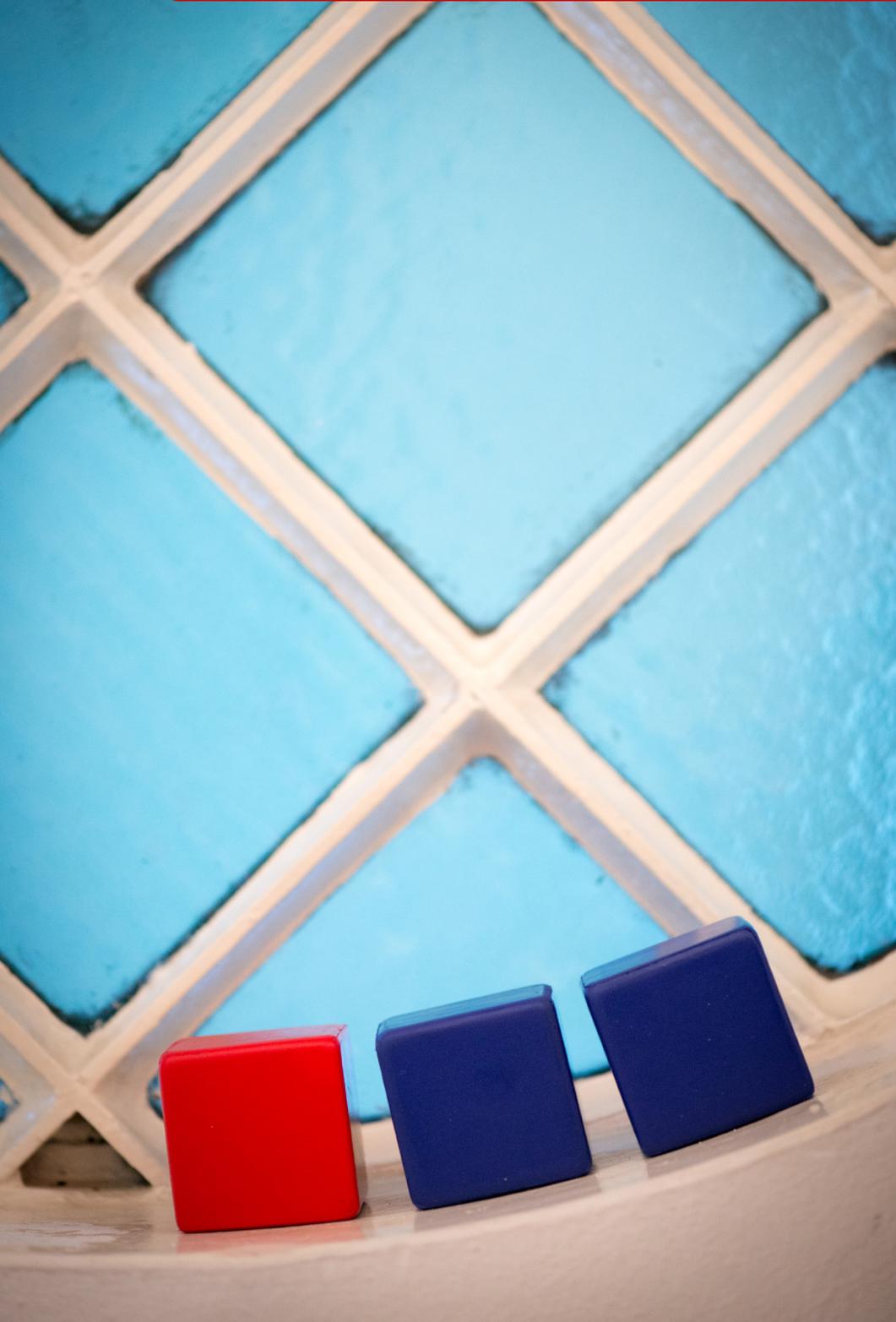


**FIFTEEN YEARS OF INDUSTRIAL MODEL-DRIVEN  
DEVELOPMENT IN SOFTWARE FRONT-ENDS:**

**FROM WEBML TO WEBRATIO AND IFML**





Text Marco Brambilla<sup>1</sup>, Stefano Butti<sup>2</sup>

<sup>1</sup> Politecnico di Milano, DEIB.  
Piazza L. Da Vinci, 32. I-20133 Milán, Italia  
marco.brambilla@polimi.it

