ActiveXML: Compound Documents for Integration of Heterogeneous Data Sources

João P. Campos and Mário J. Silva

Faculdade de Ciências da Universidade de Lisboa, Portugal
Departamento de Informática*
{jcampos, mjs}@di.fc.ul.pt

Abstract. We address the problem of automatic composition of XML documents with data from multiple sources and their presentation to large groups of users with different information requirements.

1 Introduction

Automatic document composition by retrieving information from multiple sources faces the problem of finding a tool to seamlessly query many data types and formats. The multitude of formats of the documents to be generated presents another problem. Continuous advances in technologies for information presentation demand new formats and methods.

The database approach for information management advocates the use of declarative languages for automatic construction of documents, when retrieving data from multiple digital libraries [2]. Query languages such as Lorel [9], Quilt [3], XML Query Language [5] or XML-QL [1] allow querying an XML database for elements in XML documents, but most digital libraries today are not available as XML databases.

To cope with the heterogeneity of the existing data sources it is desirable to integrate declarative and imperative languages in one scripted document. Web servers enable the generation of documents parameterized by user input. Common Gateway Interface, Microsoft Active Server Pages (ASP) and Java Server Pages are examples of techniques that allow a web site to create a page when a user hits the page address.

Although web servers can execute multiple programs in many languages to achieve dynamic behavior, they all have the limitation that their output is of text type (usually to be interpreted as HTML by their clients). To have increased flexibility, we propose the exchange of structured data among the multiple data source handlers that could be invoked to compose dynamic web pages. With the adoption of XML we have the opportunity of developing a common structured data exchange format for heterogeneous language environments.

* This work was partially funded by research program PRAXIS XXI, medida 3.1.b)

2 ActiveXML

ActiveXML is our scripting engine for XML data integration. The operational model of activeXML is described in figure 1. It receives an XML scripted document, calls the necessary interpreters to resolve the scripted parts in the document and outputs another XML document.

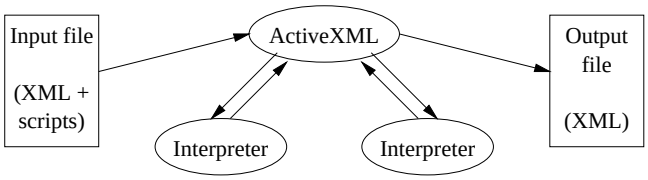


Fig. 1. ActiveXML in operation. The scripting engine receives a document with embedded scripts as input. It runs through the document and copies every element to the output document. If the element to be copied is a <*script*> element, it calls the interpreter indicated as an attribute along with the XML element as input. The interpreter then follows an established policy for handling the contents of the script elements and returns a well formed XML element that the scripting engine then appends to the output document

ActiveXML only outputs XML data, but XML is capable of representing most data structures. In addition, as it can be integrated with most other XML-based software, such as XSLT (eXtensible Stylesheet Language Transformations) [13], it can easily be extended to generate any kind of textual data. Our approach for supporting multiple interpreters for data integration from many sources enables the use of all languages proposed for handling web data sources. ActiveXML differs from mediators [4] in its approach for data integration. Mediators receive queries in a unified query language, translate them into the query method of the data source and return result data; ActiveXML receives queries in any language, invokes the interpreter for that language to handle the query and collects the output as XML data. This multi-language approach enables the use of the most expressive language available for each task.

3 Online Web News Personalization with ActiveXML

We developed ActiveXML in the context of a daily online newspaper [8, 6] that needs to publish multiple front pages generated, for a set of user profiles. The front pages contain news from the newspaper archive and other sources, such as stock quotes for the "Business" user profile or the weather forecast and movie listings for the Lisbon local news. The front pages have to be automatically generated and kept up to date. To specify the front page for a given profile, the editor selects the three most important news, the main stock quotes and two

targeted advertisements. This page is then published in HTML for the Web, as WML for WAP devices [14], and as data to be retrieved by dedicated browsers in Palm Pilots [7].

In this environment, ActiveXML is at the core of the data retrieval/generation processes. To gather information from the sources, a single script produces a document with all the data to be published (see figure 2(a)). In the next stage, we process the documents with the news of each profile to be transformed for publication in the supported media. As the documents are in XML, we use XSLT to translate them into the desired formats. As non XSLT formats might be needed (like Palm Pilot's database streams) we replay the documents through ActiveXML, but this time the interpreters have the side effect of generating new documents in specialized formats (see figure 2(b)).

This approach makes the various steps of the document generation process independent. The contents update policy specification is separated from the document to enable easy re-generation of customized front pages. The document contents generation is separated from its style specification to enable changing the style while keeping the documents structure. This also encourages reusability. In our environment, the WebGenerator, which outputs static HTML pages based on an XSL stylesheet, can also produce WML contents, given the appropriate stylesheet.

