Tools and Approaches for Developing Data-Intensive Web Applications: A Survey

PIERO FRATERNALI

Politecnico di Milano

The exponential growth and capillar diffusion of the Web are nurturing a novel generation of applications, characterized by a direct business-to-customer relationship. The development of such applications is a hybrid between traditional IS development and Hypermedia authoring, and challenges the existing tools and approaches for software production. This paper investigates the current situation of Web development tools, both in the commercial and research fields, by identifying and characterizing different categories of solutions, evaluating their adequacy to the requirements of Web application development, enlightening open problems, and exposing possible future trends.

Categories and Subject Descriptors: H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia ; D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms: Design, Experimentation, Languages, Reliability

Additional Key Words and Phrases: Application, development, HTML, Intranet, WWW

1. INTRODUCTION

Applications for the Internet in such domains as electronic commerce, digital libraries, and distance learning are characterized by an unprecedented mix of features that makes them radically different from previous applications of information technology [Myers et al. 1996]:

—Universal access by individuals with limited or no skills in the use of computer applications introduces the need of new man-machine interfaces capable of capturing the customer's attention and facilitating access to information.

—Global availability of heterogeneous information sources requires the integrated management of structured and unstructured content, possibly stored in different systems (databases, file systems, multimedia storage devices) and distributed over multiple sites.

In recent years the World Wide Web has been elected as an ideal platform for developing Internet applications, thanks to its powerful communication paradigm based on multimediality and browsing, and to its open architectural standards, which facilitate the integration of different types of content and systems.

Author's address: Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, I-20133, Italy.

CONTENTS

Modern Web applications are conveniently described as a hybrid between a hypermedia [Nielsen 1995] and an information system. As in hypermedia, i.e., applications commonly found in CD-ROMS, kiosks, information points, etc., information is accessed in an exploratory way rather than through "canned" interfaces, and the way in which it is navigated and presented is of great importance. Similar to information systems, the size and volatility of data and the distribution of applications requires consolidated architectural solutions based on technologies such as database management systems and client-server computing.

Due to this hybrid nature, the development of a Web application must cope with a number of applicative requirements, such as:

—the necessity of handling both structured data (e.g., database records) and nonstructured data (e.g., multimedia items);

—the support of exploratory access through navigational interfaces;

—a high level of graphical quality;

—the customization and possibly dynamic adaptation of content structure, navigation primitives, and presentation styles;

—the support of proactive behavior, i.e., for recommendation and filtering.

These requirements add up and typically compete with the technical and managerial issues of every software- and data-intensive application, which obviously remain valid for large Web applications also:

—security, scalability, and availability;

—interoperability with legacy systems and data;

—ease of evolution and maintenance.

As has happened in the past with other emerging technologies such as databases and object-oriented programming languages, methodologies and software tools may greatly help in mastering the complexity of innovative applications by fostering a correct understanding and use of a new development paradigm, providing productivity advantages, and thus reducing the risk inherent in application development and migration.

The goal of this survey is to address the software engineering, architectural, and applicative issues of Web development (Section 2); classify and compare current development tools in light of such aspects (Sections 4 to 10); formulate evaluations and perspectives by examining the relationship between state-of-the-practice solutions and relevant research areas (Sections 11, 12, 13 and

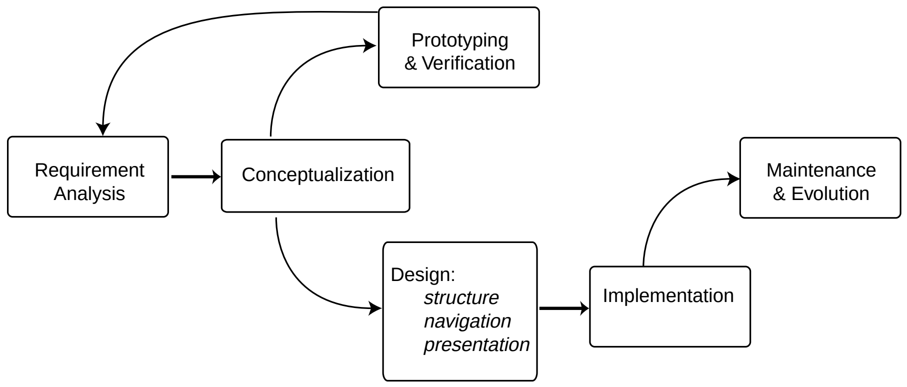


Figure 1. The lifecycle of a Web application.

14); and to summarize current choices for the developer on the basis of the characteristics of the application (Section 15).

2. PERSPECTIVES ON WEB DEVELOPMENT

The development of a Web application is a multifaceted activity, involving not only technical questions, but also organizational, managerial, and even social and artistic issues.

2.1 Process

Although there is still no consensus on a general model of the lifecycle of a Web application,[1] a scheme of typical activities involved in constructing a Web application can be obtained by interpolating the lifecycle models of traditional information systems and the proposals for structured hypermedia design [Garzotto et al. 1995; Isakowitz et al. 1995; Nanard and Nanard 1995; and Schwabe and Rossi 1995]. Figure 1 illustrates the lifecycle model used as a reference in this paper.

—*Requirements analysis*: The mission of the application is established by identifying prospective users and defining the nature of the information base. In addition to the customary requirement collection and feasibility assessment tasks, Web applications designed for universal access require special care in the identification of human-computer interaction requirements, in order to establish the interaction mode most suitable for each expected category of users, and for each type of output device that users are expected to use to connect to the application (ranging from hand-held personal communicators to high definition screens).

—*Conceptualization*: The application is represented through a set of abstract models that convey the main components of the envisioned solution. In the Web context, conceptualization has a different flavor with respect to the same activity in information system design, because the focus is on capturing objects and relationships as they will appear to users, rather than as they will be represented within the software system. Although the notation may be the same (e.g., the Entity Relationship Model [Chen 1976]), the schemas resulting from the conceptualization of a Web application and a database application usually differ.[2]

[1]Among the current proposals are those of Takahashi and Liang [1997]; Atzeni et al. [1998]; and Fraternali and Paolini [1998].

[2]A macroscopic difference is in the interpretation of relationships, in which database modeling represents semantic associations to be persistently

—*Prototyping and validation*: Simplified versions of the applications are deployed to users for early feedback. The importance of prototyping is particularly emphasized in the Web context, as it is in hypermedia, because the intrinsic complexity of the interfaces requires a timely evaluation of the joint effectiveness of structure, navigation, and presentation [Nielsen 1996]. Typically, a prototype is built prior to design and on a simplified architecture, e.g., as a set of manually implemented pages containing samples of the application content, which emulate the desired appearance and behavior.

—*Design*: Conceptual schemas are transformed into a lower-level representation, closer to the needs of implementation, but still independent of the actual content of the information base. Typically, the structural view of the application is mapped to the schema of a storage repository, the navigational view into a set of access primitives over the content repository, and the presentational view into a set of content-independent visual specifications (styles). The latter activity, called *visual design*, is of great importance in Web application development and is emerging as an autonomous discipline [Sano 1996; Horton et al. 1996].

—*Implementation*: The repository is filled with new content prepared by domain experts and/or with existing data stored in legacy systems; the actual interfaces, *pages* inWeb terminology, are constructed by embedding repository content and navigational commands into the appropriate presentation style. The mapping of design to implementation requires the choice of the network language in which the application is delivered (e.g., HTML, Java, ActiveX, or a mix thereof), and the decision on the time of "binding" between content and application pages, which can be offline or online.

—*Evolution and maintenance*: After delivery, changes in the requirements or bug fixes may require the revision of structure, navigation, presentation, or content. Changes are applied as high as possible in the development chain and propagated down to implementation.

The process model described caters to a variety of actual processes, whose applicability depends on the specific development context, including the available tool support, financial and time constraints, application complexity, and change frequency. Generally speaking, with applications of limited size and fairly stable requirements, requirement analysis is directly followed by implementation, possibly after a number of iterations through prototyping. Conceptualization and design assume more relevance as the complexity and volatility of the application increase.

2.2 Models, Languages, and Notation

A Web application is characterized by three major design dimensions:

—*Structure* describes the organization of the information managed by the application, in terms of the pieces of content that constitute its *information base* and of their semantic relationships.

—*Navigation* concerns the facilities for accessing information and for moving across the application content.

—*Presentation* affects the way in which application content and navigation commands are presented to the user.

To support the representation of application features during the development lifecycle, languages with different levels of formality and abstraction can be used.

At the conceptual level, applications

recorded, whereas in Web modeling a navigation possibility is generally implied.

are described by means of high-level primitives which specify the structural, navigational, and presentational views in a way that abstracts from any architectural issue.

Structural modeling primitives describe the types of objects that constitute the information base and their semantic relationships, without commitment to any specific mechanism for storing, retrieving, and maintaining the actual instances of such object types. Examples of notation that can be used to express structural features include the best-known conceptual data models, like the entity-relationship model [Chen 1976] and various object models [Rumbaugh et al. 1991].

Navigation modeling primitives express the access paths to objects of the information base and the available inter- and intraobject navigation facilities, again without committing to any specific technique for implementing access and navigation. The vast range of notation and techniques proposed for the more general problem of human-computer interaction specification can be employed for navigation modeling: data models extended with built-in navigation semantics [Garzotto et al. 1993; Isakowitz et al. 1995; Fraternali and Paolini 1998] or explicitly annotated with behavioral specifications [Kesseler 1995; Schwabe and Rossi 1995]; first-order logic [Garg 1988]; Petri nets [Stotts and Furuta 1989]; finite state machines [Zheng and Pong 1992]; and formal grammars [Jacob 1982] are among the viable options.

Presentation modeling aims at representing the visual elements of the application interfaces in a way that abstracts from the particular language and device used in the delivery. Many different techniques can be used, with a varying degree of formality and rigor, ranging from simple storyboard specification [Madsen and Aiken 1993] to the use of software tools [Myers 1995] and formal methods. The independent specification of presentation, separate from structure and navigation, is particularly relevant in the Web context because the final rendering of interface depends on the browser and display device, thus it may be necessary to map the same abstract presentation scheme to different designs and implementations.

At the design level, structured or semistructured data models are used to represent the features of an application in a way amenable to manipulation, query, and verification. Typically, design representations are maintained as relational or object-oriented schemas, to exploit the capabilities of database management systems, or as semistructured objects, to cope with information having partial or missing schemas [Florescu et al. 1998].

At the lowest degree of abstraction, applications are represented by means of implementation-level languages, notably network languages directly interpretable by the user's browsers. At this stage, content, navigation, and presentation are directly embedded in the physical marked-up texts or programs that make up the application.

2.3 Reuse

As in any other software process, reuse is an important aspect, related to the facility of building a novel application from existing artifacts [Biggerstaff and Perlis 1989]. Reuse may happen at all levels of the development process: conceptual schemas, design schemas, content, and physical application pages can be reused across applications.

The most common form of reuse on the Web, as in hypermedia, is content reuse, possibly facilitated by the presence of a structured repository, which enables the construction of different applications on top of the same information base. Reuse also occurs at the implementation level, where component-based libraries of self-contained elements (e.g., JavaBeans or ActiveX components) can be plugged into application pages to provide predefined functionalities (e.g., an electronic payment facility). Generation-based reuse focuses

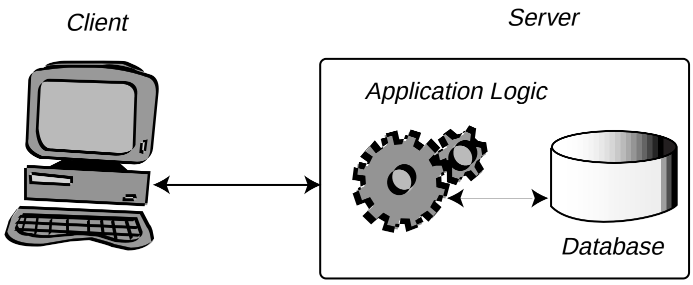


Figure 2. Two-tier architecture.

instead on reusing transformation techniques from design frameworks or partially instantiated implementations to full implementations (e.g., by generating application pages from page skeletons and database content).

2.4 Architecture

Architecture is the subject of design and implementation and reflects the spatial arrangement of application data and the spatio-temporal distribution of computation.

The minimal spatial configuration of a Web application is the so-called two-tier architecture, shown in Figure 2, which closely resembles the traditional client-server paradigm. This is different from client-server, wherein the two-tier solution clients (i.e., browsers) are *thin*, and are lightweight applications responsible only for rendering the presentation. Application logic and data reside on the server side.

A more advanced configuration, called three- or multitier architecture (Figure 3), separates the application logic from data, introducing an additional distinction of responsibilities at the back-end side. The presence of one or more distinct application tiers enables the implementation of advanced architectures that integrate the traditional HTTP protocol and client-server application distribution protocols for better performance, scalability, reliability, and security.

