An approach to support the Web User Interfaces evolution

Preciado, J.C.; Linaje, M.; Sánchez-Figueroa, F.

Quercus Software Engineering Group
Escuela Politécnica. Universidad de Extremadura (10071 – Cáceres, Spain)
{jcpreciado; mlinaje; fernando}@unex.es

Abstract. Currently, there is a growing group of Web 1.0 applications that is migrating towards Web 2.0 where their data and business logic could be maintained but their User Interface (UI) must be adapted. Our proposal facilitates the adaptation of existing Web 1.0 applications to Web 2.0, focusing on UIs and taking advantage of functionality already provided by legacy Web Models. In this paper we present, as an example, how to adapt applications already modelled with WebML to RUX-Model, a method that allows designing rich UIs for multi-device Web 2.0 UIs. One of our main goals in attending the workshop is discussing other potential adaptations for applications modelled with OOHDM, UWE or OO-H among others.

Keywords: Web Engineering, Adaptation, User Interfaces, Web 1.0, Web 2.0, Rich Internet Applications.

1 Introduction

Over the past few years, the traditional HTML-based Web Applications (Web 1.0) development has been supported by different models and methodologies coming from the Web Engineering community.

Nowadays, the complexity of activities performed via Web User Interfaces (UIs) keeps increasing and ubiquity becomes fundamental in a growing number of Web applications. In this context, many Web 1.0 applications are showing their limits to reach high levels of interaction and multimedia support, so many of them are migrating towards Web 2.0 UIs.

The majority of the features of Web 2.0 UIs may be developed using Rich Internet Applications (RIAs) technologies [8] which combine the benefits of the Web distribution model with the interface interactivity and multimedia support available in desktop applications. UI development is one of the most resource-consuming stages of application development [2]. A systematic development approach would decrease resource usage. However, there is still a lack of complete models and methodologies related with RIA [1]. An interesting partial proposal can be found in [11].

Our statement is that it is not only important to develop applications for Web 2.0 from scratch, but it is also important to adapt existing Web 1.0 applications based on Web Models to the new requirements and necessities following a methodology.

In this paper we use RUX-Model (Rich User eXperience Model) [9], a Model Driven Method for engineering the adaptation of legacy Web 1.0 applications to Web 2.0 UI expectations [8]. A case study is presented using the RUX-Model CASE Tool (RUX-Tool, available at *http://www.ruxproject.org*), which supports RIA UI code generation and it is validated by implementation.
RUX-Model proposes three UI transformation phases that we describe in Section 2. However, the main contribution of this paper focuses on the definition of the connection with the Web Model to be adapted (Section 3).

2 RUX-Model Overview

RUX-Model is an intuitive visual method that allows designing rich UIs for RIAs and its concepts are associated with a perceptive graphical representation. In the context of adapting existing Web 1.0 applications to Web 2.0, RUX-Model can be seen as an adaptor as it is depicted in Figure 1 (*left*). Due to it being a multidisciplinary proposal and in order to decrease cross-cutting concepts, the UI specification is divided into levels. According to [3] an interface can be broken down into four levels, Concepts and Tasks, Abstract Interface, Concrete Interface and Final Interface. The RUX-Model process starts from Abstract Interface and each Interface level is composed by Interface Components. Concepts and Tasks are taken by RUX-Model from the underlying legacy Web Model.
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 have the same Abstract Interface.
Abstract Interface elements are: ***Connectors***, we have included them to establish the relation to the data model once the hypertext model specifies how they are going to be recovered; ***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; ***Views***, a view symbolizes a group of information that will be shown to the client at the same time. In order to group information, RUX-Model allows the use of four different types of containers: simple, alternative, replicate and hierarchical views.
Then in Concrete Interface we are able to optimize the UI for a specific device or group 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 of the Interface Components. Temporal Presentation allows the specification of those behaviours which require a temporal synchronization (e.g. animations). Interaction Presentation allows modelling the user's behaviour with the UI.
The RUX-Model process ends with Final Interface which provides the code generation of the modelled application. This generated code is specific for a device or a group of devices and for a RIA development platform and it is ready to be deployed.
RUX-Model adaptation process from Web 1.0 applications to Web 2.0 has three different transformation phases. Figure 1 (*left*) shows the different interface levels and

transformation phases. The first transformation phase (Connection Rules), marked as **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. Then, the second phase is performed, marked as **2** in Figure 1 where Concrete Interface is automatically obtained from Abstract Interface. Finally, in the third phase, marked as **3** in the figure, Final Interface is automatically generated depending on the chosen RIA rendering technology (e.g. Lazslo, Flex, Ajax, XAML). Phases **1** and **2** can be improved by modellers to achieve their goals according to their needs.

