IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4723

# MODE: A Tool for Conceptual Modeling of Web Applications

Mario Bochicchio and Nicola Fiore
Set-Lab, Software Engineering and Telemedia Laboratory
Department of Innovation Engineering, University of Lecce
Via per Arnesano – 73100 – Lecce, Italy
{mario.bochicchio, nicola.fiore}@unile.it

## ABSTRACT

*Web application designers should be able to integrate different methods and techniques in order to correctly organize the overall design-implementation-test process. Web applications, in fact, blend navigation and browsing capabilities, common to hypermedia, with "classical" operations and transactions, common to traditional information systems. Thus, new modeling requirements constantly arise, and hypermedia design models and tools must constantly evolve.*

*In this paper we present MODE, a tool for conceptual modeling of web applications based on W2000 methodology. MODE can be used on its own - to support the design activity and produce all related design documents - or in conjunction with JWeb, a fast prototyping system for Web applications.*

## 1. INTRODUCTION

The conceptual model is the basis of the design [1], and developing a conceptual model as the first step in the design yields many benefits in the later stages of application development. In developing Web applications the conceptual modeling phase is a complex task that requires the integration of various methods and techniques. Web applications, in fact, blend navigation and browsing capabilities (common to hypermedia) with "classical" operations and transactions, common to traditional information systems. Unfortunately, the design methods from software engineering, hypermedia design and data modeling cannot just be borrowed and used together. On the contrary, to create a complete methodology and a viable conceptual framework for Web application modeling, a major integration effort is required.

In this scenario new modeling requirements constantly arise, and hypermedia design models must constantly evolve. A good example of this is HDM (Hypertext Design Model); since its first definition in 1991 [2], it has given rise to a family of variants (HDM2, HDM-lite, …). The latest revision, called HDM2000[3], enriches the previous versions with concepts and annotations in order to model complex information structures as well as operations and services accessible through the Web.

Following HDM and its evolution, the authors developed Jweb [4], a tool to support Web application designers during the whole development process: from conceptual design to fast-prototyping of final applications. Specifically, Jweb provides a set of functionalities enabling the designer to specify, document, re-use, and prototype design choices efficiently.

Unfortunately, due to the very fast evolution of HDM, Jweb was always one step behind.

Indeed, we have noted how the evolution of a CASE tool is not as smooth as the evolution of the conceptual model it supports, and leads to an almost chronic misalignment. This is why, before starting the development of the latest version of our tools to fully support HDM2000, we decided to investigate new solutions.

Accordingly, in this paper we discuss an HDM schema editor named MODE, designed to overcome the previously-stated restrictions.

Section 2 sets out the position of our work in relation to the literature. An overview of HDM2000 and the Jweb project is given in section 3. The requirements of MODE are presented in section 4. Considerations regarding the current implementation of MODE are presented in section 5. An example of use is given in section 6. In section 7 we present our conclusions and discuss future work

## 2. THE STATE OF THE ART

Various methodologies have been presented in the context of hypermedia design, including W2000[2], WebML[5], ARANEUS[6], RMM[7] and OOHDM[8]. Starting from these methodologies a number of tools have been proposed in the literature. These tools, well defined from the theoretical point of view, and very effective in terms of results in the conception and creation of "traditional" hypertexts or hypermedia, can be unsuitable for more complex Web applications. For this type of application new modeling requirements constantly arise, and hypermedia design models must constantly evolve. In this context the most thorough model is, in our opinion, HDM2000 (and the related W2000 methodology), due to its ability to integrate a number of considerations which arise when a complex Web application is designed:

- *customization aspects*: defining a uniform mechanism to provide web applications with the necessary flexibility with respect to both *context-awareness* and *personalization*.
- *requirements elicitation*: defining the *goals* (objective that the stakeholders would like the application to satisfy) and the *requirements* (lower-level objectives that the system is supposed to meet, and which can be directly understood and realized by designers) of the applications. Unlike traditional goal-oriented approaches, the method explicitly takes into account the requirements associated with the Web *environment* which are necessary to design run-time customization mechanisms.
- *operational aspects:* describing the functionalities available to the user
- *transactional aspects:* describing complex sequences of operations and their execution semantics. The transaction design model provides a set of concepts that allows the designer to specify the *semantics* of the transactional operations defined for a web application, regardless of his choice of a specific transaction model (e.g. the ACID model, nested, open-nested, etc.).