An orthogonal architectural issue is the time of binding between the content of the information base and the application pages delivered to the client, which can be *static* when pages are computed at application definition time and are immutable during application usage; or *dynamic*, when pages are created just-in-time from fresh content. Dynamicity has further nuances: it may involve only content (navigation and presentation are static), or scale to presentation and navigation.

2.5 Usability

From the customer's perspective, usability is the most important quality of a Web application. Although a thorough discussion of the emerging Web usability engineering field is outside the scope of this paper (we refer the reader to Nielsen [1996]), it is possible to identify a set of general criteria to use in assessing Web usability:

—Degree of visual quality indicates the overall coherence of the presentation and navigation metaphors and the individual quality of the graphic resources.

—Degree of customization measures the facility of tailoring the application interface to individual users or user groups. At one end of the spectrum, applications may have a fixed interface; at the other end, content, presentation, and navigation may be individually personalized.

—Degree of adaptivity is proportional to the runtime flexibility of the inter-

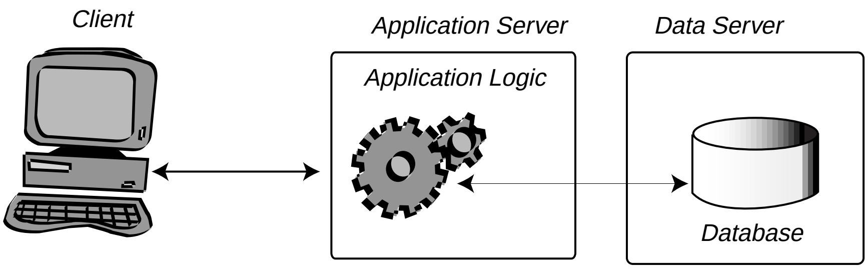


Figure 3. Three-tier architecture.

face; at one end, the application may be immutable; at the other, it may track the user's activity and change accordingly [Fraternali and Paolini 1998].

—Degree of proactivity indicates the capability of an application to interact with the user on its own initiative. Applications can be passive, i.e., provide information only in response to a user's requests, or proactive, i.e., able to send information to users upon the occurrence of relevant events.

3. TOOLS FOR WEB DEVELOPMENT

In response to the complexity factors of Web application development described above, many software vendors are providing instruments for supporting the construction of new applications or the migration of existing ones.

Notwithstanding their common goal, the available products greatly diverge in many aspects, which reflects different conceptions of the nature of a Web application and of its development process.

To help understand the state of the practice, we have grouped existing tools into six categories, which exhibit homogeneous features. This grouping is the result of a broad review, which has considered over forty products, listed in Appendix 1.

The individual categories are

(1) visual editors and site managers;

(2) Web-enabled hypermedia authoring tools;

(3) Web-DBPL integrators;

(4) Web form editors, report writers, and database publishing wizards;

(5) multiparadigm tools;

(6) model-driven application generators.

The order of presentation of the different categories reflects the increasing level of support that tools in each category offer to the structured development of Web applications.

Category 1 contains productivity tools that evolved directly from the HTML editor, which do not really support the development of large-scale Web-database applications, but are nonetheless interesting because they pioneered many concepts (like presentation styles and top-down site design) later integrated into more complex environments. The same limitation in the degree of support to structured development of large applications is shared by Category 2, which originates from a different application domain, offline hypermedia publishing, but recently added facilities for Web and database integration; the interest in these tools is motivated by their nonconventional (with respect to the standard software engineering practice) approach to application design and specific focus on navigation and presentation. Category 3 is the first one that explicitly addresses the integration of Web and databases to achieve a higher level of scalability, and includes very powerful, yet basic, products. These products can be used to implement applications on top of large databases, although at the price of a substantial programming effort. Category 4 takes quite a different, but still database-centric, approach to Web development by addressing the migration of client/server, form-based applications. These tools aim at augmenting the implementor's productivity in such tasks as form editing, report writing, and event-based programming; they also offer a higher level of support with respect to Category 3, but still concentrate only on the implementation phase. Category 5 contains a number of tools whose common feature is the integration of different development approaches and technologies, drawn from the previous four tool families. Finally, Category 6 includes those products (actually two products) that provide complete coverage of all the development activities, from conceptualization to implementation, by leveraging state-of-the-art software engineering techniques.

To complete the overview of Web application development, in Section 10 we include a mention to other classes of products which are not directly concerned with the specific focus of this paper on the design and maintenance of Web sites, but either cover fundamental architectural issues like distribution, reliability, performance, and security, or provide specialized facilities to users and site administrators, like advanced search functions and collaborative work support.

4. VISUAL EDITORS AND SITE MANAGERS

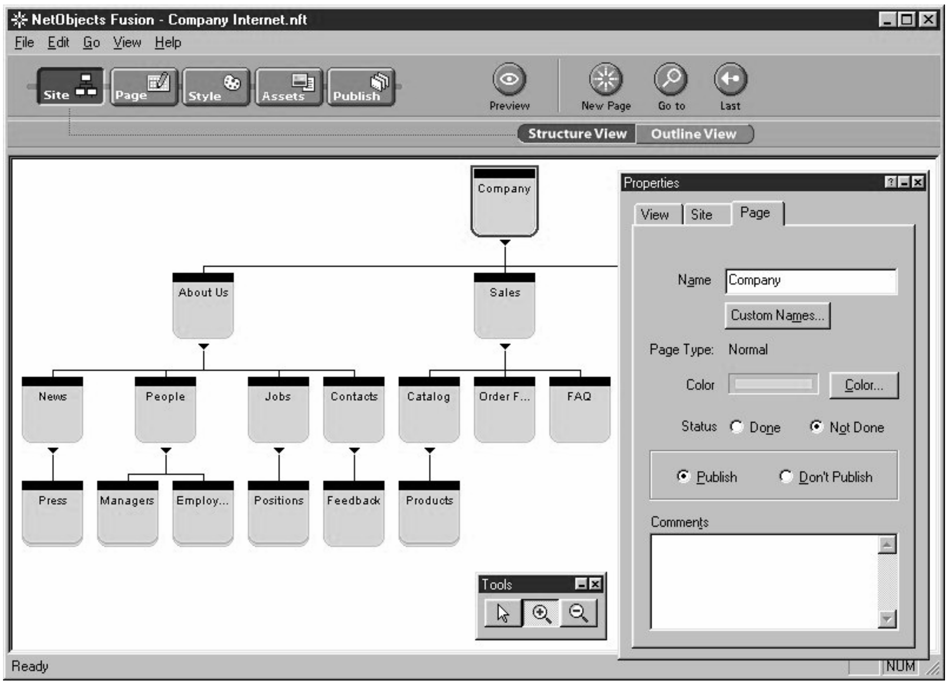
This class includes authoring and site management environments originally conceived to alleviate the complexity of writing HTML code and of maintaining the pages of a Web site in the file system. In a typical configuration these products bundle a WYSIWYG editor, which lets the user design sophisticated HTML pages without programming, and a visual site manager, which displays in a graphical way the content of a Web site and supports functions like page upload, deletion, and renaming, and broken link detection and repair. Among the many products in this category are Adobe SiteMill and PageMill, NetObject Inc.'s Fusion, SoftQuad's HotMetal, Claris Home Page, Macromedia's Backstage Designer, and Microsoft's FrontPage.

From a lifecycle perspective, these products address the implementation and maintenance of Web sites; implementation is supported by content production functions and maintenance by site-level management facilities. The most advanced products (e.g., NetObject's Fusion and Microsoft FrontPage) also offer a rather rudimentary approach to design, whereby it is possible to separate the definition of the hierarchical structure of a site from the authoring of content.

Automation concentrates on content production by generating code from visual page designs. Support to code generation is particularly relevant in those tools (like Macromedia's DreamWeaver, SofQuad's HotMetal Pro 5.0, Allaire's HomeSite, and many more) that support the latest extensions of HTML, like Cascading Style Sheets (CSS) [World Wide Web Consortium 1998a] for content-independent presentation specification, and the Document Object Model (DOM) [World Wide Web Consortium 1998b], for the object-oriented representation of page elements and their manipulation via a scripting language.

Some tools are able to generate part of the navigation logic by automatically inserting navigation buttons into pages based on their position in the layout of the site (Figure 4). Development abstractions are basically at the implementation level and oriented towards structure and navigation (pages and links); some tools also provide presentation abstractions in the form of page templates or "themes," i.e., groups of graphic resources that can be applied to multiple pages to obtain visual consistency (Figure 5).

Reuse happens mostly at the imple-



Name: Company
Page Type: Normal
Color
Status: Done [✓ Not Done]
[✓ Publish] Don't Publish

Figure 4. Hierarchical site layout in NetObject's fusion.

mentation level: content objects (e.g., multimedia, graphics, but also object-based components like JavaBeans and ActiveX controls) can be reused across applications.

The architecture originally supported by these tools is file-based and two-tier, and the binding between an application and its content is static. Recently, the most advanced tools have added functionalities for the dynamic connection to live database data, as discussed in Section 8.

Since development is conducted mostly by hand, the level of customization and the resulting visual quality of the final application is proportional to the effort spent and can be arbitrarily high; coherence of presentation and navigation metaphors is facilitated by visual templates and themes, although in this case customization becomes more expensive.

The features of these tools are shown in Table I: in summary, they are an excellent solution for small- to medium-sized applications, where publishing large information bases stored in a DBMS is not the major issue. The lack of a design-level schema of the application, independent of content, requires that the application features be defined instance-by-instance, and is a major obstacle to scale-up and integration of large masses of data. These limitations, however, are going to become less critical as these products become more integrated with Web-database tools.

5. WEB-ENABLED HYPERMEDIA AUTHORING TOOLS

Web-enabled hypermedia authoring tools share the same focus on authoring as visual HTML editors, but have a different origin because they were initially conceived for the development of offline hypermedia applications, and have been extended to support the generation of applications for the Web in recent times. The best known representatives of this class of products are

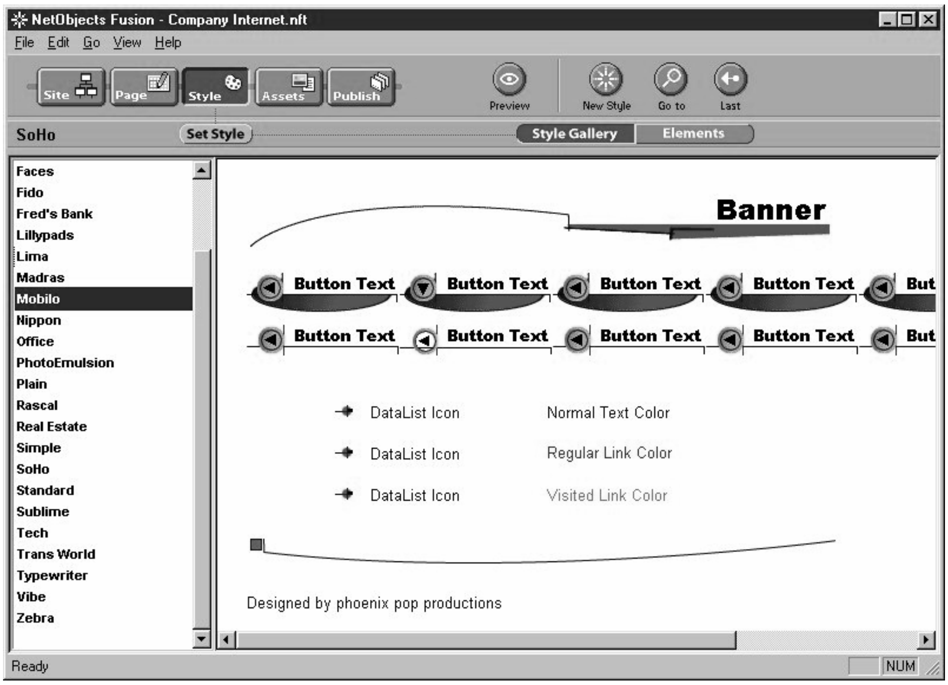


Figure 5. Graphic styles in NetObject's fusion.

Table I. Synopsis of Visual Editors and Site Managers

Process: Lifecycle Coverage	Design (limited to hierarchical layout definitions) Implementation Maintenance
Process: Automation	HTML generation from WYSIWYG page editing Generation of commands for hierarchy navigation
Abstractions	Implementation-level: pages, links, presentation styles
Reuse	Plug-in components; Reusable presentation styles
Architecture	Two-tiers, based on file system Static binding of content to pages
Usability	High graphical control through manual authoring High coherence through use of presentation styles Low customization, no adaptivity, no proactivity

Asymetrix's Toolbook II Assistant/Instructor, Macromedia's Director and Authorware, Formula Graphics' Multimedia 97, Aimtech Iconauthor, and Allen Communication Quest. Most of these products have both Web export facilities and database connectivity, but these extensions are not always compatible, and the developer must often choose between the two possibilities. These products are characterized by the following features:

—The authoring metaphor used in the definition of the application. Examples of such metaphors are: flowchart (Quest, Authorware, IconAuthor), book (Toolbook), and movie making (Director).

