Chapter XII

Publishing Model for Web Applications: A User-Centered Approach

Roberto Paiano
University of Lecce, Italy

Leonardo Mangia
University of Lecce, Italy

Vito Perrone
Politecnico di Milano, Italy

ABSTRACT

This chapter defines a publishing model for Web applications starting from the analysis of the most well-known modeling methodology, such as HDM, OOHDM, WebML, Conallen's method and others.

The analysis has been focused to verify the state of art about the modeling of Web application pages. In particular, the different types of elements that compose the Web page in the above models are taken into consideration.

This chapter describes the evolution of the HDM methodology starting from the first approach based on the definition of a LP concept up to the more structured and complex Conceptual page, based on the influence of "operations" on the modeling of the dynamics of navigation between pages.

INTRODUCTION

Design and development of WWW applications is quickly evolving to become more engineered products introducing powerful models of hypermedia applications. The entire lifecycle to obtain affordable outcomes must be considered as a complex process that should be supported by tools in order to help the designer in each phase.

Starting from a conceptual modeling makes it easier to manage the changes but it requires a well-engineered process to correctly drive the entire cycle from the model to the outcomes.

Our research activity is oriented to develop both a model to suit the complexity and a set of tools to support the designer from the analysis phase to a prototype of the Web application in order to have an effective test of model. These tools, based on a relational database, support also the multi-delivery feature to customize the application according to the user role (families of applications).

The main goal of this chapter is to define a publishing model for Web applications, starting from the analysis of the most well-known modeling methodology, such as HDM, OOHDM, WebML, Conallen's method and others.

The analysis has been focused to verify the state of art about the modeling of Web application *pages* and to capture the different types of elements that compose the Web page in the above models.

BACKGROUND

In 1993 the Hypermedia Design Model (HDM) (Garzotto, Paolini & Schwabe, 1993; Garzotto, Mainetti & Paolini, 1995, 1996) was published, the first modeling approach oriented to the design of multimedia application that was enhanced to support the hypermedia applications (Bochicchio, Paiano & Paolini, 1999).

In this environment a relevant aspect is represented by definition of Logic and Presentation pages. Logic pages have been introduced into the model to better design what the designer considers the unit of fruition of the specific WWW application (example: a painter and all his works, or the collection of all painters, and so on). The Presentation pages are a collection of logic pages that appear to the user into the HTML page (example: the home page and the previous logic page can appear in two frames of a unique HTML page), managing the dynamic behaviour.

In our research activity we developed the tools, starting from HDM concepts, to support the entire applications lifecycle through the prototyping using an engine that will be briefly described in the next section.

In the last months the Web environment has been oriented on the Web applications more than on Web sites, so the HDM model is evolving to its 2000

version (W2000) in order to best capture all the dynamic and navigational behaviour of WWW applications.

Adding the operation of the traditional Web sites means that the model structure and the behaviour may change dynamically and are strictly related to the user profile.

Our research is oriented to define an Information Conceptual Model and a Navigation Conceptual Model by rendering the design concept using customized UML diagrams. UML was extended to define a suitable framework for this task.

The next step is the definition of a Conceptual Publishing Model, which is the core of this chapter, which inherits the information and navigation definition and customizes the user behaviour.

Several authors are publishing interesting ideas about this problem, but they are starting from the page definition, while we start from a conceptual framework (Conallen, 1999; Ceri, Fraternali & Bongio, 2000; Schwabe & Rossi, 2000).

Although for small applications into a well-known application domain it is possible to directly design the Web page, having in mind, without any formalism, the information and navigation structure in order to realize a well-structured design it's needed to have a different approach to the design.

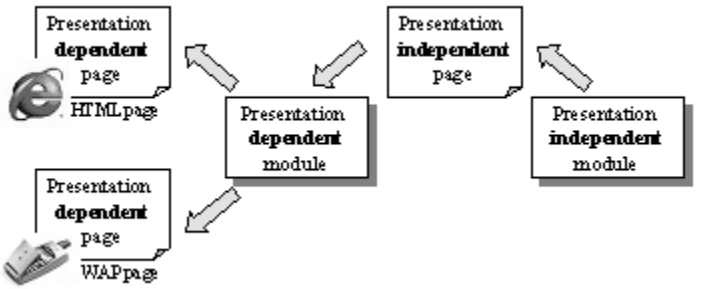
JWEB II NAVIGATION ENGINE: THE LOGIC PAGES APPROACH

The main requirements taken into account in the design and in the development of the JWeb II Navigation Engine (NE) are:

- Generation of hypermedia applications described in standard structures (easily maintainable).
- Management of different client devices with particular presentation/visualization technology (multi-delivering) such as the Internet, WAP, CD-Rom, etc.
- More different end-users, maintaining individual history for each session.

According to these requirements, the JWeb II NE design is structured in two different steps. In the first step, the part of the application independent from the client device presentation technology (presentation-independent module) is generated, while, in the next step, the other part that is dependent on this technology (presentation-dependent module) is built. So the generation of hypermedia applications consists of the creation of the physical pages (presentation-dependent) that will be published directly on the client device and LPs, (presentation-independent) that contain all the necessary information for the creation of the physical page (Figure 1).

Figure 1: Delivery Process



This distinction allows the complete reuse of the presentation-independent modules that represent the main design and development effort, while the presentation-dependent module must be edited according the particular presentation technology (WAP-WML, HTML, etc.).

