Journal of Universal Computer Science, vol. 16, no. 15 (2010), 2038-2059
submitted: 31/1/10, accepted: 28/7/10, appeared: 1/8/10

Multi-Device Context-Aware RIAs Using a Model-Driven Approach

Marino Linaje, Juan Carlos Preciado, Fernando Sánchez-Figueroa
(Quercus Software Engineering Group, Universidad de Extremadura, Caceres, Spain
{mlinaje, jcpreciado, fernando}@unex.es)

Abstract: Model-Driven Development concepts are exhibiting as a good engineering solution for the design of ubiquitous applications with multi-device user interfaces and other context-aware capacities. The Web has become an ideal platform for the deployment of such applications and therefore traditional Web development techniques are rapidly adopting Model-Driven principles to cope with the adaptation issues imposed by context-awareness and multichannel solutions. This discipline is being known as Model Driven Web Engineering. However, at the same time that the use of the Web and the number of people with mobile devices is growing, users are demanding more and better user experiences through the user interface. Web vendors answered introducing Rich Internet Applications that take advantage of the single-page paradigm and expand traditional Web features, providing richer content types, richer controls, richer temporal behaviors, richer interactivity and richer communications. While many recent devices support some type of RIA technology, RIAs extended features are showing some limitations of Model Driven Web Engineering methodologies to cope with multi-device context-awareness at the presentation level. This paper presents the combination of two different methodologies, WebML and RUX-Method, both using MDD principles, to obtain multi-device context-aware Rich Internet Applications using a Model-Driven approach. While WebML provides context-awareness at the data and business logic levels, RUX-Method deals with the presentation issues introduced by Rich Internet Applications.

Keywords: Multi-device Context-Aware, User Interfaces, Rich Internet Applications, Web Engineering, Model-Driven Development
Categories: H.5.2, H.5.4, H.3.5, D.2.2

1 Introduction

The widespread diffusion of mobile devices with advanced capabilities and the great progress in communication and network technologies are creating an ideal environment for mobile applications. Users can access services and contents with any media, at any time and from anywhere.

However, mobile devices in today's market are very heterogeneous regarding their input and display capacities and software platform configurations. This leads to a complicated development process for applications having to face a wide and increasing range of devices.

Once the Web has solved some problems such as reliability of data communications, bandwidth consumption or cost efficiency, it has become an ideal platform for deploying mobile applications. The use of the Web solves some of the platform dependent issues mentioned above abstracting from technological concerns

(e.g., Operating System). In this sense, we could say that it is time for Multi-device Web applications.

However, users are demanding more and more systems that can sense their physical environment, i.e., their context of use, and adapt their behavior accordingly. This is usually known in the literature as context-awareness. A system is context-aware when it uses the context to provide relevant information and/or services to the user, where relevancy depends on the user task. Features for Context-Aware Applications include presentation of information and services to a user; automatic execution of a service; and tagging of context to information for later retrieval [Abowd, 99]. Under this situation we could say that it is time for Multi-device context-aware Web applications.

With no doubt, one of the main parts of the applications affected by all these features is the User Interface (UI). Adapting the UI of an application to a different context for different devices exploding all the device capabilities adds an extra effort to developers, while improving the potential User eXperience (UX) that could be achieved.

This extra effort depends on the way in which the UI is created. Currently there exist two main approaches for the development of multi-device IUs: on the one hand, those proposals where the UI specification is done for a concrete delivery context and, on the other hand, the so-called Multiple or Plastic User Interfaces [Seffah, 03] where the focus is on the transformation from abstract platform independent descriptions to concrete user interfaces for various platforms.

The former can fully exploit the device capacities. However, it is not a good solution for adaptation issues. The latter improves adaptation significantly; however, they do not allow the creation of UIs able to fully exploit the device capabilities. A mixed solution seems to be a good path to follow in order to obtain benefits from both approaches. Obtaining this mixed solution for context-awareness, rather than being a mere technological concern, represents a true design and modeling issue. And this is where Model Driven Development (MDD) comes to the scene, playing a pivotal role in the development of context-aware Web applications [Daniel, 09].

MDD refers to a development approach that is based on the use of models as a primary artifact during the development lifecycle. MDD pursues a clear separation of the business and application logic from the underlying execution platform technologies so that changes in the underlying platform do not affect existing applications and business logic can evolve independently from the technology. The Web Engineering discipline has adopted MDD giving way to which is known as Model Driven Web Engineering (MDWE) [Moreno, 07]. In this sense, we can give a step forward and say that it is time for Model Driven Development of Multi-device Context-Aware Web Applications.

However, the complexity of the tasks performed through Web applications is continually increasing. Modern Web solutions resemble desktop applications, enabling sophisticated user interactions, client-side processing, asynchronous communications and multimedia. The so-called Web 2.0, which demand a high degree of usability and powerful interactions, has definitively shown the limits of the HTML/HTTP, giving way to a new wave of Web applications known as Rich Internet Applications (RIAs).

One of the key issues for the rapid growing of RIAs popularity is their powerful presentation and interaction capabilities. In a typical RIA it is not necessary to switch the perspective among different pages/subpages when browsing due to the UI exploiting the single-page application paradigm, in which an interface container at a higher level coordinates the presentation behavior without intermediate blank pages that typically decrease the UX. The UI can be progressively downloaded, by loading parts of the presentation logic and UI elements at run time on demand. Extended interaction events (e.g., event listeners and handlers) and temporal behaviors (e.g., animations based on temporal relationships among interface components) can be defined to further improve the UX. Due to the possibility of downloading dynamically new behaviors, an application is not restricted to a fixed set of predefined standard UI controls and containers. RIAs can dynamically customize existing widgets, adapting them to specific usage contexts, e.g., to the rendering capacity of the display terminal using graceful degradation techniques. Many recent mobiles, consoles and new Web-enabled gadgets support at least one RIA technology (e.g., AJAX or Flash)

However, the good results achieved using MDD for traditional Web applications development do not transfer automatically to RIAs. The work in [Preciado, 05] showed that the methodologies coming from the Web, Multimedia and Hypermedia fields do not fit completely the new RIA paradigm. This poses new challenges from the point of view of context-awareness and multi-device, so we can affirm that it is time for Model Driven Development of Multi-device context-aware User Interfaces for Rich Internet Applications.

This is precisely the contribution of this paper, giving a further step towards obtaining multi-device context-aware RIA UIs using MDD principles by combining two different methodologies, the WebML extension presented in [Ceri, 07] and RUX-Method [Linaje, 07]. While the work in [Ceri, 07] provides support for modeling context-aware features at data and hypertext levels, RUX-Method adds new adaptation possibilities at the presentation level. WebML and RUX-Method have been presented before as a mean to model some of the RIA features. Notwithstanding, this is the first time that RUX-Method specific features for context-awareness are presented. Moreover, this is the first time that RUX-Method is combined with the WebML extension for context-awareness (WebML-CA from now on). Although here presented with WebML, RUX-Method can be used on top of other Web models as stated in [Linaje, 07].