—The way navigation is defined, which may require programming in a scripting language, or may be supported by visual programming and wizard tools coherent with the product's authoring

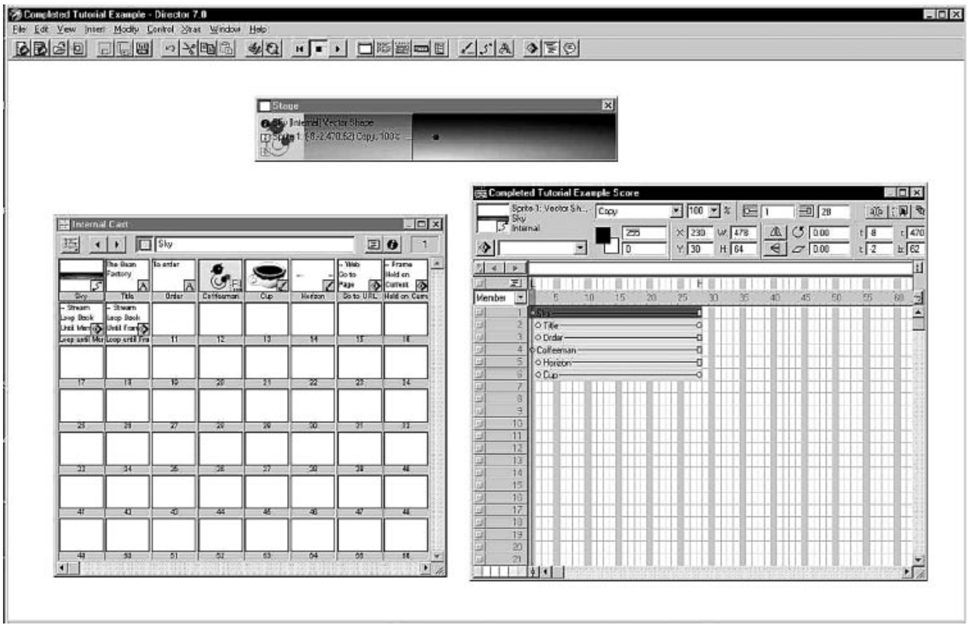


Figure 6. The film-making authoring metaphor of Macromedia's Director.

metaphor (e.g., in Director, objects' synchronization is defined by editing the score for the cast members of a stage; see Figure 6).

—The type of database connectivity, which may range from support of an internal database, of an external database via gateway software (typically ODBC or JDBC), or of an external database through DBMS API.

—The type of Web connectivity, which may be achieved by means of a plug-in application extending a Web browser, or by exporting the hypermedia application into a network language. Web connectivity may affect database connectivity, because in the present version of most tools, Web export can be done only for applications not connected to a database.

—The export language may be HTML, Java, or a mix of both.

Basically, hypermedia tools are authoring assistants; their focus is restricted to the ad hoc implementation of long-lived applications published once and for all in CDROMS or information kiosks. As a consequence, they still have little provision for a structured development lifecycle, and offer limited support to application modeling, design, testing, and maintenance.

To aid the implementation of large applications, some tools separate authoring in-the-large (i.e., the definition of the overall application structure) from authoring in-the-small (i.e., the detailed definition of the multimedia content of individual information nodes) [GMP95]. This distinction fosters an elementary form of design which helps in reconfiguring the application structure and navigation, independent of content. For example, tools like Quest, Authorware, and IconAuthor distinguish between a "design mode," where information nodes are placed on a flowchart, and an "edit mode," where each individual node can be edited. A parallel can be established between authoring in-the-large in hypermedia authoring tools and the site layout view offered by some visual HTML editors: in both cases, design is done at the instance level, without the help of a content-independent application schema, and navigation is (in part) automatically generated from the abstract structure of the application.

Reuse is pursued both at the design level by means of reusable substructures and application-independent layouts, and at the implementation level by leveraging libraries of scripts or multimedia components. Recent versions of most tools also offer interoperability with third-party components (for example, inclusion of Java applets and ActiveX components).

Regarding the development abstractions, hypermedia tools have a strong plus in the built-in facilities for explicitly modeling navigation paths orthogonal to application structure; for example, guided tours can be defined as sequential navigations based on associative properties of objects that may cut across the hierarchical structure of the application. Other navigational aids include application-wide search functions, glossaries, and access indexes.

Like visual HTML editors, hypermedia authoring tools

—do not provide a schema for distinguishing the type of objects constituting the application (beside the low-level distinction between objects of different media types). The main development abstractions are generic container objects (pages, icons, stages) whose definition intermixes structural, presentation, and behavioral aspects (and navigational features in those tools that do not support the seperate specification of navigation);

—offer presentation abstractions in the form of page styles, i.e., sets of graphic resources applied uniformly to the entire application or to part of it.

The architecture of Hypermedia tools shows a substantial immaturity, with respect to competitor solutions; all tools were originally conceived for offline, file-based publication, and later upgraded to external database connectivity and Web export. Most products offer database connectivity via external libraries using ODBC, but lose this feature after Web export. Moreover, translation from native proprietary format into a network language is still rudimentary, and the most advanced features (like those obtained by exploiting the tool's scripting language) do not carry over to the Web version of an application.

Conversely, usability is the major strength of this category of products, which focus on delivering very sophisticated user interfaces exhibiting a degree of control over graphic accuracy and multimedia synchronization hardly available through other means. The inherent navigational design paradigm, coupled to very effective and well-established aids like guided tours and user-defined access indexes, which most tools offer for free or with little programming effort, contribute to the deployment of applications that are very effective and much closer to the kind of communication found in high quality, hand-developed Web sites.

Table II summarizes the Web-related features of hypermedia authoring tools.

6. WEB-DBPL INTEGRATORS

The common denominator of products in this category is the dynamic production of Web pages from information stored in a database by integrating databases and Web technology at the language level.

Different network languages (notably HTML and Java) are merged with a database programming language (DBPL) to obtain an intermediate application programming language capable of making the best of both Web and database technology. Web-DBPL integration is pursued in three distinct ways:

—*HTML extensions* add new tags to HTML to embed both generalized programming constructs and database-specific capabilities. The developer

Table II. Synopsis of Web-Enabled Hypermedia Authoring Tools

Process: Lifecycle Coverage	Design (support of authoring-in-the-large) Implementation
Process: Automation	HTML/Java generation from WYSIWYG authoring Synchronization of multimedia objects Generation of commands for navigation
Abstractions	Implementation-level authoring metaphors: pages, stages, flowcharts
Reuse	Script libraries, application skeletons, plug-in components, reusable presentation styles
Architecture	Monolithic Two-tiers through Web export Three-tiered through libraries for database connectivity Static binding of content to pages
Usability	High graphical control through manual authoring High coherence through use of presentation styles Customization through user-defined access indexes No adaptivity, no proactivity

writes page templates, which intermix declarative HTML tags and executable code. Products in this category can be further distinguished based on the language paradigm used to extend HTML (imperative, object-based, object-oriented). Page templates must be translated into pure HTML; this task is normally executed at runtime by an interpreter. The performance of the different products vary, based on the process in which the interpreter is executed and on the way the connection to the database is managed. Examples of HTML extensions include Allaire Inc.'s Cold Fusion Web Database Construction Kit; Microsoft's Active Server Pages (ASP) and Internet Database Connector (IDC); Vignette Corporation's StoryServer; HAHT Software's HahtSite; and Informix AppPages. In general, HTML extensions have a broader objective than the mere addition of database connectivity, and include primitives for overcoming HTML's well-known limitations, like the absence of control flow instructions, modularization mechanisms, and stateful interaction.[3]

—*DBPL extensions* take the other way around and add specialized functions to an existing DBPL or to a general-purpose programming language with a database interface, to enable the output of HTML code as a result of program execution. Developers write database applications that construct HTML output from data retrieved from the database. This approach requires database programming skills, but may yield better performance when DBPL programs are executed and optimized directly by the DBMS. Examples of DBPL extensions include Oracle's PL/SQL Web Toolkit, Sybase's PowerBuilder Web.PB class library, and Borland's Web interface for the Delphi Client Server Suite.

—*Java extensions* add database connectivity to Java. The best-known effort in this direction is the JDBC open standard, which offers an ODBC-like architecture for Java applications that want to interact with a DBMS. Unlike HTML extensions, connectivity is added in the form of a library masking, under a uniform callable interface, the details of interacting with different proprietary DBMSs and SQL dialects.

Web-DBPL integrators are tools for the programmer, who may use them to simplify the task of gluing together Web

[3]Stateful interaction is the capability of retaining the application status between two consecutive client's requests; this is not supported by HTPP/1.0.

pages and database queries by hand, they are different from visual HTML editors and hypermedia authoring tools, which offer small-scale application development functions to nonprogrammers. 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, and presentation of an application, which is directly implemented by manually writing the necessary HMTL page templates or DBPL programs.

Reuse may be achieved either through the native constructs of the programming language, i.e., packages and modules, or through ad-hoc modularization mechanisms such as inclusion between HTML page templates.

The target architecture is database-centered and three-tiered; the most sophisticated products also permit multiple database tiers and distributed queries.

Not having any specific support for presentation and navigation modeling, these products have no direct impact on the usability and graphical quality of the user interface, which must be designed separately.

The major difference between Web-DBPL integrators and visual HTML editors and hypermedia authoring tools is that the former introduce a clear separation between the final application pages and the content repository, the schema of which acts as an implicit specification of the application structure.

This distinction has several consequences on the application-development process:

—it permits a more effective manipulation of the information base because changes to content can be made without affecting navigation and presentation;

—it exposes the different object types that constitute the application, although these types are not elicited from the structural, navigational, and presentation requirements of the application, but stem from the design of the underlying database;

—thanks to explicit object typing, the uniformity and coherence of the presentation are facilitated, because all objects of the same type may be easily visualized in the same way. As a counterpart, customization at the instance level must be explicitly programmed by identifying exceptional cases and treating them in an ad hoc manner.

Being database-driven, applications built with Web-DBPL integrators may leverage the reactive capabilities of the underlying DBMS to achieve adaptivity and proactivity, e.g., database triggers. For example, an Oracle trigger could react to user-generated events and produce HTML pages adapted to the specific situation.

In spite of the above positive side-effects, Web-DBPL integrators cannot be considered development tools in themselves, but are comparable to traditional client/server 4GL because they provide a high-level programming interface masking lower-level architectural details; as such they are often used to build more sophisticated products, like the ones reviewed next.

Table III summarizes the most important features of Web-DBPL integrators.

7. WEB FORM EDITORS, REPORT WRITERS, AND DATABASE PUBLISHING WIZARDS

This category collects a large number of products, all having the common characteristic of leveraging traditional database design concepts and development tools, to rapidly deploy both new and legacy data-intensive applications on the Web. A first classification can be obtained by considering the functions offered by the tools, ordered by complexity and sophistication:

Table III. Synopsis of Web-DBPL Integrators

Process: Lifecycle Coverage	Structural design (design of underlying database) Implementation Content maintenance
Process: Automation	Web-database communication Query shipment and result processing Page formatting from query results
Abstractions	Design-level: database structures (tables, classes) Implementation-level: pages, interface objects
Reuse	Reusable modules: page templates, procedures, classes
Architecture	Three/multitier Dynamic binding of content to pages
Usability	No specific support for control of graphical quality Coherence facilitated by page templates (HTML extensions) Low customization, adaptivity and proactivity manually implementable through database triggers

—Database export: support is limited to the publication of basic database elements, i.e., tables and views, and is achieved by automatically mapping database content into a network language, notably HTML. The export can be either static or dynamic, the latter case requiring on-the-fly HTML generation, which is normally implemented by means of a Web-DBPL integrator. A certain level of personalization can be obtained by applying sets of preferences to drive the output of Web pages. A typical example is the export of tables and queries in Microsoft's Access97.

—Report writing: more complex read-only applications can be obtained by defining custom reports, which can be exported to the Web either statically or dynamically. Dynamic publication requires a report interpreter working side by side with the Web server. Examples include Crystal Report Print Engine, Oracle's Reports for the Web, and Microsoft's Access97 report export facility.

—Form-based application development: assistance extends to the construction of interactive, form-based applications for accessing and updating data, and more generally for implementing business functions.

Form-based application development is where the highest number of products concentrate, and developers are faced by an impressive spectrum of alternatives. Among the reviewed products, we cite Microsoft's Visual InterDev, Visual Basic 5, and Access97, Oracle Developer 2000, Inprise IntraBuilder, Sybase's PowerBuilder, Apple's WebObjects, NetDynamic's Visual Studio, Asymetrix SuperCede Database Edition, and Allaire's Cold Fusion Application Wizards. The most relevant dimensions characterizing the different products are as follows:

—The application delivery format can be pure HTML, a combination of HTML and Java, HTML plus proprietary component technology and scripting languages (e.g., client-side ActiveX objects, VBScript or JavaScript tags), or a totally proprietary format (e.g., ActiveX documents, Sybase's PowerBuilder PBD files). The delivery format affects the client's performance and the application's portability, with minimal overhead and maximum portability in the pure HTML case, and maximum overhead and minimal portability when the application is deployed in a totally proprietary format interpreted by a browser's plug-in application.

