

# From Web Sites to Web Applications: New Issues for Conceptual Modeling

Luciano Baresi, Franca Garzotto, and Paolo Paolini

Dipartimento di Elettronica e Informazione - Politecnico di Milano  
Piazza Leonardo da Vinci 32, 20133 Milano, Italy  
{baresi, garzotto, paolini}@elet.polimi.it  
Phone: +39-02-2399 3638; fax: +39-02-2399 3411

**Abstract** E-commerce, web-based booking systems, and on-line auction systems are only a few examples that demonstrate how WWW sites are evolving from hypermedia information repositories to hypermedia distributed applications, hereafter web applications. They blend navigation and browsing capabilities, common features of hypermedia, with “classical” operations (or transactions), common features of traditional information systems. This new coupling, between hypermedia and operational features, raises a number of novel *design issues*.

In this paper, we focus upon conceptual modeling for web applications, trying to identify its scope and its main issues. We claim that conceptual modeling for web applications is not just the union of two activities performed in isolation: modeling the operations and modeling the hypermedia. Rather, the *integration of the two facets of design* (hypermedia and operations) is the issue. The co-existence of operational and navigational aspects poses, among others, the following new problems to designers: How do information structures and navigation support operations? How do operations affect information structures and navigation? How do operations and navigation interfere? How are user tasks related to both navigation and operations?

The above issues, together with other relevant ones, are discussed in the paper, providing a contribution toward possible solutions, based upon the overall framework W2000.

## 1. Introduction

Web sites, at the beginning, were essentially read-only hypermedia repositories of information. Applications running today on the web (hereafter web applications) introduce a number of innovative aspects ([7]).

For the purpose of this paper there are two relevant “new” features to be considered. First of all, the expanded role of “operations” of different nature, with respect to traditional hypermedia operations (such as *browsing* and *navigation*): *data entry*, *business transactions*, *communication* with other remote users, etc. Secondly, we have the *tight integration* (or *interference*) among the different operation paradigms.

Let us consider, for example, a “typical” e-commerce web site for buying music. Users, while navigate within the catalog of records, can filter the products of interest by

formulating a query, can send a message to the virtual shop to transmit a specific request or can send a complain. Also, they can bookmark some products and include them in the “shopping bag”; they can inspect the content of the bag, by navigating from the bag to selected records; they can exclude some of the products previously chosen, evaluate the total cost of the final list, and, after providing additional information, complete the buying transaction.

The coexistence of different interaction styles, of hypermedia features with application operations, of complex information structures with transactions operating upon them, pose new problems to designers that require a new design paradigm. This new design paradigm can be approached in two different ways:

- A web application is regarded as an extension to the notion of a traditional information system, taking into account the need to incorporate navigation and complex information structures;
- A web application is regarded as an extension to a traditional web site, taking into account the need for incorporating various kinds of application operations.

The viewpoint of this paper is that the conceptual modeling of a web application is not just the union of two tasks performed in isolation: modeling typical hypermedia aspects and modeling operations. It sums up the complexity of the two activities with the additional task of modeling their coupling. As such, conceptual modeling of web applications cannot simply pile up existing techniques of hypermedia design (borrowed from the hypermedia/web communities) with methods and notations for “traditional” operation modeling (borrowed from the information systems or software engineering communities). The crucial point is to integrate and extend models, design methodologies and techniques, in order to meet the new design challenge.

The rest of this paper will discuss the above issues and will sketch some methodological solutions. Section 2 defines requirements for modeling web applications. Section 3 discusses how to add operations to hypermedia structures. Section 4 describes how to model the dynamics of hypermedia structures. Section 5 discusses how the behavior of web operations should be modeled. Section 6 draws the conclusions.

