Building Multi-device, Content-Centric Applications Using WebML and the W3I3 Tool Suite

Angela Bonifati[1], Stefano Ceri[1], Piero Fraternali[1], and Andrea Maurino[1]

Politecnico di Milano,Piazza L. da Vinci 32
20133 Milano, Italy
{bonifati,ceri,fraterna,maurino}@elet.polimi.it

Abstract. In the forthcoming years, two factors will jeopardize the deployment of Web applications: supporting multi-device outputs and one-to-one personalization. These two factors will lead to an explosion of solutions, to be developed, maintained, and kept consistent; meanwhile, Web hosting companies will be subject to growing service demands and will be lacking the technical man-power required to master them. With these premises, the strength of the W3I3[1] tool suite is to propose a model-driven approach to Web site design. Such an approach is based on WebML, a high-level language for specifying the structure of the content of a Web application and the organization and presentation of such a content in a Web site. In this paper, after a brief presentation of WebML, we concentrate on the W3I3 tool architecture, shown at work on case-study based on the popular site http://www.softseek.com.

1 Introduction

Designing data-intensive Web sites, i.e. sites whose primary purpose is the publishing of large volumes of data, is a primary concern for many companies. This challenge is going to become more demanding in the close future, because the activity of designing, deploying and evolving sites will face the need of serving content simultaneously to a variety of individuals or user groups, possibly equipped with different devices, each one characterized by specific rendition capabilities. In particular, WAP-compliant cellular phones, featuring WML-enabled micro-browsers [14], are already spreading in the market.
The W3I3 tool suite addresses personalized and multi-device content deployment by leveraging three different aspects of its architecture:

1. The possibility of organizing content at a high-level, using the WebML conceptual site modeling language ([5], http://webml.org). Alternative forms of content composition can be expressed as **site views**, and each site view

[1] W3I3 (Web-Based Intelligent Information Infrastructures) is a project funded by the EC, involving four companies and one Academic Institution (Politecnico di Milano) from four European countries

S.W. Liddle, H.C. Mayr, B. Thalheim (Eds.): ER 2000 Workshop, LNCS 1921, pp. 64–75, 2000.

may cluster information and services at the granularity most suitable to a particular class of users and devices.
2. The availability of an abstract presentation language, by which it is possible to construct reusable page descriptions (called **style sheets**) independent of the specific markup language required by the user's device. Style sheets specify pages in terms of content elements arranged in a nested grid model. They are written in XML [15].
3. The XSL-enabled translation technology [16,17], which maps abstract XML page specifications into concrete code in the languages of choice. The choice of language regards both the presentation, in which a specific markup language is selected (e.g., WML), and the binding of data to pages, where alternative server-side scripting languages can be used (e.g, Microsoft's Active Server Pages).

In this paper, after a brief presentation of WebML, we focus on the W3I3 tool suite architecture and on its individual components; we next show the tools at work in the modeling of an existing Web site (http://www.softseek.com).

2 The WebML Site Specification Language

WebML [5] is a high-level specification language allowing designers to express the core features of a site and abstracting them from architectural details. WebML concepts are represented in an intuitive graphic fashion, which can be easily supported by CASE tools and is conceived for non-technical members of the site development team (e.g., graphic designers and content producers). WebML internally relies on an XML syntax, which can be fed into software generators for automatically producing the implementation of a Web site. The specification of a site in WebML addresses four orthogonal perspectives: the structural model, the hypertext model, the presentation model, and the personalization model.

2.1 Structural Model

WebML does not propose yet another language for data modelling, but is compatible with classical notations like the E/R model [6], the ODMG object-oriented model [4], and UML class diagrams [3]. The fundamental elements of the WebML structural model are *entities* - acting as containers of data elements - and *relationships* - enabling the semantic association between entities. Entities have named properties, called *attributes*, with an associated type; properties with multiple occurrences can be represented by means of *multi-valued components*, which express a part-of relationship. Additional classical ingredients of conceptual models are present in WebML: *generalization hierarchies* for entities and *cardinality constraints* for relationships. An example of structural model for the SoftSeek case study is described in Section 4 and shown in Figure 2.

