Navarro, A., Fernández-Valmayor, A., Fernández-Manjón, A., Sierra, J.L.
Characterizing Navigation Maps for Web Applications with the NMM Approach.
Science of Computer Programming 71 (1), pp. 1-16, 2008.

Characterizing Navigation Maps for Web Applications with the NMM Approach

A. Navarro *, A. Fernández-Valmayor, B. Fernández-Manjón, J.L. Sierra

Dpto. de Ingeniería del Software e Inteligencia Artificial, Universidad Complutense de Madrid, C/ Profesor José García Santesmases s/n, 28040 Madrid, Spain

Abstract

This paper presents the *Navigation Maps Modeling* approach (NMM), which provides platform independent models for characterizing navigation maps of web applications. The NMM approach is conceived to obtain a trade off between high and low-level design notations. As high-level design notations, NMM models permit architectural details that may hinder the overall understanding of the web application to be left out. As low-level design notations, NMM models can easily be transformed into detailed architectural designs, which are very valuable at coding and maintenance stages.

Key words: model-driven architecture, web engineering, model-driven web engineering, navigation map, multitier architecture, presentation tier

1 Introduction

A web application can be considered as a web system (web server, network, HTTP and browser) in which user input (navigation and data input) affects

⋆ *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.

* Corresponding author

Email addresses: anavarro@sip.ucm.es (A. Navarro), alfredo@sip.ucm.es (A. Fernández-Valmayor), balta@sip.ucm.es (B. Fernández-Manjón), jlsierra@sip.ucm.es (J.L. Sierra).

the state of the business [8]. Despite the apparent ease with which HTML pages are created, the successful development of large web applications is a complex activity that requires appropriate methods and tools [14]. Because the development of these applications is a complex task, modeling support is essential to provide an abstract view of the application. Modeling can help designers during design phases by formally defining the requirements, providing multi-level details as well as providing support for testing prior to implementation. Support from modeling can also be obtained at later phases via, for instance, support for verification prior to implementation [36].

Modelling web applications means characterizing each tier that makes them up [2]: (i) the *client* tier, which represents all device or system clients accessing the system or the application; (ii) the *presentation* tier, which encapsulates all presentation logic required to service the clients that access the system; (iii) the *business* tier, which provides the business services required by the application; (iv) the *integration* tier, which is responsible for communicating with external resources and systems; and (v) the *resource* tier, which contains the business data and external resources.

Our approach focuses on the presentation tier, which includes the navigational structure of the application (i.e. navigation maps), the description of the user interface (its regions and appearance), and the relation between both elements. Although navigation and presentation are presented as independent tiers in web engineering literature [12], in our opinion, from a multitier architecture point of view, the navigation view should be included as a component of the presentation tier [9][10].

Navigation maps describe a global view of a web application for an audience [1]. A navigation map describes the possible sequences of web pages displayed to a user, and is typically part of the documentation of a web application [16]. At present, many web sites include navigation maps to help users during browsing, which makes their characterization a key issue during the development of web applications [26]. Using navigation maps, developers can obtain a global view of the whole application that can help them during the development process. In addition, the presence of navigation maps can help users of web sites to find the desired information much more quickly.

Notwithstanding the importance of modeling, the development of a model is not an easy task. The *Object Management Group* (OMG) [28] has developed the *Model-Driven Architecture* (MDA) [29] approach to guide such a development. MDA promotes the development of software models during the design stage. Thus, the presence of these models leads to systems that are easier to develop, integrate and maintain, and also provides the ability to automate at least some of the construction. MDA starts with the well-known and long established idea of separating the specification of the operation of a system from the details of the way that system uses its platform capabilities [29]. MDA identifies three different models that appear during the development of a system: (i) the *Computation Independent Model* (CIM), focused on the environment and the requirements for the system; the *Platform Independent Model* (PIM), focused on the operation of a system while hiding the details necessary for a particular platform; and (iii) the *Platform Specific Model* (PSM), which combines the platform independent model with an additional focus on the detail of the use of a specific platform by a system [29].

This paper presents the *Navigation Maps Modeling* approach (NMM), which provides platform independent models for navigation maps of web applications. The NMM approach tries to obtain a trade off between the benefits offered by *high-level* and *low-level* design notations. High-level design notations (e.g. UWE [17]) present a significant abstraction level [12][15][23]. Thus, during design stage the models described using these notations characterize the main elements of the web application, hiding architectural details (e.g. the presence of Model 1 or Model 2 architecture). However, these notations do not provide guidelines to obtain detailed architectural designs, which are very valuable at the implementation and maintenance stages. On the contrary, low-level design notations (e.g. UML WAE [8]) permit detailed architectural designs to be characterized. However, due to the presence of architectural details, these designs are tied to specific architectures, and include too many details that may hinder the overall vision of the web application [12][15][23].

As high-level notations, NMM models are independent of the selected architecture (i.e. Model 1 or Model 2). Thus, NMM models omit architectural details enhancing their platform independent role. In addition, simpler applications can benefit from the simplicity of Model 1 architecture [5], while more complex applications can benefit from the flexibility of Model 2 (or *Model-View-Controller*, MVC) architecture [2].

As low-level design notations, NMM models are in tune with a presentation tier totally independent of the rest of the tiers of the web application. This is a key feature in a multitier architecture where a clear separation between business and presentation concerns should be striven for [2][8][10]. In addition, NMM models can be easily translated into UML WAE models. Thus, NMM models can be conceived as high-level versions of *UML Web Application Extension* (UML WAE) models [8]. Moreover, an explicit meaning is provided for NMM notation, which facilitates the transition from platform independent models to platform specific models.

As both types of notations, NMM models provide an independent characterization of navigation and user interface of the web application. Therefore, these components can be changed independently. This feature is present in most design notations [21].

Finally, in navigation maps, to get from one page to another, a request from one page is usually routed through a series of components on the server, ending with the display of the response page [16]. Because NMM models are focused on the presentation tier, it is possible to hide computational artifacts used during the routing of web pages, making it easier to understand these models [26]. Later, in the translation of NMM models to UML WAE models, different computational artifacts can be automatically defined once a specific presentation architecture is selected.

NMM notation is an evolution of the hypermedia notation *Pipe* [25], specifically tailored to characterize navigation maps for web applications. The use of Pipe notation in web engineering projects has demonstrated its applicability as a tool to characterize navigation maps for web applications [26].

The paper is organized as follows. Section 2 describes NMM modeling artifacts. Section 3 describes the explicit meaning of the NMM diagrams in terms of UML WAE class diagrams. Section 4 compares the NMM approach with related work. Finally, Section 5 presents conclusions and future work.

Throughout the paper, the *Virtual Campus of the Universidad Complutense de Madrid* [35] is used as an example. The virtual campus project was started in 2003 and its main objective is to provide students and teachers with all the support that information and communications technologies can provide to improve the quality of learning and research activity at the Universidad Complutense de Madrid (UCM). At present, thousands of users (lecturers and students) use this application. Thus, due to the size of this complex virtual campus, modeling has become a paramount activity [27].

2 NMM modeling artifacts

The NMM approach uses three kinds of diagrams to characterize navigation maps for web applications: (i) *page diagrams*, which characterize the navigational structure of web pages and their links; (ii) *region diagrams*, which model the regions in which the user interface's windows are divided; and (iii) *mixing diagrams*, which relate page diagrams with region diagrams, describing the user's navigational access to pages through the user interface.