## 2. Requirements for Modeling Web Applications

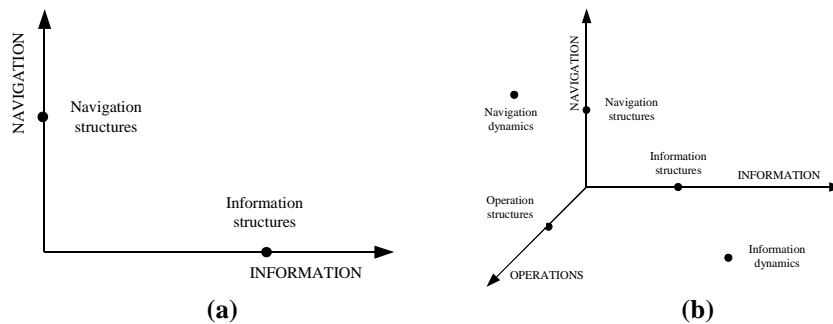
Traditional web sites can be regarded as (universally distributed) hypermedia applications, and can be largely modeled using hypermedia modeling methods. Most approaches [1, 3, 6, 11, 12, 15, 17, 16] distinguish at least two main dimensions for hypermedia/web site conceptual modeling: information modeling and navigation modeling<sup>1</sup>. Information modeling describes the contents of the web site. Navigation modeling describes its navigation capabilities, i.e., the paths that users can traverse, to explore the information universe. In the rest of this paper, the union of information modeling and navigation modeling will be globally referred to as *hypermedia modeling* or *web modeling*, indifferently.

---

<sup>1</sup> Various approaches explicitly include also presentation modeling, which we omit to discuss here since it is less affected by the introduction of operations.

Traditional hypermedia modeling focuses upon organization of information structures and navigation paths. Explicit dynamic aspects are omitted, for two main reasons. Firstly, no “reading” operation needs to be modeled explicitly: the only “operation” available to the user - “follow link” - has a built-in behavior, which can be left implicit. Secondly, information and navigation structures are not expected to evolve at run time. Therefore operations that explicitly manipulate those structures are not described.

Web applications, instead, introduce a new dimension that must be modeled explicitly, that is, non-navigational *operations*, and add to all dimensions (information, navigation, and operations) a new design aspect: the *dynamics*.



**Figure 1: Design Spaces for Web Sites and Web Applications**

Graphically, we can represent the design space for web sites as a 2D space, as shown in Figure 1(a): web site modeling includes only modeling information and navigation *structures*. Figure 1(b) describe the 3D modeling space for web applications and points out what must be “added” to traditional hypermedia modeling: i) Operations must be modeled explicitly; ii) Operations may “refer to” both information and navigation structures; iii) Not only operations but also information and navigation have dynamic properties.

### 3. Adding Operations to Hypermedia Structures

To discuss more precisely the role of operations within hypermedia structures, we need to introduce some basic notions and terminology of hypermedia modeling. Although we use as basis the model HDM2000, the latest version of the hypertext design model HDM [13, 14], most of our considerations are absolutely general, i.e., independent from the specific model being used. For the purpose of our discussion, the only concepts that we need to introduce are: *hyperbase* and *access layers*, *entity*, *component*, *semantic connection*, and *collection*.

The Hyperbase layer contains the base information structures of the web application with the associated navigation paths. It is described by:

- i) A set of entities - structured segments of information representing domain objects.  
An entity can be structured in different “parts”, called components. Components

are usually (but not necessarily) materialized as web pages interconnected by (structural) links. Entities are, in general, instances of entity types.

- ii) A set of semantic association types, whose instances are materialized as “semantic” links among pages of different entities.

The access layer has the purpose of facilitating the comprehension and accessibility of the hyperbase. It is made of *collections*, i.e., containers (of entities, components, or other collections), that provide “superimposed information” ([10]) and alternative groupings for the hyperbase elements. Assuming that there are, in the hyperbase, several instances of the entity type “Paper”, the “Accepted Papers” is an example of collection<sup>2</sup>.

We can recognize two main categories for the operations commonly found in web applications: *user operations* and *system operations*. User operations are the only functions directly visible to users. The function “include product”, for example, which allows users to put a select item in the shopping bag, is a typical user operation. Another example is a filter operation, which allows users to specify some selection criteria and to retrieve a set of elements of interest.

