Conceptualization of Hybrid Websites

Antonio Navarro, Alfredo Fernández-Valmayor
Dpto. Ingeniería del Software e Inteligencia Artificial. Facultad de Informática.
Universidad Complutense de Madrid
C/ Profesor José García Santesmases, s/n
28040 Madrid, SPAIN
{anavarro, alfredo}@sip.ucm.es

ABSTRACT

Purpose
The purpose of this paper is to provide a mechanism to characterize websites, at the conceptualization stage, whose information content can be defined in terms of hybrid conceptual models. Usually in this type of website, some of the domain information units are modeled in terms of classes/entities while others remain as heterogeneous irregular elements. Websites built on this type of hybrid conceptual domain are referred to as *hybrid* websites in this paper.

Design/methodology/approach
We have characterized a hybrid website, at the conceptualization stage - the site of the Virtual Campus of the Universidad Complutense de Madrid. For the characterization of this site we used the *Pipe* notation, a notation that has proven to be appropriate when characterizing websites built on irregular domains. In addition, the Pipe notation was combined with entity-relationship diagrams to characterize the most homogeneous and regular elements of the website. We also analyzed several alternative notations intended for the characterization of websites.

Findings
We found that if in hybrid websites, entity-relationship diagrams are complemented with other notations able to characterize the most heterogeneous and irregular elements of these websites, important benefits can be obtained from a modeling point of view. These benefits are later easily translated into an easier evolution and maintenance of the website.

Research limitations/implications
Only some web applications can be characterized as hybrid websites. Therefore, there are many web applications whose conceptual domain is best characterized using classes/entities based models.

Practical implications
Persons in charge of modeling websites must first identify the nature of the information domain of the website in order to choose the most appropriate modeling mechanism.

Originality/value
This paper explicitly identifies hybrid websites and provides a notation which is able to characterize them at the conceptualization stage.

Keywords
Web engineering, units of information, semantic relations, conceptual domain, conceptualization, hybrid website

Paper type
Research paper

1. INTRODUCTION

The complexity of modern websites makes the use of advanced software engineering techniques necessary during their development (Fraternali, 1999). Many difficulties arise when traditional models and software engineering methodologies are used to design web applications because these models and methodologies are not particularly well suited to represent the most intrinsic features of web applications such as (Fraternali, 1999): the need to handle structured data (e.g. database records) and non-structured data (e.g. multimedia items); the support of exploratory access through navigational interfaces; the importance of high quality graphical interfaces; the customization and possibly the dynamic adaptation of content structure; the importance of navigational primitives and presentation styles, and finally the importance of offering support for proactive behavior (i.e., recommendation and filtering).

A key stage during the development of a website is the conceptualization stage, when the application is represented through a set of abstract models that convey the main components of the solution envisioned (Fraternali, 1999). At the very least, these models must characterize the structure and the navigational schema of the application. The structure describes the organization of the information managed by the site, in terms of units of information and of their semantic relations, while the navigational schema describes the facilities given to users for accessing and moving through the application informational content (Fraternali, 1999).

Nowadays, there are several notations that can be used to characterize a website at its conceptualization stage. Among others we find those described in (Ceri, 2000; Hennicker, 2001; Isakowitz, 1995; Schwabe, 1996).

Most of these notations characterize the elements of the application domain in terms of classes/entities and of their relations. In this way, relations between the elements at the application domain can be derived from the relations established between the entities/classes that model them. These notations take into account the potential presence of individual elements using *singleton* classes/entities (Gamma, 1995). For example, websites like those of *Amazon* (Amazon, 2006) and *Dell* (Dell, 2006) in the USA, have application domains that can be easily represented in terms of classes/entities and their relationships.

The problem arises when, due to the nature of the information managed by the website, some elements can be modeled in terms of classes/entities (standard or singleton) while others, more heterogeneous in nature, cannot be easily assimilated to instances of classes/entities (Balasubramanian, 1997; Isakowitz, 1995). For example, in the *Novartis Medical Nutrition* website in the USA (Novartis, 2006) there is information about medical nutritional products that can be easily represented in terms of classes/entities: product presentation, technical features, ordering information, nutrition information, etc. But in addition,

there is another type of information outside of this regular structure, i.e., information about the company, the code of conduct and business ethics policies of the company, information about the ISO 9001:2000 certification, etc. Note that this information is part of the website but this information is very heterogeneous in nature and has a great number of heterogeneous links that cannot be easily represented in terms of classes/entities and their relations (Balasubramanian, 1997; Isakowitz, 1995). In addition, the information deployed by these heterogeneous elements (i.e., text, images, videos, etc.) is not suitable for being described in terms of attributes of classes/entities.

We call this type of website a *hybrid website*. We have encountered this type of site when designing the website of the *Virtual Campus of the Universidad Complutense de Madrid* (VCUCM, 2006). Part of this website describes the nature and the objectives of the project, its main actors and their roles, tutorials and support information for students and teachers, etc. All this information can be represented in terms of heterogeneous units of information and their semantic relations. Another part of the website provides information on temporally organized news, registration facilities and statistics on professors, centers, departments, etc. This part can be easily characterized by several relational databases that are dynamically queried (Bodner, 1999). The main question here is how to represent both types of information and integrate them into a common conceptualization notation.

In our solution, we have complemented entity-relationship diagrams (Chen, 1976) with *Pipe* (Navarro, 2004b), a notation that we had developed for the characterization of hypermedia applications on *irregular* domains, i.e., domains not easily represented in terms of entities/classes and their relations (Isakowitz, 1995). The abstraction level of Pipe notation has demonstrated its usefulness in modeling this type of hybrid website.

The rest of the paper is organized as follows. Section 2 introduces Pipe notation. Section 3 briefly describes the Virtual Campus of the Universidad Complutense de Madrid. Section 4 describes how Pipe

notation is used to model the website of the Virtual Campus. Section 5 compares our approach with other well-known approaches described in the literature. Finally, section 6 presents the conclusion and future work.

2. PIPE NOTATION

Pipe is a formalized graphical notation for the characterization of hypermedia applications. A more complete description of the notation can be found in (Navarro, 2004a) whereas the formalization can be found in (Navarro, 2004b). Pipe is conceived to be used in domains that cannot be easily characterized in terms of entities/classes and their relations. Consequently, Pipe is focused on characterizing: (i) the units of information as individuals and their meaningful relations; (ii) the user interface of the application; and (iii) the relations between the layer of contents and the layer of user interface to provide navigational access to the elements of the application.

Pipe provides three types of diagrams to characterize the previous information: (i) a *contents graph*, which characterizes the structure of the informational elements of the application and their relations; (ii) a *navigational schema*, which characterizes the graphical user interface of the application; and (iii) a set of *canalization functions* which relate the contents graph and the navigational schema. This section briefly describes these elements. Note that some notational elements have been changed since (Navarro, 2004b) in order to use them for modeling websites.

