Modelling the Requirements of Rich Internet Applications in WebRe

Esteban Robles Luna[1], M.J. Escalona[2], and G. Rossi[1]

[1] LIFIA, Universidad Nacional de La Plata, La Plata, Argentina
[2] IWT2, Universidad de Sevilla, Sevilla, Spain
{esteban.robles,gustavo}@lifia.info.unlp.edu.ar,
mjescalona@us.es

Abstract. In the last years, several Web methodological approaches were defined in order to support the systematic building of Web software. Together with the constant technological advances, these methods must be constantly improved to deal with a myriad of new feasible application features, such as those involving rich interaction features. Rich Internet Applications (RIA) are Web applications exhibiting interaction and interface features that are typical in desktop software. Some specific methodological resources are required to deal with these characteristics. This paper presents a solution for the treatment of Web Requirements in RIA development. For this aim we present WebRE+, a requirement metamodel that incorporates RIA features into the modelling repertoire. We illustrate our ideas with a meaningful example of a business intelligence application.

Keywords: Rich internet applications, Web engineering, Web requirements.

1 Introduction

It is widely known that the Web is constantly evolving. In this evolution, Rich Internet Applications (RIA) [1] represents a major breakthrough, as they allow combining the typical navigation flavour of the Web with the interface features of desktop applications. These applications allow reducing the communication between clients and servers since pages (differently from the "navigational" Web) do not need to be fully reloaded with each user interaction. The emergence of a well-known set of RIA patterns [2] has additionally defined a small, though complete, vocabulary for expressing desired interaction functionalities in a software system. It is now common saying: "this should be an auto-complete field" or, "we can use hover details for showing this information". Not surprisingly applications stakeholders also use this vocabulary as part of their requirements for a new application.

However, though most Web design methods have been already extended to cover the scope of RIA [3][4][5], there is still an important gap in requirement specification of RIA functionality, since requirement specification and modelling languages do not include suitable primitives for expressing this kind of requirements. In this way,

checking whether a requirement has been fully implemented becomes a subjective matter, and it is not possible to automate this process (e.g. by automatically generating tests from requirement specifications).

In this paper we analyze the new kind of requirements that occur in RIA, and how we can extend an existing approach to specify the behaviour of this kind of applications in a MDWE (Model-Driven Web Engineering style) [6]. Specifically, we use an enhanced version of the WebRE metamodel [7] to specify RIA requirements.

The paper has two aims. Firstly, we show how our modelling approach for specifying Rich web requirements is integrated in the NDT approach. In addition we show the integration between mockups and our metamodel to improve requirements elicitation with customer. Finally, we show how to derive a set of interaction tests from WebRE+ models to validate the RIA functionality.

The paper is structured as follows: Section 2 presents the background for this work; we present the NDT approach and the WebRe metamodel. In Section 3, we show the extension of WebRE for RIA, its UML profile and how our metamodel is used with Mockups. Section 4 presents how tests are derived from WebRE+ models and section 5 shows a case of study with an example in the Business Intelligence area. In section 6 we present the implementation of our metamodel and in section 7 the related works in Web requirements, Model-Driven Web Engineering and RIA. Finally we present the conclusions and future research work in this project.

2 Background

In this section we introduce the NDT approach that gives a good context for our metamodel and the original version of our metamodel called WeRe which does not support rich requirements.

2.1 NDT

NDT [14] is the acronym for Navigational Development Techniques, is a member of the growing family of MDWE approaches. Initially, NDT dealt with the definition of a set of formal metamodels for requirements, based on the WebRE metamodel. In addition, NDT defined a set of derivation rules, expressed with the standard QVT, which generate analysis models from requirements models.

Nowadays, NDT defines a set of metamodels for every phase of the life cycle of software development: the feasibility study phase, the requirements phase, the analysis phase, the design phase, the implementation phase, the testing phase, and finally, the maintenance phase. Besides, it includes new transformation rules to systematically generate models. Fig. **1** shows the first part of the NDT lifecycle[1].

The main goal of *the Requirements phase* is to build the catalogue of requirements which contains the needs of the system to be developed. It is divided into a series of activities: capture, definition and validation of requirements.

NDT classifies system requirements according to their nature: information storage requirements, functional requirements, actor requirements, interaction requirements,

[1] You can get more information about the NDT full lifecycle in `www.iwt2.org`

and non-functional requirements. In order to define them, NDT provides special patterns and UML compliant artefacts, such as use cases for functional requirements specification.