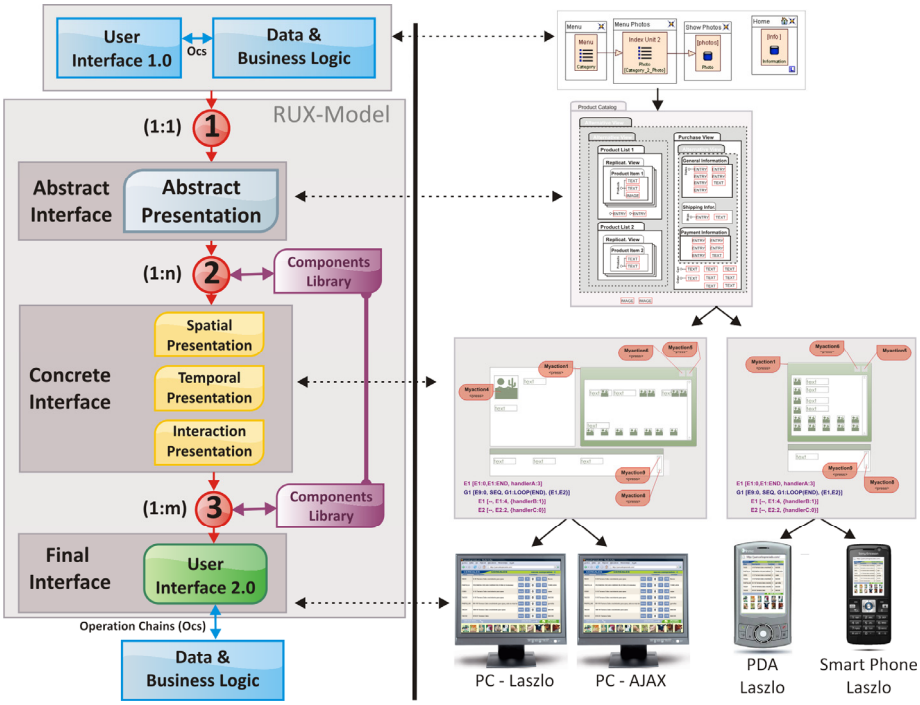

Fig 1. *Left*: RUX-Model architecture overview; *Right*: Example of RUX-Model method

In Figure 1 (*right*) we show the Interface levels and the transformation phases, but here from a practical point of view. In this figure, RUX-Model obtains Abstract Interface automatically by means of a connection to an existing Web application developed using a Web Model (e.g. WebML [4]). This Abstract Interface is transformed into two Concrete Interfaces, one with a special arrangement adapted for PCs and the other for a group of mobile devices. One Concrete Interface is transformed into two Final Interfaces for a PC using different technologies (one uses Laszlo and the other uses AJAX) and the other Concrete Interface into two Final Interfaces for similar devices (e.g. PDA and Smartphone) using Laszlo rendering technology.

3 Web Model to RUX-Model Adaptation

From now on, we focus on the adaptation process to the Web Model being adapted. RUX-Model connection process takes from the connected Web Model two kinds of information regarding its structure and navigation. Information regarding the presentation model is not considered in RUX-Model because Web 1.0 presentation models are not oriented to Web 2.0 UIs (e.g. no single page application, no partial UI refreshment, etc).The structure and navigation are for allowing Final Interface triggering the Operation Chains defined in the underlying Web application being adapted and for building the RUX-Model Abstract Interface.
The connection process starts selecting the set of Connection Rules, phase **1** in the Figure 1, according to the Web Model that we have chosen. A set of Connection Rules exists for each potential Web Model being considered (e.g. WebML [4], OOHDM [5], UWE [6], OO-H [7] or Hera [12] among others).

3.1 WebML Specific Case

The selection of WebML [4] as the Web Model for this case study is based on previous studies [1]. WebML allows modellers to express conceptual navigation and business logic of the Website. WebML is supported by a CASE tool called WebRatio (http://www.webratio.org) that generates application code. This code is based on JSP templates and XSL style sheets for building the application's presentation.
Regarding the triggering of Operation Chains, this is solved as in [10], using the "pointing" links, given that WebML links use the typical HTTP GET format: *pageid.do?PL* where *pageid* denotes a Web page and *PL* a list of tag-value pairs.

