Design Principles for Data-Intensive Web Sites

Stefano Ceri, Piero Fraternali, Stefano Paraboschi
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza Leonardo da Vinci 32, I-20133 Milano, Italy
ceri,fraterna,parabosc@elet.polimi.it

1 Introduction

The integrated design of the Web interface and of data content gives several advantages in the design of data-intensive Web sites. The main objectives of this design process are (a) associating the Web with a high-level description of its content, that can be used for querying, evolution, and maintenance; (b) providing multiple views of the same data; (c) separating the definition of information content from Web page composition, navigation, and presentation, which should be defined independently and autonomously; (d) storing the meta-information collected during the design process within a repository used for the dynamic generation of Web pages; (e) collecting information about the Web site usage, obtained both statically (user registration) and dynamically (user tracking); (f) supporting selective access to information based on users' requirements and needs; (g) using business rules to improve the generation of effective Web pages and to present each user with personalised views of the Web site.

We identify ten general principles that should be considered when implementing a Web site managing large amounts of data, yielding the following decalogue:

- I: Data-intensive Web sites should have a conceptual schema
- II: Data derivation should be supported
- III: Data navigation should be supported
- IV: Flexible page composition should be supported
- V: Flexible presentation styles should be supported
- VI: Web metadata should be managed by commercial DBMSs
- VII: Page and data materialization could be supported
- VIII: Associative query paradigms could be supported
- IX: Personalization could be supported
- X: Event-based reactive processing could be supported

The first five rules represent methodological issues that should be considered in the design of a site. Rules VI and VII are relative to the implementation of the site. Finally, the last three rules describe special functionalities that are needed in most, but not all, data-intensive Web applications.

2 Terminology

Before discussing each principle in detail, we try to clarify some of the terms that will be used in this paper.

Data-intensive Web application. Web sites are data-intensive when their primary goal is to make large amounts of data accessible to a variety of users. A typical application scenario is electronic commerce on the Web.

Structured and semi-structured data. We assume that data resources used by a data-intensive Web application are mostly structured, i.e., they are regular enough to be described by means of a schema. In addition, data-intensive Web applications may include semi-structured data; these should normally be self-describing, i.e., with local descriptors providing names and types for each data item. Models such as Lorel [1] or content description languages such as XML go in this direction.

Design process for a data-intensive Web application. It is the process by means of which Web sites are generated; it consists of the joint specifications of the Web pages which make up the application and of the DBMS structure and queries for data extraction and manipulation. We distinguish between a pure *top-down design process*, in which the data is defined together with the Web site, and a pure *bottom-up design process*, in which the structure and content are defined a priori and are immutable by the process. Many cases lie in between these two extremes, as some data content pre-exists to the design process, but data sources can be enhanced and/or restructured to take new requirements into account.

Structural model. The structural model of a Web application defines the content of the structured data which is used by the application. It should be independent from the logical and physical models used for data storage, which may be multiple heterogeneous DBMSs. Concepts used in such description are classical ones (e.g., entities with slots and lists, relationships between entities, generalization and part-of hierarchies).

Derivation. Derivation is the process of extracting data from the various sources and producing data for specific Web pages. There are two processes of data derivation. *External derivation* is the process of extracting data from the sources and producing data as described by the structural model; *internal derivation* is the process of deriving as many conceptual views of the data as needed by the various possible interfaces supported by the Web application.

Navigation. Navigation is the process by means of which it is possible to move from one page to another in a Web application. Navigation is dynamically executed by performing certain operations, associated to events on the screen (e.g., the click on suitable icons or pieces of text), which have a well-defined operational semantics.

Presentation. Presentation is concerned with the look and feel of Web pages, in particular with the design of the general page layout, with the placement of specific pieces of information on the page, and with the selection of graphical resources like backgrounds, icons and animations.