2.1 Contents graph

The Pipe contents graph models how units of information are structured in the application and their relations (Navarro, 2004b). The elements used by the model are: information units, anchors, links and

the relationship function that captures all the structure and semantic relations between units. More precisely the Pipe contents graph uses the following modeling elements:

- C_s: *set of static contents*. Units of information that exist previous to any interaction with the user.
- H_s: *set of static anchors*. Anchors at the end of static links. Their selection by users does not generate new information in the application.
- L_s: *set of static links*. Relations between static anchors.
- C_d: *set of dynamic contents*. Units of information generated by user interaction (Bodner, 1999).
- H_d: *set of dynamic anchors*. Anchors at the end of dynamic links. Their selection by users generates new information in the application.
- L_d: *set of dynamic links*. Relations between dynamic anchors.
- $C = C_s \cup C_d$: *set of contents*. Static or dynamic units of information.
- $H = H_s \cup H_d$: *set of anchors*. Static or dynamic anchors.
- $L = L_s \cup L_d$: *set of links*. Static or dynamic links.

Figure 1 depicts the graphical representation of elements of the contents graph.

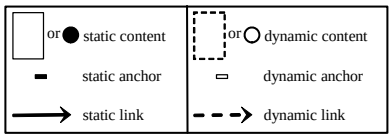


Figure 1. Visual elements of the contents graph

Since we are interested in a characterization of dynamic applications, there will be cases where the set of links L cannot be characterized by extension, but by intension (e.g., the links established as the result of a query in a form). Therefore, we introduce the *relationship function r* (where the set *Input* characterizes all the possible inputs that can affect the computational behavior linked to an anchor selection):

$$r: H \times Input \rightarrow H$$

In this way the set of links of an application can be characterized as:

$$L = \{ (a, i, r(a, i)) \mid (a, i) \in H \times Input \}$$

Therefore, in Pipe, links are simply relations established between anchors. Note that an input can be related to an origin anchor in order to represent the information that flows from the user to the application (e.g. the information collected in a web form). The key idea is to provide an extensional definition of the relationship function *r* for static links and an intensional definition of the relationship function *r* for dynamic links.

In order to provide its extensional definition it is only necessary to define a function r^0 that relates static anchors with static anchors:

$$r^0: H_s \rightarrow H_s$$

In order to provide its intensional definition it is necessary to define *generation function g*:

$$g : H_d \times Input \rightarrow H_d \times C_d \times 2^{H_d} \times 2^{H_s}$$

Where an element $g(a, i) = (a', c', GSA_d', GSA_s')$ denotes that when dynamic anchor *a* is selected with input *i*, destination anchor *a'*, content *c'*, the set of dynamic generated source anchors GSA_d' and the set of static generated source anchors GSA_s' are generated.

For example, if anchor *a* selects a detailed description of a lecturer, and input *i* identifies a concrete lecturer, content *c'* represents the detailed description of such a lecturer, anchor *a'* represents a concrete position inside such a description and sets GSA_d' *and* GSA_i' characterize the anchors (dynamic and static) included in such a content.

Therefore, relationship function *r* can be defined (where $\prod_1(x, y, z, t) = x$):

$$\begin{array}{lll} r: H \times Input & \rightarrow & H \\ (a, null) & \rightarrow & r^0(a), \quad a \in H_s \\ (a, i) & \rightarrow & \prod_1 g(a, i), \; a \in H_d \end{array}$$

For example, if anchor *a* is a static anchor which gives access to the *index* page, relationship function *r* retrieves the anchor that characterizes the said index page. If anchor *a* is a dynamic anchor as in the previously mentioned example of generation function, relationship function *r* invokes generation function *g*.

In this way, function *r* acts as a *black box* that hides the real nature of the links (static or dynamic). From an architectural point of view, the relationship function r plays the role of a controller in a *Model-View-Controller* (MVC or Model 2) architecture in web applications (Brown, 2001). More specifically, the definition of the function *r* can be made in the same way as the definition of the table that configures the controller of a framework that implements the MVC architecture, e.g. *Struts* (Struts, 2006). In this way, and in terms of the *struts-config* file (Struts, 2006), if there is no need for dynamic behavior for a specific anchor, the definition of the function r for this anchor can be made statically, i.e., the *action* (Struts, 2006) defined in the struts-config file is a simply *forward* (Struts, 2006) to a static page. On the contrary, if a dynamic behavior is necessary for a specific anchor, the definition of the function *r* for this anchor is made in terms of the generation function *g*, i.e., the action defined in the struts-config file makes reference to a *Java* (Java, 2006) class responsible for implementing this dynamic behavior. In fact, the function *r* formalizes this behavior that is depicted graphically by a continuous line (static linking) or by a broken line (dynamic linking).

As regards the integration of Pipe with entity-relationship diagrams, Figure 2 shows how a RMM (*Relation Management Methodology*) *index*[1] (Isakowitz, 1995) can be depicted graphically using the Pipe contents graph notation. In this figure an index of associate professors is extracted in Pipe considering the entities and relations depicted.

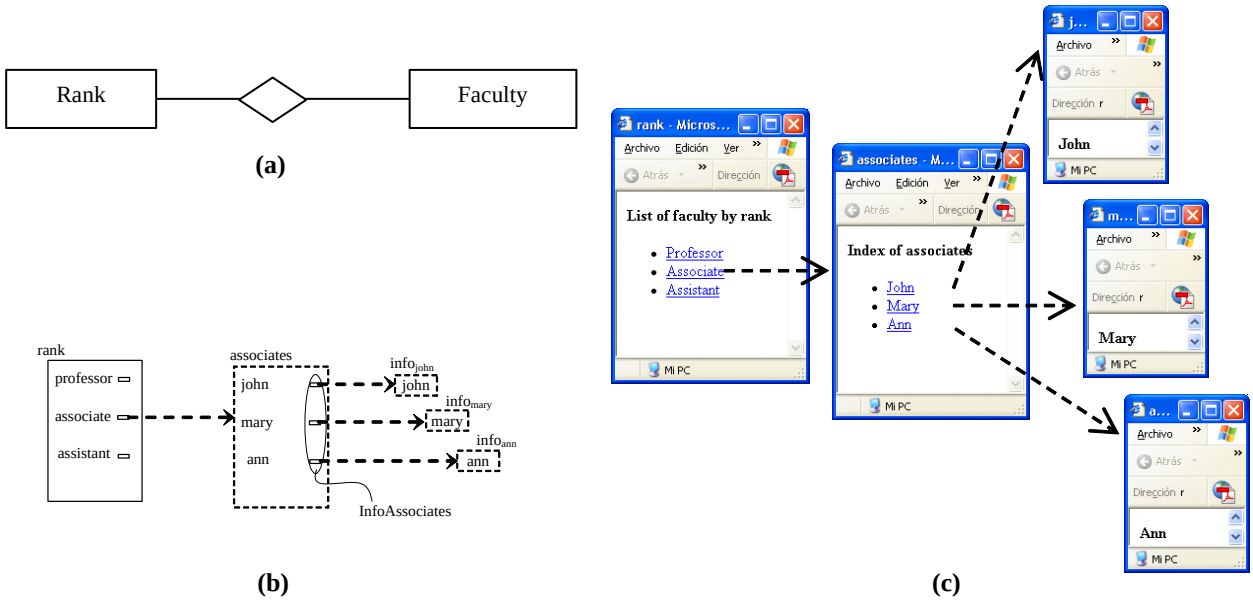


Figure 2. (a). E-R characterization of a database. (b) Content dynamically generated from elements contained in a database. (c) Meaning of the diagram depicted in (b) in terms of HTML pages