System operations are not visible to users, but are triggered by user operations or by navigation actions, or by maintenance (administrator) operations. The following are examples of system operations: the function that, triggered by “include product”, updates the list of items visible in the shopping bag; the function that registers, for profile tracing, all links traversed by a given user; the query function, that “implements” user filtering operations.

It is clear from the examples that system operations are at a lower abstraction level with respect to user operations. They are needed to specify the semantics of user operations, but are not “perceivable” to users.

Our approach is that only user operations need to be modeled during conceptual design. System operations can be modeled in a different phase (possibly, the detailed design, which is oriented toward implementation), while specifying the operation behavior.

Focusing upon user operations, there is a first design issue, innovative for traditional hypermedia design: where (i.e., to which information and navigation structures) are user operations made available? User operations can be “attached to” information “objects” at different levels of granularity and complexity. In the HDM terminology, they can be modeled as properties of entities, entity components, semantic associations, or collections. The intended meaning is that if an “object” is structured, the operation applies to all its constituents, and can be invoked from any of them. A virtual museum application, for example, may have a “create tour” facility, which allows users to collect the art works of their interest, and to create their own collection of pieces across which they can navigate linearly. If an art work is composed of several “components” (pages) each one discussing a different aspect of the art work, the designer may offer the possibility of including in the tour either a specific component (page) only, or the entire entity (i.e., all pages that represent an art work) in one shot. The two

---

<sup>2</sup> In a complete design exercise, the criterion to select the members of the collection, the topology (internal organization of the collection) the navigation features and a number of related information should also be specified.

design choices correspond, in the first case, to assign the “include tour” operation as property of a specific component only (e.g., the art work introduction), and, in the second case, to the entire entity.

In an e-commerce application, a useful operation is “convert to currency X”, which allow users to visualize the product cost from the current currency to another one. This could be usefully invoked from any product page, but also from any collection of products (determining a change of currency in all products that are members of the current collection).

Finally, operations can be also attached to navigation structures, i.e., links. Consider for example a kid-fashion web application, in which different product lines are described, each one linked to a number of products (shirts, trousers, dresses, etc.). When the user traverse the link from a product line to related products, (s)he may need to filter them according to different criteria, e.g., kid age, color, use. In this case, the filter operation should be modeled as a property of the link rather than of the "product line" page.

To be able to model “hypermedia with operations”, in all the various aspects, we have defined the framework W2000, which extends HDM2000 with UML [5, 8, 9]. The advantage of UML is to provide a well known, standardized, and customizable graphical notation, supported by a number of existing case tools. Lack of space prevents us from discussing the concepts and notations: details can be found in [2], where the framework W2000 is explained at a larger extent.

#### 4. Modeling Dynamics of Hypermedia Structures

Differently from traditional web sites, hypermedia structures of web applications are very often dynamic, in the sense that information and navigation objects evolve along the time, or by (direct or indirect) effect of user operations.

Consider, for example, a conference manager system, i.e., a web application that must advertise a scientific conference, enable authors to submit papers (in two steps: abstracts and full papers), support program committee members in reviewing and selecting papers, and help the chair setting up the conference program. An obvious entity type in this application is “Paper”. When an author executes the operation “submit abstract” (from the “how to submit” entity), an instance of Paper is created but only the "Abstract" component exists. Once the deadline for submission expires, the hyperbase must evolve: each paper has a new (optional)<sup>3</sup> component of type "Submission", which is instantiated when authors invoke the “submit full paper” operation. When the review process is over, i.e., the program committee has finally executed all the needed ranking operations, the structure of the paper has another evolution step: a new component type “Camera Ready” is added, and replaces the “Submission”<sup>4</sup>.

Evolution may involve also collections. The collection “all reviewed papers”, for example, contains each paper currently in “reviewed” state. It is replaced by the collec-

---

<sup>3</sup> The “Submission” component is optional since authors may submit abstracts and renounce to submit full papers.

