Web Modeling Language (WebML): a modeling language for designing Web sites

Stefano Ceri, Piero Fraternali, Aldo Bongio

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. da Vinci, 32 - 20133 Milano, Italy
ceri/fraterna@elet.polimi.it, bongio@fusberta.elet.polimi.it

Abstract

Designing and maintaining Web applications is one of the major challenges for the software industry of the year 2000. In this paper we present Web Modeling Language (WebML), a notation for specifying complex Web sites at the conceptual level. WebML enables the high-level description of a Web site under distinct orthogonal dimensions: its data content (structural model), the pages that compose it (composition model), the topology of links between pages (navigation model), the layout and graphic requirements for page rendering (presentation model), and the customization features for one-to-one content delivery (personalization model). All the concepts of WebML are associated with a graphic notation and a textual XML syntax. WebML specifications are independent of both the client-side language used for delivering the application to users, and of the server-side platform used to bind data to pages, but they can be effectively used to produce a site implementation in a specific technological setting. WebML guarantees a model-driven approach to Web site development, which is a key factor for defining a novel generation of CASE tools for the construction of complex sites, supporting advanced features like multi-device access, personalization, and evolution management. The WebML language and its accompanying design method are fully implemented in a pre-competitive Web design tool suite, called ToriiSoft.

Keywords: Hypermedia Design Methodologies, Navigation, Design Tools, XML

1. Introduction

In the early stage of Web development, it was current practice to approach Web applications by simply "building the solution", with little emphasis on the development process. However, many companies are now experiencing severe problems in the management of Web sites, as these grow in size and complexity, need to inter-operate with other applications, and exhibit requirements that change over time.

State-of-the-practice Web development tools help simplify the generation and deployment of data-intensive Web applications by means of page generators, such as Microsoft's Active Server Pages or JavaSoft's Java Server Pages, whose primary function is to dynamically extract content from data sources and include it into user-programmed page templates. Even if these systems are very productive implementation tools, they offer scarce support to bridge the gap between requirements collection and the subsequent phases of the development process. We have directly experienced that many companies building Web applications deeply need design methods, formalisms, languages, and tools, which could complement current Web technology in an effective way, covering all the aspects of the design process.

In response to this need, the W3I3 Project (funded by the European Community under the Fourth Framework Program) is focusing on "Intelligent Information Infrastructure" for data-intensive WEB applications. The project, driven by the requirements of two major Web developers (Otto-Versand from Germany, specialized in e-commerce, and the Dutch PPT (KPN), involved in Web-hosting services) has produced a novel Web modeling language, called WebML, and a supporting CASE environment, called Toriisoft (http://www.toriisoft.com). WebML addresses the high-level, platform-independent specification of data-intensive Web applications and targets Web sites that require such advanced features as the one-to-one personalization of content and the delivery of information on multiple devices, like PCs, PDAs, digital televisions, and WAP phones. Toriisoft is a suite of design tools, which covers the entire life cycle of Web applications and follows a model-driven approach to Web design, centered on the use of WebML.

In this paper, we focus on the presentation of WebML, and in particular on its composition and navigation

modeling primitives. More information on the W3I3 Project and on the ToriiSoft tool suite can be found at: http://www.toriisoft.com and http://www.txt.it/w3i3.

1.1 WebML in a nutshell

WebML enables designers to express the core features of a site at a high level, without committing to detailed architectural details. WebML concepts are associated with an intuitive graphic representation, which can be easily supported by CASE tools and effectively communicated to the non-technical members of the site development team (e.g., with the graphic designers and the content producers). WebML also supports an XML syntax, which instead can be fed to software generators for automatically producing the implementation of a Web site. The specification of a site in WebML consists of four orthogonal perspectives:

1. Structural Model: it expresses the data content of the site, in terms of the relevant entities and relationships (see Figure 1). WebML does not propose yet another language for data modeling, but is compatible with classical notations like the E/R model [9], the ODMG object-oriented model [5], and UML class diagrams [4]. To cope with the requirement of expressing redundant and calculated information, the structural model also offers a simplified, OQL-like query language, by which it is possible to specify derived information.
2. Hypertext Model: it describes one or more hypertexts that can be published in the site. Each different hypertext defines a so-called site view (see Figure 2). Site view descriptions in turn consist of two sub-models.
    - Composition Model: it specifies which pages compose the hypertext, and which content units make up a page. Six types of content units can be used to compose pages: data, multi-data, index, filter, scroller and direct units. Data units are used to publish the information of a single object (e.g., a music album), whereas the remaining types of units represent alternative ways to browse a set of objects (e.g., the set of tracks of an album). Composition units are defined on top of the structure schema of the site; the designer dictates the underlying entity or relationship on which the content of each unit is based. For example, the AlbumInfo data unit showing the information on an album in Figure 2 refers to the Album entity specified in the structure schema of Figure 1.
    - Navigation Model: it expresses how pages and content units are linked to form the hypertext. Links are either non-contextual, when they connect semantically independent pages (e.g., the page of an artist to the home page of the site), or contextual, when the content of the destination unit of the link depends on the content of the source unit. For example, the page showing an artist's data is linked by a contextual link to the page showing the index of reviews of that specific artist. Contextual links are based on the structure schema, because they connect content units whose underlying entities are associated by relationships in the structure schema.
3. Presentation Model: it expresses the layout and graphic appearance of pages, independently of the output device and of the rendition language, by means of an abstract XML syntax. Presentation specifications are either page-specific or generic. In the former case they dictate the presentation of a specific page and include explicit references to page content (e.g., they dictate the layout and the graphic appearance of the title and cover data of albums); in the latter, they are based on predefined models independent of the specific content of the page and include references to generic content elements (for instance, they dictate the layout and graphic appearance of all attributes of a generic object included in the page).
4. Personalization Model: users and user groups are explicitly modeled in the structure schema in the form of predefined entities called User and Group. The features of these entities can be used for storing group-specific or individual content, like shopping suggestions, list of favorites, and resources for graphic customization. Then, OQL-like declarative expressions can be added to the structure schema, which define derived content based on the profile data stored in the User and Group entities. This personalized content can be used both in the composition of units or in the definition of presentation specifications. Moreover, high-level business rules, written using a simple XML syntax, can be defined for reacting to site-related events, like user clicks and content updates. Business rules typically produce new user- related information (e.g., shopping histories) or update the site content (e.g., inserting new offers matching users' preferences). Queries and business rules provide two alternative paradigms (a declarative and a procedural one) for effectively expressing and managing personalization requirements.

In the ToriiSoft tool suite, WebML specifications are given as input to a code generator, which translates them into some concrete markup language (e.g. HTML or WML) for rendering the composition, navigation and presentation, and maps the abstract references to content elements inside pages into concrete data retrieval instructions in some server-side scripting language (e.g., JSP or ASP).

1.2 WebML by example

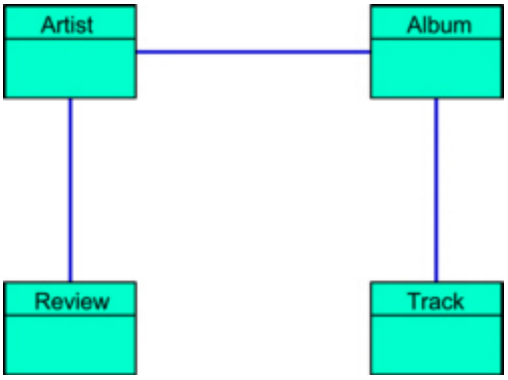

Figure 1 - Example of structure schema

Figure 1 shows a simple structure schema for the publication of albums and artists information. Artists publish albums composed of tracks, and have biographic information and reviews of their work. To publish this information as a hypertext on the Web, it is necessary to specify criteria for composition and navigation, i.e., to define a site view. Figure 2 shows an excerpt from a site view specification, using WebML graphical language. The hypertext consists of three pages, shown as dashed rectangles. Each page encloses a set of units (shown as solid rectangles with different icons) to be displayed together in the site. For example, page AlbumPage collects information on an album and its artist. It contains a data unit (AlbumInfo) showing the information on the album, an index unit (TrackIndex) showing the list of the album's tracks, and another data unit (ArtistInfo) containing the essential information on the album's artist. The AlbumInfo unit is connected to the ArtistInfo unit by an intermediate direct unit (ToArtist), meaning that the AlbumInfo refers to the (single) artist who composed the album shown in the page. The ArtistInfo unit has one outgoing link leading to a separate page containing the list of review, and one link to a direct unit pointing to the artist's biographic data, shown on a separate page. Note that changing the hypertext topology is extremely simple: for example, if the ReviewIndex data unit is specified inside the AlbumPage instead of on a separate page, then the index of reviews is kept together with the album and artist info. Alternatively, if the ReviewIndex unit is defined as a multi-data unit, instead of an index unit, all reviews (and not only their titles) are shown in the ReviewsPage. A possible HTML rendition of the AlbumPage page of Figure 2 (with some additional features omitted for simplicity in the example of figure 2) can be seen by accessing the site www.cdnow.com and then entering the page of any album.

