CLEI ELECTRONIC JOURNAL VOLUME 14 NUMBER 1 PAPER 1 APRIL 2011

A MDA Approach for Navigational and User Perspectives

Magalí González
Catholic University "Ntra. Sra. de la Asunción" - Paraguay
DSIC - Politecnic University of Valencia – Spain
mgonzalez@uca.edu.py

Jorge Casariego
Catholic University "Ntra. Sra. de la Asunción" - Paraguay
jorgekasa@gmail.com

Juan José Bareiro
Catholic University "Ntra. Sra. de la Asunción" - Paraguay
juan_bareiro@uca.edu.py

Luca Cernuzzi
Catholic University "Ntra. Sra. de la Asunción" - Paraguay
lcernuzz@uca.edu.py

Oscar Pastor
DSIC – Politecnic University of Valencia - Spain
opastor@dsic.upv.es

Abstract. The study presented in this paper focuses on a navigational and user perspectives analysis of Web applications, highlighting some of their critical points for modeling. To consider these critical points, we propose a notational definition focused on a general metamodel, consisting of three specific metamodels: the Navigational Tree, the Node, and Roles. The metamodels have been implemented with a specific MDA (Model Driven Architecture) tool called AndroMDA, to generate applications for PHP language. Both, the general metamodel and its implementation have been analyzed by means of a proof of concept. Finally, some interesting results as well as the contribution of the proposed metamodel in comparison with other methodological proposals are discussed.

Keywords: Model Driven Architecture, Navigational and User Perspectives in Web Applications, Metamodel for Navigation and User.

1 Introduction and Motivation

Software development techniques are continuously evolving with the goal of solving the main problems that still affect the construction and maintenance of software systems: mainly time and cost.

During the last years the Web Engineering community has proposed various languages, architectures, methods and processes for Web Applications development. In particular, a variety of methods for modeling such systems have been defined. Amongst others, we can mention Hera [13], OOHDM [17], OO-H [3], OOWS [8], UWE [14], WebML [5], and W2000 [2]. These methods mainly focus on the analysis and design specification for Web systems, considering the most relevant aspects of this kind of systems. Two relevant aspects are those that refer to "Navigational" and "User" perspectives.

Navigation has been highlighted to be a characteristic at the same time fundamental and critic within Web Engineering [10][17]. Nevertheless, navigational models in some situations do not provide of an appropriate syntax to model certain common behavior of current Web Systems, such as dynamic navigation obtained by the users' behavior.

Speaking in terms of user modeling, current needs focus on the identification of goals, preferences and knowledge of each user separately, jointly with navigation and presentation modeling. Starting from these aspects, user modeling is later used to personalize both the content (presentation) as well as the environment in which it unfolds (navigation). It should be noted that at the user modeling level some current proposals provide support for the personalization based on roles and adaptation, such as OOHDM [17], WSDM [6], W3I3 [5] or UWE [14].

This work intends to define a navigational and user model that intends to facilitate the development of personalisable Web applications with simple and complete navigational structures. Actual proposals define navigational structures, but we consider they are not always as complete as they should be (as explained in section 2.2). To validate the proposal, transformation rules were defined applying the MDA (Model Driven Architecture) approach [16], and a relevant proof of concept was carried out to show the way in which the proposal faces common problems to various Web application domains.

The rest of the paper is organized as follows: Section 2 introduces the Navigational and Users perspectives in Web applications, and analyzes the way in which such perspectives are covered in current proposals, concluding with the identification of critical aspects; Section 3 presents the notational proposal for modeling, the definition of the proposal using MOF (Meta Object Facility), and the definition of transformation rules with a MDA tool; Section 4 analyzes the results of the proposal's application to the proof of concept; and Section 5 concludes and points out some possible future works.

2 The Navigational and Users Perspectives in Web Applications

This section discusses definitions about the navigational and users perspectives. Subsequently, such perspectives are analyzed according to different methodological proposals, to identify the critical aspects that must be considered to create a formal definition of them.

2.1 The Concept of Navigation and its Relationship with the User Perspective

While the navigation began to be considered as a fundamental aspect in the methodological proposals for the hypermedia applications [9], it acquired a fundamental role in the methodological proposals for the design and development of Web Applications [13], [17], [3], [8], [14], [5], [2] and, consequently, each methodology defines a specific diagram for modeling it. Despite that many authors propose different definitions [7], [4], [17], a standard definition of what "navigation" means does not exist. Hereinafter, some of the most used definitions:

- Navigation is considered the navigation space model that specifies which objects can be visited through the Web application [14].
- Navigation is the cognitive process of acquiring knowledge about a space, strategies to move through space, and change the meta-space [18].
- Navigation expresses how pages and content units are linked to form hyperspaces [5].
- Navigation is the way a node is reached via a link. Navigation is the most important feature of hypermedia [19].

It can be noted in the different definitions that the navigation introduces the concept of "space" to be explored in some way through a set of nodes that conforms it. However, the granularity or definition of what represents each "navigable" node and the way of "exploring" them varies in each definition. Besides, to be able to explore the "space", the existence of a "user" or "external agent" that activates the exploration is necessary.

In the context of this work, a "Navigable Node" is defined as a functional unit of the system, and the Navigation is defined as "the change from a navigational node to another as a result of an invocation from the user or an external agent". Therefore, navigation occurs when an external agent (be it a user, an external system, or any other resource) interacts through the invocation of a "navigational node".

It is possible to note that the user fulfills a key role, since it is responsible for invoking navigation over the system. In this regard, user modeling corresponds to the process of "constructing, maintaining and using user models to store specific information about a user, and so be able to customize both the content as the environment in which it unfolds" [15]. Finally, user modeling must contemplate aspects that identify its interests, relationships with other users, knowledge, etc.

2.2 Modeling Navigational and User Perspectives: Relevant Aspects

In order to find a solution for the issues identified, an analysis of Web methods has been performed. A large numbers of methods exist; among others we can mention: OOHDM [17], WSDM [6], W3I3 [5], UWE [14].

It should be noted that all methods consider navigational and user perspectives in some way. Both aspects are usually covered by means of specific models, usually considering navigation user needs as well as presentation of the system.

From the analysis of different methods and our experience in the development of real cases, it was possible to identify a number of elements considered of paramount importance when modeling aspects of navigation and users of a system. First, we identified the need of an order in the navigation map where a well organized and structured hierarchy to facilitate user orientation exists. In this sense, the current proposals usually define notations of the navigational structure that can become very fuzzy for complex systems. A second aspect is that on several circumstances different types of navigation exist: inter-contextual navigation, widely known through conventional navigation; and intra-contextual navigation, where user navigates within the same context. An example of inter-contextual navigation is the access to new functionality (e.g. how to proceed with purchase of a product in a on-line shop). An example of intra-contextual navigation could be the culmination of the purchasing process in which a series of intermediate steps exist that require some navigation within the same context to complete the purchase. This distinction, in most cases is not given by current methods, generating complex and difficult to understand navigational models. A third critical aspect is that navigation is in response to certain behavior by a foreign agent, and can produce action invocation, execution of services, or change of context, and we believe that using behavioral rather than static models could simplify the modeling task.

