The History of WebML
Lessons Learned from 10 Years of Model-Driven Development of Web Applications

Stefano Ceri, Marco Brambilla, and Piero Fraternali

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. Da Vinci, 32. I20133 Milano, Italy
{ceri,mbrambil,fraterna}@elet.polimi.it

Abstract. This work presents a retrospective analysis on the conceptual modeling language for Web applications called WebML, which was first defined about 10 years ago. WebML has been an incubator for research on conceptual modeling, exploiting existing experiences in the field and continuously addressing new challenges concerning abstractions, methods, tools, and technologies. People working on WebML are spread among universities, technology transfer centres, and a spin-off. In this paper, we illustrate the history of WebML, we summarize the essence of the approach, and we sketch the main research branches that spawned from the initial proposal. We describe how new trends in research, application development, methodology, and tool prototyping led to the continuous growth of the modeling language.

1 Introduction

Data-intensive Web applications, i.e., software systems whose main purpose is to give access to well-organized content, represented the first industrial application of the Web, and are still predominant in terms of volume and commercial value. All companies have an institutional site showing their business and describing their offers, and many enterprises manage the relations with their customers through the Web. Therefore, these applications have been a preferred target of development methods and tools, which have been available for a long time.

Among them, the Web Modelling Language (WebML) [1] was defined, about 10 years ago, as a conceptual model for data-intensive Web applications. In the early days of Web development, technologies were immature and in perpetual change; as a reaction, WebML was conceived as a high level, implementation-independent conceptual model, and the associated design support environment, called WebRatio [9], has always been platform-independent, so as to adapt to frequent technological changes.

While other conceptual models focus more on the early phases of the development process (i.e., requirement specification [21]), WebML concentrates on the later phases, starting from design, down to the implementation. As many other conceptual modeling languages [14], WebML is based upon the principle of separation of

A.T. Borgida et al. (Eds.): Mylopoulos Festschrift, LNCS 5600, pp. 273–292, 2009.

concerns: content, interface logics, and presentation logics are defined as separate models of the application. The main innovation in WebML comes from the hypertext modelling notation (patented in 2003), which enables the specification of Web pages consisting of conceptual components (units) interconnected by conceptual links. The hypertext model is drawn in a simple and quite intuitive visual notation, but has a rigorous semantics, which allows the automatic transformation of diagrams into the complete running code of a data-intensive Web application. Originally, the focus of the design of WebML concentrated on the definition of a powerful set of units; with time, we realized that units are just specific components, which can be defined and adapted to the needs of any new technological development; instead, the essence of the WebML hypertext model lies in the rules for assembling components and links into a graph, and for inferring all the possible parameter passing rules from the component interfaces and the link types. A well-formed graph guarantees the correct data flow among units and dictates the proper component execution order when computing the content of pages. Ultimately, computing a hypertext model amounts to enacting a workflow of component execution driven by the user's "clicking behaviour". In retrospective, the choices of link and component semantics were quite adequate to the purpose and remained stable throughout ten years of language evolution.

While the Web has gone through waves of innovation, new technological scenarios have developed, and revolutionary concepts – such as enabling the interaction of software programs rather than only humans – have emerged. Several new challenges have been addressed within the WebML context, including:

- Web services and service-oriented architectures [11];
- Integration with business processes [5];
- Personalization, adaptation, context awareness, and mobility [6];
- Semantic Web and Semantic Web Services [4];
- Rich Internet Applications [3];
- Search-based applications;
- Support of reuse, multi-threading, and modularization.

This paper highlights the core nucleus of the WebML language, which has remained stable over the years, and illustrates how we have dealt with each new challenge through a four-step approach. The treatment of each extension is necessarily concise and visual, for more details we refer readers to published papers and reports.

2 The Original WebML Language

The specification of a Web application in WebML [2] consists of **a set of orthogonal models**: the application *data model* (an extended Entity-Relationship model), one or more *hypertext models* (i.e., different site views for different types of users), expressing the navigation paths and the page composition; and the *presentation model*, describing the visual aspect of the pages. We focus on the hypertext model, as the data model is not innovative; the presentation model is also quite interesting, as it enables "dressing" a hypertext model to obtain Web pages with the desired layout and look&feel for any rendition technology, but is also outside the scope of this paper.

2.1 The WebML Hypertext Model

A hypertext model consists of one or more *site views*, each of them targeted to a specific user role or client device. A site view is a collection of pages (possibly grouped into *areas* for modularization purposes); the content of pages is expressed by components for data publishing (called *content units*); the business logic triggered by the user's interaction is instead represented by sequences of *operation units*, which denote components for modifying data or for performing arbitrary business actions (e.g., sending email). Content and operations units are connected by *links*, which specify the data flow between them and the process flow for computing page content and for enacting the business logic, in reaction to user's generated navigation events.

Consider for instance a simple scenario: users browse a Home Page, from where they can navigate to a page showing an index of loan products. After choosing one loan, users are lead to a page with the loan details and the list of proposals for the chosen loan. The WebML specification for the described hypertext is depicted in Figure 1. The Home Page contains only some static content, which is not modeled. A link from this page leads to the Loans page, containing an index of all loans, graphically represented by means of an *index unit* labeled Loans Index. When the user selects a loan from the index, he is taken to the Chosen Loan page, showing the loan details. In this page, a *data unit*, labeled Loan Details, displays the attributes of the loan (e.g. the company, the total amount and the rate), and is linked to another index unit, labeled Proposals Index, which displays the plan options.

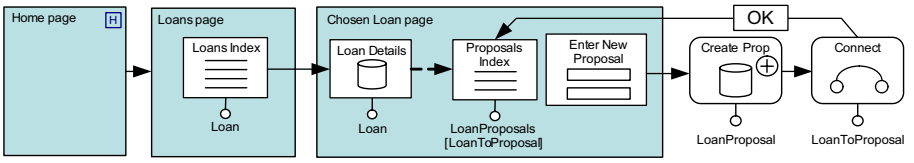

Fig. 1. A WebML hypertext for browsing and updating information

This example contains **units for publishing content** (data and index units), which display some of the attributes of one or more instances of a given entity. Syntactically, each type of unit has a distinguished icon and the entity name is specified at the bottom of the unit; below the entity name, predicates (called *selectors*) express conditions filtering the entity instances to be shown. The example of Figure 1 also shows static content units, which display fixed content not coming from the objects in the data model: this is the case of the Enter New Proposal *entry unit*, which denotes a form for data entry. The hypertext model also illustrates the use of operation units: the outgoing link of the Enter New Proposal entry unit activates a sequence of two operation units: a create and a connect unit, which respectively create an instance of the LoanProposal entity and connect it with a relationship instance to the Loan entity.