3 The Decalogue

I: Data-intensive Web sites should have a conceptual schema

Structured data should be given a conceptual schema, that describes the information content of data resources regardless of their storage structures. The conceptual model provides a better understanding of the information content of the Web, and such understanding can improve the processes of Web site generation, use, maintenance, interoperability with external data sources, query, and evolution. With suitable user interfaces, the conceptual schema can be disclosed to users and help in content analysis and query formulation.

Conceptual data models for the Web should be suitable adaptations of conceptual data models as already in use in other disciplines, such as database design, software engineering, and knowledge representation. The ingredients of such conceptual models should be the standard abstractions of classification, aggregation, and generalization. In a very broad sense, the elements of conceptual models are **containers** of data elements or **connectors** enabling the linking of data elements; data elements have **slots**, with an associated type. For instance, in the well-known Entity-Relationship (ER) model entities are containers, relationships are connectors, and attributes are slots. Additional classical ingredients of conceptual models include *generalization hierarchies*, *part-of relationships*, and *cardinality and/or existence constraints*.

A useful notion in the conceptual modeling of the Web is that of **target**, which intuitively represents a class of objects or facts of the real world that should be given an autonomous Web presentation. Targets should be identified in such a way that any piece of information of the conceptual schema belongs to a given target; target identification is a classical step in object-oriented design. Each target could correspond to several containers, connectors, and slots; in ER terms, each target may include several entities, with their attributes, and with relationships between them. Other relationships are instead used to connect different targets.

II: Data derivation should be supported

External derivation is the process of building the data content, as specified in the structural schema, from data sources. Derivation can be fully automatic if the data source is a single DBMS and the DBMS schema is automatically derived from (or mapped into) the structural schema. Otherwise, external derivation requires solving classical problems of data integration from diverse, possibly heterogeneous, data sources.

Internal derivation is the process of producing different viewpoints of the same information. Derivation queries (rules) apply to the conceptual schema and define additional concepts, whose content is intensionally derived instead of being extensionally stored. In principle, every element of the conceptual model can be intensionally derived; e.g., it is possible to derive containers, connectors, and slots. Coherently with the principle that Web information should be targeted, also each derived concept should belong to a target.

III: Data navigation should be supported

As in classical hypertext theory, it is possible to distinguish navigation into contextual and non-contextual. The former is associated to the shift of focus from one piece of application content to another one along navigation paths coherent to the conceptual structure of the site (e.g., from a target to another one along an inter-target connector); the latter permits the access to the content of an application from outside by means of suitable access commands, e.g., entry indexes.

At a conceptual level, non-contextual navigation can be modeled by means of *collections*, defined as containers of pages not attached to any target.

Contextual navigation occurs within a container (accessing the members of the container) or along a connection (accessing all elements connected to a given one). It includes the following orthogonal aspects:

- *sorting*: the way in which the members of a container or the elements associated by a connector to an object are sorted;
- *filtering*: the possibility to select a subset out of all the members of a container or connector;
- *indexing*: the possibility to present a collective preview of the (filtered) members of a container or connector by means of an index;
- *accessing*: the actual shift of focus to one (or more) of the members of a container or connector;
- *browsing*: the possibility, when one element of a container or connector is accessed, to "scroll" through the other elements of the same container or connector.

IV: Flexible page composition should be supported

Page composition is the fundamental process of assigning data targets specified during structure modelling to abstract information nodes, called *page types*, which form the hypertext to be published on the Web. In such a mapping, it must be possible to flexibly select a subset of the target's data and semantic connections to be shown on the actual Web pages of the application.

Targets should be mapped to page types in a one-to-many fashion: the same target should be representable according to different page structures, but for coherence a page type should always describe a single target. At run-time, an instance of a target is mapped to an instance of a page type, which then causes the production of a physical page. Multiple page types per target allow the designer to represent the information on the same real-world object in different ways, e.g., for serving the needs of different users.

Page composition should also be concerned with non-contextual navigation, which is supported by the definition of *collections*, i.e., ad hoc containers of pages not attached to any target, serving as entry indexes to the application objects. An application should have many well-chosen collections, to let users access the application content in several different ways.