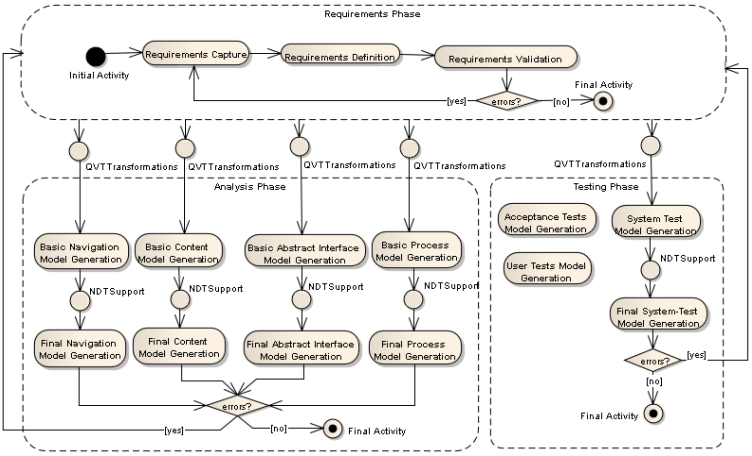

Fig. 1. Transformations from Requirements to Analysis and from Requirements to Testing

Once the requirements specification phase has been completed and the catalogue of system requirements has been drafted and validated, NDT defines derivation rules to generate the system test model and the analysis phase models. Fig. **1** shows all these transformations through the stereotype *«QVTTransformation»*.

NDT conceives the Testing phase, as an early phase of the software life cycle and proposes to carry it out together with the remaining phases. NDT defines three models in this phase: implementation tests model, system tests model and acceptance tests model. The system tests model is the only one that can be generated systematically. NDT comprises derivation rules to generate the basic model of system tests from the functional requirements defined in the requirements phase. The team of analysts can perform transformations in order to enrich and complete this basic model**.** Transformations are represented in Fig. **1** through the stereotype *«NDTSupport»*.

The Analysis phase includes the resulting products from the analysis, definition and organization of requirements in the previous phase. At this phase, NDT involves four models: the conceptual model, which represents the static structure of the system; the process model, which represents the functional structure of the system; the navigation model, which shows how users can navigate through the system and the abstract interface model, a set of prototypes of the system.

The transition between the requirements and the analysis model is standardized and automated, and it is based on QVT transformations, which translate the concepts of requirements metamodels to the first versions of the analysis models. These models are known in NDT as basic models of analysis. For example, the basic conceptual model of analysis is obtained from the storage requirements defined during the requirements phase.

Thereafter, the team of analysts can transform these basic models to enrich and complete the final model of analysis. Since this process is not completely automatic, the expertise of an analyst is required. To ensure consistency between requirements and analysis models, NDT controls these transformations by means of a set of defined rules and heuristics.

To sum up, NDT offers an environment suitable to the development of Web systems, completely covering life cycle of software development. This environment is named NDT-Suite.

The application of MDE and, particularly, the application of transformations among models may become monotonous and very expensive if there are no software tools that automate the process. To meet this need, NDT has defined a set of supporting tools called NDT-Suite. Currently, the suite of NDT comprises the following free tools:

- NDT-Profile is a specific profile for NDT, developed using Enterprise Architect [8]. NDT-Profile offers the chance of having all the artefacts that define NDT easily and quickly as they are integrated within the tool Enterprise Architect.
- NDT-Quality is a tool that automates most of the methodological review of a project developed with NDT-Profile. It checks both, the quality of using NDT methodology in each phase of software life cycle and the quality of traceability of MDE rules of NDT.
- NDT-Driver implements a set of automated procedures that enables to perform all transformations MDE among the different models of NDT that were described in the previous section.
- NDT-Prototype is a tool designed to automatically generate a set of XHTML prototypes from the navigation models, described in the analysis phase, of a project developed with NDT-Profile.
- NDT-Glossary implements an automated procedure that generates the first instance of the glossary of terms of a project developed by means of NDT-Profile tool.
- NDT-Checker is the only tool in NDT-Suite that it is not based on the MDE paradigm. This tool includes a set of sheets, different for each product of NDT. These sheets give a set of check lists that should be reviewed manually with users in requirements reviews.

To conclude, one of the most important characteristics of NDT is that has been applied in many practical environments; it has succeeded mainly due to the application of transformations among models, which has significantly reduced development time.

2.2 WebRE

WebRE is a metamodel that includes modelling artefacts to deal with requirements in Web applications; it uses the power of metamodelling to fuse different approaches. WebRE was born from the exhaustive analysis of different Web engineering proposals that deal with requirements. It unifies the criteria of these proposals and defines a unified metamodel for the CIM (Computer Independent Model) level. It

provides a base to decide which concepts or elements must be captured and defined in the requirements phase of Web applications. The metamodel defines each of these concepts and the relationships between them.

With this unification, WebRE overcomes an important gap: with the use of a common metamodel, it abstracts from the multiple notations used in each approach. Each artefact defined in WebRE can be mapped to an artefact in each of the different requirement engineering approaches. Besides, WebRE also comprises an UML Profile with a concrete syntax to represent each concept. Thus, a development team can specify an application's requirements using the WebRE profile, and later (when necessary) map them to concrete model elements to continue with the selected methodology (NDT, UWE, W2000 or OOHDM). Additionally, it would be possible to systematically derive the corresponding navigation models from requirements expressed in WebRE using suitable transformations.

