Preface

This book is about building data-intensive Web applications. By this term, we refer to Web sites for accessing and maintaining large amounts of structured data, typically stored as records in a database management system. Today, data-intensive Web applications are the predominant kind of application found on the Web; sites for online trading and e-commerce, institutional Web sites of private and public organizations, digital libraries, corporate portals, community sites are all examples of data-intensive Web applications.

The development of a data-intensive Web application is a multi-disciplinary activity, which requires a variety of skills, necessary to address very heterogeneous tasks, like the design of data structures for storing content, the conception of hypertext interfaces for information browsing and content management, the creation of effective presentation styles, the assembly of robust and high-performance architectures, and the integration with legacy applications and external services. The development and maintenance of data-intensive Web applications requires all the tools and techniques of software engineering, including a well-organized software development process, appropriate design concepts and notations, and guidelines on how to conduct the various activities.

By looking at the way in which data-intensive Web applications are built today and at the tools available to developers, one realizes soon that the software engineering principles and pragmatics are not exploited to their full potential. Designers often construct Web applications by applying the best practices and methods they have learned in developing other kinds of software systems, like enterprise information systems and object-oriented applications. Such practices work well for the "conventional" part of Web application development, for example, the design of the data structures and of the business logic at the back-end, but they do not address the specificity of a "Web" application, which is the delivery of content and services using an hypertextual front-end. This gap is particularly apparent in the design concepts and notations: when it comes to specifying the front-end of their Web application, development teams resort to rather rudimentary tools, like paper and pencil or HTML mock-ups. This situation, which we have frequently witnessed also in very large organizations well equipped with software engineering tools, demands for an adaptation of the software development process, capable of addressing the characterizing features of Web applications. The Web application lifecycle should be built around a solid nucleus of

Web-centric concepts and notations, and supported by specific guidelines on how to put such concepts to work.

The contribution of this book is the proposal of a mix of concepts, notations, and techniques for the construction of data-intensive Web applications, which can be used by Web development teams to support all the activities of the application lifecycle, from analysis to deployment and evolution.

The proposed mix blends traditional ingredients well known to developers, like conceptual data design with the Entity-Relationship model and Use Case specification with UML, with new concepts and methods for the design of hypertexts, which are central to Web development. However, the value of the proposed approach is not in the individual ingredients, but in the definition of a systematic framework in which the activities of Web applications development can be organized according to the fundamental principles of software engineering, and all tasks, including the more Web-centric ones, find the adequate support in appropriate concepts, notations, and techniques.

The distinguishing feature of this development framework is the emphasis on conceptual modeling. Conceptual modeling has proven successful in many software fields; in database design, where the Entity-Relationship model offers a high-level and intuitive notation for communicating data requirements between designers and non-technical people, and is the base for creating high quality database schemas; in object-oriented applications, where notations like the Unified Modeling Language have considerably raised the level at which developers document and reason about their applications. We advocate that these benefits should apply also to the design of data-intensive Web applications, which should be specified using a high-level, visual, and intuitive notation, easily communicable to non-technical users, and helpful to the application implementers.

Therefore, this book proposes a high-level modeling language for hypertext specification, called Web Modeling Language (WebML). In essence, WebML consists of simple visual concepts for expressing a hypertext as a set of pages made up of linked content units and operations, and for binding such content units and operations to the data they refer to.

WebML follows the style of well-known conceptual modeling languages like Entity-Relationship and UML: every concept has a graphical representation, and specifications are diagrams. Therefore, the reader should not worry about the need to learn yet another language. As for the Entity-Relationship constructs, also WebML diagrams could be represented using the UML syntax, possibly with some loss of conciseness, but not of expressive power.

However, we stress that concepts are more important than notations, and that the methods for applying concepts are even more important. Therefore, in

the book we guide the reader both in learning the needed modeling concepts, Entity-Relationship and WebML, and in applying such concepts to the specification and design of a Web application, through such activities as requirements specification, data design, and hypertext design. Moreover, despite the slant toward conceptual modeling, we also focus upon the many problems of implementing and deploying a data-intensive Web application. The first chapter and the last part of the book are entirely devoted to technological matters, and show to the interested reader how to transform the conceptual design of a Web application into software components running on the current Web and database technologies, including HTTP, HTML, XML, XSL, relational databases and SQL, server side scripting languages and tag libraries, application servers, and caching architectures.

