Computer and Information Science; Vol. 9, No. 2; 2016
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

A Model Driven Approach for Generating Graphical User Interface for MVC Rich Internet Application

Sarra Roubi[1], Mohammed Erramdani[1] & Samir Mbarki.[2]

[1] High School of Technology, Mohammed First University, Oujda, Morocco

[2] Computer Science Department, Ibn Tofail University, Kenitra, Morocco

Correspondence: Sarra Roubi, High School of Technology, Mohammed First University, BP 473 Complexe universitaire Al Qods, Oujda, Morocco. Tel: 212-666-299-336. E-mail: s.roubi@ump.ac.ma

Received: February 11, 2016 Accepted: March 16, 2016 Online Published: April 19, 2016

doi:10.5539/cis.v9n2p91 URL: http://dx.doi.org/10.5539/cis.v9n2p91

Abstract

Web applications have witnessed a significant improvement that exhibit advanced user interface behaviors and functionalities. Along with this evolution, Rich Internet Applications (RIAs) were proposed as a response to these necessities and have combined the richness and interactivity of desktop interfaces into the web distribution model. However, RIAs are complex applications and their development requires designing and implementation which are time-consuming and the available tools are specialized in manual design. In this paper, we present a new model driven approach in which we used well known Model-Driven Engineering (MDE) frameworks and technologies, such as Eclipse Modeling Framework (EMF), Graphical Modeling Framework (GMF), Query View Transformation (QVTo) and Acceleo to enable the design and the code automatic generation of the RIA. The method focus on simplifying the task for the designer and not necessary be aware of the implementation specification.

Keywords: model, Meta model, transformation, Graphical User Interface, Rich Internet Application, code generating

1. Introduction

Web applications concentrated all their activity around a client-server architecture where the processing is done on the server side and the client side is only used to display static content. These HTML-based Web applications are showing their limitations, especially when it comes to integrate complex activities to be performed via Graphical User Interfaces (GUI). To overcome these limitations, development in information technology has known a large evolution and several new technologies have been introduced. Among, desktop-like Web applications, called Rich Internet Applications (RIAs). We focus our work on this kind of application because it combines the benefits of the Web distribution model with the interface interactivity of desktop applications. Moreover, RIAs provide a new client-server architecture that reduces significantly network traffic using more intelligent asynchronous requests that send only small blocks of data.

In return, the design and implantation of graphical user interface for RIAs is known for its complexity and difficulty in using existing tools, on the one hand. In addition, designers need to know the computer platforms, users' characteristics, environmental interaction, etc ..., which can become tedious, time-consuming and requires additional efforts and can negatively affects the quality of the application. On the other, modeling approach is an efficient way to master complexity and ensure consistency of applications. That is to say that a model-driven approach adopted in the process of development of RIAs can significantly reduce the risk of rework and improve the overall quality of the development process for RIAs.

In this context, in this paper we present a new approach based on Model Driven Engineering paradigm that proposes a complete development process based on a set of models and transformations upon the Eclipse Modeling Project that allows obtaining the implementation of Rich Internet Applications with JavaFX platform as a target adopting a Model-View-Controller (MVC) architectural design pattern, focusing on the graphical part of the application. The proposed approach can be replicated for different design model and a different target technology platform.

The paper is organized as follows. Section II presents related work dealing with Model-Driven development

approaches. In Sections III we present the Model Driven Engineering Approach. Section IV describes the proposed approach and technologies used to develop it, then we report the results of the case study of designing and generating the RIA to validate the approach. Finally we conclude.

2. Related Work

Several works dealing with GUI's automatic generation have emerged recently. (Kapitsaki et al. 2009) present a model based approach and advocates in favor of a complete separation of the web application functionality from the context adaptation at all development phases. Also, a Rich Internet Application for web based product development was presented (Ahmed & Popov 2010). Besides, (Meliá et al. 2008) proposes an approach called OOH4RIA which is a model driven development process that extends OOH methodology to generate GWT Rich Internet Application. Also, a combination of the UML based Web Engineering (UWE) method for data and business logic modeling with the RUX-Method for the user interface modeling of RIAs was proposed as model-driven approach to RIA development (Linaje et al, 2007). In addition, a research based on the plasticity of User Interface with the use of Model Driven Architecture concepts for the purpose of unifying the modeling for GUI is presented (Sottet et al, 2007). Some researches apply model based approaches for multi-device user interface development. Among them we can cite: TERESA (Transformation Environment for inteRactivE Systems representations) (Berti et al, 2003) and MARIA with (Paterno et al, 2009). Also, UsiXML (USer Interface eXtended Markup Language) (Vanderdonckt, 2005).

Another related work on applying MDA approach for RIAs is found in (Martinez-Ruiz et al, 2006). The approach is based on XML User Interface description languages using XSLT as the transformation language between the different levels of abstraction. Moreover, an MDA approach for AJAX web applications (Gharavi et al, 2008) was the subject of a study that proposes an UML scheme using profiling for modeling AJAX user interfaces; it adopts ANDROMDA tool for creating an AJAX cartridge to generate the corresponding AJAX application code with back-end integration.

