UWE-R: an extension to a Web Engineering methodology for Rich Internet Applications

LEONARDO MACHADO, ORLANDO FILHO and JOÃO RIBEIRO
Faculdade de Engenharia
Universidade do Rio de Janeiro - UERJ
R. São Francisco Xavier, 524, 5º Andar, Bloco D, sala 5028, 20550-900 Rio de Janeiro – RJ
BRAZIL
lchaves@iname.com - orlando@eng.uerj.br - araujo@eng.uerj.br

Abstract: - This paper introduces UWE-R, an extension to an existing Web Engineering Methodology (UML For Web Engineering – UWE) to model Rich Internet Applications (RIA). Initially it presents the basic concepts behind RIA and UWE, showing the reasons for choosing this methodology among many others. After presenting the proposed extensions for UWE-R, a modeling instance is shown using a real web site with RIA features. This work concludes with future directions and issues to be addressed.

Key-Words: Rich Internet Applications, Web Engineering Methodology, Unified Modeling Language

1 Introduction

The same way the Internet and the World Wide Web have revolutionized how people interact with software applications, the so called Rich Internet Applications (RIA) are a revolution for Web applications. Sites like Google Maps, Gmail, Flickr, to name a few, bring a much richer user experience, but also present greater challenges for their modeling. A simple web application can be developed without modeling, but this is not the case for complex ones. Unfortunately web methodologies for RIA are still in their childhood. This paper's goal is helping to fill this gap.

2 RIA – Rich Internet Applications

There are some definitions for RIA as in [1] and [2]. Here we propose the following: RIA are applications that benefit from the web ubiquitousness and enhance the traditional web application model adding rich user interface elements altogether with a flexible and asynchronous server interaction mechanism. The final user experience resembles that of a desktop application, since the user is no longer required to wait for a synchronous server response for each link that is activated.

There is a great variety of RIA technologies. Most of them involve somehow Asynchronous Javascript with XML (AJAX), which is a set of technologies (XML, DOM, XHTML, Javascript and XmlHttpRequest) that allows an asynchronous interaction with the server [3]. However, RIA is not just AJAX. Technologies like Adobe's Flex and Sun Microsystems' Java also offer asynchronous communication with the server and a rich graphical user interface.

3 Web Engineering Methodologies

The Unified Modeling Language has become the standard for object-oriented applications. Both in the corporate and academic worlds it is used to model applications in general. Nevertheless, when Web applications must be modeled, new challenges arise, since navigation, user interaction and content type for this kind of applications do not have appropriate support in UML's kernel. Another work

[4] deals with this lack of support. In that paper 15 different Web engineering methodologies are evaluated using a set of criteria that are relevant for RIA modeling, and the conclusion is the necessity of a new methodology to encompass RIA features, like visual continuity, asynchronous interaction and multimedia content.

For the present paper we proceeded to a detailed study of those 15 methodologies and other 2 that were not referenced in [4]. Our goal was to evaluate which one could be extended to incorporate RIA features. For the sake of brevity, only this study's results are presented here. Some important notes before: 2 methodologies referenced in [4] were not considered: DMM+t and AHAM. The former because it is an extension to OMMMA-L, which was part of our study and its new aspects are not relevant for RIA. The latter was not included because it has a different modeling purpose: Adaptive Hypermedia Systems (AHS), i.e., systems that have their content adapted to the user or user profile.

Web Application Extension (WAE) as defined in [5] was included in our study since it is well referenced in several papers. Rich User eXperience model (RUX) as in [6] was also included, because it is the only one, as of this writing, which contemplates RIA aspects.

UML for Web Engineering (UWE) was chosen from all of these, for the reasons that will be explained in the next section.

3.1 Reasons for choosing UWE

One of this work's purpose is to leverage, as much as possible, existing concepts and languages. No Web engineering methodology has had a wide uniform adoption. One of the possible reasons for that is the great amount of such methodologies. Object-orientation itself only started to spread when Booch, Rumbaugh e Jacobson unified their modeling languages into UML. Therefore, although it is necessary to create new methodologies for new concepts, like RIA, it is important to rely on advances that are well established in the developer's community. Otherwise, our proposition would be "yet another web engineering methodology".

One of these concepts in systems modeling is Object-Orientation itself. That is why HMBS [7], RMM [8], HDM [9] and HMT [10] were not considered for this work's extension, since they are not OO based. HFPM [11] was not considered because it is not a methodology that adds contributions on navigation and presentation aspects. DEMAIS [12] was excluded because it is a tool an not a methodology *per se*.

