Java Framework for Database-Centric Web Site Engineering

Beat Signer, Michael Grossniklaus and Moira C. Norrie
{signer, grossniklaus, norrie}@inf.ethz.ch
Institute for Information Systems
ETH Zurich
CH-8092 Zurich, Switzerland

ABSTRACT

We present a database-centric approach to web site development in which both application and web content data are managed by a database. The development process is based on three main stages of information modelling, document content design and presentation specification. A Java framework based on the OMS object-oriented data management system has been developed to support the development life cycle from rapid prototyping through to operation. We describe how the framework supports access from heterogeneous clients and how it has been extended to include a web content manager.

KEYWORDS: web server frameworks, web content management, web databases

1 INTRODUCTION

Experiences in the development and maintenance of complex web sites have led to the development of a large variety of new products and tools for web site engineering. These range from full-scale, integrated development and operational environments to various forms of stand-alone tools to support part of the development process. In particular, the general problem of developing and managing data intensive web sites has been addressed by a new generation of web content management systems. The prime objective of such systems is to move the developer away from the level of HTML documents to instead think in terms of document contents and to assist in the general maintenance problem of ensuring consistency of content, reference and presentation.

In this paper, we review the requirements and features of existing web content management systems and discuss how existing products and research systems tend to approach the problem from one of two extremes — either a database-oriented approach or a document-oriented approach. We describe how our framework for web site engineering aims to support both views by managing both application data and document content data within a single data management system. Further, although we integrate the management of application and document data within a single system, we make a clear separation of the two notions. We thus separate not only content and presentation, but also information objects and document content.

The framework we present is based on OMS Java, a data management framework for the Java environment [1, 2]. The OMS Java framework has been extended to support access from various forms of web clients such as HTML browsers and the WML (Wireless Markup Language) browsers of mobile phones using the Wireless Application Protocol (WAP). This OMS Java web server framework is based on the use of Java servlets and XML. In a second phase, we have extended the framework further to support web content management by managing not only application data, but also document content and presentation data.

We begin in Sect. 2 with a description of related work in terms of existing web content management systems, their requirements and some of their shortcomings. Section 3 then outlines our general approach to web site development, which is centred on the use of an operational database at all stages. We describe the general OMS Java web server framework in Sect. 4 and how it supports universal client access. Section 5 then describes how a web content manager, OMS Java CMS, has been developed based on this framework. Concluding remarks are given in Sect. 6.

Konstanzer Online-Publikations-System (KOPS)
URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-0-326819

2 WEB CONTENT MANAGEMENT SYSTEMS

Web content management systems such as Vignette [3], Obtree C3 [4] and Interwoven [5] can be considered as a second generation of products designed to support web site engineering. The first generation tended to be based on CGI and scripting languages and the web sites generated proved difficult to manage in terms of ensuring consistency and supporting evolution. Web content management systems were designed to address these issues by providing database solutions to manage the content of a web site.

The services offered by web content management systems vary according to the target market. Some address the complexities of large-scale systems in terms of project management, performance, transactions, workflows and integration. At the other end of the range are the products aimed at the management of relatively small, static web sites. However, there are general requirements for web content management systems listed in the whitepapers of various products e.g. [6, 7]. These typically include:

- separation of content and presentation
- support for multi-lingual web sites
- support for on-line updating of content

The requirement to separate content from presentation is now widely accepted as desirable. It is particularly important when a web site is to be accessible from various forms of client devices such as mobile phones or PDAs and not just HTML browsers. However, while this requirement is often quoted, many solutions in practice do not make such a clean separation. It is still the case that one sees products with hard-coded statements for generating HTML.

An important part of globalisation is world-wide access to web sites. This means it is critical that content can be published in different languages and that it be easy to switch between languages. It is therefore important to distinguish between components of a document and the contents to be delivered in a specific context. Again this is an aspect that, while supported in some systems, it is often an “add-on” rather than an integrated part of the system.

On-line updating of content involves not only the editing of content, but also the creation of new documents and document components. These components must be completed and approved before being published. It is therefore necessary that a web content management system supports component life-cycles and will filter out components that are under development and not yet active.

A general problem of most current web content management systems is the fact that they tend to address the problem of web content management from the side of document generation and management rather than from that of semantic information content. In contrast, many of the research proposals from the side of web information systems tend to focus on the semantic information content in terms of presenting the contents of a database on the web (see for example [8, 9, 10, 11]). However, they tend to neglect the operational aspects of developing and managing a complex web site.