However, WebRE was originally conceived for Web 1.0 applications and therefore it does not support specification of RIA behaviours. The extension proposed in this paper allows the systematic generation of models for Web 2.0 applications and the generation of tests to validate the RIA functionality (Section 4). In the following sections we show how we enriched the WebRE metamodel with new metaclasses and metaassociations in order to provide an approach that covers both: Web 1.0 and Web 2.0 requirements.

3 Metamodelling Rich Requirements with WeRe+ in NDT

Expressing RIA behaviour and specifically supporting the use of RIA patterns in requirements using a metamodel have many benefits such as:

- Making possible to develop the application easier by automatically deriving concrete software artefacts,
- Allowing the generation of tests to automatically validate the requirements
- Supporting requirements evolution and
- Improving traceability between requirements and the implementation.

In the following subsections we show its RIA extension and the corresponding UML profile. Also, we present in which activities of the NDT approach the WeRe+ modelling is used with mockups.

3.1 WebRE+

RIA have particular features like sophisticated interactive behaviour, client-side feedback of "slow" operations and different kinds of client-side behaviour depending on the occurrence of events, among others. An example of the last feature is shown in Figure 2. The line graph shows information about the progress of a business across time. As a consequence of how progress is measured (it requires certain calculations) we only show the final computed value in the graph. The details of how those values were computed are shown only when the user shows interested in it (e.g. when the user puts its mouse over an item). This solution is well known as a hover detail pattern in the Yahoo Patterns

catalogue. This kind of RIA behaviour improves applications usability without polluting the user interface with lots of information, which could be unnecessary at first sight. To provide a precisely specification of this kind of requirement we need to deal with concepts such as events, UI elements like buttons, textfields, etc. For this reason we extended the WebRE metamodel with these concepts as shown in figure 3.

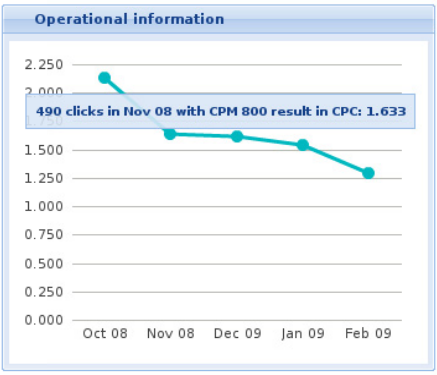

Fig. 2. Hover detail pattern on a line graph

In WebRE+ the original packages, *structure* and *behaviour*, were kept to preserve the mapping between the concepts present in WebRE+ and its ancestors.

The structure package includes each concept to deal with the conceptual aspect of Web requirements. Since RIA applications mainly deal with client side behaviour, we add the *UIElement* metaclass. Instances of this metaclass are: *buttons*, *textfields*, *images*, *checkboxes*, etc. To support RIA we extended the metamodel with two new metaclasses: *RIASpecification*, which represents a definition of a set of scenarios that a RIA behaviour must satisfy and *RIAScenarioSpecification*, which describes any *RIASpecification* in a concrete scenario. For example, in the hover detail feature (a *RIASpecification* instance), we must specify two different scenarios (*RIAScenarioSpecification* instances), namely *Hover detail appears*: When the user puts the mouse over an item then a *UIElement* must appear after 2 seconds. This *UIElement* must contain a name and a description of the item; *Hover detail disappears:* When the user moves the mouse out of the item, then the *UIElement* with the details of the item must not be shown.

The *behaviour* package includes metaclasses to represent user's interaction and navigation. We extended the package with the *Event* metaclass which is important to specify different situations; for example, when the user puts the mouse over an item, when the user types something on a field, etc. In this case, we differentiate between two different subclasses: those events which are originated with the keyboard (subclass *KeyboardEvent*) and those which are originated with the mouse (subclass *MouseEvent*). Also, we include a new metaclass *UIAction* which captures the actions that the user can perform over an element in the UI of the application (relationship between *UIAction* and *UIElement*). Instances of *UIAction* are "click", "type keys", and execution of one of the actions may produce many events, e.g. when typing a key on a user interface element three events are fired, namely *onpressdown*, *onpresskey* and *onpressup*.