These works focused on using a UML profiling to define new ways to model the RIAs application. In our case, we wanted to keep the task for designer as simple as possible and translate the user needs in terms of operations and goals to achieve. So, we proposed new meta models to simplify the task for the designer and be able to generate efficiently Rich Internet Application.

3. Model Driven Engineering

3.1 The OMG Approach and Acronyms

The OMG (The Object Management Group) initiated the Model Driven Architecture, an approach (Miller et al, 2003), to develop systems that offer more flexibility in the evolution of the system while remaining true to customer needs and satisfaction. It is based on models throughout the whole development process.

Indeed, the approach relies on the separation into three layers of abstraction and the code: Computing Independent Model (CIM), Platform Independent Model (PIM) and the Platform Specification Model (PSM). We can define the three layers as follow in figure.1:

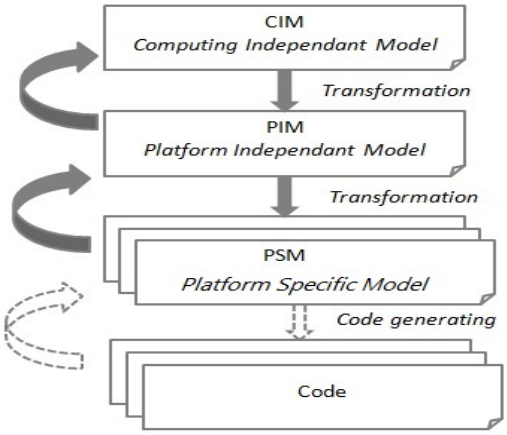

Figure 1. Levels of abstraction in Model Driven Architecture

- **CIM**: It represents a high level specification of the system's functionalities and describes what the system offers but hides all the technology specifications.
- **PIM**: It allows the extraction of the common concept of the application independently from the platform target so it enables its mapping to one or more platforms.
- **PSM**: It combines the specifications in the PIM with the details of a chosen platform. It allows having models conform to a specific platform and helps generating the appropriate code source.

3.2 Transformation Process in MDE

A transformation converts models from one level of abstraction to another, usually from a more abstract to less abstract view, by adding more detail and information supplied by the transformation rules to respect the target meta model.

There are two types of transformations in the MDA approach:

- **Model To Model**: it concerns transformation to pass from CIM to PIM or from PIM to PSM.
- **Model To Text**: it concerns the generation of the code from the entry model (the PSM) to a the programming language related to the chosen platform.

Model To Model transformation with the modeling approach allows having productive models' transformation, independently of any execution platform. The OMG has developed a standard for this transformation language which is the MOF 2.0 QVT (OMG, 2008), standing for Query View Transformation. The QVT specification has a hybrid declarative/imperative nature, with the declarative part being split into two-level architecture. For this work, we used the QVT-Operational mappings language implemented by Eclipse modeling.

For the code generation phase under the MOF Model To Text (MOFM2T) specification, there are a number of tools aimed at the automation of applications development. The principle is to parse the representation of the model in XML Metadata Interchange (XMI) form and apply templates that give the code source equivalent to each model element. Optionally, the developer will have to add or edit source code portions to complete its application code. Acceleo, among others, is an implementation of the "MOFM2T" standard, from the Object Management Group (OMG). It is used in our work for final transformation and code generation of the RIA with explicit Graphical User Interface.

4. The Model Driven Development Proposed Process

We propose a model-driven development process that adopts MDE technologies in order to enable the automatic generation of a MVC Rich Internet Application, focusing on the graphical part, starting from a simplified designed model. Indeed, we elaborated the tow meta models for the PIM and PSM as presented in the following section, then we established the transformation rules. The first transformation engine; Model To Model using QVT, generates the model respecting the MVC RIA application, while the second; Model To Text Acceleo transformations gives the source and configuration files for the JavaFX application using Acceleo.

4.1 The PIM and PSM Proposed Meta Models

While developing the meta model, that is the input to our modeling process, we focused on keeping the terms as simple as possible so the designer does not have to learn a new language or a complex design process. So, the meta model translates the user's vision of the graphical user interface in terms of actions and interactions, by describing what is expected from the application from a graphical point of view.

The major goal of the view is described through the use case that is divided into several main operations that gathers the atomic actions leading to achieve that goal. We added an enumeration of the basic actions' types that any user is familiar with (input, selection, clicks…). Finally, we assigned a property in each action (is it a password, a single choice …), to help choose the most appropriate widget while defining the transformation rules. All these elements are embraced in the UMLPackage.

Figure. 2 shows the proposed PIM meta model and the relationship between its elements described.