The formal expression that characterizes part of Figure 2 (b) is:

g(associate, null) = (associates, associates, InfoAssociates, ∅)

Note that in this expression there is no input linked to the anchor *associate* because it is not necessary (i.e., the input has a *null* value). Note also that the function *g* generates the content *associates* and an anchor of the same name. By default, in Pipe, whenever a dynamic content is generated, automatically an anchor with the same name of the content is generated and attached to it. In particular, the content *associates* is characterized by the expression:

associates = { p | faculty(p) ∧ associate(p) }

[1] A RMM index is a modeling primitive which if applied to an entity, generates an index content with anchors to every instance of such an entity.

Thus, *associates* is the set of all the faculty which are associated. In addition, *InfoAssociates*, the set of generated dynamic anchors of the content generated *associates*, is characterized by the expression:

$InfoAssociates = \{ a_p \mid \forall p \in associates \; \exists a_p \; (g(a_p, null) = (info_p, info_p, X, Y)) \}$

Thus, *InfoAssociates* is the set of anchors where for every associate *p*, there is an anchor a_p in such a set that depicts the information of every associate, $info_p$, when selected. In other words, the content *associates* is the index of associates and *InfoAssociates* is the set of anchors that give access to the description of every associate. In Pipe, as in this example, first order logic is used to explicitly characterize the elements generated by generation function *g*. If relational databases are used, specific formalisms such as *relational algebra* or *relational calculus* (Elmasri, 1994) could be used.

2.2 Navigational schema

The Pipe navigational schema makes an important abstraction of the graphical elements that compose a graphical user interface (GUI) and of the navigational relations among the elements of the GUI of hypermedia applications. The Pipe navigational schema uses four modeling elements:

- *Nexus nodes*. Windows that contain container nodes.
- *Container nodes*. Panes that are inside windows. They contain and show the contents (static or dynamic) of the contents graph.
- *Connections*. Which represent how a container node is attached to its nexus node (i.e., they represent relations of aggregation between windows and their panes).
- *Paths*. Which represent the navigational relations that exist between container nodes in the GUI. In this way if a path exists between a container node c_1 and a container node c_2, it is possible to display in c_2 the destination of a link with origin in a content that is being displayed in c_1. Paths

are also called *pipes* because they are responsible for canalizing (interpreting) the relations (links) established between the units of information (contents) in the navigational schema (GUI). Figure 3 depicts the visual elements of the navigational schema and the allowable relationships between them.

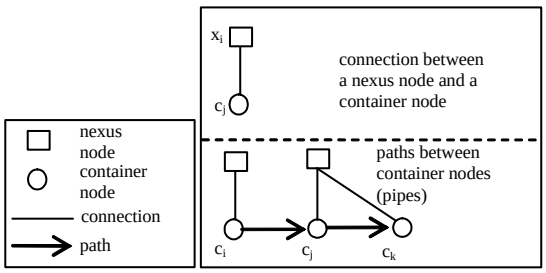
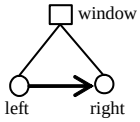
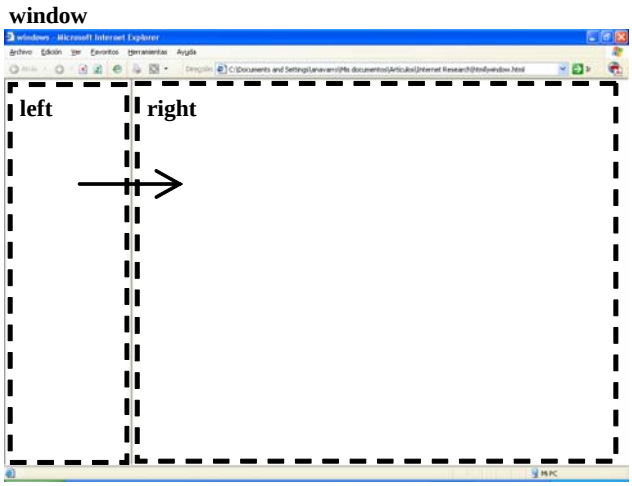


Figure 3. Visual elements of the navigational schema and allowable relationships between them

Figure 4 depicts a window (*window*) with two regions (*left* and *right*) and a navigational relationship between both regions. The meaning of this navigational relationship is explained in the next section.



(a)



(b)

Figure 4. (a) A window with two frames in Pipe. (b) Meaning of the diagram depicted in (a)

2.3 Canalization functions

The Pipe canalization functions relate the elements of the contents graph to the elements of the navigational schema. In this way different user interfaces can be used to visualize with the same content

graph. Likewise, the same user interface navigational schema can be used with different content graphs. There are two canalization functions in Pipe:

- *Content assignation function*. This function assigns units of information (contents) to container nodes (panes).
- *Canalization function*. This function assigns relations between units of information (links) to paths/pipes between container nodes (panes).

Colors and visual patterns are used in Pipe to characterize the canalization functions. A complete description of these functions is outside the scope of this paper. (Navarro, 2004b) is a valuable reference.

Figure 5 depicts the assignation of the content *rank* to the *left* region and the content *associates* to the *right* region. Note that the link between both contents is assigned to the pipe between both regions. Therefore, if the user selects the anchor *associate*, the content *associates* appears in the *right* region. A formal semantics that characterizes this behavior has been defined (Navarro, 2004b). Again, its description is outside the scope of this paper.

