XML-driven Homogeneity in E-commerce Applications[1]

ROBERT ANDREI BUCHMANN
Business Information Systems Dpt., Faculty of Economics and Business Management
University "Babes Bolyai" Cluj Napoca
Str. Teodor Mihali 58-60, 400591, Cluj Napoca
ROMANIA

Abstract: - XML vocabularies open new doors for Web applications in general and e-business applications in particular, by inducing a new level of homogeneity and interoperability and by adding new value to traditional data and data structures. Whether XML is used as a data medium, as a data repository or as a presentation source, it is able to provide a backbone which increases application modularity, homogeneity and connectivity. This paper provides an overview for a potential e-commerce application architecture which is intended to improve some software quality parameters and bring dynamic web pages closer to the 3-tier principles, while keeping clear client-server modularity. The Macromedia Flash XML parsing implementation is used to build client-side XML packages which are processed along the XML backbone of the presented model, through an abstract XML gate that clearly separates client and server modules.

Key-Words: - E-commerce, XML backbone, XML in Flash, Relational database, Hierarchical data

1 Introduction

From the software quality perspective, Web applications have developed in a heterogeneous and collaborative fashion, by continuously building interface solutions between existing technologies. The global character of the Internet, software competition and open-source efforts contribute day by day to the increasing heterogeneity which further raises difficulties to the standardization process, global usage and interoperability [5]. The 3-tier modularity is blurred by the traditional dynamic page generation model which continuously loads the connection between the client and the server. Portability and interoperability with enterprise-oriented software (ERPs, CRMs etc.) is also affected by the increasing interfacing effort between different data models, different operating or data management systems. The XML standard was generally adopted to reduce application heterogeneity and provide a way of sharing information on various levels (structure, syntax, semantic, processing recommendations). An XML backbone reclaims converting data to the backbone rather than direct interfacing between every two modules of a system and both software competitors and the open-source community adopted XML as a basic, common way of structuring data separated by the presentation or processing needs. Furthermore XML vocabularies (DTD or XMLSchema) provide consistent data interpretation and validation on various levels. Recent technologies such as OWL push XML into the knowledge management field. On a more general level, XML covers the gap between data-oriented technologies and text-oriented technologies, a gap that was not induced by the natural form of information (which is usually mixed - data points floating in text blocks), but by the way software products chose to model it.
Macromedia is one of the companies that implemented basic XML parsing in its Web-oriented products and its Flash product provides great data connectivity features allowing developers to build XML-driven multimedia applications [7].

2 Problem Formulation

The purpose is to build an e-commerce application model able to improve some quality parameters, such as [2]:

- Portability of the data repository;
- Portability of the data inputs and outputs;
- Modularity with respect to the 3-tier principles;
- Homogeneity of data structures (along and XML backbone);

[1] This article contains results of a research grant managed by its author, as part of the CEEX program 2005 (identified with code 16, module 2-ET), financed by the Romanian Ministry of Education and Research - the Research National Authority

- Low module coupling (the decrease of connection load) and more intense local processing;
- Multimedia potential and friendly user interface (by using Macromedia Flash).

Traditional Web applications usually adopt the following structure:

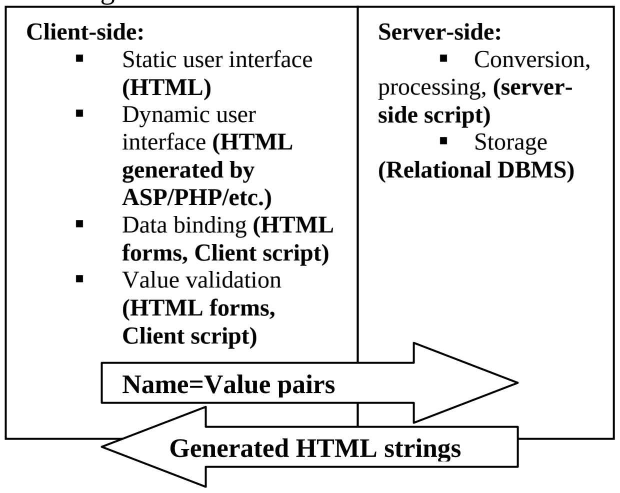
Client-side:
Static user interface (HTML)
Dynamic user interface (HTML generated by ASP/PHP/etc.)
Data binding (HTML forms, Client script)
Value validation (HTML forms, Client script)
Server-side:
Conversion, processing, (server-side script)
Storage (Relational DBMS)
Name=Value pairs
Generated HTML strings

Fig. 1. Traditional web application mode

The figure clearly shows the heterogeneity of the involved coupled technologies and the way in which 3-tier separation is blurred due to the interference of the server scripts as a provider for the dynamic user interface.