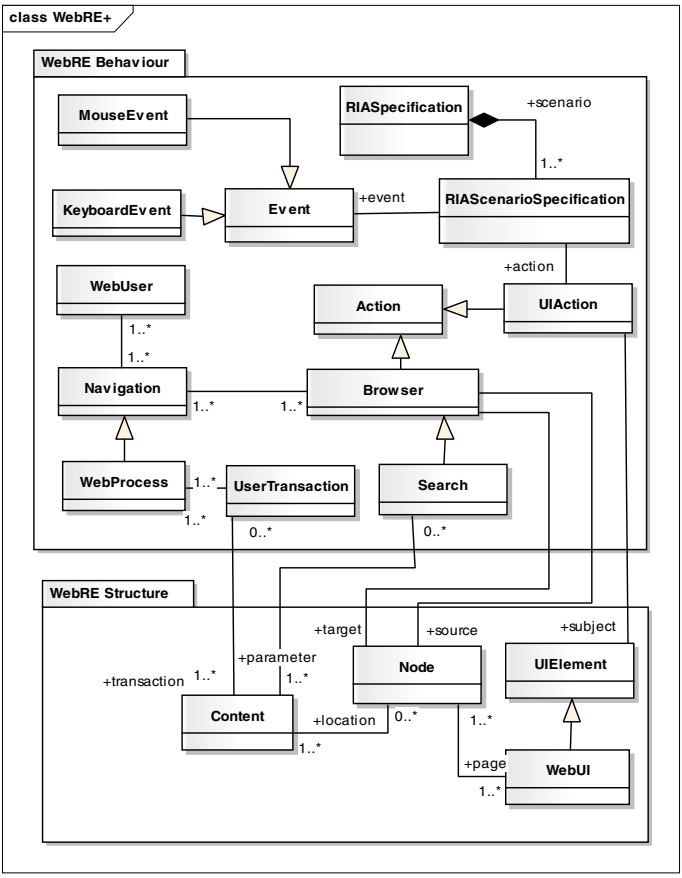

Fig. 3. WebRE+ metamodel

In the following subsection we describe our implementation of the UML profile.

3.2 A UML Profile for WebRE+

In order to provide editing support for our approach, we developed an UML profile for WebRE+, and implemented it using the Enterprise Architect tool. The use of UML profiles to provide tool supports is being used as a solution in some Web design approaches like UWE with MagicUWE [23] and specifically in NDT with NDT-Profile [24].

In Figure 4 we present the profile for WebRE+. As WebRE has its own profile, we only show our extension; that is, the metaclasses we have added to create WebRE+.

Each metaclass of WebRE extends an UML metaclass. Thus, we map our artefacts onto UML ones and define for them a set of characteristic that we could, even, improve with specific tag values or constraints.

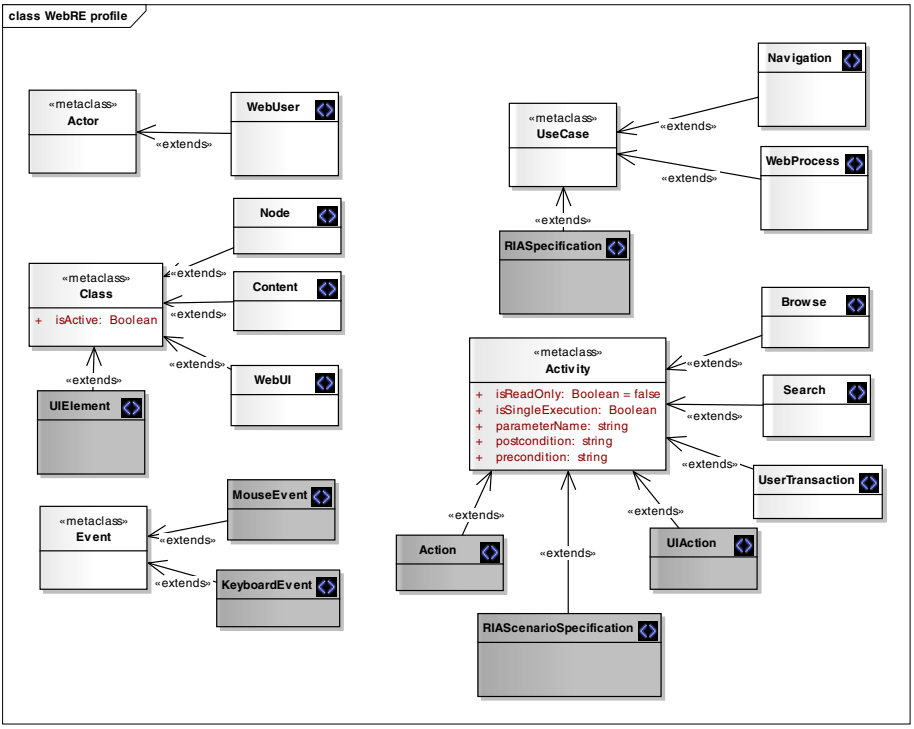

Fig. 4. WebRE+ profile

3.3 WebRE+ and Mockups in the Context of NDT

Specifying rich requirements using the presented metamodel might be easy for developers but when used in a work session with a customer it may not be suitable, since customers do not understand the concepts behind a metamodel. To overcome this problem and in syntony with existing approaches in agile web application development [31] we can use WebRE+ with mockups to simplify the requirements elicitation phase.

A mockup is a sketch of a User Interface (UI) which shows an example state of the system to be. It is useful when interacting with customers, as it is clear of what would be the software's UI. An example of our business case example is shown in Fig. 5 where we show the 2 possible states of the UI according to the mouse events.

Using mockups and WebRE+ we can specify a variety of rich requirements and allow customers to be involved in the process. However, it is important to define the activities involved in this process. In Fig. 6 we show those activities when using mockups with WebRE+ models. We start creating mockups (Step 1) as it defines a good basis to start discussing with customers. Also, they can be created really fast (within few minutes) and the feedback obtained from customers is good. Afterwards, we can create/update our WebRe+ models (Step 2); during this activity the analyst may notice incomplete and even contradictory requirements while formally specifying the requirement, and therefore it is necessary to create extra mockups to discuss with customers. After a few iterations, we can conclude this process (Step 3).