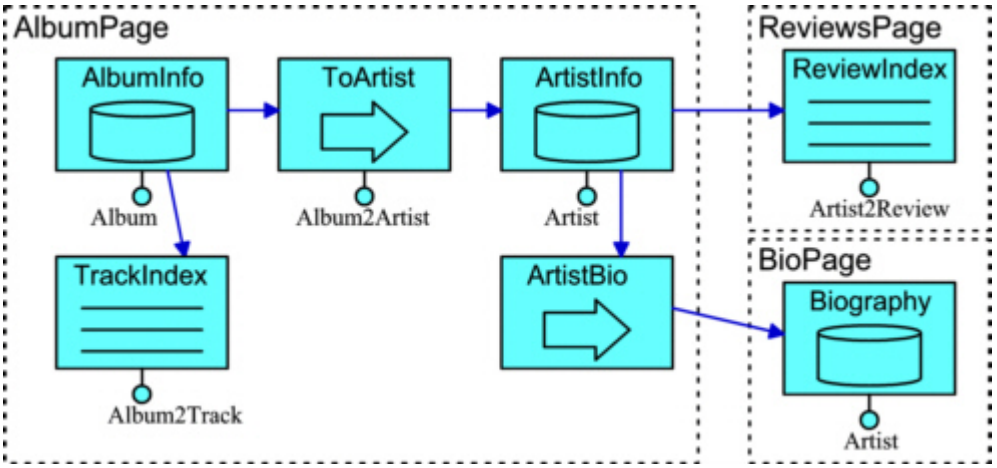

Figure 2 - Example of WebML composition and navigation specification

1.3 Design Process in WebML

Web application development is a multi-facet activity involving different players with different skills and goals. Therefore, separation of concerns is a key requirement for any Web modeling language. WebML addresses

this issue and assumes a development process where different kinds of specialists play distinct roles: 1) the data expert designs the structural model; 2) the application architect designs pages and the navigation between them; 3) the style architect designs the presentation styles of pages; 4) the site administrator designs users and personalization options, including business rules.

A typical design process using WebML proceeds by iterating the following steps for each design cycle:

- Requirements Collection. Application requirements are gathered, which include the main objectives of the site, its target audience, examples of content, style guidelines, required personalization and constraints due to legacy data.
- Data Design. The data expert designs the structural model, possibly by reverse-engineering the exisiting logical schemas of legacy data sources.
- Hypertext Design ``in the large''. The Web application architect defines the structure ``in the large'' of the hypertext, by identifying pages and units, linking them, and mapping units to the main entities and relationships of the structure schema. In this way, he develops a "skeleton" site view, and then iteratively improves it. To support this phase, WebML-based tools must enable the production of fast prototypes to get immediate feedback on all design decisions.
- Hypertext Design ``in the small''. The Web application architect concentrates next in the design ``in the small'' of the hypertext, by considering each page and unit individually. At this stage, he may add non-contextual links between pages, consolidate the attributes that should be included within a unit, and introduce novel pages or units for special requirements (e.g., alternative index pages to locate objects, filters to search the desired information, and so on). During page design in the small, the Web application architect may discover that a page requires additional information, present in another concept semantically related to the one of the page currently being designed. Then, he may use the derivation language, to add ad hoc redundant data to the structure schema and include it in the proper units.
- Presentation Design. Once all pages are sufficiently stable, the Web style architect adds to each page a presentation style.
- User and Group Design. The Web administrator defines the features of user profiles, based on personalization requirements. Potential users and user groups are mapped to WebML users and groups, and possibly a different site view is created for each group. The design cycle is next iterated for each of the identified site views. ``Copy-and-paste'' of already designed site view pages and links may greatly speed up the generation of other site views.
- Customization Design. The Web administrator identifies profile-driven data derivations and business rules, which may guarantee an effective personalization of the site.

Some of the above stages can be skipped in the case of development of a simple WEB application. In particular, defaults help at all stages the production of simplified solutions. At one extreme, it is possible to develop a default initial site view directly from the structural schema, skipping all of the above stages except the first one (see Section 4.4).

2. The structural model

The fundamental elements of WebML structure model are entities, which are containers of data elements, and relationships, which enable the semantic connection of entities. Entities have named attributes, with an associated type; properties with multiple occurrences can be organized by means of multi-valued components, which corresponds to the classical part-of relationship. Entities can be organized in generalization hierarchies. Relationships may be given cardinality constraints and role names. As an example, the following XML code represents the WebML specification of the structural schema illustrated in figure 1:

```
<DOMAIN id="SupportType" values="CD Tape Vinyl">

<ENTITY id="Album">
  <ATTRIBUTE id="title" type="String"/>
  <ATTRIBUTE id="cover" type="Image"/>
  <ATTRIBUTE id="year" type="Integer"/>
  <COMPONENT id="Support" minCard="1" maxCard="N">
    <ATTRIBUTE id="type" userType="SupportType"/>
    <ATTRIBUTE id="listPrice" type="Float"/>
    <ATTRIBUTE id="discountPercentage" type="Integer"/>
```

```
    <ATTRIBUTE id="currentPrice" type="Float"
               value="Self.listPrice *
                      (1 - (Self.discountPercentage / 100))"/>
  </COMPONENT>
  <RELATIONSHIP id="Album2Artist" to="Artist" inverse="ArtistToAlbum"
                minCard="1" maxCard="1"/>
  <RELATIONSHIP id="Album2Track to="Track" inverse="Track2Album"
                minCard="1" maxCard="N"/>
</ENTITY>

<ENTITY id="Artist">
  <ATTRIBUTE id="firstName" type="String"/>
  <ATTRIBUTE id="lastName" type="String"/>
  <ATTRIBUTE id="birthDate" type="Date"/>
  <ATTRIBUTE id="birthPlace" type="String"/>
  <ATTRIBUTE id="photo" type="Image"/>
  <ATTRIBUTE id="biographicInfo" type="Text"/>
  <RELATIONSHIP id="Artist2Album" to="Album" inverse="Album2Artist"
                minCard="1" maxCard="N"/>
  <RELATIONSHIP id="Artist2Review" to="Review" inverse="Review2Artist"
                minCard="0" maxCard="N"/>
 </ENTITY>

<ENTITY id="Track">
   <ATTRIBUTE id="number" type="Integer"/>
   <ATTRIBUTE id="title" type="String"/>
   <ATTRIBUTE id="mpeg" type="URL"/>
   <ATTRIBUTE id="hqMpeg" type="URL"/>
   <RELATIONSHIP id="Track2Album" to="Album" inverse="Album2Track"
                minCard="1" maxCard="1"/>
</ENTITY>

<ENTITY id="Review">
   <ATTRIBUTE id="text" type="Text"/>
   <ATTRIBUTE id="autho" type="String/>
   <RELATIONSHIP id="Review2Artist" to="Artist" inverse="Artist2Review"
                minCard="1" maxCard="1"/>
</ENTITY>
```

The structural schema consists of four entities (Artist, Album, Review and Track) and three relationships (Artist2Album, Artist2Review, Album2track). Entity Album has a multi-valued property represented by the Support component, which specifies the various issues of the album on vinyl, CD, and tape. Note that each issue has a discounted price, whose value is computed by applying a discount percentage to the list price, by means of a derivation query. Derivation is briefly discussed in Section 5.1.

3. The Composition Model

The purpose of composition modeling is to define which nodes make up the hypertext contained in the Web site. More precisely, composition modeling specifies content units (units for short), i.e., the atomic information elements that may appear in the Web site, and pages, i.e., containers by means of which information is actually clustered for delivery to the user. In a concrete setting, e.g., an HTML or WML implementation of a WebML site, pages and units are mapped to suitable constructs in the delivery language, e.g., units may map to HTML files and pages to HTML frames organizing such files on the screen.

WebML supports six types of unit to compose an hypertext:

- Data units: they show information about a single object, e.g., an instance of an entity or of a component.
- Multidata units: they show information about a set of objects, e.g., all the instances of an entity or all the sub-components of a composite object.
- Index units: they show a list of objects (entity or component instances), without presenting the detailed information of each object.
- Scroller units: they show commands for accessing the elements of an ordered set of objects (the first, last,

previous, next, i-th).

- Filter units: they show edit fields for inputting values used for searching within a set of object(s) those ones that meet a condition.
- Direct units: they do not display information, but are used to denote the connection to a single object that is semantically related to another object.

Data and multidata units present the actual content of the objects they refer to, whereas the remaining types of units permit one to locate objects. Data units refer to a single object. Multidata, index, filter, and scroller refer to a set of objects. Therefore, they are collectively called container units.

3.1 Data Units

Data units are defined to select a mix of information, which provides a meaningful view of a given concept of the structure schema. More than one unit can be defined for the same entity or component, to offer alternative points of view (e.g., a short or long, textual or multimedia version of the object).
The definition of a data unit requires 1) the indication of the concept (entity or component) to which the unit refers. 2) The selection of the attributes to include in the unit. Syntactically, data units are defined using the DATAUNIT element, which provides tags and attributes for the various aspects of unit definition. The selective inclusion of content in a unit is specified using the element INCLUDE. Included attributes must be chosen among those declared in the structure schema for the entity or component. The INCLUDEALL element can be used to specify that all attributes are included. For example, the following definitions introduce two units for presenting the Artist entity. The goal of these definitions is to provide a short view of artists (limited to the first name, last name, and photo) and a complete view, including all data.

```
<DATAUNIT id="ShortArtist" entity="Artist">
   <INCLUDE attribute="firstName"/>
   <INCLUDE attribute="lastName"/>
   <INCLUDE attribute="photo"/>
</DATAUNIT>

<DATAUNIT id="BiographyUnit" entity="Artist">
   <INCLUDEALL/>
</DATAUNIT>
```

Figure 3 shows the WebML graphic notation for representing a data unit and its underlying entity, and a possible rendition of the ShortArtist data unit in an HTML-based implementation.