2.2 Hypertext Model

The hypertext model includes suitable constructs for representing one or more hypertexts, which can be published on top of the information described by the structure model. Each different hypertext defines a so-called site view; **site view** descriptions in turn consist of two sub-models, which are respectively called **composition** and **navigation** models.The composition model specifies which pages form the hypertext, and which content units (the atomic information elements that may appear in the Web site) make up a page. WebML content units are: data, multi-data, index, filter, scroller and direct units. Data units are used to publish the information of a single object (e.g., a software item), whereas the remaining types of units represent alternative ways to browse a set of objects (e.g., by presenting a subset of them in the same page, or by presenting an index, a search filter, first/last/previous/next scrolling commands, or finally by giving a direct access to a specific single element). Composition units are mapped to entities or relationships of the structural schema, from which they draw their content. The navigation model expresses how pages and content units are linked to form a hypertext. Links are either non-contextual, when they connect semantically independent pages (e.g., the page of an article 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 (e.g., the list of download sites associated to a given software item). Contextual links conform to the structure schema, because they connect content units whose underlying entities are associated by relationships in the structure schema. Examples of hypertext model for the SoftSeek case study are shown in Figure 3 and 4.

2.3 WebML Presentation Languages

Presentation is the modeling perspective concerned with the appearance of pages on the screen. WebML specifies presentation at the conceptual level, i.e., independently of the particular instance to be presented and on the specific rendition language. The basic unit of presentation is the **page**, as defined in composition modeling. Each page is associated to one or more **style sheets**, each specifying a different way of presenting its instances on the screen. Style sheets are XML documents obeying the WebML presentation DTD, which can be defined visually by means of a tool called Presentation Designer. The WebML presentation DTD includes tags for layout and content modeling.

The layout of each style sheet is a bi-dimensional rectangular space (represented by element space2d shown in Figure 1), which may include a set of possibly overlapping regions. Each region can be organized into a grid having an arbitrary number of rows and columns; each cell of the grid can recursively contain other regions, which can in turn be organized as nested grids. Cells of a grid are defined as the intersection of row and column ranges; therefore they may correspond to macro-cells made of several elementary cells forming a rectangular area. After layout definition, the next step in style sheet definition is to specify which piece

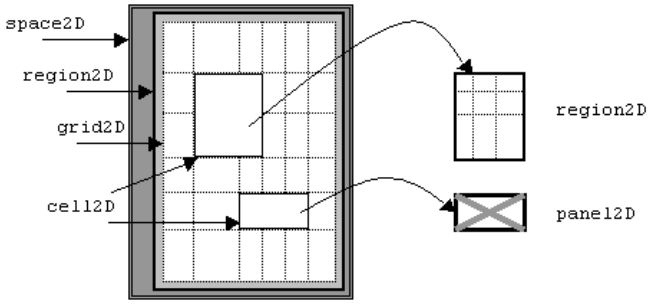

Fig. 1. Main XML elements pf WebML presentation DTD

of content goes into the various regions placed at the bottom of the hierarchical structure of the page (typically those contained in the cells of some grid). Content elements are specified using panels, i.e. XML fragments specifying an atomic or composite content element (e.g., a piece of text), which can be inserted into a region.
An example of page presentation applied to the running case of the SoftSeek site is detailed in Section 4 and shown in Figure 5.
Various rendition languages (HTML 3.2, HTML 4, WML, etc.) can be used to concretely implement the abstract presentation of pages. WebML presentation model can be extended to represent the peculiar aspects of a rendition language by including additional language-specific properties or constraints, which affect layout elements and panel templates. These extensions are again expressed in XML and collected in a document called *language profile*. For instance, a "background image" property for such layout elements as grids and cells is part of the language profile for HTML, but is not available in the profile for WML.