WebML distinguishes between normal, transport, and automatic links. **Normal links** (denoted by solid arrows) enable navigation and are rendered as hypertext anchors or form buttons, while **transport links** (denoted by dashed arrows) enable only parameter passing and are not rendered as navigable widgets. **Automatic links**

(denoted by an [A] icon) are normal links, which are automatically "navigated" by the system on page load. Orthogonally, links can be classified as contextual or non-contextual: **contextual links** transfer data between units, whereas **non-contextual links** enable navigation between pages, with no associated parameters. Operation units also demand two other types of links: **OK links** and **KO links**, respectively denoting the course of action taken after success or failure in the execution of the operation. In the example of Figure 1:

- The link from the Home page to the Loans page is non-contextual, since it carries no information, and simply enables a change of page.
- The link from the Loans Index unit to the Loan Details unit is normal and contextual, as it transports the ID of the loan chosen in the index unit and displayed in the data unit.
- The link from the Loan Details data unit to the Proposals Index unit is a transport link: when the user enters the Chosen Loan page, the Loan Details unit is displayed and, at the same time, the Loan ID is transferred to the Proposal Index unit, so that the content of the Proposals index unit is computed and displayed without user's intervention. No navigable anchor is rendered, because there is no need of the user's interaction.
- The outgoing link of the Connect unit, labelled OK, denotes that after the successful execution of the operation the Choose Loan page is displayed.

The content of a unit depends on its input links and local selectors. For instance, the Loan ID is used to select those proposals associated with a given loan by the relationship role LoanToProposal; this selection is expressed by the selector condition [LoanToProposal] below the unit's entity. In general, arbitrary logical conditions can be used, but conjunctive expressions are easily presented in the diagrams, where each conjunct is a predicate over an entity's attribute or relationship role.

2.2 Semantics of the WebML Hypertext Model

As already mentioned, WebML is associated with a page computation algorithm deriving from the formal definition of the model's semantics (based on statecharts in [10]). The essential point is the **page computation algorithm**, which describes how the content of the page is determined after a navigation event produced by the user. Page computation amounts to the progressive evaluation of the various units of a page, starting from input parameters associated with the navigation of a link. This process implies the orderly propagation of the value of link parameters, from an initial set of units, whose content is computable when the page is accessed, to other units, which expect input from automatic or transport links exiting from the already computed units of the page.

In WebML, pages are the fundamental unit of computation. A WebML page may contain multiple units linked to each other to form a complex graph, and may be accessed by means of several different links, originating from other pages, from a unit inside the page itself, or from an operation activated from the same page or from another page. The content of a page must be computed or recomputed in the following cases:

- When the page is entered through a link (contextual or non-contextual) originating in another page; in this case the contents of all units of the page are calculated afresh, based on the possible parameter values carried by the link.
- When the user navigates an intra-page link and thus supplies some new input to the destination unit of the link; in this case, part of the content of the page is calculated based on the parameter values associated with the intra-page link, but part of the content of the page is computed based on the values of parameters existing prior to the navigation of the intra-page link, so that past user's choices are not lost when navigating the link.
- When an operation is invoked, ending with a link pointing back to the same page: this case is similar to the navigation of an intra-page link, but in addition the operation may have side effects on the content visualized in the page, which may change the content displayed by the page.

The example in Figure 2 illustrates the three cases:

- When the ArtistIndex page is accessed through the non-contextual link labeled Link1 or the Artist page is accessed through the contextual link labeled Link2, the content of the entire destination page is computed afresh, taking into account the possible input values associated with the navigated link (e.g., the OID of the selected artist when Link 2 is navigated).
- When the user selects a new album from the AlbumIndex unit, new context information flows along the link labeled Link3 and determines the album to be displayed in the AlbumData unit; at the same time, the Artist displayed in the ArtistData data unit must be "remembered" and redisplayed, because the input of the ArtistData unit is not directly affected by the navigation of the intra-page link.
- When the delete operation is performed successfully and the page is re-entered through the OK link Link4, the content of the ArtistData unit is preserved, so to remember the past user's choice, whereas the content of the AlbumIndex unit and of the AlbumData unit is refreshed, so that the deleted album no longer appears in the AlbumIndex unit and in the AlbumData unit. If the delete operation fails, the KO link Link5 is followed and the content of the AlbumData unit is refreshed using the OID of the object that could not be deleted, and the content of the other units is restored. This ensures that the previously selected artist, his/her albums, and the details of the album tentatively deleted continue to be displayed when the page is re-accessed after the failed operation.

The page computation process is triggered by any of the previously discussed navigational events (inter-page link navigation, intra-page link navigation, operation activation). Based on the navigated link, a set of parameter values is collected and passed in input to the page, which determines the initial input for some of the page units. The page computation algorithm starts by tagging as computable all context-free units (e.g., units with no input parameters, like the ArtistIndex unit) and possibly

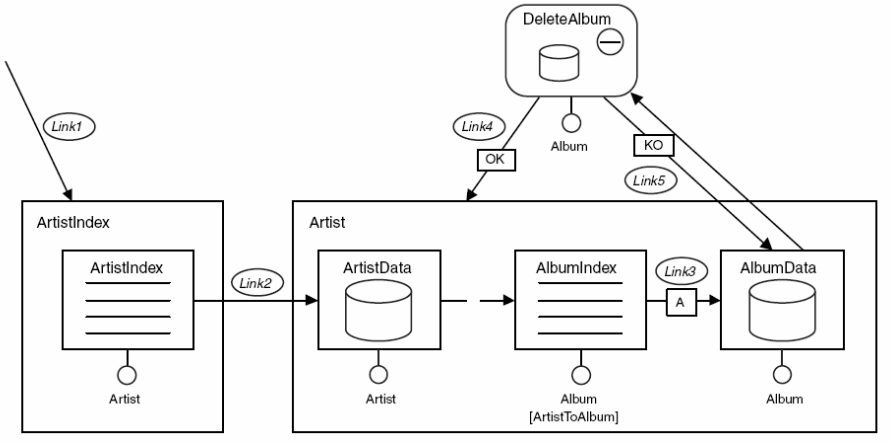

Fig. 2. Example of WebML Page with different access paths

the externally dependent units for which there are sufficient input values in the parameters passed to the page (e.g., the ArtistData unit when Link2 is navigated). Then, the computation proceeds by evaluating the units one after another, until all possible units have been evaluated. The computation process exploits the propagation of context along automatic and transport links, and a **specificity rule** telling which alternative input should be considered when multiple choices are available for evaluating the same unit.