V: Flexible presentation styles should be supported

Presentation specification should be pursued at the conceptual level, i.e., independently of the specific network language used to render the application, to enable multiple mappings from the same page style to different realizations (e.g., based on Java, HTML 3, HTML 4, XML). An approach for defining the presentation of page types at the conceptual level is to consider the screen as an abstract grid and then to place into each cell of the grid the pieces of information that constitute the page type or collection page, e.g., the slots and outgoing connectors that characterize the target associated to the page type.

Presentation specifications should be collected in *presentation styles*, which should drive the production of actual Web pages (e.g., HTML pages). WYSIWYG tools may help in the creation of effective presentations, typically by presenting several predefined styles and then giving to the designer several options for customization. For flexibility, it should be possible to associate each page type to more than one style; defaults may be used for the automatic generation of a basic style for each page type. The output of actual application pages from presentation styles, page types and database targets should be defined in a formal way, to enable automatic implementation.

VI: Web metadata should be managed by commercial DBMSs

Metadata describing the structural schema, external and internal derivations, navigation, composition, and presentation styles should be stored within a repository; for its implementation, we advocate the use of standard DBMS technology. Given that data requirements, integrity constraints and usage patterns are well understood, the derivation of both logical and physical schemas for meta-information can be supported by an automatic translation process. This choice favors the integration of meta information with the data content.

VII: Page materialization should be supported

The process of page generation maps conceptual specifications plus database content into pages, typically written in HTML. It should be possible either to perform page generation just-in-time when a page is requested by a user, or to split it into sub-steps, some of which could be anticipated for better performance: (a) database views describing derived concepts could be materialized; (b) all data relative to the same page type could be clustered into one or more views and materialized; (c) HTML page templates corresponding to page types could be generated by embedding database queries into HTML tags, with the help of commercial SQL-HTML integrators; (d) output pages could be cached, either on the client, or on the server, or on a proxy.

Such anticipation, however, may cause disalignment between the database content and the data materializations; this problem is critical if the database is constantly updated and the Web application is expected to present always up-to-date values.

VIII: Associative query paradigms could be supported

Access to information stored in a data-intensive Web site could be provided by query paradigms based on the site's semantics. To let users query the site without knowledge of its physical organization, query languages should operate at the conceptual level, leveraging the abstractions of the conceptual model used to describe the site. Conceptual query languages should have a graphical interface, based on presenting the conceptual schema to users and then asking them to logically define navigations and predicates, so as to select specific conceptual elements.

IX: Personalization could be supported

In many applicative contexts, the Web site could be personalized with respect to specific classes of users. To this end, it is necessary to collect static information about users (typically by means of user's registration) and/or dynamic information about usage (by means of access traces or explicit requests). Based on this information, the user could be presented with different versions of the site, or with different navigation commands and presentation styles. For instance, a user could be explicitly asked for preferences and then be offered suitable entry indexes based on his input. Personalized access is essential in electronic commerce for supporting one-to-one marketing, an approach where each user is served by an apparently individual application interface.

Beside technical implications, the collection of profiles and traces has also legal requirements. For example, in the European Union privacy legislation imposes that personal information (e.g., traces) be always kept in a secure way, and its usage disclosed to the user and explicitly authorised.

X: Event-based reactive processing could be supported

The dynamic generation of Web pages has the advantage that all data manipulation events (e.g., data insertions, deletions, updates) can be immediately reflected by the Web site content; in this sense, dynamically generated Web sites are implicitly highly reactive.

The use of commercial DBMSs for storing metadata and content enables the use of active rules (triggers) for expressing domain-dependent business rules. Several applications are possible: (a) if users register their interests in certain data (targets), then the news relative to those targets can be automatically delivered to them (e.g., in the form of messages or data pushes) (b) if pages are automatically generated, then content and style selection can be done dynamically by triggers, which react to event-condition pairs expressing the understanding of users' interests and needs as explicitly or implicitly collected; (c) similarly, new navigation options and/or collections can be added to a page based on users' interests.