We therefore extend the requirements of web content management to include the management of information objects as well as document content and presentation. By employing object-oriented technologies and our own OMS Java data management system, we were able to integrate facilities for web content management into our OMS Java framework.

3 DATABASE-CENTRIC APPROACH TO WEB SITE ENGINEERING

We strongly believe in rapid prototyping and a process of refinement as the basis for all kinds of information system development [12]. We therefore do not distinguish between stages of design and implementation, but rather introduce operational systems as early as possible into the development cycle. These prototypes form the basis for interaction between the developer and the customer and the validation of the underlying information models.

With respect to web site engineering, we also consider that two other factors are critical. First, it is necessary to separate content from presentation and, second, the information model is central. The first of these two factors is essential in providing universal access from a variety of clients — be they different web browsers, mobile phones, PDAs or whatever other devices will appear in the near future. Also, an important requirement in many web sites is support for multi-lingual presentation. The second factor of placing the information model at the centre of development is in part recognition of the fact that a web site is an interactive information system. It is important therefore that we separate not only content from presentation, but also the information objects from the document content. This separation enables us to manage semantic entities of the application rather than simple document components such as pieces of text or images.

Our approach leads to the general three-stage development process shown in Fig. 1. The first stage concerns the requirements analysis and design of the information model in terms of application entities, their roles and also associations between entities. At this stage, a database can already be created and, through the use of generic browsers, the database can be both viewed and updated via the web.

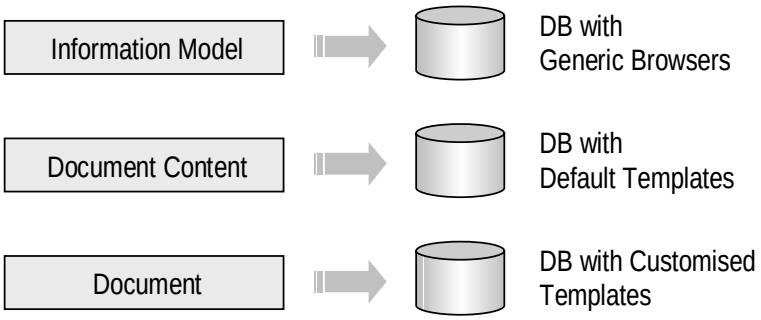

Figure 1: Database-Centric Web Site Development

The second stage is to move from the information model to the web document model. In the case of many web databases, there is assumed to be a one-to-one correspondence between application entities and document contents [13]. This can be over-restrictive, especially since web documents have tended to become more complex in terms of structure as we see with web portal sites etc. We therefore separate these two levels and consider web documents as complex structures built from web components which in turn are associated with application objects. This stage therefore involves the specification of document content, but as yet without the worries of presentation. Default templates for web components are used to generate the presentations. We note that, at this stage, the interactive aspects of a web site must also be considered [8]. This includes navigation which may be modelled using hypertext diagrams and workflows which model specific use role interaction sequences.

At the third stage, the document presentations are specified in terms of customised templates for the various components. It is important to note that a component may have more than one associated template. Each component has at least one template for every type of user agent such as web browsers requesting HTML or mobile phones with WAP support asking for WML. Furthermore a component may have additional context specific templates.

This database-centric approach is supported by the OMS suite of tools and technologies that have been developed within the Global Information Systems research group at ETH Zurich. Initially, our work on web databases focussed on architectures for web databases and client-side components [14]. These components included generic browsers and editors and also prefetching caching components [15]. Two forms of generic browser were used — one was client-based using Java applets and the other server-based using CGI and Java for the dynamic generation of HTML documents. A presentation editor was also implemented which allowed developers to graphically edit the presentation of individual objects or object types and to specify whether they should be displayed as HTML documents or as applet frames. While this system provided an easy way to develop web sites, the limitation always remained that what we were in effect providing was a web browser for our database, rather than a database for a web site. In other words, there was a one-to-one correspondence between OMS database objects and web documents and we did not have the desired separation between the notions of application objects and web documents. This led to various experiments with building a notion of web documents and workflows into the OMS database system [16].

At the same time, the development of other web-related technologies such as new markup languages for new types of user agents (e.g. mobile phones with WAP support) and also the emergence of XML together with XSLT as a means of separating document content and presentation led us to develop a second more general framework for web site engineering [17]. This framework is implemented in the Java environment as part of the OMS Java application framework and data management system [1, 2]. As a first stage, we focussed on the issue of providing universal client access as described in the next section. The second stage has then been to extend the framework to support the task of specifying web document structures and presentations which is described in Sect. 5.