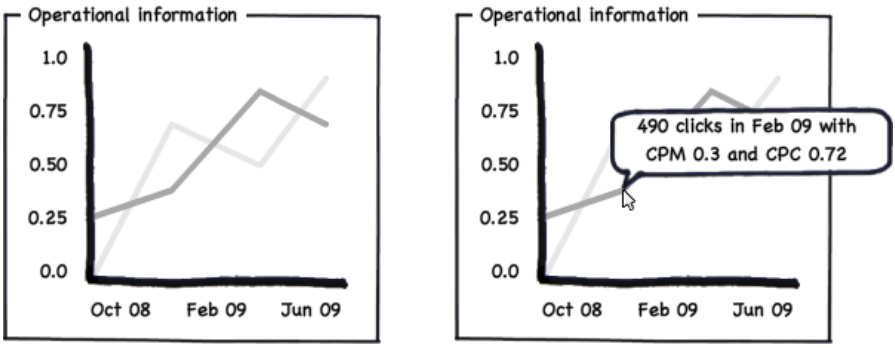

Fig. 5. Mockups for the hover detail requirement

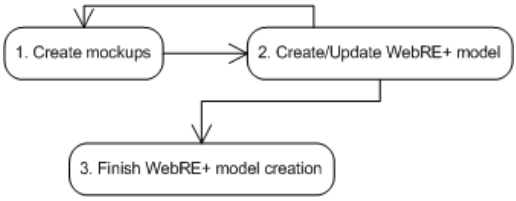

Fig. 6. Synthetic process for the creation of mockups and WebRE+ models

4 Test Derivation

Requirements validation is usually a hard and time consuming task which is performed by a quality assurance team after the application has been implemented. Generally, it is done manually (because requirements are captured using informal documents such as Use Cases or User Stories), by creating a set of tests that validate the requirements. The tests are run and if they pass, then the application can be deployed to production.

Using the formal definition that WebRE+ provides, we can use the requirement specification to derive these tests automatically thus reducing the time spent on the process and bridging the gap between requirements and tests. The process transforms a WebRE+-based model into a test model (Figure 5) that is independent of the platform. The transformation process follows these steps:

```
For each RIASpecification:
  Create a test suite.
  For each RIAScenarioSpecification:
       Create a test.
       Add the actions of the scenario in the test.
       Add an assertion for the post condition of the scenario.
```

The test model is then transformed into a concrete test implementation. So far, we have use Selenium [25] for this purpose, although we could use a different framework

such us Watir [26]. We have chosen Selenium because it is one of the most popular testing frameworks that simulate user input and it is widely used in industrial settings. Also, a Selenium test could be re-written in almost any programming language and run on a Selenium server whereas Watir depends on Ruby [30].

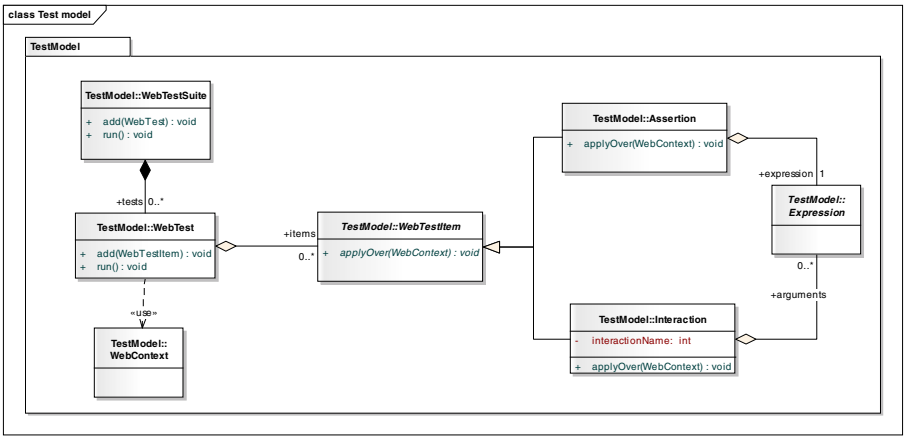

Fig. 7. Test model

In the following section we illustrate the use of the metamodel with a specific RIA requirement in the context of a Business Intelligence application showing how we specify it using WebRE+ and how tests are derived to Selenium.

5 A Case of Study

The business intelligence area is an example of how to use RIA to improve the user experience. For example, Pentaho BI suite [27] uses the Web environment to show data and allows users to execute queries to the data warehouse. A line graph that shows the progress of the business (Figure 2) could provide details about each value shown using the hover detail pattern.

Let's suppose that we are developing a Business Intelligence Web application for a company whose core business is organizing campaigns for different customers and providing summary reports to them. To improve the usability of the summary report which contains the line graph of Figure 2, we would like to add hover details to the items to show how those values are computed. For example, on a particular day there have been 3245 clicks and 15687 impressions so the CPC (Cost per click) is 0.34.

