Journal of Web Engineering, Vol. 7, No.3 (2008) 239-256

ENRICHING MODEL-BASED WEB APPLICATIONS PRESENTATION

JUAN CARLOS PRECIADO, MARINO LINAJE, & FERNANDO SANCHEZ-FIGUEROA

Quercus Software Engineering Group / Homeria

Escuela Politécnica. Universidad de Extremadura (10071 – Cáceres, Spain)

{jcpreciado; mlinaje; fernando}@unex.es

Received November 9, 2007
Revised March 11, 2008

The Web Engineering community provides Web Models that allow the design and development of Web 1.0 applications. Nowadays, there are a growing number of Web 1.0 applications that are migrating towards Web 2.0 User Interfaces, in search of multimedia support and higher levels of interaction among other features. These Web 2.0 features can be implemented using RIA technologies. However, most of the current Web Models do not fully exploit all the potential benefits of Rich Internet Applications. Therefore it is difficult for developers to adapt Web 1.0 applications to Web 2.0 using a methodology. Although there are interesting works that extend existing methodologies to deal with RIA features, either they do not fully exploit presentation issues or they only work with a single Web methodology. In this paper we use RUX-Method to facilitate the evolution of existing Web 1.0 applications based on Web Models to multi-device Web 2.0 applications. RUX-Method focuses on the enrichment of the User Interface while takes full advantage of the functionality already provided by the existing Web models. Far from explaining RUX-Method in detail, this paper focuses on the way the information provided by Web Models is retrieved and then used by RUX-Method.

Key words: Web Engineering, User Interfaces, Web 2.0, Rich Internet Applications, Web UI evolution.

1 Introduction

More and more, the future of the Web is being shaped by post-HTML/HTTP technologies, among which a prominent role is played by the so-called Rich Internet Applications (RIAs) [27]. The term refers to a heterogeneous set of solutions, characterized by the common goal of adding new capabilities to the conventional hypertext-based Web applications. Some of the novel features of RIAs affect the User Interface (UI) and the interaction paradigm; others extend to architectural issues, such as, the client-server communication and the distribution of the data and business logic. RIAs combine the richness of desktop computing with the appeal of distributed data processing. They support online and offline usage, sophisticated UIs, data storage and processing capabilities directly at the client side, powerful interaction tools leading to better usability and personalization, lower bandwidth consumption, and better separation between presentation and content.

Many Web 1.0 applications are now evolving towards Web 2.0 using RIA technologies to overcome Web 1.0 technological limitations. Among the new features provided by RIA, those related

to the UI are very appreciated by the final users, and many developers are using RIAs just to enhance existing Web applications. In most of these applications the data and business logic may remain the same but the UI must evolve. In this context, rather than rewriting the whole application, good software engineering practice would be to reuse the data and business logic provided by the Web 1.0 application applying a methodology to improve the presentation.

The Web Engineering community has proposed different models and methodologies to support the development of Web Applications covering the different parts in which a Web application can be divided (data, navigation, business logic, presentation...). However, these methodologies do not support RIA UI design [24] even when it is considered that UI development is one of the most resource-consuming stages of Web applications development [8]. Therefore, there is a real need for systematic methods and tools to facilitate the evolution of Web 1.0 UIs to Web 2.0 UIs and here is where RUX comes into the scene.

RUX-Method (Rich User eXperience Method, now on RUX) [17], is a model-driven method which supports the design of multimedia, multi-modal and multi-device interactive Web 2.0 UIs for RIAs. In the context of this paper, RUX can be seen as a systematic method for supporting the evolution of applications with Web 1.0 UIs, already developed using a Web model, to applications with 2.0 UIs. Figure 1shows the relationships between RUX, RIA, the underlying Web Model and the Legacy Web Application, both at design time and run time. At design time, RUX uses (marked **1** in Figure 1) existing data, business logic and presentation models offered by the underlying Web model being enriched. This information provides a UI abstraction which is transformed until the desired RIA UI is reached. At run time, while a new UI is generated from RUX, the data and business logic remain the same. So, the role of RUX is providing a new UI with RIA features, invoking the appropriate existing operation chains (marked **R** in Figure 1).

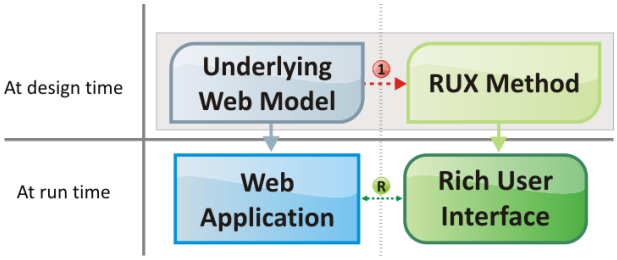

Figure 1 Relationship between RUX, RIA, the underlying Web Model and the Legacy Web Application, both at design time and run time

The objective of this paper is not to describe RUX in detail, for this purpose the reader is referred to [17]. Instead, here we present the connection process between RUX and WebML [6] as a particular case of Web Model. In this sense, this work extends [22] providing more details about the RUX connection process.

Although here applied to WebML, the connection concepts are general enough to be used with other approaches such as OO-HDM [26], UWE [15], OO-H [13], WSDM [9] or Hera [14]. However, in practical terms, each model expresses its Web concepts in a manner sufficiently different from the rest. This makes necessary a different set of rules adapted to each Web model at least at implementation level (e.g., in [35] we also extend UWE presentation capacities using RUX).

The rest of the paper is as follows. Section 2 gives an overview on Related Works. Section 3 introduces an application scenario where our approach is suitable to be used. In Section 4 RUX is briefly described. Section 5 focuses on how RUX connects with WebML. This connection is applied to the application scenario. In Section 6 several issues regarding RUX-Tool (the RUX CASE tool) are presented. In addition, experimental results are outlined. Finally, conclusions are summarized in Section 7.

2 Related Work

The rise of RIA has introduced into the life-cycle of Web applications many issues related to the four levels of RIA design: data, business logic, communication and presentation [23]. However, traditional Web Engineering models and methodologies were not created with these issues in mind. For this reason, they are being extended in several directions. For example, [2] extends WebML to cope with data persistence and task distribution capabilities of RIAs by providing an integrated model for server/client interaction and [11] extends Hera to cover some RIA presentation issues. Our approach also follows a model-driven approach, since according to [1] is more convenient to deal with changes in a domain specific way than to formulate them at the programming construct level or in a purely lexical level.

Regarding the evolution of Web applications with HTML-based UIs to RIA-based UIs, most of the approaches extend a specific existing Web model to deal with RIA features or presentation issues [4] [11] [12]. In contrast, RUX provides a method that can connect to different Web models to build a new UI. To our knowledge, there is no other method with this aim. However, the mentioned proposals together with others coming from the HCI field, deserve our attention for different reasons.