4 Classification of approaches and tools for data-intensive Web applications

Existing approaches and tools for the design of Web applications can be classified in four categories [5].

- Authoring tools.
- Web extensions of databases.
- Tools for database export, report writing, and application development.
- Integrated Web-database development environments.

Authoring tools include products like Microsoft's Front Page, NetObject's Fusion, Claris' Home Page, and many more. These tools typically focus on content production;

they offer powerful graphical design capabilities, but do not support the description of the structure and of the semantic properties of the information sources that are collected within the site and offer little or no database integration. From a development process standpoint, they mostly concentrate on the implementation and maintenance of sites, and the best tools also offer a rather rudimental support to design, limited to the definition of the hierarchical organization of the site's pages. The extraction of structured data stored in databases is not integrated in the site design process, and thus must be performed manually in ad hoc ways. In summary, these tools simplify the development of small-scale Web sites, but quickly become inadequate as the size and complexity of the Web application increase.

Web extensions of databases include a variety of tools whose common denominator is to produce Web pages from information stored in a database, either statically or dynamically. Several tools perform such task by integrating databases and Web technology at the language level, for example by extending HTML with special tags for dynamically embedding the output of database queries into static HTML page templates. Examples include: the Cold Fusion Web Database Construction Kit by Allaire Inc., Microsoft's Active Server Pages (ASP) and Internet Database Connector (IDC), StoryServer by Vignette Corporation, Informix's Web Integration Option, HAHT Software's HahtSite. Due to their focus on implementation languages, these products lack high level abstractions for describing applications and thus do not assist the developer in identifying the structure, navigation, composition, and presentation aspects of an application.

A different category of Web database extensions offer higher-level functionalities for *database export, report writing, and form-based application development*. Among the innumerable existing products there are: Microsoft's Visual InterDev, Visual Basic 5, and Access97, Borland's IntraBuilder, Sybase's PowerBuilder, Apple's WebObjects, NetDynamics, Asymetrix SuperCede Database Edition, and Allaire's Cold Fusion Application Wizards. These database publishing tools offer Integrated Development Environments (IDEs) and Rapid Application Development (RAD) tools for boosting productivity in the implementation phase. However, they do not encompass conceptual modeling nor support the model-driven design and implementation of applications. Typically, data structures, business logic, and interfaces are specified and designed separately and then implemented with a tool of choice.

Integrated Web-database development environments have the potential of satisfying all the rules of the decalogue. We have included into such category Oracle's Designer 2000 [7] as a representative of commercial products, and a few research prototypes, namely Araneus [2], AutoWeb [6], and Strudel[4]. These are more extensively described in light of our ten rules in the following sections.

4.1 The Oracle Web Development Tool Suite and Designer 2000

The Oracle Web Development Suite includes Designer 2000 [7], an environment for business process and application modeling, integrated with software generators originally designed to target traditional client-server architectures. The Web Generator enables previous applications developed with Designer 2000 and deployed on LANs to be ported to the Web, as well as the delivery of novel applications directly on the Internet or on Intranets. The Web Generator takes its inputs from the Designer 2000 design repository and delivers PL/SQL code that runs within the Oracle Web Server to produce the desired HTML pages of the application.

Designer 2000 adopts a development process for Web applications identical to that for traditional client-server database applications. The modeling abstractions offered by Designer 2000 are database-centric (mainly, tables and foreign-key links), and thus the designer can adequately represent the structure of the extensional and intensional information hosted in the site (Principles I and II), but is limited in the specification of navigation, page composition, and presentation styles (Principles III, IV, and V).