The specificity rule introduces a partial order in parameter passing, therefore a page computation is nondeterministic (but the WebRatio tool identifies such situations and prompt designers to change the model to eliminate non-determinism). Moreover, some hypertexts can be non-computable, due to circular dependencies among units or pages, causing a deadlock in the assignment of input values to some units. The complete description of the WebML hypertext model semantics is in Chapter 5 of [8] and in [10].

2.3 The WebML Design Process

The WebML methodology exploits **a formal design process**, explained in Chapter 6 of [8], shown in Figure 3. The process includes the classic phases of requirement analysis (thoroughly addressed by dedicated methods, such as [21]), data design, hypertext design, and presentation design, followed by architecture design and implementation. The 4-step procedure, from requirements to data, to hypertext, to presentation design, can be iterated multiple times with the support of WebRatio, which acts as a rapid prototyping tool; experience has shown that a crucial advantage of using the model-driven approach comes from the ability to generate incremental prototypes directly under the eyes of the application stakeholders. Web-specific data design guidelines, based on the notion of the **web mart** as a standard ER schema that is recurrent in Web applications, have also been proposed in [9].

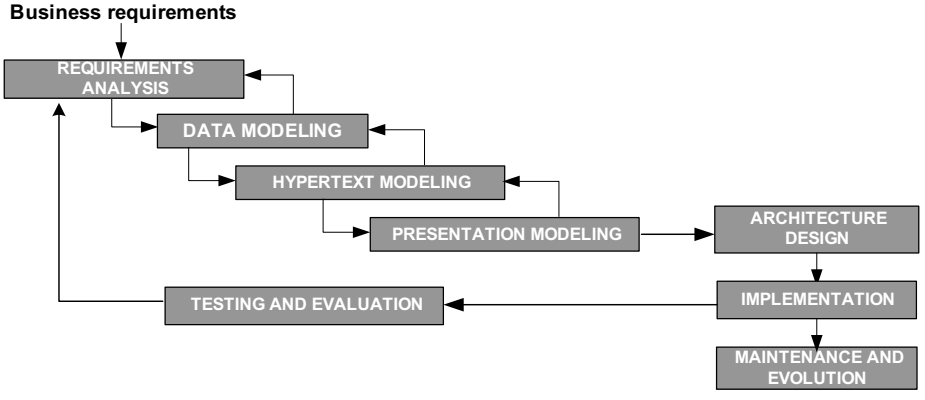

Fig. 3. The WebML Development process

2.4 The Added Value of WebML

Before describing the extensions of the original WebML model, we wish to distill the concepts that proved most valuable in the ten-year experience of development and research with WebML. Our experience demonstrated that the true added value of WebML stands in the following aspects:

- The choice of **component-based design** as the fundamental development paradigm and the **standardization of component specification**, which allow extending the language without altering its semantics.
- The use of **links of different types**, for specifying the "wiring" of components, which can be assembled into pages and sequences of operations.
- Powerful and automatic **inference rules for parameter matching**, allowing the designer to avoid explicit specification of the data carried by contextual links in all cases in which they can be deduced from the context.

The result of these design principles is an easy-to-learn formalism: the hypertext model consists of very few concepts (e.g., compared to UML): site views, areas, pages, content units, operation units, and links. At the same time, the openness of the implementation allows developers to enrich WebRatio with the components of their choice, so to achieve an almost unlimited variety of effects.

The subsequent evolution of the model built upon these aspects, adding domain-specific features to the core nucleus of the language. In retrospective, we have addressed every new challenge by using a common approach, which indeed has become evident to us during the course of time, and now is well understood and constitutes the base for all new additions. For every new research direction, four different kinds of extensions are designed, respectively addressing the development process, the content model, the hypertext meta-model, and the tool framework:

- *Extensions of the development process* capture the new steps of the design that are needed to address the new direction, providing as well the methodological guidelines and best practices for helping designers.
- *Extensions of the content model* express domain-specific standard data schemas, e.g., collections of entities and relationships, that characterize the applications in

the area of interest; the standard schema is connected with the application data model, to enable an integrated use of domain objects and special-purpose data.

- *Extension of the hypertext meta-model* refine the standard WebML concepts to capture the new abstractions required for addressing the new modelling perspective; in this way, the core semantics of WebML is preserved, but functionality is added by plugging in "libraries" of specialized concepts.
- *Extensions of the tool framework* introduce new modules in the open architecture of WebRatio (specialized data elements, new content and operation units, novel containers of model elements, etc.), implement wizards for expressing the semantics of new components in terms of existing ones, and provide code generation and runtime support for each new addition.

3 Service-Oriented Architectures

The first WebML extension discussed in this paper is towards the Service Oriented Architectures [11]. Our extension includes four components:

- The extension to the development process and the definition of some methodological guidelines for SOA design;
- A standard data schema for representing the services and the business processes to be performed;
- New WebML units for covering Web service specification and invocation, together with primitives for enforcing business process constraints;
- The support of the specified solutions through a process modeller, a translator of processes into hypertext skeletons, and an XML-to-XML mapping tool.

The extension of the **development process** to SOA requires separating application design from service design; the former addresses the front-end of a Web integration application targeted to the user, while the latter focuses on provisioning well-designed services, usable across different Web applications.