2.4 Personalization Model

Users and user groups are explicitly modelled in the structure schema in the form of predefined entities called User and Group (automatically provided by the site design tools, once a new project is created). The features of these entities can be used for storing group-specific or individual content, like shopping suggestions, list of favourites, and resources for graphic customisation, which can be published in a site view as normal content. In addition, OQL-like declarative expressions can be added to the structure schema, to define derived content based on the profile data stored in the User and Group entities, e.g., a discounted price based on the user's shopping history. Moreover, business rules can be generated in order to change the site's content according to user-specific information or to compute user profile data.

3 A Tool Architecture for Personalised, Multi-device Application Generation

WebML conceptual modelling is backed by a software architecture, which supports all the steps necessary to transform the WebML specification of a site into a running application in spectrum of industrial-strength rendition languages and server-side application platforms. The architecture of the WebML tool suite consists of three software layers: (a) the design tools, used for collecting design specifications (b) the device-specific template generators (c) the platform-specific adapters.

- The **design tools** support the modeling of Web applications at the conceptual level; the output of the design layer is a **conceptual schema** of the application, coded in WebML.
- The **code generation** layer transforms the conceptual schema of the application into an intermediate representation suitable for processing on top of commercial Web-database systems. This intermediate representation consists of a collection of **page templates**, which embody the structure, navigation, composition, and presentation of the application, but do not include to the actual data. Page templates are bound to a specific delivery language (e.g., HTM 3.2 or WML), and to a specific scripting language to be interpreted at server side (presently, MicroSoft's Active Server Pages, and JavaSoft's Java Server Pages).
- The **run-time adapters** consists of a set of lightweight Java components installed at the server-side, which give access to the actual data structures, which host the content of the entities and relationships defined at design time. This software layer shields the generated templates from the query language needed to bind the actual data to page when serving user requests. Presently W3I3 includes runtime adapters for wrapping JDBC compliant relational DBMSs and LDAP repositories.

3.1 Design Environment

The Design Environment includes three tools:

- **Site Designer**: permits the designer to define the structural model and the hypertext model of the application. Complex functions like the specification of derived data or the creation of an initial default site view are simplified by wizards, which, for instance, allow users to write OQL-like queries for expressing derived data in a visual way.the tool implements an advanced functions for user support, which perform the syntactic and semantic check of the project graph; if mistakes are detected, warning messages and tips are automatically presented to the user, which explain what is wrong and how to fix the problems
- **Presentation Designer**: deals with the specific aspect of presentation specification.The designer may define both generic and page-specific layouts. A

generic style sheet (also called presentation model, or untyped style sheet) is a specification of a page in terms of layout and fixed content elements (e.g., logos, fixed texts or images), which are independent of the specific objects used to fill the page. A **page-specific style sheet** (also called typed style sheet), instead, describes a page layout at a more detailed level, mentioning the actual elements (data fields, outgoing links, indexes, search forms, and so on) included in a certain page. Presentation Designer includes the support of multiple languages also.

– **Site Manager**: supplies all the required functions for publishing a W3I3 site on top of the runtime layer and data sources, and for maintaining it. These features are **site creation**,which invokes the Template Generator (see below) which builds the page templates necessary to run the application. the **site publishing** function is used to move all the application resources to the deployment server. The **user management** function addresses the specification of the access rights. Finally, Site Manager includes a **mapping function** for declaring the association between the structural model concepts and the repository structures chosen for the storage of data

The above tools are integrated by means of a further component, called **Repository Manager**, which manages the communications with all clients co-ordinate their access to the Central Design Repository, which hosts the WebML specifications in the form of XML documents and the graphic resources used in style sheets.