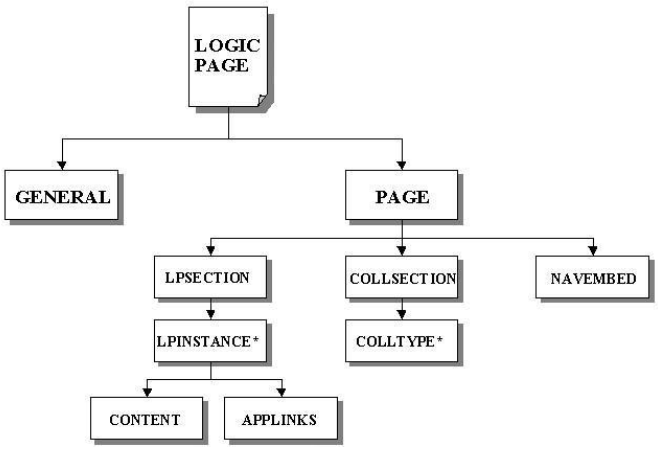
For example, in a Web-based hypermedia application the presentation-dependent module generates HTML physical pages compatible with the Web browser. These HTML pages are created from standard logical pages that contain the right contents. In particular, the logical pages contain both all the data stored in database structures and all the information maintained by the JWeb II NE to manage the correct user navigation and the user dynamic structures (ex. shopping bag). When changing the presentation technology (ex. from HTML to WML-wap) all the modules for the logical pages generation remain valid, while the mapping on HTML physical pages must be adapted to support the WAP technology.

Besides the creation of pages, both logic and physical, the JWeb II NE manages the exchange of messages with the client device. Particularly, the presentation-dependent modules receive string format commands from the client device, according to the presentation technology, and translate these commands into standard procedure calls for the presentation-independent module

In details the logical pages structured as in Figure 2, pick up and join information about:

- item (one or more) selected (ex. an artist),
- application link for the selected item (ex. the artist's works),
- active collection (actual showed element position, link to the next and previous elements, etc.),
- centres of other collection (ex. "Web sections" collection), and
- user navigation (back, user history, etc.).

Figure 2. Logic Page Structure



The LP contains the information in standard and abstract form, independent from the user device presentation technology. A logical page, as shown in Figure 2, is made up of some hierarchical sections that pick up in separated ways the information about selected elements (*LPSection*), collections (*CollSection*), and links for default navigation (*NavEmbed*).

In JWebII NE the LPs are XML files with a specific DTD; these LPs are built according a specific template and they pick up all the information needed to create the physical page respecting the end user request. The information for the logical pages is stored in a database with a structure independent from the specific application (named run-time database).

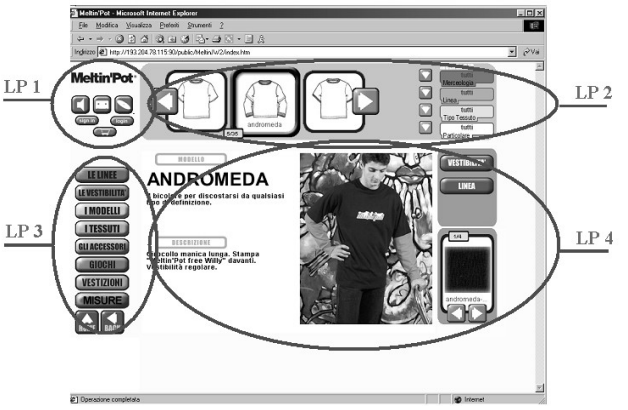
In the simple applications, there is almost a direct relationship between the logical page and information for the end user, while in the most complex applications it is possible you need to show the end user the information present in different logical pages.

Figure 3 shows a Web page of an e-commerce application that describes the information about a particular model of shirt, its relative application links, the active collection (all the models with available filters) and another collection (the Web application's sections).

This page has information present in four different LPs, respectively:

- LP1: information about support functionality
- LP2: information about active "models" collection
- LP3: information about "Web sections" collection
- LP4: information about selected "models" and their relative application links

Figure 3. The Logic Pages



In the LP2 and LP3, the furthermost information is in the section *COLLSection*; while in the LP4, the most further information is in the section *LPSection*.

The LPs (but precisely the relative physical pages) are composed together in presentation level without a precise design methodology. In the case of Figure 3 the composition is made using the HTML technique.

This LP approach to model Web pages was used to realize an e-commerce application in a European project (Bochicchio et al., 1999). This experience has underlined that the approach is efficient to model Web application for a run-time navigation engine (such as JWeb II NE), but some problems were undefined:

- the division of the final content in different logical pages is presentation-oriented and not a result of a publishing model,
- the definition of the interaction between logic pages in the same physical page, and
- what happen when the "operations" change the contents of a logic page.

W2000: THE CONCEPTUAL PAGES APPROACH

Motivation

On the basis of the former open issues on the logical pages approach, it is clear that this kind of modeling approach is quite effective for modeling pages to be used by the engine previously described. On the other hand, taking into account our experience in Web application modeling, we realized that it is ineffective for modeling pages from the user point of view, capturing the user experience already in the modeling phase. For this reason we introduced the conceptual pages modeling that focuses on modeling Web pages just as they will be perceived by the

users, introducing a user-centered semantic in defining the various aspects of the pages, modeling the interactions among pages and within the page. Based on this definition, the conceptual pages modeling should be located above the logical pages modeling (opportunely revisited) for those applications that will be executed by the JWeb engine.