From the remaining ones (WAE [5], OMMMA-L [13], W2000 [14], WebML [15], OO-H [16], UWE [17], OOHDM [18] and WSDM [19]) we used, as a first filter criteria, the amount of references in ACM Portal [20]. According to a research on the references to their main and derived papers, WebML, OOHDM and UWE are the most referenced ones (79, 69 and 60 references respectively). From those, only UWE is completely adherent to UML, which led to our choice.

Although RUX has undeniable merits, we have chosen to extend from another methodology because RUX does not use only UML. It involves an array of other languages and recommendations (like XICL, UiML, SMIL and XML Events) for concrete interface diagram models. If they allow a more precise definition for those models, they also imply in a more sloped learning curve to the software modeler. Using our proposed extension, UWE-R, a modeler only needs to know UML and a few extension mechanisms. Even the same modeling CASE tool can be used out of the box with UWE-R.

3.2 UWE Basic Concepts

In order to understand the proposed extensions in UWE-R, it is mandatory to understand UWE ([21] and [22]) first. Due to the size limitations of this paper, only its main characteristics will be introduced here.

UWE is an UML profile, or a light extension to UML. This means that only stereotypes, tagged values and constraints, which are standard UML extension mechanisms, are used. That is the reason

why any CASE tool can be used with UWE (and also with UWE-R, that follows the same conditions). There is a tool called ArgoUWE, based on ArgoUML, which goes a step further: it provides automatic checking for restrictions and perform some transformations, according to Model-Driven Architecture (MDA). Not using this tool does not prevent anyone to benefit from UWE and UWE-R immediately.

As an UML extension, UWE expresses its stereotypes and diagrams using a metamodel. In UML, a metamodel is a model, in a higher level of abstraction, which uses class diagram's notations to express features and concepts not originally conceived in UML. UWE is a conservative extension to UML version 2.0. By conservative, it means that none of the UML original elements are altered in any way. All new elements in UWE's metamodel are referenced as extensions (a similar mechanism to inheritance among classes) to elements in UML. In order to define the static semantics for these new elements, Object Constraint Language (OCL) is used. Since UWE is compatible with the Meta-Object Facility (MOF), a model that is written based on UWE's metamodel can be read in other CASE tools which support XML Metadata Interchange (XMI). As explained in UML's superstructure document [23]: in the instance model, a metaclass from the metamodel becomes a stereotype, a metaclass' attribute becomes a tagged value, and a role becomes a constraint expressed in OCL.

UWE's metamodel has two high level packages to group their metaclasses: Core and Adaptivity. Inside Core's package are located UWE's main sub-packages: Requirements, Content, Navigation, Presentation and Process. UWE-R introduces extensions to Navigation, Presentation and Process sub-packages.

4 UWE-R

UWE-R's purpose is not to be a complete Web engineering methodology for RIA. Its focus is on navigation, presentation and server interaction (processes) which are the differences between a traditional Web application and RIA. Other methodologies, including UWE, already effectively cover the remaining aspects, like requirements.

Since our goal is to keep maximum compatibility, UWE-R will also be a light extension or profile to UWE, meaning that no UWE metaclass will be modified. Every extension was conceived as add-ons to express RIA concepts.

4.1 Navigation Extensions

UWE already defines a metaclass for navigation (NavigationClass). In its definitions, though, there are restrictions which ties it to traditional Web applications: it refers to a hypertext structure and mandates a one-to-one relationship to a respective content class. Both restrictions are too constraining for RIA. For those applications, what would be a navigation class might not be in hypertext (might be inside a Flash object or Java applet) and several content classes can be related to it (see Google Maps, for instance). This required the creation of a new metaclass to represent this concept. It is important, though, that the basic UWE framework is respected and a composition relationship to UWE's NavigationProperty is leveraged (to reference the UI elements in the presentation model). That is why our new metaclass, RichNavigationClass, extends UWE's Node metaclass.

Additionally it is required to create a new metaclass for the link concept. UWE's NavigationLink metaclass can only express links between nodes that are not processes. As for RIA, a navigation can imply a local or remote process triggering, that would return to the same RichNavigationClass (visual continuity), potentially changing some of its attributes. Besides, there are characteristics, like asynchronism which are relevant to be modeled in this new type of link. This one is called RichNavigationLink. In Fig. 1, a metamodel diagram excerpt is shown. Only UWE-R's new metaclasses (in gray) and their direct associated metaclasses from UWE are presented.