Among those Web models that claim to integrate natively adaptive concepts, a well-known one is Hera [14] which is a model-driven approach that integrates concepts from adaptive hypermedia systems with technologies from the semantic web. In Hera, the design artefacts of the three models used in the development process can be adapted by annotating them with appearance conditions. An extension of Hera, Amacont [11], deals with presentation issues such as multidevice support. However, this extension does not provide temporal and interactive relationships and does not consider multiple Web models.

Other Web engineering works extend Web models with adaptive primitives. For example [12] (using PRML) and [4] (using ECA rules) are generic and valid in many different environments as well as for many different dimensions. Both are based on well-known Web models, [12] on OO-H and [4] on WebML. Both require a "rule" engine (that in the related literature is usually addressed as a specific server extension). However, their scope is wider than ours, the reason being they do not specifically focus on RIA presentation and interaction necessities. For example, using this approach is not possible to detect several user interactions that can modify only the UI state at client side [25] without the necessity to carry out unnecessary communication roundtrips. This problem is solved in RUX by including rule-based presentation adaptive support through ECA rules that are evaluated at client-side.

[10] is based on the general notion of profile that can be used to represent a variety of contexts with different detail levels. Each profile is associated with a configuration that specifies how information has to be delivered. This proposal can also address an ad-hoc fine-grained final UI, but all

the context presentation issues are specified with html+css, making them hard to maintain and difficult to cover manually due to the cross-browser layout and look&feel incompatibility problems.

Our approach is also slightly different with respect to [28], because we do not stop the process in the Abstract Interface design stage. RUX-Method includes the Concrete Interface design stage allowing us to achieve full UI code generation using RUX-Tool.

There are other interesting works coming from the HCI community which are based on the use of XML-based languages, such as UsiXML [16] and TeresaXML [33]. The research in this area has mainly focused on helping designers to obtain different versions that adapt or can be adapted to various environments. RUX follows many concepts from traditional HCI techniques such as a multi-level interface design based on the Cameleon Framework [3] and even part of the RUX notation is based on UiML [34]. RUX-Method is a point of union between the HCI and the Web engineering fields since existing presentation models/description languages cannot be applied directly to model RIA UIs [17]. Notwithstanding, this is changing since some HCI proposals are now looking at RIA [19].

There are some other approaches supporting the UI evolution of existing Web applications towards RIA but they are not focused on model driven development. Among these proposals, [18] defines a feasible approach not limited to UI related issues where reverse engineering techniques are applied over an existing Web application to obtain a new SOA and AJAX based Web application. However, major changes would be necessary to support the evolution of the UIs towards other RIA technologies. In the same way, the work in [20] has tried to use XUL, XIML and UiML as UI description languages, but the main problem they found is that many of these languages are designed for static UIs with a fixed number of UI components (widgets) making them not suitable for RIA. The authors also claim that they are attempting to design their own single-page UI metamodel, but this issue is already treated by RUX.

3 Application Scenario

This section presents an application scenario in which RUX is suitable to be used. The scenario is quite simple, facilitating the full connection process description in the following sections. It consists of an interior design on-line portfolio that is organized into categories where each category is associated to several pictures.

Figure 2 corresponds to the Web 1.0 UI running example, currently available at *http://www.graciouslivinginteriors.com/portfolio* where the portfolio is structured into three different pages. There is some presentation information about the portfolio in the initial page as well as a list of categories (i.e. kitchens, living-rooms, etc). Each category has a link for navigating the categories page. In the latter page, there is a list that contains all the photos available for the selected category. Finally, for each listed photo there is a link that, when clicked, allows us to access the detailed photo page providing a bigger image view as well as a brief description.

Before designing the example with WebML, first we give an overview on WebML and its data and hypertext models.

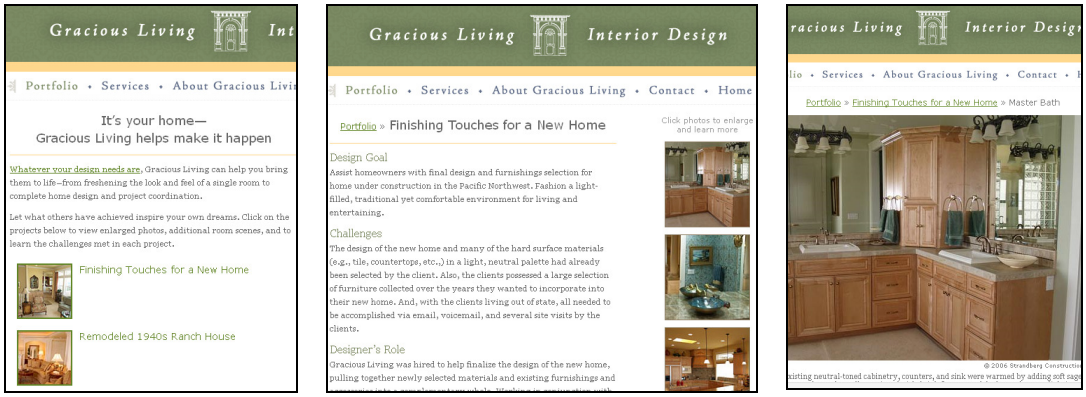

Figure 2 Application scenario using a Web 1.0 UI. Left: Initial portfolio page; Center: Category page; Right: Detailed photo page

3.1 WebML organization

WebML allows designers to conceptually express navigation and business logic of the Web site by means of two phases in charge of data and hypertext designs.

The *Data model* is based on the Entity-Relationship diagram to specify the application data. The concepts used in the Data model are entities and relationships between entities. An entity has several properties called attributes.

The *Hypertext model* supports the Web application navigation and here is where the business logic is specified. Hypertext modeling also specifies composition. The composition expresses how containers (i.e. Siteviews, Areas and Pages) arrange the content (i.e Hypertext units) describing how the content is retrieved and organized by means of information units. In this phase, the designer defines the organization of the content using content units, operation units and links among them. These content units are used for displaying the information from the data defined in the Data model (e.g., Data Unit and Index Unit). Each Content Unit is related to one underlying entity defined in the Data model. The operation units are used for performing additional operations that manage and update the content. Finally, the links are used to relate units expressing the navigation and to trigger the operation chains.

The root navigation element in WebML is Siteview which is the leading unit of grouping. It expresses a rational hypertext designed to serve the needs of a specific category of users (e.g., customers, administrators, and so on).