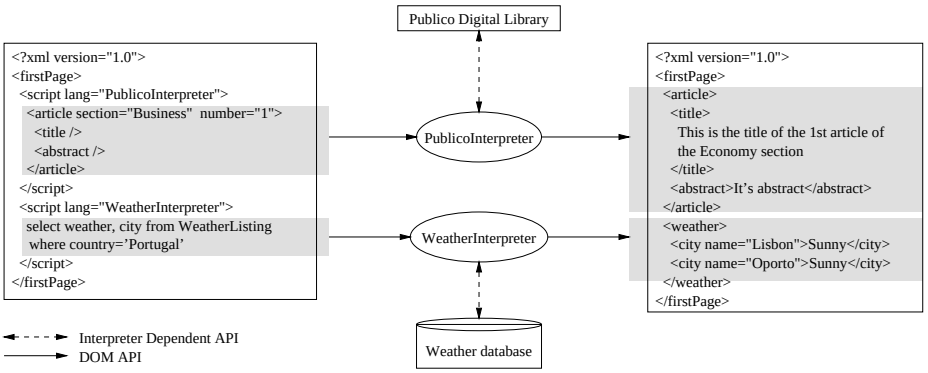
Although ActiveXML aims at integrating and publishing information, the virtual versus materialized approach for data integration is left open to the developer of the interpreters and generators. For example, in the case of the publication of daily news one might want to materialize the data, to obtain the best performance with the web server. Our WebGenerator interpreter when processing stock quotes outputs code for a script enabled web server to get the data on the fly.

4 Conclusions and Future work

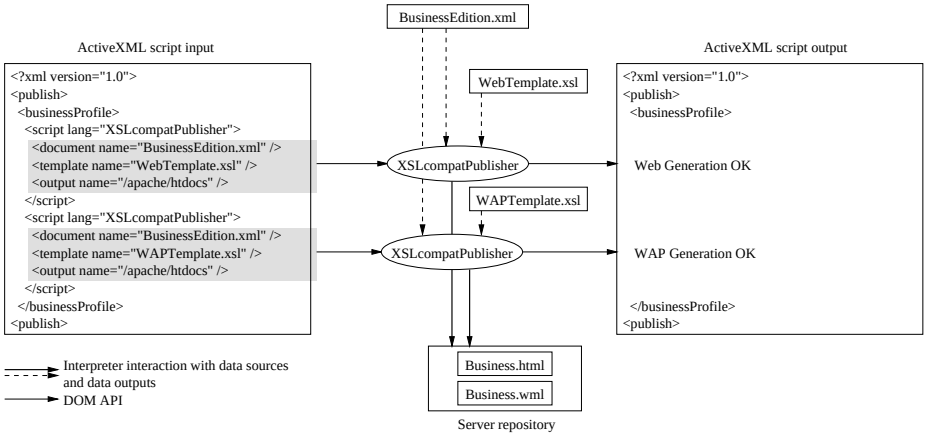
We developed a framework for integrating XML data and programs and process them in a structured and uniform way. This is achieved through a model for the publication in multiple formats of composite documents made from information items coming from multiple sources.

ActiveXML is very general and flexible but has many limitations. Among others it is hard to manage link relationships between generated documents. However some of the interpreters, such as JavaScript and VBScript can manipulate XML documents through a common API (DOM [11]). This could be used in the future for sharing data among interpreters and ActiveXML.

ActiveXML will be one of the components of XMLBase [10], our new component based system for XML data management, under development.



(a) Composing an XML document with activeXML. In our prototype, the script invokes two interpreters that query two data sources: *PublicoInterpreter* retrieves articles from *Publico's Digital Library*; *WeatherInterpreter* queries an SQL weather database



(b) Handling multiple formats: the script specifies the output formats intended for the document. The activeXML engine successively calls interpreters to retrieve the data document and a stylesheet. It generates a log of its actions as an XML file and, as a side effect, data for user presentation is saved in a repository, ready to be served to clients

Fig. 2. ActiveXML in action, generating the front pages for a personalized newspaper in multiple formats

References

1. Alin Deutsch, Mary Fernandez, Daniela Florescu *et. al.*. A Query Language for XML. `http://www.research.att.com/~mff/files/final.html`, May 2000.
2. Daniela Florescu, Alon Levy and Alberto Mendelzon. Database Techniques for the World Wide Web: A Survey. In SIGMOD Record 27(3), 1998.
3. Don Chamberlin, Jonathan Robie and Daniela Florescu. Quilt: An XML Query Language for Heterogeneous Data Sources. In WebDB 2000, Proceedings of the Third International Workshop on the Web and Databases, May 2000 (to be published as a Springer LNCS volume).
4. Gio Wiederhold. Mediators in the Architecture of Future Information Systems. IEEE Computer, March 1992.
5. J. Robie, J. Lapp, D. Schach. XML Query Language. `http://www.w3.org/TandS/QL/QL98/pp/xql.html`, May 2000.
6. Nuno Maria, Pedro Gaspar, Nuno Grilo, António Ferreira, Mário J. Silva. ARIADNE - Digital Library Architecture. In Proceedings of the 2nd European Conference on Digital Libraries (ECDL'1998), 1998.
7. Palm Inc. Handheld Computing Devices. `http://www.palm.com/`. May 2000.
8. Público Online. `http://www.publico.pt`. May 2000.
9. Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom and Janet L. Wiener. The Lorel Query Language for Semistructured Data. International Journal on Digital Libraries, 1(1):68-88, April 97.
10. XMLBase - semi-structured data management system. `http://xldb.fc.ul.pt/xmlbase/index.html`, May 2000.
11. W3C. Document Object Model. `http://www.w3.org/DOM/`. May 2000.
12. W3C. XML Linking Language (XLink), W3C Working Draft 21-February-2000. `http://www.w3.org/TR/2000/WD-xlink-20000221/`
13. W3C. XSL Transformations (XSLT), Version 1.0. W3C Recommendation 16 November 1999. `http://www.w3.org/TR/1999/REC-xslt-19991116`
14. Wap Forum. `http://www.wapforum.org/`. May 2000.