4 UNIVERSAL CLIENT SUPPORT

Many existing tools for web site engineering either rely on a specific form of client such as an HTML browser, or they require that different forms of clients be handled separately meaning that it requires significant development time to port an application to another form of client device. We wanted a universal client framework that required minimal development effort to access a web site via another form of client device. Any additional effort would rather go into optionally customising the presentation to suit the device in question. Such general frameworks are particularly important when one considers how dynamic the world of mobile devices currently is. There are many questions as to whether technologies such as WAP will really become established or whether they will be replaced by either new technologies or new devices better capable of handling existing technologies.

In Fig. 2, we show the general web server architecture for our OMS Java framework. For a specific application, all client access is via a single Java servlet — the Entry Servlet. This means that only a single URL needs be known for a specific web site, rather than having different URLs for different types of clients. The Entry Servlet detects the user agent type from the HTTP request header and delegates the handling of the request to the appropriate servlet. For example, we show in Fig. 2 servlets to handle requests from HTML browsers, XML browsers and also WAP phones in terms of WML browsers.

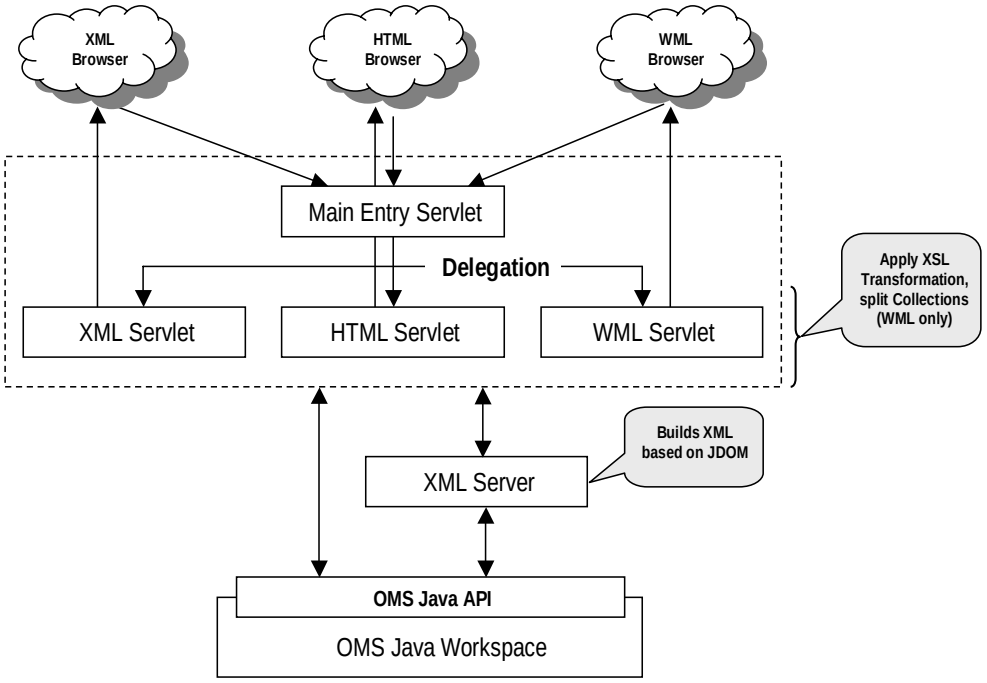

Figure 2: OMS Java Web Server Architecture

The request handling servlets then access the database by connecting to an OMS Java workspace via the OMS Java API. The connection may either be direct or via the OMS Java XML server. Direct connections deal with requests that do not involve data retrieval such as checking the membership of an object in a collection or performing updates. Any requests that involve the retrieval of data go through the XML server. The XML server forwards requests to the OMS Java workspace and generates XML representations for any data objects to be returned to the servlets. The requesting servlets then use the appropriate XSLT templates to transform the XML results to the corresponding responses to be returned to the client.

There are a few points to note in this general architecture. First, we are not storing any XML documents, but rather generating them dynamically from the application data which is stored according to the information model. Since the information model is a loosely-coupled graph model based on object collections and associations, this gives much more flexibility than the rather restrictive hierarchical-based models imposed by XML structures. At access time, a particular hierarchical view on data is derived and the appropriate XML structure generated. Second, since what we

are interested in is the XML structure rather than the document per se, what we generate as an intermediate form for the XSLT processor located at the processing by the servlet is actually the associated DOM (document object model) structure rather than the XML document.