—The application development language(s), which may be different from the delivery language(s), to preserve existing software investments and programming skills and/or increase performance and portability. Overall, the spectrum of bindings between development and delivery languages is extremely varied, ranging from products offering a single development language and multiple output formats (e.g., Oracle's Developer 2000 maps PL/SQL to HTML, Java, and Visual Basic) to products translating different input languages into the same output format (e.g., Microsoft's Visual InterDev maps design-time ActiveX controls scripted in VBScript or JavaScript into pure HTML).

—The implementation paradigm, which can be 3GL programming, Rapid Application Development (RAD), wizard-based, or a mix thereof. Almost all tools offer RAD functionalities, letting programmers define an application by dragging and dropping components onto a form, and by visually customizing their properties. Behavior customization, as necessary for the definition of complex forms, can be added by writing code in the tool's development language. A few products (e.g., Visual InterDev, IntraBuilder, NetDynamics, Cold Fusion) also include a simplified development path for non-programmers, based on "wizards" enabling the automatic generation of code from sets of preferences selected through a visual interface (Figure 7). Wizards are applied to obtain both components and complete applications; in the latter case they deliver "canned" database applications, in which the data structure and the interface logic are predefined (e.g., master-detail data entry and visualization, drill-down reports, and so on).

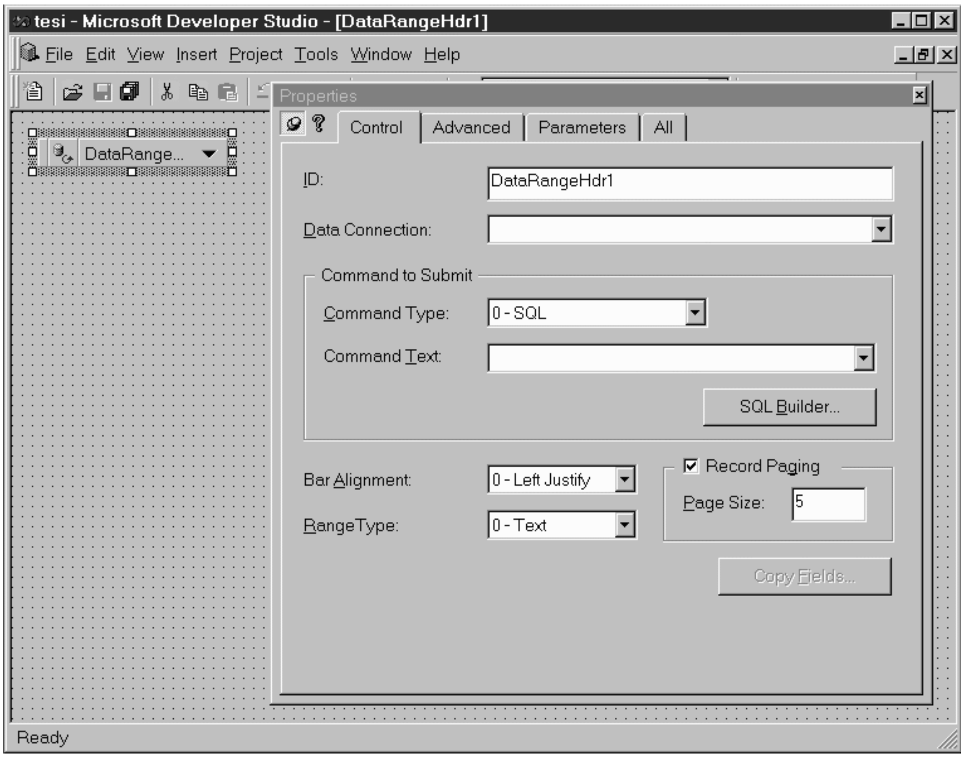
—The execution architecture: the way in which an application can be partitioned among the different tiers of the Web architecture is even more varied than the choice of development and delivery languages, because object distribution (as supported by languages like Java and architectures like Corba and DCOM) permits the same functionality, e.g., a database driver, to be located either in the client or in the application server.

From a software engineering point of view, products in this class are to Web development what Integrated Development Environments (IDEs) and Rapid Application Development tools were to traditional application programming: instruments for boosting individual and team productivity in the implementation phase. The best tools also help process management, testing, and maintenance by leveraging standard software engineering techniques like source and version control, configuration management, and visual debugging.

With one exception, reviewed later, form-based application generators do not provide upper-CASE capabilities: 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 the tool of choice.

Component-based reuse is also intrinsic to most products, which leverage the various object-level interoperability standards emerging on the Web. Components can be either client-side, in which case they encapsulate typical interface controls (e.g., menus, edit fields, grids, etc.) or server-side, in which case they typically wrap database access and data formatting primitives.

The development abstractions are interface-oriented, and drawn from the traditional client/server, database-centric world: query and data entry forms, input/output controls, record lists, and reports. The structure of the application is dictated by the forms that compose it, and is generally isomorphic to (a subset of) the database schema, especially when forms are automatically derived from database tables and views through database wizards. More complex patterns can be obtained by exploiting canonical structures like table lookups and master-details, or by hand-programming non-



ID: DataRangeHdr1
Data Connection:
Command Type: 0 - SQL
Command Text:
Bar Alignment: 0 - Left Justify
RangeType: 0 - Text
☑ Record Paging
Page Size: 5

Figure 7. Datarange wizard in Microsoft's Visual Interdev.

standard combinations of objects. The navigation semantics is deeply embedded within the application, and hyperlinks are emulated by means of form-to-form connections. In some cases, navigation is automatically inferred by the tool from database structure, as in automatically generated master-details and drill-down applications, where database foreign key constraints are translated into HTML hyperlinks. Presentation is addressed as an independent issue only in those products that provide a notion of "presentation style" (e.g., models in Access97, themes in InterDev, master pages in IntraBuilder) which can be uniformly applied to all application pages to obtain a consistent look.

On the architectural side, all products target multitier architectures; the most recent generation (e.g., NetDynamics Visual Studio) is tightly integrated with specialized middleware products, called *application servers*, which we review in Section 10.

Unlike Web-DBPL integrators, which are neutral products that can be used to shape any kind of application, form-based generators and database publishing wizards deliver complete software solutions, including user interface logic. Thus their underlying interaction model influences the user perception of application quality. The evaluation must consider the application context in which a tool is used, and consequently the kind of users addressed. In an intrabusiness, or business-to-business contexts, these products are essentially vehicles for Web-enabling traditional client-server applications. In this setting, interface quality is proportional to the similarity between the look and feel of original client-server applications and their Web counterparts, because the peculiarities of Web interactions (free browsing,

Table IV. Synopsis of Web Form Editors, Report Writers, and Database Publishing Wizards

Process: Lifecycle Coverage	Implementation Testing Maintenance
Process: Automation	Source code generation from RAD and wizard tools
Abstractions	Implementation-level: forms, reports, controls
Reuse	Plug-in components: client-side and server-side
Architecture	Multitier, integration with application servers Dynamic binding of content to pages
Usability	Canned interfaces based on query/result display loop Low customization, no adaptivity, no proactivity

associative links, multimediality and user-level customization) are less important than adherence to interface standards well-established among users. Conversely, if these products are targeted to user-oriented applications for the Internet, they must be compared to Web and hypermedia authoring tools, in which case their form-to-form navigation pattern and the limited graphic control imposed by wizards are a source of rigidity.

Table IV overviews the main features of Web form editors, report writers, and database publishing wizards.

8. MULTIPARADIGM TOOLS

The ongoing trend in the evolution of Web development tools is an increasing convergence between visual editing and site management products, Web-HTML integrators, and client-server products (reviewed in Sections 4, 6, and 7, respectively).

Multiparadigm tools integrate solutions from the aforementioned categories into a unified development framework. The most typical configuration is one in which visual HTML editing and site administration are extended with external components, which provide database connectivity, or with enhanced HTML generation capabilities able to produce scripts for pulling content from databases, or with full-fledged database publication wizards.

Examples of such convergent tools are FrontPage98, which includes database objects for connecting to external data sources; Elemental's Drumbeat, which combines sophisticated visual editing and assets management with database wizards comparable to those of specialized Web-database connectivity products; Lotus Domino and Domino Designer, which offer an integrated environment for Web-enabling Notes client-server applications and for developing Web sites that take advantage of Domino Server capabilities; and NetObject's Fusion (version 3.0), which features the Fusion2Fusion extension for connecting NetObject's visual page editor and Allaire's HTML-SQL integrator.

Multiparadigm tools do not introduce novel approaches into Web development, but combine already established features to broaden their spectrum with respect to single-paradigm products. Thus they offer advantages (summarized in Table V) that are the sum of the strengths of their components (implementation productivity, top-down design, three-tier architecture, high graphic control), but do not overcome the major limitations of the products reviewed so far, i.e., lack of a high-level, content-independent representation of the site in terms of structure, navigation, and presentation.

9. MODEL-DRIVEN WEB GENERATORS

Model-driven Web generators are at the top of the proposed categories, because they provide the highest level of automation and lifecycle coverage, by applying conceptual modeling and code generation techniques to the development of Web applications.

Table V. Synopsis of Multiparadigm Tools

Process: Lifecycle Coverage	Design (limited to hierarchical layout definitions) Implementation and maintenance
Process: Automation	HTML generation from WYSIWYG page editing Generation of commands for hierarchy navigation Web-database communication Source code generation from RAD and wizard tools
Abstractions	Implementation-level: pages, links, presentation styles, interface objects, tables, queries
Reuse	Plug-in components, reusable presentation styles
Architecture	Three and multitier Static and dynamic binding of content to pages
Usability	High graphical control through manual authoring Coherence through use of templates and presentation styles Low customization, adaptivity and proactivity manually implementable through database triggers

This category comprises a few commercial tools, which exhibit different conceptual models and code generation techniques. We review Hyperwave Server 4.0, and Oracle's Web Development Suite.

Hyperwave [Hyperwave Information Management 1998] is an advanced document management environment which permits remote users to browse, annotate, and maintain documents distributed over the Web. Hyperwave has a very basic, yet powerful, high-level model of a Web application, which is thought of as a set of *document collections* organized hierarchically. Collections may contain subcollections and documents, and have different behaviors based upon their type. Documents in a collection can be linked and annotated with a number of metadata. Links are managed by the Hyperwave server, so that any hypertextual structure can be superimposed over a set of otherwise independent documents.

From the description of collections, links, and metadata, Hyperwave generates a Web interface that enables both user-oriented and administrative functions like collection and link browsing, searching and notification, remote document management, fine-grain version and access control, collaborative work, and personal annotations. The generated interface has a default presentation, which can be personalized using a proprietary template language.

Although document-oriented, the Hyperware server relies on database technology and on a multitier architecture to store metadata and manage links spanning multiple servers.

A more database-centric approach is taken by the Oracle Web Development Suite, which comprises Designer 2000 [Gwyer 1996; Barnes and Gwyer 1996], a CASE tool for generating Web applications from augmented entity-relationship diagrams.

Designer 2000 is an environment for business process and application modeling, integrated with software generators originally designed to target traditional client-server environments, namely Oracle Developer 2000 [Hoven 1997], and Visual Basic. 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. More specifically, three inputs drive the generation process:

—A Web-enhanced database design: database design diagrams specify the structure of the database in terms of tables, views, foreign key relationships, and integrity constraints.

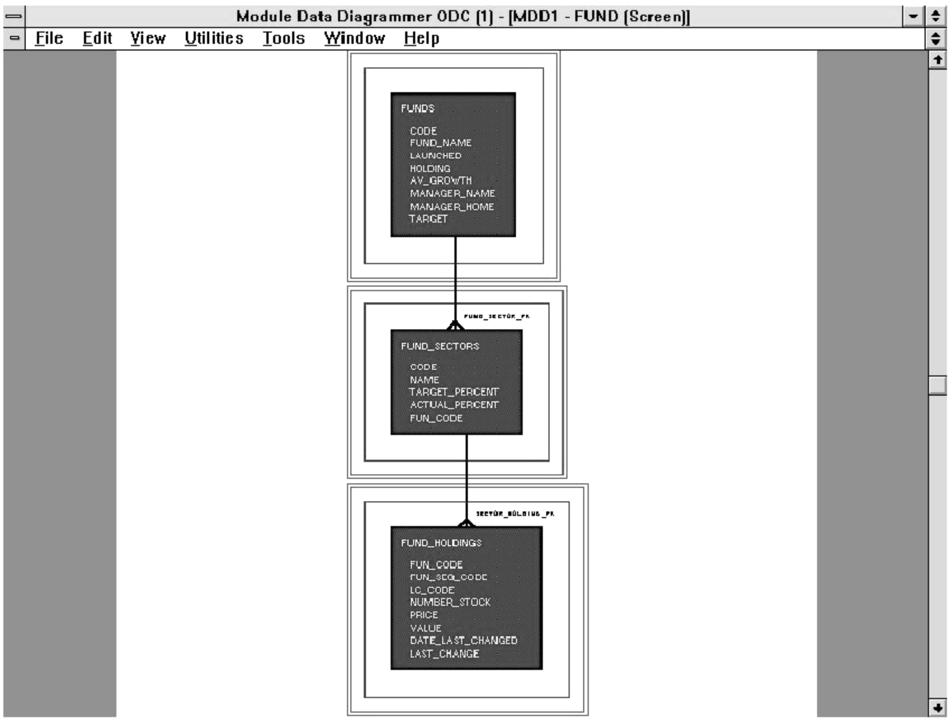


Figure 8. Module definition in Oracle Designer 2000.