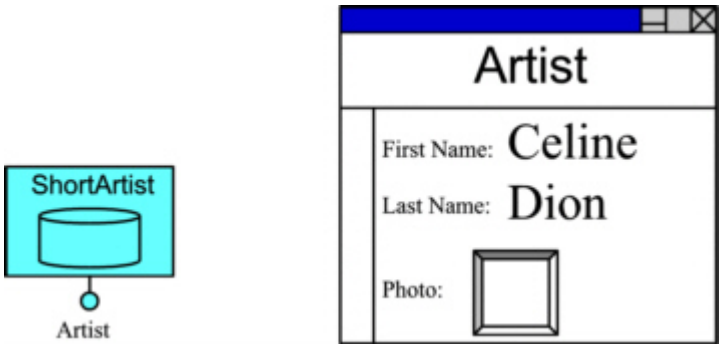

Figure 3 - WebML graphic notation for data units, and a possible rendition in HTML

3.2 Multi-Data Units

Multi-data units present multiple instances of an entity or component together, by repeating the presentation of several, identical data units. Therefore, a multi-data unit specification has two parts: 1) the container which includes the instances to be displayed, which may refer to an entity, relationship, or component. 2) The data unit used for the presentation of each instance. Syntactically, a multi-data unit is represented by a MULTIDATAUNIT element, which includes a nested DATAUNIT element. The container is an argument of the external MULTIDATAUNIT element. The following example shows how all albums can be shown in the same multidata unit, by displaying all attributes of each individual album.

```
<MULTIDATAUNIT id="MultiAlbumUnit" entity="Album">
   <DATAUNIT id="AlbumUnit" entity="Album">
      <INCLUDEALL/>
   </DATAUNIT>
</MULTIDATAUNIT>
```

Figure 4 shows the WebML graphic notation for representing a multidata unit and a possible rendition of the multidata unit in an HTML-based implementation.

Figure 4 - WebML graphic notation for multidata units, and a possible rendition in HTML

3.3 Index Units

Index units present multiple instances of an entity or component as a list, by denoting each object as an entry in the list. An index unit specification has two main parts: 1) the container which includes the instances to be displayed, which may be an entity, relationship, or component. 2) The attributes used as index key. Syntactically, an INDEXUNIT element is used, which includes a nested DESCRIPTION element. The following example shows how all albums can be shown in a list, by displaying only the title of each individual album. Figure 5 shows the WebML graphic notation for representing an index unit and a possible rendition of the index unit in an HTML-based implementation.

```
<INDEXUNIT id="AlbumIndex" entity="Album">
  <DESCRIPTION  Key="title"/>
</INDEXUNIT>
```

Figure 5 - WebML graphic notation for index units, and a possible rendition in HTML

3.4 Scroller Units

Scroller units provide commands to scroll through the objects in a container, e.g., all the instances of an entity or all the objects associated to another object via a relationship. A scroller unit is normally used in conjunction with a data unit, which represents the currently visualized element of the container. Syntactically, the SCROLLERUNIT element is used, which specifies the container (entity, relationship, or component) providing the set of objects to scroll, and suitable attributes to express which scrolling commands to use. For example, the following declaration introduces a unit for moving along the set of reviews of an artist, whereby it

is possible to move to the first, previous, next and last review.

```
<SCROLLERUNIT id="AlbumScroll" entity="Album" first="yes"
                 last="yes" previous="yes" next="yes"/>
```

Figure 6 shows the WebML graphic notation for representing a scroller unit and a possible rendition in an HTML-based implementation.

Figure 6 - WebML graphic notation for scroller units, and a possible rendition in HTML

3.5 Filter Units

Filter units provide input fields to search the objects in a container, e.g., all the instances of an entity whose attributes contain a given string. A filter unit is normally used in conjunction with an index or multidata unit, which present object matching the search condition. Syntactically, the FILTERUNIT element is used, which specifies the container (entity, relationship, or component) providing the set of objects to search. Inside the FILTERUNIT element, the SEARCHATTRIBUTE element is used to specify a search predicate on the value of a specific attribute. This element tells the attribute on which the search has to be performed and the comparison operator to use. In the following example, the AlbumFilter unit specifies a search form over the set of all albums. The form includes two input fields: the former for inputting a string to be located in the album's title, the latter for inputting the publication time interval of the album.

```
<FILTERUNIT id="AlbumFilter" entity="Album"/>
  </SEARCHATTRIBUTE name="title" predicate="like">
  </SEARCHATTRIBUTE name="year"  predicate="between">
</FILTERUNIT>
```

Figure 7 shows the WebML graphic notation for representing a filter unit and a possible rendition in an HTML-based implementation.

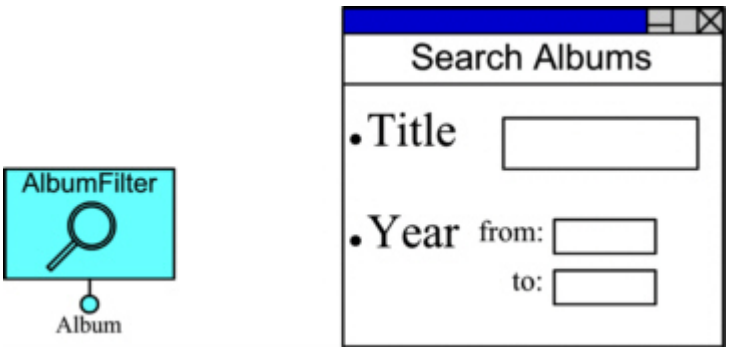

Figure 7 - WebML graphic notation for filter units, and a possible rendition in HTML

3.6 Direct Units

Direct units are a syntactic convenience to express a particular kind of index, which always contains a single object associated to another object by a one-to-one relationship. Differently from index units, direct units are

not displayed, but merely support the specification of the relationship that associates the two related objects. Syntactically, direct units are expressed with the DIRECTUNIT element, as shown in the following example, which expresses the connection between an album and its unique artist:

```
<DIRECTUNIT id="ToArtist" relation="Album2Artist"/>
```

Figure 2 includes two direct units. The former (ToArtist) connects each album to its (single) artist; the latter (ArtistBio) connects two data units over the same artist, one showing only a short presentation, the other including all biographic information.

3.7 Pages

The granularity of units may be too fine for the composition requirements of an application, which normally demand that the information contained in several units be delivered together (e.g., the data of an artist and the index of the albums he has published). To cope with this requirement, WebML provides a notion of page. A page is the abstraction of a self-contained region of the screen, which is treated as an independent interface block (e.g., it is delivered to the user independently and in one shot). Examples of concrete implementations of the abstract concept of page in specific languages may be a frame in HTML or a card in WML.
Pages may be internally organized into units and/or recursively other pages. In the latter case, sub-pages of a container page are treated as independent presentation blocks (similarly to the notion of frames within frame sets in HTML). Nested sibling sub-pages may be in conjunctive form (i.e., displayed together) or in disjunctive form (i.e., the display of one sub-page is alternative to the display of another sibling sub-page). AND/OR sub-pages permit one to represent many complex page structures occurring in practice. The simplest case occurs when a portion of a page is kept fixed (e.g., the left frame in an HTML page), and another portion may display variable information based on user commands (e.g., the information in the right frame may be replaced by different data after a user's click in the left frame).
Syntactically, the organization of units into pages is specified using the PAGE element, as shown in the XML fragment of Figure 8.a, where a page portion (the sub-page named "leftmost") contains the indexes of past and recent issues, and the remaining portion (the sub-page named "rightmost") displays album information. The graphic notation and possible HTML rendition of this XML specification is also illustrated in Figure 8.a. Note the that pages are shown as dashed boxes around their enclosed units and/or sub-pages.

```
<PAGE id="outermost">
<PAGE id="leftmost"><UNIT id="pastIndex"/><UNIT id="thisYearIndex"/></PAGE >
<PAGE id="rightmost"><UNIT id="AlbumInfo"/></PAGE >
</PAGE>
```

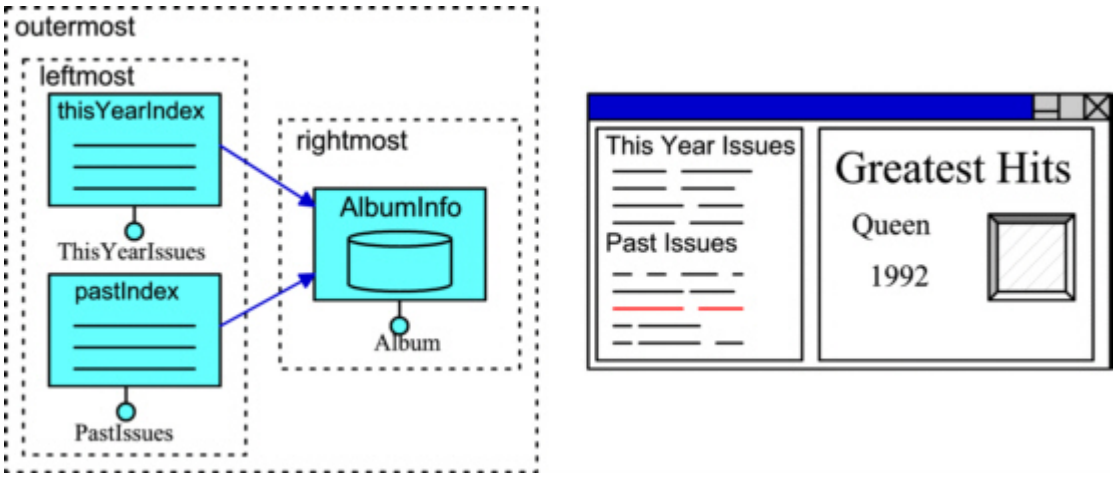

Figure 8.a - WebML textual and graphic notation for nested AND pages

Suppose now that we want a page to include the index of albums and artists, together with the information of either the album or the artist. This requires the introduction of alternative sub-pages, which is done in WebML using the ALTERNATIVE element. The XML specification in Figure 8.b describes the needed page structure. Note that since the page composition (and not only the object to display) changes, if we select an artist or an

album from the indexes, the ALTERNATIVE element is required to specify which alternative sub-pages should be used to display artist and album information. The graphic notation and possible HTML rendition of the XML specification are also shown in Figure 8.b.

```
<PAGE id="outermost">
 <PAGE id="leftmost"> <UNIT id="artistIndex"/>  <UNIT id="albumIndex"/>  </PAGE >
 <PAGE id="rightmost">
   <ALTERNATIVE>
     <PAGE id="rightmost1"> <UNIT id="artistInfo"/> </PAGE >
     <PAGE id="rightmost2"> <UNIT id="albumInfo"/> </PAGE >
   </ALTERNATIVE>
 </PAGE >
</PAGE>
```