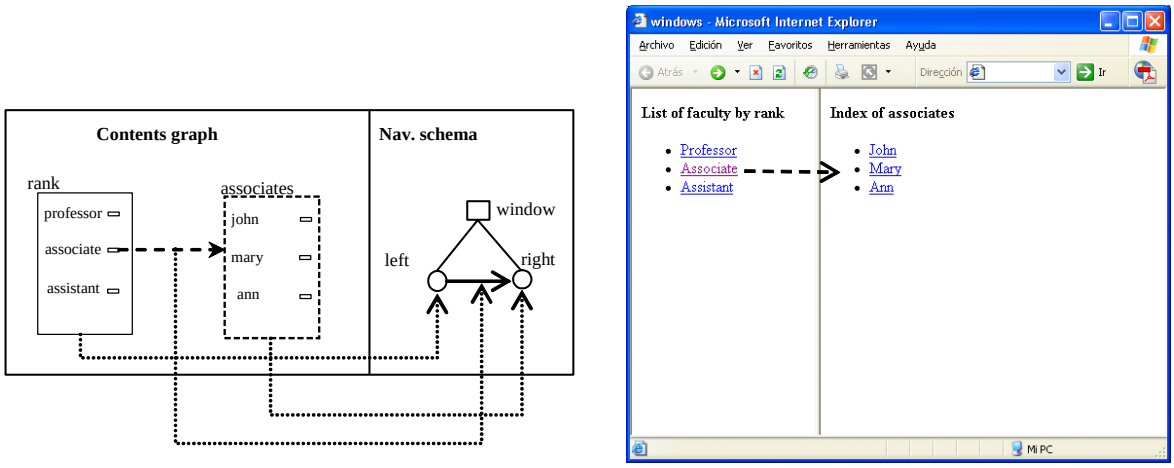


Figure 5. (a) Relations between contents graph and navigational schema. (b) Meaning of the diagram depicted in (b) in terms of HTML pages

3. THE VIRTUAL CAMPUS OF THE UNIVERSIDAD COMPLUTENSE DE MADRID

The *Universidad Complutense de Madrid* (UCM) is an old university (it was founded in the year 1499) and is currently the largest non-open university in Spain. In the academic year 2005-2006 there were almost 90,000 students and more than 6,000 faculty members (UCM, 2006). The UCM teaches 77 official programs and 174 doctoral programs. Today, as many other universities are doing, the UCM is making a major effort to introduce modern e-learning technologies to improve its educational processes. In the academic year 2003-2004 the *Virtual Campus of the UCM* (VCUCM, 2006) was set up. The main objective of the project was to provide students and teachers with all the support that modern information and communications technologies can provide to improve the quality of learning and research activity at the university. The Virtual Campus includes the management of students enrolled in the courses, the management of the contents of courses and collaborative and communication facilities: work groups, chats, forums, etc. At present, more than 46,000 students and 2,700 faculty members are registered in the Virtual Campus.

In the website of the Virtual Campus there are two areas. The web of the Virtual Campus (Figure 6) and the course management system, the web application that manages courses and learning activities (Figure 7).

Figure 6. The Web of the Virtual Campus of the UCM

From its conception, the creation and management of the Virtual Campus has been the responsibility of members of our group. We selected the commercial application, *WebCT* (WebCT, 2006), which supports the courses taught at the Virtual Campus. We also created the website that provides information and support for the Campus. This paper is focused on the website itself, because WebCT is a complex system beyond the scope of this paper.

The largest part of the Virtual Campus web describes the use of the tool (Figure 6). In addition, there is a section devoted to news, registration services and statistics on the Virtual Campus. For example, Figure 8 shows some statistics on the number of professors registered in the Virtual Campus. At present, only the number of teachers by center is provided but in the near future it will provide more information as the web is constantly under development.

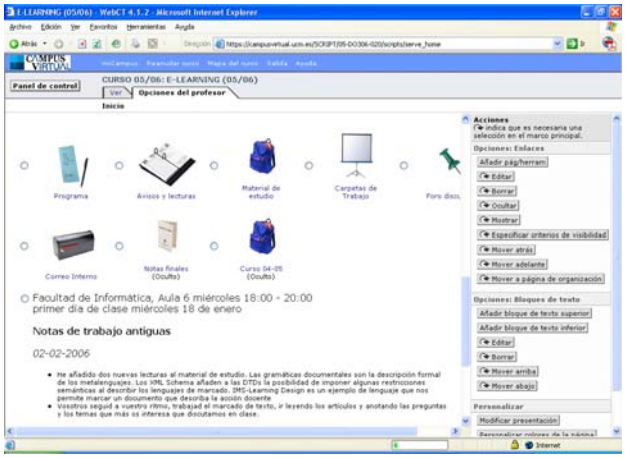


Figure 7. The Virtual Campus of the UCM

In this website, there is no natural way to represent many of the units of information about the Virtual Campus as classes/entities. Items such as: a brief description of the virtual campus, a virtual tour, or the description of some organizational aspects and objectives of the virtual campus cannot be modeled as elements with common characteristics that can be abstracted in terms of entity-relationship or class diagrams. On the contrary, the information needed for teachers and students to register in and to compute the statistics of the Virtual Campus (i.e., students, courses, professors, etc.) can be easily

characterized in terms of entities and their relations. This is a hybrid situation where two types of models have to be used to characterize the informational domain of a website.

The following sections describe the website of the Virtual Campus of the UCM and how the Pipe notation has been used to characterize this website at the conceptualization stage.

Figure 8. Statistics on registered teachers in the Virtual Campus

4. PIPE CHARACTERIZATION OF THE WEB OF THE VIRTUAL CAMPUS OF THE UCM

This section discusses the characterization of the web of the UCM Virtual Campus in terms of Pipe and entity-relationship diagrams.

4.1 Navigational schema

For the sake of conciseness we shall imagine a GUI divided into three frames. An up frame, a left frame, and a main frame. Figure 9 (a) depicts these frames in the website. The Pipe navigational schema for characterizing this GUI is depicted in Figure 9 (b).

In Figure 9 (b), the nexus node *virtual campus* represents the entire *frameset* (W3C, 1999), while the container nodes *up*, *left*, and *main* represent the *frames* (W3C, 1999) or regions of the page. In addition,

there are navigational relationships established between the nexus nodes *left* and *main*, and *main* with itself.