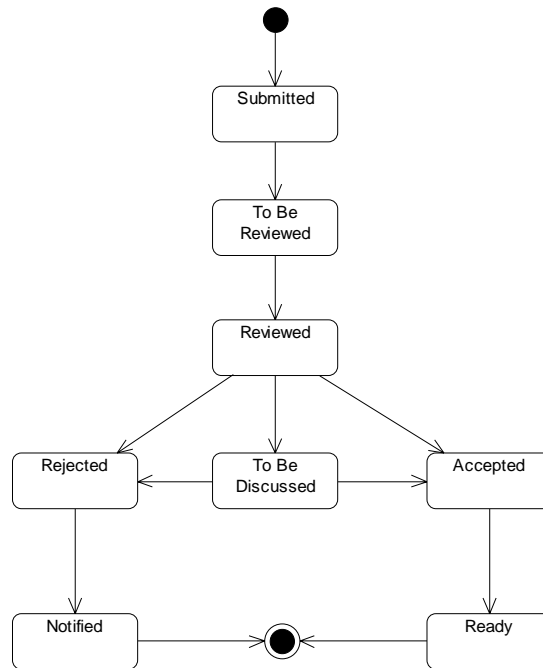
<sup>4</sup> We assume that the component “Abstract” always remains available.

tions “definitively rejected papers” and “definitively accepted papers” only after the program committee has finally discussed each paper, which is either in the state “rejected” or “accepted”.

Similarly, also semantic associations may correspond to states: the semantic association type “presented-in” (which connects each accepted paper to the session in which it is presented) exists only for “accepted” papers.

Finally, evolution defines also whether an operation can be invoked or not (and sometime the way the operation is executed, as discussed in the following section). Consider for example the operation “include paper in session”, which is assigned to each session entity and allows the program chair to update the description of the papers to be presented in a session and to link the session page to its papers. It becomes available only when all papers are in “accepted” or “rejected” state.

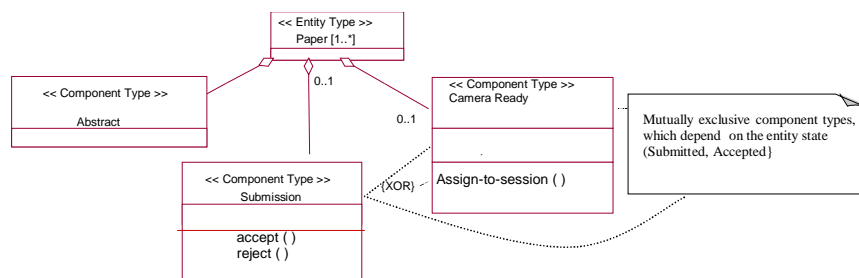
Modeling the dynamics of hypermedia structures is a novel task with respect to traditional web design, and can be expressed with various means. The direction we have explored is to model the states along which the various types of hypermedia objects (entities, associations, collections) can evolve by borrowing *statechart diagrams* from traditional object-oriented technology. Consider for example the UML statechart diagram for entity type Paper shown in Figure 2.



**Figure 2: Statechart Diagram for entity type Paper**

Considering the notion of state (of the different hypermedia objects) the following activities are needed by designers:

- a) To elaborate a state diagram for each (information or navigation object) subject to evolution;
- b) To decide which states are “perceivable” by end users,
- c) To correlate the possible different structures to the different possible states, by enriching the hypermedia schema with formal or informal “meta-properties” (comments, cardinality labels, or OCL constraints in UML). An example is shown in Figure 3, which describes the structure (in-the-large, omitting the details about data attributes) of entity type paper.



**Figure 3: The Structure in-the-large of Entity Type *Paper***

## 5. Modeling Operation Dynamics

Designing web operations is something different from designing traditional operations, for a number of reasons:

- Web applications are not software systems built from scratch, but are designed on top of a hypermedia schema. In particular, operations do not operate on generic data structures, but on peculiar structures that capture hypermedia specific abstractions.
- Most operations for web applications are something different from operations in conventional software: they can be invoked during navigation (and may need to take into account the navigation context as an implicit parameter), and they may require some navigation steps to be completed.