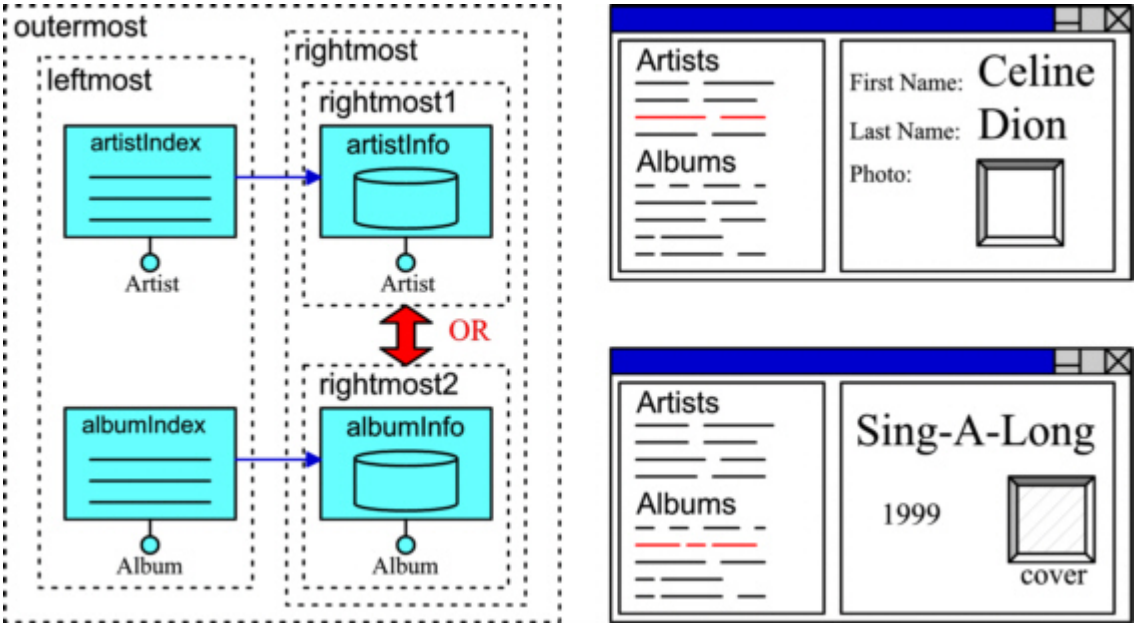

Figure 8.b - WebML graphic notation nested AND/OR pages, and a possible rendition in HTML

4. The navigation model

Units and pages do not exist in isolation, but must be connected to form a hypertext structure. The purpose of navigation modeling is to specify the way in which the units and pages are linked to form a hypertext. To this purpose, WebML provides the notion of link. There are two variants of links:

1. Contextual links: they connect units in a way coherent to the semantics expressed by the structure schema of the application. A contextual link carries some information (called context) from the source unit to the destination unit. Context is used to determine the actual object or set of objects to be shown in the destination unit.
2. Non-contextual links: they connect pages in a totally free way, i.e., independently of the units they contain and of the semantic relations between the structural concepts included in those units. Syntactically, contextual and non-contextual links are denoted by element INFOLINK and HYPERLINK, respectively nested within units and pages.

The following example demonstrates the use of contextual links, by showing a piece of hypertext composed of three linked units: a data unit showing an artist's data, an index unit showing albums, and a data unit showing album's data.

```
<DATAUNIT id="ArtistUnit" entity="Artist">
   <INCLUDEALL/>
   <INFOLINK id="link1" to="AlbumIndex"/>
</DATAUNIT>
```

```
<INDEXUNIT id="AlbumIndex" relation="Artist2Album">
   <DESCRIPTION key="title"/>
   <INFOLINK id="link2" to="AlbumUnit"/>
</INDEXUNIT>

<DATAUNIT id="AlbumUnit" entity="Album">
   <INCLUDEALL/>
</DATAUNIT>
```

The ArtistUnit data unit, based on entity Artist, is linked via an INFOLINK to the index unit, which is based on the relationship ArtistToAlbum. Such index unit in turn is linked by a second INFOLINK to the AlbumUnit data unit, based on entity Album. The semantics of the above contextual links is that:

- Due to the first link (link1) a navigation anchor is added inside the artist's data unit by means of which the user can navigate to the index unit listing all the albums of a specific artist.
- Due to the second link (link2), a set of navigation anchors (one per each entry in the index) is added to the index unit by means of which the user can navigate to one of the listed albums.

Context information flows along both links. The identifier of the artist whose albums are to be listed in the index unit flows from the source to the destination of the former link (link1). The identifier of the selected album flows from the source to the destination of the second link (link2), to determine the object shown in the data unit. Figure 9.a shows the WebML graphic notation for representing the above contextual links and a possible rendition of such piece of hypertext in an HTML-based implementation. In this example, each unit is placed in a separate page, therefore three distinct HTML pages are generated. Grouping units within pages and establishing contextual links are two orthogonal design primitives, as demonstrated by figure 9.b, where units ArtistUnit and AlbumIndex are kept on the same page. By linking data units and container units it is possible to obtain a variety of navigation modes, as shown in figure 9.c and 9.d where the index and album data unit are replaced by a multidata unit showing all albums of an artist together, both on the same page of the artist (case c), and in a separate page (case d).

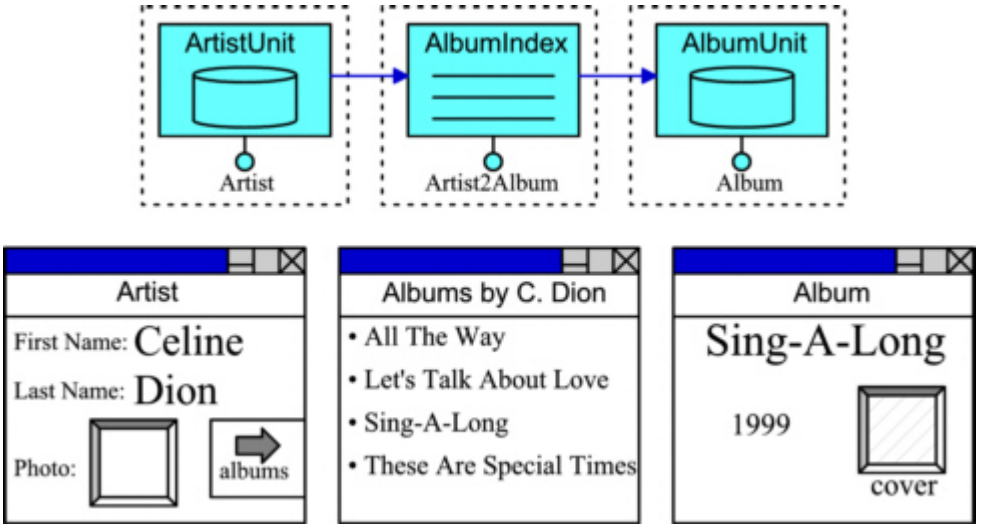

Figure 9.a - Index-based navigation (index in a separate page), and a possible rendition in HTML

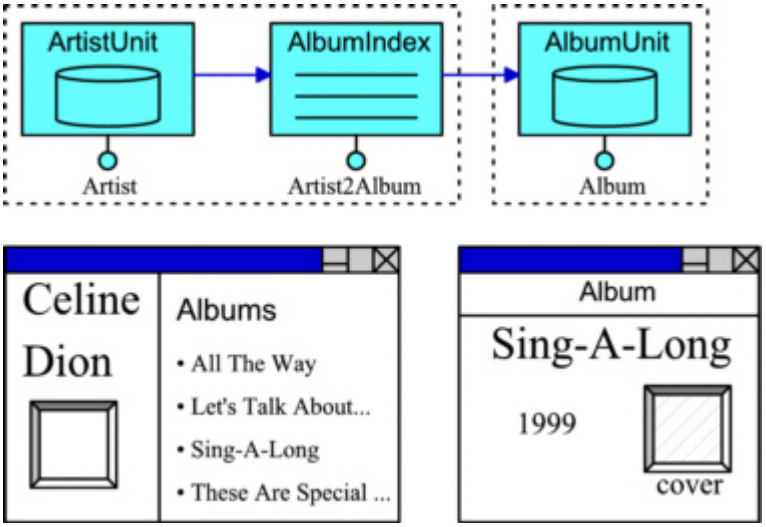

Figure 9.b - Index-based navigation (index in the source page), and a possible rendition in HTML