Regarding the modeling of users, we consider a necessity to represent differentiated users. In this sense, as a Web system usually represents a multi-user environment where multiple users have different permissions and access privileges, it is essential to differentiate roles that diverge in capabilities. In addition, it is important to represent users with more than one role and identify individual users and not as a group, with the purpose of improving both the navigation and the presentation according to the needs of the project.

3 A MDA Proposal for Navigation and User Management

This section introduces the MDA approach for the definition of a proposal for Navigation and User management in Web Applications. First, we present the proposed notation for modeling the PIM (Platform Independent Model) using UML as a notational language. Second, we present the proposed formal definition with MOF (Meta Object Facility). The section ends with the definition of transformation rules for the generation of PSMs (Platform Specific Model) and an ISM (Implementation Specific Model) using an open source MDA tool (AndroMDA) [1].

To clarify the proposal, an example that presents the critical elements identified in section 2.2 is considered. The example is a Web-based Academic System for the Catholic University of Asunción. The system is aimed at teachers, students, staff and public in general and covers a range of basic functions such as: processing student registrations, course monitoring, and management of the different schools, departments and careers. Teachers have sufficient privileges to manage their courses providing also students with information regarding their current status. Students have the required privileges to track the courses they attend and can also access their current academic status. Finally, the system provides the facility to perform administrative tasks such as management of the faculties, courses, departments, and subjects.

3.1 The Navigational Model

To model the navigation concept considering the definition given in Section 2.1, we define two diagrams: *Navigational Tree Diagram* and *Navigational Node Diagram*.

The Navigational Tree represents the application's navigational space and is composed of zero or more navigational elements that define a hierarchical structure. These elements may be *nodes* or *links*. A navigational node corresponds to a basic functional unit of the system, and connects to other nodes by means of elements called *hard links* (hLinks). Hard links, connect two nodes, and denote a hierarchy in the navigational tree, indicating that the origin context node contains the context of the destination node.

The Navigational Tree is defined with notational elements of the Use Cases UML Diagram. In this sense, a Use Case with stereotype <<node>> represent nodes, and associations between use cases stereotyped with <<hLink>> represent hard links. An example of a navigational tree is shown in figure 1. This diagram presents the navigational structure of the Web Academic System. The tree root corresponds to the home page of the system, and from it the system functionalities are hierarchically organized: Faculty Management, Login, etc.

It should be noted that while the vast majority of Web Methods propose navigational maps (generic graphs), this proposal seeks to define the navigational structure as a hierarchy of functional units connected by hard links. With this contribution it is possible to generate various levels of exploration through menus and submenus, and keep the user located through "breadcrumbs" and "history of navigation".

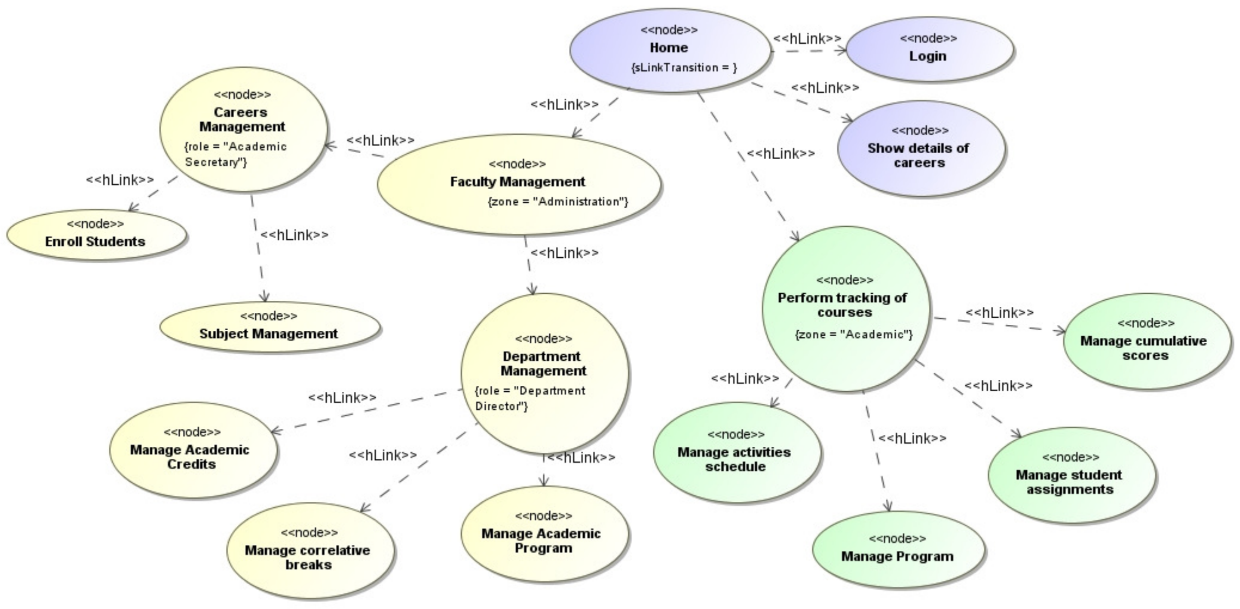

<<node>> Home {sLinkTransition = }
<<hLink>>
<<node>> Login
<<node>> Show details of careers
<<node>> Careers Management {role = "Academic Secretary"}
<<node>> Faculty Management {zone = "Administration"}
<<node>> Enroll Students
<<node>> Subject Management
<<node>> Department Management {role = "Department Director"}
<<node>> Manage Academic Credits
<<node>> Manage correlative breaks
<<node>> Manage Academic Program
<<node>> Perform tracking of courses {zone = "Academic"}
<<node>> Manage cumulative scores
<<node>> Manage activities schedule
<<node>> Manage student assignments
<<node>> Manage Program

Figure 1. Example of Navigational Tree for Academic System

It is worth noting that hard links are not sufficient to model the navigational structure of an application, because there are situations in which navigation through a different context will be necessary (e.g, once authenticated, the user must specify the destination node). To meet this need, we define other type of links (*soft link*). The soft links (sLink) will be specified in the node diagram.

Each tree node must have an associated *Navigational Node Diagram* representing the internal navigation of it. The node diagram is defined using the UML State Diagram. In this sense, the node diagram consists of two types of elements, *states* and *transitions*. There are three categories of states: *Flow states*, *Virtual states* and *Final states*. Flow states are transient and so are crossed momentarily to create linkages with other elements of the model. Among these we have four types: 1) the *initial* states, 2) *pseudo* states, such as choice, 3) *junctions*, 4) and finally the s*ervice* states that models the services provided by the node. *Virtual* states, represent stationary states indicating the fact that the model remains in a "virtual point" within a node and should be linked to a presentation page.