3 Problem Solution

3.1 Overview

Data portability is provided by the use of XML trees validated against a problem-specific vocabulary (such as an e-commerce vocabulary).

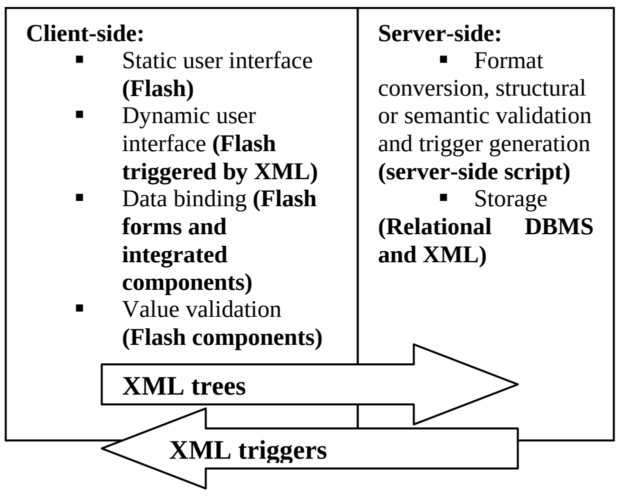
Client-side:
Static user interface (Flash)
Dynamic user interface (Flash triggered by XML)
Data binding (Flash forms and integrated components)
Value validation (Flash components)
Server-side:
Format conversion, structural or semantic validation and trigger generation (server-side script)
Storage (Relational DBMS and XML)
XML trees
XML triggers

Fig. 2. Multimedia and XML-driven web application model

The model is based on two connected XML parsers:

- the parser built-in the Flash player, which is able to pack XML trees and send them via HTTP, using a simplified DOM model manipulated with ActionScript, the Flash built-in scripting language;
- the server-side parser, provided by the server platform of choice (DOM or SAX), which is able to receive XML trees loaded with data from the user interface, impose vocabulary validation and generate trigger signals that control how the user interface behaves and what it displays.

3.2 Details

The client module is a Flash movie which contains all user forms and interface, which are fully loaded in the browser when the application is initiated. Flash files are optimized for HTTP transfer while strong local processing takes place in the browser plug-in. This permits a reasonable load time with respect to the bandwidth capabilities of nowadays. The internal structure of the Flash movie follows the timeline authoring principles: even if the movie has a linear nature (along the timeline), it is populated with certain key frames which provide the user interface. Navigation between key frames is controlled by:

- user navigation decision, with navigation buttons or movie clips;
- server-generated XML signals which trigger the behavior of the playhead (when and where it jumps between keyframes).

While user navigation within a Flash interface tends to be common knowledge in multimedia-oriented web development, navigation based on XML signals is provided by ActionScript and the Flash parser, which is able to implement a *sendAndReceive* method that emulates traditional event handling. The main difference is that events occurring on the client-side are handled on the server-side. The signal is small XML content produced in the server internal memory, delivered to the client internal memory and finally destroyed or stored in an event log. Actually, a single element with several status attributes would be sufficient to describe a diversity of possible redirecting or rejection states. The status attributes would serve as indicators for the movie playhead regarding the next keyframe to be displayed. This mechanism is a bandwidth saver, since XML signals are short strings rather than complex generated HTML code, the case of traditional applications. However, the client transfers more complex data than name=value pairs over HTTP. This reflects more relations within data, nesting being more complex than simple enumeration. Thus, the server receives a data structure less raw than the traditional query string and may apply vocabulary validation rather than simple value validation.

The server module consists of several scripts responsible with the generation of signals and the conversion of data structures between the client and the data storage. XML trees transferred on HTTP are received by the server parser of choice and the key data points (such as user password) are checked against stored values in order to decide what happens to data and what kind of response the user should get. After the decision is made, the script generates an XML element (the signal) and sets its status attributes to describe the error or to indicate which client key frame should be displayed as a response. Further on, the listening client object should be prepared to understand all status cases possible and move the playhead accordingly, to inform the user. A detailed example of this mechanism is presented in [4].

Besides the signals, the server script should also provide data required for display (product details, account details etc.). Thus, on the storage side, the server scripts have to provide storing and querying procedures. Even if relational databases are mature and widely accepted when it comes about answering queries with massive sets of data, they don't provide the needed interoperability. That is why the server module has to be able to operate an XML repository, modeled upon a context-specific vocabulary.