Using generic XSLT templates for the various client devices, we are able to provide generic browsers and editors for the current set of client types. Adding a new client type involves implementing the corresponding servlet and writing the appropriate XSLT templates. Our initial framework had the client set for HTML, XML and WML browsers and we were able to add support for NTT DoCoMo's Imode internet access system (mainly used by mobile phones in Japan) with only a few hours work.

In the case of mobile phones, there are severe limitations in terms of both the screen size and also the memory size. For example, WAP phones typically have a deck size limited to around 1.5 kB. This means that there are problems in handling large objects such as arise with collections of similar objects. For example, in the case of an electronic address book, the user may start by browsing a collection of persons and then select the person of interest. This collection is likely to exceed the limited deck size. In the case of a generic browser, we therefore have to find a way of automatically splitting large objects. This is done by first transforming the DOM tree to WML and then going back and reducing the tree if the generated WML exceeds the maximum deck size. In this way, collections can be split into sections and each section fetched separately. Since the output size varies from object to object and the resulting document size is also dependent on the applied XSLT stylesheet, the splitting must be done dynamically. To support the prediction of the correct splitting size, a splitting factor is stored for each object type and updated only as and when necessary.

Specific application interfaces are supported through the customisation of XSLT templates. As an example, a community agenda was implemented which manages an electronic diary for a user community. Figure 3 shows the client interfaces for an HTML browser and also a WAP phone.

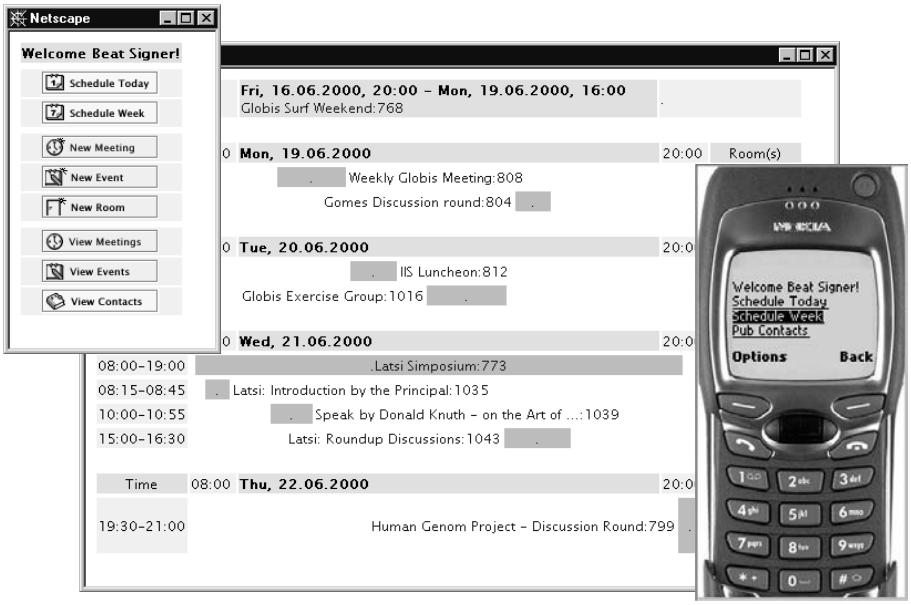

Figure 3: Community Agenda System

The window on the left-hand side of Fig. 3 is the HTML browser interface for individual users logging on to the system. Clicking on the Schedule Week icon then causes the HTML document shown in the centre of the figure to be displayed. In the case of the WAP phone, clearly such a layout for a weekly schedule is not feasible. In this case, the user is presented with minimal options and a weekly or daily schedule is presented as a scrollable list of links to various events.

While the development of the community agenda application demonstrated the benefits of the OMS Java web server architecture in terms of development time and also site maintenance, it is still tiresome to compose existing information

objects to build application specific web objects. The next stage therefore was to extend the framework in terms of web content management support and this is described in the next section.

5 OMS JAVA CMS

OMS Java CMS is the name given to the extended OMS Java framework with in-built support for web content management. The role of the OMS Java CMS is to facilitate the deployment and management of a web site by providing a dynamic and flexible solution for the context dependent generation of web documents. To achieve this, the OMS Java framework has been extended with pre-defined object types and collections for the management of document structures, contents and presentations [18].