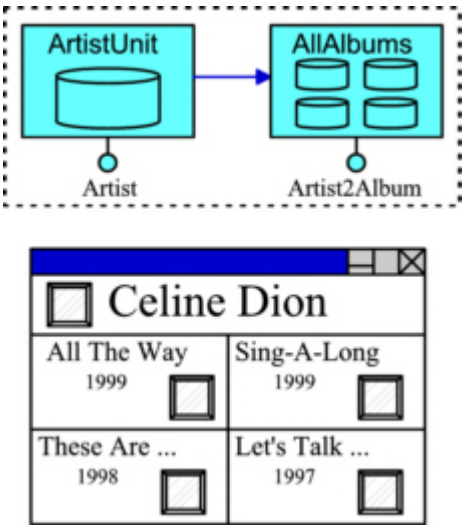

Figure 9.c - Composite page including one data and one related multidata unit

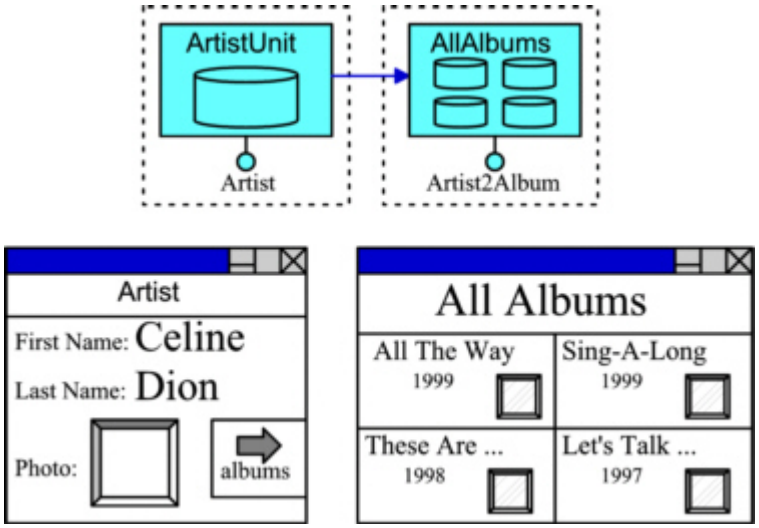

Figure 9.d - Separating the page including the artist data unit and the page including the multidata unit showing all the artist's albums

The following rules summarize the context information that flows out of a unit through a contextual link:

- Data Units: the identifier of the object currently shown in the unit.
- Index Units: the key value selected from the index list.
- Scroller Units: the identifier of the object selected by using the scrolling commands.
- Filter Unit: the attribute values given in input by the user in the data entry form.
- Direct units: the key value of a single object.
- Multidata unit: the context information associated with the data units nested within the multidata unit.

Non-contextual links are demonstrated by the example of Figure 10, where the page of an artist is linked to a separate, unrelated page, which contains the index of all albums. In this case, no context information flows along the link, because the content of the unit in the destination page (AllAlbums) is totally independent of the source page of the navigation. Note that, to underline the absence of context flow between units, non-contextual links are drawn between pages.

```
<DATAUNIT id="ArtistUnit" entity="Artist">
   <INCLUDEALL/>
</DATAUNIT>
<INDEXUNIT id="AllAlbums" entity="Album">
   <DESCRIPTION  key="title"/>
</INDEXUNIT>

<PAGE id="ArtistPage">
   <UNIT id="ArtistUnit"/>
   <HYPERLINK id="link1" to="AllAlbumsPage"/>
</PAGE>
<PAGE id="AllAlbumsPage">
  <UNIT id="AllAlbums"/>
</PAGE>
```

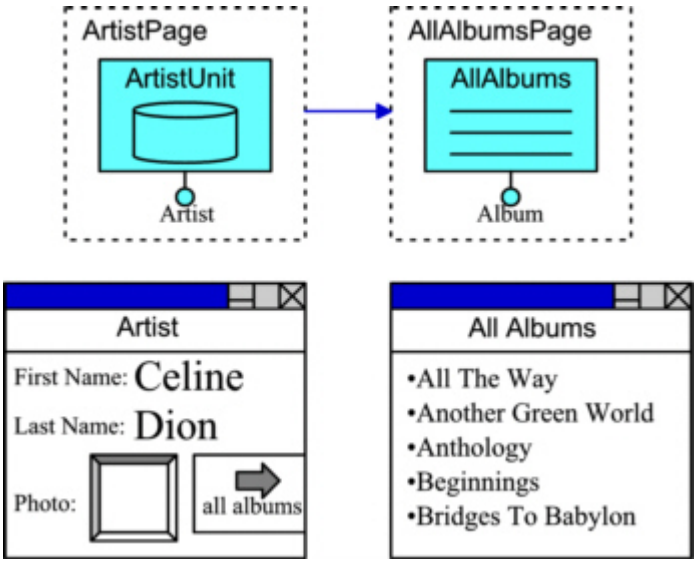

Figure 10 - WebML notation for non-contextual links

4.1 Hidden Navigation

In many sites, the interaction between the user and the application is proactive in two senses: not only the user chooses what content to see by clicking on hyperlinks, but sometimes also the system autonomously determines which page to show, by "anticipating" the effect of some user clicks. This feature can be modeled in WebML, by expressing the "filling semantics" of pages containing multiple units. For example, the user may access a page, which contains two units: an index unit over an entity pointing to a data unit on that entity. In this case, the content of the pointed data unit is "pending", i.e., it depends on the user's choice of one element in the preceding index unit. WebML offers three alternatives to cope with pending units: 1)

Leaving the pending unit empty, so that the user must explicitly perform a selection in one or more preceding units to display the content of the pending unit. 2) Filling the pending unit with a predefined default value (e.g., the first element chosen from a preceding index unit). 3) Filling the pending unit using with a default value expressed by means of a declarative query (e.g., the object of a preceding index unit that satisfies a given predicate).

Syntactically, the treatment of a pending unit is specified by choosing one of the above three options as the value of an ad hoc filling attribute, located in the "pointing" unit. If no value is specified, the pending unit is left empty.

4.2 Navigation chains and "Web patterns"

The typical configuration of a structured hypertext alternates data units, showing information on objects, with units that support the navigation from one object to another related object. Figure 9 shows two elementary forms of such a configuration, where an index unit and a multidata unit are used to move from an Artist to his/her Albums. WebML units and links can be composed to express more complex navigation structures, where multiple intermediate pages support the navigation towards a data unit; we call these configurations "navigation chains". Frequently used navigation chains are sometimes referred to as "Web patterns" (see Section 6 and [3,13,16,17]). In this section, we briefly present a selection of representative examples of navigation chains, to show how WebML concepts can be composed to formally describe a wide variety of situations occurring in practice.

Figure 11 shows an example of a navigation chain called multi-step index. A sequence of index units is defined over a given entity, such that each index unit specifies as its description key one of the attributes forming the key of the destination object. As shown in the figure, the semantics of this pattern is a hierarchical index, where the final object is located by means of a multi-step selection of its key value.

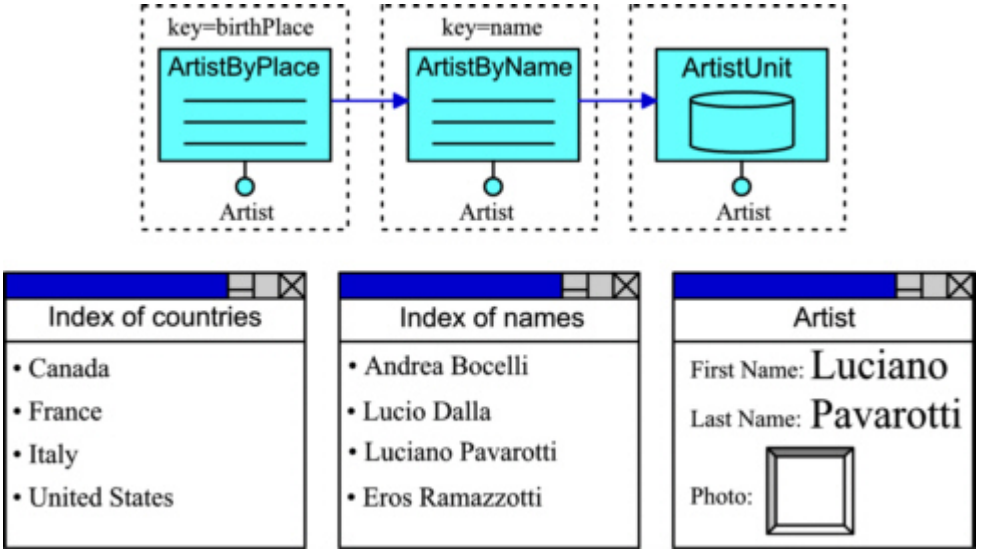

Figure 11 - WebML graphic notation for multi-step index, and a possible rendition in HTML

Figure 12 shows an example of a navigation chain configuration called filtered index. A sequence formed by a filter unit followed by an index unit is defined over a given entity. As shown in the figure, the semantics of this pattern is a three-step selection. First, the user provides input values to use as a search condition, then the objects matching such condition are presented in an index, finally the user may choose his object of interest from the (smaller) set shown in the index.