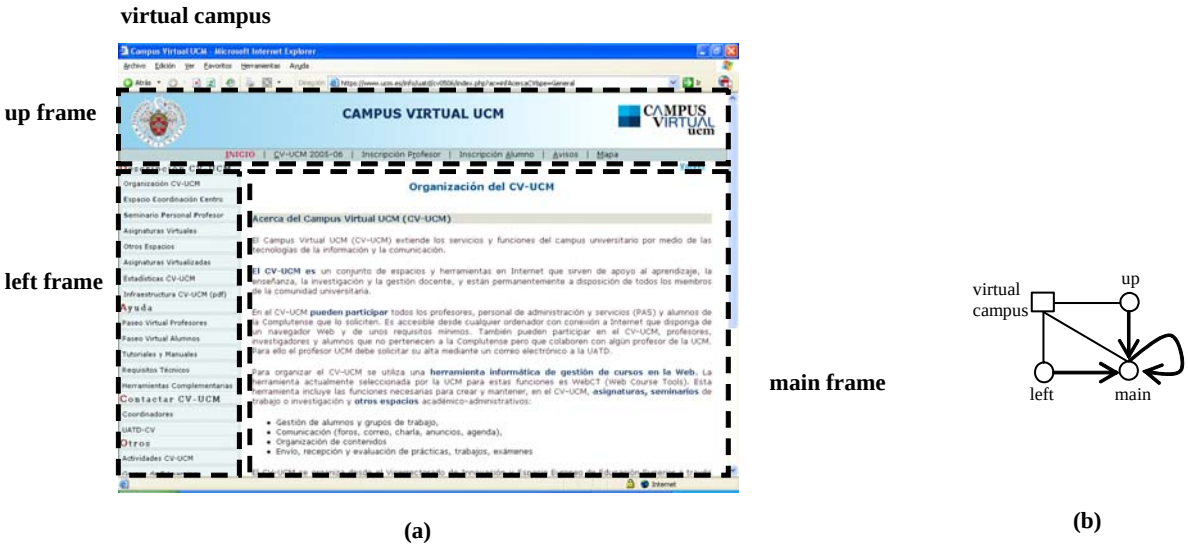


Figure 9. (a) The division of frames in the website of the Virtual Campus. (b) Pipe navigational schema for the website

4.2 Contents graph

There are too many contents in the website of the Virtual Campus to characterize all of them in this paper. For the sake of conciseness we shall only focus on some of the contents that appear in the *left* and *main* frames. We shall ignore the contents that are not essential to the example.

Figure 10 depicts part of the contents graph of the contents of the Virtual Campus website.

In this figure, most of the contents are static, with static links between them[2]. For example, the static content *LeftIndex* is linked with the static content *CVUCM*. However, note that the content *Statistics* has several dynamic links with the dynamically generated units of information that provide some statistics on the faculty, students and courses. Although the Pipe contents graph depicts the hypertext structure of the application, there is important information that is not captured in it. This information is on the

structure of the underlying database, which is considered when computing the statistics. To characterize this type of information, an entity-relationship diagram (Chen, 1976), for example, can be used. Figure 11 depicts such a diagram.

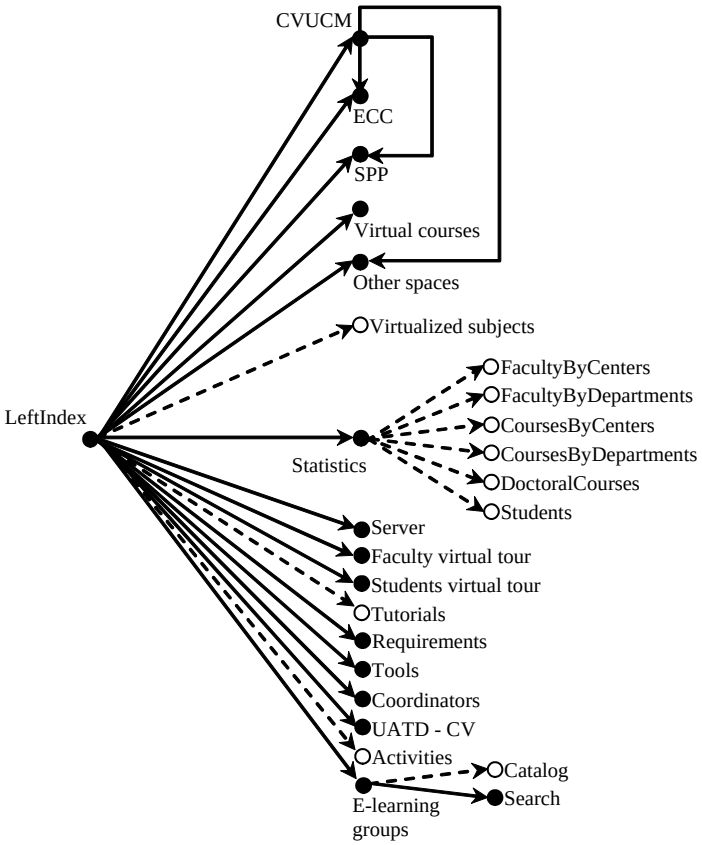


Figure 10. Part of the contents graph of the Virtual Campus

The hybrid nature of the website makes the presence of two types of models necessary to characterize the units of information and their relations. The first, the content graph, models heterogeneous units of information with heterogeneous relations (e.g. *LeftIndex*, *CVUCM*, *ECC*, *SPP*, etc.). The second, the entity-relationship diagram, models homogeneous regular units of information in terms of entities with homogeneous relations between them. In this section of the Virtual Campus website, only statistics on

[2] As previously mentioned, in Pipe, links are established between anchors. The figure shows the compact notation where anchors are not depicted. In addition, in Pipe there is a default anchor assigned to every content, which represents the initial position in the content. This anchor is represented as the content itself.

the entities are computed. In other sections of the website that relate teachers with subjects and with programs, more elaborate information on the entities is displayed.

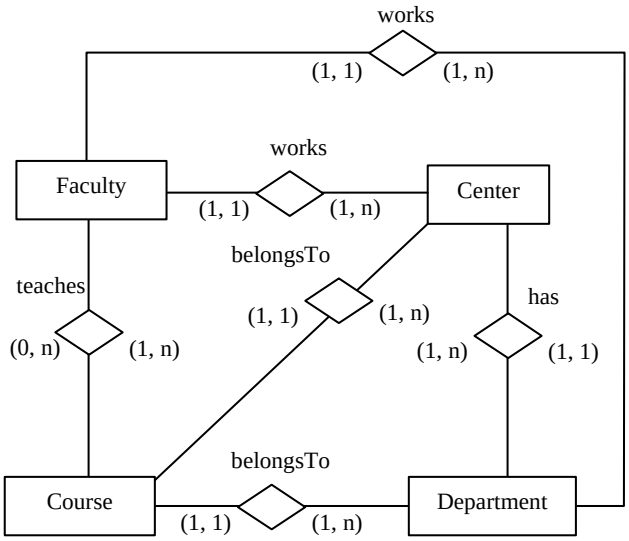


Figure 11. Entity-Relationship diagram for regular units of information in the Virtual Campus.

Note that because of the high level of abstraction of Pipe notation, and because of the representation of dynamic links in terms of the generation function *g*, it is possible to dynamically model the units of information (e.g. *FacultyByCenters*) extracted from the relational database. In this way, hybrid websites can be homogeneously treated using the function *r*, extensionally defined for static links, and intensionally defined, in terms of the generation function *g*, for dynamic links. This reasoning is the same as that used in (Navarro, 2005) to characterize navigational maps of web applications using Pipe notation, but in this case, the heterogeneous contents of the website (e.g. *CVUCM*) are contents on their own instead of being views of the regular contents of the website.

