On XML-driven Rich Web Clients[1]

Lect. BUCHMANN ROBERT ANDREI Ph.D.
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: The Rich Web Client paradigm is definitely changing the way Web applications are built and designed. This paper promotes a particular rich client module based on Flash and XML, previously included as an e-commerce user interface within the application model presented in the series of papers [2],[3], [4]. Even if rich clients have been proposed since the nineties, the real need for them raised as usage patterns change from transient usage (intermittent user interaction) to sovereign usage (continuous user interaction).

Key-Words: - Rich client, XML in Flash, Bandwidth, User experience, Connectivity

1 Introduction

The term Rich Client refers two requirements of modern Web applications: to provide a client module that is rich. The *client* aspect is inherent in the Web environment and the *rich* aspect is a necessity for the user interaction model. Traditional user interfaces are clearly separated into rich non-client GUIs and thin Web clients. Rich GUIs are the implementation of the presentation tier but usually they provide access to local data storage. Thin Web clients are the classic dynamically generated presentations and documents provided by any Web site. The rich web client tries to accomplish the interaction of a rich GUI in a client-server environment.

2 Problem formulation

As new requirements are stated in the context of user experience and usability, the current trend is to close the gap between user interfaces which access remote and local resources. This involves:

- strong local processing on the client side;
- event-based data exchange between the client and the server;
- interoperable modules that communicate in an interoperable manner.

Local processing implies that a full client module is initially downloaded in order to set-up the environment. Event-based data exchange means that transfers are controlled by events triggered in the user interface or by the client-server interaction and states.. Interoperability can be imposed by using portable data structures using XML or an XML-based vocabulary (schema, DTD) as a data medium. Several solutions and frameworks for this approach have already been successful and the model proposed here was developed and described in parallel with Ajax, as a Flash-oriented rich client. It was initially used to improve e-commerce application usability. The essential usability requirement was defined in the context of the e-activities where applications address social needs of user groups with the least experience and understanding regarding computer applications. The user experience within the e-commerce interaction model was defined in [5] as one of the determinants for the social digital divide. It is essential to promote an e-commerce application as nothing more complicated than any domestic technology such as a phone, TV etc. The rich client needs to set a balance between the concepts of thin client (client reduced to presentation layer, based on processing and data on the server) and thick client (client application that stores data remotely) so that neither the initial loading nor the connection load affects user experience.

3 Problem solution

One of the most successful rich client solutions, Ajax is based on several coupling technologies: HTML for structuring the presentation, CSS for layout and formatting, DOM as the object model, JavaScript as the client-side programming language and XMLHttpRequest as the data exchange solution.

[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

The proposed model uses Flash for layout and presentation, ActionScript as the programming language based on a model similar to DOM, the XML Connector component as the data exchange solution and the WebService Connector component as a Web Service integrator over SOAP. The key components of a modern rich client are:

- Entities: XHTML elements in Ajax, User Interface Components in the proposed model;
- Events: triggers set by the user or by server responses, handled by JavaScript in Ajax. In the proposed model the triggers control the timeline and are usually delivered by a server module as XML elements with status attributes or by user interactions;
- Attributes: properties of elements in Ajax, or properties of objects in Action Script;
- Styles: layout properties based on CSS in Ajax, handled by the Flash design tools in the proposed model.