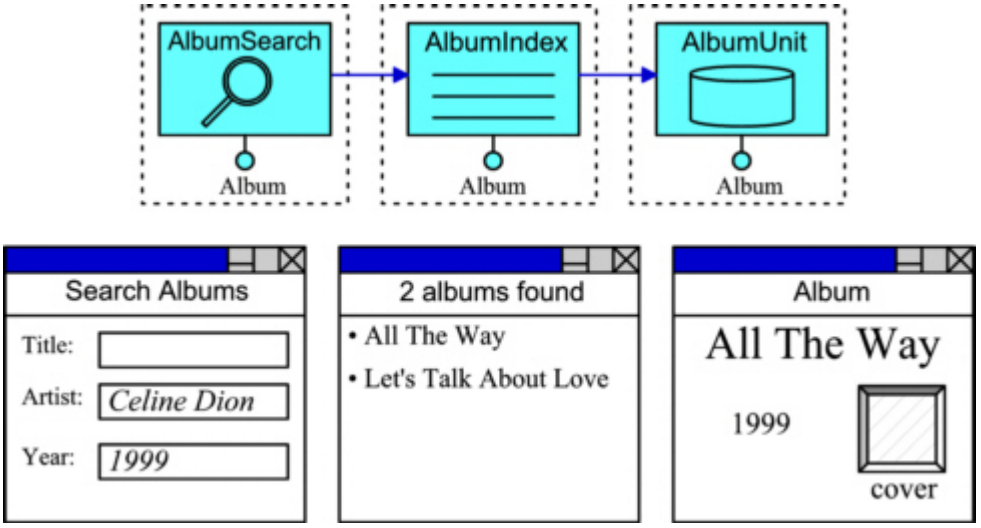

Figure 12 - WebML graphic notation for filtered index, and a possible rendition in HTML

Figure 13 shows an example of a navigation chain configuration called indexed guided tour. The configuration includes an index unit and a scroller unit which both are linked to the same data unit; in this case, the index and scroller units are synchronized: when the user performs a selection on either of them, the context of the other unit is changed so as to reflect the user's selection. Usually, the user chooses his object of interest from the index, then he is presented the selected object together with commands to access the first, last, previous, next in the sequence, and thus he can explore the adjacent objects of the given one.

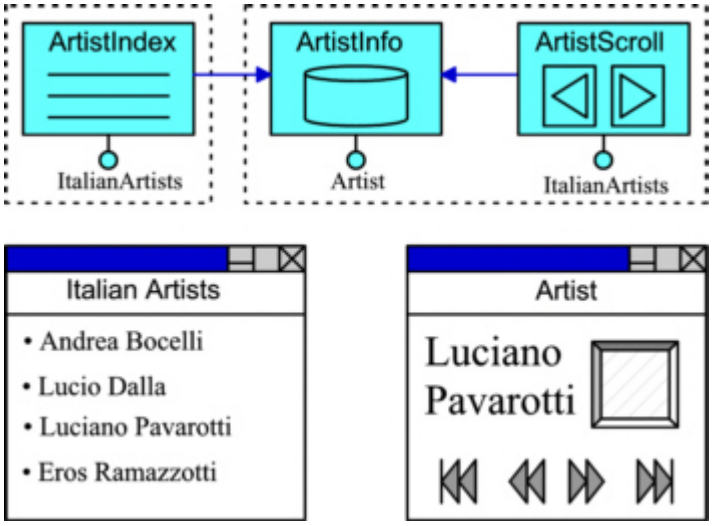

Figure 13 - WebML graphic notation for indexed guided tour, and a possible rendition in HTML

As a conclusive example, Figure 14 shows a ring. In this configuration, two data units are linked via a direct unit defined over the identity relationship (i.e., the predefined relationship linking each object to itself). The two data units show different attributes of the same object (e.g., a long and a short presentation) thus enabling multiple views of the same item with variable details.

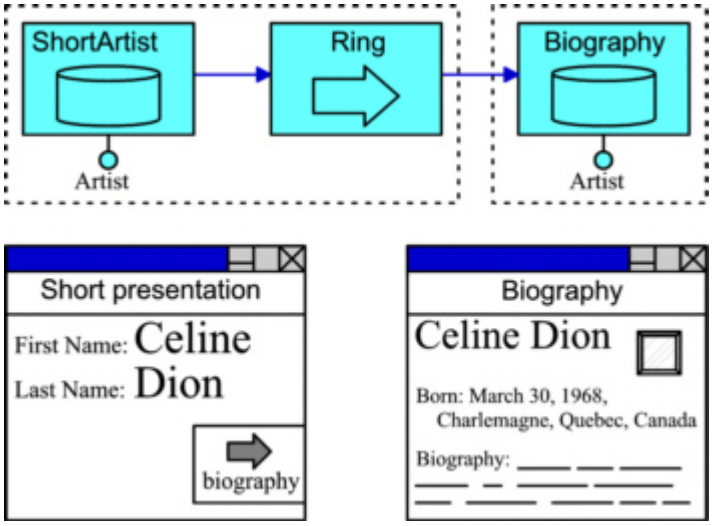

Figure 14 - WebML graphic notation for ring, and a possible rendition in HTML

4.3 Site Views

The separation between the structure model and the hypertext model advocated by WebML enables the definition of multiple views of the same data, which can be tuned to meet complex access requirements. For instance, different site views can be designed for alternative access devices or for distinct user groups. A WebML site view comprises a set of pages and links, with associated presentation styles, all reachable from a designated home page. All the site views of a given Web application share the same structural schema, which represents at high level the data sources underlying the site. The structural schema, in turn, is mapped to one or more data sources, possibly embodied within legacy systems.

4.4 Default hypertext

To enable fast prototyping, WebML includes several shortcuts to obtain a running application from incomplete specifications. In particular, given the structural model, WebML supports the notion of default hypertext, automatically generated according to the following rules:

- For each entity, a data unit is generated which includes all attributes.
- For each one-to-many or many-to-many relationship R between an entity A and an entity B, an index unit P is provided over the relationship R, based on the primary key of the entity B; two contextual links are established from the data unit of A to P and from P to the data unit of B.
- For each one-to-one or many-to-one relationship R between an entity A and an entity B, a direct unit P is provided over relationship R; two contextual links are established from the data unit of A to P and from P to the data unit of B.
- For each component C of an entity A, a multi-data unit P is provided over component C, and a contextual link is established from the data unit of A to P.
- For each entity, an index unit is created on the entity's primary key, which includes the list of all instances of the entity.

The default hypertext maps every concept of the structural model into exactly one unit and provides default indexes over all the defined entities. Given the default hypertext, a default site view is defined by associating units to pages as follows:

- Data unit over entities and index pages over relationships are put in distinct pages.
- Component multi-data units are kept in the same page as the data unit of the entity or component that encloses them.
- Index units over entities are put in distinct pages. An empty page is created as the home page, and connected by means of non-contextual links to the all these pages.

Figure 15 illustrates the default site view for the structure schema of Figure 1.

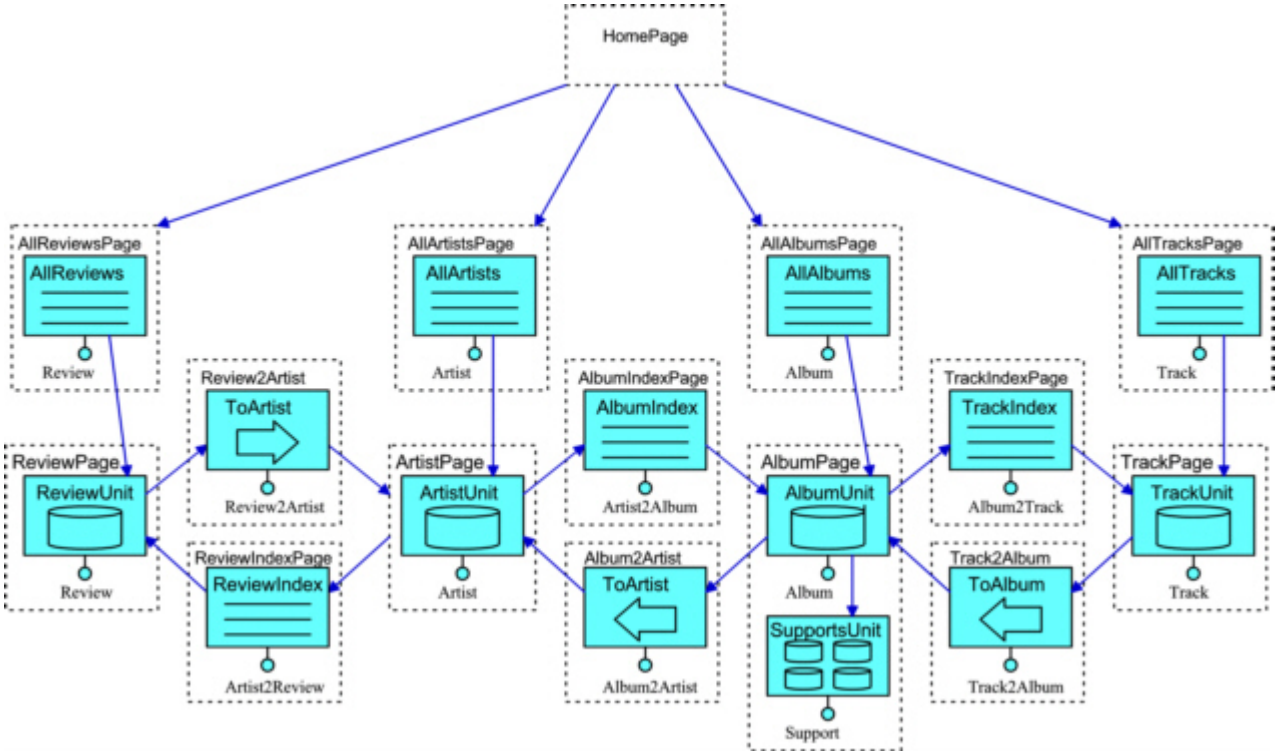

Figure 15 - A default site view

4.5 Validity of WebML hypertexts

Not all the WebML specifications obtained by linking units and by clustering them into pages correspond to conceptually correct and practically implementable Web sites. First, we give a collection of rules for the progressive construction of correct logical hypertexts (units connected by contextual links) and then rules for verifying physical hypertexts (how units are clustered into pages).