The model aims to reach the following objectives:

- To define a self-standing model: to allow the designer to directly model the Web pages starting from the requirements.
- To define a model that could become part of a more complex methodology, such as HDM and its evolution, W2000 (Baresi, Garzotto & Paolini, 2001).

The first objective is due to the fact that, from the analysis of other modeling techniques, a methodology supporting the direct modeling (even from a user-centered point of view) of the Web pages does not still exist. On the other hand, such a methodology should be the most used for modeling small and well-known applications using a structured approach. In this scenario, it is very important to manage both the complexity of the structural composition of the pages (generally the pages join several different kinds of elements) and the navigation dynamics between pages. Furthermore, using a structured approach, it is possible to reuse both the model and the multimedia elements in a framework-like environment.

The second objective concerns the need to complete the model W2000 in order to have a complete methodology to design Web applications. Even though this model has been thought as a natural extension of the HDM model, it is rather general for being used to model the pages of an application designed using one of the other known methodologies like Ceri et al. (2000) and Schwabe and Rossi (2000). Generally, these methodologies split the overall application design into different phases concerning the different aspects of a Web application, that is, contents, navigational characteristics, pages and, only someone, the operations and transactions (UWA, http://www.uwaproject.org).

The W2000 model has two distinct levels:

- *Information Design:* where the *information structures* are designed. Information structures represent the contents available to the users of the application. The information is modeled from the user point of view, meaning that the designer design each object of interest for the user choosing only that information that could be of interest for a particular user. Moreover, in this design phase the *access structures* are designed. The access structure, named *collections*, provides the user with a way to explore the information contents of the application.

- *Navigation Design:* where the *navigation structures* are designed. Navigation structures re-organize contents for a navigational purpose. They define both the atomic piece of content that will be presented to the user, named *nodes*, and the way the user can pass from one piece of content to another; that is, the accessibility relationships form each node towards the other nodes. Moreover the navigation model in W2000 contains the notion of cluster, which is a way to organize the nodes for *context navigation* purposes. Each cluster represents a context of interest for a user, and the designers have to design both the contextual navigation, that is navigation within the cluster, and the infra-clusters navigation, that is navigation among the various clusters.

Like the above levels, the conceptual pages model deeply uses the notion of "type" for all its elements, allowing a more flexible page structure to better reuse components and contents. Furthermore, to satisfy the requirement to complete the W2000 model, some guidelines and rules are defined to map the Navigation level to the Page level. We have already said that a relevant aspect of the model is the possibility to model the "user experience." The way we made this possible was by the introduction of a primitive used to organize information and navigational features needs to the user to perform a certain task in a specific context. For example, some tasks could be browsing items in your shopping bag, scrolling through the pictures of a painter, exploring all book information, etc.

The Model

One of the most important aspects of a model is a precise definition of its focus. The model we are presenting in this chapter aims to reach a number of objectives that we can summarize as in the sequel:

- The main purpose of the model is the identification of the pages of the application. Due to the complexity of a Web page, we have to characterize what a Web page is.
- We model only the composition of a Web page in terms of its contents and its navigational capabilities.
- We separate the modeling of the contents from the modeling of the graphical and visualization aspects related to these contents. In this way we can define the meaning (even in terms of contents and navigational capabilities) of a Web page from a user- centered point of view independently from how it will be visualized and with which graphical solution. This way of designing should separate the designer contribution from that of the experts of graphic or visual communication.

On the other hand, we do not explicitly take into consideration temporal, spatial, and graphical aspects. For this reason both a text and a video are simple content units; events through text can be only read, while the video can be played, stopped, zoomed, etc.; that is, the user can have further interactions with it. Moreover, if a piece of content is delivered as simple HTML or as a more complex flash file, it does not matter because it is often a piece of context. We decided to neglect these aspects for having a lighter model and because of the richness of multimedia kinds of contents and the various way of rendering the same content using different graphical solutions. Anyway, we let the designer specify some temporal, spatial, and graphical aspects in an informal manner, using the comment property we added to each model primitive.

Based on the former considerations, our conceptual pages model describes the organization of information content of a hypermedia application into pages in terms of the following:

- it identifies the basic elements for presentation, i.e., publishing units,
- it organizes them into sections and pages, and
- it organizes the navigation within and across pages.

In the next sections the model primitives will be described in terms of their properties, a possible notation, and their usage. A number of examples will be shown so the reader can make himself familiar with the concepts we present and better understand both their meaning and their usage. Regarding the notation we adopted for the model, some considerations could be done. Once the semantic and properties of each primitive have been defined, whichever notation could be used, depending on who will use this notation for making the application design and who will be the reader of the diagrams and so forth. Moreover, using a graphical notation or not, which implies making sometimes an excessive number of pictures, can be influenced by the availability of a tool supporting the design phases. In our case we chose an UML-like notation that means a notation based on UML, conveniently customised and adapted. The reasons for this choice can be found in the spread of UML in this field. More precisely, we made an UML-profile, but its definition is not included in this chapter.

Publishing Unit Type, Single Publishing Unit

A publishing unit is the atomic element within the page structure. It is a set of information content shown to the user as presentation unit. All the contents of a publishing unit should be perceived by the user as a "consistent" portion of information (in a multimedia meaning) on the page.