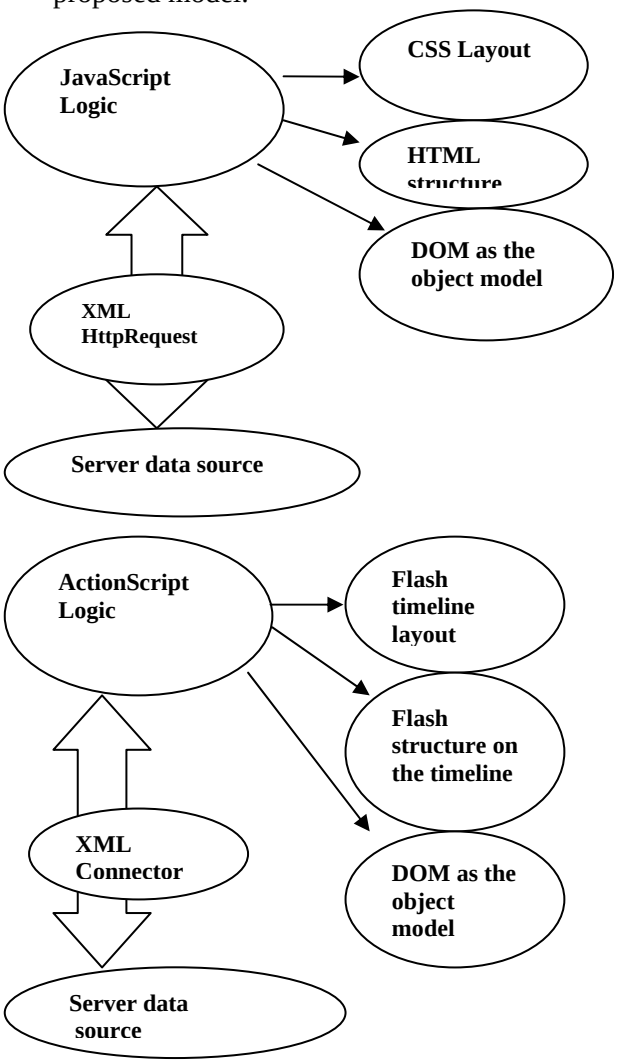

Fig.1. The general structure of the proposed rich client compared to an Ajax client

The Flash module moves an important part of the business logic on the client-side, compared to the classic Web applications. Data is passed along a series of connections from the data source to the user interface, with most of these connections imposing local processing and filtering on the data. This is accomplished by the Component Architecture model provided by Flash, based on three layers:

- **The connectivity layer** builds an XML structure in the client module and binds it to the elements of the XML or WSDL Connector, two components which map the XML or Web Service input from the external data source;
- **The data management layer** holds objects and data during the client-side processing. The XUpdate Resolver component binds and converts local object attributes to XML packets delivered to or received from the connectivity layer;
- **The data binding layer** binds user interface components from the Flash movie timeline to attributes from the other layers. Thus, the proposed rich client is able to present data both synchronously and asynchronously.

2.1. Advantages of the proposed model

Table 1 checks essential features where the proposed model is superior to Ajax:

Feature	Flash rich client	Ajax rich client
Audio / Video	Embedded or dynamically loaded	Needs plug-ins
Versions	Minor variations between recent player versions	Major variations between recent browser versions. Needs degrading code
Server communication	Communicates with most server scripts in various ways. May use FlashRemote.	Uses IFrame or XMLHTTPRequest
XML support	Full	Not natively in JavaScript
Graphics	Full vector and raster support	Dynamically loads static images

Table 1

Cross-browser ubiquity

The proposed model inherits the advantage of Flash over JavaScript and CSS implementation regarding

the cross-browser experience. Also, browser interaction limits Ajax on various non-desktop devices, where Flash is supported by multiple form factors and on-line devices. On the programming level, ActionScript works in a fully functional homogeneous framework, while JavaScript tries to integrate a heterogeneous environment.

User interface

The first reason to promote any kind of rich client is the user experience which is closer to traditional graphic user interfaces, with fast reactions to user interaction that do not involve remote function calls or content delivery from the server. Flash brings a robust model to replace JavaScript and CSS with timeline authoring. The client module is built as an interactive animation which is dynamic both in space and time. The timeline is covered by jumps between keyframes, which are triggered by the user interaction and by server XML signals. Preloaded progress bar components are fit to signal network latency and update delays.