Performance is guaranteed by the tight integration between the Designer 2000 Web Generator and the Oracle Web Server, which manages metadata in the DBMS (principle VI) and uses page caching transparently to the user (Principle VII). Advanced functions like user modeling and profiling and business rules (Principles IX and X) are not integrated with the modeling notations of Designer 2000, but can be added manually to the application generated by the tool.

4.2 Araneus

Araneus is a project of Università di Roma Tre, defining an environment for managing unstructured and structured Web content in an integrated way, called Web Base Management System (WBMS). In a WBMS, database technology is used to store both data and metadata describing the hypertextual structure of Web sites. The major components of Araneus are [8]:

- A **conceptual data model**, called ADM (Araneus Data Model), used to represent the structure of the site's documents. ADM is a hypertextual data model based on the notion of *page scheme*, a language independent page description notation based on such elements as attributes, lists, link anchors, and forms.

- Several **languages and tools** for Web sites management and querying. Ulixes is a language for querying Web sites, which implements a navigational algebra. Polyphemus is a diagrammatic query interface for expressing Ulixes queries. Editor is a text-management language for the definition of wrappers, which are used for loading external data stored within semi-structured or unstructured data sources (e.g., existing Web sites). Minerva is an extension of Editor, based on a declarative specification of the grammar of the source text to parse and on an exception mechanism to manage inconsistencies in the sources. Penelope is the system for the definition and maintenance of new sites.

- A **methodology** for Web site design and implementation. The methodology is based on the distinction between data structure, navigation, and presentation. The structure of the application domain is described by means of the Entity Relationship model; the navigation aspects are specified using the Navigation Conceptual Model (NCM) [2], a notation inspired to RMM, simplified in several operational details. Conceptual modelling is followed by logical design, using the relational model for the structural part, and the Araneus Data Model (ADM) for the navigation aspects. Presentation is built starting from HTML files with appropriate tags.

Araneus focuses on data-intensive Web sites by adopting a database-centric process. Development proceeds according to a structured process organized along two tracks: database and hypertext. Database design and implementation are conducted in the customary way using the Entity-Relationship Model and mapping it into relational structures. After ER modeling, hypertext conceptual modelling formalizes navigation by turning the ER schema into a NCM schema; this shift requires several design activities. The next step, hypertext logical design, maps the NCM schema into several page-schemas written in ADM. Finally, implementation requires writing page-schemas as declarations in the Penelope language, which specifies how physical pages are constructed from logical page schemas and content stored in a database, in a way similar to commercial HTML-SQL integrators.

Araneus offers a semantic description of the site content. A conceptual schema is used to represent the structure of data (Principle I). Internal derivation is supported within NCM, while external derivation is available, if needed, by means of wrappers (Principle II). Data navigation is explicitly represented in ADM (Principle III). Pages are composed with the Editor and Minerva tools (Principle IV). Presentation may be defined with any commercial HTML construction tool, as it is based on ad hoc tags in an HTML file (Principle V). A relational DBMS is used as a repository of the metadata (Principle VI). Araneus offers both a pull and push approach to page generation (Principle VII): in the former, pages are dynamically generated, in the latter pages are materialized off-line, with mechanisms to propagate updates from the database to the materializations. The Ulixes and Polyphemus tools support declarative queries on the site (Principle VIII). Currently, the system does not offer support for personalized access (Principle IX): the access to certain pages may be protected by the standard protection mechanisms of the Web server, requiring a username and password to access them, but this is not integrated into the model. Finally, no support is currently offered for event-based reactive processing (Principle X), except for the recomputation of materialized pages following database and schema updates.

4.3 AutoWeb

AutoWeb [6] is a project developed at Politecnico di Milano with the goal of applying a model-driven development process to the construction and maintenance of data intensive Web sites. The AutoWeb approach emphasizes the use of conceptual modeling to specify the semantics of Web applications and of a structured development process backed by CASE tools for reducing the development effort through the automatic generation of application pages. AutoWeb consists of three ingredients:

- A **concepual model for Web sites**, called *HDM-lite*, which is an evolution of previous hypermedia and database conceptual models, specifically tailored to the Web.
- Several **automatic transformations**, which address the description of conceptual schemas into relational database structures, and the production of pages (in HTML and Java) from data and metadata stored in the database.
- A set of **design-time and run-time CASE tools** that completely automate the design, implementation, and maintenance of a Web application.