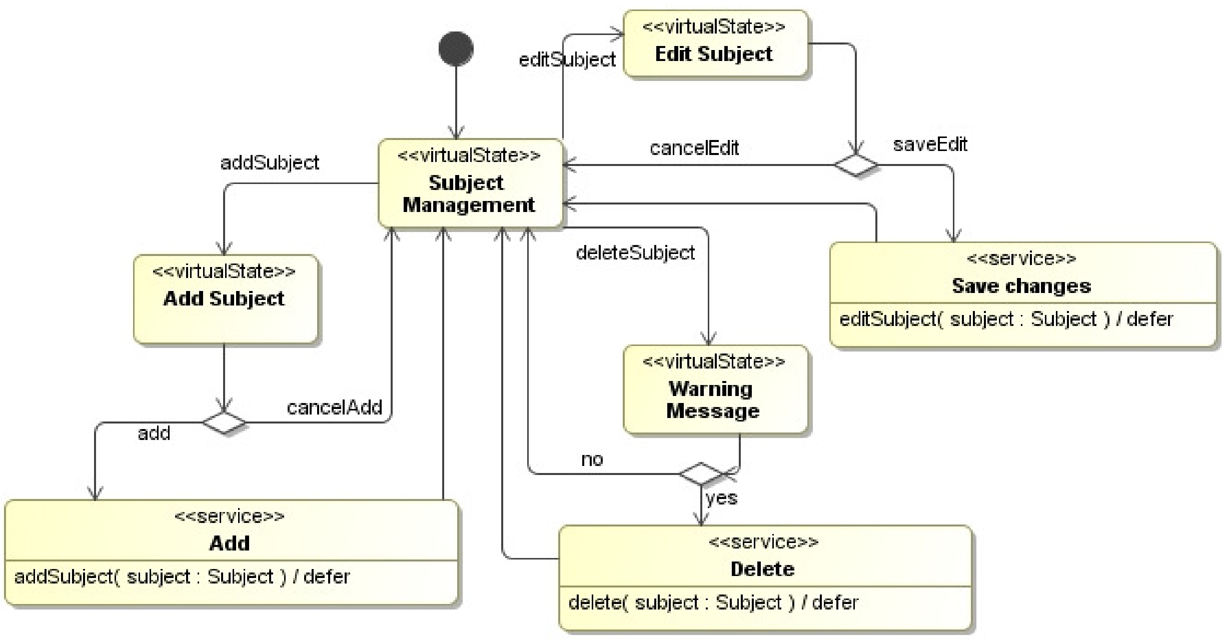

<<virtualState>> Subject Management
<<virtualState>> Edit Subject
editSubject
cancelEdit
saveEdit
addSubject
<<virtualState>> Add Subject
deleteSubject
<<service>> Save changes
editSubject(subject : Subject) / defer
<<virtualState>> Warning Message
cancelAdd
add
no
yes
<<service>> Add
addSubject(subject : Subject) / defer
<<service>> Delete
delete(subject : Subject) / defer

Figure 2. Example of Node Diagram for “Subjects Management” tree node

The node diagram allows modeling aspects related to navigation behavior obtained from dynamic interactions with the user. Figure 2 shows a diagram example for the navigational node "Subjects Management". This figure first shows the virtual state "Subject Management" which will display all existing subjects for a particular career. The user has the possibility to add a new subject, delete or edit an existing one.

Another element of the diagram node is called *Transition*. Transitions specialize in two subtypes: the *control flow* transitions and *hyperlinks*. The first transition models the natural control transfer that occurs between the states during the generation of the node. The second models the user activation of an *internal link*, which leads to an interaction between the user and the system. The first type of transition can only have as source a *flow state*, and can have as target any type of state (e.g., the transition between the service "Add" and "Subject Management"). The second type of transition can only have as source a *virtual state*, and as target any state (e.g., the transition between "Subject Management" and virtual state "Warning Message" which represents an intermediate warning page to the user before deleting a subject). *Hyperlinks* defined in the node diagram correspond to possible internal navigations, triggered from user interactions. The *Hyperlink* whose target state corresponds to a *final* state can be connected to another node in the navigational tree; if there is such a linkage, it defines a *soft link* (sLink) that will allow navigation to a unit not directly linked to the functional node of the navigational tree structure.

3.2 The User Model

The User Model includes the following diagrams: *Role and Zone Diagrams*.

The *Role Diagram* represents the hierarchy of user roles. To set this diagram notation we propose the use of the actor's hierarchy provided by UML in which, in addition, each actor may have a number of attributes in order to model personalization aspects. These attributes are called "personalized attributes" (examples of attributes are customizable language, style pages, colors, grants, etc.). Figure 3 shows an example of a role diagram.

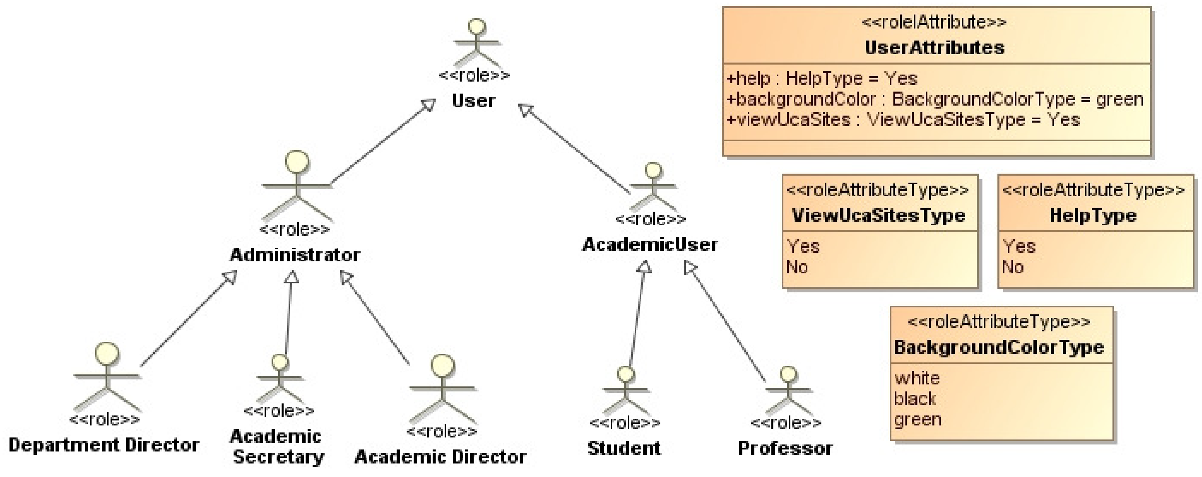


Figure 3. Example of Role Diagram

Another diagram for user modeling is called *Navigational Zone Diagram*. This diagram allows defining contexts of navigation. Contexts represent zones that contain certain behavioral profiles mutually related. The zones give the system designer the possibility to explicitly define different contexts within which multiple roles assumed by users can be identified. A role contains a default behavior within a zone, with its privileges and limitations. An example of a navigational zone could be one which both students and teachers can access, consisting of navigational nodes that provide access to subjects, career, etc. On the other hand, managers may be defined in a different zone of the academic system, since the specific type of functionality they need to access is different.

In general the idea is that there may be several zones defined in a system, each one can have several roles assigned, and each role can be played by zero or more users. The zone can be relative, that is, it depends on a class of the domain model, indicating that for a user to assume certain role, additional information is needed. To specify that a zone is dependent on a domain element, an association with a dependency relationship to a domain model class is represented. As an example of relative zone, we could have an "Academic" zone, which would include the roles of "Professor, Student", and users would take at most one of these roles for each subject, for example a user could take the role of "Professor" in a particular subject and the role of "student" in another one. An example of zone diagram is shown in Figure 4.