3.2 Template Generation

The Template Generator transforms a style sheet into a give rendition language (presently, HTML3.2 and WML are supported). The first use of the Template Generator is at design time to obtain a preview of the style sheet under construction. A **preview function** (launched from Presentation Designer) processes the XML specification of the style sheets, fetches the page characteristics from the design repository, and outputs a static file in the markup language of choice, in which the data content of the page is mocked-up (e.g., the value of attributes of type image are replaced by a reference to a constant image file). The second use of the Template Generator is at publication time, when the pages and style sheets of the site are transformed into ASP or JSP templates including instructions in a server-side scripting language for accessing the real data from the runtime data sources.
The Template Generator implements a multi-step process for transforming a WebML style sheet into a page template in a specific mark-up and server-side scripting language pair. The translation proceeds according to the following steps:

1. **Unfolding**: the original style sheet may contain composite panels and references to sub-pages, whose layout is described in a separate style sheet. The unfolding phase fetches all the necessary panel template and style sheets

definitions, and recursively replaces composite panels and sub-pages with their layout specification. The result is an unfolded style sheet equivalent to the original one but consisting only of atomic panels (images, texts, and anchors).
2. **Layout annotation** and optimisation: the unfolded style sheet is traversed to compute auxiliary information (e.g., the coordinates of the starting and ending point of all cells) and to apply optimisation operators to the unfolded layout (e.g., the removal of unnecessary nesting levels introduced by the procedure for recursive unfolding). At this stage, the process is still independent of the mark-up and server side scripting language.
3. **Data reference** translation: abstract data references contained in panels are converted from the WebML syntax to the syntax of the chosen server-side scripting language (e.g., Visual Basic Script). At this stage, the partially translated style sheet is bound to a specific server side platform.
4. **Mark-up translation**: the partially translated style sheet is fed to an XSL processor, which applies to it a set of rules, contained in an XSL file designed for the specific output mark-up language. The XSL file includes templates for mapping the WebML abstract layout tags into the most suitable constructs of the chosen rendition language. The output of mark-up translation is the final template, ready to be installed in the deployment server.

The core benefit of the described architecture is flexibility: a new rendition language can be easily added without changing the implementation. All the language dependent features are expressed in an XML-based syntax and the only tasks to be performed for integrating new languages are the creation of a new language profile (which requires the editing on an XML file), the creation or extension of panel templates to introduce language-specific properties (XML-based too) and the addition of an XSL file giving the rules for the mark-up conversion.

4 The SoftSeek Case-Study

As an example of WebML-driven site design, we now show how the popular SoftSeek Web Site (http://www.softseek.com) can be modelled using the WebML tools, and re-engineered to obtain a version of the same content accessible via a WAP-enabled mobile phone. The SoftSeek Web Site allows searching, downloading and accessing documentation about software products. The software items are classified into several groups (editor's picks, top downloads, spotlight products, new releases, and so on) and are clustered in categories; each category has a name and a brief textual description, and can contain further sub- and sub-sub-categories. Categories and subcategories relate to spotlight and top products, whereas the sub-sub-categories include the complete listing of all products featured in that sub-sub-category. Each product is characterized by a set of technical data (e.g., version, size, release date, sample screenshot, descriptive text etc..), and is connected to the product's supplier, to related products from the same author, and to a set of download sites. Figure 2 shows the information published in the SoftSeek site represented as entities (categories, software items,

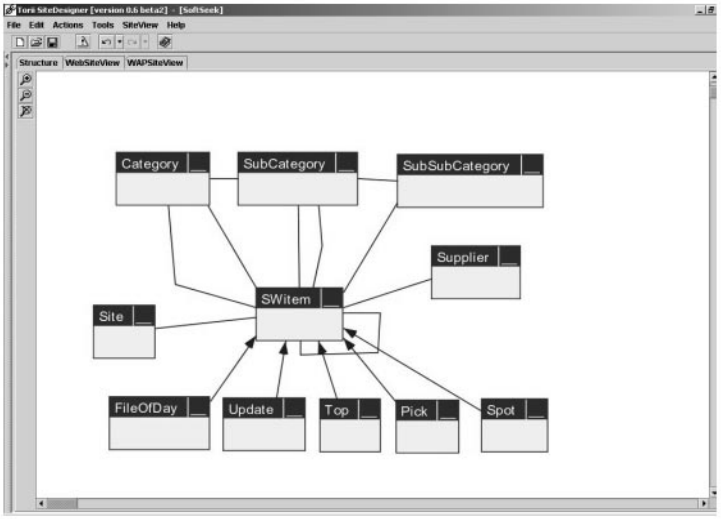

Fig. 2. Structural model of the case study edited with Site Designer

suppliers, download sites) and relationships (categories to subcategories, software items to download sites, suppliers and software items of the same supplier). An inheritance hierarchy represents classification of special software items.