These constitute the schema of the future Web application. A few visual features can be specified in the schema: for example, column definitions can be supplemented with caption text and display format (e.g., a pop-up list). Moreover, some integrity constraints (e.g., valid ranges) can be attached to columns and tables, and the Web generator can be instructed to produce code for checking them on the server via PL/SQL or on the client via JavaScript.

—The definition of applications and modules: modules correspond to basic application units; each module consists of a sequence of tables, linked by foreign key relationships (Figure 8). The order of tables in a module determines the sequence of HTML pages that will be produced for that module. Navigation is established by drawing links between modules: the designer may define which modules can be reached by a given module and introduce fictitious modules acting as hierarchical indexes over other modules.

—User preferences are parameters that can be set to govern the presentation of the generated application; they can be defined either globally, at the module, or at the component level. Examples of preferences are colors, headers and footers, background images, and help text.

From these inputs, the WEB generator produces fixed-format Web pages; one set of related pages is generated for each module, and links between different modules are turned into hyperlinks between the HTML startup pages of modules.

Model-driven generators apply the fundamental principles of software engineering to Web development; apart from all other product categories, applications are first modeled at the conceptual

Table VI. Synopsis of Model-Driven Generators (Oracle Designer 2000 and Hyperwave)

	Designer 2000	**Hyperwave**
Process: Lifecycle Coverage	Conceptualization (E/R) Design (relational model) Implementation Reverse engineering	Conceptualization (collection and link definition) Implementation
Process: Automation	Relational schema generation from ER Generation of HTML from design models and presentation preferences	Relational metaschema generation Navigation generation Presentation generation
Abstractions	Conceptual-level: entity, relationships, attributes Design-level: modules, tables, columns, constraints Implementation-level: pages, links	Conceptual-level: collections, views, abstract links, documents, virtual collections Implementation-level: pages, HTML links
Reuse	Module reuse; Preferences reuse	Multiple views over the same document base
Architecture	Multitier, dynamic binding	Multitier, dynamic binding
Usability	Low graphical control of generated pages High coherence through use of presentation preferences Low customization, no adaptivity, no proactivity	Low graphical control of generated pages High coherence through use of presentation preferences Programmable customization, no adaptivity, proactivity through notification

level and then implemented through code generation. This approach and its benefits are comparable to the ones delivered by CASE tools for object-oriented application development, and range from reduced development effort, reverse engineering, and multilevel reuse.

However, Hyperwave and Designer 2000 diverge in the underlying conceptual model, and both exhibit limits in the description of Web applications.

Hyperwave adopts a simplified hypermedia model, which is well suited to represent navigation, but lacks a proper structural model; as a consequence, the information base is reduced to a set of flat documents annotated with metadata.

Conversely, Designer 2000 draws the development abstractions from the database world, and adapts such concepts as entities and relationships to Web modeling by adding to them some navigation and presentation flavor. This attempt to twist a database conceptual model and apply it to a Web application is responsible for some limitations in the usability of the resulting applications, where the absence of proper modeling abstractions for navigation and presentation limits the exploitation of the communication capabilities of the Web.

The limitations of commercial Web generators are addressed by a few research prototypes, namely Araneus [Atzeni et al. 1997]; Autoweb [Fraternali and Paolini 1998]; Strudel [Fernandez et al. 1998]; WebArchitect [Takahashi and Liang 1997]; W3I3 [Ceri et al. 1998]; HSDL [Kesseler 1995]; RMC [Diaz et al. 1995]; and OOHDM [Schwabe and Rossi 1995], which are discussed in Section 12.

10. MIDDLEWARE, SEARCH ENGINES, AND GROUPWARE

Besides support for conceptualization, design, and implementation, Web development requires tackling other issues which, although outside the focus of this paper, are critical for the delivery of effective applications in performance, availability, scalability, security, information retrieval, and support for collaborative administration and usage.

These needs are served by ad hoc tools or by specialized functions integrated into Web design products. In the sequel, we briefly review the most important features of three categories of products: application servers, search engines, and groupware and collaborative design tools.

10.1 Middleware

Industrial strength data-intensive applications require not only proper design tools but also a solid architecture, good performance, availability, scalability, and security. These goals have prompted the extension of the original two-tier HTPP architecture to more sophisticated configurations, encompassing three, and even multiple, tiers. The key to multitiered applications is the capability to separate data, interfaces, and application logic, and to distribute each aspect to distinct network nodes. Such distribution leverages Internet-enabled application protocols (like Corba Internet InterOrb Protocol (Corba IIOP [Object Management Group 1996]) and Microsoft's Distributed Common Object Model, DCOM [Microsoft Corp. 1996]) and the native remote procedure call capabilities of network languages (notably, Java's Remote Method Invocation, RMI [Sun Microsystems 1995]).

The in-depth presentation of the alternative architectural and technological options for multitiered applications is outside the scope of this paper, and is thoroughly addresses by many authors (see, for example, the special issue of IEEE Internet Computing on Internet Architectures [Benda 1998] for a recent review of the state of the art, and Byte's special report on networked components [Montgomery et al. 1997] for a comparison of Corba and Microsoft architectures).

However, in the tool market, several vendors are also addressing multitiered architectures by offering specific products, called *application servers*, which are middleware facilities either integrated into HTTP servers, or working side by side to them. Examples of application servers are NetDynamics, Lotus Domino Server, Oracle 8i Java Server, Sybase's Jaguar CTS, and Inprise's VisiBroker for Java.

Application servers do not directly impact the client-side design process, but offer several extensions to the standard functions of HTTP engines, which can be used to support server-side development:

—Efficient execution of server-side programs that implement the business logic of the Web application. Products differ in the development languages supported, which commonly include C++, Java, and scripting languages like Perl, Visual Basic, and JavaScript. Efficiency is improved by replacing the Common Gateway Interface (CGI) between the Web server and the application programs by means of optimized protocols and architectures like FastCGI (http://www.fastcgi.com) and Java Servlets [Chang 1998].

—High-performance client-server communication. Such capability may be based on a comprehensive architecture for application distribution, on programming language features, or on proprietary remote procedure calls. Rather than using HTTP, enabled clients may bypass the Web server and directly connect to the application server using either open protocols like Corba/IIOP and Java RMI, or proprietary protocols, like Microsoft's DCOM, Domino's Notes Remote Procedure Calls (NRPC), and Sybase's Jaguar CTS remote procedure calls.

—Optimized and extensible connection to multiple external data sources. This function includes pooling connections to a database across multiple clients, caching result sets, and possibly extending the application server with plug-in gateways to heterogeneous data sources like Enterprise Resource Planning (ERP) and legacy systems (examples of built-in gateway components and programmable toolkits are NetDynamics 4.0 Platform Adapter Components (PACs) and Domino Enterprise Connection Services (DECS)).

—Flexible and dynamic load-balancing of client requests by means of automatic replication of server functions. High-volume incoming traffic may be dynamically routed to multiple instances of server-side functions, implemented either as separate threads of the application server or as external processes, possibly residing on remote hosts. Dynamic redirection also enables transparent failure-handling: a user's request for an unavailable service can be routed to a replica of the same function, either dynamically spawned or statically instantiated at system configuration time.

—Implementation of a secure infrastructure for both data and messages. Security may be granted by a variety of means: user logging and authentication over both HTPP and non-HTTP connections (e.g., by supporting the X.509 open standard certificate format, or authentication via third-party directories), message and content encryption (e.g., by supporting the IETF Secure MIME standard and RSA/RC2 cryptography), and password quality testing.

—Transactionality, i.e., the capability of performing atomic and recoverable transactions over a single or multiple tiers. This feature, traditionally offered by distributed database servers and TP monitors, requires the application server to implement write-ahead logging and two-phase commit protocols.

10.2 Search Engines

The proper design of structure and navigation normally results in Web sites with a self-evident organization, consequently reducing the need for full-text searches over the information base [Halasz 1988]. However, Web applications offered to the general public must also consider the needs of casual readers and readers with highly specific interests, for which a content search is the most effective interaction paradigm.

Designing the search functions for a data-intensive Web site is an orthogonal issue with respect to the design of structure, navigation, and presentation, and is supported either by ad hoc functions integrated into Web development tools, or by specialized products called *search engines*.

Examples of Web development tools that bundle integrated search functions are Hyperwave (which also comes with a separate commercial search engine), and Lotus Domino Designer. Among the numerous stand-alone search engines, we mention Verity Search97, Harvest, OpenText's LiveLink Search and Spider, Altavista Search, QuarterDeck's WebCompass, and Excite for Web Servers.

Search engines basically consist of two main components: a user interface and query processor, whereby users can pose queries and obtain a ranked list of pages whose content satisfies the query; and an indexing component (also called spider or crawler) which creates and maintains indexes over the data sources.

The available commercial products differ in a variety of dimensions: the kind of queries they support (keyword-based, Boolean, natural language, fuzzy); the customizability of the display of results; the flexibility of the index creation and maintenance process (e.g., the possibility of scheduling the updates of indexes differently for different data sources or to analyze different file formats); and the adherence to the so-called *robot-exclusion standard*, by which webmasters can deny access to crawlers at their sites.

For a broad review of commercial search engines and a comparison of their features, the reader may refer to Lange [1997].

10.3 Groupware

Collaboration requirements affect Web development in several ways, going from the concurrent construction of a Web site by geographically distributed development teams, to the provision of limited content editing and interaction functions directly to end-users, to real-time interaction integrated into a Web site.

In the commercial market several tools offer groupware capabilities, including

—concurrent access control via content locking and checkin/checkout procedures;

—distributed file systems for concurrent content upload;

—offline collaboration tools like calendars, schedules, notification lists, and discussion groups;

—real-time collaboration facilities like virtual rooms, white boards, chats, and video conferencing.

—full-fledged workflow support, including offline and online collaboration facilities and workflow modeling and implementation.

Examples of design tools supporting collaborative development are Hyperwave, TeamSite, and Lotus Domino Designer. Products for the definition of virtual workteams are reviewed in Wong [1998], and a comprehensive survey of Web-enabled workflow systems is contained in Miller et al. [1997].

11. EVALUATION

Table VII summarizes the features of the categories of Web development tools described in Sections 4 to 9, in light of the perspectives of Web development discussed in Section 2. To complete this picture, developers must also consider middleware products, which address the enhancement of performance and security and offer a platform for implementing advanced business logic, search engines, and groupware tools, which add specialized functions orthogonal to the design process.

The state of the practice summarized in Table VII can be better understood if compared with the current situations of mature software technologies, like object-oriented systems and databases (see Figure 9).

In these contexts, development tools cover the entire spectrum of the application lifecycle, and approach development from the right angle: they exploit well-established abstractions suited to the specific context (as the conceptual and logical models for database design and the object-oriented notation for OO systems) and follow well-proven development guidelines, like those described in Ceri et al. [1993] for database design, and in Rumbaugh et al. [1991] for object-oriented design. Pictorially, mature tools and approaches are represented as "light cones" that enlighten the various phases of the development cycle without crossing the borders of their target application field.