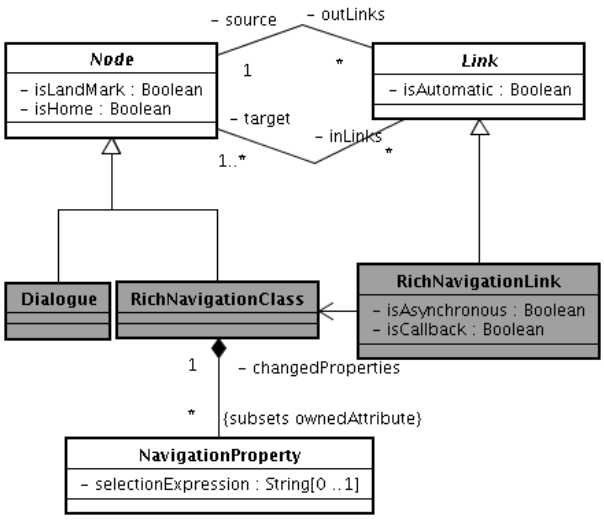

Fig. 1: UWE-R Navigation's extensions

A class to represent a dialogue was also introduced, since this type of navigation entity interposes itself in a different way to the navigation nodes.

An example that illustrates those metaclasses is in Fig. 2:

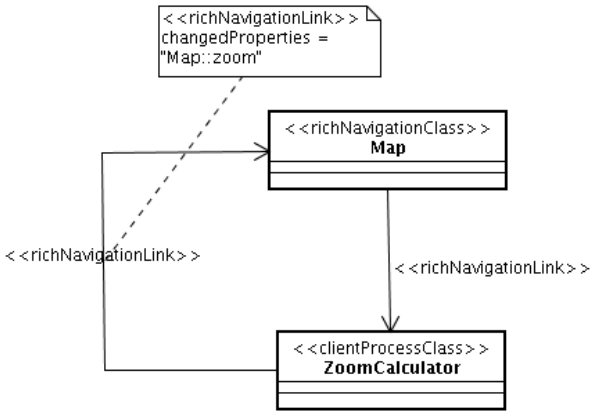

Fig. 2: Synchronous navigation example

A new stereotype (clientProcessClass) is used above, which is part of our UWE-R extension in the Process sub-package. It will be further detailed in section 4.3, but it is already possible to figure out that it represents a process (set of actions taken by the application) that runs on the client side (browser). In the case where a process needs to be executed on the server side, the rich navigation link can be modeled using the asynchronous attribute from its metaclass. And its response will be typically a callback to the client. This is the main novelty in RIA (asynchronous server interaction) and can be expressed as in Fig. 3:

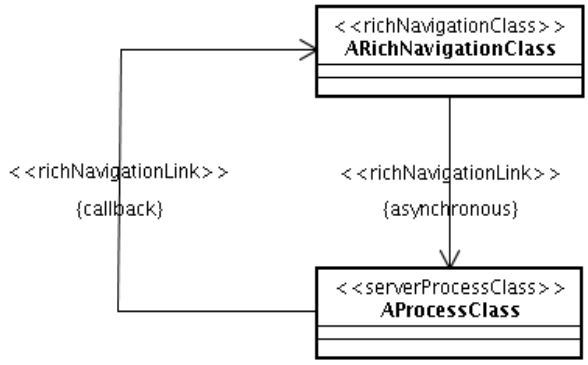

Fig. 3: Asynchronous navigation example

One might argue that there is some information overload in the diagram with stereotypes and tagged values on the links. In a complete model this can really harm readability. If that is the case, then some of these can be suppressed. Others can be conventionally taken for granted, like the one above: if there was an asynchronous request to the server, the response will always be a callback. Some UML CASE tools allow adding all those informations, but choosing to show them or not. What matters most is that the concept and according notation is clear for developers and modelers. A suggested good practice when using diagrams as the above, is not trying to grasp the whole application with a single one, but having one navigation diagram per use case. If in a traditional Web application it is possible to follow a great variety of links from a node, in RIA this is even more critical, due to its visual continuity.

4.2 Presentation Extensions

From the presentation point of view, it is important to add some metaclasses to express RIA richness in the UI aspect. However it is not necessary to create a new metaclass to extend the already existing PresentationClass from UWE. This one carries the idea of nesting various presentation elements (by the so called inclusion trees). It also allows composing UIElement metaclasses (which inherits from PresentationElement) with PresentationProperty metaclass. This last one is associated with a PresentationElement, providing such composition.