<sup>2</sup> WebRatio.  
Piazza Cadorna, 10. I-20123 Milán, Italia  
marco.brambilla@polimi.it

**Abstract:** This paper discusses the history behind the standard IFML, recently adopted by the Object Management Group. We show how our initial proposal called WebML has been an incubator for research and industrial exploitation on conceptual modeling, exploiting existing experiences in the field and continuously addressing new challenges concerning abstractions, methods, tools, and technologies. We summarize the essence of the approach and we show the supporting modelling tool WebRatio at work.

# Table of Contents

1. Introduction.....	1
2. The Original WebML Language.....	2
3. Service-Oriented Architectures.....	4
4. Business Processes on the Web .....	6
5. User Personalization and Context Awareness .....	7
6. Semantic Web and Services .....	9
7. Rich Internet Applications.....	9
8. The Interaction Flow Modeling Language .....	10
9. The WebRatio development platform .....	11
10. Conclusions.....	13
References – WebML .....	13

# 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.

In this paper we report on our efforts towards the definition of a model-driven approach that targets one of the most relevant aspects in the development of such systems: the front-end. Front-end development continues to be a costly and inefficient process, where manual coding is the predominant development approach, reuse of design artifacts is low, and cross-platform portability remains difficult. The availability of a platform independent interaction flow modeling language can bring several benefits to the development process of application front-ends: it permits the formal specification of the different perspectives of the front-end (content, interface composition, interaction and navigation options, and connection with the business logic and the presentation); it separates the stakeholder concerns by isolating the specification of the front-end from its implementation-specific issues; it improves the development process, by fostering the separation of concerns in the user interaction design, thus granting the maximum efficiency to all the different developer roles; it enables the communication of interface and interaction design to non-technical stakeholders, permitting early validation of requirements.

We present our work on the Interaction Flow Modeling Language (IFML), adopted as a standard by the Object Management Group in March 2013. Its definition is particularly attentive to model usability and understandability, by carefully considering all the factors that can contribute to making a PIM quickly learned, easy to use, amenable to implementation, and open to reuse and extensibility:

- It is concise, avoiding redundancy and reducing the number of diagram types and concepts needed to express the salient interaction design decisions.
- It provides model inference rules at the modelling level that automatically apply default modeling patterns and allows automatic inference of details when needed.
- It includes extensibility in the definition of new concepts (e.g., novel interface components or event types).
- It ensures implementability, that is, it supports the construction of model transformation frameworks and code generators that can map the PIM into a suitable PSM and ultimately into executable applications for a wide range of technological platforms and access devices.
- It ensures model-level reuse, that is, it supports the definition of reusable design patterns that can be stored, documented, searched and retrieved, and re-used in other applications.

IFML can be seen as the consolidation of the the Web Modelling Language (WebML) [1] defined and patented about 15 years ago as a conceptual model for data-intensive Web applications. Its main innovation was the hypertext modelling notation, 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. Indeed, this is one of the main values transferred to IFML.

WebML underwent a set of extensions along the years, so as to follow the main software development technologies and trends:

- 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, quickly describes its main extensions, and shows the approach at work on its companion development platform WebRatio.

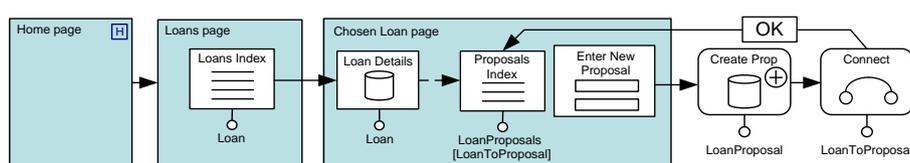
## 2. The Original WebML Language

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.