The resulting OMS Java schema is rather large and complex so we use Fig. 4 to give an overview of the main parts of the schema.

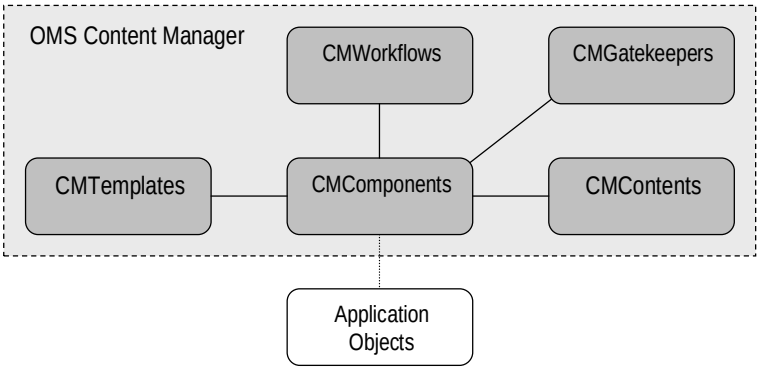

Figure 4: OMS Java CMS Schema Parts

Documents are composed from components which are modelled by the `CMComponents` part of the CMS schema. Components are represented by objects of type `cmcomponent` and these are specialised into subtypes corresponding to the various component types such as picture, text, URL, link and directory (see Fig. 5). But there are also computed component objects providing special services such as automatically generated site maps and navigations. In contrast to most existing solutions however, our system does not only allow composition of structured documents from pre-defined types, but also supports the definition of complex user-defined types.

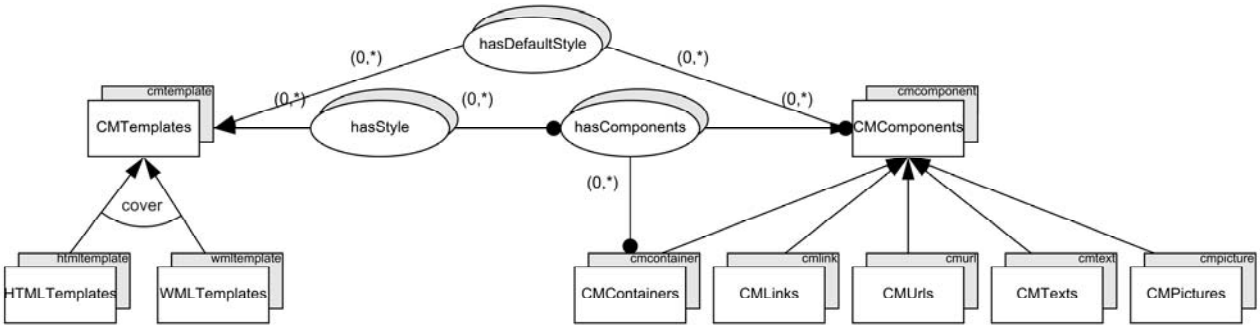

Figure 5: Association of Document Components and Templates

In addition a component may be of type container to represent a recurring component group. Over these containers we introduced the notion of context. This concept is used in our system to perform a different rendering of components of the same object type based on the context of their occurrence. For example, an image in a table container can be rendered differently from an appearance of the same image in a text container.

The rendering of objects is based on XSLT templates which are also stored in the database as indicated by the schema part `CMTemplates`. However, the transformation is not based on a single static XSLT stylesheet as is the case in numerous existing web content management systems, but rather is performed using a dynamically generated stylesheet. Each component may be associated with a default template and, optionally, several context specific templates. A template is selected based on the position of the object within a container to build the stylesheet which is used for the transformation of an entire document.

As discussed previously, a key requirement of content management is multi-lingual support. This is supported by a separation of components and contents (see Fig. 6). The `CMContents` part of the schema is used to represent language-dependent parts of components. A `cmcomponent` object may be associated with several `cmcontent` objects — one for each supported language of the web site. Note that this separation of components and contents can be used in any case where one has different content versions such as different image formats or abbreviated versions of texts for use with mobile phones.