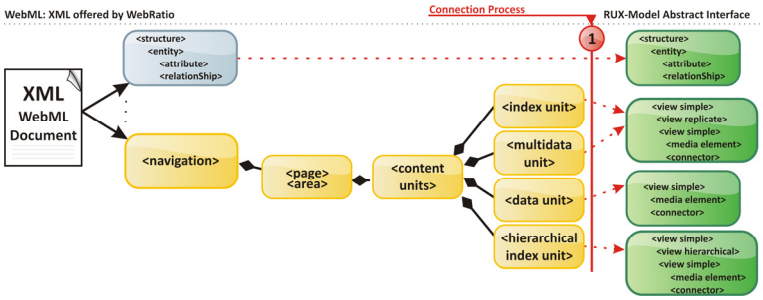

Fig 2. WebML Connection process schema

Regarding the building of Abstract Interface, it is important to note that all the concepts of WebML are associated with a graphical notation and a textual XML syntax. WebML XML is composed of several tags (and content). Next we show those ones most relevant for our connection process.

- **<Structure>**: related to Entity, Attribute and Relationship,
- **<Navigation>**: related to containers (<Siteview>, <Area> and <Page>) and units (<ContentUnits>, <DataUnit>, <IndexUnit>, <HierachicalIndexUnit>, <MultidataUnit> and <EntryUnit>) to express and to organize the Web Model.

This information is used all along the RUX-Model designing process as depicted in Figure 2. The Connection Rules filter the information offered by WebRatio, obtaining only the information needed to build Abstract Interface, that is the **<Structure>** and **<Navigation>** elements.
Due to the fact that the WebML navigation model is composed by several siteviews, the first step that takes place in the connection process is to create a basic empty presentation abstract model, in order to insert in it an alternative root view that will contain a simple view for each defined site view. Later on, for each one of these site views, we will process the content placed in each page using the algorithm whose pseudocode is shown in Table 1.
The algorithm works following a basic rule: if the page contains only one WebML Unit it is transformed directly to Abstract Interface Component(s) according to the Connection Rules. If the page contains more than one Unit, a RUX-Model simple view (Figure 2) will be created. This simple view will contain the results of WebML Unit processing. <Page> and <Area> are treated in the same way.
All the nodes contained in <Structure> are used as in the previous Web Model, using their original connectors. All the nodes of the hierarchy defined in <Navigation> will be transformed according to the Connection Rules, calling to their identifiers of connectors described in <Structure>.

Table 1. Connection Rules pseudocode.

```
ConnectionRules( AI : AbstractInterface, WML : WebML_Element )
Vars
   AIE : AbstractInterfaceElement
   AIC : Connector
   AIV : 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 DATA_UNIT ...
         AIV ← AI.new_simple_view( WML.name )
      Else
         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 ← Root.insert_element( AI,  ConnectionRules( Descendant ) )
End
```

3.3 Case Study

With the aim of validating our proposal, we show a simple real-life case study. This case study is inspired by the *"Pedro-Verhue"* Website *(http://www.pedro-verhue.be),* an on-line catalogue for home interior decoration, based on RIA technologies to provide high interaction and presentation capacities.
Due to the case study extension, we only focus on the Connection Rules that is the main objective of this paper. Notwithstanding, the full engineering process is available on-line through a video tutorial and the Web 2.0 application is deployed at *http://www.ruxproject.org*.

At the top of Figure 3 the underlying Web Model is depicted (i.e. WebML hypertext model) and at the bottom the RUX-Model Abstract Interface automatically obtained using the connection process (i.e. Connection Rules).

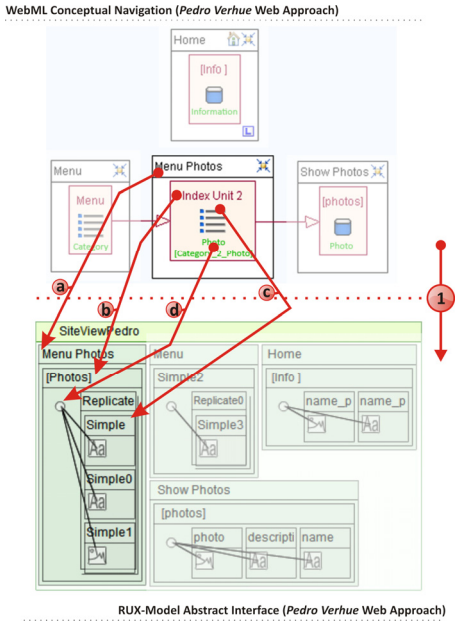

Fig 3. From WebML hypertext Model to RUX-Model Abstract Interface.