An AutoWeb application is constructed starting from an HDM-lite schema, drawn with the Visual HDM Diagram Editor; presentation specification is assisted by an ad hoc tool (called Visual HDM Style Sheet Editor), which permits the designer to define presentation styles applicable to conceptual objects in a WYSIWYG manner. The conceptual model is automatically translated by a tool called Visual HDM Relational Schema Generator into the schema of a relational database for storing application data, plus a mini-database (called metaschema database) containing a relational representation of the site's structure, navigation and presentation.

The site is populated either using an automatically constructed data entry application, produced by the AutoWeb Data Entry Generator, or manually defining a mapping between the automatically generated relational schema and the schema of the pre-existing database. As the last step, application pages are dynamically constructed from database content and metadata by the AutoWeb Page Generator, in such a way that all the prescriptions of the conceptual model are enforced. A tool called AutoWeb Page Grabber also permits the user to selectively materialize portions of the site based on semantic criteria.

The AutoWeb conceptual model, HDM-lite, consists of primitives for the orthogonal specification of structure, navigation and presentation, as required by Principles I, III, and V. However, HDM-lite presently neither supports a language for data derivation nor the flexible composition of the page (Principles II and VI). As a consequence, the page structure is directly inferred from the schema of application targets (called entity in the HDM-lite jargon), which induces a regular structure also in the HTML pages generated from the conceptual schema.

AutoWeb stores into a relational DBMS both metadata and application data; this greatly enhances the possibility of quickly adapting the output pages to the user's needs, even at run-time (Principle VI). To increase performance, dynamic page generation can be bypassed by a flexible caching mechanism (Principle VII).

AutoWeb presently does not support any form of query language, as required by Principle VIII; AutoWeb extensions support the addition of explicit user modeling facilities, which constitute the basis for conceptual-level business rules tracking the user and reacting to significant events by manipulating the page generation process (Principles IX and X). For example, such extensions permits the registration of users, the collection of explicit and implicit preferences, and the specification of business rules mapping users with given preferences to dedicated navigation commands and presentation styles. In the present version, conceptual-level business rules are manually transformed into relational triggers.

The AutoWeb project is currently under evolution in the context of the W3I3 (Web-based Intelligent Information Infrastructure) Consortium (a project of the Esprit IV Framework with a 3 million Euro budget, sponsored at 50% by the EU). The Consortium includes Politecnico di Milano as technology provider, TXT Ingegneria Informatica as solution integrator, and a large-scale Web development company Digia (Digital Information Architects) Inc., together with pilot applications OTTO Versand (the world's largest mail order company) and KPN Research (the research branch of the major telecom company of the Netherlands).

W3I3 modeling will distinguish the five perspectives of structure, derivation, navigation, page composition, and presentation; thus it will fully comply with the first five rules. Meta-data will be stored on a relational server, thus adhering to rule VI; content, possibly distributed across different data sources, will be integrated into a global view of the site, obtained by mapping the conceptual schema into a relational representation. A special feature of W3I3 is the integration of user modeling and business rules: users are explicitly modeled through demographic and psychographic variables and business rules are used to map users or user groups to personal views of the site, computed dynamically, thus obeying to rules IX and X. Presently, W3I3 is in the design phase; most of the AutoWeb technology will be reused.