The Flash presentation brings certain advantages against the XHTML-based forms:

- Validation, calculation, error messages and other frequent client-side processing is possible by creating binding custom objects on the data binding layer instead of several script functions mingled with the presentation structure. This permits a design pattern in which frequent operations are reusable;
- XHTML forms cannot be loaded from a separate document and the prefill requires dynamically generated pages. In the proposed model, the initial download of the rich client is able to feed the browser with several XML documents related to the user, ready to be used when triggered by events. A common example of asynchronous use of background client data would be a search suggestion or recommendation system.
- XHTML forms support only flat and unstructured data, as name-value pairs. The Flash client is able to build run-time XML documents and objects directly from the user interaction, use their data or structure to trigger events along the timeline or send it to the server for permanent storage;
- XHTML forms provide a very limited set of controls compared to the Flash forms toolkit.

Bandwidth usage

By doing most of the processing within the browser, and only using client scripting (ActionScript parsing a DOM variety) that consume server XML data instead of constantly receiving content from the server, the payload coming down is much smaller in size after a short period of usage.

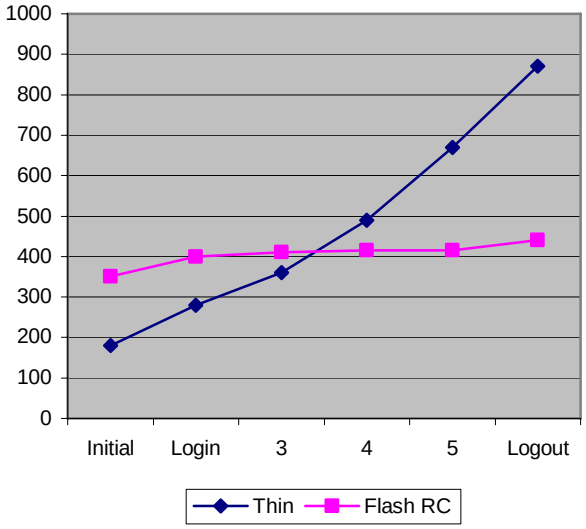

Fig.2. The cumulative download for a thin client and for the proposed rich client

Fig.2 shows the variation of download size between the initial request and the logout, for an e-shop with basic 6-steps functionality and bare design (no banners and strictly functional elements). The cumulative variation greatly affects user interaction, as most of the download happens while the site is navigated. In the proposed rich client, the initial download precedes the user experience and consequent downloads involve only XML/WSDL Connector requests and server responses as XML signals that trigger the timeline behavior or XML/Web service feeds. One of the factors involved in the great variation for the thin client is that in the classic approach content is often duplicated in subsequent requests, causing redundant connection load.

A clear separation between storage and processing with the possibility of temporary off-line interaction.

The proposed model promotes the modularized approach in building pluggable application components.

- A separation between the raw data and delivered content:

- A separation between the client processing and the server-side module which is not responsible anymore with generating dynamic content. Instead, it is reduced to the role of a permanent storage feeding client procedures with parameters in the form of XML;
- A separation between the functional layers of the client (connectivity, data management and user interface binding);
- A separation between art design and interface coding.

Due to the clear separation between the client and server module of a rich client Web application, the client module is pluggable to any server-side code able to provide and consume XML. PHP would be a common choice for generating server-side XML from a database query or directly from an XML repository.

Client authentication, card number, shopping cart content and other private data can be secured over the network by encoding it with an action script that implements the MD5 algorithm.

Web services and XML connectivity

The connectivity layer provides a rich set of visual tools for exchanging XML with the server over HTTP POST or WSDL over SOAP, besides the Ajax solution of loading XMLDocument objects into the client by event handlers. This greatly improves the design effort and inner modularity of the client.

The native support for SOAP and XML-RPC web services is not affected by the web service platform, with the exception of .NET DataSets which are not supported. Also, REST web services are not natively supported but are easily parsed on the server-side, since they respond with valid XML. Both RPC-style (with wrapper element) and document-style web services are supported

The consumption of web services over SOAP or XML-RPC is possible with specific APIs. The connectivity components support only single port SOAP and single service WSDL, but using the APIs these limitations can be overcome.