At the implementation level, the WebML concepts are hierarchically organized using XML nodes, attributes and contents. The most relevant for our purposes are:

<Structure>: related to Entity, Attribute of entity and Relationship between entities. These concepts are defined in the Data Model.

<Navigation>: related to containers (*<Siteview>*, *<Area>* and *<Page>*), units (*<DataUnit>*, *<IndexUnit>*, *<HierachicalIndexUnit>*, *<MultidataUnit>*, *<EntryUnit>* and Operation Units) and links. All of these are defined in the Hypertext model.

As the reader can infer, the composition (i.e. elements grouping) of the presentation is specified by the Hypertext model, representing those concepts that many other models specify in the Presentation model. The rest of the WebML presentation model is not considered in the context of this work due to it not being oriented to the way that Web 2.0 UIs behaves (e.g. single page application).

3.2 WebML in the application scenario

The Data model is shown in Figure 3, where each photo can be related with many categories and each category with many photos.

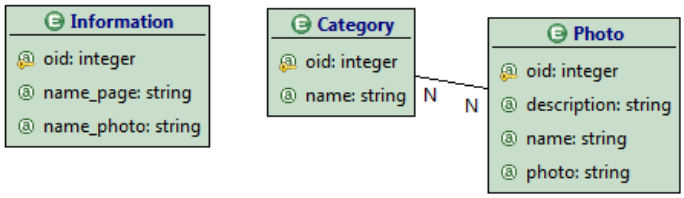

Figure 3 WebML Data model for the application scenario

Based on the Data model, the Hypertext model is depicted in Figure 4. The hypertext organization is as follows: "Menu" page contains an Index Unit related with the entity Category that represents the list of categories that will be rendered in a page.

"Menu Photos" page contains an Index Unit related with the Photo entity and constrained by a role condition to specify the relationship between Photo and Category entities. This page represents the list of photos related with a specific category that will be rendered in a page.

"Show Photos" page contains a Data Unit related with the Photo entity standing for the detailed photo page. To clarify the example, the general information about the portfolio has been included in a separated page called "Home" that is marked as homepage of the Web application (marked with a house icon in the page representation – Figure 4).

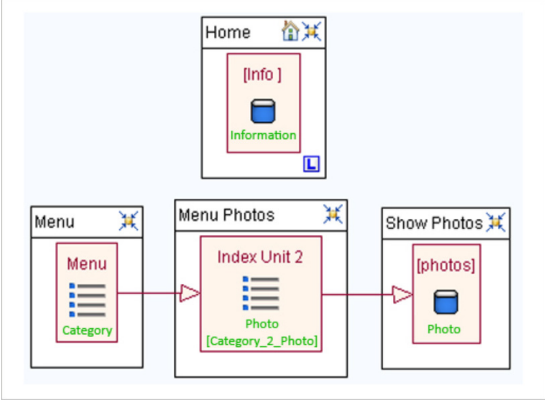

Figure 4 WebML Hypertext model for the application scenario

Relations among pages and units are specified with links that for the scenario are normal WebML links with automatic coupling (where the OID is automatically used as unique identifier for the link

context). “Menu” unit has a link to navigate to the “Index Unit 2” and “Index Unit 2” has another link to navigate to the photo details page. The navigation is performed by means of user clicks.

By using the rest of the WebML Presentation model (not shown here) and WebRatio (its associated CASE tool) the Web 1.0 application can be deployed with an appearance similar to that one exhibit in Figure 2. However, this Web 1.0 UI has several limitations such as poor user interaction, limited number of controls/widgets and non selective UI refreshment among others [27]. So the final Web technology is constraining both, the user experience and the developer’s creativity.

To overcome the aforementioned limitations, many Web applications are being built on the basis of Rich Internet Applications. An example of this is the Pedro-Verhue on-line portfolio. A Web application that offers high quality photos organized into categories using a main horizontally placed menu. In the Web application, each category showed has an *onmouseover* event that displays a second level category placed vertically over the current category showing its photograph index. When one of these photographs is selected, the selected one covers the whole application screen. The entire process is carried out using the single page application paradigm as the base of many other RIA concepts such as animations and transitions. Figure 5 depicts a screenshot of the application where arrows labeled “Menu”, “Home”, “Menu Photos” and “Show Photos” represent the relation with the pages of the Hypertext model. With the aim of receiving a more real user experience we recommend the reader visit the running Web application at *http://www.pedro-verhue.be*.

If we want to adapt the UI of Figure 2 to the UI of Figure 5, the changes must not affect the data and the hypertext design. Conceptually, the changes only affect the presentation level. Here is where RUX comes into play. RUX uses the data and business logic of Figure 3 and Figure 4, providing a UI abstraction which is transformed through several steps until the desired RIA UI is reached. At design time all the information for triggering the operation chains is also extracted so, at run time, the new UI can invoke the appropriate operations according to the underlying Web model.

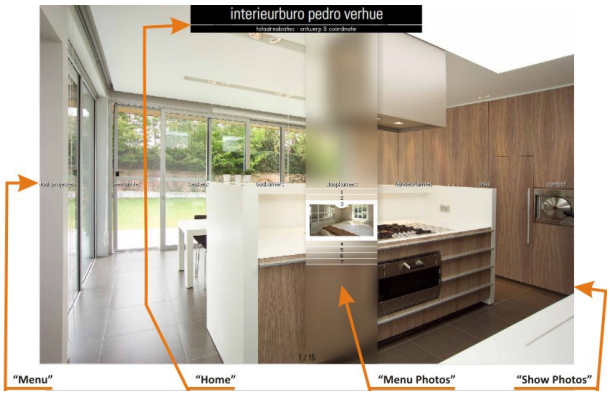

Figure 5 Application scenario using a Web 2.0 UI

4 Rich User eXperience Model in brief

The method was designed with many Web methods in mind which are responsible for the data and business logic design of the Web application, while RUX focus on the model-driven specification of

multidevice and interactive multimedia Web UIs. RUX provides an intuitive visual method that allows designing rich UIs for RIAs and its concepts are associated with a perceptive graphical representation. Due to it being a multidisciplinary proposal and in order to decrease cross-cutting concepts, the presentation model is divided into three Interface levels: Abstract, Concrete and Final Interface. These levels are mainly composed by Interface Components which are specific for each level. All the information related to the Interface Components is stored in a Component Library.

Abstract Interface provides a UI representation common to all RIA devices and development platforms without any kind of spatial arrangement, look&feel or behaviour, so all the devices that can run RIAs use a common Abstract Interface. The Abstract Interface meta-model is depicted in Figure 6. The different elements are:

- ***Connectors***, included to establish the relation to the data model. The way in which the data is going to be retrieved is provided by the hypertext model;
- ***Media***, they represent an atomic information element that is independent of the client rendering technology. We have categorized media into discrete media (texts and images) and continuous media (videos, audios and animations). Each media gives support to Input/Output processes; and
- ***Views***, which symbolize a group of information that will be shown to the client at the same time. In order to group information, RUX allows the use of four different types of containers: simple, alternative, replicate and hierarchical views.