The NMM artifacts are formalized for a better characterization of the approach. In addition, this formalization is the basis for the definition of the NMM browsing semantics (similar to the one defined in Pipe [25]) and helps to define node reachability algorithms [11][34]. In particular, NMM formalization is very suitable to define the *links-automaton* of the document [34]. Using this links-automaton and adequate formalisms, it is possible to check if browsing specifications are met by the application [34]. Moreover, NMM artifacts have a visual representation that simplifies their use. For the sake of conciseness, this paper focuses on the visual representation of these artifacts, leaving out several details of the formal components of the approach as well as its browsing semantics.

2.1 Page Diagram

NMM page diagrams provide a characterization of the navigational structure of web pages and their relationships. According to Conallen [8], a web page can basically be anything that can be requested by a browser using HTTP protocol. In NMM, web pages are more similar to the concept of *client page* [8], a web page that is managed (i.e. browsed) by the client. In particular, HTML pages or XML pages with associated style sheets can be understood as NMM pages.

From a navigational point of view, in web applications there are anchors inside these pages. Anchors are endpoints of links, while links are relationships between two anchors [38]. In NMM an anchor represents a device able to start up an HTTP request to a web server. Therefore, HTML anchors, or buttons inside HTML forms can be considered anchors [37]. At present, most of these anchors start up computing processes at the server's side, and therefore they may have some information attached to them (e.g. the data collected in a web form or the identifier of a product). In NMM this information is called the anchor *input*.

NMM supposes the existence of three theoretical sets, which characterize pages, anchors and inputs, and which are used as types to define the NMM approach: (i) *Page*, the *set of all the pages* that can exist in the universe of web applications; (ii) *Anchor*, the *set of all the anchors* that can exist inside the pages of web applications; and (iii) *Input*, the *set of all the inputs* that can exist related to an anchor.

Once these sets are defined and given a web application called A, the page diagram for application A is a tuple $< Page^A, Anchor^A, anchor^A, acc^A >$, where:

- $Page^A \subseteq Page$ is the *set of pages of application* A.
- $Anchor^A \subseteq Anchor$ is the *set of anchors of application* A.
- $anchor^A : Page \rightarrow 2^{Anchor}$ is the *anchoring function of application* A.
- $acc^A : Anchor \times Input \rightarrow Page$ is the *access function of application* A.

The set of pages of application A, $Page^A$, characterizes all the web pages of the application. NMM characterizes two types of pages: (i) lasting pages, $Page_l^A$, which are static pages that exist and are completely defined prior to any user interaction with the application (e.g. an HTML page retrieved by the web server); and (ii) transient pages [1] ($Page_t^A$), which are pages dynamically built by a computational artifact invoked by the web server, and therefore, which temporally exist as responses generated by these computational artifacts (e.g. an HTML page generated by a JSP [20]).

The set of anchors of application A, $Anchor^A$, characterizes all the anchors inside the web pages of the application. In NMM two main types of anchors are characterized: (i) retrieval anchors, $Anchor_r^A$, which give access to lasting pages directly retrieved by the web server without the need for further computing; and (ii) computing anchors, $Anchor_c^A$, which give access to pages provided to the user when the web server delegates to an external computational artifact (e.g. a JSP) that generates the page.

In addition, computing anchors are classified as: (a) form computing anchors, $Anchor_f^A$, which characterize submit buttons of web forms [37]; and (b) non-form computing anchors, $Anchor_{nf}^A$, which characterize computing anchors (i.e. invoke some computational process) different from submit buttons of web forms. At a conceptual level there is no significant difference between these types of anchors. The main difference is that inputs of form anchors are not defined by the designer (or by the application at run-time), while inputs of non-form anchors are predefined by the designer (or by the application at run-time).

The anchoring function of application A, $anchor^A : Page \rightarrow 2^{Anchor}$, is a function that assigns a set of anchors to a page. In this way, if $anchor^A(p) = B$, we say that the anchors of set B are inside of page p.

For the sake of simplicity, in NMM, links are established between source anchors and destination pages [2]. NMM uses the access function that relates anchors with pages to specify this relationship. Due to the presence of computing anchors, the access function is defined on anchors and their input, if it exists. Thus, the access function of application A, $acc^A : Anchor \times Input \rightarrow Page$ is the mechanism used in NMM to relate anchors and inputs with the pages they access. In this way, if $acc^A(a, i) = p$, we say that anchor a with input i gives access to page p.

[1] We have used the term *transient* page instead of the term *dynamic* page, because dynamic pages (or page templates) build these transient pages (or page instances). In addition, to keep a consistent nomenclature, we have used the term *lasting* page instead of *static* page.

[2] Although in practice links can be defined between anchors and anchors, this makes the formalization of the NMM approach more difficult. In addition UML WAE is unable to characterize link targets inside of pages. Thus, in the translation process from NMM to UML WAE this information would be useless.

Because at the design stage the relationship between anchor and pages has to be stated, one of the basic aims of NMM is to specify the access function. To facilitate this task, the definition of the access function is split into two functions: the retrieval and the computing function.

The retrieval function of application A, $ret^A : Anchor_r^A \rightarrow Page_l^A$, characterizes the relations between retrieval anchors and lasting pages. In this way, if $ret^A(a) = p$, we say that retrieval anchor a gives access to the lasting page p.

The computing function of application A, $comp^A : Anchor_c \times Input \rightarrow Page \times AI$, acts on anchor a with input i, and defines the generated response web page and its anchors. This response web page is built by a computational artifact when the web server delegates in it. The set *Anchoring Information*, AI, is a complex set that describes all the types of anchors that can be included in a generated page. There are six sets that make up the AI set. These sets consider the nature of the anchor (retrieval, form computing and non-form computing) and their presence in every page generated, with independence of the input (non-dependent or common anchors), or present in the page due to the specific input (dependent anchors[3]). Thus, if $comp^A(a, i) = (p_i, A_i)$, we say that when the web server delegates in a computational artifact to process the computing anchor a with input i, the computer artifact provides the web page p_i, and the anchors A_i attached to that page.

The NMM approach defines only the signature of the computing function, and every specific representation using the NMM approach for a specific application must provide the actual definition of this function. In this way, the computing function acts as an interface in the object-oriented sense: only the description of the behavior is provided, while the specific behavior of the function has to be defined in every case. The NMM approach uses formal expressions (and their visual representation) to characterize the definition of the computing function. Later, these definitions can be complemented using UML interaction diagrams [30].

Once these functions are defined, the access function is defined as (1) (where $\Pi_1(x, y) = x$).

$$
\begin{aligned}
acc^A : Anchor \times Input &\rightarrow Page \\
(a, null) &\rightarrow ret^A(a), \qquad\qquad \text{if } a \in Anchor_r \\
(a, i) &\rightarrow \Pi_1 comp^A(a, i), \text{ if } a \in Anchor_c
\end{aligned}
\tag{1}
$$

In other words, given an anchor a and an input i, if a is a retrieval anchor (and

[3] These dependent anchors cannot be taken into account in the node reachability algorithms, because they are dependent on specific inputs provided at runtime.

therefore, there is no associated input), acc^A is defined in terms of retrieval function ret^A. If a is a computing anchor, acc^A is defined in terms of computing function $comp^A$. Therefore, function acc^A acts as a *black box* that hides the nature of the relationships between anchors (retrieval or computing) and pages (lasting or transient), providing a uniform view of the navigation in the application. Thus, the definition of a browsing semantics is facilitated. With retrieval anchors, function acc^A will have an extensional definition (in terms of function ret^A). With computing anchors, function acc will have an intensional definition (in terms of function $comp^A$). Therefore, the NMM access function is similar to the table that guides a controller in a MVC architecture, and therefore, it can be used to define such a table. The visual characterization of the modeling components of NMM page diagrams is depicted in Fig. 1.