On the other hand, Flash Remoting offers an alternative with speed advantages due to using the binary ActionMessage Format instead of XML. Also, Flash Remoting does not need a cross-domain policy on the service provider server, which is a general security specification for Flash. However, the cross-domain policy can be avoided by using a server object as a proxy that communicates with the provider and delivers the service results from within the application domain.

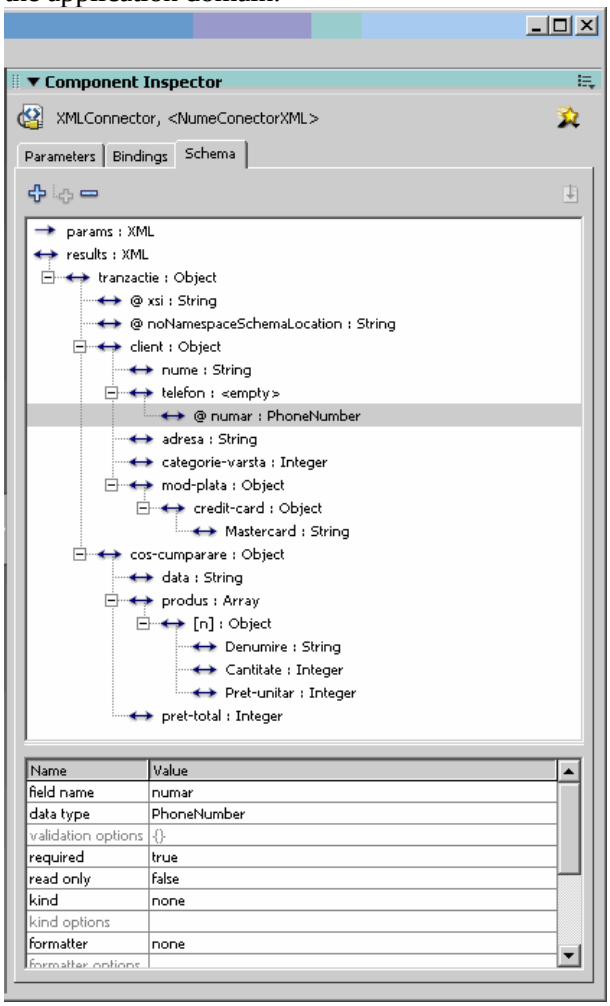

Fig.3.The Component Inspector for binding an XML Connector

4 Conclusion

Since Ajax and its frameworks emerged, the rich clients are trying to close the gap between the traditional user experience and the Web user interface. The rich client paradigm is also supported by the general trend of processing strength and memory storage evolving faster than the remote communication capabilities. Thus, the proposed rich client unloads the connection and its effort to support remote calls and content delivery, by an initial download followed by data exchange in an interoperable manner such as XML. The model is essentially linked to the sovereign usage pattern which tend to replace traditional transient usage patterns of Web applications.

The present focuses interest in a real competition for rich client solutions, a competition where Java applets seem to be obsolete, with a trend in defining XML vocabularies oriented on user-interface and widget markup: Mozilla's XUL, Microsoft's

XAML. Other emerging proposals are *Aflax*, that tries to integrate Ajax libraries with Flash or *OpenLaszlo*, the rich client platform that provides a user interface-oriented XML language and a servlet that compiles this language for targeted environments.
However, due to cross-browser inconsistencies, it is fairly plausible that Ajax is just an intermediate hack solution for text-heavy sites, before their transition towards mature XML-driven hypermedia based on ubiquitous plug-ins.

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] 5. Crane Dave et al., *Ajax in action*, Manning, 2006
[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
- http://www.aflax.org
- http://www.openlaszlo.org