3.3 Data Modeling

The way data is modeled from the traditional structures (name=value pairs, relational tables, query strings) to XML has a great impact on performance, both for the parsing process and for the data storage management. An XML data model is provided by a vocabulary, which is the general frame in which all data and its nesting relations must fit. When such a vocabulary is available, it has to be shared with potential consumers of the XML documents. The common solutions for building vocabularies are:

- DTD (document type definition) which is generally accepted by all software but doesn't provide content typing and can be used if the vocabulary rules only state how data is nested and its occurrence or cardinality [8,9];
- XML Schema provides more complex rules, with strong content typing and is considered to be a replacement for DTD. However, XML Schema is not generally adopted yet [8,9].

The available data models for XML content are: element-only, void elements (EMPTY) with attributes, text-only content (used as name=value pairs), mixed content (used for text markup) and vague (ANY content, difficult to process). Considering several quality parameters (readability, document size, relational compatibility, value constraints, DOM parsing effort, SAX parsing effort) it is easy to demonstrate that the optimal data model is the one using void elements with data points stored in attributes [1,9]. This also applies to the Flash parser, which implements a slow node constructor when compared to navigating attribute arrays.

On the other hand there are several frequent errors that may occur when modeling XML content:

- **the presentation model should not be inherited** by the XML model - an essential quality of XML is the clear separation between presentation and content: content must provide the base for presentation generation and not vice versa;
- **formatting details and calculated data should not be stored** with data - an XML transformation document will be used at presentation-time, similarly to the generation of database reports;
- the length and structure of element/attribute **names** should be minimized if the potential consumer is an automated process and should have improved readability (at the cost of file size) if the consumer is human;
- **the scope of the vocabulary** should fit the potential use of documents - this is given by chosen document granularity: one document could encapsulate data destined for one or several presentation forms; of course, granularity affects performance with respect to the expected processing: small atomic documents are easier to handle but difficult to aggregate, while large documents are concurrently handled but the extraction of aggregated data is easy.

Modeling query strings or presentation data do not raise specific complications, but converting existing relational structures need to be approached with a more careful methodology. Even if tables are usually modeled as elements while records provide content and attributes, there are certain problems regarding relationships and key fields. There are two ways of reflecting relationships in XML:

- **simple containment**, by nesting one ore more associated records in each element to which they are related;
- **cross references** between related elements, provided by special pairs of attributes (reference=id, similar to HTML hyperlink definitions).

Each method has its advantages and disadvantages:

- containment is easy to navigate but creates **redundancy** (a record-content will be repeated for all the table-elements it is related to);
- cross-references eliminate redundancy but are more **difficult to navigate**, especially if the reference occurs after the referred id (multiple parsing may be necessary for such references).

That is why an optimal balance must be set between redundancy and navigation impediments, by following a number of steps/rules:

- Not all relational content must be converted to XML - the potential consumer application must be established, with its specific input needs: a CRM might not be interested in the same data as an accounting or a product management application;
- Not all XML content must reflect relational content - certain data may describe documents or define document granularity (by stating the scope), usually as attributes of the root element, which has to be designed as an envelope for its content;
- Each content table[2] is converted to an empty element and each field to an attribute, except for the foreign keys;
- Each element must be uniquely identified with an ID attribute - this is not necessarily the primary key, since different tables modeled in the same XML document might have the same primary key value and primary key values do not generally fit the ID attribute value constraints;
- The detail tables[3] and their relationships to content tables are not converted to elements, but to attributes with enumerated values, placed within the content table elements;
- The primary tables[4] are identified together with all its relationships, and converted to elements of appropriate cardinality, within the root element;
- A graph is built with the directions in which relationships will be most probably navigated by queries, starting with the primary tables;
- The tables descending from a single arc should be modeled by containment, with cardinality given by the type of relationship;
- The tables descending from multiple arcs should be contained once, in the root element, and referred by their ascendant elements (in the graph[5]);
- The bidirectional arcs will be modeled with bidirectional references;
- All non-referred ID attributes should be removed in the end.

3.4 Optimization

A common practice for complex Flash movies is to create a preamble (based on a Loader object) that presents the loading progress bar to the user if the estimated loading time for the client module is in danger to temporarily block functionality. Intense XML node parsing might have this effect, another reason to adopt the attribute-based model which provides greater performance in all the commonly used parsers.