As in every RIA pattern, there are some features that can be configured and should be specified during the requirement elicitation phase. A simplified instance model of the WebRE+ specification for this requirement is shown in figure 6. The model shows that when the item receives an *onmouseover* event, a detail of the item must be shown in the page in less than 2 seconds. This widget must contain a label with the money used in the campaign and the number of clicks.

The WebRE+ instantiation describes the possible scenarios that the RIA behaviour must satisfy. Using the transformation explained in Section 4 we transform this model into an instantiation of the test metamodel and then we derive the test suite to the Selenium framework. The derived tests are shown next:

```
Test 1
(01) s.open(reportURL);
(02) s.mouseOver("id=item1");
(03) Thread.sleep(1000);
(04) assertTrue(s.isElePresent("id=d1"));

Test 2
(01) s.open(reportURL);
(02) s.mouseOver("id=item1");
(03) Thread.sleep(1000);
(04) s.mouseOut("id=item1");
(05) Thread.sleep(1000);
(06) assertFalse(s.isElePresent("id=d1"));
```

The test suite contains 2 tests, one for each scenario described in the WebRE+ model of figure 6. The first test opens the report (line 1), passes the mouse over the item (2) and waits till the item detail is shown (3), then the assertion verifies that the detail is present (4). The second test opens the report (1), passes the mouse over the item (2) and waits till the item detail is shown (3). Then the mouse is put out off the item and waits (4 and 5) and the assertion verifies that the detail is not present (6).

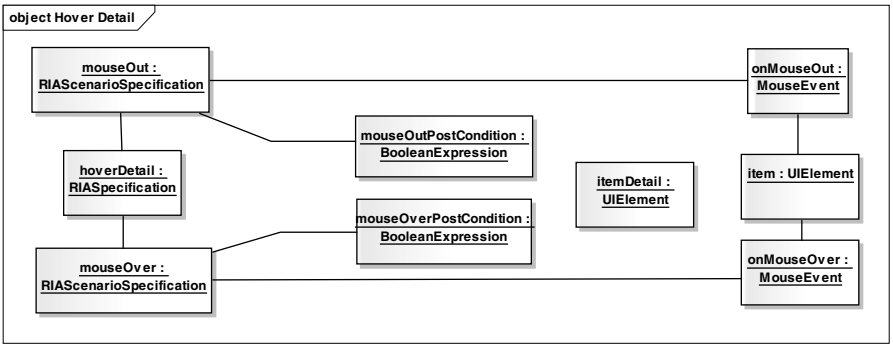

Fig. 8. Hover detail's specification in WebRE+

6 Implementation

The idea of using metamodels and UML profiles, allowed the incorporation of our approach in NDT and its tools easily. The metamodel for requirements of NDT is based on WebRE and WebRE+ is an extension of WebRE. The incorporation of the new classes presented in Figure 3 is easy and, proposing a way to represented them using a UML profile, as described in Figure 4, let us introduce it in the NDT-Profile. According to Figure 2, this extension enriches the possibilities of the Requirements Definition tasks. Thus, using WebRE+, a development team could include in the requirements a catalogue of RIA requirements.

But, WebRE+ could be also added in the NDT-Profile. According to section 2.1, NDT offers a suite, NDT-Suite, which is based on a tool, NDT-Profile, which supports its UML profiles and let the definition of NDT artifacts and elements using the UML notation. With the definition of a UML profile for WebRE+, we can enrich the NDT-Profile and Enterprise Architect to support it.

The next step to let the inclusion of our approach in NDT is the definition of a set of transformations to be included in NDT-Driver. This paper covers a first set of transformations, the set of transformations from requirements to tests. The rest of transformations are part of our future work.

In Figure 7, we show an example work screen of WebRE+ in Enterprise Architect. On the left, we can see a special toolbox for creating instances of the metaclasses. The user can select each WebRE+ artefact to deal with it in his diagrams. In Figure 7, a WebUser instance (WebUserExample) and a RIASpecification instance (RIASpecification) example is presented. In our profile (Figure 4), the RIASpecification is defined as an extension of the UseCase metaclass, thus, it could be related with a User, like WebUserExample.