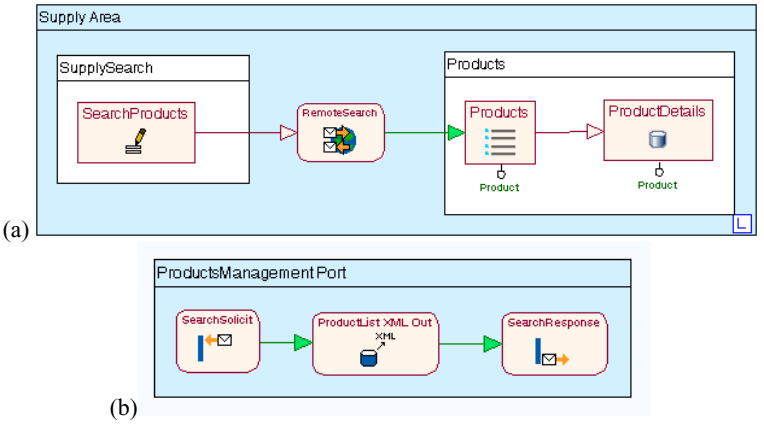

Fig. 4. Example of WebML hypertext model with invocation of remote service

The **contet model** for SOAs (not shown here for sake of brevity) supports the description of Web Services according to WSDL, including the notions of services, ports, and input/output messages.

Extensions to the **hypertext model** cover both Web Service publication and Web Service consumption. **Web Service publication** is expressed as a novel container (called *Service View*), which is analogous to a site view, but contains specifications of services instead of pages. A service specification is denoted by a *Port*, which is a container of the operations triggered upon invocation of the service.

Service invocation and **reaction to messages** are supported by specialized components, called **Web Service units**. These primitives correspond to the WSDL classes of Web service operations and comprise:

- **Web service publication primitives**: *Solicit unit* (representing the end-point of a Web service), and *Response unit* (providing the response at the end of a Web service implementation); they are used in a service view as part of the specification of the computation performed by a Web Service.
- **Web Service invocation primitives**: *Request-response* and *Request* units; they are used in site views, and denote the invocation of remote Web Services from the front-end of a web application.

For instance, Figure 4 shows a hypertext that specifies a front-end for invoking a web Service (Figure 4a) and the specification of the web Service within a port container (Figure 4b).

In the *Supply Area* of Figure 4a, the user can access the *SupplySearch* page, in which the *SearchProducts* entry unit enables the input of search keywords. The submission of the form, denoted by the navigation of the outgoing link of the entry unit, triggers a request-response operation (*RemoteSearch)*, which builds the XML input requested by the service and collects the XML response returned by it. From the service response, a set of instances of the *Product* entity are created, and displayed to the user by means of the *Products* index unit in the *Products* page; the user may continue browsing, e.g., by choosing one of the displayed products and looking at its details.

Figure 4b represents the service view that publishes the *RemoteSearch* service invoked by the previously described hypertext. The *ProductManagementPort* contains the chain of operations that make up the service: the sequence starts with the *SearchSolicit* unit, which denotes the reception of the message. Upon the arrival of the message, an XML-out operation extracts from the service provider's database the list of desired products and formats it as an XML document. The service terminates with the *SearchResponse* unit, which returns the response message to the invoker[1].

For supporting service design, WebRatio has been extended with:

- The novel Web service units.
- The *Service view* and *Port* containers.

[1] Service ports are an example of a WebML concept that has nothing to do with the user's interaction, which shows how the original target of the model (hypertext navigation) has been generalized to cover new requirements. Even more radical shifts will be needed to deal with semantic Web Services, as illustrated in the sequel.

- The runtime and code generator features necessary to produce the actual executable code corresponding to the additional modeling primitives.

4 Workflow-Driven Applications for the Web

The Web has become a popular implementation platform for B2B applications, whose goal is not only the navigation of content, but also the enactment of intra- and inter-organization business processes. Web-based B2B applications exhibit much more sophisticated interaction patterns than traditional Web applications: they back a structured process, consisting of activities governed by execution constraints, serving different user roles, whose joint work must be coordinated. They may be distributed across different processor nodes, due to organizational constraints, design opportunity, or existence of legacy systems to be reused. WebML has been extended to cover the requirements of this class of applications [5], by:

- The integration in the development life-cycle of workflow-specific deign guidelines;
- Two different models for representing the business processes;
- New design primitives (namely, WebML units) for enforcing business process constraints;
- New tools for workflow-driven application design: a process modeller and a translator of processes into hypertext skeletons.

The incorporation of business processes in WebML has been pursued in two distinct, yet complementary, scenarios:

1) S*tatic business process*, i.e., processes defined once during the design phase and then preserved for the entire application lifetime.
2) D*ynamic business processes*, in which the process schema is subject to continuous evolution.

Some aspects are common to the two scenarios, while others differ significantly. In the following, we will highlight the differences, when needed.

The WebML design process is extended with a new phase, **business process modeling**, preceding data and hypertext design. In case of *dynamic BP*, changes to the process are allowed at runtime too, and the system automatically adapts its behavior.

A **content model for static processes,** shown in Figure 5, represents meta-data about the business processes. A process is represented by the *Process* entity, associated with the *ActivityType* entity, representing the kinds of activities that can be executed in the process. An instance of a process is modeled by the *Case* entity, related to its *Process* (via the *InstanceOf* relationship) and to its activities (via the *PartOf* relationship); entity *ActivityInstance* denotes the actual instances of activities within cases.

With **dynamic processes**, the basic content model is completed by a few additional entities and relationships that represent the sequence constraints between activities, the branching and joining points, and the execution conditions. Thanks to the more refined meta-data, the application can infer the process structure and execution status at runtime and adapt to dynamic changes of the process schema.