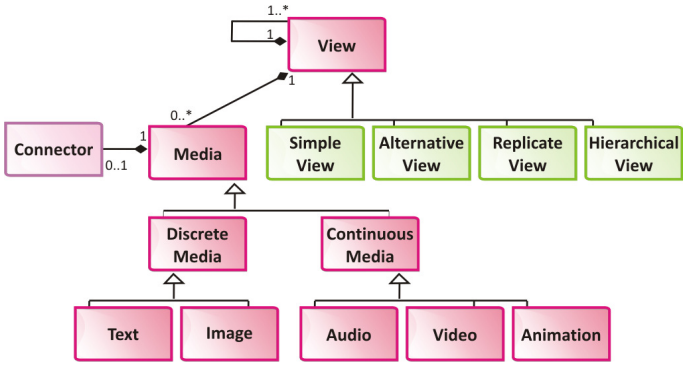

Figure 6 Abstract Interface meta-model

In the Concrete Interface, we are able to optimize the UI for a specific device or a set of devices. Concrete Interface is divided into three Presentation levels: Spatial, Temporal and Interaction Presentation. Spatial Presentation allows the spatial arrangement of the UI to be specified, as well as the look&feel. Temporal Presentation allows the specification of those behaviours which require a temporal synchronization (e.g. animations). Interaction Presentation allows modelling the behaviours that the user produces through events over the UI.

These two levels (abstract and concrete) allow RUX to achieve an ad-hoc UI development approach while maintaining an abstract interface description common to all the devices.

The RUX process ends with Final Interface which provides the code generation of the modelled application when using RUX-Tool. This generated code is specific for a device or a set of devices and for a RIA development platform and it is ready to be deployed together with the data and business logic provided by the underlying Web application.

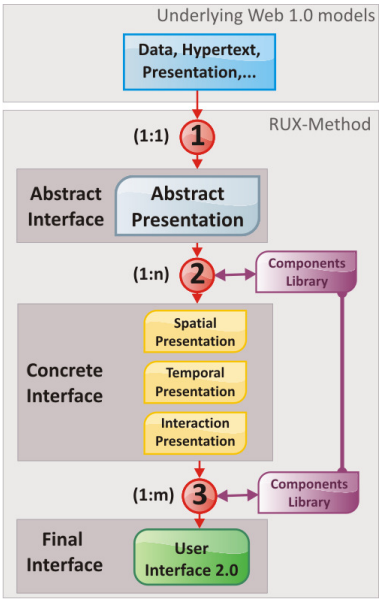

Figure 7 RUX architecture overview

RUX supports Web 2.0 UI design through three different transformation phases. Figure 7 shows the different interface levels and transformation phases. The first transformation phase (*Connection Rules* - marked **1** in the Figure) is automatically performed and extracts all the relevant information from the previous Web Model to build a first version of Abstract Interface. The second transformation phase is then performed (*Transformation Rules 1* - marked **2** in the Figure) and the Concrete Interface is obtained from Abstract Interface. Finally, in the third transformation phase (*Transformation Rules 2* - marked **3** in the Figure), Final Interface is automatically generated depending on the chosen RIA rendering technology (e.g. Lazslo, Flex, DHTML+Ajax, XAML). The results of phases **1** and **2** can be refined by designers to achieve their goals according to the application needs.

5 Connecting RUX with WebML

From an abstract point of view, the role of the Connection Rules is to extract information from the Data and Business Logic models. The data model is mainly for allowing Abstract Interface to identify the different types of media specified in the Web design. The hypertext is mainly used to recognize and categorize the information grouping and to extract the information needed to trigger at run time the operation chains defined in the underlying Web application. Parts of the information retrieved are used in building the Abstract Interface, while other parts (e.g., marked **L** in Figure 8) are propagated to other RUX interface levels (e.g. to trigger the operations).

Finally, the connection process must know the pattern used for transforming each Web model in the final Web application in order to trigger the operation chains defined in the underlying business logic from the RUX Final Interface.

5.1 Building the Abstract Interface

The connection process starts by selecting the set of Connection Rules to be applied according to the underlying Web model to be used (that is WebML in this case). The Connection Rules (marked **1** in Figure 8) filter the information offered by WebML (Figure 8- left), obtaining the information needed to be used in each phase of RUX (for building Abstract, Concrete and Final Interfaces; Figure 8 - right). The WebML ***<Structure>*** (Data design) and ***<Navigation>*** (Hypertext design) are the main elements used, avoiding the rest of information regarding Presentation.

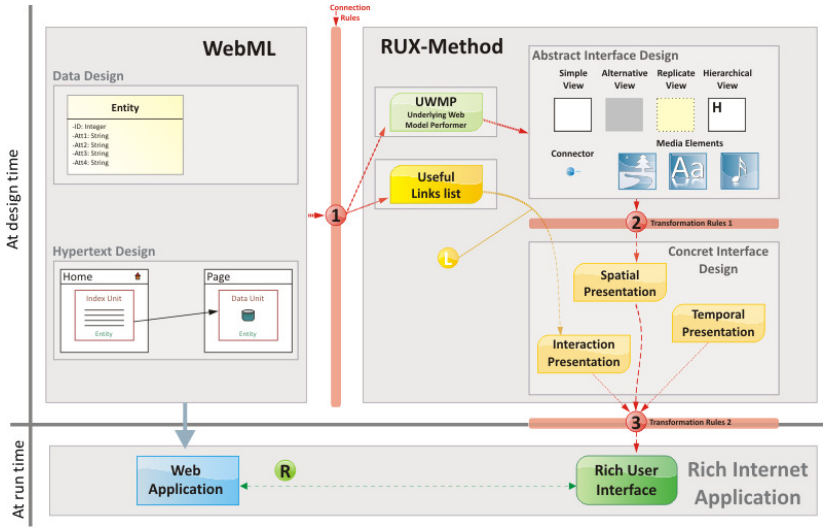

Figure 1 Schema of the connection process in WebML

From a specific point of view and regarding the presentation composition, the connection process for WebML is performed surfing top-down the WebML hierarchy matching each element according to the relation pattern established by the RUX Connection Rules.When building the Abstract Interface the UWMP (Underlying Web Model Performer – Figure 8 top center) has the objective to recognize all the WebML grouping concepts and classify them according to the potential correspondences with the RUX Abstract Interface elements (Figure 8 top right). After applying the Connection Rules the designer gets a first draft of the Abstract Interface, obtaining a hierarchical categorization for the Abstract Interface elements which retain the composition expressed by WebML Hypertext model. The designer can then refine this first draft to reach the UI goals.