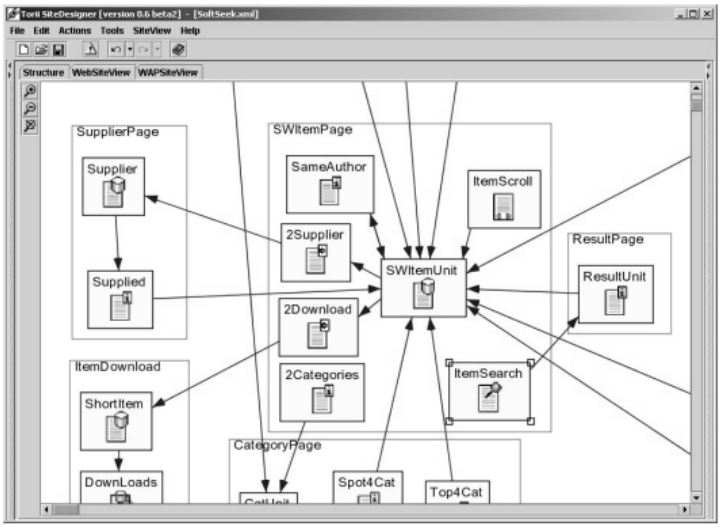

Fig. 3. An excerpt from the SoftSeek Web site view modeled in Site Designer

After consolidating the structural model, the hypertext model is designed: a different site view must be defined for the different devices,to cluster information differently based on the capability of each medium. Figure 3 shows a portion of the site view for the web version of the SoftSeek application. Due to space limitation,

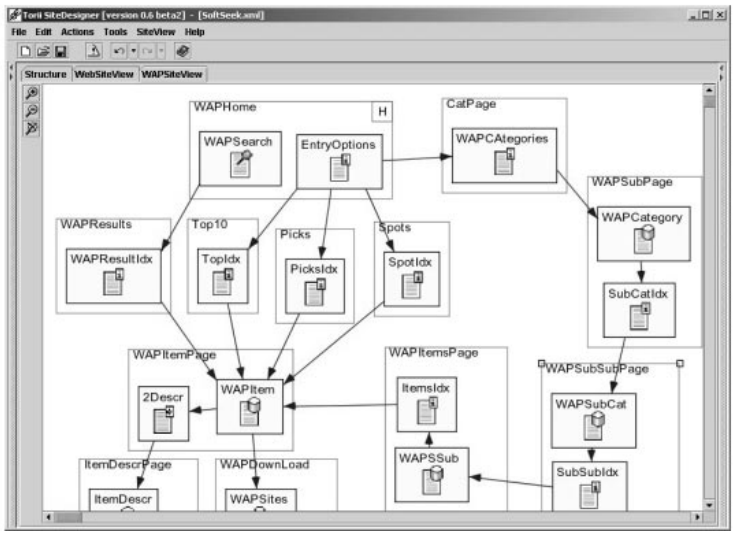

Fig. 4. Example of WAP site view for the SoftSeek case study

it is impossible to describe the complete schema and thus we concentrate on the design of the most representative pages only. At the top of the site view diagram, page *SWItemPage* describes the core information of a product.