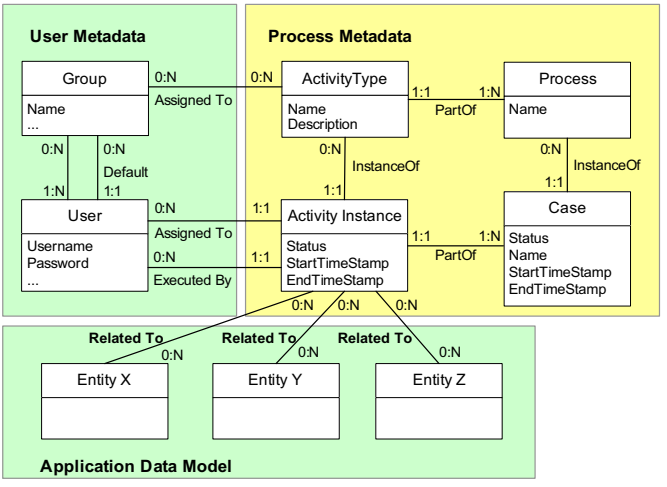

Fig. 5. Content model for the specification of a business process

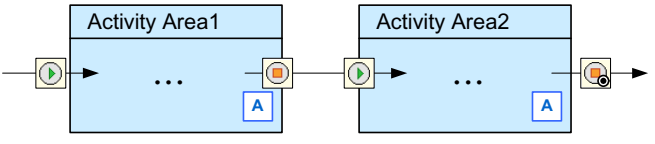

Fig. 6. Two activity areas and the *start* and *end* links that denote the initiation and termination of an activity

In case of static processes, the process structure is embodied in the topology of the hypertext. The intuition is that the process progresses as the actors navigate the front-end, provided that the hypertext model and the process metadata are kept in synch. To this end, new primitives are added to the hypertext model, for specifying activity boundaries (namely *activity areas*) and process-dependent navigation (namely *workflow links*). Figure 6 shows some of these primitives: "Activity Areas" denote groups of pages that implement the front-end for executing an activity; specialized links represent the workflow-related side effects of navigation: starting, ending, suspending, and resuming activities. Distributed processes deployed on *SOAs* can be obtained by combining the workflow and Web Services primitives [5].

For *dynamic business processes*, the next activity to be executed is not statically specified by an activity area, but is determined at runtime by a unit (called *Next unit*), which encapsulates the process control logic. It exploits the information stored in the process meta-data and log to calculate the current process status and the enabled state transitions. It is associated with the current ActivityInstance, and needs the following input parameters: caseID (the currently executed process instance ID), activityID (the activity instance ID that has just terminated), and the conditionParameters (the values required by the conditions to be evaluated). The Next unit finds all the process constraints related to the specified activity instance, evaluates them according to the

defined precedence constraints (i.e., sequence, AND-join, etc.), and, if the conditions hold, enables the execution of the subsequent activities in the workflow. If the activities are automatic, they are immediately started. If they involve human choice, the application model consists of the site view for the user to choose when to start the activity. An example of Next unit can be found in Section 8, dealing with Search-based Web applications.

For supporting the design of workflow-driven Web applications, several **tool extensions** have been prototyped and are currently being ported to the commercial version of WebRatio:

- A workflow modeling editor for specifying business processes in the BPMN notation.
- Model transformations that translate a business process model into a skeleton of WebML hypertext model.
- The abovementioned operation units and special-purpose links for implementing the static and dynamic workflow enactment.

5 User Personalization and Context Awareness

WebML has been also applied to the design of adaptive, context-aware Web applications, i.e. applications which exploit the context and adapt their behaviour to usage conditions and user's preferences [6].

In these applications, the design process is extended by a preliminary step dedicated to the **modeling of the user profiles** and of the **contextual information**.

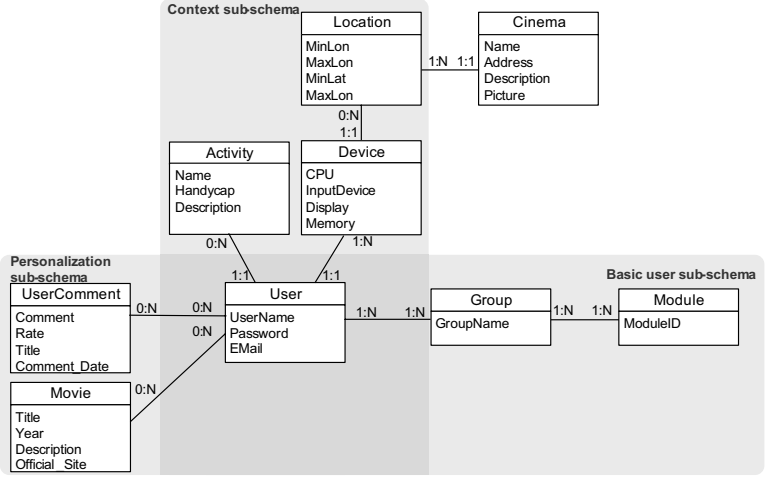

Fig. 7. Three models representing user, personalization, and context data

User and context requirements are described by means of three different models, complementing the application data (see Figure 7):

- The **user model** describes data about users and their access rights to the domain objects. In particular, entity *User* expresses a basic user profile, entity *Group* enables access rights for groups of users, and entity *Module* allows users and groups to be selectively granted access to any hypertext element (site views, pages, individual content units, and even links).
- The **personalization model** associates application entities with the *User* entity by means of relationships denoting user preferences or ownership. For example, the relationship between the entities *User* and *UserComment* in Figure 7 enables the identification of the comments s/he has posted, and the relationship between the entities *User* and *Movie* represents the preferences of the user for specific movies.
- The **context model** includes entities such as *Device*, *Location* and *Activity*, which describe context properties relevant to adaptivity. Context entities are connected to the *User* entity, to associate each user with her/his (personal) context.

During hypertext design, context-awareness can be associated with selected pages, and not necessarily with the whole application. **Location-aware pages** are tagged with a *C-label* (standing for "Context-aware") to distinguish them from conventional pages. Adaptivity actions are clustered within a *context cloud* which must be executed prior to the computation of the page. Clouds typically includes WebML operations that read the personalization or context data and then customize the page content or modify the navigation flow defined in the model.

A prototype extension of WebRatio generates pages with adaptive business logic; such a prototype has been used in some applications but has not been included yet into the commercial version.

