A client-intensive, model-based approach to web application development: The *AtariX* system

Oscar Díaz,Felipe Ibañez, Mikel Larrañaga, Jon Iturrioz
The EKIN team, Dpto. de Lenguajes y Sistemas Informáticos
University of the Basque Country
jipdigao@si.ehu.es

July 19, 2001

1 The issue

Web application development is currently suffering from a severe bottleneck as the gap between available implementation tools and application's requirements is enlarging. There is tremendous pressure on developers to "code-and-publish". And the background of developers can be quite diverse with typically no experience in software engineering. They are guided by the features in the tools and language constructs. This free-form style of development can lead "to use ad-hoc, hacker-type approaches, which lack rigor, systematic techniques, sound methodologies, and quality assurance" [3]. And these practices can be disastrous as web masters have to face maintenance. This situation is specially stressful in the area of e-commerce. In today´s e-commerce world, companies should adapt to changing conditions and rapid evolution. Some practitioners have reported that "most web sites today change their appearance at least once a year to stay attractive. Minor changes in the look and feel of a site several times a year are common, and mayor modifications occur frequently"[4]. However, it is a frustrating experience to see how often the web site bottleneck slows and restricts the evolution of the organization the site is supposedly serving.

An indication of this need is the recent upheaval around "web engineering", a term coined to stress the need for a systematic approach to web application construction [8]. Within this area, distinct projects have been launched which aim at providing design guidelines and supporting tools for systematic web site construction [2]. Among the distinct stages of the application lifecycle, our work focuses on design and architectural issues.

Design-wise, one of the most frequently cited guideline is separation of concerns by using a model-based approach [5]. This approach aims to find declarative models, preferably orthogonal, that allow designers to declaratively specify a specific concern of the application without being immediately immersed in details of implementations. A design is then conformed by a set of schemata (i.e. model instances) which describe distinct aspects of the application. Declarativeness and orthogonality accounts for maintenability: the separation of concerns and their declarative specification allow to easily update a schema while minimizing the impact on the rest of the application. Moreover, this separation of concerns addresses the multidisciplinary nature of web applications. Web development is a mixture between print publishing and software development, between marketing and computing, between art and technology [3]. This setting calls for knowledge and expertise from many different disciplines and requires heterogeneous teams. A model-based approach eases workgroup by keeping separate each concern: aesthetic experts can focus on the presentation model, hypermedia and usability practitioners can address navigation concerns,

and task integration can be addressed by knowledgable engineers on EAI technologies.

Task integration is also becoming an increasing concern among practitioners [6][7]. Besides the challenges posed by supporting each task, our contention is that task integration is paramount to enhance site usability. Since a site normally offers a large set of tasks, it is fundamental to struggle for the site to appear to function as a single whole. For the user, the web site should offer a *workview* rather than a set of loosely couple set of services. This idea of workview strives to provide a sense of coherence and unity among the distinct tasks that can be accomplished through the site.

As for the architecture, web applications commonly follow a thin browser approach [1]. Here, all of the application control resides on the server, and the browser is just used for rendering purposes. There is little control of the client's configuration. Most e-commerce applications use this architecture as it does not make good business sense to eliminate any sector of customers just because they do not have sufficient client capabilities. Data-intensive applications are also good candidates for exhibiting this architecture. In these applications, data is not brought to the client but processed at the server, near where the data lies, and just the final outcome, the HTML page, is sent to the browser. On the other hand, a browser request will be required every time a new page is displayed. This could lead to an increase in network traffic, low site promptness and site bottlenecks. By contrast, a thick browser architecture attempts to mitigate this problem at the price of increasing the demands on the browser configuration. Both intranet and B2B applications are good candidates for this architecture.

2 Our approach

OUr work faces the above design and architectural issues by following a document-centric, client-intensive and model-based approach. **"Document-centric"** implies that a document (in the sense of an XML document) becomes the unit of delivery. The web implementation model is based on the notion of *resource* that supports mostly a self-contained chunk of information. An HTML page, a ".gif" file or a XML document are examples of resources which are aggregated into web sites by means of HTML links. The HTML page becomes both the unit of delivery and the unit of presentation. This work dissociates these two units. The HTML page keeps on being the unit of presentation but the *document* becomes the unit of delivery. Instead of looking at a web site as a set of pages, this view conceives a web site as an inter-related set of documents. A document is brought to the browser, and once there, HTML pages are generated to render distinct parts of the document as needed.

"Document-centricness" is intimately related with **client-intensive** architectures. This feature allows a whole bulk of pages to be generated from one document with a single connection to the server. This can account for a sensible reduction on the network traffic as connections to the server are limited to requests for another document, invoking services or connecting to DBMS for extracting data. Hence, our approach moves to the browser part of the processing currently loaded in the server. In particular, workflow dependencies among tasks, content rendering and navigation control are mainly achieved at the browser. It should be noticed that this architecture increases the demands on the browser configuration. In particular, the browser should be able to analyse and process XML documents. However, the wide spread use and support received by XML makes us feel confident about the feasibility of this approach. Both Internet Explorer and Netscape are moving in this direction.

Finally, the **"model-based"** feature tackles the separate and declarative description of the distinct concerns risen during application design. Each concern is described in a

separate document: how data is integrated and described (the *content* document), the topology of links (the *navigation* document), the layout of each element (the *presentation* document), trace configuration requirements (the *trace* document) and, task specification and integration (the *workview* document). It is not surprising that XML is the underlying technology. Model constructs are introduced through an XML vocabulary, i.e. a name space defined by means of XML-Schema. Thus, a model is realized through an XML document which is validated against this model's XML-Schema.

This approach has been borne out by *AtariX*, a model-based system for the specification and support of web applications using a thick web client architecture. The system requires Internet Explorer 5.0 with the Microsoft XML parser 3.0.

References

[1] Jim Conallen. *Building Web Applications with UML.* Addison-Wesley, 2000.

[2] P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: a Survey. *ACM Computer Surveys*, 31(3):227–263, 1999.

[3] A. Ginige and S. Murugesan. Web engineering: An introduction. *IEEE MultiMedia*, pages 15–18, January - March 2001.

[4] E. Kirda, M. Jazayeri, and C. Kerer. Experiences in engineering flexible web services. *IEEE MultiMedia*, pages 58–65, January - March 2001.

[5] F. Paterno and C. Mancini. Model-Based design of interactive applications. *ACM Intelligence*, pages 27–37, Winter 2000.

[6] W. Rajput. *E-Commerce Systems Architecture and Applications.* Artech House Publishers, 2000.

[7] R. Kalakota M. Robinson. *e-Business: Roadmap for Success.* Addison-Wesley, 1999.

[8] WebEngineering.org. Webengineering.org community homepage at http://www.webengineering.org, 2001.