The WebML hierarchy can be separated from the containers (marked as source in the rows **b** and **c** in Figure 9) to the units (**d** and **e** in the Figure). Due to the fact that the WebML navigation model can be composed by many Siteviews, we need to modularize these alternative groups of information. In this sense, the first step that takes place in the connection process is to create a basic empty presentation abstract model. Then, an Alternative view is inserted as root element containing a Simple

view for the content of every defined Siteview. Later on, the content placed in each one of these Siteviews is processed (e.g., **c**, **d**, **e**, **f**, and **g** in Figure 9) using the algorithm shown in Table 1.

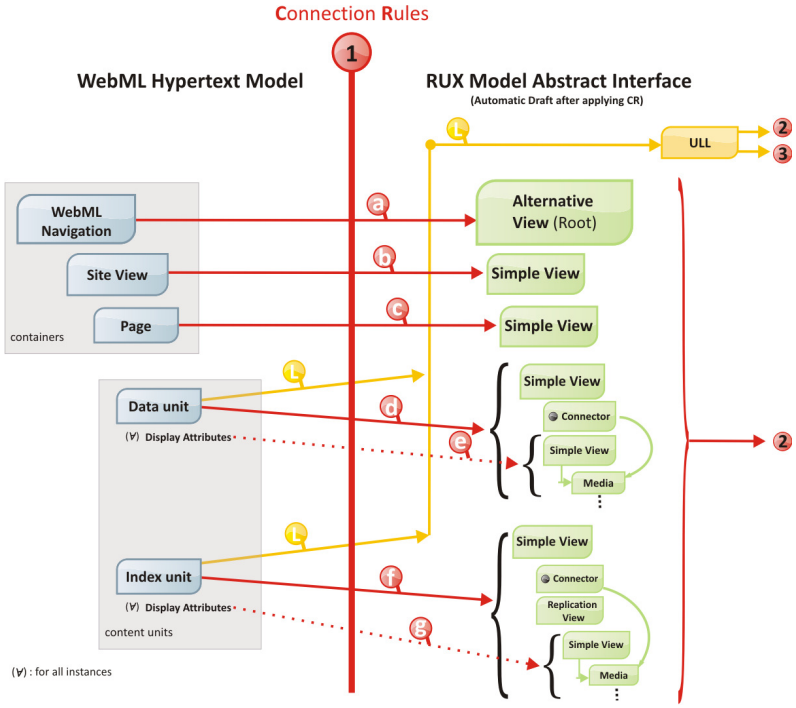

Figure 9 WebML Hypertext (left)/RUX Abstract Interface (right). A chunk of the matching process

Table 1. Connection Rules pseudocode.

```
ConnectionRules( AI : AbstractInterface, WML : WebML_Element )
Vars
  AIE : AbstractInterfaceElement
  AIC : AbstractInterface Connector
  AIV : AbstractInterface SimpleView
Begin
  If WML is SITEVIEW or ALTERNATIVE
    AIE ← AI.new_alternative_view( WML.name )
  EndIf
  If WML is PAGE or AREA
    AIE ← AI.new_simple_view( WML.name )
  EndIf
  If WML is CONTENT_UNIT
    AIE ← AI.new_simple_view( WML.name )
    AIC ← AI.new_connector( WML.id )
    AIE.insert_connector( AIC )
     If WML is (Index-unit orMultiData-unit)
       AIV ← AI.new_replicate_view( WML.name)
    EndIf
    AIE.insert_view( AIV )
  EndIf
```

```
ForEach( Descendant in WML )
    If Descendant is DisplayAttribute
      NV (New View): SimpleView ← AI.
        new_simple_view( Descendant.name )
      NC (New Component): MediaComponent ← AI.
        new_media_component( Descendant.name, Descendant.type or TEXT ) )
      NC.connect( AIC, Descendant.attr_name )
      NV.insert_element( NC )
      AIV.insert_view( NV )
    Else
      AIE.insert_element( ConnectionRules( AI, Descendant ) )
    EndIf
  EndForEach

  Return AIE
End
MAIN
Vars
  AI : AbstractInterface
  Root : SimpleView ← AI.new_simple_view( "root" );

Begin
  Foreach( Descendant in Navigation )
    Root.insert_element( AI,  ConnectionRules( Descendant ) )
End
```

The algorithm works following a basic rule: if the Page (i.e. <Page> node) contains only one Unit then it is transformed directly to (an) Abstract Interface Component(s) according to the Connection Rules. This avoids creating those Simple views that only contain a Simple view inside. If the Page contains more than one Unit, a RUX simple view (**c** in Figure 9) is created. This Simple view contains the results of WebML Unit processing. <Page> and <Area> are treated in a similar way by the Connection Rules.

Hypertext content units include the relation with the Data model through the ID attributes. Accessing IDs the Connection Rules are able to retrieve the type of each entity attributes. All the nodes of the hierarchy defined in <*Navigation*> will be transformed according to the Connection Rules, connecting with their identifiers of connectors described in <*Structure*>. Next we show the process for those most relevant units:

- Data unit: A Simple view is first created for a Data unit (**d** in Figure 9).The Data unit data source is related with a connector in the Abstract Interface (every data unit has a property called "*entity*" to denote the data source associated in the data model). According to the set of its attributes (recognized as displayable in the hypertext model; **e** in Figure 9) an equivalent set of Media elements is placed in the Simple view.
- Multidata unit and Index unit can be treated similarly (**f** in Figure 9). These content units are used to show a set of entity instances. In the same way, every Index unit has a property called "*entity*" to denote the data source associated. Firstly, when the connection process finds this kind of unit, it produces a Simple view. Later, a Repeater view is created inside containing a Simple view for each related instance in order to group instance attributes (**g** in Figure 9).
- In WebML the Entry unit is usually used for receiving information from the user, collecting input values into fields. The Entry unit needs a field for input data, establishing an input Media element in the Abstract Interface for each one. The Entry unit uses the link subnode to express the action to carry out the operation related to the information collected.
- The WebML Hierarchical Index unit is conceived as a special type of Index unit that shows hierarchies of entity instances. When it appears, the connection process creates a Simple view. Then, it creates a Repeater view inside to collect the information offered by the instances in every level. For each level, it first creates a Simple view to put inside the information of instances attributes via Media elements. Secondly, it creates another Repeater view for each Hierarchical index level and so on.