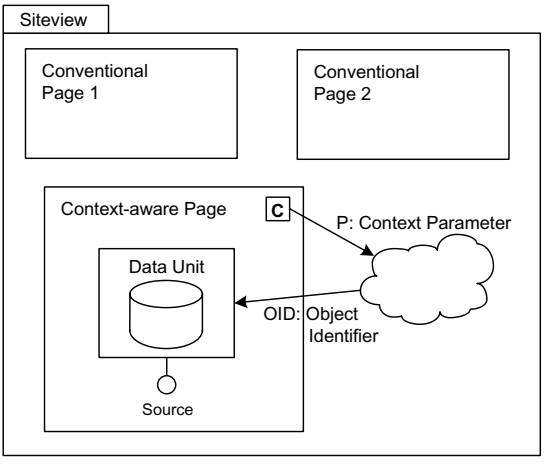

Fig. 8. Model with a context-aware page, labelled with a "C" and associated with a "context cloud"

6 Semantic Web Services

Traditionally, the service requestor and service provider are designed jointly and then tightly bound together when an application is created. The emerging field of Semantic Web Services (SWS) [26] provides paradigms for semantically enriching the existing syntactic descriptions of Web services; then, the service requestor can search, either at design or at run time, among a variety of Web-enabled service providers, by choosing the service that best fits the requestor's requirements. Such a flexible binding of requestor and providers allows for dynamic and evolving applications to be created, utilizing automatic resource discovery, selection, mediation and invocation.

We extended WebML in [4] so as to generate, on top of conventional models (of: processes, data, services, and interfaces), a large portion of the semantic descriptions required by the SWS in a semi-automatic manner, thus integrating the production and maintenance of semantic information into the application generation cycle.

To address the new SWS requirements, we defined a **process for semantic service design** by extending the SOA design process with two additional tasks:

- Ontology Importing, for reusing existing ontologies that may be exploited for describing the domain of the Web application under development.
- Semantic Annotation, for specifying how the hypertext pages or services can be annotated using existing ontological knowledge.

At the conceptual level, the **content model** for Semantic Web applications addresses the integration of existing third-party ontologies in the conceptual data model. At the logical level, imported ontological data can be either copied into an application-specific implementation of the E-R model (typically a relational database) or maintained in remote semantic repository and queried on demand.

The basic WebML primitives have been extended with components for **ontology querying and navigation**, exploiting the expressive power of ontological languages (inspired by SPARQL and RDF-S). These units allow queries on classes, instances, properties, and values; checking the existence of specific concepts; and verifying whether a relationship holds between two resources. Further units import content from an ontology and return the RDF description of a given portion of the ontological model. Operations such as lifting and lowering have been introduced too, by extending the XML2XML mapping components already developed in the context of SOAs. These units, together with the standard WebML primitives and the SOA extensions, allow designers to specify new kinds of applications. For instance, it is possible to define WSMO mediators [4], as demonstrated in the context of the SWS Challenge.

The SWS primitives have been implemented in two versions: when ontological data are maintained in an external repository, the implementation exploits ontological query languages; when ontological data are integrated within an internal relational source, the implementation is directly mapped to such source.

The WebRatio development tool has been extended with prototypical **automatic generators of WSMO-compliant descriptions** (goals, choreographies, capabilities, and mediators) from the models already available in WebML, i.e., business processes, content models, and service models. The automatically generated annotations cannot

express the full semantics of services and applications, but they provide an initial skeleton, which can be completed manually.

7 Rich Internet Applications

Due to the increasingly complex requirements of 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; they are a variant of Web-based systems minimizing client-server data transfers and moving the interaction and presentation layers from the server to the client. 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 even, in some technologies, as persistent client-side objects. Also, in RIAs more powerful communication patterns are possible, like server-to-client message push and asynchronous event processing. WebML has been extended 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 [3].

The design process is extended by defining the **allocation to the client or server side** of data elements (entities and relationships) and hypertext components (pages, content and operation units), and by establishing the relevant **client-server communication patterns** (consisting of policies for event notification, recipient filtering, and synchronous/asynchronous event processing).

In the **content model**, concepts are therefore characterized by two different dimensions: their **location**, which can be the server or the client, and their **duration**, which can be persistent or temporary. For example, in Figure 9 the *Wish Lists* entity is tagged as *client (C)* and temporary (unfilled icon) to denote that the data are temporarily stored at the client side, for the duration of the application run.

Similarly, the notion of page in WebML has been extended, by adding **client pages,** which incorporate content or logics managed (at least in part) by the client; their content can be computed at the server or client side, whereas presentation, rendering and event handling occur at the client side. The events generated by the user's interaction can be processed locally at the client or dispatched to the server. Event handling operations are also introduce (send event and receive event) which enable the expression of flexible communication patterns, including real-time

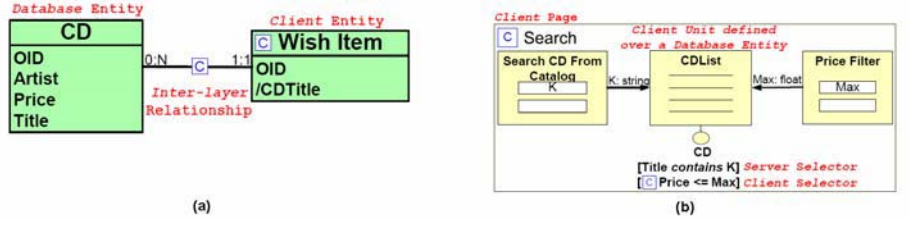

Fig. 9. Example of RIA-enabled WebML data (a) and hypertext model (b)

collaboration, server push, and asynchronous event processing. Classical WebML content units are also extended with the possibility of specifying that the source entity, the selector conditions, or ordering clauses be managed either on the server or on the client. Figure 9 shows a client page which contains an index unit with the population fetched from the server, but filtered using a predicate (price<=max) computed on the client. In order to fit the more flexible way in which RIAs handle the content of pages, the semantics of page computation has also been revised.

The RIA modelling primitives have been implemented in WebRatio through a prototypical **code generator for the client-side pages**, exploiting an open source RIA platform (www.openlaszlo.org) for 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. A subset of the prototyped features, including AJAX behaviours and client-side event management, is already available in the commercial version of WebRatio.