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.



**Figure 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.

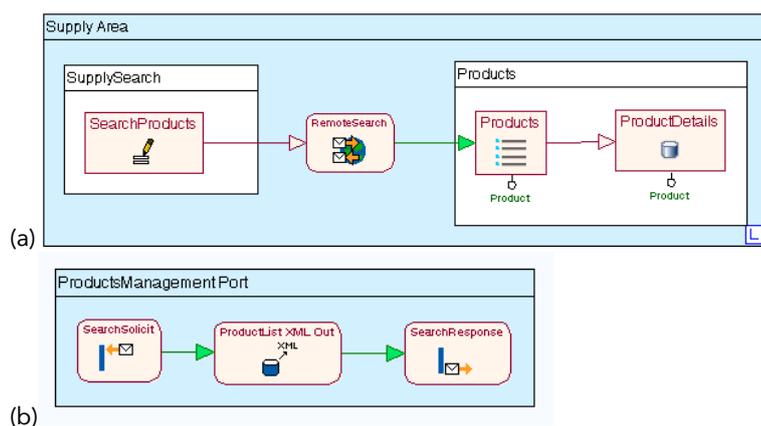
As already mentioned, WebML is associated with a page computation algorithm deriving from the formal definition of the model's semantics (based on statecharts in [10]). The essential point is the **page computation algorithm**, which describes how the content of the page is determined after a navigation 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.

### 3. Service-Oriented Architectures

The first WebML extension is towards Service Oriented Architectures [11]. 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.

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.



**Figure 4.** Example of WebML hypertext model with invocation of remote service.

**Service invocation** and **reaction to messages** are supported by specialized components, called **Web Service units**. These primitives correspond to the classical 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<sup>1</sup>.

## 4. Business Processes on the Web

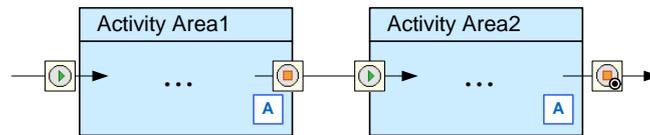
In time, the Web has become a popular implementation platform for B2B applications, whose goal is not only the navigation of content, but also the enactment of intra- and inter-organization business processes. Web-based B2B applications exhibit much more sophisticated interaction patterns than traditional 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 and WebRatio have been extended to cover the requirements of this class of applications [5], by:

- The integration in the development life-cycle of workflow-specific design 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 approach to business process –based modeling exploits the BPMN notation for the description of the business requirements, and then maps them to hypertext model chunks that describe the user interaction of every task of the business process. 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 synchrony. 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].

---

<sup>1</sup> 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.



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

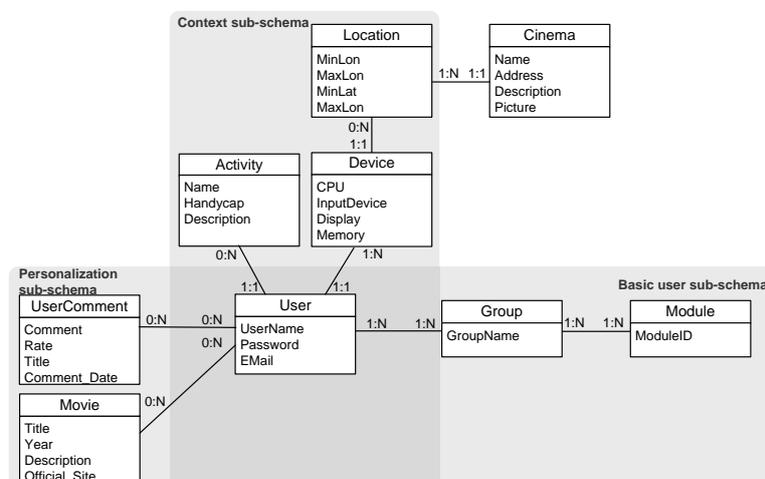
The runtime calculation of the process execution steps is determined by a process control logic component. It exploits the information stored in the process meta-data and logs to calculate the current process status and the enabled state transitions.