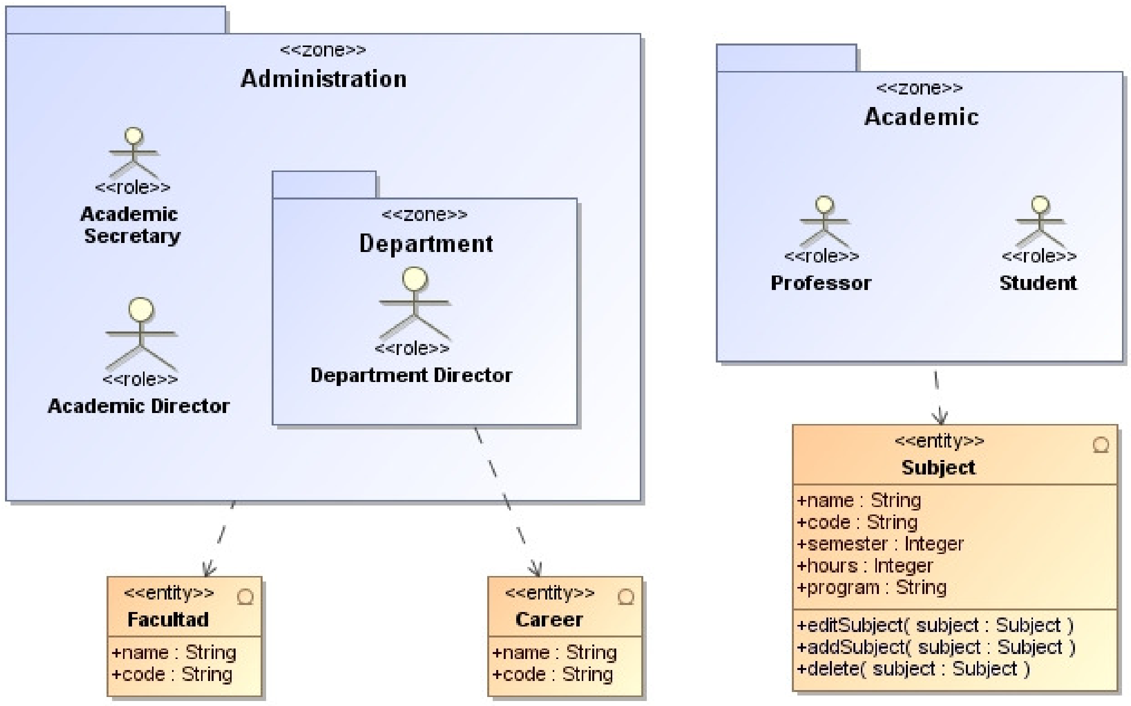


Figure 4. Example of Navigational Zone Diagram

The zone diagram, use the UML packages notation with the stereotype <<zone>>. Each package contains the available roles in that zone and a possible dependency relationship with a domain class. Then, whenever defining the accessibility level on the *Navigational Tree* diagram is necessary, we can specify a zone or a particular role for any navigational node. Finally, if instead of associating a zone to a node, we associated it to a particular role, the access node would be limited by the specified role.

The roles or zones association to navigational nodes can be visualized in the navigational tree (Figure 1). In this figure, it is possible to observe that some nodes do not have access privileges (i.e., can be accessed by any user), as is the case of the root node "Home". The node "Perform tracking of courses" assigns access privileges to the zone "Academic" (this zone contains the roles "Student" and "Professor" and depends on the domain class "Subject"). This indicates that both the node "Perform tracking of courses" as those which are below of it have access privileges of such a zone. The role each user assumes will depend on the courses he selects. Finally, the node "Faculty Management" assigns access privileges to the zone “Administration”. However, the child node "Career Management" assigns access privileges to "Academic Secretary”, indicating that it may be viewed and accessed by one of the roles played in the "Administration", and such access is restricted to the nodes children ("Enroll Students" and "Courses Management").

3.3 MOF Definition of the Navigational and User Notational Models

With the notational definition proposed in previous sections, we define a general metamodel which in turn is composed of three specific sub-metamodels: the Navigational Tree, the Node, and the Role (see Figure 5).

In the *Navigational Tree Metamodel* a general element, called *Navigational Tree*, represents the entire navigational structure and consists of zero or more navigational elements. These elements include: *Node* or *Link*. Node represents the navigational nodes that comprise the web application and connects to other units by means of elements called *hLink*.

The proposed metamodel for internal navigation of the nodes, named *Node,* consists of a “Node Diagram”, and all the elements it contains are represented in its most general level *SD Element*. SD Element is specialized in two types of elements: s*tate,* representing all states of the diagram (flow states, final state and virtual states); and *transition,* representing all transitions between different states (Hyperlinks, ControlFlow and sLink). The association between *FlowState* and *ControlFlow* suggests that the output transition for a *FlowState* can only be of type *ControlFlow*. The association between *VirtualState* and *Hyperlink* suggests that the output transition for a *VirtualState* can only be of type *Hyperlink*. The association between *FinalState* and *sLink* indicates that it is possible to define a soft link transition to a tree node as a final transition of the node diagram.

Following, we present the *Role Metamodel*. The element *RoleDiagram* represents the diagram of different roles of the system, which consists of *RD Elements*. The RD elements are specialized in three different elements: *Zone*, *Role* and *User*. The first, *zone*, represents the contexts with certain behavioral profiles related to each other. Zones can be related to each other by an aggregation relationship and can contain one or more roles. A *role* contains a predefined behavior within a zone, with its privileges, attributes, customizable and well-defined limitations. Roles are related to each other by a hierarchy relationship, and it is possible to assign to them one or more *roleAttributes*. The *user* element represents the different users that interact with the system.