Note that in this example, static contents can be assimilated to static HTML pages. In the case of web applications where even the non-interactive content and the content that appears to be static are dynamically retrieved from a database, the linking between contents determines the modeling approach. Thus, if these contents are independent items without links, the entity-relationship diagram that

characterizes these contents may be enough (as in the case of statistics on the faculty presented in this paper). Otherwise, if there is a significant amount of links between these contents, a Pipe contents graph, with dynamic links between their contents (ultimately these contents are dynamically extracted from a database) should be used. This approach is extensible to relational, object-oriented or XML databases.

In addition, Pipe notation is not concerned with the specification of the computational behavior represented by dynamic links (e.g. a Java class named in the strut-config file) because Pipe is intended to be used at the conceptualization stage. If desired, appropriate modeling mechanisms, e.g. UML *sequence diagrams* (OMG, 2004), can be used to describe this computational behavior. In this case, note that the dynamic links work like identifiers of *use cases* (OMG, 2004) specified by sequence diagrams. In other words, taking into account Figure 10, designers have to provide use cases and sequence diagrams to specify the computational behavior of the class responsible for the generation of the content *FacultyByCenters* that, among others, in Figure 10 are graphically represented by a dynamic link (broken arrow), and that are formally represented (Navarro, 04b) by the relationship function *r*, defined in this case, in terms of the generation function *g*.

Moreover, the units of information of Figure 10 could have been modeled as instances of an entity of items with relations among themselves, but this is closer to a modeling trick than to a true model of these elements and their relations. Figure 12 (a) depicts such a model. Another choice is to identify every irregular element with a singleton entity. This provides cumbersome diagrams as the one depicted in Figure 12 (b). In addition, this last choice makes it necessary for the contents of the web page (text, images, etc.) to be characterized by the attributes of the entity. Finally, note that UML *class diagrams* (OMG, 2004) could have been used instead of entity-relationship diagrams to characterize the regular part of the application domain, because the use of entity-relationship or class diagrams does not interfere with Pipe notation.

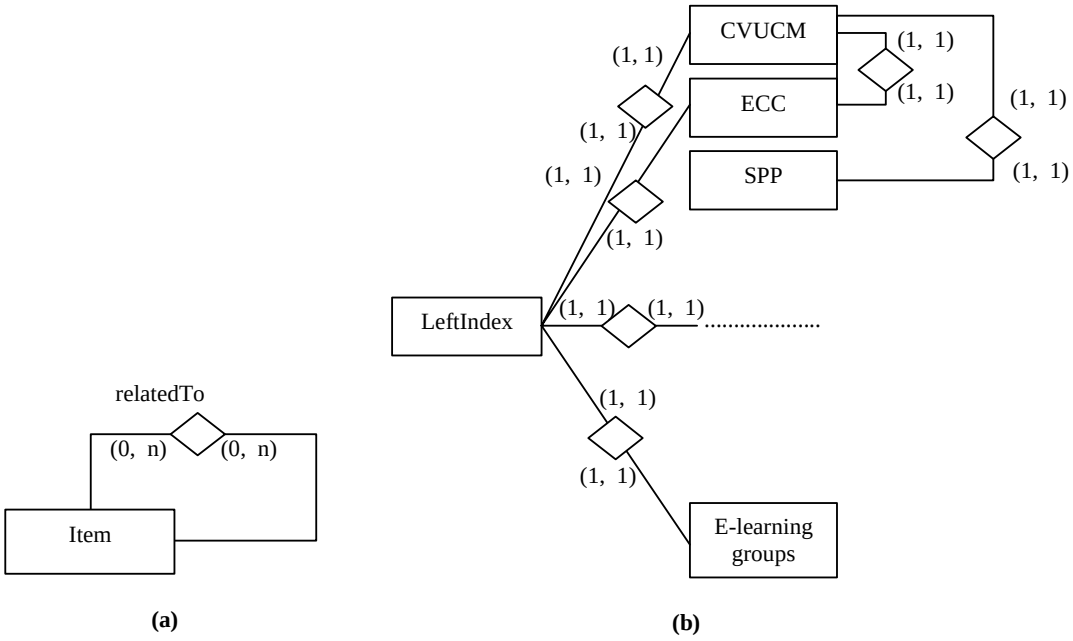


Figure 12. (a) Modeling trick for the informational items of Figure 10 and their relations. (b) The singleton approach to characterize the irregular elements of the domain

4.3 Canalization functions

Once the elements of the contents graph and of the navigational schema have been defined, it is time to define how they are related. Figure 13 depicts the assignment of contents to container nodes and the canalization of content links into the pipes or paths of the navigational schema.

In particular, *LeftIndex* is assigned to the *left* frame (must appear in it) and the rest of contents is assigned to the *main* frame. That is why in Figure 13, *LeftIndex* is assigned to *left* (i.e., content *LeftIndex* is assigned to container node *left*). As regards links, the destination of the links originating in *LeftIndex* must appear in the *main* frame. Consequently, all these links are canalized by the pipe/path defined between the frames *left* and *main*. That is why in Figure 13 these links are assigned to this pipe (i.e., links between *LeftIndex* and *CVUCM, ECC, SPP, ..., Statistics*, etc. are assigned to the pipe between *left* and *main*). The destination of the links originating in *CVUCM, Statistics* and *E-learning groups* must

also appear in the *main* frame. Consequently, these links are also canalized by the pipe with origin and destination in the *main* frame.

Note how, regardless of the nature of the units of information considered (i.e., static contents or contents dynamically generated from the database), these units are equally modeled and interpreted at the navigational level. Again, this is due to the abstract power of Pipe and of the relationship function *r* that allows the real nature of the contents and links to be hidden.