Referring to a navigational model, a publishing unit can be defined using one of the following methods:

1. We derive the publishing unit from a whole node defined in the navigation model (according to the widespread Web application models, a node defines the *elementary granules* of information from /to which a user can navigate); the content of the node is "presented" to the user in one unit. This unit may inherit links and navigation features from the node from which it is derived.
2. We derive the publishing unit from a part of a node, defined in the navigation model; the content of the node can be "presented" to the user in more units organized in a *publishing cluster* (discussed later in this chapter). These units may inherit links and navigation features from the node from which they are derived, but they also may have *pure links* (discussed later in this chapter) that make the navigation across them possible.
3. The publishing unit does not derive from any node. In this case, it represents a special element of the page, such as site "logo" or copyright information.

The way we derive a publishing unit from a node or we pick up the information from a data base (or an other information model) is defined by a *mapping rule*.

A publishing unit is described by the following properties:

Name

It univocally identifies the publishing unit in the Publishing Model. In a publishing cluster and/or in a page section, different publishing units must have different names. If two publishing units have the same name, then the two units will be the same.

Graphics

It specifies the graphical properties of the publishing unit. A comment, and eventually sketches, can describe it.

Content

- It can specify the way through which the content of a publishing unit is derived from a node or picked up from a database or an information model (*mapping rule*)
- It can be a description of a pure publishing content, meaning a kind of content that does not derive from any information source, for example a logo of the society.
- Nothing. Some publishing units may not have content associated with them because they only act as link placeholders.

Based on the kind of information that the publishing unit models, we can have:

- **Publishing Unit Type:** it describes the common structure, properties, and features of a class of units; it is derived from a node type.
- **Single Publishing Unit:** it describes the structure, the properties, and the features of an individual unit. A single publishing unit is derived from a single node or it is not derived from any node; in the last case, it represents special content (logo, copyright information and so forth, or an index of high level collections).

In Figure 4 the publishing units contained in a page of our example site are shown.

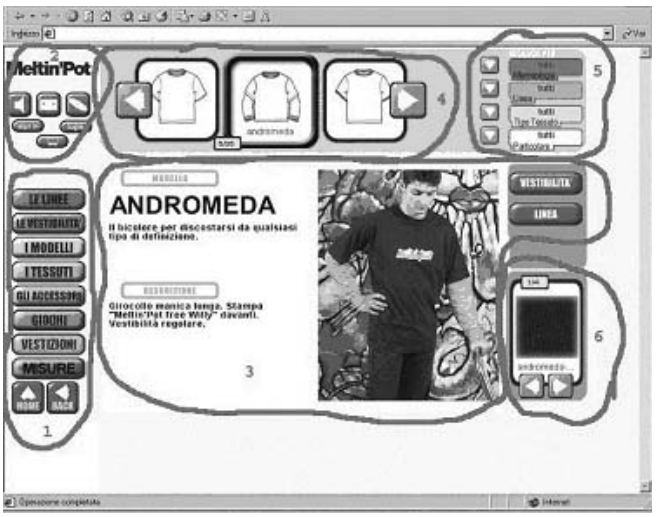
For specifying the properties of the publishing units in the design document, the refer to the graphical set in Figure 5.

For each publishing unit, a table, shown in Figure 6, is used for specifying all the properties.

Publishing Cluster Type, Single Publishing Cluster

The publishing cluster represents the way to put together the information and the navigational features performed by a user during a task. It groups publishing units and links. The relationship between a publishing cluster and its publishing units is an aggregation. The designer has to define navigation across the publishing units within

Figure 4. Publishing Units in Meltin Pot's Page



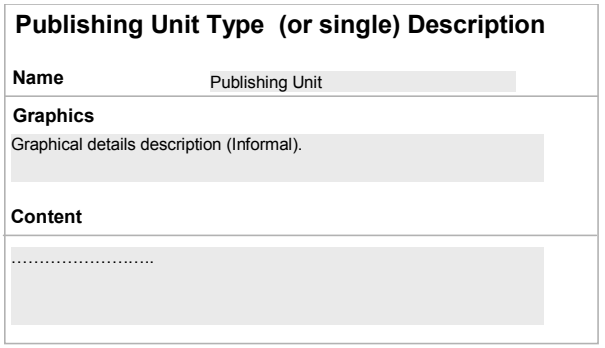
Note: 1,2 and 5 are single publishing units, while 3, 4 and 6 are publishing unit types.

Figure 5. Graphical Notation Used for Specifying the Single Publishing Unit and the Publishing Unit Type Primitives

the publishing cluster, and among publishing units belonging to different publishing clusters. Examples of publishing units belonging to the same publishing cluster could be a publishing unit representing a pictures index of a painter and a set of publishing units representing the pictures the shopping bag and its items, and so forth.

Through the publishing cluster, we perform the above-mentioned separation between contents and their visualisation aspects. For each publishing cluster of the section, we separately specify the contents, in terms of publishing units, and the possible visualisation configurations, in terms of subsets of the formerly defined publishing units. For understanding this key concept of the model, let us show an example. Consider the main section of the page of a product that includes only a publishing cluster (that can be omitted because of being the only one). Let us suppose that in this application the product is described by a general description and a technical description, and the designer decided to show these two parts of the overall product description one at a time. Conceptually we have only one page for a product with two publishing units (one for each product description) that is the *content*, but with two visualisation configurations — one with the publishing unit containing the general description, and the other one with the publishing unit containing the technical description. This second part represents the visualisation

Figure 6. Graphical Notation Used for Specifying the Properties of the Single Publishing Unit and Publishing Unit Type Primitives



aspects of a page. To realise the advantages of the separation of these aspects, let us take into consideration the same page with the same content, but with a different visualisation strategy. Let us suppose that the designer wants to limit the overall number of application pages — for example, for reducing the number of the overall navigational clicks by the user. A suitable solution fulfilling the designer's goal is to have only a visualisation configuration containing both of the publishing units. This simple example shows as it is possible to specify several visualisation solutions starting from the same page's contents.