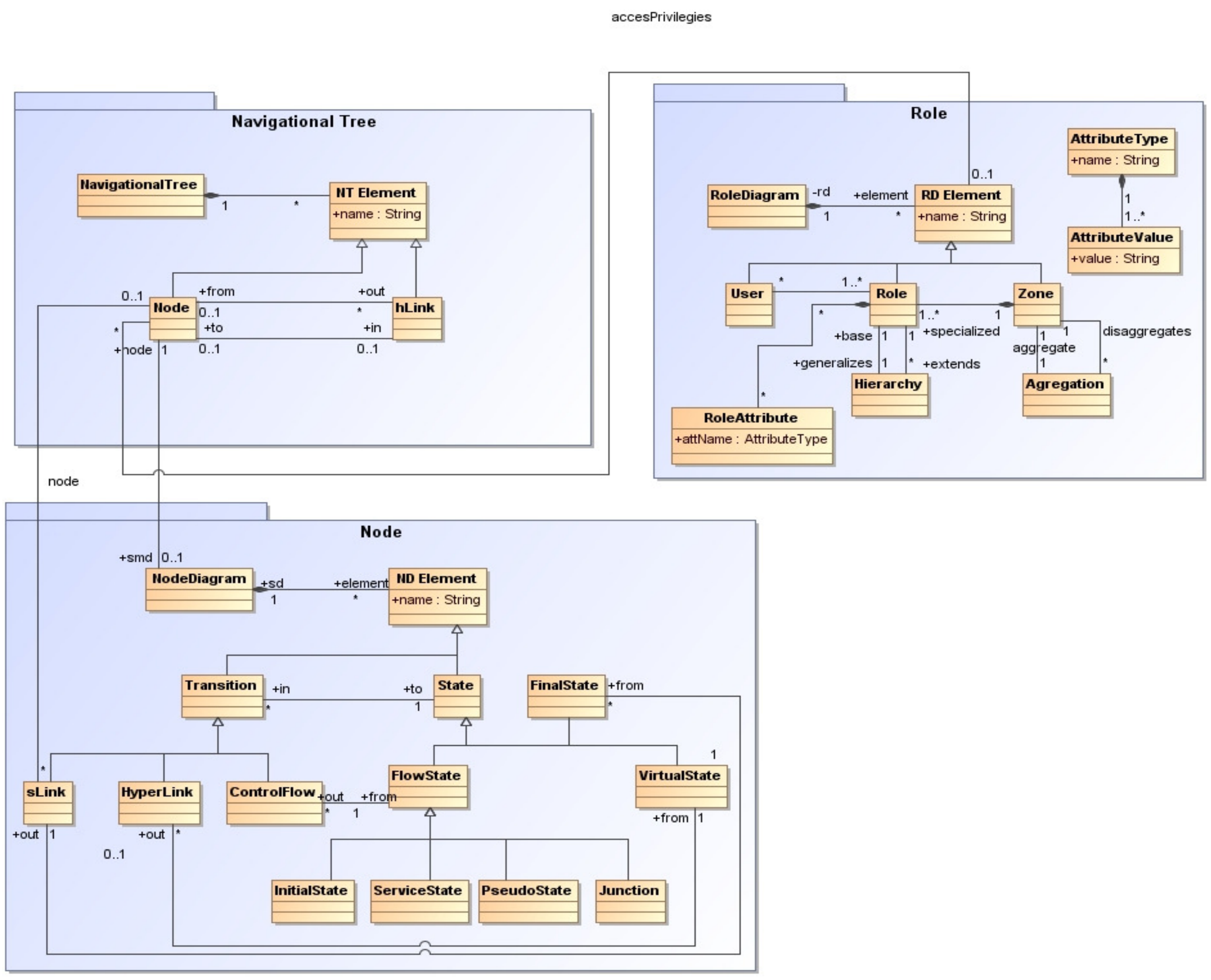
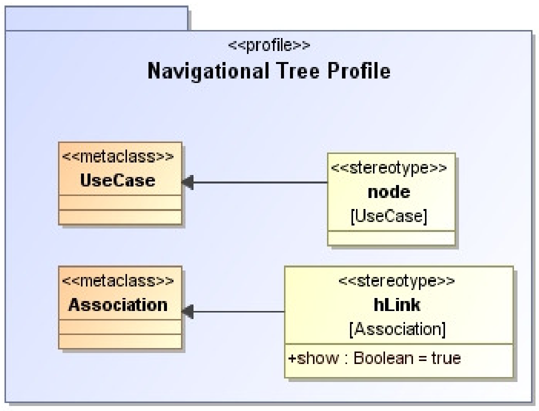


Figure 5. Navigational and User Metamodels

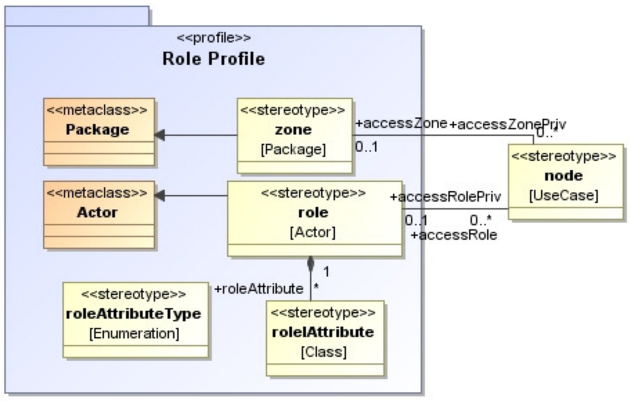
Finally, we can mention the different relations between the three metamodels. The *node* element in the *Navigational Tree* has an association with *NodeDiagram* (node metamodel) meaning that for each node there exist at most one *NodeDiagram*. *Node* also has an association with *sLink* (node metamodel) since it is possible to assign a soft link transition to any node of the tree as a final state of a node diagram. The association between *node* (navigational tree metamodel) and *RD Element* (role metamodel) indicates the level of accessibility of each node of the tree (zone, role or user).

In order to apply the Navigational and User metamodels we define UML profiles for them. These UML Profiles allow us to specify the different models with the proposed notation using a Case Tool for modeling. Figures 6, 7 and 8 present the profiles defined for this purpose.



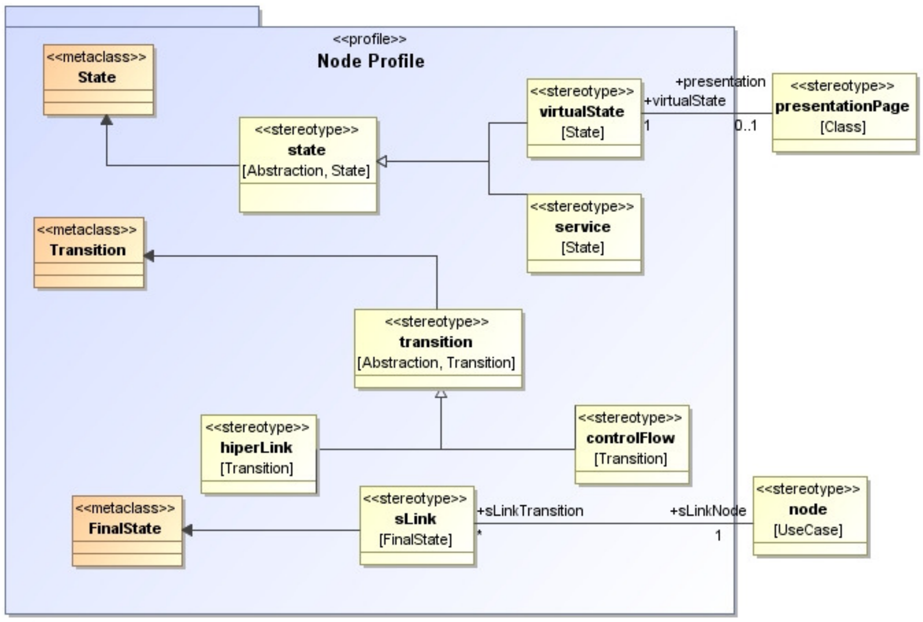
<<profile>>
Navigational Tree Profile
<<metaclass>> UseCase
<<stereotype>> node [UseCase]
<<metaclass>> Association
<<stereotype>> hLink [Association]
+show : Boolean = true