A valid logical hypertext is defined by the following constructive rules:

1. A logical hypertext constituted by a navigation chain over an entity followed by a data unit on such entity is valid.
2. The logical hypertext obtained by adding a linked sequence of container units over a relationship or component of the structural schema, from a data unit of a valid hypertext to another data unit (an existing one or a new one) is valid.

Figure 16 illustrates an invalid logical hypertext, made of a contextual link between two data units upon the entities Artist and Album; the hypertext is invalid because the album to be shown after following the link is undefined.

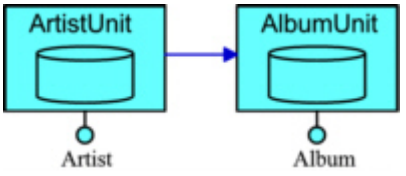

Figure 16 - Example of invalid logical hypertext

A second notion of correctness is based on the definition of valid physical hypertext and checks that the aggregation of units into pages produces application screens whose content is well defined. Given a valid logical hypertext, a valid physical hypertext is obtained by grouping units within pages and adding non-contextual links between pages according to the following rules:

- Reachability: there must not be pages (with the exception of the home page) without any incoming link (contextual or non-contextual);
- Context flow: if a page contains a unit that needs context information, then such context information must be supplied by a contextual link. There are two sub-cases:
    1. The unit may receive context from another unit in the same page that does not require context information (e.g., an index unit over an entity).
    2. If the above case does not hold, the unit must receive its context from all the entry units in the same page, where an entry unit is any unit which is the destination of contextual links coming from outside the page.
- Uniqueness of context: if a unit in the page has more than one path from which it receives context information from another unit inside the same page, then only for one such paths the initial filling option should be enabled.

Note that physical hypertext validity does not restrict the presence of non-contextual links between pages. These can be placed at will, without hampering the site correctness.

Figure 17 (left part) illustrates an invalid physical hypertext: the Album page includes the ArtistInfo unit, which requires context information to determine the album to display, but is not linked to any other unit. On accessing the page by means of the link into the AlbumInfo unit, the content of the TrackIndex unit is well defined, but the content of the ArtistInfo unit is not, because this unit has no incoming contextual link supplying the identifier of the artist to show. Figure 17 (right part) illustrates a second potentially invalid physical hypertext, in which the AlbumInfo unit has two incoming contextual links, one from the index unit showing all albums of an artist, the other one from the index unit showing only this year's albums. If both indexes specify a non-empty initial filling option, the content of the destination data unit may be not uniquely defined upon accessing the page.

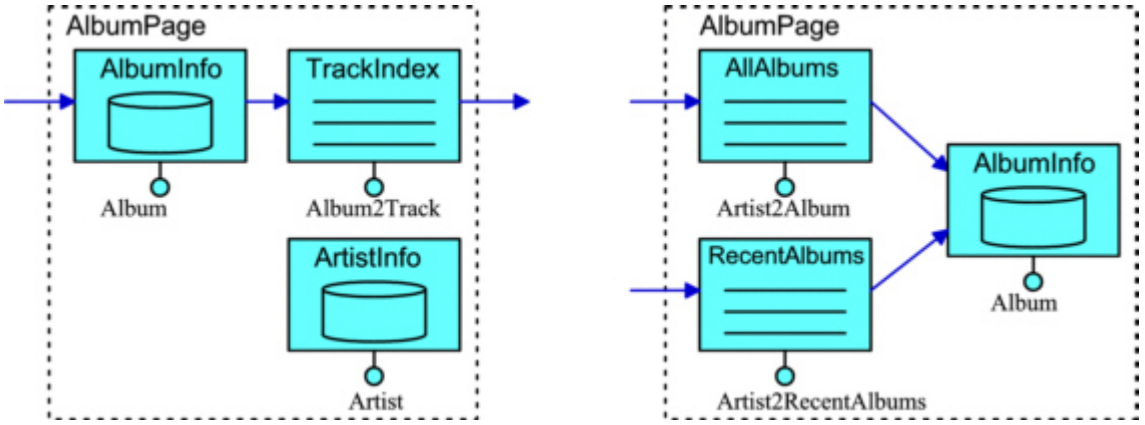

Figure 17 - Two examples of invalid physical hypertext

5. Other features of WebML

We next present briefly the other features of WebML; for greater detail, we refer the reader to [7] and to the W3I3 project's documentation available at the site http://www.toriisoft.com.

5.1 Derivation

Derivation is the process of adding redundant information to the structure schema, in order to augment its expressiveness. Derivation in WebML is expressed in a formal query language, which is a restricted version of OQL [5]; it is possible to derive entities, attributes, components, and relationships. An example of derivation is in the definition of the attribute currentPrice of the component Support shown in Section 2.

5.2 User Modelling

In order to support personalization, WebML includes an explicit notion of group and user. Groups describe sets of users with common characteristics, whereas users denote individuals. Users always belong to at

least one group, possibly the default one (called everyone). Users may belong to more than one group, but in this case they are required to provide a default group or to indicate one preferred group when accessing the site. Each user or group is described by means of specific properties (collectively called profile), modeled as special type of entities in the structure schema. As the normal entities, user and group profiles may be internally sub-structured into attributes and components, classified by means of inheritance, semantically related to other entities, and used for writing derivation queries. Typically, profiles include user- or group-specific data (e.g., the most frequently or recently visited objects, the list of the last purchases), whose content is progressively modified during the Web site evolution as result of user's actions.

5.3 Declarative and Procedural Personalization

Personalization is the definition of content or presentation style based on user profile data. In WebML, units, pages, their presentation styles, and site views can be defined so to take user- or group-specific data into account. This can be done in two complementary ways:

- Declarative personalization: the designer defines derived concepts (e.g., entities, attributes, multi-valued components) whose definition depends on user-specific data. In this way, customization is specified declaratively; the system fills in the information relative to each user when computing the content of units.
- Procedural personalization: WebML includes an XML syntax for writing business rules that compute and store user-specific information. A business rule is a triple event-condition-action, which specifies the event to be monitored, the precondition to be checked when the event occurs, and the action to be taken when the condition is found true. Typical tasks performed by business rules are the assignment of users to user groups based on dynamically collected information (e.g., the purchase history, or the access device), the notification of messages to users upon the update of the information base (push technology), the logging of user actions into user-specific data structures (user tracking), and so on.

As an example of declarative personalization, the computation of an album's discounted price could be based on personalized discounts, associated with users or user groups. As an example of procedural personalization, a business rule could assign a customer to the "best buyer" group, based on his/her purchase history. WebML declarative and procedural personalization are discussed in greater details in [7].

5.4 Presentation model

Presentation modeling is concerned with the actual look and feel of the pages identified by composition modeling. WebML pages are rendered according to a style sheet. A style sheet dictates the layout of pages and the content elements to be inserted into such layout, and is independent of the actual language used for page rendition. For better reusability, two categories of style sheets are provided: untyped style sheets (also called models) describe the page layout independently of its content, and thus can be applied regardless of the mapping of the page to a given concept; typed style sheets are specified at a finer granularity and thus apply only to pages describing specific concepts.

5.5 Updates and operations

WebML is currently being extended to support a new type of pages for performing operations and updating the site content. Write access modeling is achieved by extending the current set of WebML concepts with the introduction of operation units, a novel type of unit whereby users can invoke operations on the site. An operation unit specifies the operation to be performed, and is linked to other units, which may show the alternative results of performing the operation (e.g., in case of success or failure). A set of generic update operations (such as: insert, delete, modify for entities, and drop, add for relationships) are predefined and need not be declared. Parameters needed to perform the operation either come from the context flowing to the operation unit via an incoming link, or are supplied by the user via forms. Operation units generalize the notion of filter units, which can be regarded as a particular operation unit associated to a search operation over a set of objects. Thanks to the orthogonal nature of WebML, operation units can be freely combined with the other types of units to compose complex pages, mixing information to be presented to the user and interfaces to perform operations on the site. Operation units are currently being designed by looking at trolleys and online purchase procedures used in the most sophisticated e-commerce sites.

6. Previous Related Work

WebML builds on several previous proposals for hypermedia and Web design language, including HDM, HDM-lite, RMM, OOHDM, and Araneus. Its design principles are stated in [6]; WebML as presented in this paper is a (quite radical) evolution of an earlier version presented in [7].