Last but not least, the book ends with a mention about CASE tools supporting the proposed lifecycle, because the benefits of applying conceptual modeling and a structured development process multiply, if adequate tools are available. All the proposed notations fit perfectly in the commercial tool suites popular among developers, like Entity-Relationship and UML editors and code generators. In particular, WebML can be easily supported, either by representing WebML diagrams using UML, or by exploiting WebML-aware tools, an example of which is presented in the last chapter of the book.

Book Organization and Chapter Summaries

The book is structured in four parts. The first part introduces the technological context in which development takes place; the second part presents the modeling languages used in the book, Entity-Relationship and WebML; the third part defines the software development process; the fourth part focuses on the implementation of data-intensive Web applications on top of modern Web-enabled architectures.

All chapters have a regular structure, with a motivational introduction that states the problem treated in the chapter, a central part that defines the proposed solution, and a conclusion, which summarizes the results. In the chapters devoted to the development process, the design steps are applied to a running case, which is progressively followed from requirements analysis to implementation.

Part I, including Chapter 1, summarizes the technologies relevant to data-intensive Web application development.

Chapter 1 contains a broad overview of the fundamental technologies employed in the construction of data-intensive Web applications. The chapter briefly illustrates the basic protocol and languages of the Web (HTTP, HTML, and client-side scripting and components); it focuses on XML, the new paradigm for content

structuring and exchange, and on its collateral standards for document transformation (XSL and XQuery); then it discusses the second ingredient of data-intensive Web applications, relational databases, and the associated query language (SQL) and interoperability standards (ODBC and JDBC). Finally, it explains the architectures and languages for building dynamic Web pages, including Java servlets, server-side scripting languages such as ASP and JSP, tag libraries, and application server architectures. The chapter ends with the discussion of multi-device content publishing.

Part II, including Chapters 2–5, is dedicated to the presentation of the modeling languages used in the book.

Chapter 2 describes the primitives of the Entity-Relationship data modeling language. The fundamental elements of structure modeling are *entities,* defined as containers of data elements, and *relationships,* defined as semantic associations between entities. Entities have named properties, called attributes, with an associated type. Entities can be organized in generalization hierarchies, and relationships can be restricted by means of cardinality constraints. The chapter also shows how to specify attributes and relationships whose content can be determined from other data elements, by writing declarative expressions using the Object Constraint Language (OCL).

Chapter 3 describes the WebML hypertext modeling language, which is based on the notion of units, pages, and links. *Units* describe the elementary pieces of content to be displayed, *pages* indicate how units should be assembled together, and *links* describe the connections between units and/or pages. Multiple hypertexts, called *site views,* may be defined over the same content, to offer different viewpoints to different users. The modeling primitives are introduced gradually, using many examples inspired to frequently used hypertext configurations.

Chapter 4 describes the extension of the hypertext model for supporting content management functions, like the update of personal information, the filling of shopping carts, and so on. New constructs are introduced for representing operations, which are either predefined or generic. Predefined operations represent typical content management and utility functions normally found in Web sites, like the creation, deletion, and modification of objects, the user's login and logout, and the delivery of e-mail messages; generic operations represent black-box functions and enable the integration of WebML applications with external services.

Chapter 5 concentrates on clarifying the meaning of hypertexts with an arbitrary structure of pages, units, and links. The chapter also presents a simple but complete high-level procedure for computing the content of hypertext pages,

which highlights the operational semantics of WebML and paves the way for the discussion on how to implement hypertext constructs, which is the subject of Part IV of the book.

Part III, including Chapters 6–9, presents the development process of data-intensive Web applications.

Chapter 6 is an overview of the application lifecycle. It discusses the specification, design, and implementation activities required to build a data-intensive Web application, by briefly describing the goals and tasks of each development phase.

Chapter 7 focuses on requirement analysis, an activity dedicated to the collection and specification of the application requirements, preliminary to the modeling and design phases. Requirements collection focuses on identifying users and groups, defining functional, data, and personalization requirements, as well as on nonfunctional requirements about presentation, usability, performance, availability, scalability, security, and maintainability. Functional requirements are formalized by means of UML use case diagrams; the core concepts and site views are expressed by means of a data dictionary and of site view maps; finally, visual style guidelines are expressed in the form of interface mock-ups.