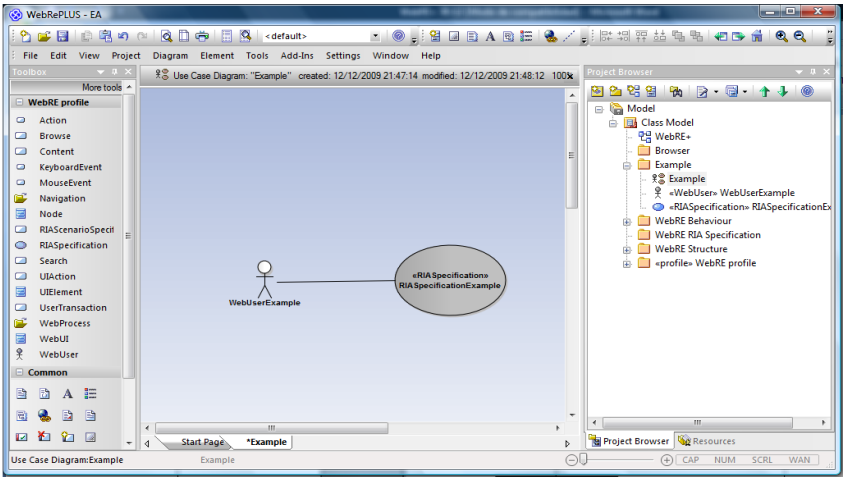

Fig. 9. The WebRE+ profile in Enterprise Architect

7 Related Works

The research of this paper is related with research in two different areas: the specification of Web Requirements in the context of Model Driven Engineering (MDE) and RIA. We analyze both areas in separate sub-sections.

7.1 Web Requirements Engineering and MDE

Web engineering is nowadays an important field in software engineering [9]. However there is an important gap in the treatment of requirements. In the first design approaches, OOHDM (Object-Oriented Hypermedia Design Model) [10], WebML

(Web Modeling Language) [11] or UWE (UML Web Engineering) [12] the main focus was put on modelling and design issues, while the requirements phase was almost neglected as reported in [13].

The importance of a full-fledge requirements phase is nowadays common in Web methodologies like NDT (Navigational Development Techniques) [14] or OOWS [15]. Additionally, some of the previous approaches started to include their own formalisms for requirement specification. For instance, OOHDM was enriched with UIDs (User Interface Diagrams) [16] or WebML with business models [17].

Other relevant problem in the requirements specification stage is the lack of standards and therefore the proliferation of proprietary notations; each approach tends to offer its own notation. To make matters worse similar formalisms are used in different approaches with slightly different semantics, or several names are used for the same concept.

In order to solve this problem, some authors have used the concepts in MDE [18]. In this development approach, building models is the main activity, and software is built by a series of model transformations ending, eventually, in a running application. Models are built using instances of concepts and relationships which are formally described by metamodels.

In summary, MDE offers a suitable solution for Web requirements for several reasons:

It mainly focuses on concepts; the way to represent them is a secondary aspect. It offers a systematic way to translate requirements knowledge into the next phases in the development life cycle.

Additionally, as some relations are defined between requirements and analysis concepts, it can control the traceability and the coverage of requirements.

Finally, if an UML profile is defined for the requirements metamodel (as it is in WebRE), software support tools for modelling activities can be built in a cheap way.

7.2 Rich Internet Applications

The Web as it was originally conceived has dramatically changed since 2003 when the concept of Rich Internet Applications (RIA) appeared. This new kind of Web applications mixes the old navigation style of Web Applications with the behaviour of traditional desktop applications: client side feedback, drag and drop features, etc. Since then, almost any desktop application has a Web counterpart, allowing users to take advantage of automatic updates since no instalment is necessary at the client side. Some examples of Web applications with RIA behaviour are Google Maps [20], GMail and Google calendar [21], Meebo [22], etc.

As developers faced the same problems repeatedly and found good solutions using the concepts in RIA, some patterns arose. As in the design patterns field, different catalogues showing RIA solutions to abstract problems have been described; one of the most popular catalogues is the so called Yahoo Patterns catalogue [2]. In contrast with software design patterns, RIA patterns are near to the stakeholder's perspective thus they use patterns' names when they describe specific RIA requirements. ADV-charts [5] were proposed as a modelling approach to design the structural and behavioural of user interface (UI) elements of RIA applications. However their level of abstraction (close to implementation) is inadequate to be used during requirements specification.

8 Conclusions and Future Works

In this paper we presented a metamodel for capturing RIA requirements. The metamodel allows us to express different well known RIA patterns such as those in the Yahoo patterns catalogue. The metamodel has been implemented as a UML profile and used within the EA environment to capture different RIA requirements in the context of a business intelligence application. Also, we have shown how to use the metamodel with mockups to improve the requirements gathering phase as it is hard for customers to understand the models and mockups give an intuitive way for expressing rich requirements.

Some aspects of our research still need some further work. In this matter we are working on deriving part of the RIA functionality using well known Javascript libraries such us YUI [28] or ExtJS [29]. Finally, since this kind of requirements not only affect the UI but also the software backend, we are trying to indicate which part of the functionality could not be implemented automatically and thus needs manual intervention from developers.

Besides, this approach opens new research lines for NDT. In the paper, we discussed how it enriches the requirements definition and presented a set of transformations (from RIA requirements to test); some others, such as for instance transformations from RIA requirements to analysis models, should be proposed. We are also working in these issues to complete our set of development tools.

Acknowledgements. This research has been supported by the project QSimTest (TIN2007-67843-C06_03) and by the Tempros project (TIN2010-20057-C03-02) of the Ministry of Education and Science, Spain.