Therefore, only metaclasses that extend from UIContainer and UIElement were created. They are shown in Fig. 4:

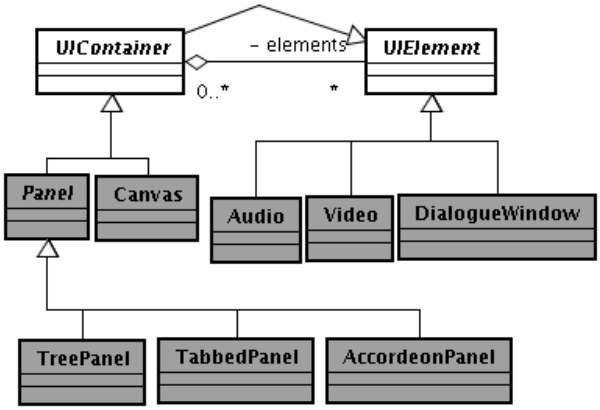

Fig. 4: UWE-R presentation's extensions

The Canvas metaclass is the most important, from the RIA perspective in this sub-package. It represents a free drawing area where mouse events can be captured. As shown in Fig. 4, other elements can be composed and visually overlaid in a Canvas, like images and other canvases. Before this extension, there were only 2 types of UIContainers: Form and AnchoredCollection in UWE. With Canvas it is possible to model Geographical Information Systems (GIS) and games. Each overlaid canvas can be a map layer or the game scenario and characters.

UWE's presentation metamodel intends to be a rather abstract one, i.e., not concerned with user interface details, like component size, exact position, etc.. Nonetheless it is interesting to provide a generic advice of visual layout for the UI elements. The Panel metaclass was created with that purpose. Some sub-metaclasses were also proposed as in Fig. 4.

Much of the richness in RIA comes from the possibility to mix video an audio content in a Web application. This justifies the corresponding metaclasses above. At last, DialogueWindow represents a very common type of UI element.

One last note on this sub-package that will be further detailed in the next section: according to UWE, when UI elements belong to the same inclusion tree, the respective navigation nodes are presented as if the links were automatically followed. If they don't belong to it, then it is necessary a user action (modeled by the metaclass UserAction) to trigger this presentation. As it will be shown in 4.3, in RIA there is a type of action for which this is not true. Actions can be related not only to the user, but also to browser events and server callbacks. Those events can trigger a presentation change as well. They will be modeled as the AutonomousAction metaclass in the Process sub-package.

4.3 Process Extensions

In this sub-package, there are three extensions proposed by UWE-R: distinguishing client from server processes, sequence diagram usage with a specific message type (ControlMessage) and creation of AutonomousAction metaclass to model user independent actions. Below, UWE-R's extensions are shown diagrammatically:

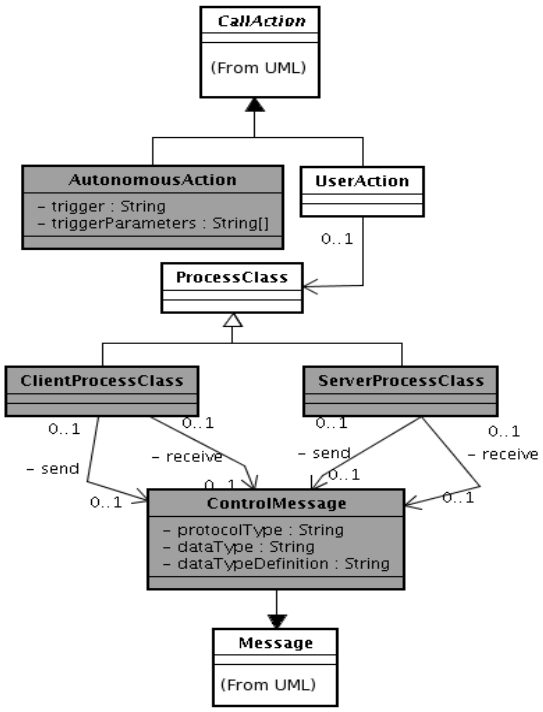

Fig. 5: UWE-R process' extensions

Distinguishing if a process runs on the client side (in a browser as a script or plugin API) or on the server side is very relevant in the RIA context. Since for this type of Web applications there is an important part of the code that runs on the client side, it is worthwhile having mechanisms to model

this fact. Eventually one might use the ProcessClass metaclass from UWE in the analysis phase. That stage has a higher level of abstraction. However at the design phase, this separation is very significant. If UWE-R were not intended to be a light extension to UWE, the ProcessClass metaclass could be changed into an abstract metaclass.