Figure 6. Navigational Tree Profile



<<profile>>
Role Profile
<<metaclass>> Package
<<stereotype>> zone [Package]
<<metaclass>> Actor
<<stereotype>> role [Actor]
<<stereotype>> node [UseCase]
+accessZone
+accessZonePriv
+accessRolePriv
+accessRole
0..1
0..*
<<stereotype>> roleAttributeType [Enumeration]
<<stereotype>> roleIAttribute [Class]
+roleAttribute
1
*

Figure 7. Role Profile



<<profile>>
Node Profile
<<metaclass>> State
<<stereotype>> state [Abstraction, State]
<<stereotype>> virtualState [State]
<<stereotype>> presentationPage [Class]
+presentation
+virtualState
1
0..1
<<stereotype>> service [State]
<<metaclass>> Transition
<<stereotype>> transition [Abstraction, Transition]
<<stereotype>> hiperLink [Transition]
<<stereotype>> controlFlow [Transition]
<<metaclass>> FinalState
<<stereotype>> sLink [FinalState]
+sLinkTransition
+sLinkNode
*
1
<<stereotype>> node [UseCase]

Figure 8. Node Profile

3.4 PSM and ISM Transformations Using a MDA Tool

To carry out a deeper analysis of the metamodel proposal, the transformation rules for the generation of both PSMs and ISM have been implemented using the Open Source MDA Tool called AndroMDA [1]. The tool selection resulted out of a comparative study of various MDA tools. AndroMDA was selected mainly due to its flexibility for cartridge creation, language standardization, MDA approach suitability, and its Open Source nature [11].

In the AndroMDA transformation approach, PSMs are generated instantiating the metafacades in a series of metaclasses. Subsequently, the templates use the metafacade to retrieve the necessary data from the models, and are therefore able to transform the final code.

The steps for implementing the metamodels can be summarized in the following: definition of the diagrams to be used with their respective UML profiles: 1) creation of the UML profiles; 2) identification of transformation rules; 3) modeling, creation and generation of metafacades; 4) implementation of transformation rules on metafacades; 5) configuration of the respective descriptors (cartridge.xml, metafacade.xml, profile.xml); and 6) tests of the cartridge with the proof of concept.

PHP was chosen as the target platform, due mainly to the fact that AndroMDA still did not offer any cartridge for this language, and also due to PHP’s widespread diffusion in the Web environment.

Once the implementation of metamodels finishes, the transformation process with AndroMDA is as follows:

1. All the required cartridges are loaded by the engine.

2. Some special repositories load into memory the XMI model and create run-time meta-classes and meta-objects that represent instances of them.
3. The engine encapsulates each meta-object called "helper classes", better known as metafacades, which can be defined in the cartridge or outside it.
4. The engine runs through each meta-object instantiated in step 2, looking for classes labeled with known stereotypes.
5. When a labeled class is located, the correct template cartridge is identified and used to generate code. On the other hand, the Metafacades mentioned in step 3 are available to the templates as specified in the descriptors of the cartridge. Multiple source files can be generated for every labeled class.
6. Step 5 is repeated for each class loaded from XMI.

Figure 9 graphically explains the steps just mentioned. In this figure, it is possible to visualize the major components and how they participate in the process of transformation and code generation. Flows are labeled corresponding the steps (2 to 5).

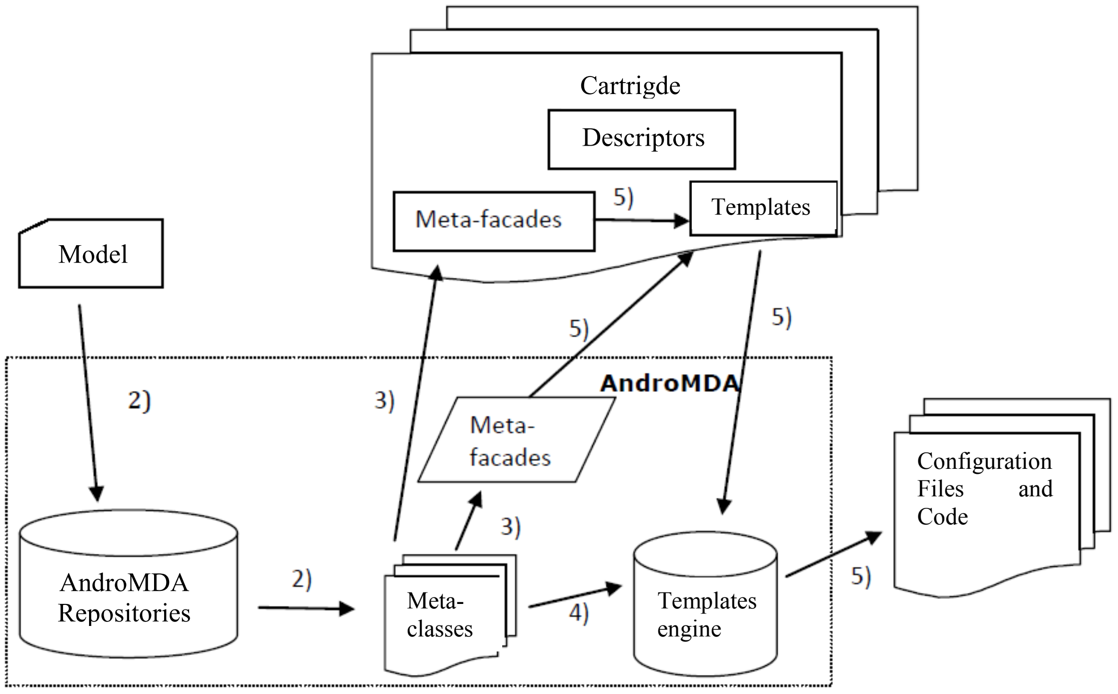


Figure 9. Transformation Process with AndroMDA

4 Proof of Concept with AndroMDA

After implementing the proposed metamodel, we applied the cartridge to a proof of concept, which allowed us to obtain feedback on its usefulness. This section presents the generation of the application using the cartridge developed for AndroMDA and the analysis of the results obtained.

4.1 Generation of the Application

For the implementation of the Academic System we modeled various diagrams, such as *Navigational Tree Diagram*, and for each node a *Node Diagram*. For *User Modeling,* the *Role and Zone Diagram* was defined. Examples of such diagrams were presented in sections 3.1 and 3.2.