HDM [14] pioneered the model-driven design of hypermedia applications and influenced several subsequent proposals like HDM-lite [12], a Web-specific version of HDM, RMM [15], Strudel [11], and OOHDM [18]. All these methods offer powerful built-in navigation constructs, as opposed to WebML, which includes simple, yet orthogonal, composition and navigation primitives. Araneus [2] is a recent proposal for Web design and reverse-engineering, in which the data structure is described by means of the Entity Relationship Model and navigation is specified using the Navigation Conceptual Model (NCM). Conceptual modeling is followed by logical design, using the relational model for the structural part, and the Araneus Data Model (ADM) for the navigation aspects. Araneus also includes predefined navigation primitives and does not model presentation at an abstract level.

The relevance of using conceptual modeling techniques in the context of the WEB was addressed in a specific workshop [8]. In particular, [18] describes an evolution of OOHDM (Object-Oriented Hypermedia Design Method), a methodology for designing hypertexts that shares with WebML the vision of orthogonal design dimensions; specifically, OOHDM is concerned with the conceptual modeling, navigation design, interface design, and implementation. Navigational contexts in OOHDM provide a rich repertoire of fixed navigation options.

Given that WEB applications are, after all, software artifacts, it is not surprising that several proposals exist for using UML [4] for their specification. In particular, [10] shows how the architecture of Web applications can be modeled in UML. Web pages are modeled as UML components, distinguishing among their "server side aspects" (i.e., their relationship with middle tiers, databases, and other resources) and their "client side aspects" (i.e., their relationships with browsers, Java applets, ActiveX controls and so on). This distinction of roles and the use of stereotypes enables a quite effective modeling of WEB applications using standard UML notations, however the article does not show Web-specific evolutions of the notation, which are described as ongoing and traceable on the UML and Rose sections of http://www.rational.com.

A closer approach to WebML is described in [1]; the article presents UIML, a user interface language designed for abstracting from appliance-specific details while designing the user interface of a Web application. UIML designers model all aspects of a user interface in XML, so that only a portion of the specification (the style section) is appliance-dependent. Although WebML approaches the broader context of web conceptual modeling, UIML and WebML's presentation model share the goal of obtaining independence of specifications from output devices and the use of XML as a vehicle to achieve this goal.

Recent articles have shown increasing interest on model-driven design of WEB applications. In particular, several authors have addressed the importance of using patterns for describing navigation across WEB sites [3,13,16,17]. Several examples in Section 4.2 show how the design patterns described in [14,17] can be formally described, and indeed WebML is a very effective description language for WEB patterns.

7. Conclusions

In this paper, we have presented the core of WebML, a high-level specification language for designing data-intensive Web applications. With respect to previous proposals, WebML: 1) stresses the definition of orthogonal navigation and composition primitives, which the designer can arbitrarily compose to model complex requirements; 2) includes an explicit notion of site view, whereby the same information can be structured in different ways to meet the interests of different user groups or to obtain a granularity optimized for users approaching the site with different access devices; 3) covers advanced aspects of Web site modeling, including presentation, user modeling, and personalization.

WebML is the backbone of Toriisoft, an environment for the computer-aided design of Web sites currently in an advanced development state. In particular, the Toriisoft tool suite comprises Site Designer, for editing the WebML specifications of the structural, hypertext, and personalization models; Presentation Designer, for

visually defining presentation style sheets; Site Manager, for site administration and evolution. The architecture is completed by a Template Generator, which transforms WebML specifications into Microsoft's Active Server Page (ASP) templates running on top of relational DBMSs for data storage. Code generation is based on standard XML technology (XSL) and therefore Toriisoft can be easily extended to support template generation in more than one markup language and for multiple server-side scripting engines. Work is ongoing on the translation of WebML specifications into WML-based ASP templates, thereby providing evidence that the model-driven approach of WebML is particularly effective in supporting multi-device Web sites.

Acknowledgements

WebML is the result of research work done in the context of the W3I3 Esprit Project sponsored by the European Community. We wish to thank all W3I3 participants for the helpful feedback on the definition of the various WebML constructs. In particular, thanks to David Langley, Petra Oldengarm, Wim Timmerman, Mika Uusitalo, Stefano Gevinti, Ingo Klapper, Stefan Liesem, Marco De Michele, Fabio Gurgone, Alessandro Agustoni, Simone Avogadro, Marco Brioschi, and the innumerable POLI students who spent their time in the project.

References

[1] M. Abrams, C. Phanoriou et. al.: UIML: an Appliance-independent XML User Interface Language, Proc. WWW8, Elsevier, pp. 617-630.
[2] P. Atzeni, G. Mecca, and P. Merialdo: Design and Maintenance of Data-Intensive Web Sites. Proc. EDBT 1998, pp. 436-450.
[3] M. Bernstein: Patterns of Hypertexts, Proc. ACM Int. Conf. On Hypertext 1998, ACM Press, pp. 21-29.
[4] G. Booch, I. Jacobson, and J. Rumbaugh, The Unified Modeling Language User Guide, The Addison-Wesley Object Technology Series, 1998.
[5] R. G. G. Cattell, Douglas K. Barry, and Dirk Bartels (Eds.), The Object Database Standard : ODMG 2.0, Morgan-Kaufmann Series in Data Management Systems, 1997.
[6] S. Ceri, P. Fraternali, S. Paraboschi: Design Principles for Data-Intensive Web Sites, ACM Sigmod Record, 27(4), Dec. 1998, pp.74-80.
[7] S. Ceri, P. Fraternali, S. Paraboschi: Data-Driven, One-To-One Web Site Generation for Data-Intensive Applications. Proc.VLDB 1999, pp. 615-626.
[8] P. P. Chen, D. W. Embley, and S. W. Liddle eds, Proc. Int. Workshop on the World Wide Web and Conceptual Modeling (Paris, Oct. 1999) Springer-Verlag, LNCS 1727.
[9] P. P. Chen, The Entity-Relationship Model, Towards a Unified View of Data, ACM-Transactions on Database Systems, 1:1, 1976, pp. 9-36.
[10] J. Conallen: Modeling Web Application Architectures with UML, Communications of the ACM, 42:10, Oct. 1999, pp. 63-70.
[11] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, D. Suciu: Catching the Boat with Strudel: Experiences with a Web-Site Management System. Proc. ACM-SIGMOD Conference 1998, pp. 414-425.
[12] P. Fraternali, P. Paolini: A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications, Proc. EDBT 1998, pp. 421-435.
[13] F. Garzotto, P. Paolini, D. Bolchini, and S. Valenti: "Modeling by patterns" of Web Applications, in [8], pp.293-306.
[14] F. Garzotto, P. Paolini, D. Schwabe: HDM - A Model-Based Approach to Hypertext Application Design. TOIS 11 (1): 1-26 (1993)
[15] Tomas Isakowitz, Edward Stohr, P. Balasubramanian: RMM: A Methodology for Structured Hypermedia Design. CACM 38(8): 34-44 (1995).
[16] M. Nanard, J. Nanard, and P. Kahn: Pushing Reuse in Hypertext Applications Development, Proc. ACM Int. Conf. On Hypertext 1998, ACM Press, pp. 11-20.
[17] G. Rossi, D. Schwabe and F. Lyardet: Improving Web information systems with navigational patterns, Proc. WWW8, Elsevier, pp. 589-600.
[18] G. Rossi, D. Schwabe, F. Lyardet: Web Application Models are More than Conceptual Models, in [8], pp.239-252.

Vitae

Stefano Ceri is full professor of Database Systems at the Dipartimento di Elettronica e Informazione, Politecnico di Milano; he has been visiting professor at the Computer Science Department of Stanford University between 1983 and 1990. His research interests are focused on: data distribution, deductive and active rules, and object-orientation design methods for data-intensive WEB sites. He is responsible of several projects at Politecnico di Milano, including W3I3: "Web-Based Intelligent Information Infrastructures" (1998-2000). He was Associate Editor of ACM-Transactions on Database Systems (1989-92) and he is currently an

associated editor of several international journals, including IEEE-Transactions on Software Engineering. He is author of several articles on International Journals and Conference Proceedings, and is co-author of the books: Distributed Databases: Principles and Systems (McGraw-Hill, 1984) Logic Programming and Databases (Springer-Verlag, 1990) Conceptual Database Design: an Entity-Relationship Approach (Benjamin-Cummings, 1992) Active Database Systems (Morgan-Kaufmann, 1995) Advanced Database Systems (Morgan-Kaufmann, 1997) The Art and Craft of Computing (Addison-Wesley, 1997) Designing Database Applications with Objects and Rules: the IDEA Methodology (Addison-Wesley, 1997) Database Systems: Concepts, Languages, and Architecture (McGraw-Hill, 1999).

Piero Fraternali is associate professor of Software Engineering at the Dipartimento di Elettronica e Informazione, Politecnico di Milano. His research interests are focused on: active rules, object-orientation, design methods for data-intensive WEB sites, CASE tools for automatic Web site production, and wireless applications. He is author of several articles on International Journals and Conference Proceedings, and is co-author of the book: Designing Database Applications with Objects and Rules: the IDEA Methodology (Addison-Wesley, 1997). He is the technical manager of the W3I3 Project : "Web-Based Intelligent Information Infrastructures" (1998-2000).

Aldo Bongio graduated at Politecnico di Milano in 1999, where he presently coordinates the development of the ToriiSoft Web Site Design Tool Suite. His research interests include XML, Web modeling languages, and Web design patterns.