The rest of the paper is as follows: Section 2 introduces a motivating example where some context-aware features arise. Section 3 briefly presents the extension of WebML for context-awareness and also shows its limitations. Section 4 introduces how RUX-Method faces WebML-CA limitations. In section 5 the combination of both, WebML-CA and RUX-Method is used to solve the example shown in section 2. Finally, sections 6 and 7 are devoted to related works and conclusions respectively.

2 Motivating Example

This section presents a simple RIA highlighting the necessity for adaptation to different contexts from the UI perspective. The chosen case study, *Next2Student (N2S) (http://www.next2student.com)*, is an example of the new up-and-coming applications with richer data and services and that is delivered in a more helpful

presentation by means of a RIA UI. N2S is a running Web application at the University of Extremadura that allows users (students and staff) to search, manage and view houses for renting before the beginning of the academic year. In addition, it allows users to discover available nearby services taking advantage of context-awareness. It has been specially designed for foreign students that usually do not know their way, nor which restaurants, museums, shops, public services, etcetera are available to them.

Figure 1: N2S in a mobile phone

N2S is regularly examining the environment to react accordingly to its current context. Next we describe different context-aware needs of N2S according to [Abowd, 99]:

a) *Device-aware presentation*: N2S has been designed for offering the same functionalities in different devices, adapting automatically the UI particularities to the concrete terminal. It can be accessed with desktop browsers (e.g., Google Chrome or Firefox), iPhone, Android as well as other AJAX-enabled Web browsers (e.g., webkit-based ones).

b) *Time-aware presentation properties:* N2S changes automatically the color of the fonts and vivacity of the UI according to the time when being accessed.

c) *Location-aware services:* N2S has a map where the services (banks, faculties, etc.) that fit in the user preferences are displayed according to the current position of the user device (Figure 1). These services are selected/filtered "on the fly" changing the results list according to the distance from the current position, preferences, and so on. E.g., the resulting flats on the map can be filtered according to the user necessities (e.g., number of rooms, baths, etc.). The services are created in N2S also using the time service information to show the services in a on/off mode (e.g., bus stops have not service during the night from 00h to 07h, some drugstores are 24h opened meanwhile others are closed during the night).

d) *User-aware personalization:* Users can e.g., choose the menu position and the font-size. This information can be stored for the next N2S usage.

Next we show how WebML-CA faces this motivating example, highlighting its advantages and limitations regarding the four features mentioned above.

3 Identifying the Modeling of Context-Aware practicalities in WebML

WebML is a well-known Web methodology and freshly addresses the adaptability to the context of use combining mainly two ideas: (i) rethinking the data model concepts for representing the context state and (ii) introducing new capacities at business logic level for managing the current/established application context state.

This section briefly points out the traditional WebML methodology and its proposed extensions for dealing with context-aware concepts in Web applications.

3.1 WebML in brief

WebML is a visual conceptual language for the specification of data-intensive Web applications. The application domain (i.e., data) is modeled using an extended Entity-Relationship (E-R) schema. This *data model* is the basis of the data intensive applications that can be designed using WebML. On top of the data model, WebML allows specifying the business logic and the content/containers composition by means of the *hypertext model*, whose key ingredients include siteviews, areas, pages, content units, operation units and links.

Content units are the atomic pieces of publishable content (e.g., an index of items); they offer alternative ways of arranging the content extracted dynamically from the entities and relationships of the data model, and also permit the specification of data entry forms for accepting user input. Units are the building blocks of pages, which are the actual interface elements delivered to the users. Pages are typically built by assembling several units of various kinds. Page and units are linked to form a hypertext structure: links express the possibility of navigating from one point to another one in the hypertext, and allow passing parameters among units, which is required for the proper computation of the content of a page. WebML also allows specifying operations implementing business logic; in particular, a set of operations for data updates is predefined, whereby one can create/delete/modify the instances of an entity, and create or delete the instances of a relationship. WebML can manage the information regarding data stores types (e.g., session parameters) through its data model. A recent extension includes new types of data stores [Bozzon, 06].

The WebML data schema also allows having a Web site that fit Web content to user preferences and other profile information. Hence, for introducing the role-based personalization the basic schema organizes the users into groups and groups into modules where a module (e.g., a siteview, a page, a unit) contains the appropriate information and/or services. A global parameter that identifies the active user in each session allows offering the desired personalized information (e.g., it can be used for content personalization, by means of recovering the name/surname of the active user

in a simple way or for discriminating the "wish list" items that correspond to the current user in a shopping cart application).

3.2 WebML for Context-Awareness

WebML has been used for managing context-awareness concepts [Ceri, 07], taking in mind applications that can take advantage of contextual information regarding user location, time of day, rendering capabilities in the target devices and user activity/preferences.

WebML-CA deals with context-awareness issues extending the concepts implied in the data modeling (for representing the context situation) and hypertext modeling (for defining the logic of the application regarding the context changes). At data design level the context model is defined by a concrete data sub-schema for representing the information related to devices, user preferences, location, etc. At run-time, the context environment is recovered and stored in that data sub-schema, which is the image of metadata for supporting the personalization. This information flow is controlled by the hypertext in charge of the context-aware capabilities of the application. Briefly, the pages conceived for managing context-aware capabilities are marked with a *C* label in the hypertext. The basic context-aware controller is placed in the *C* page. Hence, a marked page has content refresh mechanisms and extended operation units that manage (capturing and responding) the context of use and its changes when such page is accessed by the active user. The *C* pages also act as context-aware capabilities containers.

Like a puzzle, the set of pages that conforms the whole application (those marked *C* or not) are connected in the hypertext using communication mechanisms and operation chains. The actions to manage the updated context information can be decomposed into operation chains that are performed at two levels: actions for managing the data sub-schema related to the context model and actions for treating the hypertext adaptability. For the second set of actions WebML-CA proposes three kind of new (or reconceived) operation units: *(i)* units for managing the updated context information (*GetURL Parameter Unit* and *GetData Unit*, where both extend the original WebML *Get Unit* capacities for accessing session parameters), *(ii)* two units for evaluating conditions (for *If* and *Switch* control flow concepts), and *(iii)* those actions related to changes in the user interface due to the multichannel environment of the user (*Change Site View Unit* for changing the current siteview to another one that tails the rendering capacities and spatial positioning for the presentation of contents in a specific device and the *Change Style Unit* that is used for the adaptation of the presentation style appearance by means of a set of CSS files).

3.3 WebML–CA facing the motivating example

Next we show how WebML-CA addresses the context-aware issues that arise from the motivating example. For each issue we also highlight the limitations of the approach.

a) Device-aware presentation

Device-aware presentation implies to specify or adapt the UI for a device or set of devices. WebML addresses this issue using its native concept of siteview that is used

to design each device screen particularities. All the siteviews are defined over the single data schema of the application.