References

1. Duhl, J.: Rich Internet Applications. A white paper sponsored by Macromedia and Intel, IDC Report (2003)
2. Yahoo Patterns, `http://developer.yahoo.com/ypatterns/` (last visit: 04/11)
3. Meliá, S., Gómez, J., Pérez, S., Díaz, O.: A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. In: Proceedings of the 2008 Eighth International Conference on Web Engineering, July 14 - 18, pp. 13–23. IEEE Computer Society, Washington, DC (2008)
4. Preciado, J.C., Linaje, M., Comai, S., Sanchez-Figueroa, F.: Designing Rich Internet Applications with Web Engineering Methodologies. In: Proceedings of the 2007 9th IEEE International Workshop on Web Site Evolution, WSE, October 05 - 06, pp. 23–30. IEEE Computer Society, Washington, DC (2007)
5. Urbieta, M., Rossi, G., Ginzburg, J., Schwabe, D.: Designing the Interface of Rich Internet Applications. In: LA-WEB 2007, pp. 144–153 (2007)
6. Moreno, M., Romero, J.R., Vallecillo, A.: An overview of Model-Driven web Engineering and the MDA. In: Web Engineering and web Applications Design Methods. Human-Computer Interaction Series, vol. 12, ch.12, pp. 353–382. Springer, Heidelberg (2007)
7. Escalona, M.J., Koch, N.: Metamodelling the requirements of Web Systems. In: Int. Conferences on Web Information Systems and Technologies, WEBIST 2005 and WEBIST 2006. LNBIP, vol. 1, pp. 267–280. Springer, Heidelberg (2007)

8. Enterprise Architect, http://www.sparxsystems.com.au (last visit: 04/11)
9. Deshpande, Y., Marugesa, S., Ginige, A., Hanse, S., Schawabe, D., Gaedke, M., White, B.: Web Engineering. Journal of Web Engineering 1(1), 3–17 (2002)
10. Rossi, G., Schwabe, D.: Modeling and implementing Web Applications with OOHDM. In: Web Engineering: Modelling and Implementing Web Applications. Springer, Heidelberg (2008)
11. Ceri, S., Fraternali, P., Bongio, A.: Web Modelling Language (WebML): A Modelling Language for Designing web Sites. In: Conference WWW9/Computer Networks, vol. 33(1-6), pp. 137–157 (2000)
12. Koch, N., Knapp, A., Zhang, G.: UML-Based Web Engineering. In: Web Engineering: Modelling and Implementing Web Applications, pp. 157–191. Springer, Heidelberg (2008)
13. Escalona, M.J., Torres, J., Mejías, M., Gutierrez, J.J., Villadiego, D.: The treatment of navigation in web Engineering. Advances in Engineering Software 38, 267–282 (2007)
14. Escalona, M.J., Aragon, G.: NDT. A Model-Driven approach for web requirements. A Model-Driven approach for web requirements. IEEE Transaction on Software Engineering 34(3), 370–390 (2008)
15. Fons, J., Pelechano, V., Albert, M., Pastor, ó.: Development of Web Applications from Web Enhanced Conceptual Schemas. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 232–245. Springer, Heidelberg (2003)
16. Vilain, P., Schwabe, D., Sieckenius de Souza, C.: A Diagrammatic Tool for Representing User Interaction in UML. In: Evans, A., Caskurlu, B., Selic, B. (eds.) UML 2000. LNCS, vol. 1939, pp. 133–147. Springer, Heidelberg (2000)
17. Brambilla, M., Fraternali, P., Tisi, M.: A Transformation Framework to Bridge Domain Specific Languages to MDA. In: Chaudron, M.R.V. (ed.) MODELS 2008. LNCS, vol. 5421, pp. 167–180. Springer, Heidelberg (2009)
18. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Software 20(5), 36–41 (2003)
19. Koch, N., Zhang, G., Escalona, M.J.: Model Transformations from Requirements to Web System Design. In: ACM (ed.) ACM International Conference Proceeding Series. Proceedings of the 6th International Conference on Web Engineering (ICWE 2006), Palo Alto, California, USA, pp. 281–288 (2006) ISBN: 1-59593-352-2
20. Google Maps, http://maps.google.com (last visit: 04/11)
21. Gmail, http://www.gmail.com (last visit: 04/11)
22. Meebo, http://www.meebo.com (last visit: 04/11)
23. MagicUWE, http://uwe.pst.ifi.lmu.de/toolMagicUWE.html (last visit: 04/11)
24. NDT-Profile, http://www.iwt2.org/ndt (last visit: 04/11)
25. Selenium, http://seleniumhq.org (last visit: 04/11)
26. Watir, http://watir.com/ (last visit: 04/11)
27. Pentaho, http://www.pentaho.com (last visit: 04/11)
28. YUI, http://developer.yahoo.com/yui/ (last visit: 04/11)
29. ExtJS, http://www.sencha.com/products/extjs/ (last visit: 04/11)
30. Ruby, http://www.ruby-lang.org (last visit: 04/11)
31. Martin, R.C.: Agile Software Development: Principles, Patterns, and Practices. Prentice Hall PTR, Upper Saddle River (2003)