Its center is the *SWItemUnit* data unit, which holds the product data (name, version, size, description text, image, and so on). *SWItemUnit* is linked via two direct units to the single supplier of the product (unit *Supplier* in *SupplierPage*), and to an alternative product page (*ItemDownload*), which includes a short description of the product (data unit *ShortItem*) and the set of sites wherefrom the product's file can be downloaded (multidata unit *Downloads*). *SWItemUnit* is also linked to the index of the other products by the same author (unit *SameAuthor*), from which it is possible to move to the page of another item, and to a scroller unit (*ItemScroll*), which permits the reader to move to the previous and next item in the same sub-sub-category. *SWItemPage* also contains an index of all the top-level categories (the index unit *2Categories*) and a search form (unit *ItemSearch*) to locate a product by keyword.

As an example of the support offered by the W3I3 tool suite to the specification and management of multi-device applications, figure 4 shows a second site view constructed on the same structure model of Figure 2, but aimed at

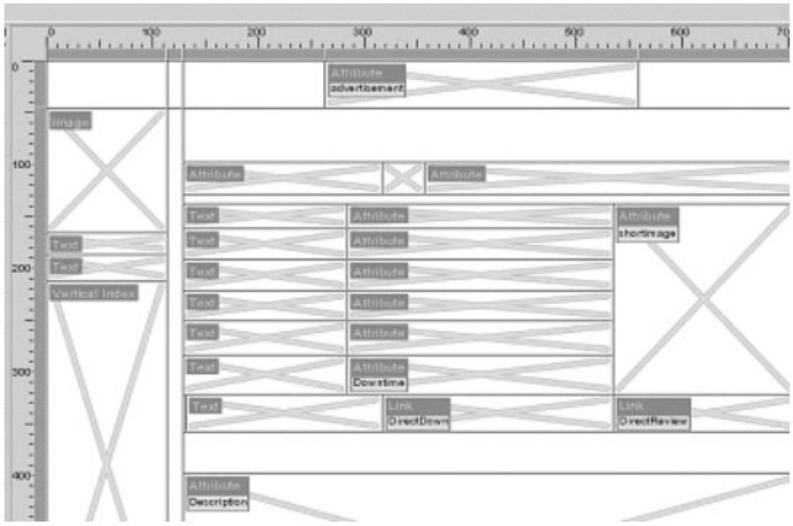

Fig. 5. Style sheet for the SWItemPage constructed in Presentation Designer

WAP devices. The WAP view is characterized by a finer granularity and less navigation options than the Web view, because wireless devices have a smaller display and thus entities must be split on different pages, and only the essential navigation facilities can be effectively used. Finally, we show the last step of application design in W3I3, which requires the use of Presentation Designer to define style sheets for the site view pages. Two sets of style sheets are required, one for PC browsers and one for WAP devices. Figure 5 shows a screenshot from the construction of the layout of the Web *SWItemPage*: different atomic and composite panels embedding the content elements of the page are arranged in a grid-based layout. The page was generated as so to be very similar to

Fig. 6. Output of the WML Template Generator on WAPItemPage

the actual page for showing software items in the softseek application(e.g. see www.softseek.com/Programming/C/Review 32546 index.html). Figure 6 show a wml page, obtained by running the Template Generator on a styel sheet edited

in Presentation Designer using the WML language profile. Both page templates contain mark-up and sever side scripting code generated in a totally automatic way by the W3I3 tools.

5 Related Work

An array of tools is developed by the industry and academia to create and manage data-intensive Web sites. For brevity, we only present a selection of products that leverage some form of model-driven design, while we refer to [9] for a complete survey. **Designer 2000** [10] is a CASE environment included into the Oracle platform for deploying Web application based on a Web-enhanced entity-relationship design. Its Web Generator delivers PL/SQL code, which runs within the Oracle Web Server to produce HTML pages. Designer 2000 adopts a very database-centric approach while the W3I3 architecture adopts a mix of data, hypertext, and presentation modelling. In **Strudel**, a research project developed at AT&T Labs [7], Web sites are created from the declarative specification of the site's structure and content, in the form of queries over a data model for semi-structural information.**Homer** [2] is a CASE tool for building and maintaining data intensive Web site developed by researches of Università di Roma tre. The site schema is described by a formal model called AMD [1], which mixes database and hypertext concepts. Like the W3I3 tool suite, Homer can generate output for HTML and XML/XSL format, but Homer does not address presentation design. In the field of multi-device site development, Oracle's **Portal-To-Go**, [12] is a new server product that enables any existing database and Internet application to be made accessible from WAP phones, PDAs and other mobile devices. Portal-to-Go makes existing Internet or database applications device-independent, by extracting their content, dynamically converting it to XML, and then to the mark-up language supported by the user's device, including WML, TinyHTML, and VoxML. Portal-to-go, unlike W3I3, does not rely on a model-driven approach, and could be used as a runtime layer from the W3I3 tool suite, by adding an XSL translator targeting the Portal-To-Go XML DTD.