A publishing cluster is descried by the following properties:

- **Name:** It univocally identifies the publishing cluster in the Publishing Model. It is usually the name of the navigation element from which the publishing cluster is derived. In a Publishing Model, different clusters must have different names. If two clusters have the same name, then the two clusters will be the same.
- **Content:** It is specified in terms of the publishing units belonging to the publishing cluster.
- **Visualisation Configurations:** They specify the possible visualisation configurations of the publishing cluster. Each visualisation configuration is composed of a subset of the publishing units defining the publishing cluster content. Optionally, an order can be assigned to the publishing units within a visualisation configuration to show the importance rank in visualising them. Moreover, a default visualisation configuration has to be defined.

Based on the kind of publishing units belonging to the publishing cluster, we can have:

- **Publishing Cluster Type:** it has at least a publishing unit type.
- **Single Publishing Cluster:** it has only single publishing units.

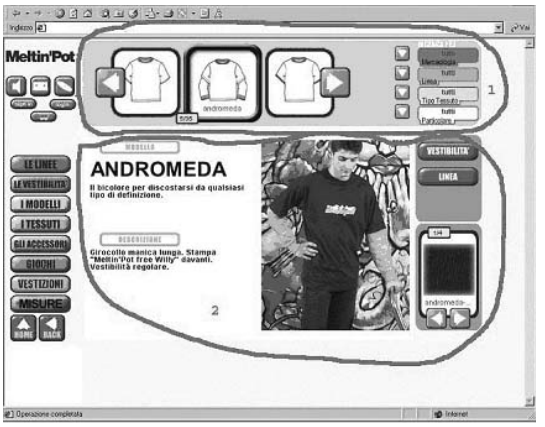
In Figure 7, two publishing cluster types are shown. The first publishing cluster corresponds to the user task "to explore the collection," while the second one corresponds to the user task "to look at the details of a particular article."

For specifying the properties of the publishing cluster in the design document, the graphical set in Figure 8 is associated to the section.

For specifying the content in terms of publishing units, we put two possibilities at the designer's disposal. He can use an aggregation relationship between the publishing cluster and its publishing units in a graphical way, or specify the publishing units in the associated table as shown in Figure 9. The same notation can be used for both publishing cluster types and single publishing clusters.

Any additional comments can be added based on the designer's preferences.

Figure 7. Publishing Clusters in Meltin Pot's Page



Section Type, Single Section

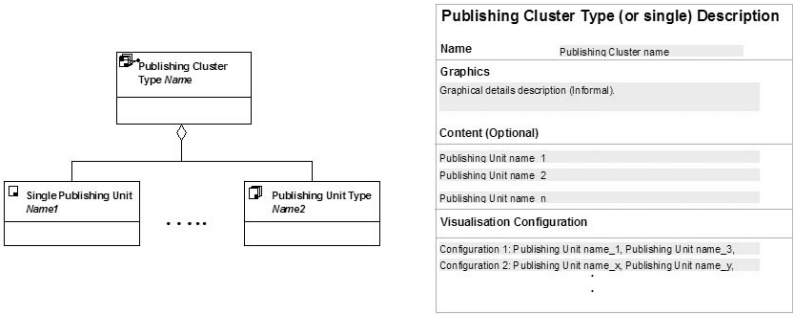
A section is a grouping of content, semantically correlated. It is defined in order to provide the pages of the application with a better organization. A section puts together publishing clusters that are semantically correlated. When a user goes from one cluster to another within the same section, he does not change his task, but he is changing context performing the same task.

A section is described by the following properties:

- **Name:** It univocally identifies the section in the overall publishing model of the application.
- **Graphics:** It specifies the graphical properties of the publishing section. It can be either described by a comment and eventually sketches or omitted because recursively defined by the graphical properties of the single publishing unit belonging to.
- **Content:** It specifies the content associated with the section. This content is expressed in terms of Publishing Cluster Types and/or Single Publishing Clusters.

Figure 8. Graphical Notation Used for Specifying the Publishing Cluster Type and the Single Publishing Cluster Primitives

Figure 9: *Graphical notation used for specifying the properties of the Single Publishing Cluster and Publishing Cluster Type primitives*



Based on the kind of publishing units belonging to the page section, we can have:

- **Page Section Type:** it has at least one publishing unit type.
- **Single Page Section:** it has only single publishing units.

In Figure 10, we show a page of our example application where there are three sections, two of which are Single Sections (Sections 1 and 2) while the third one, the main section, is a Section Type (Section 3). In this page each section has only one visualisation configuration, therefore section configurations do not have to be described.

For specifying the properties of the section in the design document, refer to the graphical set in Figure 11.