UWE's process sub-package has three goals: 1) integrating business processes into the navigation model, 2) defining a user interface to support the process and 3) defining the application behaviour. In RIA, there are asynchronous communications and data is not always transferred as simple HTTP GET or POST with parameters. Other formats, like XML (using or not the Simple Object Access Protocol – SOAP) or JSON can be used as well. Therefore, extensions are provided in UWE-R to model this reality. Sequence diagrams are used for that purpose (in UWE, only activity diagrams were used to detail the processes). A specific metaclass is proposed: ControlMessage (extending UML's "Message" metaclass from BasicInteractions package). Its name is borrowed from the Model-View-Control (MVC) pattern, since it is a message between model and control layers. UML's Message already has an attribute (messageSort) that indicates whether the message is synchronous or not. It can be visually distinguished by the arrow head. ControlMessage new attributes are: isCallback (only true if it is a callback from the server to the client), protocolType (reserved values: "LOCAL" when inside the client, "HTTP" for simple GET/POST, "FTP", "REST" for Representation State Transfer and "SOAP"), dataType (reserved values: "JSON", "XML", "KML", "GML" and "STRING") and dataTypeDefinition (W3C's Schema or DTD element that defines the XML type, or EBNF that defines the JSON format). As an example, see Fig. 6:

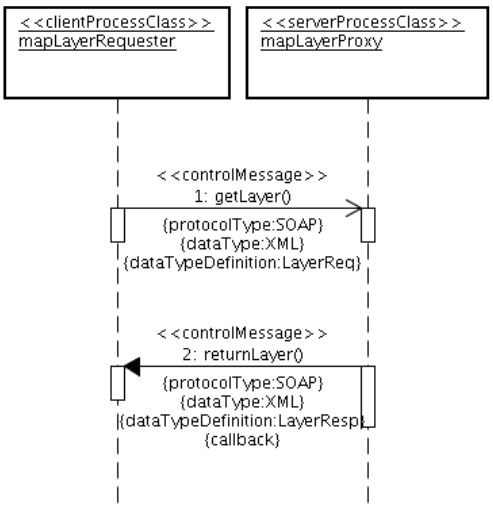

Fig. 6: ControlMessage example

In Fig. 6 an asynchronous request is performed from the client to the server to get a map layer (with type LayerReq, from a W3C's Schema element) and the response (with type LayerResp, also from a Schema) comes synchronously from the server as a callback. Consequently during the design phase a very clear picture – with frequency and data type details – on how the application exchanges data is formed. This is particularly important to ensure that the application is "rich", not only from the UI point of view but also in the user interaction perspective. Nowadays it is very common to find Web sites that use RIA technologies but their design is the same as in traditional Web applications. Many of them do not take advantage of asynchronous interaction and leads to a poor result. With a diagram as in Fig. 6, the lack of a RIA oriented design is made evident before implementing the code.

There are other advantages of using ControlMessage metaclass in testing and performance evaluation. Testing as soon as possible is one of the reasons for modeling an application. The same way that use cases definition enables test cases definition, using diagrams as above enables unit and integration tests definitions sooner. As for performance, data types and interaction's frequency are clearly shown in such a diagram. From those, data transfer volumes can be estimated and eventual bottlenecks can be easily detected and solved.

UWE-R's last extension in this sub-package is AutonomousAction metaclass creation. This one is

different from UWE's UserAction, because it does not depend on user interaction. In RIA, an action can be triggered by a browser event (on page load, for instance), by a timer or a server callback. This metaclass' attributes are: triggerType (reserved values: "EVENT" for browser events, "TIMER" and "ONCALLBACK"), triggerName (browser's event name of callback function name. Not used for timer) and triggerParameters (parameter set for browser event or callback function, if necessary, or the timer interval). To avoid confusion with UML's AcceptEventAction metaclass, a name like "EventAction" was not chosen for this metaclass.

5 Applying UWE-R to Google Maps

This section will show how to apply UWE-R to Google Maps Web site as a demonstration of this extension's expressive power. Web GIS are complex enough to justify UWE-R's usage. Since that site's source code is not available, what is shown here is a reverse engineering model, observing the site's behaviour. Having a browser opened at Google Maps [24] will ease this section's understanding.