These aspects are generally not taken into account, by tools like Araneus [6] or WebRatio [5] (based on WebML methodology), which stress the distinction between data structure, navigation, and presentation instead.

## 3. MODE AND THE JWEB PROJECT

Before describing the actual architecture of MODE, a few words are needed about the HDM 2000 modeling language, the related W2000[2] methodology and the JWeb design and prototyping environment.

In brief, W2000 methodology requires that after the indispensable Requirement Analysis phase, conducted in accordance with a goal-oriented approach, the following steps are performed:

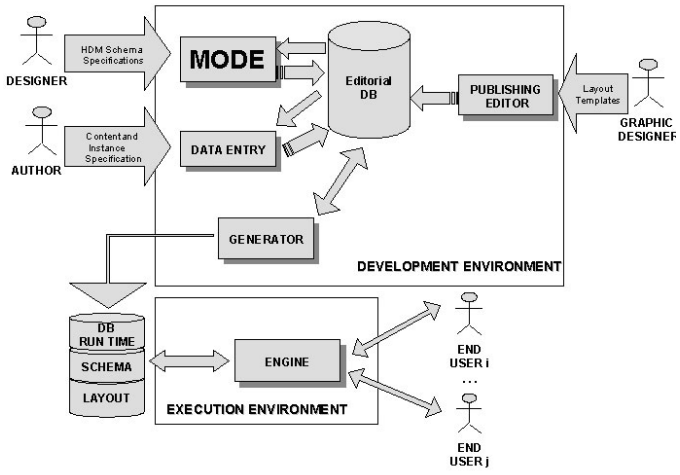
- **Information Design**: the goal is to describe the information that the application is going to deal with, giving it a structured organization. The schema is organized into two different parts: the hyperbase schema and the access schema. The hyperbase schema describes the basic navigational capabilities offered by the application whose components are "Entities" (organized into Entity types) and "Links" (organized into Link types). The access schema describes the organization of the access structures ("Collections" in HDM terminology). Both for the hyperbase schema and the access schema there is a sharp distinction between design in-the-large, where the general features of the design are defined, and design in-the-small, where the details are provided.
- **Navigation Design**: this makes clearer the most important aspect of hypermedia applications, reconsidering the information and its organization more typically from the viewpoint of its fruition and defining the navigational paths the user can follow.
- **Publishing Design**: what emerges from the previous steps must be completed with considerations on the desired presentation strategy, and organized into "pages" and "fruition units".
- **Operations Design**: this is the step in which all the functional and transactional features (such as "register", "submit", etc.) beyond the pure hypermedia paradigm are modeled. Here the model allows the user to invoke the "functionalities" of the application.

HDM is the modeling language of W2000. HDM is an UML extension able to support the previously described design steps required by W2000.

The JWeb system is a design environment with fast prototyping capabilities, based on the HDM language, created to support the design of the hyperbase and access structures, to allow the creation of a repository environment and to allow "execution" of the application in order to "visualize" the implications of the design. The JWeb system is made up of two main parts as shown in Figure 1:

- development, for defining and customizing the application and the data for a particular set of users and/or a particular presentation using HDM2000, translating the detailed design into a relational database schema and inserting a sample of actual data in order to create an instance of the application;
- execution, to deliver the application on different devices starting from the HDM schema and from the content repository defined in the previous steps.

*Figure 1: JWeb System*




The overall environment is modular and flexible. The designer, therefore, does not need to use all the available features in all situations.

## 4. REQUIREMENTS

In this section we examine the most important features of MODE: we define the goals to be reached by the application, the user types and the conceptual requirements to be satisfied by the application for each different user type.

### 4.1 User Types

MODE is one of the results of the UWA (Ubiquitous Web Applications) European research project (www.uwaproject.org). The tool is being used at University level in our "Software Engineering", "Data Base" and "Web Design" courses, and for our research projects. In accordance with our goals, MODE has been designed and developed for the following user types:

- **Expert Designer**, who develops complex and articulated applications. He requires support from the first steps of the design phase, when the application's functionalities and structure are incompletely defined, to the final step which consists of design evaluation. The Expert Designer needs to create an application *ex-novo* using design patterns to speed up the development process; he needs to personalize the application's existing models or derive the application from families of models.
- **Rookie Designer,** who develops low complexity applications. He needs a tool to support the design activity . The tool is used to enforce the best design practice . This user type may not create applications *ex-novo*; rather, the applications are derived from pre-existing models, by personalizing them with specific contents and visual layouts. This type of user includes the students of the Hypermedia Modeling class.