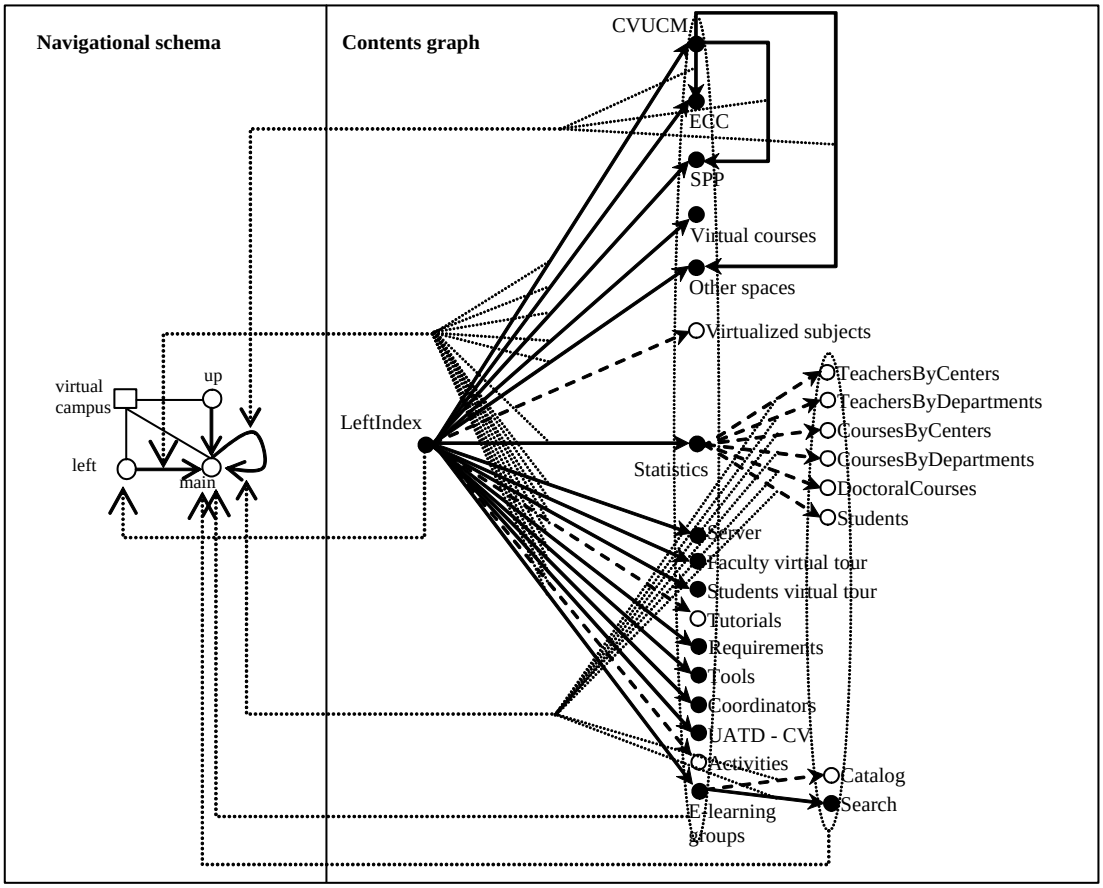


Figure 13 Relations between the contents graph and the navigational schema

5. RELATED WORK

In the literature, in general, most of the case studies modeled do not belong to the hybrid type of website. In most of these case studies, usually both the units of information and their relations are modeled in terms of classes/entities and their relations.

Relation Management Methodology (RMM) (Isakowitz, 1995) models the domain of the application in terms of entity-relationship diagrams. The model uses access primitives to derive from the relations defined between entities, the links established between the elements belonging to the entities. RMM uses the foreign keys defined in the relational schema (Codd, 1970), associated with the entity-relationship diagrams to automatically define these links between elements. Moreover, although considered an activity of the process, RMM does not explicitly characterize the components of the user interface or the relations between these components and the units of information in the application.

Object-Oriented Hypertext Design Model (OOHDM) (Schwabe, 1996) models the domain of the application in terms of object-oriented classes and their associated relations. OOHDM also defines navigational classes that convert traditional object-oriented classes into hypermedia classes. Thus, navigational classes are views of the object-oriented classes of the domain. OOHDM provides a very advanced characterization of the user interface, and of the way in which this user interface is related to the navigational classes.

The two aforementioned models have their origins in the hypermedia domain, but in recent years, several models have appeared in the web engineering domain. In particular *Web Modeling Language* (WebML) (Ceri, 2000) is a modeling language based on the ideas of RMM. WebML provides an XML representation of the components of the model, providing prototyping/development characteristics. In addition, it provides a more advanced description of the elements in the application's user interface and

its relations with the contents of the application. *UML-based Web Engineering Approach* (UWE) (Hennicker, 2001) is a piece of work based on OOHDM that defines access primitives similar to those presented in RMM.

All the case studies presented in these models correspond to domains that can be easily represented in terms of classes/entities and their relations, omitting the modeling of heterogeneous individuals with heterogeneous relations. Table I depicts the case studies included in these models and the entities/relations considered.

Table I. Several modeling approaches with their case study

Approach	**Case study**	**Entities/classes**
RMM (Isakowitz, 1995)	University	Faculty, Courses, Program, ...
OOHDM (Schwabe, 1996)	Museum	Person, Picture, Artwork, ...
WebML (Ceri, 2000)	Music Database	Artist, Album, Track, ...
UWE (Hennicker, 2001)	Company	Company, Department, Employee, ...

Note that when we say "heterogeneous individuals with heterogeneous relations", we mean *irregular* elements and relations in the sense of (Isakowitz, 1995), no individual element instances of singleton classes. In any case, if any of these modeling notations are used to characterize hybrid domains, diagrams as those depicted in Figure 12 arise. In addition, note that in this case, all the heterogeneous contents of the irregular elements (e.g. streams of texts and images) have to be described in terms of attributes of classes or entities.

Conallen´s extension to UML (Conallen, 1999) provides a set of modeling facilities for web applications. Although based on UML, Conallen's extension to UML permits the *pages* (Conallen, 1999) of the applications to be considered as individual elements without the need to use singleton classes. These pages can be stereotyped as client or server pages, and their heterogeneous relations with other pages are characterized in terms of navigational associations. Conallen's extension also provides a specific characterization of the user interface in terms of frames and framesets (Conallen, 1999). In the

case of dynamic relations between units of information, Conallen explicitly provides the object-oriented class responsible for the computational processing of their relations. In fact, the use of web stereotypes and the presence of object-oriented classes (which constrain the implementation of the application) are the main drawbacks when using Conallen extensions for the conceptualization stage of websites (Navarro, 2005).

Although this paper is focused on web engineering approaches, some of the issues related to hybrid information are being addressed in the knowledge management community. Thus (Satyadas, 2001) identifies *structured* and *unstructured* knowledge as types of *explicit* knowledge (opposed to *tacit* knowledge) (Marwick, 2001). In a halfway approach between knowledge management and information systems, (Jhingran, 2002) identifies *structured* and *unstructured* contents. Creation/synthesizing, capturing/storing, organization/integration and dissemination/sharing of knowledge are some of the key issues addressed by the knowledge management community (Cody, 2002; Hotho, 2001; Satyadas, 2001). This paper, on the contrary, is mainly focused on the modeling of the navigational interpretation of the contents (structured and unstructured) managed for a web application.

6. CONCLUSIONS

The conceptualization stage is a key step in the process of building and maintaining websites, but it is far from being a simple task. In the case of non-hybrid websites, there are several models created to deal with the conceptualization of this type of application.

The fact is that none of the case studies included in the descriptions of these models considers hybrid domains where some informational elements can be modeled in terms of entities/classes and their relations, while others remain as irregular elements with heterogeneous relations. In our opinion, these hybrid domains are not uncommon. In this type of site there is, at least, one relational database which

contains persistent and highly-structured data, but these sites also have a set of pages with contextual, heterogeneous and linked contents that are not included in the database. In particular, we encountered this situation while designing the website of the Virtual Campus of the Universidad Complutense de Madrid.

To model this website at the conceptual stage we used the Pipe notation combined with entity-relationship diagrams. Pipe's high level of abstraction permits the presence of heterogeneous diagrams because ultimately, the Pipe contents graph omits the nature of the information considered. Pipe provides a useful tool, the relationship function *r* that makes a navigational interpretation of the contents and of their links possible, regardless of their nature.