Navigation, presentation and process models will be presented covering one application use case with emphasis on UWE-R's new elements. Search by address was the chosen use case. Navigation model follows:

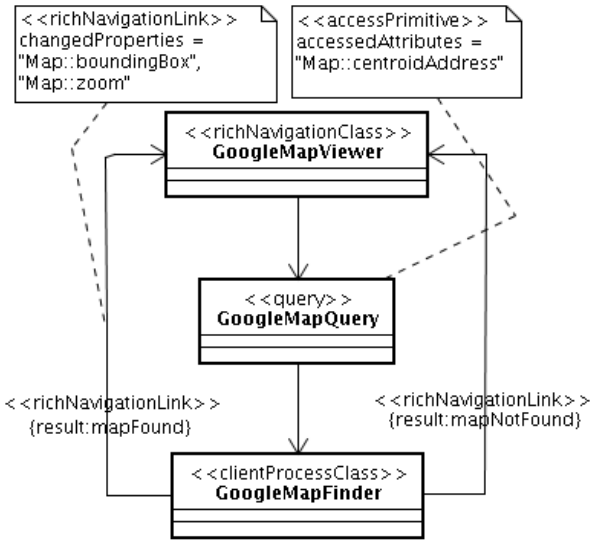

Fig. 7: UWE-R for Google Maps: navigation

Most noticeable is the rich navigation class (GoogleMapViewer). In this case and the vast majority, while navigating all links return to this very same node, showing the typical RIA visual continuity. A navigation class with <<query>> stereotype is also used. This stereotype (a metaclass in the metamodel) is UWE's. The other element, GoogleMapFinder, has UWE-R's <<clientProcessClass>> stereotype. Therefore this is a process that runs on the browser, with a possible server interaction to be further explained later on. The <<richNavigationLink>> stereotypes denote the process return to the same node, but with some changed attributes: the maps bounding box and zoom are changed to reflect the search result. Both situations (map was found or not) are shown by the "result" tagged value (between curly braces) defined in the respective metaclass as its attribute. That is a synchronous request situation, thus no "callback" tagged value is shown.

Following UWE's guidelines only the main process (GoogleMapFinder) is drawn in the navigation model. It is necessary to have a process model (represented by an activity diagram) that details how the whole process is executed. Before that, a class diagram must indicate the processes that are part of this main one. In our case, this is such diagram:

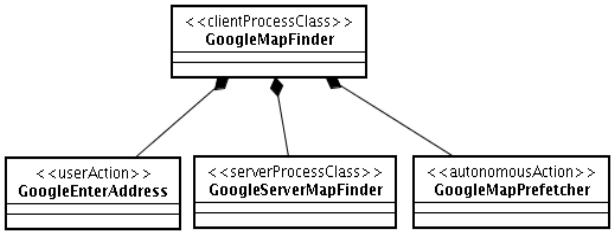

Fig. 8: UWE-R for Google Maps: process classes

In Fig. 8 a UWE's stereotype to represent a user action is employed. UWE-R's <<serverProcessClass>> stereotype is used to indicate another process that might be invoked by the client as it will be clear in the following UWE's process model:

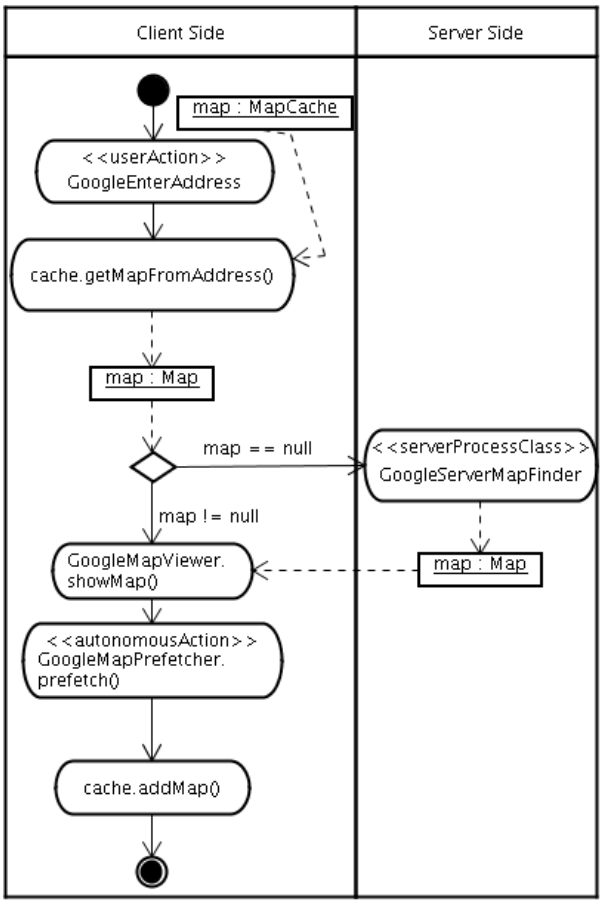

Fig. 9: UWE-R for Google Maps: process model

In this model swimlanes were used to make it visually explicit the client and server side interactions. Initially on the client side the map is searched in the cache, according to the user input address. As any well designed RIA, Google Maps shows this behaviour: the server is accessed only when it is absolutely necessary. It can be observed, while using it, that if the map is already in the local machine no server request is performed. This suggests some caching mechanism, although we had no access to Google Maps source code. This behaviour can also be found when the user tries to displace the map to contiguous regions. Google Maps brings those adjacent maps asynchronously while the user is not interacting with the application. Therefore Google Maps can provide the illusion of a desktop application, that is server independent, because the user displaces the map and the neighbour image is promptly displayed. This can only be modeled with UWE-R's AutonomousAction as in the following sequence diagram:

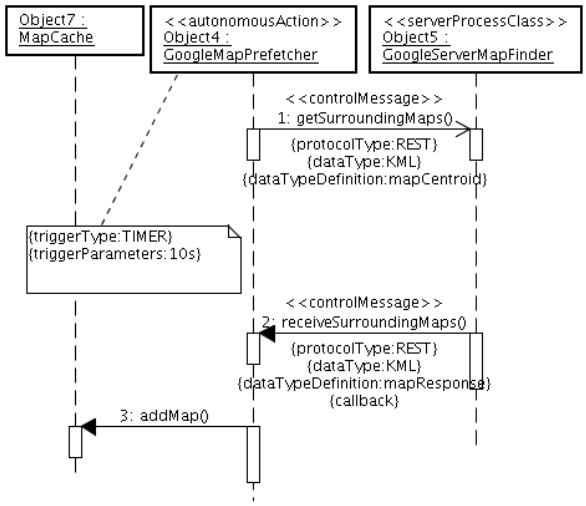

Fig. 10: UWE-R for Google Maps: map prefetch

The autonomous action GoogleMapPrefetcher has the tagged values “TIMER” and “10s” as its trigger type and parameter. This means that after 10 seconds of user inactivity, this action is triggered. The received map is added to the cache to avoid future unnecessary server requests.

In Fig. 10 the message traffic is explicit. <<controlMessage>>, as proposed by UWE-R, is used and shows an asynchronous server interaction. The tagged values make it clear what are the used data types and protocols. In this case, KML was assumed, since this is Google's format for this type of data. The response comes with a callback, as indicated.

At last, the presentation model is as follows:

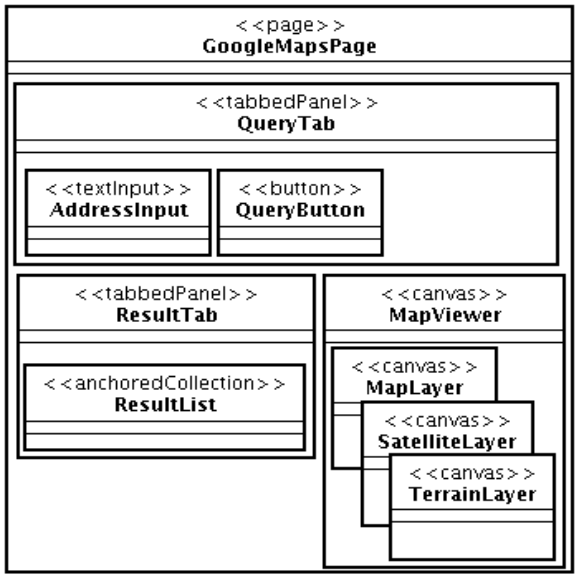

Fig. 11: UWE-R for Google Maps: presentation

This model abstracts from other UI elements that can be found in other panels (like texts, images and links) for the sake of readability. If a more literal model is desirable, it would be enough to add a presentation class inside GoogleMapsPage, with those additional elements. Notice the convenient usage of tabbed panels, which are in fact present in Google Maps site, and the canvas classes to represent the map region with its several layers.

4 Conclusion

In this paper UWE-R was introduced and used in a complex web site. Google Maps has a great variety of features and makes extensive usage of RIA concepts. In our models, those concepts were accordingly modeled. Therefore, we believe that UWE-R has a good expressive power to model RIA and leverages existing concepts and tools. Naturally there are some future possible works. Among them, we can include: more complete example models, more UI elements metaclasses and a complete and formal metamodel specification.