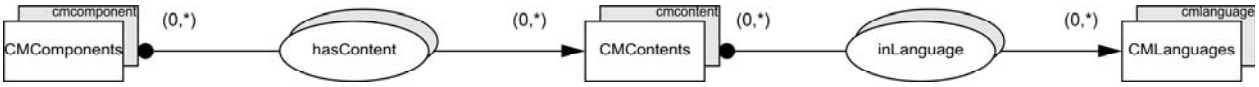

Figure 6: Separation of Components and Language-Dependent Contents

Document components have a life cycle usually consisting of development, approval and active phases. The objects in the development and approval phases are in the database, but should not be published on the web site. In OMS Java CMS, every component can be associated with a user-defined workflow state as represented by the `CMWorkflows` schema part. Workflow states can be linked into an overall workflow graph to control the evolution of components.

The `CMGatekeepers` schema part models document component filters known as gatekeepers. Given the criteria implemented by specific gatekeeper objects, the publication of document components is controlled. For example, as described above, one specific gatekeeper will only allow components with workflow state ‘active’ to be included in a given web document. Other uses are to control the time period during which a component is visible.

6 CONCLUSIONS

We have presented a general framework for web site engineering that supports both universal client access and requirements of web content management. The framework was developed in two stages. The first stage was to build a general framework that enabled access to information objects stored in an application database via various forms of user agents. The second stage was to extend the database to also manage web content objects. Web documents are dynamically composed from document parts with the content and presentation determined by context.

The second stage is on-going work and the framework is currently being evaluated. Future extensions will include user and user role management. In addition, we propose to develop interactive tools to assist the process of creating and managing the web content objects in the database

REFERENCES

1. A. Kobler and M. C. Norrie. OMS Java: Lessons Learned from Building a Multi-Tier Object Management Framework. In *Proceedings of the Workshop on Java and Databases: Persistence Options*, Denver, USA, January 1999.

2. A. Steiner, A. Kobler, and M. C. Norrie. OMS Java: Model Extensibility of OODBMS for Advanced Application Domains. In *Proceedings of the 10th Conference on Advanced Information Systems Engineering (CAiSE’98)*, Pisa, Italy, June 1998.

3. Vignette Corporation. Vignette Content Management Server. *Whitepaper*, February 2001.

4. Obtree Technologies Inc., http://www.obtree.com.

5. Interwoven Inc. Application Development Using Interwoven: Version 1.1. *Whitepaper*, December 2000.

6. Percussion Software Inc. Rhythmyx Content Manager - Virtual XML Architecture for Automating Web Content Management. *Whitepaper*, 2000.

7. Siebel Systems Inc. Siebel Janna Contact Enterprise 2000. *Whitepaper*, November 2000.

8. P. Atzeni, S. Ceri, S. Paraboschi, and R. Torlone. *Database Systems: Concepts, Languages and Architectures*. McGraw-Hill, 1999.

9. S. Ceri, P. Fraternali, and S.Paraboschi. Design Principles for Data-Intensive Web Sites. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 28(1), 1999.

10. D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 27(3), 1998.

11. M. Fernández, D. Suciu, and I. Tatarinov. Declarative Specification of Data-Intensive Web Sites. In *Proceedings of the 2nd Conference on Domain-Specific Languages*, Berkeley, CA, October 1999.

12. A. Kobler, M. C. Norrie, and A. Würgler. OMS Approach to Database Development through Rapid Prototyping. In *Proceedings of the 8th Workshop on Information Technologies and Systems (WITS'98)*, Helsinki, Finland, December 1998.

13. A. Erni and M. C. Norrie. Approaches to Accessing Databases through Web Browsers. *INFORMATIK, Journal of the Swiss Informaticians Society*, October 1998.

14. M. C .Norrie. Client-Server Database Architectures for the Web. In *Advances in Databases and Multimedia for the New Century - A Swiss/Japanese Perspective*. World Scientific Publishing, 2000.

15. B. Signer, A. Erni, and M. C. Norrie. A Personal Assistant for Web Database Caching. In *Proceedings of the 12th International Conference on Advanced Information Systems Engineering, CAiSE'2000*, Stockholm, Sweden, June 2000.

16. A. Deller. Framework for Web Information Systems Based on OMS. Master's thesis, Institute for Information Systems, ETH Zurich, 1999.

17. M. Kistler. OMS/XML - WAP and HTML Interface based on an XML Server Component for OMS Java. Master's thesis, Institute for Information Systems, ETH Zurich, 2000.

18. M. Grossniklaus. CMServer - An Object-Oriented Framework for Website Development and Content Management. Master's thesis, Institute for Information Systems, ETH Zurich, 2001.