These peculiarities impose designers to rethink traditional operation modeling, to define a suitable means that addresses the characteristics of web operations and exploits built-in hypermedia concepts. Traditional object-oriented modeling specifies a user operation as a set of cooperating objects that exchange information and require services each other. Every involved object knows directly, through references or pointers, all objects it can interact with. Only the notion of “object” (with data and functional attributes) is built-in in the data model needed for specifying operations. Everything else must be designed explicitly.

In contrast, web operations work on a variety of “objects”, of different complexity, as we discussed in Section 3: elementary objects (single “pages” or “components” in HDM, very much similar to conventional objects) and clusters of objects (entities and collections, in HDM).

Compared to traditional object-oriented design, the “data model” exploited by web operations should deal explicitly with these hypermedia complex objects: entities, associations (links) and collections. In web applications, objects are connected by means of associations among them and within their constituents. Associations should not be considered (and designed) as static properties of objects. In fact, hypermedia objects do not know directly all objects they can interact with. As a consequence, associations cannot be modeled neither as simple references (pointers) among objects, nor as parts of objects, since they are not automatically created when the element they belong to is created. They can be created, modified, and deleted explicitly as if they were objects, and must become first-order elements in the data model used to specify operations. Treating links as first class objects has been traditionally accepted by the hypermedia community, sometimes accepted by the database community, and not fully acknowledged by the software engineering community.

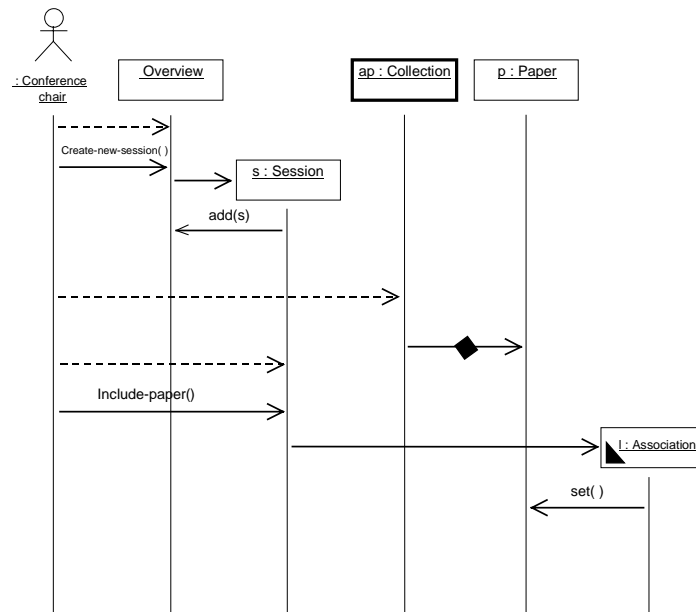
In traditional object oriented design, any operation is designed through explicit methods (and possibly special-purpose objects) and any interaction among objects through methods invocation and message passing. Only methods for creating and deleting objects are built-in. Navigation, if needed, must be explicitly designed. In web applications, which intensively involve navigation, the designer should not be forced to re-define each time the “follow a link ” operation: basic navigation, with the user “moving” from one piece of information to another, should be a built-in mechanism, complementary to method invocation.

The designer, for example, may need to specify that an operation can be invoked from an object of a given type (entity, collection, or association) no matter how the user has reached that object during a session. From the above consideration it follows that the modeling notation must support explicitly a specialized mechanism of message passing: *free navigation*, that is, the possibility of showing only the final destination of some navigation steps without specifying the followed path. Free navigation is not strictly necessary, but helps supplying a usable specification means. It is a shortcut that makes lighter and simpler the process of identifying fixed points that user must reach to complete the execution of a given operation, or to invoke it.

An opposite situation is when an operation can be invoked only in a given navigational context, or it requires, during its execution, that the user reaches a given object, but only under some constraints. Consider for example the operation “include-paper” that can be invoked in a web based conference manager from an entity of type “session”. Clearly, “include-paper” requires that the paper to be included in a conference session is identified, and is among those “definitively accepted”. A possibility (among several others) is to require that the user navigate to the collection “definitively accepted papers”, and chooses the paper from this context. The above design choice can be described by means of another specialized mechanism of message passing: *constrained navigation*, that allows the designer to specify a specific path as a precondition for invoking a given operation, or for completing it.