In the motivating example, the modeler can design a single data-schema. However, if N2S is going to be deployed in a PDA with a touch-screen and also in a PC device the modeler needs to define two different hypertext models (one siteview per device).

To deploy the same content and services in different devices in a multi-channel way depends on a wide range of parameters and there is no single solution that fits all situations. The problem in WebML is that it mixes in the hypertext model concepts related to operations (business logic), content/container composition (acting also like a pseudo presentation layer), managing of global/session parameters and so on. WebML presentation is based on a set of templates (e.g., XHTML) associated in a 1:1 way with the hypertext model. This fact forces to have as many hypertexts as final deployment devices.

An enhanced solution should be that the hypertext collects the business logic of the application independently of the rendering particularities that each device demands, modeling then those features at the presentation level (e.g., screen size, interaction possibilities).

b) Time-aware presentation

In the motivating example time-aware implies UI changes based on the current time. To solve it, WebML can manage a set of CSS files for presenting the UI features by defining at data level the files that will be used at hypertext level. Hence, the WebML solution, mainly based in the *Change Style Unit*, allows managing a set of presentation properties dynamically (e.g., according to the current hour).

WebML hypertext model is quite abstract for different Web rendering technologies using ad-hoc templates. However, CSSs introduced at the hypertext model invalidate many potential Web rendering technologies that do not use CSS (e.g., JavaFX or Flash). The CSS standard has limitations (e.g., dynamic values for style properties). Additionally, some of the most used browsers do not fully support the CSS standard, so complex CSS rules must be taken into account and must be added to the CSS file according to specific browsers.

An enhanced solution should be to manage all the UI properties in a canonical specification, allowing static and dynamic values. These properties (grouped or not) could be managed using abstractions, allowing us to specify them only once and then be transformed for different rendering technologies which only must take into account cross-browsing compatibility when the development technology requires it.

c) Location-aware services

Location-aware services in the motivating example imply that according to the position of a device, the application will change its responses to the user questions. The presentation model of WebML supports the specification of the look&feel and layout of the content units within each page. It uses the full refreshing page technique used in traditional Web 1.0 UIs. Under this situation, the entire page needs to be reloaded for responding to each user action synchronously. In addition, and due to its conception, WebML cannot launch several operations chains at the same time because it was conceived under the synchronous HTTP request/response paradigm,

where a user interaction initiates one operations chain that finishes rebuilding the HTML to be presented to the user. This issue is well addressed in WebML-CA marking the pages with the *C* label, when needed, being the marked pages translated by the code generator with an additional refreshment mechanism for partial content updates.

An enhanced solution can be the use of the single-page application paradigm available in RIA UIs, where a page is a combination of different presentation elements that can be updated independently triggering/consuming different underlying operations synchronously and asynchronously. This paradigm allows using highly interactive capabilities for context-aware UIs, allowing to improve the UX.

d) User-aware personalization

User-aware personalization in N2S implies changes in the UI according to the previous preferences of the user or default ones. WebML offers support for content and services personalization at data and hypertext level successfully.

However, as it is explained above in *b)Time-aware presentation*, also user-aware personalization issues require to define CSS styles that will be altered according to the user that visit the application using data-driven values, and those values (e.g., the position of the menu) cannot be stored in a CSS file.

Following the conclusion in *b)Time-aware presentation*, an enhanced solution should be using a canonical representation of the UI look and feel properties independently of its value (dynamic or static).

4 RUX-Method for Context-Aware User Interfaces

4.1 RUX-Method all-purpose introduction

RUX-Method is a model driven method which supports the design of multimedia, multi-modal and multi-device interactive UIs for RIAs. RUX-Method focuses on the enrichment of the UI while takes full advantage of the content and functionality already provided by the existing Web models (e.g. WebML). RUX-Method supports natively the Single-page paradigm, allowing highly interactive capabilities for context-aware UIs, improving this way the UX to be achieved.

RUX-Method overview is depicted graphically in Figure 2. At design time, RUX-Method uses existing data, business logic and presentation information offered by the underlying Web model being enriched (e.g. WebML). 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-Method, the data and business logic remain the same. To sum up, the responsibility of RUX Method is providing a new UI with RIA features. To facilitate the UI development process, RUX-Method is divided into three Interface levels: Abstract, Concrete and Final UIs. Each UI level is mainly composed by UI Components whose specifications are stored in the Component Library where one Component can only belong to one Interface level. The Library also stores how the transformations among Components of different levels are carried out.

There are two kinds of adaptation phases in the RUX Method according to the UI levels defined above. Firstly, the adaptation phase that catches and adapts a Web 1.0 model (taking the data, navigation, as well as presentation concepts when it is possible) to RUX-Method Abstract UI that is called Connection Rules (CR). Secondly, the adaptation phase that adapts this Abstract UI to one or more particular devices that is called Transformation Rules 1 (TR1). Finally, there is an additional transformation phase, Transformation Rules 2, (TR2) that completes the MDD life-cycle of RUX-Method supporting and ensuring the right code generation. Thus, in TR2, the Final UI is automatically obtained depending on the chosen RIA rendering technology (e.g. Laszlo, JavaFX or AJAX). This process is performed automatically because TR2 establishes the way the matching takes place among Concrete and Final UI Components.

RUX-Method defines a Device Repository (RUX-DR) that extends the capabilities of Wurfl [Wurfl, 10] due to the latter being only focused on mobile devices. These extensions have been done mainly to specify RIA rendering technologies not specified previously in Wurfl (e.g., JavaFX); to filter all those non-RIA capable devices; and to add some devices (e.g., netbooks, consoles) not included in Wurfl. RUX-DR is managed using the standard Device Description Repository (DDR) Simple API [DDR, 10], so RUX-Method is not limited to use its Wurfl repository extension, taking advantage of third-party repositories (e.g,, UAProf).

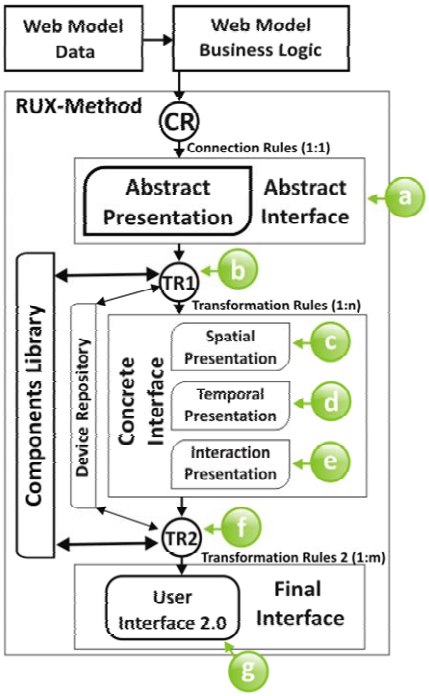

Figure 2: RUX-Method User Interface Design

The idea behind RUX-Method is that an Abstract UI can be transformed to many Concrete UIs, where Concrete UIs are focused in the capabilities of a device or the common capabilities of a group of devices. Final UIs can be generated from each pair of Concrete UI and available final rendering technology. The set of Final UIs is organized in a sorted (by preference) list where only one of them can be marked as default (to be used when none of the Final UIs fits with the device used at runtime to access the Web application.