### 4.2 Conceptual Requirements

As discussed by Nanard and Nanard in [9], the design of a complex application is a sophisticated "tango-like" process that does not always proceed in a predefined linear fashion. Thus a design tool should support a flexible design process, which doesn't bind the mental processes of the expert designers to a fixed procedural schema. On the other hand, keeping in mind the needs of inexperienced designers, the tool should enforce a (possibly minimum) number of priority constraints. These constraints should define which design constituents must be specified before others. Constraints are useful to guide the design activity, especially for novice designers, helping them to master the complexity of organizing the different facets of design (***Design Process Flexibility***).

It is important that a design tool can easily be updated to reflect the latest version of the model (***Model Adaptability***).

One of the best strategies to efficiently produce high quality design components is to re-use previously developed building blocks [10]. Re-using previous pieces of design is effective both for teaching purposes and for real-life applications. A design tool should therefore support the creation, inspection, and updating of a library of design projects, and the possibility of partially or totally re-using them (***Support for Design-by-Reuse***).

It is important that a design tool enables the integration of model-based definitions with additional (non formalized) specifications of all the relevant aspects, including those not completely supported by the model (***Model Integration***).

A design tool should be able to support different degrees of completeness, and to allow design specification to be refined at multiple levels ***(Completeness Adaptability).***

It is important that a design tool support the use of design patterns within the design process and the extension of the tool with new patterns as they become available (***Support for Design Patterns***).

We use the term *application framework* to denote a family of applications (e.g., catalogues for fashion houses) from which specific applications (e.g. a specific catalogue for a particular fashion house) can be derived. A number of operations (filtering, specializing, cut-and-paste, etc.) are needed to support an optimal use of application frameworks. A design tool must facilitate their development and re-use (***Support for Application Frameworks***).

A design tool should support the creation of design documentation and the production of well-organized design reports at each step of the design process (***Documentation Support***).

## 5. MODE: JWEB MODEL EDITOR

MODE enables designers to define all the different design features of an application as specified by HDM2000. Currently, MODE supports the creation of the following models: Requirements Elicitation, Information Model and Navigation Model.

MODE is based on Microsoft Visio 2000 [11]. Figure 2 shows the working environment. Windows are arranged in a friendly way for a Microsoft user. For example in Figure 2 the design of the Information Model is shown. The main work area is in the center of the window and it is composed of two pages: the first describing the in-the-large application schema, and the second describing the Access Structures (i.e. collections of objects stored in a hyperbase). As long as an object is defined in-the-large, it requires a detailed description ("in-the-small") on a separate sheet. At the top, where the different available toolbars are arranged, we find the standard and formatting tools (i.e. fonts, design, alignment, etc.).

On the left, 'Stencils' groups together the graphical HDM2000 primitives called 'shapes'.

Thus the user will build diagrams simply by dragging and dropping the available shapes and stencils into the work area. Then the designer can define and modify element properties using particular forms which guide him during the operations.

The designer can work simultaneously at different levels and on different parts of the design, in accordance with HDM2000 methodology, with two main restrictions:

- before using an element X in the definition of element Y, X must be defined (i.e. it's impossible to define a Component in an undeclared Entity Type or define a Collection of Painters if an Entity "Painter" is not created.
- There must be consistency among the different definitions.