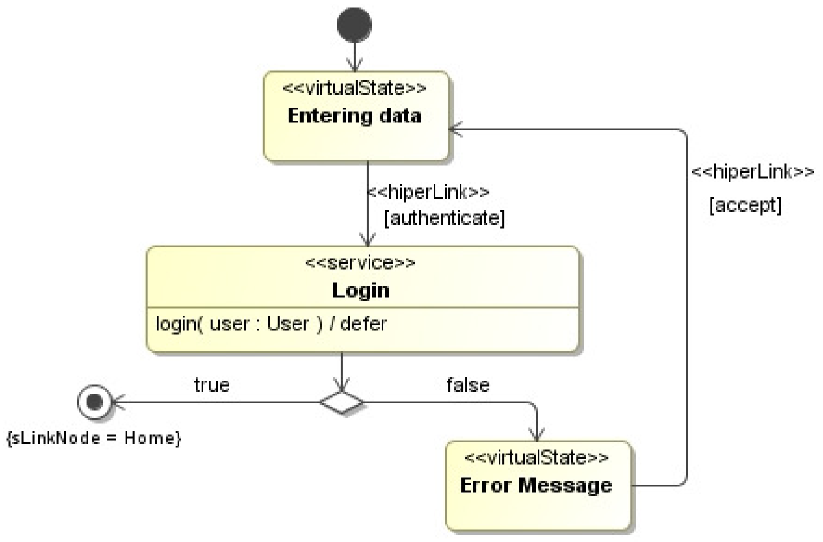
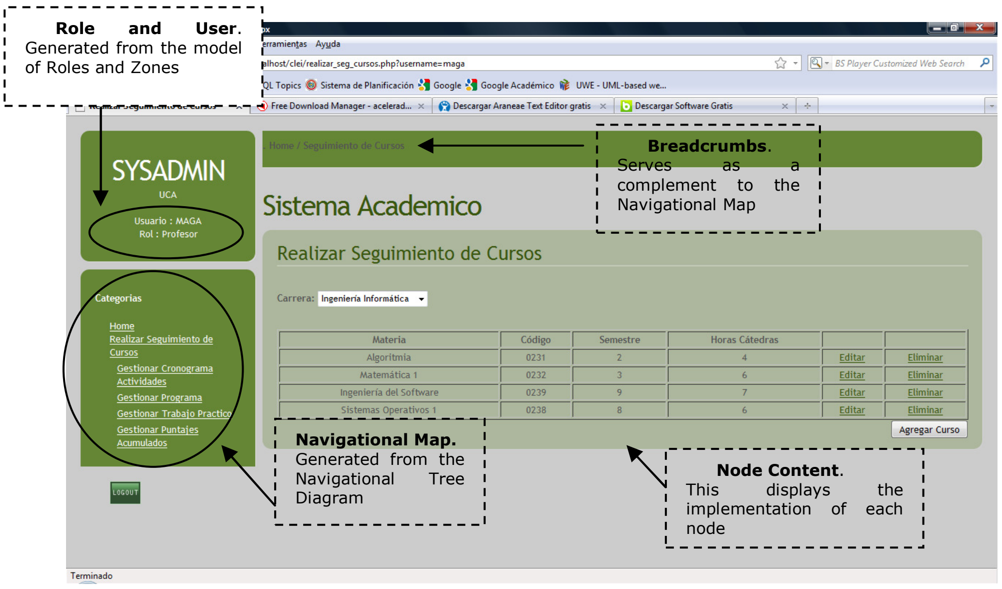


Figure 10. Node Diagram for *Login* tree node

In addition to the navigational tree diagram (figure 1), the role diagram (figure 3), and the zone diagram (figure 4), we can show another node diagram in figure 10. This figure models the login process in which the user has to type a user name and a password ("Entering data" virtualState), then a login service is executed to validate data, and finally, depending on the results, an error message will appear ("Error Message" virtualState), or a soft link will take the user to the "home" node of the system.

With the *navigational tree diagram* it was possible to dynamically generate the menus, depending on the role / roles of the user accessing the system, and "breadcrumbs" to keep the user located. Figure 11 shows the main structure of the application. The *node diagram* allowed the generation of the internal behavior of each function of the system. The *role diagram* indicates the hierarchy of roles within the system. Furthermore, this diagram is used to carry out user customization from custom attributes available to each role. Similarly, attributes of each of the roles are also used as session variables. These variables can be initialized from the model itself, which causes the user accessing the system for the first time to have certain preferences that distinguish it from other users with different roles. It is worth noting that in regard to navigation and users, the system generates 100% of the final code. However, in order to add the presentation elements it has been necessary to integrate the proposal with an additional presentation model, not presented in this work, but developed as part of a global project [12].


Role and User. Generated from the model of Roles and Zones

Breadcrumbs. Serves as a complement to the Navigational Map

Navigational Map. Generated from the Navigational Tree Diagram

Node Content. This displays the implementation of each node

SYSADMIN
UCA
Usuario : MAGA
Rol : Profesor

Home / Seguimiento de Cursos

Sistema Academico

Realizar Seguimiento de Cursos

Carrera: Ingeniería Informática

Materia	Código	Semestre	Horas Cátedras		
Algoritmia	0231	2	4	Editar	Eliminar
Matemática 1	0232	3	6	Editar	Eliminar
Ingeniería del Software	0239	9	7	Editar	Eliminar
Sistemas Operativos 1	0238	8	6	Editar	Eliminar

Agregar Curso
Categorias
Home
Realizar Seguimiento de Cursos
Gestionar Cronograma Actividades
Gestionar Programa
Gestionar Trabajo Practico
Gestionar Puntajes Acumulados
LOGOUT
Terminado

Figure 11. An example of generated page

4.2 Analysis of Results

The actual proposal adopts standard modeling languages and architectures. On one hand, the notational proposal corresponds to an extension of UML. In addition, MOF has been used to define the different metamodels. The transformations were defined using the MDA approach with an open source MDA tool. These features were designed to facilitate: i) the understanding of the usefulness of the proposal; ii) the adoption of the models by using CASE tools supporting UML notation; and, iii) the possible integration of the proposal with other methodological approaches.

With regard to the notational proposal defined for PIM, here are some important considerations. While the vast majority of Web methods propose navigational maps, the contribution of the *Navigational Tree Diagram* focuses on one side in the differentiation between the "hard-links" and "soft-links" for navigational structures definition. The "hard-links" are used to define hierarchical navigation structures related to functional units, while the "soft-links" are intended to model those navigations that are made from changes in functional units or contexts. With this contribution it is possible to generate various levels of explorations through menus and submenus, while maintaining the user located through "breadcrumbs" and "history of navigation." Although other proposals do not consider these types of navigation, OOWS provides the ability to model hierarchical structures and navigation levels with *sequence* and *exploration links,* and *subsystems* definition, which is considered to be useful but we think they have not the same conceptual clarity introduced by the *hard* and *soft links*. Another contribution of the *Navigational Tree* corresponds to the granularity used to model the navigational structure, considering the navigational elements as functional units. In fact, this proposal adopted the *Use Case Diagram* to define the navigational structure. Use cases represent functional units that may include process flows. The navigational hierarchies are based on these principles and not on the contexts that other methods use to represent one or several web pages with the same structure. Finally, the navigational model of the implementation is simplified for two reasons. The first is due to navigational nodes represent a higher aggregation level than normally used in other proposals, therefore the number of navigational nodes tends to be much smaller. The second reason is that displaying only the navigational hierarchy in the overall structure of the application makes the overall system vision much simpler, because the number of links is strongly reduced.

The *Navigational Tree Diagram* is independent of the conceptual model and could correspond to the starting point of the system modeling process. The navigational tree diagram is defined by structuring the basic functionality that the system should provide instead of using a conceptual model view (in the case of OOHDM, UWE, WebML the navigational structure is created from the conceptual model). We believe this concept can help to define simpler functions oriented navigational structures, which can simplify user orientation.