4.2 RUX-Method capabilities for CA constrains

This section explores how the context-aware limitations identified in Section 3.3 are managed by RUX-Method according to its UI levels and transformations. To illustrate it, Figure 2 is marked with letters (from *a* to *g*) that will be used to specify where each context-aware issue is taken into account in RUX-Method.

a) Device-aware presentation

RUX-Method is able to constrain the Concrete UI using expressions over one or many device capabilities. These constrains are applied at TR1 (marked *b* in Figure 2) using an expression algebra for minimal capabilities. This algebra is based on the Device Description Structures [DDS, 10] working draft. As an example the code below is used to select all the devices with a display width between 240 and 299 (pixels) and touchscreen as interaction mechanism:

```
<expression>
    (([displayWidth] >= 240 and [displayWidth] < 300) and
    [inputDevices]contains('touchScreen'))
</expression>
```

This kind of expressions returns, on the one hand, a set of real UI constrains (e.g., devices selected with the expression above may also impose constrains over the e.g., displayHeight to be between 450 and 600) and, on the other hand, a set of *user_agent* strings which are stored to be used later.

Spatial presentation (marked *c* in Figure 2) is also constrained by the device selection e.g., the available screen size is limited, but also the kind of components that could be used can be limited when e.g., devices are not able to support the quicktime plug-in to show videos.

Interaction presentation (marked *e* in Figure 2) is also dependent of the selected devices that could impose constrains over the events that can be used in a component even when the event is already defined for that component. For example, *mouseover* effect is available for the *text_field* component when the target is a PC but not when the target is a touchscreen phone).

When transforming from a Concrete to a Final UI, using TR2 (marked *f* in Figure 2), those constrains imposed by the selected devices could make not available all the RIA technologies for the Final UI. So when an iPhone has been selected as a target device, a new constrain has arisen specifying that the only possible rendering platform for the RIA UI is AJAX (due to the fact that the iPhone does not support flash-based RIAs).

As we had advanced, the set of *user_agent* strings were stored to use them later, and it is at the Final UI (marked *g* in Figure 2) where at runtime a switch that is automatically generated by TR2 takes action. First of all, it captures the *user_agent* string of the user device and secondly, according to the *user_agent* strings previously stored, it redirects the user to that UI that was designed to maximize the UX of his device with the designed application.

At the Final UI (market *g* in Figure 2) level there is a switcher that is automatically generated to be used at run-time. When a user with a particular device access to the Web application, it is captured its *user_agent* string to be used to retrieve from RUX-DR the device capabilities. According to these capabilities, the switcher redirects the user to the first Final UI of the list that fits with the device capabilities.

b) Time-aware presentation

RUX-Method is able to manage dynamic UI properties. Abstract UI level (marked *a* in Figure 2) is used to specify those dynamic data that come from the underlying Web Model business logic and to specify how the collected data (e.g., using form fields) must be sent to the business logic level to be understood.

RUX-Method Spatial Presentation (marked *c* in Figure 2) specifies the type of UI component to be used, so all the properties of those components can be got or set at this presentation level. Each property in RUX-Method can be fixed with a static value (e.g., 14 points for the *font_size* property) or a dynamic one (i.e., a data-driven value for a property).

c) Location-aware services

RUX-Method is able to fully exploit the single-page application paradigm, avoiding unnecessary refreshments and allowing asynchronous communications between the client and the server. Synchronous and asynchronous links, UI modifications, etc. are conceptualized by RUX-Method as actions at the Concrete UI level. Different kinds of actions are available in RUX-Method e.g., UI-Actions, which make it possible to perform dynamic changes over a UI component through its methods and properties or Call-Actions, which are defined to specify calls to the underlying business logic. Actions are always specified inside a handler that can be triggered by a predefined behavior (i.e., time-based) by the temporal presentation (marked *d* in Figure 2) and/or by a non-predefined behavior (i.e., user interaction) by the interaction presentation (marked *e* in Figure 2).

d) User-aware personalization

RUX-Method support for UI personalization uses the same mechanisms as those for time-aware presented before. All the UI component properties in RUX-Method are categorized into families and can be reused along all the UI using styles, which share many concepts with the CSS classes, but are more abstract. Indeed, RUX-Method has been used to create Adobe Flash-based UIs and Flash do not fully support CSSs. Other important issues are that CSSs are not fully cross-browser (i.e., there is not a browser actually covering the last CSS standard) and style properties in a CSS file cannot contain dynamic values, so the personalization of a UI using only a CSS approach should be incomplete.

5 Combining RUX-Method and WebML-CA

The combination takes the best from both, WebML-CA and RUX-Method. The data model is the WebML original one. Regarding the specific WebML-CA extensions at hypertext level, this work proposes to use *(a)* the *GetURL Parameter Unit* for accessing fresh context information (but it does not use the *Get Data Unit* that in our work is substituted by the *Selector Unit*); *(b)* the units for condition evaluation. The presentation is modelled with RUX-Method that is able to take advantage of the UI composition available in the WebML hypertext model (e.g., pages). This choice avoids those WebML-CA extensions related to the UI (i.e., *Change Site View Unit* and *Change Style Unit*).

Next we present how this combination faces the motivating example. First a general solution to the four problems identified in Section 2 is introduced. Then, a particular solution to the data, hypertext and presentation levels of N2S is presented.

We will not focus thoroughly in context information provider due to it being a more technological aspect, which does not affect significantly the specification of the data, hypertext or presentation design.

5.1 *The motivating example revisited using an MDD approach*

N2S has been developed using WebRatio [Acerbis, 08] and RUX-Tool [Linaje, 09] the CASE tools of WebML and RUX-Method respectively. N2S can be accessed at *http://www.next2student.com*.

Section 2 summarized briefly the context-aware features available in *N2S*: *a) device-aware presentation*, *b) time-aware presentation properties*, *c) location-aware services* and *d) user-aware preferences.* These features have been considered at different levels in the *N2S* development design process.

Regarding *a)*, it was solved at the presentation level using the multi-device approach proposed by RUX-Method.

The issues regarding *b)* are solved at different levels:

- at WebML data level, by extending the WebML schema storing such attributes,
- at WebML hypertext level, for recovering the suitable set of values for those attributes regarding the current time and,
- at presentation level, using the RUX-Method dynamic properties for the presentation elements fed by the appropriate values in each case.

The *c)* feature involves a cross cutting solution that also implies the three levels. Thanks to the possibility for UIs generated by RUX-Method for triggering/consuming several services provided by the underlying business logic at the same time, the *N2S* maps show the filtered results by different criterions “on the fly” under the single-page paradigm.

The features in *d)* have been treated through the data storage in a *Preferences* entity associated to the *User* entity in the data level. In this case, the WebML hypertext also allows managing it for sending such information to the presentation level, defined by RUX-Method.