8 Related Work

Web Application Modeling. Several methodologies and notations address conceptual modeling of Web applications [31]. Among more recent projects, WebML is closer to those based on conceptual methodologies like W2000 [16] and OO-HMETHOD [29] (based on UML interaction diagrams), Araneus [35, 36], Strudel [27] and OOHDM [40]. The WAE UML extension by Conallen [23] focuses mainly on implementation and architectural issues of Web application design. Commercial vendors are proposing tools for Web development, however most of them have only adapted to the Web environment modeling concepts borrowed from other fields. Among them, Oracle JDeveloper 10g [38], Code Charge Studio [22], Rational Rapid Developer [39], and ArcStyler [13], which also features business process to Web model translation and direct implementation.

Business Processes. Several existing platforms and languages allow integrating the design of Web applications and business processes. Among the existing models, we can mention Araneus [36], that has been extended with a workflow conceptual model, allowing the interaction between the hypertext and an underlying workflow management system. The Process Modeling Language (PML) [37], a lightweight formalism similar to BPMN that can be automatically compiled into a simple Web-based application, starting from imperative programming-style syntax. Among the Web design proposals, OO-H and UWE have specifically addressed the integration of process and navigation modeling. The authors of OO-H and UWE propose a joint approach [34] to the integration of process and navigation modeling. In particular, both methodologies converge in the requirements analysis phase, where UML Use Case, Class, and Activity Diagrams are exploited to capture the requirements, and then, the methods slightly diverge in the design phase. In OOHDM [40], the content and navigation models are extended with activity entities and activity nodes respectively, represented by UML primitives. In WSDM [25], the process design is driven by the user requirements and is based on the ConcurTaskTrees notation.

Web Services. A flurry of activity is currently taking place in the field of Web service description [44]. Several XML languages for orchestration and choreography of services have been proposed (e.g., BPEL4WS [19]). WebML is expressive enough to capture any BPEL4WS-style service composition pattern [5]. The ActiveXML system [15] manages XML documents including calls to services, but ignoring Web interfaces and complex processes.

Semantic Web. Several traditional Web design methodologies (like OOHDM [41]) and new approaches (like Hera [43]) are focusing on Semantic Web applications. MIDAS is a framework based on MDA for Semantic Web applications [12]. Research efforts are converging on the proposal of combining Semantic Web Services (SWS) and Business Process Management (BPM) to create one consolidated technology, called Semantic Business Process Management (SBPM) [33]. Our extensions to the Semantic Web Services benefit from the WSMO [26] framework for handling Semantic Web Services.

Adaptivity and context-awareness. Within the domain of theWeb, so-called adaptive hypermedia systems [20] address advanced adaptation and personalization mechanisms, and recent research efforts also address the special needs of mobileWeb applications and portable device characteristics. HyCon [32], for example, is a platform for the development of context-aware hypermedia systems with special emphasis on location-based services. AHA! [24] is a user modeling and adaptation tool originally developed in the e-learning domain. Other works [18] address the fast development of context-aware (Web) applications along a technological, database-driven approach, combining a universal context engine in combination with a suitable content management system [30]. On top of the Hera project, Fiala et al. [2004] propose implementation and deployment of component-based, adaptive Web presentations. [17] extend the previous approach by addressing the lack of dynamism.

RIAs. Some approaches address the complexity of RIAs through the exploitation of state models for interface design. Exploiting MDA life-cycle is a missing feature in the related work. Our approach is also different with respect to other recent proposals in the Web Engineering field to represent the RIA foundations (e.g. [42]), because we include a more abstract level of representation of states and events.

9 Conclusions

The “WebML approach” has acted as a framework for continuous innovation and exploration of new research directions. This is made possible by a unique combination of environmental conditions:

- Availability of well-defined conceptual models;
- Extensibility of the model thanks to a plug-in based structure;
- Availability of a CASE tool for fast prototyping of application and easy integration of new features and components;
- Formally defined development process for Web applications;
- Strong link between the research (mostly performed in university) and the technology transfer into an industrial-strenght product (performed within a spin-off);

- Interactions with real world requirements, enabled by interaction with customers of the spin-off.
- Participation to the international research community, through experience and people exchange and several EU-funded projects.

This mix of ingredients has allowed us to follow our own pathway to innovation in conceptual modeling.

Wide adoption of the approach has been secured thanks to low learning barrier to the newcomers and availability of suitable tool support. Basic modelling skills are usually taught in 6 hours of academic lessons, and the full training program for certified professionals requires a total of 8 days. On-the-field statistics estimate that 2 to 3 months are needed for enabling full productivity at industrial level. Once this is achieved, high efficiency is granted in the development process, thanks to automatic code generation that reaches 90% of the total software artifacts in typical industrial applications [2].

Adoption of WebML can be inferred from the spreading of the associated WebRatio toolsuite. The WebRatio company registered more than 27,000 downloads of the tool since its first release (among them, more than 7,500 of the last Eclipse-based release, WebRatio 5). 70 companies adopted the commercial version of the tool for medium-to-large development projects and 158 universities subscribed to the academic program, that currently involves more than 5,900 students and researchers worldwide.

Acknowledgements

We wish to thank all the people who work for developing WebML within the "Database and Web" group at Politecnico di Milano and the developers of the Web Models spin-off.

References – WebML

[1] Acerbis, R., Bongio, A., Brambilla, M., Butti, S., Ceri, S., Fraternali, P.: Web Applications Design and Development with WebML and WebRatio 5.0. TOOLS, pp. 392–411 (2008), http://www.webratio.com/

[2] Acerbis, R., Bongio, A., Brambilla, M., Tisi, M., Ceri, S., Tosetti, E.: Developing eBusiness solutions with a model driven approach: The case of acer EMEA. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607, pp. 539–544. Springer, Heidelberg (2007)

[3] Bozzon, A., Comai, S., Fraternali, P., Toffetti Carughi, G.: Conceptual Modeling and Code Generation for Rich Internet Applications. In: International Conference on Web Engineering, pp. 353–360. Springer, Heidelberg (2006)

[4] Brambilla, M., Celino, I., Ceri, S., Cerizza, D., Della Valle, E.: Model-Driven Design and Development of Semantic Web Service Applications. ACM TOIT 8(1) (2008)

[5] Brambilla, M., Ceri, S., Fraternali, P., Manolescu, I.: Process Modeling in Web Applications. ACM TOSEM 15(4) (2006)

[6] Ceri, S., Daniel, F., Matera, M., Facca, F.: Model-driven Development of Context-Aware Web Applications. ACM TOIT 7(1) (2007)