Mainly, *Pedro-Verhue* has a first level category (called "Menu" in Figure 3) that *onmouseover* displays a second level category (called "Menu Photos" in Figure 3) in order to show the photograph index. When one of the photographs is selected, the detailed information (called "Show Photo" in Figure 3) is shown to the user. All this process is carried out in a single page, following RIA concepts.
Figure 3 focuses on "Menu Photos" to explain how the transformation is carried out. WebML "Menu Photos" page becomes (Figure 3 arrow *a*) the "Menu Photos" *Simple View* in the RUX-Model Abstract Interface. "Index Unit 2", that uses *photo* entity from the WebML *structure*, becomes (Figure 3 arrow *b*) the "[Photos]" *Simple View* with a *Replicate View* inside. Finally, for each attribute available in the WebML "Index Unit 2" the process creates (Figure 3 arrows c and d) one *Media* element with a common *Connector* inside a *Simple View*.

4 Conclusions and Future Work

In this paper we use RUX-Model (Rich User eXperience Model) [12], a Model Driven Method for the systematic adaptation of RIAs UIs over existing HTML-based Web Applications based on Models in order to give them multimedia support, offering more effective, interactive and intuitive user experiences.

Among the transformation phases proposed in RUX-Model, we have focused on the definition of the connection process with the Web Model being adapted. This phase is crucial in the process, due to it being the only part of RUX-Model that depends on the Web Model selected.
We have proposed a case study to demonstrate our approach in practical terms using RUX-Tool. Currently, RUX-Tool is able to take advantage of applications generated using WebML and it is able to auto-generate RIA UIs code for several extended rich client rendering technologies such as AJAX (using DHTML) and OpenLaszlo[8].
At the implementation level, RUX-Tool has a series of prerequisites about the models that can be used in order to extract from them all the information automatically. Moreover, conceptually, RUX-Model may be used on several existing Web Models such as OOHDM, UWE, OO-H or HERA among others. Discussing this issue is one of our main objectives at the workshop.

Acknowledgments. Partially funded by PDT06A042 and TIN2005-09405-C02-02.

References

1. Preciado, J.C., Linaje, M., Sánchez, F., Comai, S.: *Necessity of methodologies to model Rich Internet Applications*, IEEE International Symposium on Web Site Evolution (2005) 7-13
2. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: *Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities*, Journal on Internet Computing, IEEE (2007) vol. 11 iss. 3 59-66
3. 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) vol. 3425 207-228
4. Ceri S., Fraternali P., Bongio A., Brambilla M., Comai S., Matera M.: *Designing Data-Intensive Web Applications*, Morgan Kauffmann (2002)
5. Schwabe, D., Rossi, G., Barbosa, S. D.: *Systematic hypermedia application design with OOHDM*, ACM Conference on Hypertext, ACM Press (1996) 116-128
6. Koch N., Kraus A.: *The Expressive Power of UML-based Web Engineering*, International Workshop on Web-oriented Software Technology, Springer-Verlag (2002)
7. Gómez, J., Cachero, C.: *OO-H Method: extending UML to model web interfaces*, Information modeling for internet applications, Idea Group Publishing (2003)
8. Brent S.: *XULRunner: A New Approach for Developing Rich Internet Applications.* Journal on Internet Computing, IEEE (2007) vol. 11 iss. 3 67-73
9. Linaje, M., Preciado, J.C., Sánchez-Figueroa, F.: *A Method for Model Based Design of Rich Internet Application Interactive User Interfaces*, International Conference on Web Engineering, LNCS (2007), vol. 4607
10. Ceri S., Dolog P., Matera M., Nejdl W.: *Model-Driven Design of Web Applications with Client-Side Adaptation*, LNCS (2004) vol. 3140 201-214
11. Bozzon, A., Comai, S., Fraternali, P., Toffetti Carughi, G.: *Conceptual Modeling and Code Generation for Rich Internet Applications*, International Conference on Web Engineering, LNCS (2006), 353-360
12. Houben, G.J., van der Sluijs, K., Barna,P., Broekstra, J., Casteleyn, S., Fiala, Z., Frasincar, F.: *Hera*, Web Engineering: Modelling and Implementing Web Applications, Human-Computer Interaction Series (2007), Springer, vol. 12