WebRatio supports the business process modeling phase through:

- 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 components and containers for implementing the workflow enactment.
- A one-click code generator from the BPMN models, which generates running prototypes starting from the business processes, without the need of going into the WebML modeling at all.

## 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].



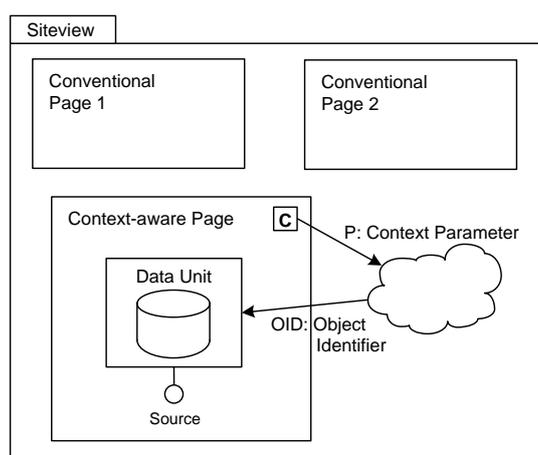
**Figure 7.** Three models representing user, personalization, and context data.

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**.

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.



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

## 6. Semantic Web and Services

Traditionally, the service requestor and service provider are designed jointly and then tightly bound together when an application is created. The field of Semantic Web and Semantic Services provides paradigms for semantically enriching the existing syntactic descriptions of Web content, Web sites and Web services; then, the requestor can search, either at design or at run time, among a variety of Web-enabled 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 Semantic Web paradigm in a semi-automatic manner, thus integrating the production and maintenance of semantic information into the application generation cycle.

We defined a **process for semantic resources 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.

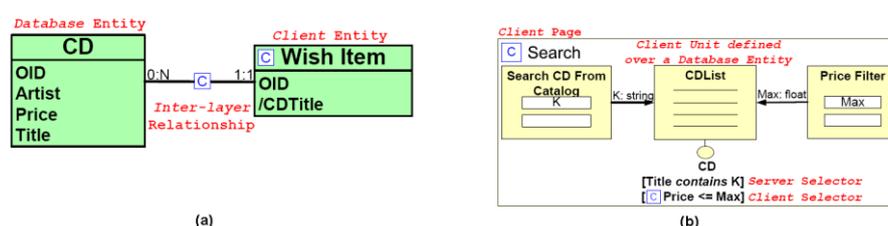
The basic WebML primitives have been extended with components for **ontology querying and navigation**, exploiting the expressive power of ontological languages (inspired by SPARQL and RDF-S). These units allow queries on classes, instances, properties, and values; checking the existence of specific 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.

## 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).



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

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. Figure 9 shows a client page which contains an index unit with the population fetched from the server, but filtered using a predicate ( $price \leq max$ ) 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.

## 8. The Interaction Flow Modeling Language

The standard Interaction Flow Modeling Language (IFML) [12] has spun off from the WebML experience, and aimed at generalizing it to any kind of software platform.

IFML has been designed for expressing the content, user interaction and control behaviour of the front-end of applications belonging to the following domains:

- Traditional, HTML+HTTP based Web applications
- Rich Internet Applications, as supported by the forthcoming HTML 5 standard.
- Mobile applications.
- Client-server applications.
- Desktop applications.
- Embedded Human Machine Interfaces for control applications.
- Multichannel and context-aware applications.

It's worth noting again that IFML does not cover the modeling of the presentation issues (e.g., layout, style and look&feel) of an application front-end and does not cater for the specification of bi-dimensional and tri-dimensional computer based graphics, videogames, and other highly interactive applications. It is mainly aimed at business-oriented, data-intensive applications instead.