Next we show a BNF-like notation to define the syntax and the order followed in the connection process according to the underlying WebML elements. Keeping in mind the algorithm illustrated in Table 1 we also use the words *AI* (AbstractInterface) and *WML* (WebML_Element) in this syntax.

<AI> := <WML>(<WML>*)

<WML> := <SiteView>(<SiteView>*)

<SiteView> := Alternative_View_RUX <Area>(<Area>*) | Alternative_view_RUX <Page> (<Page>*)

<Alternative> := Alternative_view_RUX | Alternative_view_RUX <Page>(<Page>)

<Area> := Simple_view_RUX | Simple_view_RUX <Pages> (<Page>*)| Simple_view_RUX <Alternative>(<Alternative>)

<Page> := Simple_view_RUX | Simple_view_RUX <Content_unit>(< Content_unit >*)

<Content_unit>:=(<Data_unit>|<Index_unit>|<MultiData_unit>|<Entry_unit>|<Hierarchical_unit>|<MultiChoice_unit>) Connector_RUX

5.2 Application scenario in brief

Figure 10 focuses on "Menu Photos" to explain how the transformations are carried out in the application scenario. WebML "Menu Photos" page becomes (Figure 10 a arrow) the "Menu Photos" Simple View in the RUX Abstract Interface.

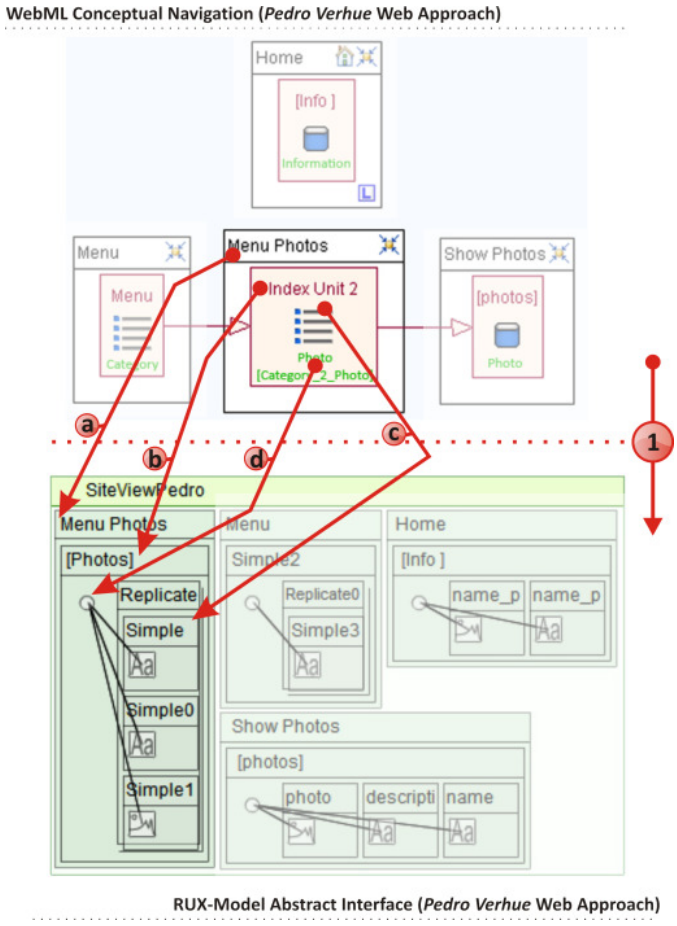

Figure 10 From WebML hypertext Model to RUX Abstract Interface

"Index Unit 2", that uses photo entity from the WebML structure, becomes (Figure 10 b arrow) the "[Photos]" Simple View with a Replicate View inside to perform the multiple data instances provided by the WebML "Index Unit 2". Finally, for the group of attributes available in each instance offered by the "Index Unit 2", the process creates (Figure 10 arrows c and d) one Media element with a common

Connector inside a Simple View. This way three Media elements are created for the three string fields placed in Photo entity. The connector related to such entities is also created.

5.3 Propagating the rest of information to the following interface levels

The information regarding links is retrieved and processed to generate a ULL (Useful Link List) (Figure 8-center) according to the different kinds of Actions defined in RUX Model, (e.g. UIActions, CallActions) [17]. WebML is Web 1.0 oriented, so the elements contained in the ULL are always Handlers and CallActions, which are able to describe simple user interaction, like a mouse click, to trigger a hypertext link. This information is included here due to it is also produced after applying the connection rules but it is out of the scope of this paper to apply it to our particular application scenario.

From a specific point of view and regarding the operation chains, in the WebML description language every unit is marked by a unique identifier called "ID". In order to build the ULL from the links appearing in the hypertext, the connection process needs to review each unit description because the links are described inside the source unit, while the target unit is defined in the link node attributes (i.e. *to="target unit ID*"). Hence, for each link, the ULL is composed by the link name (using its original ID link), the "from" concept (using the ID of the source unit that contains such link) and the "to" property (that appears defined in the link).

ULL is used for understanding the outgoing links in every WebML content unit. The ULL is propagated (marked **L** in Figure 8) to the Concrete Interface level for building the handlers and CallActions in the Interaction Presentation. Triggering the business logic is solved as in [5], using the "pointing" links, given that WebML links use the standard HTTP GET format: *pageid.do?PL* where *pageid* denotes a Web page and *PL* a list of parameters (tag-value pairs separated by the '&' character).

6 Implementation in brief and experimental results

In practical terms, the rich user interfaces developed with RUX-Method are supported by RUX-Tool [32]. At the moment, RUX-Tool works with WebRatio 5.0 in commercial and academic terms allowing the design of rich UIs over Web applications designed and generated with WebRatio. RUX-Tool is browser-based, it works online and the generated final application is automatically deployed online. Currently, the available UI code generators are FLEX, Laszlo and AJAX.

RUX-Tool is built as a distributed Rich Internet application. The logic needed for designing the rich user interface is performed at client side. On the contrary, the logic required for retrieving the underlying WebML model information and the transformation between phases are carried out at server side. The general architecture of RUX-Tool is depicted in Figure 11.

The client side layer includes a graphical UI for designing the Abstract Interface, Concrete Interface and many other issues such as importing the WebML project or configuring the code generation options among others. Both client and server side layers produce an internal representation of each model by means of XML specifications.

The RUX-Tool layers, the Component Library and the code generator have a plug-in architecture for allowing the inclusion of new components where each component is a template based on Velocity

Style Language. Such architecture has been already used to extend the Component Library to take into account additional languages for the final UI code generation.