Chapter 8 addresses the activity of data design and shows the particular flavor that this task assumes in the Web context. The data structure of Web applications often presents a regular organization, in which several interconnected sub-schemas can be recognized, each one centered on a "core entity" representing a fundamental business object. As a consequence, the design process assumes a regular shape too; it starts from the specification of the core concepts, which form the backbone of the data schema, and proceeds iteratively by adding four kinds of sub-schemas, which represent the internal components of core concepts, the interconnections for supporting navigation, the auxiliary objects for facilitating the access to the core content, and the concepts for supporting personalization.

Chapter 9 describes the hypertext design activities. Design proceeds in a top-down way: initially, a draft hypertext schema is obtained by partitioning each site view identified during requirements analysis into areas, and assigning a set of functions to each area, which support the browsing of core, access or interconnection objects, or content management operations. Then, the draft schema of each area is refined into a detailed schema, specified in WebML; in this phase, the designer establishes the actual units, links, operations, and pages of each site view. Hypertext design is facilitated by the usage of design patterns, which offer proved solutions to typical page configuration requirements.

Part IV, comprising Chapters 10–14, is dedicated to the implementation and deployment of data-intensive Web applications.

Chapter 10 concentrates on architecture design and is preliminary to the discussion of implementation. It reviews the reference architectures that can be used for building data-intensive Web applications and the criteria for choosing among the alternative options. The chapter specifically addresses the nonfunctional requirements of performance, security, availability, and scalability, and outlines the design decisions and trade-offs that must be faced to ensure the required level of service. The chapter ends with a section devoted to performance evaluation and caching, two important aspects of the design of Web architectures.

Chapter 11 deals with the mapping of conceptual data schemas onto the physical data sources. Various alternative scenarios are discussed, with a different degree of reuse of the existing schemas and content. The chapter starts by presenting the standard mapping rules for transforming a given Entity-Relationship schema into a relational database schema. Then it addresses the implementation of the relational schema in the context of the corporate data infrastructure, a task that presents several design choices and trade-offs, related to the problems of schema integration, data integration, and replication management.

Chapter 12 describes how to encode WebML pages into server-side programs. As a reference, the explanation adopts the Java Server Pages (JSP) scripting language and the JDBC database connection interface, but the discussion can be easily adapted to different platforms, such as the Microsoft's .NET architecture or the PHP scripting language. The explanation of the implementation techniques starts with simple page configurations, yielding relatively straightforward JSP page templates, and then progresses to cover a wide spectrum of features of dynamic hypertext pages.

Chapter 13 presents a more sophisticated implementation strategy, exploiting the Model View Controller (MVC) design pattern, which grants a well-balanced distribution of responsibility among the software components that collaborate to the page construction. In addition, the chapter illustrates other implementation techniques suited to large-scale applications, such as the definition of generic unit and operation services using XML descriptors, the development of distributed business objects with the Enterprise JavaBeans standard, and the centralized management of presentation with the help of CSS and XSL rules.

Finally, Chapter 14 describes an example of CASE tool, called WebRatio Site Development Studio, supporting the design of data-intensive Web applications and the automatic generation of code from Entity Relationship and WebML specifications. The chapter illustrates the architecture and functions of the tool, which covers the application lifecycle from data and hypertext design to their imple-

mentation. The annotated bibliography provides references to other tools supporting the specification and delivery of Web applications.

Several appendices complete the book; they summarize the elements of the WebML model, the syntax of WebML and of the Object Constraint Language, and the implementation techniques for transforming hypertext specifications into dynamic page templates and database queries.

Audience

This book has the ambitious objective of proposing a "paradigm shift" in the way Web applications are developed, rooted in the tradition of conceptual modeling and software engineering. It is directed not only to the IT specialists, but also to all the professionals involved in the construction of a Web application, an audience as broad as the spectrum of problems faced by Web application developers.