6 Conclusions

W3I3 tools have been applied to the modeling of a large number of case studies and applications, both in the context of the user companies of the W3I3 project, and by graduate students of our department. The W3I3 approach guarantees the following advantages:

- Increased productivity of Web developers. The use of a high-level, model-driven approach coupled to fully automatic code generation facilitates the design and thus lowers the technical edge of Web developers, alleviating an essential problem of most Web hosting companies.
- Lower ownership costs. The use of a model-driven approach eases maintenance and evolution, because changes can be analyzed at a higher level and

propagated to the implementation. This feature is essential in view of the exponential growth of complexity of Web sites caused by the interplay of multi-device output and one-to-one delivery.
– Higher consistency across applications delivered to different output devices. We envision Web applications that offer a consistent and stateful interaction to users who, e.g., initially connect from home (e.g., via digital TV), then access the application while traveling (by means of a mobile device), then in the office (via personal computer). Interaction uniformity is guaranteed by W3I3's content-centric design, where information is modeled once and then adapted to different media and deployed automatically.

All the key features of the W3I3 tools are fully implemented, and the final version of the W3I3 tool suite will be available by the end of the W3I3 project (October 2000).

References

1. P.Atzeni, G. Mecca, P. Merialdo, To Weave the WEB, VLDB 1997
2. P. Merialdo, P. Atzeni, M. Magnante, G. Mecca, M. Pecorone: Homer: a Model-Based CASE Tool for Data-Intensive Web Sites. SIGMOD Conference 2000
3. G. Booch, I. Jacobson, and J. Rumbaugh, The Unified Modeling Language User Guide, The Addison-Wesley Object Technology Series, 1998.
4. 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
5. Ceri, Fraternali, Bongio, Web Modeling Language (WebML): a modeling language for designing Web sites, WWW9 conference, Amsterdam, 15-19 May 2000
6. P. P. Chen, The Entity-Relationship Model, Towards a Unified View of Data, ACM-Transactions on Database Systems, 1:1, 1976, pp. 9-36.
7. Fernandez, Florescu, Levy, Suciu, Catching the boat with Strudel: Experiences with a Web-Site Management System. In Haas and Tiwary eds, Proc. Int. Conf, Sigmod 1998
8. P, Fraternali, P. Paolini, Model-Driven Development of Web Applications: the Autoweb System, to appear into Transaction on Information System, 2000
9. P. Fraternali, Tools and Approaches for Developing Data Intensive Web Application: a Survey. ACM Comp. Surveys, Volume 31, No. 3 (Sep. 1999),
10. Oracle Designer/2000 WebServer Generator Technical Overview, Oracle Cor.
11. Lightweight Directory Access Protocol (v3), http://www.cis.ohio-state.edu/htbin/rfc/rfc2251.html , 1997
12. Oracle portal-to-go http://www.oracle.com/mobile/portaltogo/index.html
13. M. Abrams, C. Phanoriou et. al.: UIML: an Appliance-independent XML User Interface Language, Proc. WWW8, Elsevier, pp. 617-630.
14. Wireless markup language http://www.wapforum.org, Wap Forum Ltd.
15. Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/1998/REC-xml-19980210, 1998,
16. Extensible Stylesheet Language (XSL) 1.0, http://www.w3.org/TR/xsl/, 2000
17. XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt, 1999