Context-Aware functionality	**WebML-CA Data design**	**WebML-CA Hypertext Design**	**RUX-Method presentation**
a) Device-aware presentation	*Not relevant*	*Not relevant*	*Multi-Device spatial design presentation*
b) Time-aware presentation properties	*Data entities for properties*	*Acquisition of time information*	*Presentation elements with dynamic properties*
c) Location-aware services	*Data entities for services information*	*Filtering and managing results*	*· Presenting services results* *· Triggering/consuming several underlying operation chains* *· Single-page paradigm*
d) User-aware preferences	*Data entities for properties*	*Selecting current user and its preferences*	*Presentation elements with dynamic properties*

Table 1: Relation between levels and contexts

Table 1 summarizes the relations between the different levels and the considered contexts. In the next subsections we show some details of these different levels for N2S. For space reasons, we focus on a reduced version of N2S with the aim of showing better the context-aware features described in this paper. So we avoid other complementary issues such as multi-language content, reminders, volatile capabilities and other architectural features.

5.1.1 Defining the *N2S* context-aware sub-schema Datadata model

The N2S example uses three main groups of entities for expressing the location-aware and time-aware capabilities at data level expressing a basic user profile/preferences and context-aware schema (Figure 3). The *User* entity is related to *Rol/SiteView* entities for expressing an essential profile for each active user in the application regarding its privileges and consequently the Siteviews that can be accessed by that user according to his Role. The user preferences are directly related to each user item. The *Preferences* entity represents the user preferences regarding the number of baths and rooms in the desired houses. In this entity there is an integer attribute called *radio* that represents the distance preferred by the user for searching services and houses around its current position expressed in meters.

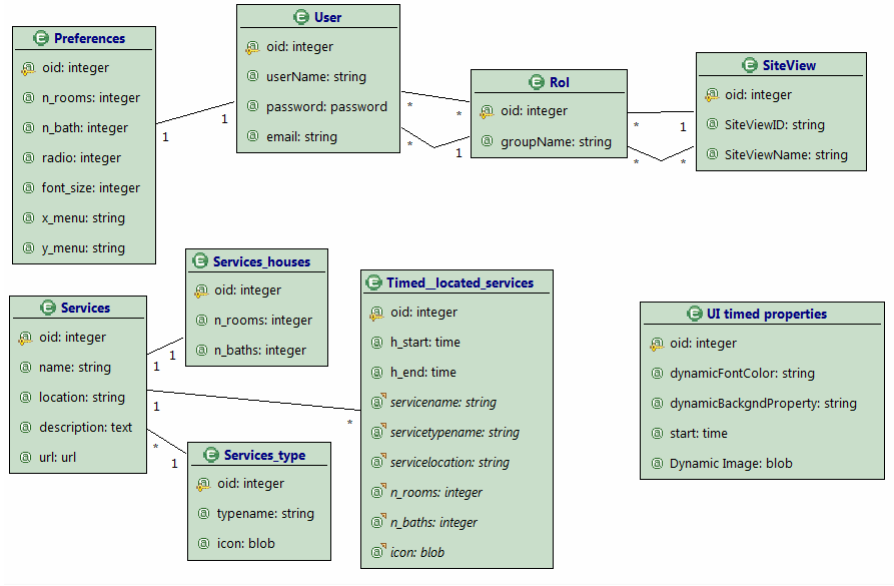

Figure 3: Data model excerpt of N2S in WebRatio

The context entities are not always related to the User entity because some context features are related to the device, location and other information that can be treated in a volatile way (e.g., session variables).

There are another group of entities associated to *Services*. The example offers different types of services categorized by type (e.g., drugstores, banks, faculties, houses) using the *Services_type* entity. Each item of this entity has a blob attribute called *icon* (the icon associated to each service, e.g., a green cross for drugstores). Only when an item of Services is a house such item is associated to its corresponding item in the *Services_houses* entity for storing the number of existing baths and rooms. Each service has a *location* for representing longitude and latitude concatenated and it has states according to its time activity, e.g., a drugstore can have two items depending on it is closed or not. This feature is expressed in the *Timed_located_services* entity. In this entity we can find several derived attributes that do not store information in the database but are very useful for making simple the design process of the WebML hypertext. The last entity that can be found in this data context model excerpt is the *UI_timed_properties*. It is used for managing the dynamic UI properties at runtime according to the current time. The general UI properties are associated to a starting time for establishing a concrete set of values for the fonts, background and image used.

5.1.2 Defining *N2S* Context-Aware features in the hypertext model

The hypertext excerpt depicted in Figure 4 provides some of the context-aware capabilities needed at UI level. At the top we can find the managing of dynamic properties according to the current time. This functionality has been placed in a

master page (called *Timed_Dynamic_Properties*) with the aim of being accessible from all the pages of the siteview showed in Figure 4.

The business logic process always begins as follows when a user has already signed on the online application and has been redirected to the protected siteview (depicted on Figure 4). Firstly, the *Time Unit* retrieves the current time and this information is used for choosing the appropriate set of dynamic UI properties in the *Data Unit* (called *current_Timed_Dynamic_Properties*) regarding fonts color, background color and dynamic image (e.g., *#BFC0C3* from 09h, *#F2F7FA* from 17h and *#FFFFFF* from 20h for fonts color). Secondly, the user sign on information (available at *GetUserCTX*) is used to get the *User UI preferences* for that specific user.

In Figure 4 we also see two pages, *Simple Version of UserPreferences* and *Simple Version of CurrentLocation filtered services*. The first page is used for retrieving the user preferences (i.e., choices about the services) or selecting new ones. The second page is used for calculating the group of services that fit in the group of user preferences taking into account the selected distance from the current location.

In the *UserPreferences* page the current user key is obtained from the context parameter information using the *Get Unit.* Based on this information the current user details are shown (*User Information Data Unit*) and its basic preference values regarding the number of the baths and rooms in the desired houses and the original searching distance established are retrieved (*Preferences Selector Unit*) for preloading such information in the suitable data input unit (*New/Current PreferencesEntry Unit*). The values of these preferences must be changed to perform a new searching with new parameters (e.g., a new searching *radio* value). The *New/Current Preferences Unit* is also fed by the list of existing service types (retrieved by the *Services_Type Selector Unit*) in a multi-entry field to be used (by means of check boxes) in the searching preferences, e.g., including banks, restaurants, etc. This entry unit triggers the filtering of services, which is performed in the *Simple Version of CurrentLocation filtered services* page. The searching of services to be displayed over the map is carried out using several parameters and preferences: on the one hand, the user preferences established in the entry unit and, on the other hand, the location and time information. We can see a *Get Unit* (*GetLocationCTX*) for retrieving the current location context parameter and the *Time Unit* (*GetCurrentHour*) for knowing the current time. This information is collected in a *Script Unit* (*Nearby Services*) that filters the services fed by the *Services Selector Unit* according to the activity hour (for showing only those services with a time slot in the current time) and are in the indicated radio action. The *Nearby Services Unit* gives the list of services that meets the context and preference values, which is used for feeding the UI map using the *Map Information Index Unit*.