*Figure 2: MODE: User Interface*

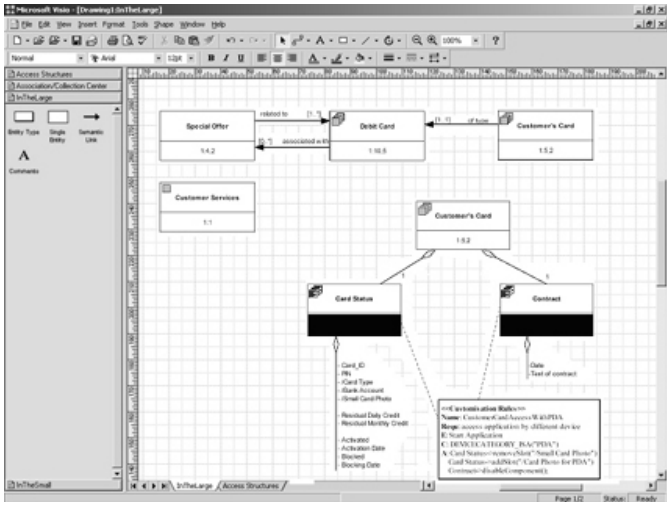
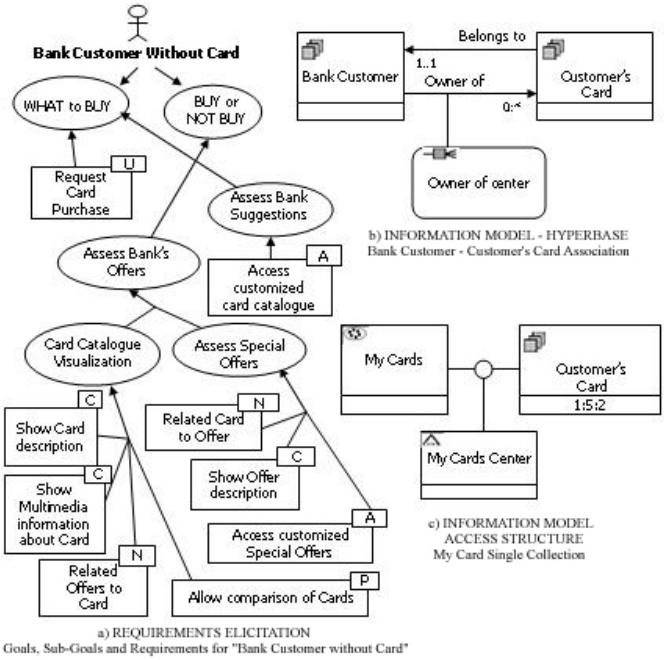


*Figure 3: UWA Project - Application for Banca 121: W2000 Methodology for the "Bank Customer" Stakeholder*




All information on the created models is stored in a relational database. Because of this, the tool allows the designer to summarize the information according to various criteria. Thus we can find out how many collections an Entity belongs to, or which requirements are satisfied by an element in the Requirements Elicitation. As soon as a change occurs, the resulting consequences for the entire project are known.

## 6. EMPIRICAL VALIDATION

In this section we describe the results obtained by applying W2000 methodology to an application for the sale of credit cards by "Banca 121" as part of the UWA project. The application was selected for the following reasons:

- we are dealing here with three quite distinct types of application user (**Multi-user**).
- the application needs to be accessible from more than one type of device so as to maximize the facility of access for all user types (**Multi-Device**).
- this application satisfies the criterion of ubiquity because in many ways it is important for the user to be able to access the system in a variety of situations (**Ubiquity**).
- some of the users of the application, generally bank customers, have a profile which allows them to personalize the content of the application and the functions available to them (**Profiling**).

In Figure 3 we present some schemas which were created in the different phases of the design process, to better demonstrate the importance of having a tool to support the designer when he works with complex web applications.

The stakeholders we identified are: Generic User, Bank Customer without Card, BANCA 121 strategy manager, Bank Customer with Card, Salesman, and Product Manager. The Bank Customer without Card is already a customer of the bank but has not yet obtained his first card. Figure 3 shows the stake holder's goals, detailed and subdivided into subgoals.

In this case two possibilities must be taken into account: The user already has in mind what to purchase, because he has made the decision based on information from some other source; or, the user does not yet have a clear idea of what he requires and above all of what the bank has to offer. These two potential user situations are modeled respectively on the high-level goals *What to Buy* and *Buy or not Buy*. In the former case the user may wish to briefly review the characteristics of the card which he intends to purchase, or directly carry out the request for purchase of the card (*Request Card Purchase*). While viewing the available cards, the user may wish to see the general catalogue of the bank or receive suggestions from the bank on the basis of their customer profile, established when their contract was signed. These two potential user situations are modeled on the goals *Assess Bank's Offers* and *Access Bank's Suggestions*. The bank's suggestions require a specific personalized access structure based on the user's profile (*Access Customized Card Catalogue*), whereas if the user decides to view the bank's products in general, then he may wish to see only current offers or the general catalogue of the bank.