Figure 10. Page Sections in Meltin Pot's Page

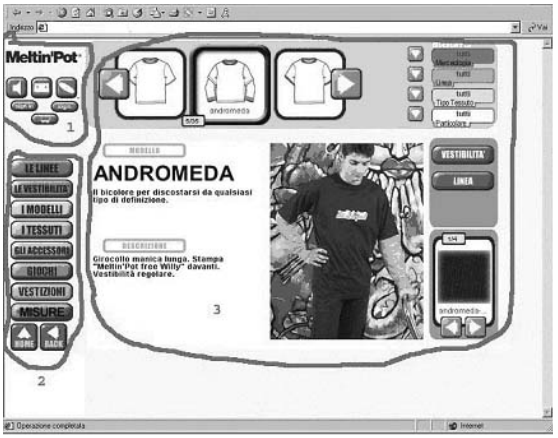
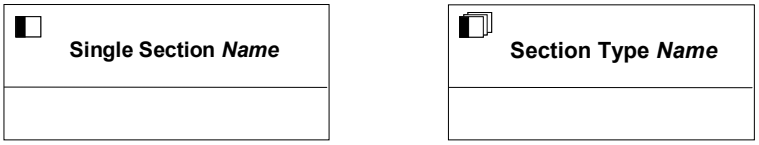


Figure 11. Graphical Notation Used for Specifying the Section Type and the Single Section Primitives



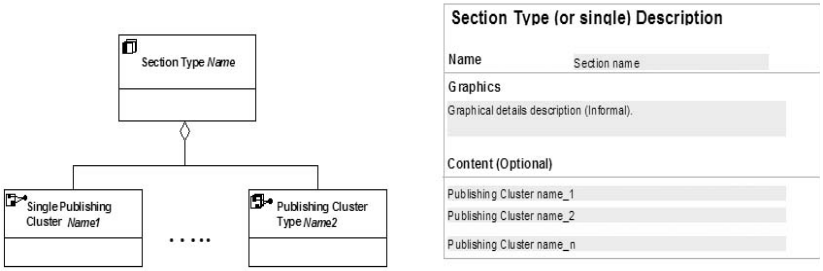
For specifying the content in terms of publishing clusters, we put two possibilities at the designer's disposal. He can use an aggregation relationship between the section and its publishing clusters in a graphical way or specify the publishing clusters in the associated table as shown in Figure 12. The same notation can be used for both Section Types and Single Sections. If the section contains only a Publishing Cluster, in the graphical representation, the publishing units of this cluster can be directly associated to the section.

Page Type, Single Page

A page represents what the user sees in a browser window. It is quite arduous defining precisely what a page is from the user's point of view. In particular, it could create confusion about what changing page means. To avoid possible confusion we use a semantic definition of a page. We assign a page to each semantic element of the application. We picked up these elements from the Information Model, meaning the following:

- **Entity:** is a "virtual object" of interest for the user. An entity, in general, makes more sense for the reader if it can be related to an object of the "real world." The entity "Raffaello," for example, is immediately understood as associated with a famous painter.
- **Semantic Association:** connects two different objects in the sense that it creates the "infrastructure" for a possible navigation path and it has a semantic

Figure 12. Graphical Notation Used for Specifying the Properties of the Single Section and Section Type Primitives



associated to it. The Semantic Association between Smith (an *Author*) and P121, P128 *(paper*s) has a semantic meaning (authorship) and also provides a (possible) navigation path.

- **Collection:** is the basic element for building an access schema to the application contents. A collection, very shortly, is an organized set of objects, called members. A collection is created in order to assemble all the objects that, under certain circumstances, can be interesting for the user.

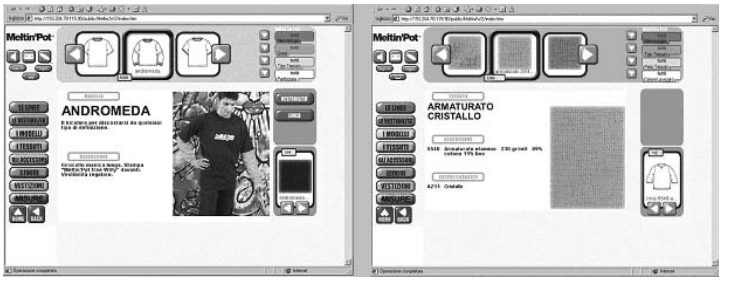
In Figure 13 we show two pages from our example application. The page on the left is for a dress article, while the other one is for the cloth choice.

The semantic definition of the Web page we have done appears independent from the device through which the application will be delivered. From the structural point of view, a page is a grouping of different sections whose contents could not be correlated semantically. The same sections may be used in more than one page. Generally, a page is composed by a main section dealing with the semantic element that gave rise to the page, and a number of secondary sections that allow the user to change the focus element. The different sections belonging to a page are very often completely unrelated from each other. Within a page, for example, may coexist a section showing a general menu (that never changes), a section showing specific objects, and a section listing auxiliary services. The three sections do not relate to each other, but for the fact that they are offered in the same page.

A page is described by the following properties:

- **Name:** It univocally identifies the page in the overall publishing model of the application.
- **Graphics:** It informally specifies the graphical properties of the page type. It can be either described by a comment or a sketch, or omitted because recursively defined by the graphical properties of the section types.
- **Content:** It specifies the content associated with the page. This content is expressed in terms of Publishing Section Types and/or Single Publishing Sections.

Figure 13. Page Examples from the Meltin Pot's Application



- **Main Topic Section:** It indicates the section containing the most important element of the page.