The N2S application retrieves the current time regarding the device location for managing the dynamic properties and the active services by means of Web services and more complex operations chains that those here described. For simplicity reasons, N2S has been represented here in a simpler way using the current server time.

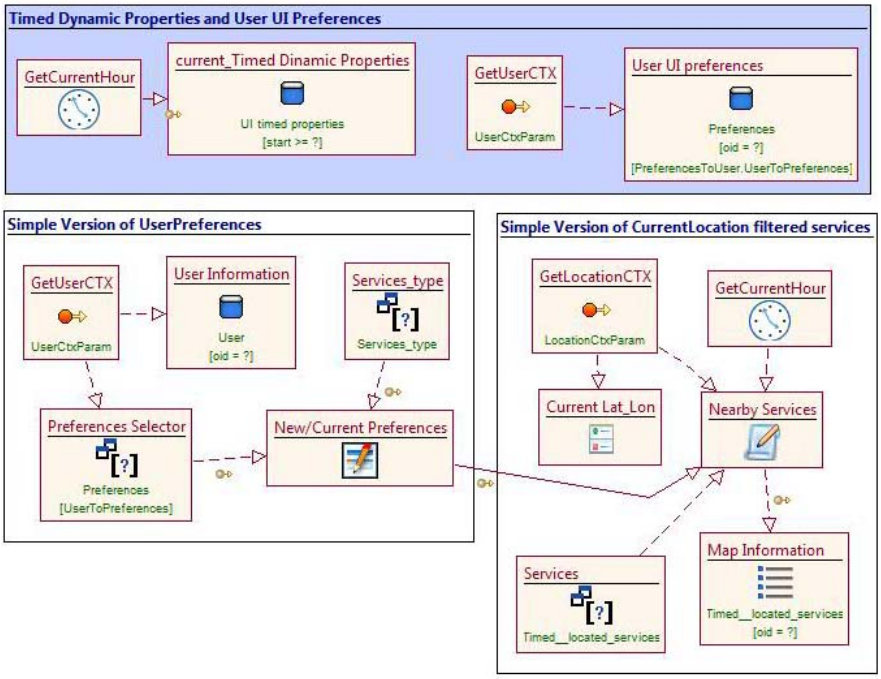

Figure 4: Hypertext model excerpt of N2S application using WebRatio

5.1.3 Defining *N2S* Context-Aware presentation

Figure 5 illustrates how in the presentation design process the single-page paradigm composition allows to specify an UI spatial presentation combining different pages of the hypertext model simultaneously. Through the use of asynchronous communications between the client and the server, the UI composition capability of RUX-Method allows retrieving, consuming and updating content asynchronously from one or many operation chains simultaneously.

Based on the single-page application paradigm, the basic application's activity is carried out in the same page, updating the page's content and presenting new information as the preferences change and the context evolves.

After applying the connection rules (CR in Figure 2) over the WebML model, the Abstract UI is obtained. Applying TR1, a first draft of the spatial presentation is automatically obtained according to the device particularities that usually establish constrains over the Concrete UI. Then, the designer can refine this first draft to meet the particular spatial design requirements in order to get a better UX.

For example, Figure 5 depicts in the left-up side the different containers (commonly called views in RUX-Method) that are hierarchically composing the main application presentation (shown in the center in the same Figure). This part of the Figure illustrates the spatial UI design for a specific set of devices (e.g., iPhone series) taking in mind its particularities (e.g., screen size and resoltution) by means of RUX-DR (RUX Device Repository).

In the right side of Figure 5 the set properties of each UI component are shown, where the names, properties and behaviors for each component can be customized. The figure shows how the Multi_Input_Google_Map component is configured. This presentation component is used for managing a google map element in a data-driven way filtering the results according to the options selected by the user for showing the nearby services available and current location issues.

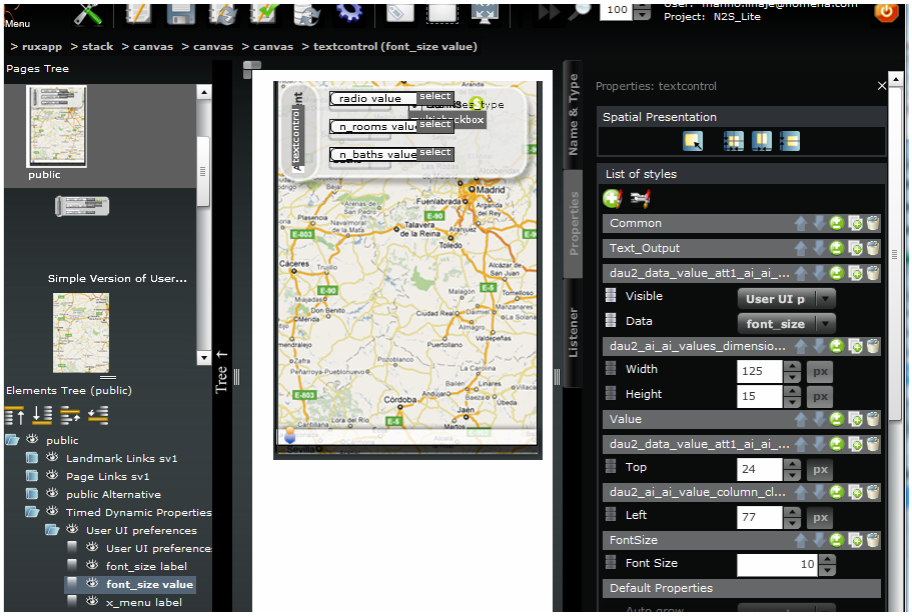

Figure 5: Presentation design process of N2S using RUX-Tool

Regarding dynamic properties, Figure 6 shows how a presentation property (i.e., *Font Size*) can be treated statically (i.e., "*10*" value in Figure 6 left) or dynamically (i.e., *font_size* in Figure 6 right). When a presentation property is static, the value remains fixed in the code generation phase; when it is dynamic, its value is recovered from the data model through the hypertext model. Figure 6 (right side) shows how the *Font Size* presentation property is dynamically used in the N2S case study.

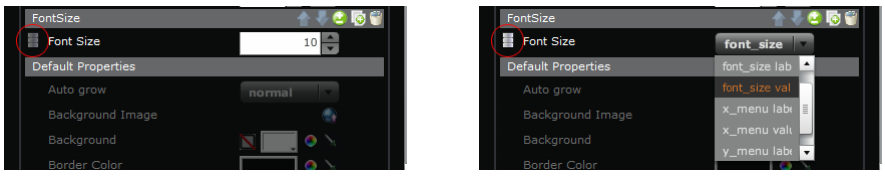

Figure 6: Snapshot of static (left side) and data-driven dynamic (right side) properties selection using RUX-Tool

This property was represented in the WebML data model (Figure 3) for symbolizing the user preferences and managed at hypertext level (Figure 4) by the User UI Preferences Data Unit placed in the master page. In the case of those properties whose values are dynamically based on time the treatment is always the same.