All peculiar requirements of web operations described so far can easily be met by extending UML *interaction diagrams*. For example, Figure 4 presents an extended UML sequence diagram that shows what a conference chair should do to create a new conference session and associate a paper with it in a hypothetical conference manager. The following are the main extensions to UML:

- To distinguish among different categories of objects - entities, associations, and collections, rendered with different shapes. Different object types also imply different rules for message passing, but we have no space to discuss them here.
- To refine the UML primitive for general object oriented message passing with constrained and free navigation. Constrained navigation is depicted using a line with a diamond, while free navigation is rendered with a dashed line.



**Figure 4: An Example of Extended UML Sequence Diagram**

The simple example of Figure 4 specifies that the conference chair can freely navigate to the conference *overview* entity and ask for creating a new session. This user operation makes the system create a new empty component - session *s* - and adds it to entity *overview*. Then the chair navigates “freely” to the collection of accepted papers (*ap*) and follows a specific association (see constrained navigation arrow) to identify a particular paper *p* in the collection. After browsing back to the session *s*, (s)he invokes the *includes-paper* operation, which creates a new association (*I*) and links *s* to the selected paper *p*.

A final concern is on the detail level that should be achieved. As discussed in Section 3, operations can be organized in: *user operations* and *system operations*. User operations are the only operations directly visible to users, thus callable by external actors. System operations are not visible to users, but are system operations necessary to and

triggered by user operations. Among system operations, we can distinguish between *base operations* and *application-specific operations*. The first set comprises all general-purpose functions (e.g., the operations to add an entity to a collection, or to add a component to an entity) that should be considered as built-in and can be left unspecified during the design process. The second set comprises system operations that are application specific and must be defined by designers (e.g., the operation to compute V.A.T. for a product once the product has been definitively bought).

In conceptual modeling, it is important to describe the application from a user-oriented perspective, describing not only information and navigation structures, but also operations, as perceived by end users, rather than for implementation purposes. Therefore the detailed description of system operations can be considered part of the implementation design, rather than as part of the conceptual design.

An additional problem with operations and hypermedia is the introduction of the notion of transactions. There are two main problems (as long as several ones more traditional): how to implement “atomic transactions” in an inherently volatile and “unstable” environment as a web application; and, above all, how to define “long lived” transactions, and which properties they should have. The second aspect is of particular relevance for web application: a browsing/navigation session may take minutes, hours or days. Within a session a user may perform a number of operations (e.g., reservations in a tourism applications) that at the end must coalesce in a transaction, in the sense that all of them must be “committed”. Lack of space prevents us from further discussing this interesting issue.

## 6. Conclusions

This paper discusses some modeling problems raised by the peculiar features of the new generation web applications, mostly induced by the need of capturing the complex intertwining between navigation and operations of different kinds. The paper proposes also some modeling concepts that could be introduced in current models – for object oriented, hypermedia or web sites design – and sketches how they can be rendered with suitable extensions to standard UML. The concepts and primitives described in this paper are part of W2000. W2000 extends HDM2000 and provides an UML-based unified modeling framework to support the various activities involved in the design process of web applications. Modeling is only a fraction of W2000: the framework (not yet completed) includes also modeling guidelines and heuristics to guide the designer, considers requirements analysis and the correlation between requirements and design choices, and considers usability evaluation and testing for web applications.

One of the interesting aspects to consider is consistency. Having to design several different aspects, using different concepts and primitives, it may be difficult to ensure the consistency among the different parts of the design effort. The advantage of adopting UML as basic notation is that UML-based CASE tools are easily available, and can support the specification activity and the needed consistency checks. For such a reason, in parallel to the conceptual definition of W2000, we are developing a set of

tools to support design in all its different aspects. This set of tools represents an evolution of the previous set of Hypermedia tools, Jweb [4], based upon the previous version of HDM.