The Information Model for the Bank Customer is shown in Figure 3.b and 3.c. As can be seen, a bank user has access not only to the catalogue of cards and special offers but also to the information relating to his own cards, and service information. The "Customer's Card" entity type contains information relating to a card held by a client: the contract and information on the current state of the card (if active or suspended, the residual monthly and daily credit, etc.). The Bank Customer – Customer's Card association links the customer's cards to the relative bank accounts with which they are associated. Thus a bank account can have a number of active debit cards, whereas each card is linked to just one bank account. The "My Cards" collection contains all cards belonging to the current user.

Firstly, the hypermedia design of the application was performed by a class of ten students without using CASE tools and then using MODE. The comparison of the result and the experience allows us to make these considerations:

- The lack of a deliberately planned editing element, able to facilitate the creation of diagrams, involves a considerable cost in terms of time for the designers. In particular, it was noted that in many cases the step from information to navigation, and thus to publishing, can be partly automated; often more time is wasted using inadequate editing instruments than in the thorough preparation of the application.
- The drawing-up of the project's documentation can be laborious in terms of time because all the diagrams created with another program have to be imported into the document. We estimate that about 25% of the time was spent on resolving problems linked to the insertion of diagrams into the design document.
- The introduction of customization rules increases the complexity of the diagrams, with a consequent increase in the difficulty of interpreting them. This general observation has led us to conclude that the rules must be used with moderation and in general there should be one, or at most two rules per diagram; beyond this limit, legibility worsens considerably. In addition, not all the rules have the same impact on legibility. We decided to use the rules only when they required the addition or the elimination of slots, components, or associations.
- On the whole we noticed a certain redundancy in the diagrams, which translates into a waste of design resources. A greater flexibility in the methodology would probably be a good thing, so as to allow different modeling of the aspects of the application depending on their complexity.

## 7. CONCLUSION AND FUTURE WORK

In our opinion, MODE can be used very effectively as it is, for both introductory and advanced training of design, and (integrated with JWEB) fast prototyping. A trial version of MODE is available and can be requested from mario.bochicchio@unile.it.

The next steps in the development of MODE are:

- Verification with other small to medium companies like Banca121;
- Adding help facilities to explain modeling concepts or to suggest design examples in a context-dependent way.
- Creation of semi-automatic documentation facilities.

## REFERENCES

[1] Johnson, J., & Henderson, A. (2002). Conceptual Models: Begin by Designing what to Design. **Interactions** (IEEE)

[2] Garzotto, F., Paolini, P., & Schwabe, D (1993). HDM - A Model-Based Approach to Hypertext Application Design. **TOIS 11**(1), 1-26

[3] Baresi, L., Garzotto, F., & Paolini, P. (2000). From Web Sites to Web Applications: New Issues for Conceptual Modeling. ***I*n Proceedings WWW Conceptual Modeling Conference,** Salt Lake City.

[4] Bochicchio, M., Paiano, R., & Paolini, P. (1999). JWEB: An Innovative Architecture for Web Applications. **ICSC 1999**, 453-460

[5] Fraternali P, Ceri S, Bongio A., "Web Modeling Language (WebML): a modeling language for designing Web sites", **Proceedings of the International Conference WWW9,** Springer Verlag LNCS, May 2000

[6] Mecca, G., Atzeni, P., Masci, A., Sindoni, G., And Merialdo, P. 1998. The Araneus Webbased management system. **SIGMOD** Rec. 27, 2, 544–546.

[7] Isakowitz T., Stohr E.A., Balasubramanian, "RMM: A Methodology for Structured Hypermedia Design", **Communications of ACM, 38**(8), Aug.1995, pp. 33-44.

[8] Schwabe D., Rossi G., "An Object Oriented Approach to Web-Based Application Design", **Theory and Practice of Object Systems, 4**(4), J. Wiley, 1998

[9] Rumbaugh J., Jacobson I.,Booch G., **The Unified Modeling Language Reference Manual**, Addison Wesley Longaman, 1999

[10] Gellersen H., Wicke R., Gaedke M., "WebComposition: An Object-Oriented Support System for the Web Engineering Life Cycle", in **Electronic Proceedings of the 6th WWW Conference**, Santa Clara, USA 1997.

[11] www.microsoft.com