6 Related Work

There are not many works in the area of MDD of Multi-device context-aware RIA IUs. However, two different fields deserve our attention: the Human Computer Interaction (HCI) and Web Engineering fields. Indeed, RUX-Method supposes a bridge between both fields. For this related work, we only focus on MDD approaches because they provide a more abstract and general solution not being dependent of the e.g., RIA rendering technology. For instance, [Buttler, 07] is only focused on XUL so other RIA technologies cannot take direct advantage of this research.

Regarding the HCI field, a classification of UI adaptations and methods are carried out in [Pihkala, 03]. Among these methods, those ones based on the Cameleon Framework [Calvary, 03], are commonly accepted MDD approaches for multi-device UI design. RUX-Method is based on this approach, so it inherits its context-awareness advantages. Many efforts have been carried out in the HCI field for the UI adaptation to multiple devices e.g., [Calvary, 02] where a plastic approach is presented. However, as stated in [Martínez, 08], these works are mainly applied to text or forms based UIs, so more complex structures like RIA UIs cannot take full advantage of these solutions.

In this sense, UsiXML is a Cameleon Framework based approach used to generate RIA UIs [Martínez, 06] and it has been extended in many directions. The main differences with regard to our approach is that UsiXML establishes a 1:1 relation between a Concrete Interface and a rendering technology and it is focused in the concept of one UI that is adapted to fit all the devices while RUX-Method is focused in the design of a UI for a set of devices with common capabilities in order to maximize the potential UX. This UI adaptation of UsiXML evolved in [Martínez, 08], that proposes the generation of context-aware UI containers providing a general solution to automatically adapt a UI to different devices (mainly based on the screen capabilities). However, other context-aware implications (e.g., user personalization) are not considered.

Regarding the Web Engineering field, there exist Web models that claim to integrate natively adaptive concepts such as Hera [Houben, 08] and other that have been extended to cover them [Ceri, 07] [De Virgilio, 05] [Garrigós, 07]. Hera has been also extended in [Fiala, 05] to deal with presentation issues, but this extension does not provide temporal and interactive relationships and does not consider multiple Web models. [De Virgilio, 05] uses WebML and extends it with profiles that can be used to represent a variety of contexts with different detail levels. It differs from other Web Engineering techniques because it can address an ad-hoc fine-grained final UI. However, context presentation issues are specified using HTML+CSS, presenting the same problems already specified in this paper for WebML-CA. The proposal presented in [Garrigós, 07] and [Ceri, 07] are generic and valid in many different environments. Both require a “rule” engine that in the related literature is usually

addressed as a specific server extension. However, they do not specifically focus on RIA presentation and interaction necessities. For example these approaches are not able to specify interactions that modify the UI state at client side [Bandelloni, 07] avoiding unnecessary communication roundtrips.

Focused on RIAs, the work in [Garrigós, 09] proposes an interesting extension of OOH4RIA [Meliá, 08]. This work treats both, personalization of content and presentation, to offer user and device-aware personalization capabilities. For this purpose, authors make use of the extra presentation possibilities offered by RIAs to establish the concrete presentation capabilities in the target device. Firstly, they tag the elements that need to be reorganized in the Presentation Model with the aim of applying the particular spatial arrangement settings (when needed) of the content containers and secondly they define a User Model to collect the user preferences.

In [Wright, 08] some Web Modeling approaches are analyzed to compare their suitability to model interactive applications (like RIAs are). According to the conclusions of this survey, [Ceri, 07] is a good approach while it lacks an events model, more control over the web browser, and scripting support. We have demonstrated in practical terms through the case study presented in this paper that WebML-CA limitations can be solved by connecting it with RUX-Method.

According to [Pietschmann, 09] the field of context-aware Web applications is still restricted to basic hypermedia systems and these approaches fail for Rich Internet Applications (RIA) and dynamic content adaptation. To the best knowledge of the authors, out of these engineering fields and closely related with our objective, we only found two relevant research publications. On the one hand, [Schmidt, 07] that use ontologies, but it does not contemplate the personalization of the presentation features. On the other hand, [Heidenbluth, 09] proposes *status senstive components* as regular UI elements extended to react to the change of watched statuses (using a publish/subscribe message service approach). It presents a pattern and a case study using it. However, due to the fact that all the *sensitive* components must listen to each *sensitive* event (e.g., signed on, going offline…), according to our development experience in the field, run-time performance will be damaged and this is a drawback in real applications such as N2S.

7 Conclusions and Future Work

Web technology is increasingly being used to deliver services. However, it is also increasingly becoming the individual's right to choose how they are supported. This then becomes personalized services that encompass a wide range of technology issues including different devices. This personalization forces applications to adapt to some contexts such as location, identity, activity or time. Adaptations on multi-device Web applications particularly affect their UIs. These UI adaptations are far from being trivial and there are some interesting works addressing the issue as stated in section 6. Among all of them, those incorporating MDD concepts are exhibiting many advantages due to them showing the technology details only in the last transformation phases. However, client requirements continue increasing with regard to UX (e.g. multimedia contents, high interactivity) and technology is answering with a new wave of Web applications known as Rich Internet Applications that adds wood to the fire and makes context-awareness a little bit more difficult. The fact is that Web

methodologies (Model-Driven or not) are not able to face some of the new UI requirements imposed by the appearance of Rich Internet Applications.

Among those Web methodologies following an MDD approach, WebML has been recently extended for context-awareness (WebML-CA). Although this approach presents many advantages at the data and hypertext levels, it also exhibits some limitations at the presentation level. This paper not only has identified these limitations but also has shown how the combination of WebML-CA and RUX-Method supposes a significantly advance in the Model Driven Development of Multi-device Context-Aware User Interfaces for RIAs by defining an independent multilevel presentation layer. For this purpose, a running context-aware application recently deployed has been used.

This example has helped us to show the main contributions of the approach that can be summarized as follows:

- It solves device-aware presentation by using RUX-DR which is a repository of devices and their properties.
- It solves time-aware presentation by using the dynamic properties provided by RUX-Method.
- It solves location-aware services by using the single-page paradigm inherent to RIAs and managed by RUX-Method.
- It solves user-aware personalization by using again the dynamic properties of RUX-Method.

Although here presented with WebML, RUX-Method can be used on top of other Web models with the aim of enriching the presentation layer with RIA concepts.

A work we are facing now and that is complementary to the one here presented embraces the inclusion of accessibility features in the applications generated with RUX-Tool. In this sense, the UI components of RUX-Tool at the level of the Concrete Interface are being enriched with roles, states and properties coming from the WAI-ARIA specification draft [WAI-ARIA, 10].