The situation of Web development is different and typical of a not yet mature technology (it could be easily compared to the OO tool market in the eighties): most products limit their focus to implementation, with some provision for design (as represented by the shorter "light cone" in Figure 9); a few tools are trying to cover the lifecycle in a broader way, but do so by approaching development from an unnatural angle, typically using models, abstractions, and processes drawn from other contexts. The "slanted" approach of these tools and approaches are represented by the light cone covering all phases of development,

Table VII. Synopsis of the Different Categories of Web Development Tools

	Visual Editors	Hypermedia Tools	Web-DBPL Integrators	Form Editors	Multi-paradigm Tools	Model-driven Generators
Lifecycle coverage	Implement., hierarchical site design, link maintenance	Implement., design (authoring in-the-large)	Implement.	Implement., maintenance (debugging)	Implement., hierarchical site design, link maintenance, debugging	Conceptual. design, implement., maintenance, reverse eng.
Automation	Generation of HTML	Generation of HTML/ Java	Database connection, query shipment, result formatting	Generation of HTML/ Java	Generation of HTML, database connection	Generation of design schemas, navigation commands, interfaces
Abstractions	Page, link, presentation style	Authoring metaphors	Table, page elements	Form, report, client-side and server-side control	Page, link, presentation style, form, table	Entity, relationship, module, table, column, collection, link
Reuse	Components, presentation styles	Libraries, skeletons, components, styles	Page templates, DBPL units	Client-side and server-side components	Components, presentation styles, templates	Modules, preferences, collections, links
Default architecture	2-tiers, static	2-tiers, static	3-tiers, dynamic	3-tiers, dynamic	3-tiers, dynamic	3-tiers, dynamic
Support to usability	Good graphic control and coherence (manual)	Very good graphic, navigation, synchronization control (manual)	Interface neutral, proactivity through triggers	Canned interfaces	Good graphic control and coherence (manual and with templates)	Predefined interfaces, low graphic control

but with an inclination due to its origin from the database area.[4]

12. RESEARCH PERSPECTIVES

Web application development has recently gained much attention not only in the commercial field but also in the research community, where several projects address the extension of the capabilities of site design tools in various directions, and propose innovative design processes and methodologies.

In this Section we briefly review a number of projects that have proposed solutions for overcoming some of the limitations currently experienced by state-of-the-practice commercial tools: Araneus [Atzeni et al. 1997; Atzeni et al. 1998 ; Atzeni et al. 1998]; Autoweb [Fraternali and Paolini 1998]; Strudel [Fernandez et al. 1998]; Web Architect [Takahashi and Liang 1997]; and W3I3 [Ceri et al. 1998].

The common denominator in such efforts is the goal of supporting the activities of data-intensive Web design from conceptualization to maintenance by properly distinguishing between the different dimensions of Web design, organizing the development activities into a structured process, and providing tools for partially automating repetitive development tasks.

However, besides such commonalities,

[4]A similar phenomenon took place in the early days of object-orientation, when many proposals were put forth to adapt structured analysis/structured design concepts to object-oriented implementation.

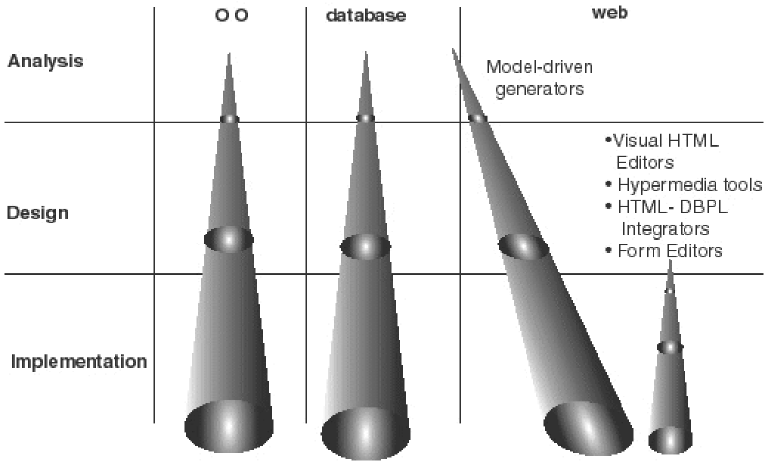


Figure 9. A pictorial view of the state-of-the-practice of Web development tools.

each project has its own special focus, and addresses only some of the issues left open among the currently available commercial tools.

In Table VIII we summarize the features of the various projects using the same categories as commercial tools (see Table VII), and in Table IX we underline the special focus and strong points of each project.

For completeness, we conclude with background research on Web development, which has taken advantage of contributions from several fields, in which hypermedia and databases are prominent.

13. RESEARCH PROJECTS IN DATA-INTENSIVE WEB DEVELOPMENT

13.1 Araneus

Araneus[5] [Atzeni et al. 1997 ; Atzeni et al. 1998 ; Atzeni et al. 1998] is a project at Univ. di Roma Tre, which focuses on the definition and prototype implementation of an environment for managing unstructured and structured Web content in an integrated way (called a Web Base Management System, WBMS). The WBMS should allow designers to effectively deploy large Web sites, integrate structured and semistructured data, reorganize legacy Web sites, and Web-enable existing database applications.

On the modeling side, Araneus stresses the distinction among data structure, navigation, and presentation. In structure modeling, a further distinction is made between database and hypertext structure: the former is specified using the entity-relationship model, the latter using a notation that integrates structure and navigation specification called 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 navigation and page composition. ADM offers the notion of *page scheme*, a language-independent page description notation based on such elements as attributes, lists, link anchors, and forms. The use of ADM introduces *page composition* as an independent modeling task: the specification of data and page structure is orthogonal, and therefore different page schemes can be built for the same data.

Presentation is specified orthogonally to data definition and page composition, using an HTML template approach.

[5]The project Web site is http://www.dia.uniroma3.it/Araneus.

Table VIII. Synopsis of Reviewed Research Projects in Data-Intensive Web Development

	Araneus	Autoweb	Strudel	Web Architect	W3I3
Lifecycle Coverage					
Conceptualization	Y	Y	N	Y	Y
Logical design	Y	Y	Y	N	Y
Prototyping	N	Y	N	N	Y
Implementation	Y	Y	Y	Y	Y
Maintenance	Y	Y	Y	Y	Y
Restructuring	Y	N	Y	N	N
Rev. engineering	Y	Y	Y	N	Y
Process Automation					
Generation of supporting database	N	Y	N	N	Y
Generation of mapping to legacy databases	Y	N	N	N	Y
Generation of navigation structures	Y	Y	N	Y	Y
Generation of HTML pages from templates	Y	N	Y	N	N
Generation of HTML pages from abstract specs	N	Y	N	N	Y
Modelling Abstractions					
Structure model	Y	Y	Y	Y	Y
Derivation model	N	N	Y	N	Y
Navigation model	Y	Y	N	Y	Y
Page composition model	Y	N	N	N	Y
Presentation model	Y	Y	Y	N	Y
Reuse & Components					
Plug-in components	N	N	N	N	N
Reusable presentation styles	N	Y	N	N	Y
Application skeletons	N	Y	N	N	Y
Default Architecture					
Two tiers static	N	N	Y	Y	N
Three tiers static	Y	Y	N	N	Y
Three tiers dynamic	Y	Y	N	N	Y
Automatic caching	N	Y	N	N	Y
Support to Usability					
Navigation uniformity	N	Y	N	Y	Y
Presentation uniformity	Y	Y	Y	N	Y
Usability guidelines	N	Y	N	N	Y

The development process follows two tracks: database and hypertext. Database design and implementation are conducted in the usual way using the entity-relationship model and mapping it into relational structures. After that, the entity-relationship schema is transformed into a NCM schema; this shift requires several design activities. The next step, hypertext logical design, maps the NCM schema into several page schemes written in ADM. Finally, implementation requires writing page schemes as templates in the Penelope language [Atzeni et al. 1997], which specifies how physical pages are constructed from logical page schemes and content stored in a database, in a way similar to commercial template-based HTML-SQL integrators.

Araneus includes several tools to support automation of the aforementioned design tasks. A specific point in the Araneus project is the integration into the WBMS of languages and tools for querying and restructuring HTML data, so that the designer can deploy a new site, Web-enable a database application, and "reverse-engineer" a legacy HTML site, all within one system.

From an architectural viewpoint, Araneus offers both static and dynamic page generation.

The system does not currently support

Table IX. Advanced Features of Reviewed Projects in Data-Intensive Web Development

Araneus	Querying and restructuring of HTML-based sites
	Full design methodology
	Orthogonal structure, navigation, composition and presentation modeling
	Tool support
Autoweb	Full design methodology
	Orthogonal structure, navigation, and presentation modeling
	Relational representation of both data and metadata
	Full CASE support
	Tool-supported usability guidelines
Strudel	Querying and restructuring of semistructured data
	Declarative site definition
WebArchitect	Full design methodology
	Scenario-based modeling
	Role-based specification of content
	Tool support
W3I3	Full design methodology
	Orthogonal structure, derivation, navigation, composition and presentation modeling
	User modeling and profiling
	Adaptive behavior through Web business rules

user profiling and personalization. Finally, no support is offered for event-based reactive processing, except for recomputation of materialized pages following database and schema updates.

13.2 Autoweb

Autoweb[6]Fraternali and Paolini [1998] 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.

Autoweb consists of three ingredients:

—A Web modeling notation called HDM-lite, which evolved from previous hypermedia and database conceptual models, specifically tailored to the Web.

—Two transformation techniques that address the mapping of conceptual schemas into relational database structures and the production of application pages (in HTML and Java) from data and metadata stored in the database.

—A set of design-time and runtime CASE tools that completely automate the design, implementation, and maintenance of a Web application.

The Autoweb conceptual model, called HDM-lite, includes primitives for the independent specification of structure, navigation, and presentation. Apart from other projects, presentation is specified at an abstract level totally independent of the implementation language, and is automatically mapped to HTML (a protoype implementation of the mapping to Java has also been developed). This makes presentation styles reusable across applications, as well as across different object types within the same site.

HDM-lite presently supports neither the orthogonal composition of the page, whose content is inferred from the structure schema, nor a language for data derivation. An original feature of Autoweb is the storage into a relational DBMS not only of application data, but also of metadata about navigation and presentation schemas, which greatly enhances the possibility of quickly adapting the output pages to the user's needs, even at runtime.

The main focus of Autoweb is the

[6]The project Web site is http://www.ing.unico.it/autoweb.

automation of the development process. An Autoweb application is constructed starting from an HDM-lite schema, with a tool called Visual HDM Diagram Editor; presentation specification is assisted by the Visual Style Sheet Editor tool, which permits the designer to define presentation styles applicable to conceptual objects in a WYSIWYG manner. The conceptual model is automatically translated into the schema of a relational database for storing application data, and into a metaschema database containing a relational representation of the site's structure, navigation, and presentation. The site is populated by either using an automatically constructed data entry application, produced by the Autoweb Data Entry Generator, or by manually defining a mapping between the automatically generated relational schema and the schema of the pre-existing database. As a 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.

During the development process, reuse happens at two levels: partially instantiated application skeletons and abstract presentation specifications.

Autoweb has a three-tier architecture, featuring both static and dynamic page generation. Dynamic generation is optimized by a flexible caching system that caches and refreshes pages of selected types based on user's requests.

A unique feature of Autoweb is the provision for interface design guidelines within the design tools: uniformity of navigation and presentation is enforced by the design and page generation tools at the level of both individual object types and of the entire application.

13.3 Strudel

Strudel[7] is a project of AT&T Labs [Fernandez et al. 1998], which aims at experimenting with novel ways of developing Web sites based upon declarative specification of the site's structure and content.

Strudel's core idea is to describe both the schema and content of a site by means of a set of queries over a data model for semistructured information.

Content is represented using the *Uniform Graph Model*, a graph-based data model capable of describing objects with partial or missing schema. As a starting point of the construction of a site, external data sources, e.g., HTML files or relational databases, are translated by means of wrappers into the Strudel internal format. In this way it is possible to either restructure an existing HTML site or Web-enable a legacy data repository.

Then design of a Web site requires writing one or more queries over the internal representation of data, using the Strudel query language (StruQL). Such queries permit the designer to select the data to be included in the site, along with links and collection of objects for navigation. In this way Strudel separates description of content from definition of the structure and the navigation of the site. Presentation is added as a separate dimension by means of HTML templates; these mix HTML presentation tags and special-purpose tags, which are bound at HTML generation time to objects resulting from site definition queries. The templates determine rendering of the site definition queries in HTML.

Specification of navigation is intertwined with structure and presentation because navigable links and index collections are specified together with the queries that define the site, and also because the structure of HTML pages and their links depend on templates that describe the presentation.

Presently, Strudel has a two-tier static architecture, in which queries are evaluated and transformed into HTML pages in advance. However, it could be possible to evalute queries and render their results dynamically.

[7]The project Web site is http://www.research.att.com/sw/tools/strudel.

The declarative definition of structure and content by means of queries opens the way to personalization: different sites or different versions of the same site can be built on top of the same content simply by changing the StruQL site definition queries.

13.4 Web Architect

WebArchitect [Takahashi and Liang 1997] is a project aimed at developing methods and tools for the construction of Web-based information systems (WBIS). The authors propose a structured design process that goes through analysis, design, construction, and maintenance of a Web site.