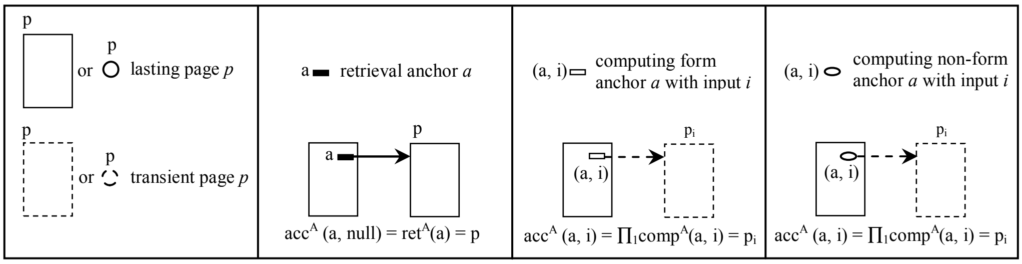

Fig. 1. Graphical notation for the modeling components of NMM page diagrams. Retrieval function is characterized in terms of straight lines, while computing function is characterized in terms of dashed lines.

The distinction between retrieval and computing function is a key issue of the NMM approach. Retrieval function is used in order to assign destination lasting pages to retrieval anchors. Computing function is used in order to assign the definition of generated transient pages (and the different anchors included in them) to computing anchors. The information provided by both functions is used to provide the detailed design of the application at later stages of the development.

In addition, as previously mentioned, in navigation maps, to get from one page to another, a request from one page is usually routed through a series of components on the server, ending with the display of the response page. In the NMM approach, the computing anchors are the devices that permit the components involved in the routing of a page request to be hidden. Thus, NMM computing anchors are associated to the computational components responsible for processing the dynamic request. Later, when the platform independent model evolves towards the platform specific model, NMM computing anchors are the basis for the definition of computational artifacts (e.g. object-oriented classes) responsible for this computing.

NMM page diagrams focus on the characterization of the navigational relationships established among the web pages. Therefore, the characterization of the inner structure of these pages (e.g. the HTML code that makes them up) or the computational artifacts involved in the routing of web pages, are outside the scope of NMM. In any case, this information can be incorporated in the UML WAE diagrams derived from NMM diagrams as outlined in Section 4. In addition, NMM does not provide modeling components to characterize the data model of the application. If needed, as in the case of navigation maps provided by *UML WAE User Experience* diagrams [9], UML class diagrams can be provided [10]. The data of the application are present in NMM page diagrams through computing function. The pages generated by this function include data extracted from the data model of the application. Therefore, these pages are the views of the data model that the computational views of the application (e.g. JSPs) generate for the user. In particular, the definition of the computing function describes the pages that these computational views have to generate. Later, this information can be made explicit using transfer objects [2] (see Section 4 for further details).

Fig. 2 depicts an NMM page diagram in which, a page *upIndex* is linked to pages $facultyMembership$ and *notices*. $facultyMembership$ is linked to page $getDataUCMFaculty$ that contains a form computing anchor ($UCMFaculty$ $Membership$), which collects the data of the faculty members who wish to register in the UCM Virtual Campus. Finally, after membership request is analyzed, it is possible to display a success ($UCMFMembershipOK_i$) or failure ($UCMFMembershipKO_i$) page. This figure encodes the formal elements of the page diagrams. Thus, the access function is encoded according to the visual representation depicted in Fig. 1. Visual notation can represent every component of the formal notation whenever the anchors included in a computed page can be extensionally defined. If an intensional definition is needed (e.g. as in the case of input-dependent anchors), these formal definitions should complement visual diagrams in terms of annotations.

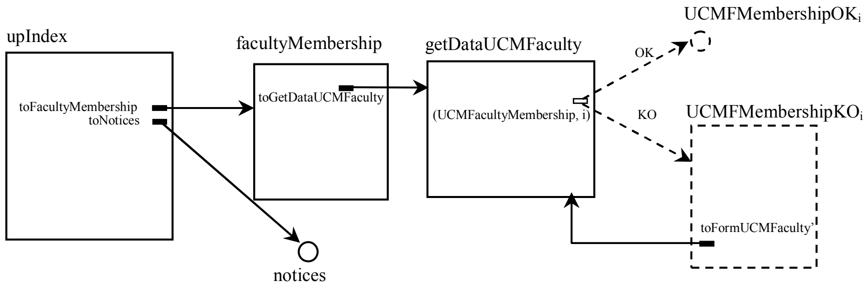

Fig. 2. Page diagram for the elements used in the example of the Virtual Campus.

As in the case of Pipe [25], the use of a CASE tool to generate visual diagrams (and their underlying formal representations) is encouraged in NMM. This CASE tool should be entrusted with: (i) the definition of NMM diagrams and their relationships; (ii) the implementation of node reachability algorithms; (iii) the generation of UML-WAE diagrams from NMM diagrams; (iv) the generation of fast prototypes; and (v) the export of NMM models to other formats (e.g. a format suitable to define the links-automaton [34]). Certainly, the formal specification of NMM notations facilitates the precise specification of such a CASE tool.

2.2 Region Diagram

NMM region diagrams represent the different regions in which the browser window is divided to depict the web pages. Therefore, these diagrams include the definition of regions, windows, and an aggregate relationship that characterizes the regions inside of a window. Several definitions are necessary in order to describe region diagrams.

NMM supposes the existence of two theoretical sets which are used as types to define the NMM approach: (i) $Window$, the *set of all the windows* that can exist in the universe of web applications; and (ii) *Region*, the *set of all the regions* that can exist inside of the windows of web applications.

Once these sets are defined, and given a web application called A, the region diagram for application A is a tuple $< Window^A, Region^A, Agg^A >$ where,

- $Window^A \subseteq Window$ is the set of *windows* of application A. This set represents the windows (e.g. HTML framesets [37]) used by the GUI of a web application.
- $Region^A \subseteq Region$ is the set of *regions* of application A. This set represents the regions (e.g. HTML frames [37]) used by the windows of a web application.
- $Agg^A \subseteq Window^A \times Region^A$ is the set of aggregations between windows and regions. This set represents the aggregation relationship established among windows and regions. Therefore if $(w, r) \in Agg^A$, we say that region r is part of window w.

Fig. 3 depicts the visual characterization of the elements of the notation. The terms *window* and *region* are used instead of terms such as *frameset* and *frame* because, in our opinion, during the development of a platform independent model these details should be omitted. Indeed, during design this simple and abstract conception of the user interface could be refined, provided that the basic interaction behavior is preserved. For example, some designs could decide to use frameset and frames to represent windows and regions, while others could decide to use tables and cells instead. Finally, other designs could decide to aggregate headers and footers to every page of the application omitting the use of frameset/frames.