Acknowledgements

This work has been partially supported by Fondo Europeo de Desarrollo Regional (FEDER) and the Spanish projects: TSI-020501-2008-47 (granted by Ministerio de Industria) and TIN2008-02985 (granted by Ministerio de Ciencia e Innovación)

References

[Abowd, 99] Abowd, G.D., Dey, A.K., Brown, P.J. Nigel Davies, N. Mark Smith, M. Steggles, P.: Towards a Better Understanding of Context and Context-Awareness Source. International Symposium on Handheld and Ubiquitous Computing, pp. 304-307, 1999.

[Acerbis, 08] Acerbis, R., Bongio, A., Brambilla, M., Butti, S.: WebRatio 5: An Eclipse-Based CASE Tool for Engineering Web Applications. International Conference on Web Engineering, Springer Berlin/Heidelberg. LNCS 4607, pp. 501-505, 2007

[Bandelloni, 07] Bandelloni, R., Mori, G., Paternò, F., Santoro, C., Scorcia, A.: Web User Interface Migration through Different Modalities with Dynamic Device Discovery.

International Workshop on Adaptation and Evolution in Web Systems Engineering at ICWE'07, pp. 58-72, 2007

[Bozzon, 06] Bozzon A., Comai S., Fraternali P., Toffetti Carughi G.: Conceptual Modeling and Code Generation for Rich Internet Applications. International Conference on Web Engineering, pp. 353-360, 2006

[Butter, 07] Butter, T., Aleksy, M., Bostan, P., Schader, M.: Context-aware user interface framework for mobile applications. 27th International Conference on Distributed Computing Systems - Workshops, pp. 39, 2007

[Calvary, 02] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Souchon, N., Florins, M., Vanderdonckt, J.: Plasticity of User Interfaces: A Revised Reference Framework. International Workshop on Task Models and Diagrams for User Interface Design, pp. 127-134, 2002

[Calvary, 03] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, Vol. 15, No. 3, pp. 289-308, 2003

[Ceri, 07] Ceri, S., Daniel, F., Matera, M., Facca, F. M.: Model-driven development of context-aware Web applications. ACM Transactions on Internet Technology, vol. 7, pp. 1-33, 2007

[Daniel, 09] Daniel, F.: Context-Aware Applications for the Web: a Model-Driven Development Approach. Context-Aware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications, IGI Global, pp. 59-82, 2009

[DDR, 10] http://www.w3.org/TR/DDR-Simple-API/. Last visited: 1-June-2010

[DDS, 10] http://www.w3.org/TR/dd-structures. Last visited: 1-June-2010

[De Virgilio, 05] De Virgilio, R., Torlone, R.: A general methodology for context-aware data access. ACM International Workshop on Data Engineering for Wireless and Mobile Access, pp. 9-15, 2005

[Fiala, 05] Fiala, Z., Hinz, M., Meissner, K.: Developing component-based adaptive Web applications with the AMACONTBuilder. IEEE International Symposium on Web Site Evolution, pp. 39-45, 2005

[Garrigós, 07] Garrigós, I., Cruz, C., Gómez, J.: A prototype tool for the Automatic Generation of Adaptative Websites. International Workshop on Adaptation and Evolution in Web Systems Engineering at ICWE'07, pp. 13-27, 2007

[Garrigós, 09] Garrigós, I., Meliá, S., Casteleyn, S.: Personalizing the Interface in Rich Internet Applications. International Conference on Web information Systems. Springer-Verlag LNCS vol. 5802, pp. 365-378, 2009

[Heidenbluth, 2009] Heidenbluth .Status Sensitive Components: Adapting Rich Internet Applications to their Runtime Context. International Conference on Digital Society, pp. 133-138, 2009

[Houben, 08] Houben, G.J., van der Sluijs, K., Barna, P., Broekstra, J., Casteleyn, S., Fiala, Z., Frasincar, F.: Web Engineering: Modelling and Implementing Web Applications. Springer, Chapter 10, pp. 263-301, 2008

[Limbourg, 05] 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. pp. 207-228, 2005

[Linaje, 07] Linaje, M., Preciado, J.C. and Sánchez-Figueroa, F.: Engineering Rich Internet Application User Interfaces over Legacy Web Models. IEEE Internet Computing, vol. 11, iss. 6, pp. 53-59, 2007

[Linaje, 09] Linaje, M., Preciado, J. C., Morales-Chaparro, R., Rodríguez-Echeverría, R., Sánchez-Figueroa, F.: Automatic Generation of RIAs Using RUX-Tool and Webratio. International Conference on Web Engineering. Springer-Verlag LNCS 5648, pp. 501-504, 2009

[Martínez, 06] Martinez-Ruiz, F., Muñoz Arteaga, J., Vanderdonckt, J., Gonzalez-Calleros, J. Mendoza, R.: A first draft of a Model-driven Method for Designing Graphical User Interfaces of Rich Internet Applications. Latin American Web Congress, pp.32-38, 2006

[Martínez, 08] Martinez-Ruiz, F.J., Vanderdonckt, J., Muñoz Arteaga, J.: Context-Aware Generation of User Interface Containers for Mobile Devices. Mexican International Conference on Computer Science, pp. 63-72, 2008

[Meliá, 08] Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. International Conference on Web Engineering, pp.13-23, 2008

[Moreno, 07] Moreno, N., Romero, J.R., Vallecillo, A.: An Overview of Model-Driven Web Engineering and the MDA. In Web Engineering: Modelling and Implementing Web Applications. G. Rossi, O. Pastor, D. Schwabe, L. Olsina (eds.) Springer-Verlag, pp. 353-382, 2007

[Pietschmann, 09] Pietschmann, S., Voigt, M., Meißner, K.: Dynamic Composition of Service-Oriented Web User Interfaces. International Conference on Internet and Web Applications and Services, pp. 217-222, 2009

[Pihkala, 03] Pihkala, K. Extensions to the SMIL Language. PhD dissertation. http://lib.tkk.fi/Diss/2003/isbn9512268043/. Last accessed: 29-Jan-2010

[Preciado, 05] Preciado, J.C., Linaje, M., Sánchez, F., Comai, S.: Necessity of methodologies to model Rich Internet Applications. IEEE Web Site Evolution, pp. 7 - 13, 2005

[Schmidt, 07] Schmidt, K., Stojanovic, L., Stojanovic, N., Thomas, S.: On Enriching Ajax with Semantics: The Web Personalization Use Case. European conference on The Semantic Web, pp. 686-700, 2007.

[Seffah, 03] Seffah, A., Javahery, H.: Multiple User Interfaces: Crossplatform Applications and Context-Aware Interfaces. J. Wiley, 2003

[WAI-ARIA, 10] Accessible Rich Internet Applications. http://www.w3.org/TR/wai-aria/. Last visited: 1-Jun-2010

[Wright, 08] Wright, J. Dietrich, J.: Survey of existing languages to model interactive web applications. Conferences in Research and Practice in Information Technology Series, Australian Computer Society, vol. 325, pp. 113-123, 2008

[Wurlf, 10] http://wurfl.sourceforge.net. Last visited: 1-Jun-2010