The notations and concepts do not vary much with respect to the WebML ones, except for some terminology. The two big evolutions with respect to WebML are:

- The definition of the event concept as first-class citizen of the standard, so as to cover the management of a broad spectrum of event types, spanning user events and also system-triggered events.
  - The removal of orchestration chains of business logic components, so as to make the language more precisely focused on the front-end design (and delegating the modelling of orchestrations to other models, e.g., UML sequence or activity diagrams).

IFML has been adopted as a standard by the OMG in Beta version in March 2013, and has been finalized in IFML 1.0 version in March 2014.

## 9. The WebRatio development platform

The WebRatio development platform is now adopting IFML as official notation. Since 2001, WebRatio has been working side by side with IT managers, web architects and Java developer to support them in the development of new enterprise applications, the updating of existing ones and the integration of open and legacy systems. More than 60 software engineers in 3 different countries, keen on Model-Driven Development, develop WebRatio for corporate customers, primarily in the areas of finance, energy, transportation, government and retail, also covering big brands like Acer (their world-wide Web presence is built and maintained with WebRatio), UniCredit, Interflora and others<sup>2</sup>. WebRatio invests 22% of its turnover in research and development and has always collaborated within European research projects, often in partnership with Politecnico di Milano. WebRatio has been named Gartner Cool Vendor in 2013.

The user interface of WebRatio's IFML editor is shown in Figure 10. WebRatio automatically generates fast, secure, scalable and robust Web and mobile applications out of IFML and BPMN models. WebRatio Platform can deploy applications across Web and mobile devices, with

---

<sup>2</sup> See a comprehensive list of customers at: <http://www.webratio.com/portal/content/en/customers>

customized and consistent user interaction. It produces both client-side and server-side optimized code. WebRatio Platform is scalable, reliable and complies with the most stringent corporate security policies to develop fully branded B2C and B2E enterprise applications (like Portals, e-Commerce sites, Help Desk Ticketing Solutions, Customers Self-Service Desks, Subscription Management Systems, Corporate custom applications, etc.). The generated applications are compliant with HTML5, CSS3 and Java to keep up with the continuous codes and technologies changes. Web applications built with WebRatio Platform comply with the Java/JSP 2.0+ standard and can be deployed on any application server. Applications can be automatically deployed On-Premises or in the Cloud (Public or Private). Deployment plans are fully customizable.

WebRatio Platform integrates with Application Lifecycle Management tools (Atlassian JIRA, IBM Rational Team Concert, etc.). All the resources of the project are shared through a collaborative work server and a versioning server (CVS or Subversion). It also helps address DevOps (Development & Operations) challenges by facilitating collaboration between development and operation teams.

With WebRatio custom components, it's possible to integrate the application with a variety of systems and services like SAP, Tibco, IBM Mainframe and SaaS applications like Salesforce.com, Dropbox etc. including own custom legacy code. The WebRatio Platform facilitates the integration of IT systems with external solutions and helps to invoke and publish Web Services (REST, SOAP), as well as with Single Sign On login services, content sharing, contacts' management, and Social Networks such as Facebook, Twitter, LinkedIn, G+, etc.