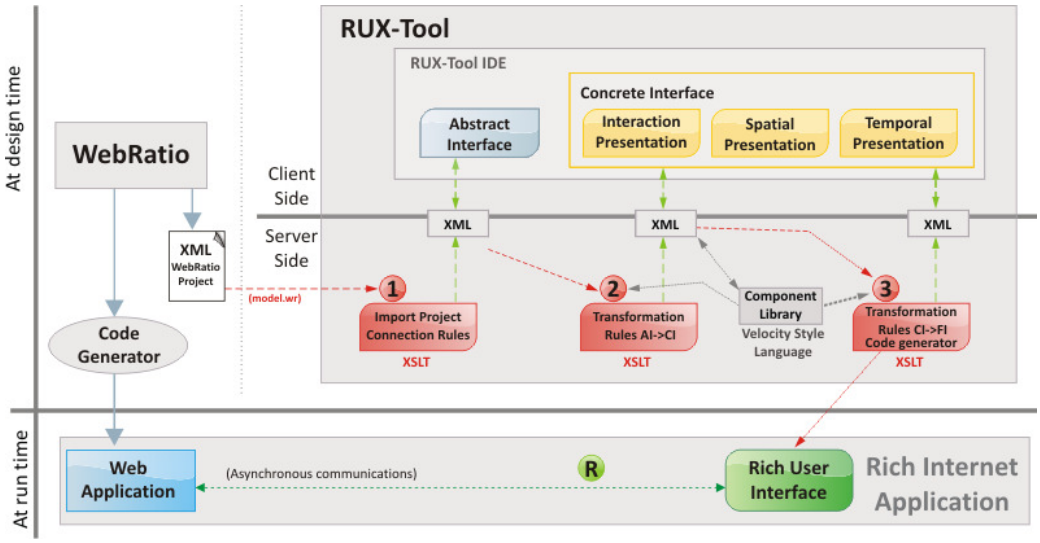

Figure 11 RUX-Tool Architecture

As an experimental result, we recommend a particular full example of a virtual store called Acme. Acme [31] is an unreal small furniture company for selling products on internet using a Web site for publishing a product catalog. The Web application offers information about products, offers of the day (composed by combinations of products to be sold at a discounted price), and stores. We have developed this Web 1.0 application using Webratio. Then, by means of RUX-Tool a new richer UI has been designed. This new UI has been deployed using different rendering technologies (Ajax [29] and Laszlo [30]), sharing the same RUX Abstract interface and Concrete interface, as well as the same WebML Data and Hypertext models in the sense described in this paper.

The reader is referred to [32] for more details about RUX-Tool where many videodemos, examples and so on are available.

7 Conclusions

RUX is a Model Driven Method for the systematic evolution and enrichment of HTML-based Web applications based on Models to Web applications with User Interfaces based on RIA technologies. RUX provides different interface levels and transformation phases. Among them, this paper has focused on the connection process between RUX and the underlying Web model, using WebML as a particular case.

The objectives of the connection process are twofold: on the one hand, building an Abstract Interface based on existing data, business logic and presentation models offered by the underlying Web model being enriched; on the other hand, extracting relevant information regarding the operation

chains to specify handler and actions in the Concrete Interface level. This information is propagated to other phases of RUX which are out of the main scope of this paper.

To our knowledge, there is no other work addressing the connection with different Web models to enrich their UIs with RIA features. However, several works have been identified to be reasonably related because they extend existing Web models to include some RIA features, or they apply interesting ideas coming from the HCI field.

The connection process is the only part of RUX that depends on the selected Web Model. Although here applied to WebML, the process is general enough to be used with other approaches. However, in practical terms, each model expresses its Web concepts in a different way. This makes necessary a different set of rules adapted to each Web model at least at implementation level. However we are following the work developed in [21] that tries to build a common metamodel for the interoperability of different Web Models. If this work succeed, it could be possible to have a generic connection process for all the Web Models supported by this metamodel.

For the sake of simplicity, a simple application scenario has been shown to describe the connection process. First, the application has been modelled with WebML and latter a new UI has been designed with RUX by using the information already provided by WebML.

From a practical point of view, the connection process is implemented in RUX-Tool, the RUX CASE tool. Currently, RUX-Tool is being used together with Webratio 5.0 in the design of applications with a RIA-based UI. Moreover, it is being used for evolving the HTML-based UIs of already deployed applications to RIA-based UIs. RUX-Tool is accessible online and provides code generators for FLEX, Laszlo and AJAX.

RUX Method has been also successfully combined with UWE [35]. UWE describes a systematic design methodology for Web applications exclusively using UML techniques, the UML notation and the UML extension mechanisms. For the specific Web concepts, it uses a UML extension that consists of three steps that are performed in an iterative design process. These steps are the conceptual, navigational and presentational design. For our connection process we mainly use the conceptual and navigation information that is also specified in the UWE presentation model. After applying the connection rules over the UWE models, the designer will obtain a first draft of RUX Abstract Interface.

Regarding the specific set of connection rules to connect UWE with RUX, the Conceptual Model phase in UWE corresponds with the Data design phase in WebML. The idea of Entities used in WebML can be related to Classes in UWE while the WebML Hypertext design phase is related to the UWE Navigational phases. Concerning the navigation, the concept of Navigation Class node used in UWE can be treated similarly to the Content Units defined in WebML.

Acknowledgements

The authors wish to acknowledge the collaborative funding support from the grants no. PDT06A042 and TIN2005-09405-C02-02. In particular we wish to thank Rober Morales-Chaparro and Giovanni Toffetti Carughi for their contributions to this research project.