An additional issue, that for lack of space we did not address in this paper, is *customization*. Web applications may address a potentially huge variety of different users with different navigational and functional needs or constraints. A generic user of a conference web application, for example, is not allowed to “see” the reviews of papers; authors could be allowed to access only a selection of the reviews, i.e., those concerning their papers; in addition, an author could not be entitled to use the link connecting a review with the reviewer. At a lower design level, specific information items could become not visible to certain roles under certain circumstances. Customization is assuming increasing relevance since complex web applications, such as portals, for example, are offered to a large community of users with different needs and tastes.

Customization is a complex task since several facets must be considered:

- Some customization choices depend upon the user profile and role.
- Some customization choices depend upon the wishes and the requests explicitly stated by the user.
- Some customization choices depend upon the overall state and evolution of the web site (e.g. the single reviews may be invisible to the authors until all the reviews have been completed).

In order to provide a global framework where customization could be addressed, we are working on refining the notion of *hyperview*, as a conceptual mechanism capable of supporting advanced customization.

## References

1. P. Atzeni, G. Mecca, and P. Merialdo: “Design and Maintenance of Data-Intensive Web Sites”, Proc. EDBT 1998, pp. 436-450.
2. L. Baresi, F. Garzotto, P. Paolini, and S. Valenti: “HDM2000: The HDM Hypertext Design Model Revisited”, Tech. Report, Politecnico di Milano, Jan. 2000.
3. H. Baumeister, N. Koch, and L. Mandel: “Towards a UML extension for hypermedia design”, in Proceedings of UML’99, LNCS 1723, Fort Collins, USA, October 1999, Springer Verlag.
4. M. Bocchichio, R. Paiano, and P. Paolini: “JWEB, An HDM Environment for Fast Development of Web Applications”, in Proceedings of the IEEE Multimedia Computing and Systems, 1999.
5. G. Booch, J. Rumbaugh, and I. Jacobson: “The Unified Modeling Language User Guide”, Addison Wesley, 1998.
6. S. Ceri, P. Fraternali, A. Bongio: “Web Modeling Language (WebML): a modeling language for designing Web sites”, to appear on Proc. Int. Conf. WWW9, Amsterdam, May 5 2000
7. P. P. Chen, D. W. Embley, and S. W. Liddle eds, Advances in Conceptual Modeling - Proc. WWCM099 -Int. Workshop on the World Wide Web and Conceptual Modeling, (Paris, Nov. 1999) Springer-Verlag, LNCS 1727.
8. J. Conallen: “Building Web Applications with UML”, Addison-Wesley, 2000.

9. J. Conallen: "Modeling Web Application Architectures with UML", *Communications of the ACM*, 42:10, Oct. 1999, pp. 63-70.
10. L. Delcambre, D. Maier, "Models for Superimposed Information", In [5]
11. O.M.F. De Troyer, C.J. Leune, "WSDM: a user centered design method for Web site", in *Proc. Of Int. Conf. WWW7*
12. P. Fraternali, P. Paolini: "A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications", *Proc. EDBT 1998*, pp. 421-435.
13. F. Garzotto, P. Paolini, D. Schwabe, "HDM - A Model-Based Approach to Hypertext Application Design", *TOIS* 11(1) (1993), pp.1-26
14. F. Garzotto, L. Mainetti, P. Paolini, "Navigation in Hypermedia Applications: Modeling and Semantics", in *Journal of Organizational Computing and Electronic Commerce*, 6 (3), 1996
15. T. Isakowitz, E. Stohr, P. Balasubramanian: "RMM: A Methodology for Structured Hypermedia Design", *CACM* (1995), 38(8), pp. 34-44.
16. G. Rossi, D. Schwabe, F. Lyardet "Web Application Models are More Than Conceptual Models", in [5]
17. D. Schwabe, G. Rossi, "An Object Oriented Approach to Web-Based Application Design", *Theory and Practice of Object Systems*, 4 (4), J. Wiley, 1998