Data, Proceedings of ACM Symposium on Principles of Database Systems (PODS), 2003.
7. The AXML project, INRIA, http://activexml.net.
8. P. Buneman, S. Davidson, W. Fan, C. Hara, W. Tan, Keys for XML, Proceedings of WWW10, 2001.
9. P. Buneman, S. Davidson, W. Fan, C. Hara, W. Tan, Reasoning about keys for XML, International Workshop on Database Programming Languages (DBPL), 2001.
10. The Object Database Standard: ODMG-93, editor R. G. G. Cattell, Morgan Kaufmann, San Mateo, California, 1994.
11. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tata, Tree Automata Techniques and Applications, www.grappa.univ-lille3.fr/tata/
12. T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, F. Dang Ngoc, Exchanging Intensional XML Data, In Proc. of ACM SIGMOD, 2003.
13. M.T. Ozsu, P. Valduriez, Principles of Distributed Database Systems, Prentice-Hall, 1999.
14. A. Muscholl, T. Schwentick, L. Segoufin, Active Context-Free Games, Symposium on Theoretical Aspects of Computer Science, 2004.
15. J.D. Ullman, Principles of Database and Knowledge Base Systems, Volume I, II, Computer Science Press, 1988.
16. The SOAP Specification, version 1.2, http://www.w3.org/TR/soap12/
17. Universal Description, Discovery and Integration of Web Services (UDDI), http://www.uddi.org/
18. The Extensible Markup Language (XML), http://www.w3.org/XML/
19. XML Typing Language (XML Schema), http://www.w3.org/XML/Schema
20. An XML Query Language, http://www.w3.org/TR/xquery/
21. The W3C Web Services Activity, http://www.w3.org/2002/ws/
22. The Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl/