Capturing RIA Concepts in a Web Modeling Language

Alessandro Bozzon
bozzon@elet.polimi.it

Sara Comai
comai@elet.polimi.it

Piero Fraternali
fraterna@elet.polimi.it

Giovanni Toffetti Carughi
toffetti@elet.polimi.it

Dipartimento di Elettronica e Informazione
Politecnico di Milano
P.zza L. da Vinci, I-32 20133 Milano, Italy

ABSTRACT

This work addresses conceptual modeling and automatic code generation for Rich Internet Applications, a variant of Web-based systems bridging the gap between desktop and Web interfaces. The approach we propose is a first step towards a full integration of RIA paradigms into the Web development process, enabling the specification of complex Web solutions mixing HTTP+HTML and Rich Internet Applications, using a single modeling language and tool.

Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Design Tools and Techniques, computer-aided software engineering (CASE), evolutionary prototyping, user interfaces; H.4.0 [**Information Systems Applications**]: general; H.5.4 [**Information Interfaces and Presentation**]: Hypertext / Hypermedia architectures, navigation, theory, user issues.

General Terms: Design.

Keywords: Rich Internet Applications, Web Engineering, Web Site Design.

1. INTRODUCTION

In the last years the Web has become the reference platform for the development of a wide variety of integrated business solutions. Due to the increasing complexity of these applications, current Web technologies are starting to show usability and interactivity limits. Rich Internet Applications (RIAs) have been recently proposed as the response to such drawbacks [2]. They are a variant of Web-based systems providing sophisticated interfaces for representing complex processes and data, minimizing client-server data transfers and moving the interaction and presentation layers from the server to the client. Typically, a RIA is loaded by the client along with some initial data; then, it manages data rendering and event processing, communicating with the server when the user requires further information or must submit data. So far RIAs lack development methodologies and CASE tools catering for their specificity, notwithstanding the number of existing methodologies and tools for Web and Hypermedia development [3], which allow one to specify the application at an abstract level and derive the implementation code (semi-) automatically.
In this work we consider a conceptual Web modeling language (WebML [1], www.webml.org) and extend it with the aim of reducing the gap between Web development methodologies and the RIA paradigm, leveraging the common features of RIAs and traditional Web applications. The extended conceptual model has been implemented in a CASE tool capable of automatically producing a running RIA.

WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-332-9/06/0005.

2. CONCEPTUAL MODELING FOR RIAS

The typical layers of a Web conceptual model include data, hypertext in the large, and hypertext in the small modeling. In the following paragraphs we briefly introduce and discuss the extensions to the main ingredients of a Web conceptual model needed to support RIA features.

Data model. While in traditional data-intensive Web applications content resides solely at the server-side (in the form of database tuples or as user session-related main memory objects), in RIAs content can also reside in the client, as main memory objects with the same visibility and duration of the client application, or, in some technologies, as persistent client-side objects. Data are therefore characterized by two different dimensions: the architectural tier of existence, which can be the *server* or the *client*, and the level of persistence, which can be *permanent* or *temporary*. In WebML, where the data model is represented by Entity-Relationship diagrams, we stereotype entities and relationships with their persistence level. Figure 1 depicts a well-formed data schema. For example, we stereotype as *database* the data permanently stored in a server-side data management system (e.g., a relational or XML database); as *client* the data that are temporarily stored at the client side, for the duration of the application run. A data schema extended with these two persistence levels is well-formed if the following constraint holds: relationships with database persistence connect entities with database persistence only(i.e. persistent relationships cannot connect temporary entities).

Hypertext model in the large. Hypertext modeling in the large specifies the general structure of the front-end: it organizes the hypertext taking into account the different classes of users, and structures it into pages, possibly clustered into areas having a specific purpose, and possibly organized in a hierarchy composed of nested pages. From the technological standpoint RIAs have a different physical

structure than traditional data-intensive Web applications: the former typically consist of a single application "shell" (e.g., a Java applet or a FLASH movie), which loads different data and components based on the user's interaction. The latter consist of multiple independent templates, processed by the server and simply rendered by the client. However, the hypertext modeling metaphor remains a good description of the dynamics of the interface also for RIAs, especially in the case of hybrid applications, which comprise a mix of traditional page templates and RIA components.

To cope with the specificity of RIAs, where pages, or fragments thereof, can be executed either at the server-side or at the client-side, the notion of page in WebML has been extended, by stereotyping it as: 1) *Server page*: it represents a traditional Web page; content and presentation are calculated by the server, whereas rendering and event detection are handled by the client. Events triggering some business logic (not bound to the presentation layer) are processed at server-side.2) *Client page*: it represents a page incorporating content or logics managed (at least in part) by the client. Its content can be computed at the server or client side, whereas presentation, rendering and event handling occur at the client side. Events can be processed locally at the client or dispatched to the server.

Hypertext model in the small. Hypertext modeling in the small refines the coarse model of the application with details about the *content of pages*, the links for user's interaction, and the *operations* triggered by the user.

The **content** of pages in WebML is represented with a visual notation as a graph of *content units* connected by links. Figure 2 depict an example of a RIA hypertext model. Links express both parameter passing, needed for computing the data of parametric units, and user interaction, needed for triggering page (re)computation. In traditional Web applications, content unit processing occurs on the server: data is extracted from a *database entity*, logical conditions (called *selector conditions* in WebML) can be specified to filter the entity instances, and *ordering clauses* specify how they have to be sorted. In RIAs, *computation is distributed between the server and the client*, according to the page type: units contained in a server page are computed by the server (*server units*), like in traditional Web applications, and units contained in a client page are managed by the client (*client units*). The WebML content unit is extended with the possibility of specifying the source entity, the selector conditions and ordering clauses either on the server or on the client. A unit is well-formed if the following constraints hold: a) server units cannot be specified on a client entity and cannot comprise client-side selectors and ordering clauses; b) a client unit that draws content from a client entity, cannot contain server-side selector conditions or ordering clauses. These constraints ensure that all the computations performed by the server rely only on data and operations computable at the server-side and thus cope with the asymmetric nature of the Web, where the client calls the server and not vice versa.

WebML **operations** model arbitrary business logic and predefined content updates (creating, deleting or modifying entities, connecting or disconnecting pairs of entity instances belonging to a relation). In the RIA context, operations can be executed by the client or by the server, as captured by the following definitions: 1) *server operation*: a piece of business logic or data update executed by the server; 2) *client operation*: a piece of business logic executed by the client or an update on a client-side entity or relationship; 3) *operation chain*: a sequence of operations, possibly mixing client and server operations.

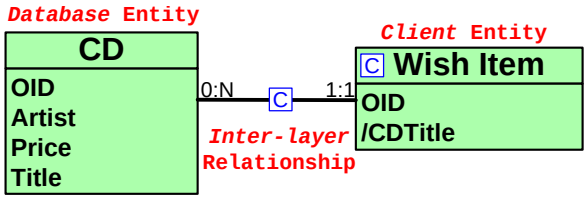

Figure 1: Example of RIA Data model

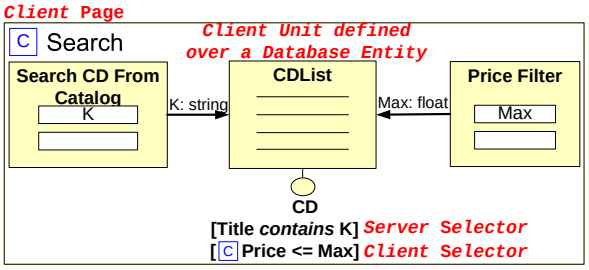

Figure 2: Example of RIA Hypertext Model

In order to fit the RIA paradigm, operation composition constraints and a new semantic of page computation have also been defined.

3. IMPLEMENTATION

The RIA modelling primitives discussed above have been implemented in WebRatio (`www.webratio.com`), a CASE tool for the visual specification and the automatic code generation for Web applications. WebRatio adopts an MVC-based organization and maps the various concepts of WebML (pages, content units, and operations) into the components of the MVC 2 architecture. We adopted a similar design for generating the client-side code, exploiting Laszlo LZX (`www.openlaszlo.org`) as the implementation technology. A client-side controller coded in LZX is responsible of handling events and managing the computation of client-side content units and operations. Each content unit is mapped into: (1) a view component for rendering, (2) a model component for data management, business logic, and server communication, (3) possibly a service on the server-side for data query and result formatting in XML. Operations are implemented like content units, without view components. Presentation is also generated automatically, by incorporating into the view component of units the look&feel mockups manually coded in LZX. **A complete example** of an application developed with the abovementioned approach is presented on the full-size poster.

4. REFERENCES

[1] S. Ceri and P. Fraternali and M. Brambilla and A. Bongio and S. Comai and M. Matera , "Designing Data-Intensive Web Applications", *Morgan Kaufmann*, 2002.

[2] Joshua Duhl, "White paper: Rich Internet Applications", *IDC*, 2003.

[3] P.Fraternali, "Tools and Approaches for Developing Data-Intensive Web Applications: A Survey", *ACM Comput. Surv. Volume 31 Number 3*, 1999.