References

1. Bebjak, M., Vranic, V., Dolog, P., "Evolution of Web applications with Aspect-Oriented design patterns". *Proc. of the 2rd International Workshop AEWSE'07*, Como, Italy, July 2007, 80-86
2. Bozzon, A., Comai, S., Fraternali, P., Carughi, G.T., "Conceptual modeling and code generation for rich internet applications". In *Proceedings of the 6th international Conference on Web Engineering* (Palo Alto, California, USA, July 11 - 14, 2006). ICWE '06, vol. 263. ACM, New York, NY, (2006) 353-360.
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J., "A Unifying Reference Framework for Multi-Target User Interfaces". *Interacting with Comp.*, Vol. 15, No. 3 (2003) 289-308
4. Ceri, S., Daniel, F., Matera, M., Facca, F. M., "Model-driven development of context-aware Web applications". *ACM Trans. Inter. Tech.* 7, Feb. 2007
5. Ceri, S., Dolog, P., Matera, M., Nejdl, W. "Model-driven design of web applications with client-side adaptation". In *Proceedings of the 4th International Conference on Web Engineering (ICWE), Lecture Notes on Computer Science, 3140*, (2004), 201-214.
6. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M. 2002 *Designing Data-Intensive Web Applications*. Morgan Kaufmann Publishers Inc.
7. Conallen, J. 1999. *Modeling Web application architectures with UML*. Commun. ACM 42, 10, 63-70
8. Daniel, F., Matera, M., Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., "Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities". *Internet Computing, IEEE*, vol.11, no.3, pp.59-66, May-June 2007
9. De Troyer, O., Casteleyn, S., Plessers, P., "WSDM: Web Semantics Design Method", bookchapter in *Web Engineering: Modelling and Implementing Web Applications*, Human-Computer Interaction Series Vol. 12, pp. 303-352, Eds. Gustavo Rossi, Oscar Pastor, Daniel Schwabe, Louis Olsina, Publ. Springer, ISBN 978-1-84628-922-4, (2007)
10. De Virgilio, R., Torlone, R., "A general methodology for context-aware data access". In *Proceedings of the 4th ACM international Workshop on Data Engineering For Wireless and Mobile Access* (Baltimore, MD, USA, June 12 - 12, 2005). MobiDE '05. ACM, New York, 9-15.
11. Fiala, Z., Hinz, M., Meissner, K., "Developing component-based adaptive Web applications with the AMACONTBuilder". *Web Site Evolution, 2005. (WSE 2005). Seventh IEEE International Symposium on* , vol., no., pp. 39-45, 26 Sept. 2005
12. Garrigós, I., Cruz, C., Gómez, J., "Aprototype tool for the Automatic Generation of Adaptative Websites". *Proc. of the 2rd International Workshop AEWSE'07*, Como, Italy, July 2007, 13-27
13. Gómez, J., Cachero, C., Pastor, O., "Conceptual modeling of device-independent Web applications". *Multimedia, IEEE* , vol.8, no.2, pp.26-39, Apr-Jun 2001
14. Houben, G.J., van der Sluijs, K., Barna, P., Broekstra, J., Casteleyn, S., Fiala, Z., Frasincar, F., book chapter: *Web Engineering: Modelling and Implementing Web Applications*, G. Rossi, O. Pastor, D. Schwabe, L. Olsina (Eds), Chapter 10, p. 263-301, 2008, Human-Computer Interaction Series, Springer.
15. Koch, N., Kraus, A., "The expressive Power of UML-based Web Engineering", *Proc. of IWWOST02*, CYTED, pp. 105-119.
16. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Lopez, V., "UsiXML: a Language Supporting Multi-Path Development of User Interfaces". *IFIP Working Conference on Engineering for HCI.* LNCS. 2005. 207-228
17. Linaje, M., Preciado, J.C., Sanchez-Figueroa, F., "Engineering Rich Internet Application User Interfaces over Legacy Web Models". *Internet Computing, IEEE*, vol.11, no.6, pp.53-59, 2007
18. Lucca, G. A., Fasolino, A. R., Pace, F., Tramontana, P., de Carlini, U., "WARE: A Tool for the Reverse Engineering of Web Applications". In *Proceedings of the Sixth European Conference on*

Software Maintenance and Reengineering (March 11 - 13, 2002). CSMR. IEEE Computer Society, Washington, DC, 241.
19. Martínez-Ruiz, F., Muñoz Arteaga, J., Vanderdonckt, J., González-Calleros, J. M., "A first draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications". *Web Congress, 2006. LA-Web '06. Fourth Latin American* , vol., no., pp.32-38, Oct. 2006
20. Mesbah, A., van Deursen, A., "Migrating Multi-page Web Applications to Single-page AJAX Interfaces". *Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference on* , vol., no., pp.181-190, 21-23 March 2007
21. Moreno, N., Fraternali, P., Vallecillo, A., "WebML modeling in UML". *IET Software Journal.* Vol. 1, Number 3, 2007.
22. Preciado, J.C., Linaje, M., Sánchez-Figueroa, F., "An approach to support the Web User Interfaces evolution". *In proceedings of AEWSE, First International Workshop on Adaptation and Evolution in Web system Engineering*. ICWE 2007
23. Preciado, J.C., Linaje, M., Comai, S., Sánchez-Figueroa, F., "Designing Rich Internet Applications with Web Engineering Methodologies". *Web Site Evolution, Seventh IEEE International Symposium on*, *2007. WSE 2007. 9th*, vol., no., pp.23-30, 5-6 Oct. 2007
24. Preciado, J.C., Linaje, M., Sanchez, F., Comai, S., "Necessity of methodologies to model rich Internet applications". *Web Site Evolution, 2005. (WSE 2005). Seventh IEEE International Symposium on* , vol., no., pp. 7-13, 26 Sept. 2005
25. Bandelloni, R., Mori, G., Paternò, F., Santoro, C., Scorcia, A., "Web User Interface Migration through Different Modalities with Dynamic Device Discovery". *Proc. of the 2rd International Workshop AEWSE'07*, Como, Italy, July 2007, 58-72
26. Schwabe, D., Rossi, G., and Barbosa, S. D., "Systematic hypermedia application design with OOHDM". In *Proceedings of the the Seventh ACM Conference on Hypertext* (Bethesda, Maryland, United States, March 16 - 20, 1996). HYPERTEXT '96. ACM, New York, 116-128
27. Stearn, B., "XULRunner: A New Approach for Developing Rich Internet Applications". *Internet Computing, IEEE* , vol.11, no.3, pp.67-73, May-June 2007
28. Urbieta, M., Rossi, G., Ginzburg, J., Schwabe, D., "Designing the Interface of Rich Internet Applications". *Web Congress, 2007. LA-WEB 2007. Latin American* , vol., no., pp.144-153, Oct. 31 2007-Nov. 2 2007
29. URL ACME Ajax, WebRatio+RUX-Tool: http://ruxproject.org/www/projects/AcmeWebv2.0/index.html
30. URL ACME Laszlo, WebRatio+RUX-Tool: http://ruxproject.org:8080/lps/projects/AcmeWebv2.0/index.lzx?lzt=swf
31. URL ACME WebRatio: http://ruxproject.org:8080/hide.acme
32. URL RUX Project: http://www.ruxproject.org
33. URL TeresaXML: http://giove.cnuce.cnr.it/teresa/teresa_xml_aui.html
34. URL UiML: http://www.uiml.org/
35. URL Preciado, J.C., Linaje, M., Morales, R., Sánchez-Figueroa, F., Zhang, G., Kroiß, C., Koch, N., "Designing Rich Internet Applications Combining UWE and RUX-Method". In *Proceedings of the 8th international Conference on Web Engineering* (Yorktown Heights, New York, USA, July 14 - 18, 2008). ICWE '08, IEEE CS Press / also available in http://www.webengineering.org/