A *Node Diagram* is defined for each node (functional unit) of the *Navigational Tree*. The abstraction of functional unit provides more flexibility for modeling aspects related to navigation behavior obtained from dynamic interactions with the user. Moreover, the diagram provides the ability to model navigational elements as "soft links", "index", "guided tours", and "service links" in a more natural way and without using specific constructors for this purpose.

Roles and Zones Diagrams, in addition to providing a hierarchical model of user roles, establish relationships between zones and single users with multiple roles, and guide the navigation depending on the role that the user play within a particular zone. In addition, the possibility of allowing nodes access to both zones and roles, gives greater flexibility and simplicity to the application model. In this sense, when we assign zones privileges, we are indicating that the node can be accessed by any of the roles of that zone, and if it is linked to an element of the domain, the assumed role depends on the domain element linked to the node.

With regard to the ISM (i.e., the generated code), it is worth noting that navigational and user aspects were 100% covered in the final code. However, to add the elements of presentation it was necessary to integrate the proposal with an additional presentation model, no presented in this work, but developed as part of a global project [12].

5 Conclusions and Future Works

The present study focuses on an analysis of the navigational and user perspectives in Web Applications, highlighting some critical aspects for the navigation and users modeling. Specifically, we present a notational proposal and its definition through a metamodel using the MOF language. The Metamodel in turn is composed of three sub-metamodels: Navigational Tree, Node Tree, and Roles Tree. Metamodels have been implemented on a specific MDA tool (AndroMDA) through the development of a user and navigational cartridge that considers PHP applications as the target platform. Both the Metamodel and the Cartridge for AndroMDA have been analyzed by means of a proof of concept.

The experimental result shows that the considered aspects for the definition of the Users and Navigational model present meaningful contributions compared with other current methodological proposals. Particularly, in the navigational tree modeling, hard and soft link constructors have been introduced to enhance the definition of the hierarchy structure and to differentiate them from the links that imply a context change between functional units.

Furthermore, a new way of considering nodes as functional units is proposed, allowing to model behavioral diagrams. Besides providing a hierarchical model for the users, associations were established with zones that allow the assignment of various roles to the same user and to guide the navigation depending on the role that the same user has on a particular zone.

It is also important to mention that with this study, a new cartridge for AndroMDA has been developed, to generate application in the PHP platform.

This work has made possible to distinguish some challenges considered important as future works; such as the metamodels proposal with its transformation rules for the other layers (presentation, domain, services); the extension of the cartridge to other target platforms, such as JSP; and, the integration of the current proposal into other methods to complement the navigational and user layer modeling. In this sense, studies about the integration with OOWS could be made, replacing the navigational map context by nodes with the navigational nodes diagram. Each virtual state of the node diagram would represent a navigational context. In the UWE case, the navigational structure could be enriched with the node diagram, defining it for each navigational class, and then analyzing the results from the obtained models.

ACKNOWLEDGEMENT
The first author would like to thank the “Itaipu Technological Park” (Parque Tecnológico Itaipú – PTI) for partially funding the research under a scholarship program.

References

1. AndroMDA. http://www.andromda.org/
2. Baresi, L. and Mainetti, L. “Beyond Modeling Notations: Consistency and Adaptability of W2000 Models”. In Proc. Of ACM Symposium on Applied Computing, USA, (2005)
3. Cachero C., Gómez J., Pastor O. “OO-HMethod: Un Método de Diseño de Lugares Web”. Conference Proceedings, IDEAS 00. Cancún, pp 133-144, México (2000)
4. Cachero, C., Koch, N. “Conceptual Navigation Analysis: a Device and Platform Independent Navigation Specification”. 2nd International Workshop on Web-oriented Software Technology (IWWOST’02). Málaga, España (2002)
5. Ceri, S., Fraternali, P., Bongio. “Web Modelling Language: A Modelling Language for Designing Web Sites”. Conference WWW9/Computer Networks 33, pp. 137-157 (2000)
6. De Troyer, O. y Casteleyn, S. “The Conference Review System with WSDM”. In First International Workshop on Web-Oriented Software Technology, (2001)
7. Escalona M. J. “Modelos y Técnicas para la Especificación y el Análisis de la Navegación en Sistemas Software”. Tesis Doctoral. Universidad de Sevilla. (2004)
8. Fons, J. - Pelechado, V. - Pastor, O. - Valderas, P. - Torres, V., “Applying the OOWS Model-Driven Approach for Developing Web Applications: The Internet Movie Database (IMDB) Case Study” en Pastor, O. - Olsina, L. et al (eds.) Web Engineering: Modeling and Implementing Web Applications. Springer, Chapter 5, pp 65-108, London (2007)
9. Garzotto, F., Paolini, P., Schwabe, D. “HDM- a Model-Based Approach to Hypertext Application Design”. ACM Transaction On Database Systems, 11(1), pp. 1-26 (1993)
10. Ginige, A., and Murugesan, S. (Eds), “IEEE Multimedia Special Issue on Web Engineering: Part 1”, 8,1, (2001)
11. González M., Bareiro J., Rodriguez R., Cernuzzi L. “Estudio Comparativo de Herramientas MDA para el Desarrollo de Sistemas Web”. XXXV Conferencia Latinoamericana de Informática (CLEI). Pelotas, pp. 56 , Brasil (2009)
12. González M., Cernuzzi L., Pastor O. “Una Aproximación para Aplicaciones Web: MOWEBA”. XIV Congreso Iberoamericano en “Software Engineering” – CibSE, Río de Janeiro, Brasil (2011)
13. Houben, G., Van der Sluijs, K., Barna, P., Broekstra, J., Casteleyn, S., Fiala, Z., and Frasincar, F., “HERA” en Pastor, O., Olsina, L. et al (eds.) Web Engineering: Modeling and Implementing Web Applications. Springer, London, chapter 10, pp 1-6 (2007)
14. Koch, N., Knapp, A., Zhang, G., Baumeister, H., “UML-based Web Engineering, An Approach Based on Standards” en Pastor, O. et al (eds.) Web Engineering: Modeling and Implementing Web Applications. Springer, Chapter 7, pp 157-192, London (2007)
15. Nurmi, P, Laine, T., Seminar on User modeling: Introduction to User Modeling, (2007)
16. MDA Guide Version 1.0.1. http://www.omg.org/docs/omg/03-06-01.pdf (2003)
17. Rossi, G., Pastor, O., Schwabe, D., Olsina, L. “Web Engineering: Modelling and Implementing Web Applications”. (Human-Computer Interaction Series), 1 edition. ISBN:9781846289224 . Springer, London, (2007)
18. Schwabe, D., Rossi, G. “An Object Oriented Approach to Web-Based Application Design”. Theory and Practice of Object Systems 4(4), 1998. Wiley and Sons, New York, ISSN 1074-3224 (1998)
19. Suh, W., Lee, H. “A Methodology for Building Content-oriented hypermedia systems”. The Journal of Systems and Software, Vol 56, pp. 115-131. (2001)