Fig. 4 depicts a window of the Virtual Campus with its identified regions and the NMM representation of this window. The window is divided into three regions. One on the top of the window (*up* region), another on the left of the

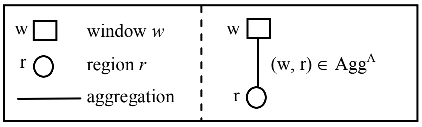

Fig. 3. Graphical notation for the modeling components of NMM region diagrams

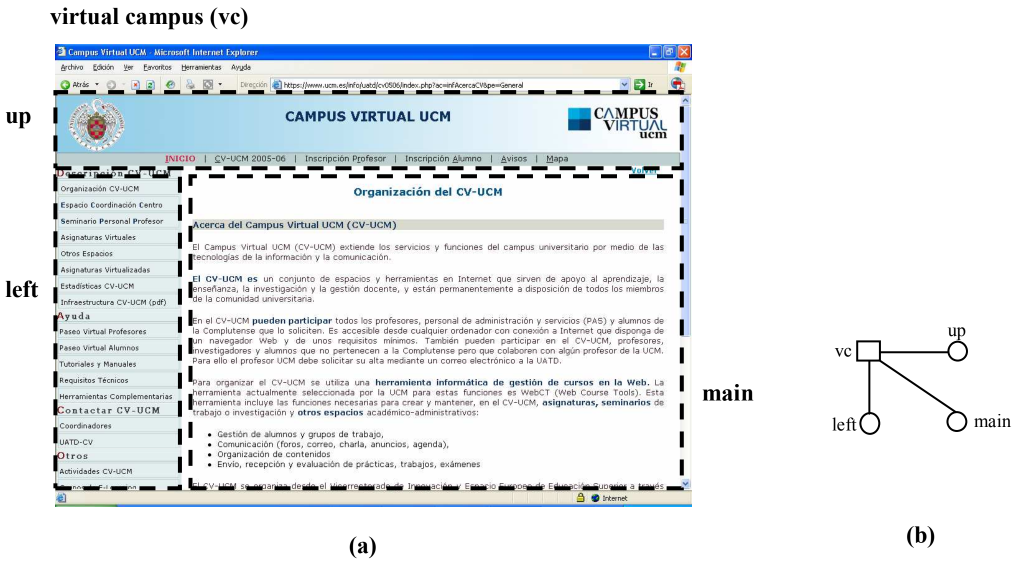

Fig. 4. (a) Window of the Virtual Campus with its regions. (b) NMM region diagram for the Virtual Campus screenshot

window ($left$ region) and another on the center-right of the window ($main$ region).

2.3 Mixing diagram

Mixing diagrams are the third element of NMM, and provide it with most of its flexibility. These diagrams relate page diagrams with region diagrams. Thus, the user's navigational access to pages through the user interface is described. By using mixing diagrams, the same page diagram can be mapped (adapted) to different region diagrams, and the same region diagram can be used with different page diagrams. A *mixing diagram* for application A is a tuple $< def^A, dest^A >$ where,

- $def^A : Region^A \rightarrow Page^A \cup \{blank\}$ is the *default page assignation function* of application A. This function assigns a default page to every region (*blank* page, if there is no default page). Therefore, if $def^A(r) = p$, p is the default page of region r (e.g. the page depicted in the frame when the frameset is accessed).
- $dest^A : Anchor^A \rightarrow Region^A$ is the *destination region function* of application A. This function assigns a region to every anchor. Thus, given an anchor

inside a page depicted in a region and its eventual input, this function offers information about the region where the page accessed by the anchor and its associated input has to be displayed. Therefore, if $dest^A(a) = r$, r is the destination region for anchor a.

Fig. 5 depicts a mixing diagram that relates previous page and region diagrams using colors. According to this diagram $leftIndex$, $upIndex$, and $CVUCM$ are the default pages for the regions $left$, up, and $main$ respectively (note that contents $leftIndex$ and $CVUCM$ were not previously used, but they are included here for the sake of completeness with reality). Because colors relate the anchors (defined at the page diagram level) with the regions, if the user traverses the links established between $upIndex$ and $facultyMembership$ or $notices$, these pages appear in region $main$. Moreover, $main$ is the destination region for the pages accessed by the rest of the anchors. Thus, if the user activates the anchor $toGetDataUCMFaculty$, $getDataUCMFaculty$ will appear in pane $main$ because such an anchor is assigned to region $main$.

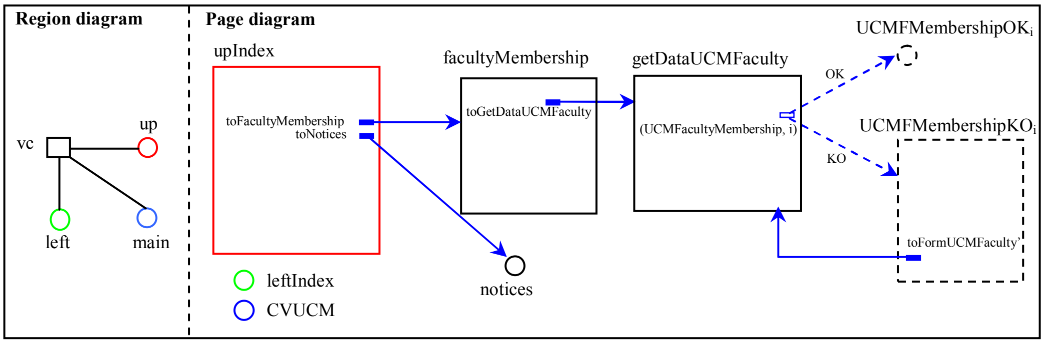

Fig. 5. Mixing diagram for the Virtual Campus

To a certain extent, in region diagrams, destination region function plays the role of the target attribute defined in the anchors in HTML [37]. In this way, there are two well-defined layers that permit the reuse of the same page diagram with different region diagrams, or the same region diagram with different page diagrams. This is a very important feature in prototyping environments where constant changes appear in any component of the application [24]. Moreover, note that the representation provided is abstract enough so as not to impose architecture restrictions at the design stage [26].

If a non frame-based implementation approach is chosen, it is possible to use a modelling style in NMM where the region and mixing diagrams are not necessary. For example, if there are no regions, the anchors of page $upIndex$ can be included in every page depicted in region $main$. In our opinion, this is a bad modelling practice that restricts the final implementation of the application. Therefore, in NMM, the presence of region and mixing diagrams is encouraged with independence of the final implementation. This is the reason why, it is necessary to assign a region to every anchor in NMM. If a frame-based implementation is chosen, all the required information is included in NMM diagrams. Otherwise, it is only necessary to include the anchors of the pages assigned to some regions in the rest of the pages (e.g. the anchors inside of page *upIndex*). Finally, in NMM nested framesets are represented by regions in the platform independent model. These regions can later be translated into nested framesets or into tables nested in cells of tables if desired.

Regarding NMM notation scalability, in our opinion, it is similar to the scalability of other visual notations (e.g. UML WAE). If a large number of pages appear in the diagram, their separation into several subdiagrams may be the best choice [9]. These subdiagrams can be defined using the *contexts* [6], which partition the data within a graph.

3 From NMM diagrams to UML WAE diagrams

UML-Web Application Extension, UML WAE, is a design notation that has found a considerable impact in industry [8][9][10]. However, the explicit presence of computational artifacts, besides the interaction architecture between the presentation and business tiers, reduce the abstraction level of this notation [12][15][23][26].

NMM models can be conceived as a high-level version of UML WAE models where computational artifacts and interaction architecture are hidden. Thus, NMM models can be easily translated into UML WAE models. This translation permits an explicit meaning to NMM notation to be provided. In addition, the transition from platform independent models to platform specific models is facilitated. Note that UML WAE models permit web pages and other architecturally significant elements to be represented in the model alongside the *normal* classes of the model [9]. Thus, although UML WAE models can still be considered as platform independent models, they include all the ingredients to make a smooth transition from platform independent models to platform specific models.

This section depicts the translation of NMM diagrams to UML WAE diagrams that make the computational artifacts of the application and the interaction architecture between presentation and business tiers explicit.