A page can be either a *Single Page* or a *Page Type.*

- **Page Type:** it represents a set of pages that share some common properties. It defines a "category" of pages that a user can find in the application associated to the same conceptual element type.
- **Single Page:** it represents a specific page of the application that does not share its property with any other pages. Typical examples of this kind of page are the Home page, the Presentation page, etc.

For specifying the properties of the Page in the design document, the graphical set in Figure 14 is used.

For specifying the content in terms of sections, we put two possibilities at the designer's disposal. He can use an aggregation relationship between the page and its sections in a graphical way or specify the sections in the associated table as shown in Figure 15. The same notation can be used for both page types and single pages. In the following figures both the generic notation and its usage for specifying the page of the dress article are shown for a better understanding of the notation usage.

For each page (both single page and page type), a table is associated in order to describe the rest of its properties.

Any additional comments can be added based on the designer's preferences.

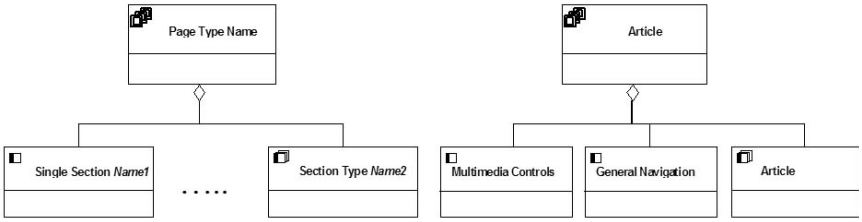
Depending on the designer's preferences, a layout sketch can be associated to the page. This sketch should express the designer's ideas about the section position within the page. Based on the former example (article page) a possible layout sketch may be what is illustrated in Figure 17.

Publishing Link

In order to model navigation at the publishing level, we redefine the notion of link in more general terms to denote a general user's interaction on a page. Most interactions are induced and constrained by the navigation design upon which

Figure 14. Graphical Notation Used for Specifying Page Type and Single Page Primitives

Figure 15. Graphical Representation of a Page and its Sections: General Definition and Usage



publishing design is built, but we also allow designers to introduce new interactions at this level for efficiency or usability reasons.

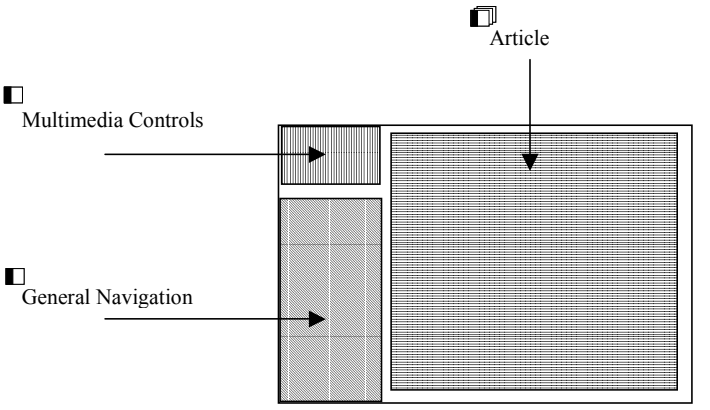
If the designer is using the model below a navigational model, we may distinguish between two kinds of links: *derived link* if it derives from a link defined in the navigation model, or *pure link* if it is not derived from any link. The pure links

Figure 16. Graphical Table Associated to a Page: General Definition and Usage

Page Type (or single) Description	
Name	Page name
Graphics	
Graphical details description (Informal).	
Content (Optional)	
Section name_1	
Section name_2	
Section name_n	
Main Topic	
Section name	

Page Type Description	
Name	Article
Graphics	
The page should be suitable for a 800 x 600 display mode. The main section.....	
Content	
Article	
Multimedia Controls	
General Navigation	
Main Topic	
Article	

Figure 17. Layout Sketch for the Product Page



are introduced in the presentation to add further navigation possibilities, allowing movement of the focus of the page on a particular unit or to navigate towards a particular page.

The source of a publishing link is always either a publishing unit type or a single publishing unit; the kind of the target of a publishing link depends on the kind of the link.

The navigation dynamics, in presentation, lead to the following kinds of links:

1. **Focus Link:** a focus link allows movement of the focus of the page from a publishing unit (or publishing cluster) to another one maintaining the current content of the page at the same time. Following this kind of link, we navigate within the same page instance. The target of this kind of link can be one of the following publishing elements: publishing unit (single, type), publishing cluster (single, type), or page section (single, type).
2. **Intra-Page Link:** an intra-page link allows navigating across the instances of the same page type. Following this kind of link results in the change of the page content, but not the change of the page structure. This link may or may not involve a focus change. The target of this kind of link can be one of the following publishing elements: publishing unit (single, type), publishing cluster (single, type), or page section (single, type).
3. **Page Link:** a page link allows navigation across the instances of different page types. Following this kind of link results in the change of the page content and structure. The target of this kind of link can be one of the following publishing elements: single page or page type.

In this case, the focus may sometimes (but rarely) change determining an automatic page scroll on the screen.