4.4 Strudel

Strudel is a project originally developed at AT&T Research, now being extended at AT&T Research, INRIA in France, and the University of Washington. The major features of Strudel are [3]:

- **Uniform Graph Model**. Every application object is represented as a named node in a graph. Each node has a set of attributes. Edges connect nodes, labeled with an attribute name.
- **Data integration**. Strudel emphasizes the integration of data originating from different sources. Wrappers are defined enabling the mapping from the data source to the internal Graph Model. Sources may be data in a relational database, or even pages of an existing Web site.
- **Query and transformation language**. A language is provided to extend the graph. Pages are represented as nodes in the graph; pages and page links can be specified starting from queries on the graph. The graph defines only the structure of the pages; HTML code is generated by a separate page composer, responsible of managing data presentation.

Strudel conceptually separates content, structure, and presentation. The description of content is based on the Uniform Graph Model. Strudel focuses on a bottom-up design process, where data usually pre-exists to the Web application design.

The conceptual schema is based on the Uniform Graph Model for semistructured data (Principle I). Data derivation and navigation are supported by StruQL, the query and transformation language operating on the Graph Model (Principles II and III). StruQL can also be used as a tool for the specification of page structure (Principle IV). Presentation is managed by writing HTML templates that are then fed to the HTML generator. Strudel offers the generation of independent presentations on the same data, satisfying Principle V.

Strudel differs significantly from other proposals with respect to DBMS integration (Principle VI), as it stores metadata in an ad hoc repository for semistructured data. Data that resides in a DBMS may be part of a Strudel site via a wrapper translating StruQL queries in the corresponding DBMS queries, but the DBMS is external to the system. The current Strudel prototype materializes the pages; there is a plan to extend the system, giving the designer the choice between materializing and composing pages on the fly (Principle VII).

Strudel does not currently envision facilities for the definition of user queries (Principle VIII). The designer can access the schema of the site (which is kept in the internal repository and is directly accessible by StruQL), but no user-friendly interface is currently provided for the unexperienced user. Personalized access (Principle IX) could be supported. In [4] there is a reference to the interest demonstrated by industrial users on the capabilities of user personalization; Strudel has the potential for offering this functionality, but to our knowledge it does not currently integrate user tracking facilities. Event-based reactive processing is not addressed (Principle X).

5 Conclusions

We have presented ten design principles which should drive the development of data-intensive Web applications; current practice highlights very high development and maintenance costs for such Web applications, and we argue that this situation is in part motivated by the lack of suitable design tools and environments. However, this paper reports of a trend in research prototypes, partially reflected in commercial systems, to improve the compliance to our decalogue. We believe that this trend will characterize future environments for the design of large Web applications.

Acknowledgement

We thank Paolo Atzeni and Alon Levy for very useful comments on an early draft of this paper, and for providing feedbacks on our descriptions of Strudel and Araneus.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *Int. Journal on Digital Libraries*, 1(1):68–88, 1997.

[2] P. Atzeni, G. Mecca, and P. Merialdo. Design and maintenance of data intensive Web sites. In *Proc. EDBT98*, Valencia, Spain, March, 1998.

[3] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Strudel: A web-site management system. In *Proc. ACM SIGMOD*, pages 549–552, Tucson, Arizona, May 1997.

[4] M. Fernandez, D. Florescu, J. King, A. Levy, and D. Suciu. Catching the boat with Strudel: Experiences with a web-site management system. In *Proc. ACM SIGMOD*, pages 414–425, Seattle, Washington, June 1998.

[5] P. Fraternali. Tools and approaches for data-intensive Web applications: A survey. To appear in *ACM Computing Surveys*.

[6] P. Fraternali and P. Paolini. A conceptual model and a tool environment for developing more scalable and dynamic Web applications. In *Proc. EDBT98*, Valencia, Spain, March, 1998.

[7] M. Gwyer. Oracle Designer/2000 WebServer Generator Technical Overview (version 1.3.2). Technical report, Oracle Corporation, Sept. 1996.

[8] G. Mecca, P. Atzeni, A. Masci, P. Merialdo, and G. Sindoni. From databases to web-bases: The ARANEUS experience. Technical Report 34-1998, Università di Roma Tre, May 1998.
http://www.dia.uniroma3.it/Araneus/articles.html.