3.1 Page diagram

UML WAE notation considers the principle of *separation of concerns* [8]. According to this principle: (i) web pages executed in the server are UML classes stereotyped with the `server page` stereotype; (ii) web pages presented to

Table 1
Translation from NMM page diagram elements into UML WAE elements

NMM element	**UML WAE element. Model 1**	**UML WAE element. Model 2**
lasting page p	client page p	client page p
transient page p	client page p generated by a server page	client page p generated by a server page
retrieval anchor a	-	-
non-form computing anchor a	-	-
form computing anchor a inside of page p	form a aggregated to client page p	form a aggregated to client page p
link from retrieval anchor a inside of page p_1 to lasting page p_2	link from page p_1 to client page p_2	link from page p_1 to controller + forward from controller to client page p_2
link from non-form computing anchor a inside of page p_1 to transient page p_2	link from page p_1 to server page aSP + operation a in facade + application service aAS + transfers $aInputTransfer$ and $aOutputTransfer$, with dependencies from server page aSP and application service aAS to them + build from server page aSP to client page p_2	link from page p_1 to controller + action $aAction$ + operation a in facade + application service aAS + transfers $aInputTransfer$ and $aOutputTransfer$, with dependencies from $aAction$ and $aApplicationService$ to them + forward from controller to server page $aView$ + dependence from server page $aView$ to transfer $aOutputTransfer$ + build from server page $aView$ to client page p_2
link from form computing anchor a inside of page p_1 to transient page p_2	submit from form a to server page aSP + operation a in facade + application service aAS + transfers $aInputTransfer$ and $aOutputTransfer$, with dependencies from server page aSP and application service aAS to them + build from server page aSP to client page p_2	submit from form a to controller + action $aAction$ + operation a in facade + application service aAS + transfers $aInputTransfer$ and $aOutputTransfer$, with dependencies from $aAction$ and $aApplicationService$ to them + forward from controller to server page $aView$ + dependence from server page $aView$ to transfer $aOutputTransfer$ + build from server page $aView$ to client page p_2

the client are UML classes stereotyped with the `client page` stereotype; and (iii) the navigational relationships among pages is mainly represented using navigated associations stereotyped with the `link` stereotype.

NMM transient pages are translated into UML WAE client pages generated by a server page (e.g. a JSP). NMM lasting pages are translated into UML WAE client pages that exist without needing to be generated. Links defined between NMM anchors and pages are translated into UML WAE links defined between UML WAE pages. These UML WAE links are stereotyped according to the NMM anchors where these links have their origin. Depending on the target architecture, these links are directly established among pages, or are centralized by a controller. Table 1 describes this translation.

The translation depicted in this table, supposes the existence of a facade [13] which centralizes the business logic of the application. Another option is to choose a business delegate instead of this facade [2]. The data flow is represented by transfer objects, whose inner structure depends on the data model of the application [2]. In the translation, dependencies of the facade on these transfers are left out for the sake of clarity in the generated UML WAE diagrams. With independence of the target architecture, every time that an application service [2] is defined, a dependence between the facade and this application service is included. In a Model 1 architecture, every time that a server page is defined, a dependence from the server page to the facade is defined. In a Model 2 architecture, every time that an action [13] is defined, a dependence from the action to the facade is defined. Of course, interfaces and implementations are defined for actions, facade and application services objects. For the sake of conciseness, the elements belonging to the integration tier are left out in this translation.

Therefore, as previously mentioned, in the NMM approach, computing anchors are the devices that permit the components involved in the routing of a page request to be hidden. In this way NMM computing anchors are associated to the computational components responsible for processing the dynamic request. The translation from NMM computing anchors to object-oriented classes depends on the architecture chosen for the web application. For example, the NMM page diagram of Fig. 2 is transformed into the UML WAE diagram of Fig. 6 if a Model 1 architecture is selected.

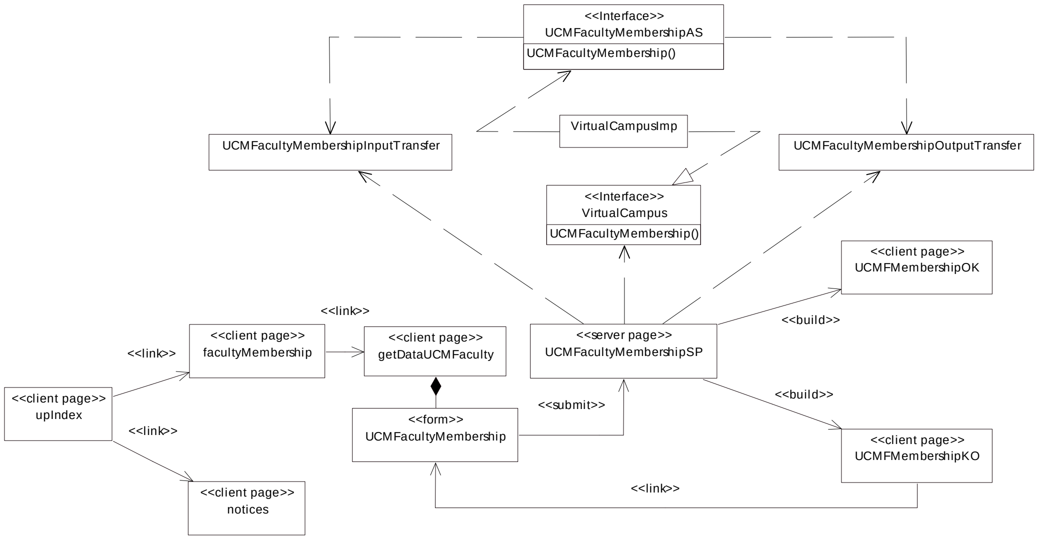

Fig. 6. UML WAE class diagram for the NMM page diagram of Fig. 2 using Model 1 architecture

Note how in Fig. 6, the server page `UCMFacultyMembershipSP` uses the facade `VirtualCampus` that explicitly represents the component responsible for the computational behavior of the membership for UCM faculty (and for other computational behaviors). In particular, this facade uses the application service `UCMFacultyMembershipAS` to implement this functionality. In the NMM diagram of Fig. 2, the server page, the facade and the application service that processes the membership are not present due to the existence of the computing function that directly acts on the form computing anchor $UCMFacultyMembership$. The most important advantage of this approach is that during the development of a platform independent model for the navigation of the application, computational artifacts (i.e. classes) are omitted [26]. In addition, this figure includes a `UCMFacultyMembershipInputTransfer` transfer object used to move information from the input form to the business logic, and a `UCMFacultyMembershipOutputTransfer` transfer object used to move information from the business logic to the presentation tier.

Previous design conforms to the simple page-centric (or Model 1) architecture [5]. If the more complex Model 2 architecture is chosen, the UML class diagram of Fig. 7 is obtained. This diagram becomes more detailed and complex, getting further away from the platform independent model and getting closer to the platform specific model. In a multitier architecture, a controller and a facade (or a business delegate) are components that are always present. Therefore, regarding the presentation tier, the client and server pages that conform the navigational map are the target elements to be described. In addition, in this architecture a class responsible for the computational behavior of the application, input and output transfers of this class, and input and output views have to be defined. NMM permits these components to be derived from the structure of lasting and transient pages and the anchors they include. Thus, from the presence of the NMM computing anchor $UCMFacultyMembership$ inside of the lasting page $getDataUCMFaculty$ that links with the transient pages $UCMFMembershipOK_i$ and $UCMFMembershipKO_i$ shown in Fig. 2, the presence of the following is derived: the computational class `UCMFaculty MembershipAS`, an input (`UCMFacultyMembershipInputTransfer`) and an output (`UCMFacultyMembershipOutputTransfer`) transfer, and an input (`getData UCMFaculty`) and output (`UCMFacultyMembershipView`) view. Of course, the computational behavior of the `UCMFacultyMembershipAS` class is outside the scope of the presentation tier. Regarding the behavior of the view `UCMFaculty MembershipView`, UML interaction diagrams can be provided. Therefore, NMM page diagrams are simpler than UML WAE diagrams because they have fewer elements and do not include information about processing [26]. In addition, note also that NMM page diagrams are architecture-independent, i.e., the same NMM page diagram can be mapped into a Model 1 or Model 2 UML WAE class diagram.