Analysis includes both static and dynamic modeling. The former is conducted with the entity relationship model, the latter requires the identification of scenarios in the tradition of object-oriented modeling [Jacobson 1994]. During ER modeling, entities are classified according to the different roles they play in the definition of the site (agent, product, or event). Design is conducted in parallel with scenario analysis and aims at pinning down the structure and navigation schema of the Web site. Design results are represented using a variant of the Relation Management Data Model by Isakowitz Diaz et al. [1995], which incorporates the roles of the entities forming the Web site.WBIS implementation and maintenance are supported by WebArchitect and PilotBoat. The former tool supports definition of the structure and navigation of the site, as well as maintenance of metadata on the site's resources; the latter is a client application permitting the user to browse an application based on *metalinks*, implementing the navigation semantics specified in WebArchitect. Metalinks are navigable connections stored outside the application objects, which are managed by extended HTTP engines supporting the special-purpose methods LINK and UNLINK.

13.5 W3I3

W3I3 (WWW Intelligent Information Infrastructure)[8] Ceri et al. [1998] is a project of the W3I3 Consortium, a partnership of four European industries and one academic institution, funded by the European Community. W3I3's goal is to advance Web modeling and development tools for data-intensive Web applications, with a special focus on user profiling, personalization, and reactive behavior.

W3I3 modeling points out the five perspectives of structure, derivation, navigation, page composition, and presentation, and includes models and languages for specifying a site under these perspectives (see http://webml.org).

The development process stresses the automatic generation of application pages from the conceptual model stored in a relational database, and content stored in external structured or semistructured data repositories.

The architecture is multitier: application content can be distributed across different data repositories 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 psycographic variables, and business rules are used to map users or user groups to personal views of the site computed dynamically.

Presently, W3I3 is in the implementation phase; a prototype version of both the design tools and the runtime environment supporting site personalization and the Web business rules has been constructed.

14. BACKGROUND RESEARCH

Web design tools and approaches owe much to the debate on hypermedia modeling and design, semistructured data modeling, and hypermedia development

[8]The project Web site is http://www.txt.iy/w3i3.

tools proposed prior to the advent of the Web. In the following sections, we review some of the most important contributions.

14.1 Modeling Notation

Historically, most of the modeling notation adopted by current Web development methodologies stems from the evolution and hybridization of previous conceptual models for database and hypermedia design.

The common ancestors of many subsequent proposals are the entity relationship model [Chen 1976], in the database field, and the Dexter model [Halasz and Schwarz 1994] in the hypermedia area.

The **Dexter model** originates from the effort to provide a uniform terminology for representing the different hypertext structuring primitives offered by hypertext construction systems; the core of the model is the ordered representation of a hypertextual application at three levels: *storage*, *within-component*, and *runtime* levels. The storage level describes the network of nodes and links of the hypertext, without details on the inner structure and node content, which is the focus of the within-component layer. The runtime level deals with the dynamics and presentation of the hypertext.

The modeling concepts available at the storage level are very basic: *components* describe pieces of information that constitute the hypertext, and can be either atomic or composite. *Links* are a special kind of component used to represent navigable paths. Although not conceived for hypertext design, the Dexter model advocates many concepts, such as the distinctions among the structure, navigation, and presentation of a hypertext, whose influence has been long-lasting.

Several subsequent contributions arose from criticism of the Dexter model, and added more complex forms of hypertext organization and more powerful navigation primitives [Garzotto et al. 1993]; time and multimedia synchronization features [Hardman et al. 1994]; and formal semantics of navigation and a structured design process [Isakowitz et al. 1995]. Among these evolutions, HDM [Garzotto et al. 1993] and RMM [Isakowitz et al. 1995] have been particularly influential in the design of hypermedia applications.

HDM and its variants [Schwabe et al. 1992; Garzotto et al. 1993; Garzotto et al. 1991; Garzotto et al. 1993] shifted the focus from hypertext data models as a means to capture the structuring primitives of hypertext systems, to hypertext models as a means for capturing the semantics of a hypermedia application domain. HDM integrates features of the entity relationship model and the Dexter model, to obtain a notation for expressing the main abstractions of a hypermedia application, their internal structure and navigation, and application-wide navigation requirements. Web structure is expressed by means of entities, substructured into a tree of components. Navigation can be internal to entities (along part-of links), cross-entity (along generalized links), or noncontextual (using access indexes, called collections [Garzotto et al. 1994]).

RMM (Relationship Management Methodology) [Isakowitz et al. 1995] evolves HDM by embedding its hypermedia design concepts into a structured methodology, splitting the development process into seven distinct steps and giving guidelines for the tasks. RMM's data model (called RMDM) structures domain entities into *slices*, and organizes navigation within and across entities using associative relationships and structural links.

Most of the recent proposals for Web modeling are built on top of the entity relationship model and hypermedia design models (notably HDM and RMM), adapted to the specificity of the Web context: Araneus and WebArchitect draw from RMM and the entity-relationship model, Autoweb and W3I3 have proposed a Web-specific evolution of concepts first proposed by HDM.

Finally, another source of inspiration

to Web modeling comes from research on the representation and querying of semistructured data, i.e., data with partial or missing schema. The proposed data models, thoroughly reviewed in Florescu et al. [1998], express Web content by means of relations, labeled graphs, hypertrees, and logic. Araneus and Strudel are examples of Web content management systems based on semistructured data models and query languages.

14.2 Processes

Web development processes evolved in parallel with Web design notation, and have the same hybrid origin from the information system and hypermedia fields.

In hypermedia, the evolution from the creative definition of content to the content-independent organization of the structure of a hypermedia application is attributed to the work on **HDM** [Garzotto et al. 1993], which stresses the difference between authoring in the large, i.e., designing general structure and navigation, and authoring in the small, i.e., deciding the layout and synchronization aspects of specific component types.

HDM, however, did not prescribe a formal development lifecycle, which was first advocated by **RMM** [Isakowitz et al. 1995], where the following seven activities, i.e., *entity relationship design*, *slice design*, *navigation design*, *conversion protocol design*, *interface design*, *behavior design*, and *implementation and testing* are proposed. The first three activities provide a conceptualization of the hypermedia application domain in terms of entities, substructured into slices, and navigable relationships. Conversion protocol design is a technical activity that defines the transformations to be used for mapping the conceptual schema into implementation structures. In addition to defining the development lifecycle, RMM also gives guidelines for slice and navigation design, the two tasks most particular to hypermedia design.

OOHDM [Schwabe and Rossi 1995] takes inspiration from object-oriented modeling and simplifies the RMM lifecycle to only four steps: *domain analysis*, *navigation design*, *abstract interface design*, and *implementation*. In domain design, classical object-oriented techniques are used, instead of the entity relationship model. Navigation design adds specific classes (e.g., node, link, index) to represent different forms of navigation. The same is done for presentation, which is described by means of classes (e.g., button, text field) added during interface design. Implementation then fleshes out the classes identified during design with code in the implementation language of choice.

In the Web context, most methodological proposals concentrate on visual design and usability criteria [Sano 1996], much as the hypermedia field authoring guidelines were the first concern of development. The requirement of application scale-up drives the most recent contributions, like Araneus, Autoweb, Web Architect, and W3I3, which organize the development process into activities drawn both from the above mentioned hypermedia methodologies and from traditional object-oriented and database design methods.

14.3 Other Design Tools

Besides the projects described in Section 13, other research efforts have contributed to the development of prototypes of hypermedia and Web design tools.

RMC [Diaz et al. 1995] gives CASE support to the design and implementation phases of the RMM methodology. It consists of a diagram editor, which assists the input of RMDM schemas, and a code generator, which outputs HTML pages from data stored in an internal repository. Pages are produced offline in a precompiled fashion. RMC is not based on a database architecture, and thus does not provide facilities for scal-

Table X. A Match Between Categories of Web Development Tools and Types of Applications

	Visual editors	Hypermedia tools	Web-DBPL integrators	Form editors	Multi-paradigm tools	Model-driven generators
Small-scale business to customer	X	X			X	
Business to business			X	X	X	X
Large-scale business to customer			X		X	

ing-up and updating the information base.

HSDL [Kesseler 1995] has an architecture similar to RMC, but starts from a HDM-like design model, augmented to support a fine-grain personalization of navigation and presentation semantics. HSDL objects are annotated by means of programs in a scripting language, called *expanders*, which drive the production of HTML pages. Expanders can be attached both to schema elements and to instances, to provide uniformity and exceptional handling at the same time. Expander programming, although powerful, is a low-level task that resembles the programming of page templates in Web-DBPL integrators. The difference is that HSDL, like RMC, does not rely on database technology to store the application content, but has an internal repository.

15. CONCLUSIONS

Although the present situation in the market of Web development tools is characterized by a substantial overlap between the different classes of products (despite some difference in the level of maturity), an attempt can be made to identify a plausible choice, based on the application requirements. Table X summarizes the adequacy of each product category to a specific kind of application:

—*Small-scale business-to-user applications*: this category includes such applications as companies' points of presence, training and education, infocenters, etc. Last-generation visual editors and site managers (Category 1) seem to be the appropriate solution, because they ensure the high level of visual accuracy and customization necessary for application targeted to the general public, coupled with productivity tools, substantially reducing the development effort. This role could be undermined in the near future by hypermedia authoring tools (Category 2), which share the same focus on presentation quality and are even more effective in the design of navigational and multimedia interfaces. The current limit to the applicability of both these classes of solutions is the size and volatility of the information base; if this has to be kept in a database, then presently these tools do not provide the adequate means to integrate databases and the Web for design, implementation, and maintenance. In this case, multiparadigm tools (Category 5) may be a better choice.

—*Intrabusiness, or business-to-business applications*: this category comprises all legacy information systems and EDI applications, and is characterized by a different kind of user, already trained to the transactional and form-based interaction paradigm. In the present situation, Web form editors and database publishing wizard (Category 4) and model-driven generators (Category 6) offer a powerful opportunity for migrating existing applications to Intranets. This technical advantage largely balances the limited exploitation of the communication capabilities of the Web. However, novel intra- and interbusiness applications are emerging (for example, hypermedia for technical documentation and computer-based training), which demand the integration of large masses of data, hypertextual multimedia interfaces, and deployment on the Web. The gradual introduction of these applications may promote the evolution of the interaction paradigm for conventional information systems as well.

—*Large-scale business-to-user applications*: this is the most challenging area, comprising such applications as electronic commerce, virtual libraries, and all sorts of Internet services. Presently, it seems that no specific product or class of products is fully addressing the analysis, design, implementation, and evolution of these kinds of applications, which require the same communication paradigm and interface quality as small-scale user-oriented applications, and the same performance and scalability as client-server database applications. In this scenario, neutral implementation-oriented products like Web-DBPL integrators (Category 3) and flexible, multiparadigm tools (Category 5) seem the most adequate choice, although development and maintenance with these tools still require a substantial coding effort. We are convinced that the best of these tools can be obtained by using them in conjunction with a model-driven development approach, based on design notations and processes like the ones described in the previous section; after the supporting database has been designed, visual page designers and Web-DBPL integrators can be used to generate the application's physical pages.

As several research projects demonstrate, large-scale Web applications have more dimensions than the mere structure of the underlying database, prompting a refocusing of the existing conceptual models and software development processes to achieve the level of maturity of a consolidated software field.

APPENDIX

LIST OF URLS OF REVIEWED PRODUCTS (ALPHABETIC ORDER)

1. Access97, Microsoft, http://www.microsoft.com/access/
2. Altavista Search, Digital, http://www.altavista.software.digital.com/search/index.htm
3. ASP, Microsoft, http://www.microsoft.com/iis/LearnAboutIIS/ActiveServer/default.asp
4. Authorware, Macromedia, http://www.macromedia.com/software/authorware
5. Backstage Designer, Macromedia, http://www.macromedia.com/software/backstage
6. Cold Fusion, Allaire Inc., http://www.allaire.com/products/ColdFusion/31
7. Crystal Report Print Engine, Seagate, http://www.crystalinc.com/crystalreports
8. Delphi Client/Server Suite, Inprise, http://www.inprise.com/delphi
9. Data Director 1.0, Informix, http://www.informix.com/products/tools/datadir
10. Designer 2000, Oracle, http://www.oracle.com/products/tools/des2k/collateral/wwwgen.pdf
11. Developer 2000, Oracle, http://www.oracle.com/products/tools/dev2k/index.html
12. Director, Macromedia, http://www.macromedia.com/director
13. Domino Designer R5, Lotus, http://www.lotus.com/home.nsf/tabs/r5preview4
14. Drumbeat 2.0, Elemental Software, http://www.drumbeat.com
15. Excite for Web Servers, Excite, http://www.excite.com/navigate/home.html
16. Formula Graphics97, Formula Graphics, http://www.formulagraphics.com
17. FrontPage98, Microsoft, http://www.microsoft.com/frontpage
18. Fusion, NetObjects Inc., http://www.netobjects.com/html/nof.html
19. HahtSite, HAHT Software, http://www.haht.com
20. Harvest, Harvest, http://harvest.transarc.com
21. Home Page, Claris, http://www.claris.com/products/claris/clarispage/clarispage.html
22. HotMetal Pro, SoftQuad, http://www.softquad.com/products/hotmetal
23. Iconauthor, Aimtech, http://www.aimtech.com/iconauthor