To address this target, we have made efforts to purge the book from any unnecessary formalism and academic discussion, and we have instead made intensive use of practical and motivating examples for explaining every new concept introduced to the reader. Therefore, the book should be approachable with limited effort by readers with a general background of database systems, software development, and Web technologies. Throughout the chapters, modeling concepts are shown at work, applied to the description of popular, real-life Web sites. In the same way, development tasks are exemplified with the help of a running case, taken from a real industrial project. In our intention, this book should emphasize "showing" things, with the help of progressive examples, rather than "telling" how things should be done.

The book could also be used in computer science courses dealing with data-driven design methods, especially now that computer science schools and universities are more and more orienting their curricula towards Web technologies and applications. Additional material for supporting professors in their lecturing and students in doing course work is available on the book's online Web site (see below).

Online Resources

The book is associated with several online resources. The Web site http://www.webml.org includes a variety of materials dedicated to model-driven Web development and to WebML, including examples of hypertext modeling, technical and research papers, teaching materials, and resources for developers (for instance, stencils for the popular Microsoft Visio diagram editor, which can be

used to draw WebML diagrams quickly). In particular, the section http://www.webml.org/book is dedicated to this book. It contains the full text of the JSP programs discussed in Chapters 12 and 13, and a number of exercises, some of which accompanied by solutions. An entry form in the Web site permits qualified instructors to contact the authors, to obtain further high quality and up-to-date teaching materials.

The Web site http://www.webratio.com describes WebRatio Site Development Studio, the CASE tool presented in Chapter 14; an evaluation program is available for trying the software, and academic licenses are granted upon request to teachers willing to use the tool in their classrooms.

Background

The model-driven approach to Web application development at the base of this book is the result of more than five years of research at Politecnico di Milano, the largest Italian IT School, accompanied by an intense development activity in the industry. The first research prototype of a model-driven CASE tool for Web applications, called AutoWeb, was designed by Piero Fraternali and Paolo Paolini between 1996 and 1998. The tool, operational since 1997, has been used to develop several Web applications, and has demonstrated the possibility of automating the construction of data-intensive Web sites specified with a high level conceptual language.

WebML was conceived in the context of the Esprit project "Web-Based Intelligent Information Infrastructures" (W3I3, 1998–2000), supported by the European Community, with the participation of five partners (Politecnico di Milano and TXT e-solutions from Italy, KPN Research from Holland, Digia Inc. from Finland, Otto Versand from Germany); the project delivered a prototype development environment, called ToriiSoft. Since 1999, WebML has been used for the development of industrial Web applications, both inside research contracts with companies such as Microsoft and Cisco Systems, and in industrial projects with companies like TXT e-solutions and Acer Europe. In the fall 2001, a team of WebML designers and developers founded a start-up company with the goal of further developing, distributing, and marketing WebRatio Site Development Studio, a tool suite based on WebML.

Acknowledgments

We acknowledge the work and dedication of a huge number of developers, researchers, and students, who have contributed to the design of WebML and to the

subsequent development of AutoWeb, Toriisoft, and WebRatio. We would like to thank, among others, Fabio Surini, Nicola Testa, Paolo Cucco, Roberto Acerbis, Stefano Butti, Claudio Greppi, Carlo Conserva, Fulvio Ciapessoni, Giovanni Toffetti, Marco Tagliasacchi, Andrea Rangone, Paolo Paolini, Stefano Paraboschi, Ioana Manolescu, Andrea Maurino, Marco Guida, Giorgio Tornielli, Alvise Braga Illa, Wim Timmerman, Pekka Sivonen, Stefan Liesem, Ingo Klapper, Daniel Schwabe, and Graham Robson.

Special thanks to Adam Bosworth, who was one of the first people to appreciate our effort to "change the way in which people think of the Web development." We owe to him many precious technical discussions, conducted on both sides of the Atlantic.

We thank Gianpiero Morbello, Massimo Manzari, and Emanuele Tosetti from Acer for permission to use the Acer-Euro application throughout Parts III and IV of the book.

Many thanks to the people of the CISCO IKF team, including Mike Kirkwood, Shirley Wong, Deepa Gopinat, Seema Yazdani, and Irene Sklyar. These people really know what a "large" Web application is!

We are also deeply indebted to Prahm Mehra and Paolo Atzeni, who assisted us with extremely careful comments and annotations, which greatly helped us in the revision of the manuscript.