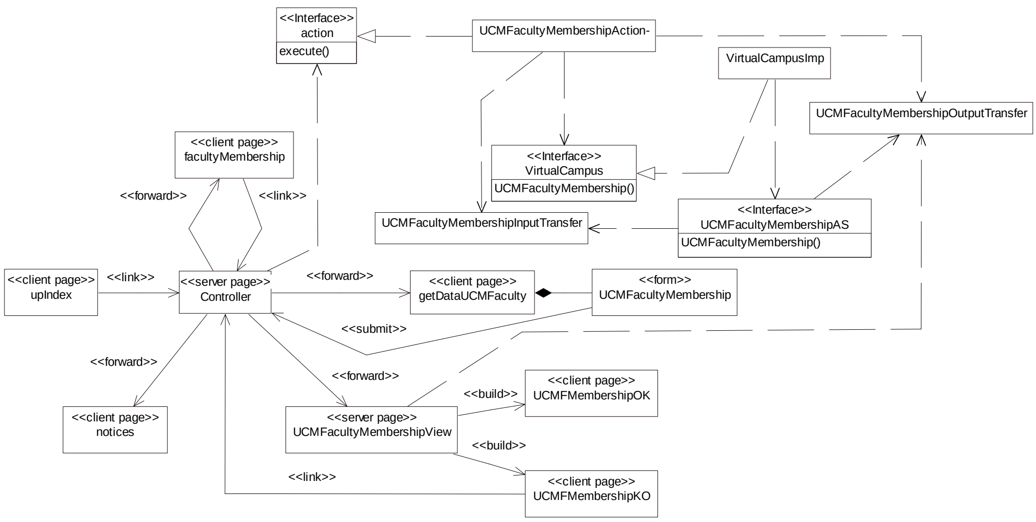

Fig. 7. UML WAE class diagram for the NMM page diagram of Fig. 2 using Model 2 architecture

Finally, no information regarding the underlying data model appears in the

Table 2
Translation from NMM region diagram elements into UML WAE elements

NMM element	**UML WAE element. Model 1 or Model 2**
window w	frameset w
region r	frame target r
connection from window w to region r	aggregation from frameset w to frame target r

diagrams of Fig. 6 and Fig. 7. This is because, as previously mentioned, NMM diagrams do not characterize the data model of the application. Only transfer objects are depicted. This is a view that is consistent with a multitier architecture [2], where the independence between tiers is paramount.

3.2 Region diagram

Regarding the user interface, UML WAE uses `frameset` and `target` stereotyped classes to represent these HTML *framesets* and *frames* that are the target of a link.

In this case, NMM windows are translated into UML WAE `frameset` stereotyped classes and NMM regions are translated into UML WAE `target` stereotyped classes. The NMM connection relationship is translated into a UML WAE aggregation relationship. This translation is valid, with independence of the target architecture. Table 2 depicts this simple translation.

Therefore, the region diagram of Fig. 4 can be translated into the UML WAE diagram as depicted in Fig. 8.

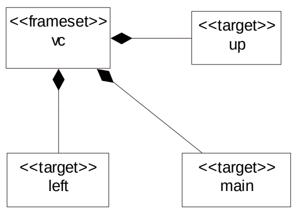

Fig. 8. UML WAE class diagram for the NMM region diagram of Fig. 4

If a non-frameset approach is chosen (see discussion in case study of page diagrams), the client pages `upIndex` and `leftIndex` should be aggregated to every client page (except to themselves, of course) and the diagram of Fig. 8 could be left out.

3.3 Mixing diagram

Finally, the NMM default page assignation function is used to assign default pages to the frames in UML WAE class diagrams. The NMM region destination function is used to decorate the UML WAE navigated associations with

Navarro, A., Fernández-Valmayor, A., Fernández-Manjón, A., Sierra, J.L.
Characterizing Navigation Maps for Web Applications with the NMM Approach.
Science of Computer Programming 71 (1), pp. 1-16, 2008.

Table 3
Translation from NMM mixing diagram elements into UML WAE elements

NMM element	**UML WAE element. Model 1 or Model 2**
default page assignation from region r to page p	aggregation from frame target r to page p
destination region assignation from anchor a to region r	constraint $\{target = r\}$ in the UML WAE link (or submit) in which the NMM link with origin in anchor a is translated into

the stereotype `target`. This translation is valid, with independence of the target architecture. Table 3 depicts this simple translation.

Fig. 9 depicts the UML WAE version of Fig. 5.

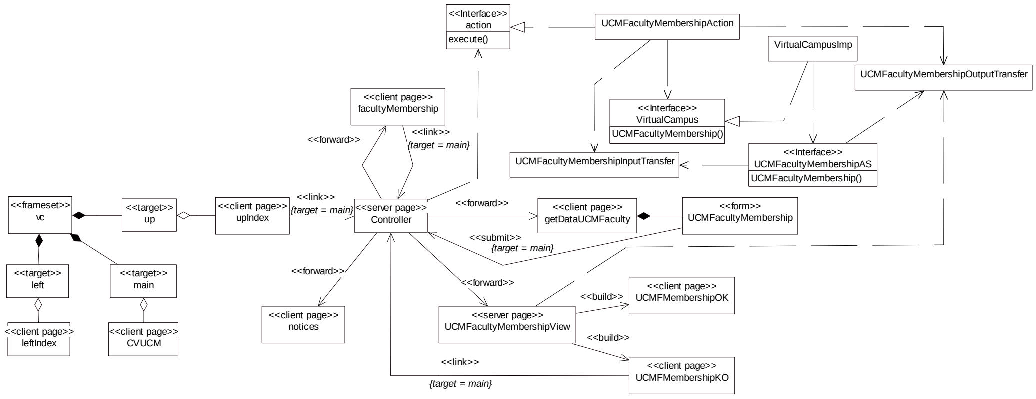

Fig. 9. UML WAE class diagram for the NMM mixing diagram of Fig. 5

4 Related work

At present, there is a plethora of design notations in the web engineering domain. *Object-Oriented hypermedia* (OO-H) [15], *Object-oriented Hypertext Design Model* (OOHDM) [32], *Relationship Management Model* (RMM) [19], *UML-based Web Engineering* (UWE) [17], and *Web Modeling Language* (WebML) [7] are some of the most relevant design notations in the hypermedia domain.

These notations can be classified as high-level notations. They consider three major design dimensions during the design of a web application [12]: (i) the *structural* dimension characterizes the data model of the web application in terms of classes or entities and their relations; (ii) the *navigational* dimension characterizes the navigation throughout the application content; and (iii) the *presentation* dimension describes the way in which application content and navigation are presented to the user. In addition, at present, most design notations have included some extensions to explicitly represent the business logic of the application [4][22][31].

These notations provide modeling primitives to characterize the structural model of the application (i.e. the data model and the business logic of web applications). In addition, these notations provide a navigational model of the application, which is built on its structural model. This navigational model has two components: (i) navigational classes, which are the perceptible representation of the conceptual/structural classes and their business processes; and (ii) a navigational architecture model that characterizes the navigational relationships (i.e. the navigation map) between these navigational classes. Thus, OO-H uses *navigation access diagrams*, OOHDM uses *navigational context schema*, RMM uses *RMDM diagrams*, UWE uses *navigation diagrams*, and WebML uses *navigation specifications*.

These navigational models make references to navigational classes and their relationships (which are in fact built on the classes and relationships of the data model) and to the business logic of the application. The presence of this business logic varies in each approach: OO-H uses links with fragments of code as well as object-oriented classes with methods, OOHDM uses *activity nodes*, UWE uses *navigational process classes*, and WebML uses *activities* that make up a process.

Regarding user interface (presentation dimension), these notations provide modeling components to provide an in-depth characterization of the user interface and its relationships with the navigational model.