24. IDC, Microsoft, http://www.microsoft.com/msoffice/developer/access/articles/itk/Idcfaq.htm
25. IntraBuilder, Borland, http://www.borland.com/intrabuilder
26. Jaguar, Sybase, http://www.sybase.com/products/jaguar
27. LiveLink Search and Spider, OpenText, http://www.opentext.com/livelink
28. NetDynamics 4.0, NetDynamics, www.netdynamics.com/product/overview/nd400overview.html
29. Oracle 8i, Oracle, http://www.oracle.com/products/oracle8i
30. Perspecta, Perspecta, http://www.perspecta.com
31. Page Mill/Site Mill, Adobe, http://www.adobe.com
32. PL/SQL Web Development Toolkit, Oracle, http://www.oracle.com
33. PowerBuilder, Sybase, http://www.sybase.com/products/powerbuilder
34. Quest, Allen Communication, http://www.allencomm.com/p&s/software/quest
35. StoryServer, Vignette, http://www.vignette.com/PressKit/ProductOverview.pdf
36. SuperCede, Asymetrix, http://www.supercede.com/products/supercede/dbdetails.html
37. Teamsite, Interwoven, http://www.interwoven.com
38. Toolbook II Instr., Asymetrix, http://www.asymetrix.com/products/toolbook2/instructor
39. Toolbook Assistant, Asymetrix, http://www.asymetrix.com/products/toolbook2/assistant
40. Verity Search 97, Verity, http://www.verity.com
41. VisiBroker for Java, Inprise, http://www.inprise.com/visibroker
42. Visual Basic 5.0, Microsoft, http://www.microsoft.com/vbasic
43. Visual InterDev, Microsoft, http://www.microsoft.com/vinterdev
44. Visual JS, Netscape, http://developer.netscape.com/library/documentation/visualjs/vjs.html
45. WebCompass, Quarterdeck, http://arachnid.qdeck.com/qdeck/products/wc20
46. Web Data Blade, Illustra/Informix, http://www.informix.com
47. WebObjects, Apple, http://software.apple.com/webobjects

REFERENCES

ATZENI, P., MECCA, G., AND MERIALDO, P. 1998. Design and maintenance of data intensive Web sites. In *Proceedings of the Sixth International Conference on Extending Database Technology* (Valencia, Spain, Mar.), H. -J. Schek, F. Saltor, I. Ramos, and G. Alonso, Eds. 436–450.

MECCA, G., ATZENI, P., MASCI, A., SINDONI, G., AND MERIALDO, P. 1998. The Araneus Web-based management system. *SIGMOD Rec. 27*, 2, 544–546.

ATZENI, P., MECCA, G., AND MERIALDO, P. 1997. To weave the Web. In *Proceedings of the 23rd International Conference on Very Large Databases* (VLDB '97, Athens, Greece, Aug.), M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, Eds. VLDB Endowment, Berkeley, CA, 206–215.

BACHIOCHI, D., BERSTENE, M., CHOUINARD, E., CONLAN, N., DANCHAK, M., FUREY, T., NELIGON, C., AND WAY, D. 1997. Usability studies and designing navigational aids for the World Wide Web. *Comput. Netw. ISDN Syst. 29*, 8-13, 1489–1496.

BARNES, H. AND GWYER, M. 1996. Designer/2000 Web enabling your applications, Oracle white paper. Tech. Rep.. Oracle Corp., Redwood City, CA.

BATINI, C., CERI, S., AND NAVATHE, S. B. 1992. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings series in computer science. Benjamin-Cummings Publ. Co., Inc., Redwood City, CA.

BENDA, M., Ed. 1998. Internet Architecture. *IEEE Internet Comput. 2*.

BIGGERSTAFF, T. J. AND PERLIS, A. J., Eds. 1989. *Software Reusability: Vol. 1, Concepts and Models*. ACM Press, New York, NY.

CERI, S., FRATERNALI, P., PARABOSCHI, S., AND POZZI, G. 1998. Consolidated specification of WWW intelligent information infrastructure (W3I3). Tech. Rep. Dip. di Elettronica e Informazione, Politecnico di Milano, Milan, Italy.

CHANG, P. I. 1998. Inside the Java Web server: An overview of Java Web server 1.0, Java servlets, and the JavaServer architecture. http://java.sun.com/features/1997/aug/jws1.html.

CHEN, P. P. 1976. The entity-relationship model: Toward a unified view of data. *ACM Trans. Database Syst. 1*, 1, 9–36.

DIAZ, A., ISAKOWITZ, T., MAIORANA, V., AND GILABERT, G. 1995. RMC: A tool to design WWW applications. In *Proceedings of the Fourth International Conference on World Wide Web* (Boston, MA), 11–14.

FERNÁNDEZ, M., FLORESCU, D., KANG, J., LEVY, A., AND SUCIU, D. 1998. Catching the boat with

Strudel: Experiences with a Web-site management system. *SIGMOD Rec. 27*, 2, 414–425.

FLORESCU, D., LEVY, A., AND MENDELZON, A. 1998. Database techniques for the World-Wide Web: A survey. *SIGMOD Rec. 27*, 3, 59–74.

FRATERNALI, P. AND PAOLINI, P. 1998. A conceptual model and a tool environment for developing more scalable and dynamic Web applications. In *Proceedings of the Sixth International Conference on Extending Database Technology* (Valencia, Spain, Mar.), H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso, Eds. 421–435.

GARG, P. K. 1988. Abstraction mechanisms in hypertext. *Commun. ACM 31*, 7 (July 1988), 862–870.

GARZOTTO, F., MAINETTI, L., AND PAOLINI, P. 1995. Hypermedia design, analysis, and evaluation issues. *Commun. ACM 38*, 8 (Aug. 1995), 74–86.

GARZOTTO, F., MAINETTI, L., AND PAOLINI, P. 1994. Adding multimedia collections to the Dexter model. In *Proceedings of the 1994 ACM Conference on Hypermedia Technology* (ECHT'94, Edinburgh, Scotland, Sept. 18–23), I. Ritchie and N. Guimarães, Eds. ACM Press, New York, NY, 70–80.

GARZOTTO, F., PAOLINI, P., AND SCHWABE, D. 1993. HDM—a model-based approach to hypertext application design. *ACM Trans. Inf. Syst. 11*, 1 (Jan. 1993), 1–26.

GARZOTTO, F. AND MAINETTI, L. 1993. HDM2: Extending the E-R approach to hypermedia application design. In *Proceedings of the 12th International Conference on Entity Relationship Approach* (ER'93, Dallas, TX), R. Elmasri, V. Kouramajian, and B. Thalheim, Eds. 178–189.

GARZOTTO, F., PAOLINI, P., AND SCHWABE, D. 1991. HDM—a model for the design of hypertext applications. In *Proceedings of the 3rd Annual ACM Conference on Hypertext* (San Antonio, TX, Dec. 15–18), J. J. Leggett, Ed. ACM Press, New York, NY, 313–328.

GWYER, M. 1996. Oracle Designer/2000 Web-Server generator (vers. 1.3.2). Oracle Corp., Redwood City, CA.

HALASZ, F. AND SCHWARTZ, M. 1994. The Dexter hypertext reference model. *Commun. ACM 37*, 2 (Feb. 1994), 30–39.

HALASZ, F. G. 1988. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Commun. ACM 31*, 7 (July 1988), 836–852.

HARDMAN, L., BULTERMAN, D. C. A., AND VAN ROSSUM, G. 1994. The Amsterdam hypermedia model: Adding time and context to the Dexter model. *Commun. ACM 37*, 2 (Feb. 1994), 50–62.

HORTON, W., TAYLOR, L., IGNACIO, A., AND HOFT, N. L. 1996. *The Web Page Design Cookbook: All the Ingredients You Need to Create 5-Star Web Pages*. John Wiley and Sons, Inc., New York, NY.

HOVEN, I. V. 1997. Deploying Developer/2000 applications on the Web, Oracle white paper. Oracle Corp., Redwood City, CA.

HYPERWAVE INFORMATION MANAGEMENT, 1998. Hyperwave User's Guide, Version 4.0. Hyperwave Information Management.

ISAKOWITZ, T., STOHR, E. A., AND BALASUBRAMANIAN, P. 1995. RMM: a methodology for structured hypermedia design. *Commun. ACM 38*, 8 (Aug. 1995), 34–44.

JACOB, R. J. K. 1983. Using formal specifications in the design of a human-computer interface. *Commun. ACM 26*, 4 (Apr. 1983), 259–264. http://www.eecs.tufts.edu/˜jacob/papers/cacm.txt; http://www.eecs.tufts.edu/˜jacob/papers/cacm.ps.

JACOBSON, I. 1992. *Object-Oriented Software Engineering*. ACM Press, New York, NY.

KESSELER, M. 1995. A schema-based approach to HTML authoring. In *Proceedings of the Fourth International Conference on The World Wide Web* (Boston, MA),

LANGE, A. 1997. Sorting through search engines. *Web Tech. M. 2*, 6 (June). http://www.webreview.com/97/06/13/webtech/index.html.

MADSEN, K. H. AND AIKEN, P. H. 1993. Experiences using cooperative interactive storyboard prototyping. *Commun. ACM 36*, 6 (June 1993), 57–64.

MICROSOFT CORPORATION 1996. The distributed common object model. Microsoft Corp., Redmond, WA. http:/msdn.microsoft.com/workshop/components/contents.htm.

MILLER, J., SHETH, A., KOCHUT, K., AND PALANISWAMI, D. 1997. The future of web-based workflows. In *Proceedings of the International Workshop on Research Directions in Process Technology* (Nancy, France).

MONTGOMERY, J. 1997. Distributing components. *BYTE 22*, 4, 93–98.

MYERS, B. A. 1995. User interface software tools. *ACM Trans. Comput. Hum. Interact. 2*, 1 (Mar. 1995), 64–103.

MYERS, B., HOLLAN, J., CRUZ, I., BRYSON, S., BULTERMAN, D., CATARCI, T., CITRIN, W., GLINERT, E., GRUDIN, J., AND IOANNIDIS, Y. 1996. Strategic directions in human-computer interaction. *ACM Comput. Surv. 28*, 4, 794–809.

NANARD, J. AND NANARD, M. 1995. Hypertext design environments and the hypertext design process. *Commun. ACM 38*, 8 (Aug. 1995), 49–56.

NIELSEN, J. 1990. *Hypertext and Hypermedia*. Academic Press Prof., Inc., San Diego, CA.

NIELSEN, J. 1996. Computer Science and Engineering Handbook. CRC Press, Inc., Boca Raton, FL.

OBJECT MANAGEMENT GROUP 1998. The common object request broker: Architecture and specification. Ver. 2.0. Tech. Rep. Object Management Group, Framingham, MA. http://www.omg.org/corba/corbiiop.htm.

RUMBAUGH, J., BLAHA, M., PREMERLANI, W., EDDY, F., LORENSEN, B., AND LORENSON, W. 1991. *Object Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, NJ.

SANO, D. 1996. *Designing Large-Scale Web Sites: A Visual Design Methodology*. John Wiley and Sons, Inc., New York, NY.

SCHWABE, D. AND ROSSI, G. 1995. The object-oriented hypermedia design model. *Commun. ACM 38*, 8 (Aug. 1995), 45–46.

SCHWABE, D., CALOINI, A., GARZOTTO, F., AND PAOLINI, P. 1992. Hypertext development using a model-based approach. *Softw. Pract. Exper. 22*, 11 (Nov. 1992), 937–962.

STOTTS, P. D. AND FURUTA, R. 1989. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Trans. Inf. Syst. 7*, 1 (Jan. 1989), 3–29.

SUN MICROSYSTEMS, 1995. Remote method invocation specification. Tech. Rep. Sun Microsystems, Inc., Mountain View, CA. http://java.sun.com/products/jdk/rmi/index.html.

TAKAHASHI, K. AND LIANG, E. 1997. Analysis and design of Web-based informations systems. In *Proceedings of the Sixth International Conference on the World Wide Web* (Santa Clara CA, Apr.),

WONG, W. 1998. Team-building on the fly. *BYTE 23*, 2 (Feb.). http://www.byte.com/art/9802/sec10/art1.htm.

WORLD WIDE WEB CONSORTIUM, 1998. Cascading style sheets: Level 2 specification. Tech. Rep. World Wide Web Consortium. http://w3c.org/TR/REC-CSS2

WORLD WIDE WEB CONSORTIUM, 1998. The document object model (DOM): Level 1 specification. Tech. Rep. World Wide Web Consortium. http://www.w3.org/TR/REC-DOM-Level-1/.

ZHENG, Y. AND PONG, M.-C. 1992. Using statecharts to model hypertext. In *Proceedings of the ACM Conference on Hypertext* (ECHT '92, Milan, Italy, Nov. 30–Dec. 4), D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, Eds. ACM Press, New York, NY, 242–250.

Received: March 1998; revised: August 1998; accepted: December 1998