References:

[1] Klein, N., Carlson, M. and McEwann, G., *Laszlo in Action*, Manning Publications, 1st. Edition, Chapter 1, 2007.

[2] Adobe. *Rich Internet Applications*, http://www.adobe.com/resources/business/rich_internet_apps/, 2007.

[3] Zakas, N., McPeak, J. and Fawcett, J., *Professional AJAX*, Wiley Publishing, Inc., 2nd. Edition, 2007, pp. 299-335.

[4] Preciado, J., Linaje, M. and Sanchéz, F., Necessity of methodologies to model Rich Internet Applications*, Proceedings of the 2005 Seventh IEEE International Symposium on Web Site Evolution (WSE'05)*, 2005, pp. 7-13.

[5] Conallen, J., *Building Web Applications with UML*, 2nd Edition, Addison-Wesley Longman Publishing Co., 2002.

[6] Preciado, J., Linaje M. and Sanchéz, F., Enriching Model-based Web Applications Presentation, *Journal of Web Engineering*, Vol. 7, Number 3, 2008, pp 239-256.

[7] Oliveira, M., Turine and M., Masiero, P., A Statechart-Based Model for Hypermedia Applications, *ACM Transactions on Information Systems*, Vol. 19, No. 1, 2001, pp. 28–52.

[8] Isakowitz, T. and Balasubramanian, P., RMM, A Methodology for Structured Hypermedia Design, *Communications of the ACM*, ACM Press, vol 38, issue 8, 1995, pp. 33-44.

[9] Garzotto, F., Paolini, P and Schwabe, D., HDM — A Model-Based Approach to Hypertext Application Design, *ACM Transactions on Information Systems*, Vol. 11, IWO. 1, 1993, pp. 1-26.

[10] Specht, G. and Zoller, P., HMT: Modeling Temporal Aspects in Hypermedia Applications, *1st International Conference on Web-Age Information Management*, Springer-Verlag, Shanghai, 2000, pp. 256-270.

[11] Olsina, L., Building a Web-based information system applying the hypermedia flexible process modeling strategy, *1st ACM International Workshop on Hypermedia Development with Hypertext*, 1998.

[12] Bailey, B., Konstan, J. and Carlis, J., DEMAIS: Designing Multimedia Applications with Interactive Storyboards, *9th ACM international conference on Multimedia*, ACM Press, 2001, pp. 241-250.

[13] Sauer, S. and Engels, G., Extending UML for Modeling of Multimedia Applications, *IEEE Symposium on Visual Languages*, IEEE Computer Society, 1999, pp 80-87.

[14] Barei L., Garzotto, F. and Maritati, M., W2000 as a MOF Metamodel, *6th World Multiconference on Systemics Cybernetics and Informatics - Web Engineering track*, vol. I, 2002.

[15] Ceri, S., Fraternali, P. and Bongio, A., Web Modeling Language (WebML): a Modeling Language for Designing Web Sites, *9th International WWW Conference*, 2000, pp. 137-157.

[16] Gómez, J. and Cachero, C., *Information Modeling for Internet Applications*, Idea Group Publishing, pp. 144-173, 2003.

[17] Koch, N. and Kraus, A., The Expressive Power of UML-based Engineering, *Second International Workshop on Web Oriented Software Techonlogy*, CYTED, 2002, pp. 105-119.

[18] Schwabe, D., Rossi, G. and Barbosa, S., Systematic Hypermedia Design with OOHDM, *7th ACM International Conference on Hypertext*, ACM Press, 1996, pp.116–128.

[19] De Troyer, O. and Leune, C., WSDM: A User Centered Design Method for Web Sites, *Proceedings of the 7th International World Wide Web Conference*, 1998, pp. 85–94.

[20] ACM Portal, *The ACM Portal*, http://portal.acm.org, 2008.

[21] Koch, N., Knapp, A., Zhang, G. and Baumeister, H., UML-Base Web Engineering – An approach based on standards, chapter 7 in *Web Engineering: Modelling and Implementing Web Applications*, Springer Verlag, 2008, pp. 143-177.

[22] Kroiβ, C. and Koch, N., *The UWE Metamodel and Profile – User Guide and Reference*, http://www.pst.informatik.uni-muenchen.de/projekte/uwe/download/UWE-Metamodel-Reference.pdf, 2008.

[23] OMG, *OMG Unified Modeling Language (OMG UML), Superstructure V2.1.2*, http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF/, 2007.

[24] Google Maps Homepage, *Google Maps*. http://maps.google.com/, 2008.