Maybe the most important optimization opportunity comes with the component architecture provided by Macromedia in order to further support data connectivity within Flash movies. A robust set of classes and components covers the three layers of the architecture, from the back-end to the front-end of the client, according to [10]:

- **Data Connectivity** provides two connectors responsible for transferring data between the client module and an external producer or consumer of XML (the server module). The Web service connector is able to feed or consume WSDL operations transferred with SOAP, while the more generic XML connector provides ways for mapping (with XPath) sent or received data in XML structures and also coordinates POST transfers via HTTP [3]. The components can be easily configured for several types of transfer, mapping and value validation then, programatically, they can be managed by a set of classes and events handlers

[2] Working definition: A content table suffers frequent updates and queries, its number of records is unstable and its data is frequently queried for intense processing.

[3] Working definition: A detail table is updated non-frequently, its number of records is stable and its content is formed of ID=description pairs used to lookup details on records from the content tables

[4] Working definition: The primary table is a table from which relationships will be navigated by the consumer, in order to reach the rest of data

[5] Note that the descent relations in the graph will not reflect the parent-child descent.

which detail the predictable types of errors occurring during the transfer;

- **Data Management** is the layer responsible with managing data within the client module, just before it is delivered to a connector, or right after it's been received. Usually, it serves as a client cache and filtering mechanism between the user forms that gather data and the connectors that have to work ready-to-transfer data points. Two types of components implement this layer:
  - Cache and filtering components (a data set and a data holder) which transform or just hold data for various purposes;
  - Resolvers (an XML resolver and a relational resolver) which convert data with respect to its predictable destination (an XML consumer/repository or a relational database)
- **Data Binding** is the layer responsible for binding values of a user form/interface to attributes of components from the other layers (management or connectivity). Because default form fields use native strings, this layer provides several validation filters such as mask, format or type of the data that has to be further processed or was received from other processes.

4 Conclusion

XML and Flash open great opportunities for multimedia applications producing and consuming XML and e-commerce or e-marketing may be the first fields to benefit from such opportunities even if, for the present, traditional systems found in exploitation are preferable to the experimental ones, at least from the common business point of view.

The future may focus interest on a specialized XML transfer protocol to replace HTTP and to implement a robust XML gate between client and server. Even if the current needs are fulfilled by transferring XML strings via HTTP, some services, such as vocabulary validation could take place on the protocol level, since both protocols and vocabularies are ways of agreeing on a common set of rules for sharing data (and its structure or meaning, with XML). An XML transfer protocol would clearly separate "pluggable" client-oriented modules (such as those provided by Flash) from complex back-end environments (such as an on-line ERP or a knowledge base).

Another interesting, conclusive issue is that hierarchical data structures, once considered obsolete, are returning in the general focus as a complement to the relational model, which is unable to meet interoperability requirements but can be easily connected to a data medium such as the one provided by XML technologies.

References:

[1] 1. Amiano Mitch et al., *XML Problem-Design-Solution*, Wrox, 2006

[2] 2. Buchmann Robert, *Conceperea, proiectarea si realizarea afacerilor pe Internet*, Ed. Risoprint, 2004

[3] 3. Buchmann Robert, Mocean Loredan, *Xml Connectivity within Flash Presentations*, in *Information And Knowledge Age - Proceedings Of The 7th International Conference On E-Informatics - Bucharest*, 2005, pp. 1075-1081

[4] 4. Buchmann Robert, *On-line Authentication and User Registration with ActionScript 2.0, XML and ASP*, in *E-COMM LINE Conference Preprint - Bucharest*, 2004, pp. 269-274

[5] 5. Buchmann Robert, *The Internet Problem - Homogeneous or Heterogeneous*, in *Digital Economy – Proceedings of the 6th international conference on e-informatics - Bucharest*, 2003, pp. 65-68

[6] 6. Harold Elliotte, *Xml Bible*, Idg Books, 1999

[7] 7. Swann Craig, Caines Greg, *Xml in Flash*, Ed. Teora, 2002

[8] 8. Walmsley Priscilla, *Definitive XML Schema*, Prentice Hall, 2002

[9] 9. Williams Kevin et al., *Professional XML Databases*, Wrox, 2000

[10] 10. ***, *Macromedia Flash 2004 Professional Documentation*

[11] 11. ***, *Web resources:*
- http://www.xmlinflash.com
- http://www.xml101.com
- http://www.flash-db.com
- http://www.macromedia.com