Finally, these notations provide high-level characterizations of web applications, and most of them include CASE tools, but they do not provide explicit mechanisms to obtain detailed architectural designs or platform specific models of the application.

As in [2], NMM relies on UML (without extensions) to characterize resource, integration and business tiers (and therefore, the structural dimension). Regarding the presentation tier (including the navigation dimension), the NMM navigational map is represented in terms of the pages perceived by the user, and is not built over the data model. This is a consistent characterization of the presentation tier of the user interface in a multitier architecture, where the resource tier is represented by *transfer* objects [2], and the business tier is represented by a facade or by a business delegate object [2]. In addition, in NMM these computational artifacts are hidden. Later, if a UML WAE model is generated, these computational artifacts, derived from the NMM computing anchors, are made explicit. Regarding the rest of the components of the presentation tier (i.e. the presentation dimension), NMM is only focused on the regions of the user interface and their relations with the navigation maps. In contrast, the previously mentioned design notations provide more in-depth descriptions of the components of the user interface.

UML WAE defines *navigational maps* as a part of the *User Experience Model* (UX) [9]. These maps are defined in terms of *screen*s and their links. In UML

WAE UX screens are a mixture of page information, links between these pages, and the frameset structure in which pages are being displayed. As Conallen defines: "A screen is something that is presented to the user. It contains the standard user interface structure, such as menus and controls, as well as business-relevant content" [9]. In our opinion, the definition of UX screen and their links is not so clear as the definition of UML WAE class diagrams. In particular, this mixture of user interface and business-data contents makes the definition of UX navigational maps and their translation to UML WAE class diagrams more difficult, i.e. there are no systematic rules to translate UX diagrams into UML WAE diagrams. In contrast, NMM notation independently defines the page diagram from the region diagram. In our opinion: (i) this leads to clearer models; (ii) page diagrams and region diagrams can be changed without interferences; and (iii) the transition to UML WAE class diagrams can be made more systematically.

Regarding UML WAE class diagrams, the use of stereotyped classes permits the presence of pages (e.g. the client page `upIndex` of Fig. 7) that can be used to characterize navigation maps for Web applications. Although UML WAE notation is able to depict Model 1 or Model 2 architectures, it is necessary to fix the architecture in order to define the model of a specific application. In other words, models described in terms of UML WAE notation are not architecture-independent. In addition UML WAE models make the presence of computational artifacts (e.g. the class `UCMFacultyMembershipAS` of Fig. 7) explicit, which makes them more difficult to understand [26], and lowers the abstraction level of the notation [23]. Finally, note that in a multitier architecture, the WAE extension is only used at the presentation tier.

Dialog Flow Notation (DFN) [3] and *State WebCharts* (SWC) [36] are focused on the characterization of the navigation in Web applications. DFN represents the dialog flow within an application as a directed graph of states connected by transitions, and SWC uses statewebcharts to describe the navigation between documents. To some extent, both approaches characterize the UML WAE separation of concerns. DFN uses *masks* and *actions* while SWC uses *static*, *transient* and *dynamic* states. Therefore, like UML WAE, these approaches do not hide computational artifacts. In NMM the separation among client and server pages is not considered because the computing processing is hidden behind the access function. In addition, DFN and SWC use the same diagram to characterize the navigation through the pages and their user interface, while in NMM these items are separated using page and region diagrams. Finally, DFN enforces the presence of a MVC architecture, while NMM is independent of the final architecture [26].

Regarding Pipe, NMM can be understood as a web specialization of Pipe. Therefore, the general set of dynamic anchors of Pipe [27] is specialized with the set of NMM computing anchors (form and non-form computing anchors).

Table 4
Comparison between NMM and the related work

Approach	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
DFN	×	×	×	×	×	×	×
NMM	√	√	√	√	√	×	×
OO-H	×	√	√	×	×	√	×
OOHDM	×	√	√	×	×	√	×
Pipe	√	√	√	√	√|×	×	×
RMM	×	√	√|×	×	×	×	×
SWC	×	√	×	×	×	×	×
UML WAE	√	×	√	×	√	√	√
UWE	×	√	√	×	×	√	×
UX	√	√	×	√	√|×	√	×
WebML	×	√	√	×	×	√	×

(i) independent characterization of the presentation tier
(ii) independence of the interaction architecture between presentation and business tier
(iii) independent characterization of navigation structure and regions of the user interface
(iv) hiding of computational artifacts
(v) translation to UML WAE
(vi) in-depth characterization of the user interface
(vii) detailed characterization of every component of every tier

The Pipe navigational relationships established at the user interface level (i.e. the *pipes*) are omitted in NMM, because in web applications the timing and navigational relationships that may appear in hypermedia applications (e.g. n-ary links) do not appear. Therefore, Pipe functions that relate contents (NMM pages) with user interface are significantly simplified in NMM. In addition, this paper does not provide an ad hoc browsing semantics for NMM models. Instead, the meaning of the NMM is provided in terms of UML WAE models.

Table 4 shows these works considering the seven characteristics aforementioned: (i) independent characterization of the presentation tier; (ii) independence of the interaction architecture between presentation and business tier; (iii) independent characterization of navigation structure and regions of the user interface; (iv) hiding of computational artifacts; (v) translation to UML WAE; (vi) in-depth characterization of the user interface; (vii) detailed characterization of every component of every tier. This last item depicts the unmatched ability of UML WAE to explicitly characterize every element of every tier of web applications, with independence of the architecture selected (unlike DFN). In some applications, or during the provision of platform specific models, the explicit presence of these elements can be particularly interesting. Table 4 takes into account the translation from the related approaches to UML WAE. The translation to UML itself (without the WAE extension) it is not considered, because in a multitier architecture, UML WAE it is the best choice to characterize the presentation tier in an UML design.

Despite Table 4, Pipe is not specially tailored to characterize navigation maps. Our work [26] provides a web navigation map using Pipe, and sketches the translation of Pipe models into UML WAE models. In any case, Pipe: (i) does not provide specific modeling artifacts to characterize web navigation maps;

and (ii) includes several modeling artifacts that cannot be translated into UML WAE. In particular, this paper provides the adaptation of Pipe into NMM, which overcomes these drawbacks and simplifies Pipe modeling artifacts. That is the reason why "$\surd|\times$" appears in item (v). Moreover, note that RMM does not provide modeling components for the user interface. Therefore "$\times$" appears in item (i), and "$\surd|\times$" appears in item (iii).

Finally, there are another group of models which relies on well-known formal specifications, such as statecharts [11] or Petri Nets [34], in order to define hypertexts. Using these formal specifications several advantages can be gained, such as checking node reachability, specifying synchronization of simultaneous displays, specifying access control, defining tailored versions [11] or defining dynamic adaptation [33]. NMM uses its own ad-hoc formal definition. Thus, this definition has to be understood before applying the NMM approach. Moreover, this formal definition cannot rely on an underlying formal structure in order to automatically carry out model checking, although it can be very valuable in the definition of checking algorithms (whenever temporal relationships and input-dependent anchors are excluded from the requirements). These algorithms use a graph-based translation of NMM models (similar to the Pipe graphs [25]) in order to perform their function. However, NMM is not specially suited to describe adaptive behaviors. In addition, in our opinion, NMM models can fit reasonably well in the view of hyperdocuments as automata [34]. In this view, model checking can be used to verify that browsing specifications are met by the behavior defined by the automaton view of the hyperdocument (the *links-automaton*).

5 Conclusions and future work

The characterization of navigation maps is a key issue during the development of Web applications. This paper presents the NMM approach, which provides platform independent models for navigation maps of web applications.