Conallen´s extension could also have been used to model the Virtual Campus of the UCM. But Conallen´s extension is more suited to the design stage, because it includes specific design decisions (e.g. processing in server pages) that should be omitted at the conceptualization stage (Navarro, 2005). In addition, our experience in the past indicates that Conallen´s UML-based extension is more difficult for non-computer science customers to understand than Pipe, and the presence of the customers at the conceptualization stage is very valuable (Navarro, 2005).

There is a lot of work to be done. In this paper we have depicted how Pipe can be combined with an entity-relationship model. In the future, we have to analyze the integration of Pipe with other models (e.g. WebML). In particular the relations of both models at a navigational level must be carefully analyzed. Moreover, Pipe has fast-prototyping facilities (Navarro, 2004a) using XML (W3C, 2004) and Java. Thus, at present, we are trying to expand these facilities to include information extracted from relational databases in fast prototyping (as WebML does). Furthermore, we are working on automatic translation from Pipe notation to Conallen´s extension (in both directions) to alleviate the transition from the conceptualization stage to the design stage and to facilitate reverse engineering. Finally, the

relationships between the approach for the modeling of structured and unstructured contents presented in this paper, and the approaches proposed by the knowledge management community, should be analyzed in-depth.

7. ACKNOWLEDGMENTS

El Ministerio de Educación y Ciencia de España (TIN2004-08367-C02-02 and TIN2005-08788-C04-01), *La Dirección General de Universidades e Investigación de la Consejería de Educación de la Comunidad de Madrid* and *La Universidad Complutense de Madrid* (*Grupo de investigación consolidado 910494*)" have supported this work.

8. REFERENCES

Amazon US (2006), http://www.amazon.com

Balasubramanian, V., Bashian, A. and Porcher, D. (1997) “A Large-scale hypermedia Application using Document Management and Web Technologies”. In Proceedings of Hypertext 97, Southampton.

Bodner, R.C., and Chignell, M.H. (1999) “Dynamic hypertext: querying and linking”. ACM Computing Surveys, Vol. 31, nº 4.

Brown, S., et al. (2001) Professional JSP. 2nd Edition, Wrox Press.

Ceri, S., Fraternali, P, and Bongio, A. (2000) “Web Modeling Language (WebML): a modeling language for designing Web sites”. Computer Networks, Vol. 33 No. 1-6, pp. 137-157.

Chen, P.P. (1976) “The Entity-Relationship Model - Toward a Unified View of Data”. ACM Transactions on Database Systems Vol. 1 No. 1, pp. 9-36.

Codd, E.F. (1970) “A Relational Model of Data for Large Shared Data Banks”. Communications of the ACM Vol. 13 No. 6, pp. 377-387.

Cody, W.F, Kreulen, J.T., Krishna, V., Spangler, W.S (2002). "The Integration of business intelligence and knowledge management". IBM Systems Journal, Vol. 41 No. 4, pp. 697-713.

Conallen. J. (1999) "Modeling Web Application Architectures with UML". Communications of the ACM Vol. 42 No. 10, pp. 63-70.

Dell US (2006), http://www.dell.com

Elmasri, R. and Navathe, S.B. (1994) Fundamental of Database Systems, Benjamin/Cummings Publishing Company, Inc.

Fraternali, P. (1999) "Tools and Approaches for Developing Data-Intensive Web Applications: A Survey". ACM Computing Surveys Vol. 31 No. 3, pp. 227-263.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Hennicker, R., and Koch, N. (2001) "Systematic Design of Web Applications with UML". In K. Siau and T. Halpin (Eds.), Unified Modeling Language: Systems, Analysis, Design and Development Issues.. Idea Group Publishing , pp. 1-20.

Hotho, A., Maedche, A., Staab, S., Studer, R. (2001) "SEAL-II. The Soft Spot Between Richly Structured and Unstructured Knowledge". Journal of Universal Computer Science, Vol. 7 No. 7, pp. 566-590.

Isakowitz, T., Stohr, E.A., and Balasubramanian, P. (1995) "RMM: A Methodology for Structured Hypermedia Design". Communications of the ACM Vol. 38 No. 8, pp. 34-44.

Java Technology (2006). http://java.sun.com/

Jhingran, A.D., Mattos, N. Pirahesh, H. (2002) "Information Integration: A Research Agenda". IBM Systems Journal Vol. 41 No. 4, pp. 555-562.

Marwick, A.D. (2001) “Knowledge Management Technology”. IBM Systems Journal Vol. 40 No. 4, pp. 814-830.

Navarro, A., Fernandez, B, Fernández-Valmayor, A. and Sierra, J.L. (2004a) “The PlumbingXJ Approach for Fast Prototyping of Web Applications”. Journal of Digital Information Vol. 5 No. 2, special issue on Information Design Models and Process.

Navarro, A., Fernández-Valmayor, A., Fernandez, B., and Sierra, J.L. (2004b) “Conceptualization, Prototyping and Process of Hypermedia Applications”. International Journal of Software Engineering and Knowledge Engineering Vol. 14, No. 6, special issue on Modeling and Development of Multimedia Systems, pp. 565-602.

Navarro, A, Sierra, J.L., Fernández-Valmayor, A. and Fernández-Manjón, B. (2005) “Conceptualization of Navigational Maps for Web Applications”. In Proceedings of The ICWE 2005 Workshop on Model-Driven Web Engineering 2005, Sydney, pp. 80-88.

Novartis Medical Nutrition US (2006), http://www.novartisnutrition.com/us/home

Object Management Group (2004), Unified Modeling Language (UML). Version 2.0. http://www.omg.org/technology/documents/formal/uml.htm

Satyadas, A, Harigopal, U., Cassaigne, N.P. (2001). “Knowledge Management Tutorial: An Editorial Overview”. IEEE Transactions on Systems, Man, and Cybernetics – Part C Vol. 31, No. 4, pp. 429-437.

Schwabe, D., Rossi, G., Barbosa, S.D.J. (1996) “Systematic Hypermedia Application Design with OOHDM”. In Proceedings of Hypertext 96, Washington DC, pp. 116-128.

Struts. The Apache Software Foundation (2006). http://struts.apache.org/

Universidad Complutense de Madrid (2006). http://www.ucm.es/

Virtual Campus of the Universidad Complutense de Madrid (2006). https://www.ucm.es/info/uatd/CVUCM/index.php

Web Course Tools, WebCT (2006). http://www.webct.com/

World Wide Web Consortium (1999). HTML 4.01 Specification. http://www.w3.org/TR/html4/

World Wide Web Consortium (2004). Extensible Markup Language (XML) 1.0 (Third Edition). http://www.w3.org/TR/2004/REC-xml-20040204/