[7] Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a modeling language for designing Web sites. WWW9 / Computer Networks 33 (2000)
[8] Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, San Francisco (2002)
[9] Ceri, S., Matera, M., Rizzo, F., Demaldé, V.: Designing Data-Intensive Web Applications for Content Accessibility using Web Marts. Communications of ACM 50(4), 55–61 (2007)
[10] Comai, S., Fraternali, P.: A Semantic Model for Specifying Data-Intensive Web Applications Using WebML. In: Semantic Web Workshop, Stanford, USA (July 2001)
[11] Manolescu, I., Brambilla, M., Ceri, S., Comai, S., Fraternali, P.: Model-Driven Design and Deployment of Service-Enabled Web Applications. ACM TOIT 5(3) (2005)

References – Related Work

[12] Acuña, C.J., Marcos, E.: Modeling semantic web services: a case study. In: Proceedings of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, California, USA, pp. 32–39 (2006)
[13] ArcStyler, http://www.arcstyler.com
[14] Brodie, M., Mylopoulos, J., Schmidt, J. (eds.): On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages. Springer, Heidelberg (1984)
[15] Abiteboul, S., Bonifati, A., Cobéna, G., Manolescu, I., Milo, T.: Dynamic XML Documents with Distribution and Replication, SIGMOD (2003)
[16] Baresi, L., Garzotto, F., Paolini, P.: From Web Sites to Web Applications: New Issues for Conceptual Modeling. In: Mayr, H.C., Liddle, S.W., Thalheim, B. (eds.) ER Workshops 2000. LNCS, vol. 1921, pp. 89–100. Springer, Heidelberg (2000)
[17] Barna, P., Houben, G.-J., Frasincar, F.: Specification of Adaptive Behavior Using a General-Purpose Design Methodology for Dynamic Web Applications. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 283–286. Springer, Heidelberg (2004)
[18] Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M.C., Palinginis, A.: Interplay of content and context. In: Koch, N., Fraternali, P., Wirsing, M. (eds.) ICWE 2004. LNCS, vol. 3140, pp. 187–200. Springer, Heidelberg (2004)
[19] BPEL4WS: Business Process Execution Language for Web Services, http://www.ibm.com/developerworks/Webservices
[20] Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia. User Model and User-Adapted Interaction 6(2-3), 87–129
[21] Castro, J., Kolp, M., Mylopoulos, J.: A requirements-driven development methodology. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 108–123. Springer, Heidelberg (2001)
[22] Code Charge Studio 2.3, http://www.codecharge.com/studio
[23] Conallen, J.: Building Web Applications with UML, October 2002. Addison-Wesley, Reading (2002)
[24] De Bra, P., Houben, G.-J., Wu, H.: AHAM: a Dexter-based Reference Model for Adaptive Hypermedia. In: HYPERTEXT 1999: Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots, pp. 147–156 (1999)
[25] De Troyer, O., Casteleyn, S.: Modeling Complex Processes for Web Applications using WSDM. In: Third International Workshop on Web Oriented Software Technology, Oviedo 2003, pp. 1–12 (2003)

[26] Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services – The Web Service Modeling Ontology. Springer, Heidelberg (2006)
[27] Fernandez, M.F., Florescu, D., Kang, J., Levy, A.Y., Suciu, D.: Catching the Boat with Strudel: Experiences with a Web-Site Management System. In: SIGMOD 1998, pp. 414–425 (1998)
[28] Fiala, Z., Hinz, M., Houben, G.-J., Frasincar, F.: Design and Implementation of Component-based Adaptive Web Presentations. In: ACM SAC, pp. 1698–1704
[29] Gómez, J., Cachero, C., Pastor, O.: Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia 8(2), 26–39 (2001)
[30] Grossniklaus, M., Norrie, M.C.: Information Concepts for Content Management. In: WISE Workshops, pp. 150–159
[31] Fraternali, P.: Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. ACM Computing Surveys 31(3), 227–263 (1999)
[32] Hansen, F.A., Bouvin, N.O., Christensen, B.G., Grønbæk, K., Pedersen, T.B., Gagach, J.: Integrating the Web and the World: Contextual Trails on the Move. In: Proc. of ACM-Hypertext 2004, pp. 98–107 (2004)
[33] Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In: Proceedings of the IEEE ICEBE 2005, Beijing, China, October 18-20, 2005, pp. 535–540 (2005)
[34] Koch, N., Kraus, A., Cachero, C., Melia, S.: Integration of Business Processes in Web Application Models. Journal of Web Eng. 3(1), 22–49 (2004)
[35] Mecca, G., Merialdo, P., Atzeni, P., Crescenzi, V.: The (Short) Araneus Guide to Web-Site Development. In: WebDB (Informal Proceedings), pp. 13–18 (1999)
[36] Merialdo, P., Atzeni, P., Mecca, G.: Design and development of data-intensive Websites: the Araneus approach. ACM TOIT 3(1), 49–92 (2003)
[37] Noll, J., Scacchi, W.: Specifying process-oriented hypertext for organizational computing. Journal of Network and Computer Applications 24, 39–61 (2001)
[38] Oracle, Oracle Developer Suite, JDeveloper 10g, `http://www.oracle.com/tools`
[39] Rational, Rational Rapid Developer, `http://www.ibm.com/software/awdtools/rapiddeveloper`
[40] Rossi, L., Schmid, H., Lyardet, F.: Engineering Business Processes in Web Applications: Modeling and Navigation Issues. In: Third International Workshop on Web Oriented Software Technology, Oviedo 2003, pp. 81–89 (2003)
[41] Schwabe, D., Rossi, G.: The Object-Oriented Hypermedia Design Model. Communications of the ACM 38(8), 45–46
[42] Urbieta, M., Rossi, G., Ginzburg, J., Schwabe, D.: Designing the Interface of Rich Internet Applications. In: Latin-American Conference on the WWW, pp. 144–153. IEEE, Los Alamitos (2007)
[43] Vdovjak, R., Frasincar, F., Houben, G.-J., Barna, P.: Engineering semantic web information systems in Hera. Journal of Web Engineering 2(1-2), 3–26 (2003)
[44] Web Services Description Language 1.1, W3C Note (March 2001)