NMM notation is conceived to obtain a trade off between the good characteristics of high-level and low-level design notations. Thus, as high-level design notations, NMM models are independent of the selected architecture (i.e. Model 1 or Model 2). As low-level design notations, NMM models are in tune with a presentation tier totally independent of the rest of tiers of the web application, and can be easily translated into UML WAE models. As both types of notations, NMM models provide an independent characterization of navigation and user interface of the web application. Finally, because NMM models are mainly focused on presentation tier, computational artifacts used during the routing of web pages can be hidden. Later, in the translation of NMM models to UML WAE models, different computational artifacts can be automatically defined once a specific presentation architecture is selected.

Throughout the paper, the Virtual Campus of the Universidad Complutense de Madrid is used as a case study. Although the examples provided in this paper are deliberately simple, this web application is a complex system used by thousands of users. At a first step, UML WAE models were built, but to obtain high-level versions of these diagrams that facilitate the navigation at the website, NMM visual diagrams are under development. Thus, we are adapting our CASE tool that supports the Pipe notation [25] to use it with NMM notation. Moreover, our aim is to define NMM notation in terms of a *UML profile* [30] and to define transformation rules between this profile and the UML WAE profile. Note that this approach allows the use of general purpose CASE tools (e.g. [18]), instead of ad hoc tools (e.g. [25]). In addition, these CASE tools are able to support a full model-driven architecture approach. Thus, once suitable transformation rules were defined, platform specific models in terms of J2EE components (for example) could be automatically obtained. Finally, the translation of NMM models to links-automaton should be further analyzed in order allow the automatic verification of browsing specifications.

References

[1] S.M. Abrahão, L. Olsina, and O. Pastor, Towards the Quality Evaluation of Functional Aspects of Operative Web Applications, in: Proc. IWCMQ 2002, Lecture Notes in Computer Science, Vol. 2784 (Springer, Berlin, 2002) 325-338.

[2] D. Alur, J. Crupi and D. Malks, Core J2EE Patterns. Best Practices and Design Strategies, 2nd edition (Sun Microsystems Press - Prentice Hall, Upper Saddle River, 2003).

[3] M. Book, and V. Gruhn, Modeling Web-Based Dialog Flows for Automatic Dialog Control, in: Proc. ASE 2004, (IEEE Computer Society, 2004) 100-109.

[4] M. Brambilla, S. Ceri, P. Fraternali and I. Manolescu, Process Modeling in Web Applications, ACM Transactions on Software Engineering and Methodology 15 (2006) 360-409.

[5] S. Brown et. all, Professional JSP, 2nd edition (Wrox Press LTD, Birmingham, 2001).

[6] B. Campbell and J.M. Goodman, HAM: A General Purpose Hypertext Abstract Machine, Communications of the ACM 31 (1988) 856-861.

[7] S. Ceri, P. Fraternali and A. Bongio, Web Modeling Language (WebML): a Modeling Language for Designing Web Sites, Computer Networks 33 (2000) 137-157.

[8] J. Conallen, Modeling Web Application Architectures with UML, Comm. of the ACM 42 (1999) 63-70.

[9] J. Conallen, Building Web Applications with UML, 2nd Edition (Addison-Wesley Professional, Boston, 2002).

[10] P. Eeles, K. Houston and W. Kozaczynski, Building J2EE Applications with the Rational Unified Process (Addisson-Wesley, Boston, 2003).

[11] M.C. Ferreira, M.A. Santos and P.C. Masiero, A Statechart-Based Model for Hypermedia Applications, ACM Transactions on Information Systems 19 (2001) 28-52.

[12] P. Fraternali, Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, ACM Computing Surveys 31 (1999) 227-263.

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

[14] A. Ginige and S. Murugesan, Guest Editors' Introduction: The Essence of Web Engineering-Managing the Diversity and Complexity of Web Application Development, IEEE MultiMedia 8 (2001) 22-25.

[15] J. Gómez, C. Cachero and O. Pastor, Extending a Conceptual Modelling Approach to Web Application Design, in: Proc. CAiSE 2000, Lecture Notes in Computer Science, Vol. 1789 (Springer, Berlin, 2000) 79-93.

[16] M. Han and C. Hofmeister, Separation of Navigation Routing Code in J2EE Web Applications, in: Proc. ICWE 2005 (Springer, Berlin, 2005) 221-231.

[17] R. Hennicker and N. Koch, Systematic Design of Web Applications with UML, in: K. Sian and T. Halpin, ed., Unified Modeling Language: System Analysis, Design and Development Issues (Idea Group Publishing, 2001).

[18] IBM Rational Software Architect, http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html

[19] T. Isakowitz, E.A Stohr and P. Balasubramanian, RMM: a Methodology of Structured Hypermedia Design, Communications of the ACM 38 (1995) 34-43.

[20] Java Technology. Java Server Pages Technology. http://java.sun.com/products/jsp/

[21] N. Koch and A. Kraus, Towards a Common Metamodel for the Development of Web Applications, in: Proc. ICWE 2003, Lecture Notes in Computer Science, Vol 2722 (Springer, Berlin, 2003) 497-506.

[22] N. Koch, A. Kraus, C. Cachero and S. Meliá, Integration of Business Processes in Web Application Models, Journal of Web Engineering 3 (2004) 22-49.

[23] D. Lowe, Web System Requirements: an Overview, Requirements Engineering 8 (2003) 102-113.

[24] J. Nanard and M. Nanard, Hypertext Design Environments and the Hypertext Design Process, Communications of the ACM 38 (1995) 49-56.

[25] A. Navarro, A. Fernández-Valmayor, B. Fernández-Manjón and J.L. Sierra, Conceptualization, Prototyping and Process of Hypermedia Applications, International Journal of Software Engineering and Knowledge Engineering 14 (2004) 565-602.

[26] A. Navarro, J.L. Sierra, A. Fernández-Valmayor and B. Fernández-Manjón, Conceptualization of Navigational Maps for Web Applications, in: Proc. MDWE 2005 (University of Wollongong, Sydney, 2005) 80 - 88.

[27] A. Navarro and A. Fernández-Valmayor, Conceptualization of Hybrid Websites, Internet Research 17 (2007) 207-228.

[28] Object Management Group website, http://www.omg.org

[29] Object Management Group, Model-Driven Architecture (MDA), http://www.omg.org/mda

[30] Object Management Group, Unified Modeling Language (UML), Version 2.1.1, http://www.uml.org, 2007.

[31] G. Rossi, H.A. Schmid and F. Lyardet, Customizing Business Processes in Web Applications, in: Proc. EC-Web 2003, Lecture Notes in Computer Science, Vol. 2738 (Springer, Berlin, 2003) 359-368.

[32] D. Schwabe, L. Esmeraldo, G. Rossi and F. Lyardet, Engineering Web Applications for Reuse, IEEE MultiMedia 8 (2001) 20-31.

[33] P.D. Stotts and R. Furuta, Dynamic Adaptation of Hypertext Structure, in: Proc. Hypertext 91 (ACM, 1991).

[34] P.D. Stotts, R. Furuta and C. Ruiz, Hyperdocuments as Automata: Verification of Trace-Based Browsing Properties by Model Checking, ACM Transactions on Information Systems 16 (1998) 1-30.

[35] Virtual Campus of the Universidad Complutense de Madrid, https://www.ucm.es/info/uatd/CVUCM/index.php

[36] M. Winckler and P. Palanque, StateWebCharts: a Formal Description Technique Dedicated to Navigation Modelling of Web Applications, in: Proc. DSV-IS 2003 (Springer, Berlin, 2003) 61-76.

[37] World Wide Web Consortium, HTML 4.01 Specification, 1999, http://www.w3.org/TR/html4/

[38] World Wide Web Consortium, Hypertext Terms, http://www.w3.org/Terms.html