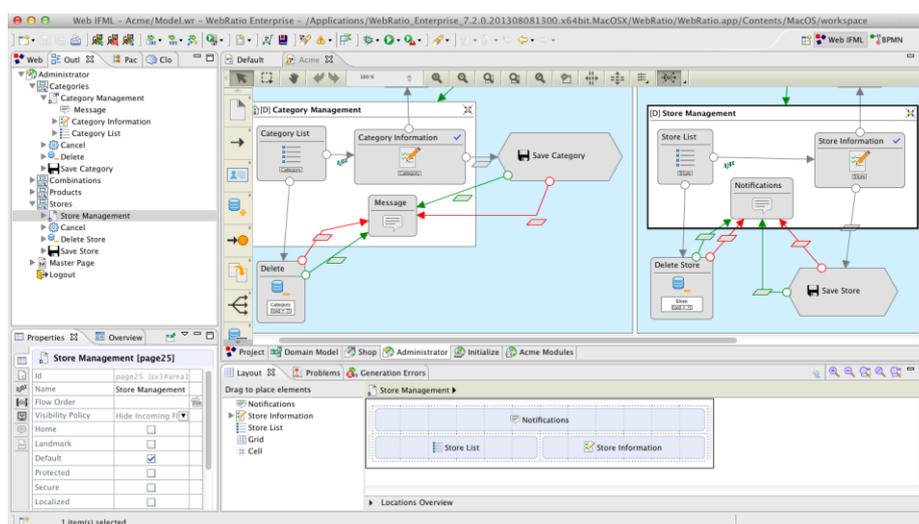


Figure 10. WebRatio screenshot showing an excerpt of IFML diagram.

## 10. Conclusions

In this paper we presented the IFML language and the WebRatio model-driven development platform, together with the path starting from WebML that lead to the standardization of IFML in OMG. The standard and the tool are now widely adopted in large industries in Europe and United States, as well as in Latin America and Asia. 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, with gains up to 60% in development effort and up to 80% savings in the maintenance and evolution phase, thanks to automatic code generation and possibility of working always at the modelling level [2].

## References – WebML

- [1] R. Acerbis, A. Bongio, M. Brambilla, S. Butti, S. Ceri, P. Fraternali. Web Applications Design and Development with WebML and WebRatio 5.0. *TOOLS 2008*, 392-411 and <http://www.webratio.com/>.
- [2] R. Acerbis, A. Bongio, M. Brambilla, M. Tisi, S. Ceri, E. Tosetti. Developing eBusiness Solutions with a Model Driven Approach: The Case of Acer EMEA. 7th International Conference on Web Engineering, ICWE 2007, Como, Italy. Springer LNCS 4607, ISBN 978-3-540-73596-0, pp. 539-544.
- [3] A. Bozzon, S. Comai, P. Fraternali, G. Toffetti Carughi. Conceptual Modeling and Code Generation for Rich Internet Applications. International Conference on Web Engineering, Springer, 2006, pp. 353-360.
- [4] M. Brambilla, I. Celino, S. Ceri, D. Cerizza, E. Della Valle. Model-Driven Design and Development of Semantic Web Service Applications. *ACM TOIT*, 8:1, 2008.
- [5] M. Brambilla, S. Ceri, P. Fraternali, I. Manolescu. Process Modeling in Web Applications. *ACM TOSEM*, 15:4, 2006.
- [6] S. Ceri, F. Daniel, M. Matera, F. Facca. Model-driven Development of Context-Aware Web Applications, *ACM TOIT*, 7:1, 2007.
- [7] S. Ceri, P. Fraternali, A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *WWW9 / Computer Networks* 33, 2000.
- [8] S. Ceri, P. Fraternali, A. Bongio, M. Brambilla, S. Comai, M. Matera. Designing Data-Intensive Web Applications. Morgan Kaufmann, 2002.
- [9] S. Ceri, M. Matera, F. Rizzo, V. Demaldé. Designing Data-Intensive Web Applications for Content Accessibility using Web Marts. *Communications of ACM* 50: 4, 55-61, 2007.
- [10] S. Comai, P. Fraternali. A Semantic Model for Specifying Data-Intensive Web Applications Using WebML. Semantic Web Workshop, Stanford, USA, July 2001.
- [11] I. Manolescu, M. Brambilla, S. Ceri, S. Comai, P. Fraternali. Model-Driven Design and Deployment of Service-Enabled Web Applications. *ACM TOIT*, 5:3, 2005.
- [12] Object Management Group, M. Brambilla, P. Fraternali et al. The Interaction Flow Modeling Language (IFML) 1.0. March 2014. [www.ifml.org](http://www.ifml.org)