A publishing link is described by the following properties:

- **Name:** It univocally identifies the link in the Publishing Model. It is usually the name of the relationship from which the link is derived. Given two publishing elements, if there are more links between them, each link must have its own name, different from the name of other links.
- **Link Placeholder:** It describes the placeholder of the link, that is, the point on the page where the user should "click" to follow the link. Examples could be: the name of the link label, the rule which the label is caught from the application data, some presentation aspects such as button, underlined writing, etc.
- **Population Criteria:** It specifies the population rule used to define the content of the target, e.g., the publishing unit type (or single publishing unit) content belonging to the target. If the conceptual page model has been made below a navigational model, it could easily specify the target content in terms of node. Moreover it specifies the visualisation configuration if the clusters of the destination page have more than one.
- **(Optional) Focus:** It specifies if the link involves a focus moving.
- **(Optional) Navigation Pattern:** It models the meaning of instantiating a link, since it concisely

In Figure 18, some link placeholders of the central section of the example page are shown.

For specifying the properties of the page in the design document, the graphical set in Figure 19 is shown.

Each link is linked to a publishing unit as shown in Figure 20, and is described by a graphical table, as shown next to the figure.

FUTURE WORK

The model will be completed at the end of this year. Now we are developing a specific application in a bank environment, both to verify the whole W2000 model and to open the issues needed to adapt the model from the developer's point of view.

The next step will be the development of a navigation engine independent from a specific application domain, similar to the former engine named JWEB that was described in the above sections.

At the same time we are working on the definition of a suite of metrics to estimate the effort needed to develop a Web application. Currently we have this

Figure 18. Some Link Placeholders

Figure 19. Graphical Notation Used for Specifying the Focus Link, Intra-Page Link and Page Link Primitives

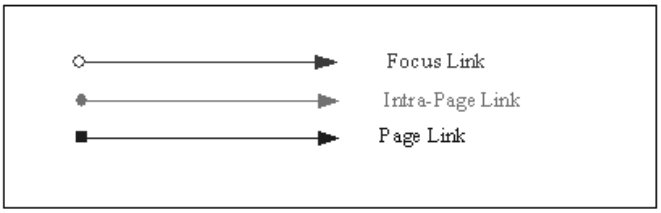
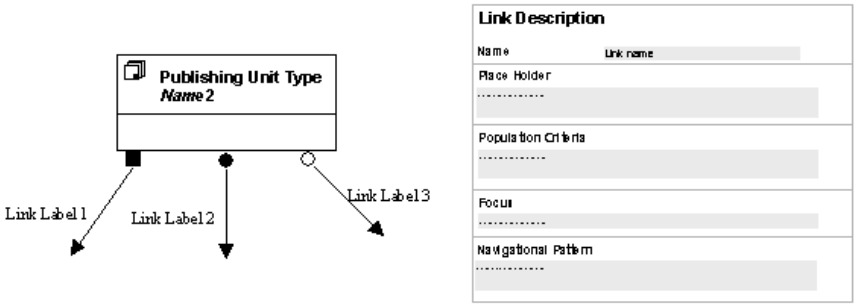


Figure 20. Graphical Notation Used for Attaching a Link to a Publishing Unit and for Describing the Link's Properties



suite based on the methodology of W2000 and we are working to generalize this suite to estimate the effort to develop a Web application independently from the W2000 model.

CONCLUSIONS

When this research project is completed, we will have a complete environment to develop Web application using a structured approach similar to what happens for the traditional applications.

Our suite will supply the developers with a complete design methodology, a way to estimate the effort (and the cost) in the early design phase, and a set of tools that will aid the developers in their work. In addition, they will have also a navigation engine, which represents a significant part of the whole effort.

REFERENCES

Baresi, L., Garzotto, F., & Paolini, P. (2001). Extending UML for Modeling Web Applications. In *Proceedings of the 34th Hawaii International Conference on System Sciences, Decision Technologies For Management track, Unified Modeling Language: A Critical Review and Suggested Future Minitrack* (Vol. 3, p. 3055). Maui, HI.

Bochicchio, M.A., Paiano, R., Paolini, P. (1999). JWeb: An HDM Environment for fast development of Web Applications. *Proceedings of Multimedia Computing and Systems 1999 (IEEE ICMCS '99), vol. 2* (pp. 809-813). Florence, Italy.

Ceri, S., Fraternali, P., & Bongio, A. (2000, May 5). Web Modeling Language (WebML): a modeling language for designing Web sites. In *Proceedings Int. Conf. WWW9* (pp. 137-157). Amsterdam.

Conallen, J. (1999). Modelling Web Application Architectures with UML. *Communications of the ACM*, 42(10).

Garzotto, F., Mainetti, L., & Paolini, P. (1995). Hypermedia Application Design: a Structured Approach. In J.W. Schuler, N. Hannemann, & N. Streitz (Eds.), *Designing User Interfaces for Hypermedia*. Springer Verlag.

Garzotto, F., Mainetti, L., & Paolini, P. (1996). Information Reuse in Hypermedia Applications. *Hypertext 1996*, Washington DC, USA, 93-104.

Garzotto, F., Paolini, P., & Schwabe, D. (1993). HDM - A Model Based Approach to Hypermedia Application Design. *ACM Transactions on Information Systems, 11* (1), 1-26.

Perrone, V., Maritati, M., Paolini, P., Baresi, L., Garzotto, F. & Mainetti, L. (2000). *Hypermedia and Operation Design: Model, Notation and Tool Architecture*. Official Deliverable D7 of the European Project UWA IST2000-25131.

Schwabe, D., & Rossi, G. (2000). *An Object Oriented Approach to Web-Based Application Design*. Available on the World Wide Web at: http://www.telemidia.puc-rio.br/oohdm/oohdm.htm.

UWA (Ubiquitous Web Applications) Project. Available on the World Wide Web at: http://www.uwaproject.org.