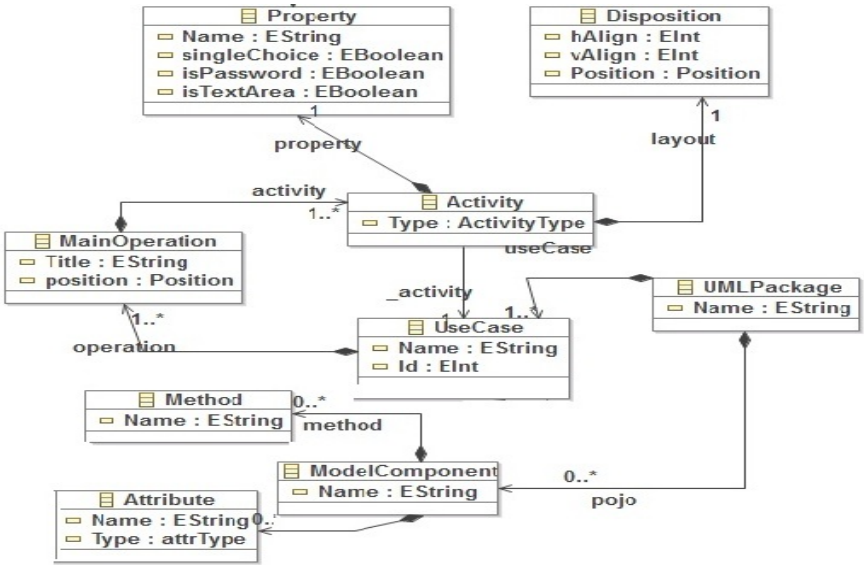

Figure 2. Platform Independent proposed Meta Model

The PSM meta model for the JavaFX was developed taking into account the relation between the different layers of the MVC architecture; which will avoid the GUI designer to worry about understanding the dynamics of appearance between the different layers of application. That is to say, as shown in figure. 3, we have the three packages: ViewPackage, ModelPackage and ControllerPackage that gather respectively the application's views, models and controllers.

Note that the View/Controller layers are responsible for describing the structure and content of views in terms of graphical elements while the navigation flow is ensure through the controller's handler that are connected to the specified services from the model layer.

In our proposed PSM meta model, we detailed the presentation layer where the composite design pattern was used to define the overlapping of graphical components in a RIA view; called the scene. The scene is composed of graphical components, named controls that could be containers as Roots.

In addition to that, we added a hierarchical relationship between graphical components base on their type, and we associate each component with its position in the grid that will divide the whole scene so we can put each component in a specific position as defined in the input model.

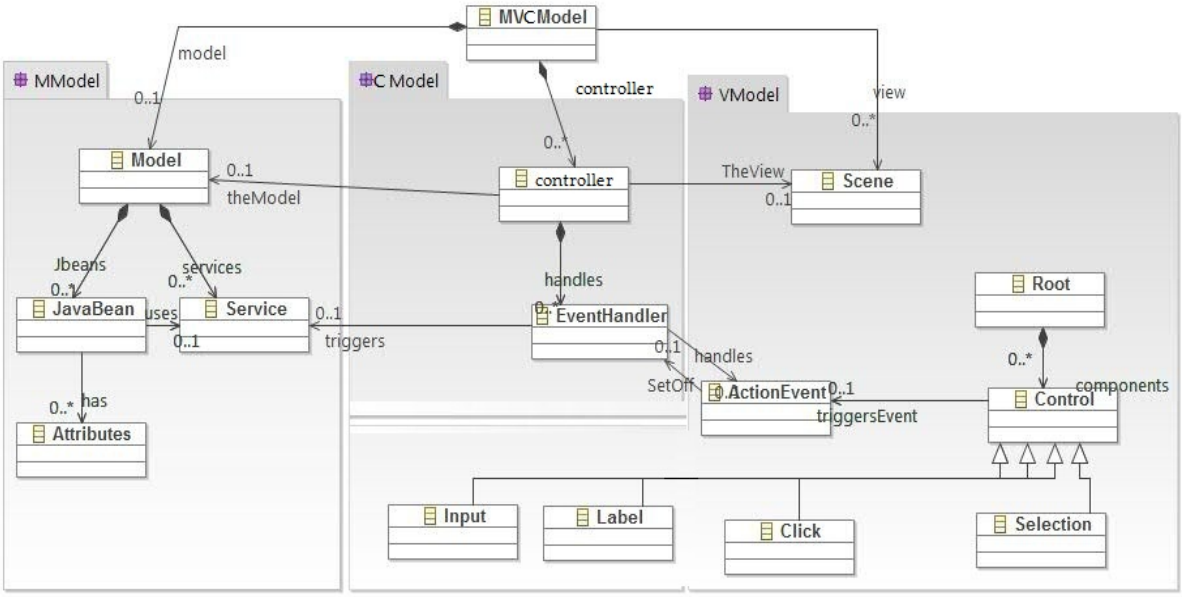

Figure 3. MVC for JavaFX RIA proposed Meta Model

4.2 The Transformation Process

Once the meta modeling phase established, comes the most important phase in the MDE approach, which is the transformation process. Figure 4 summarizes the process with adopted in our approach:

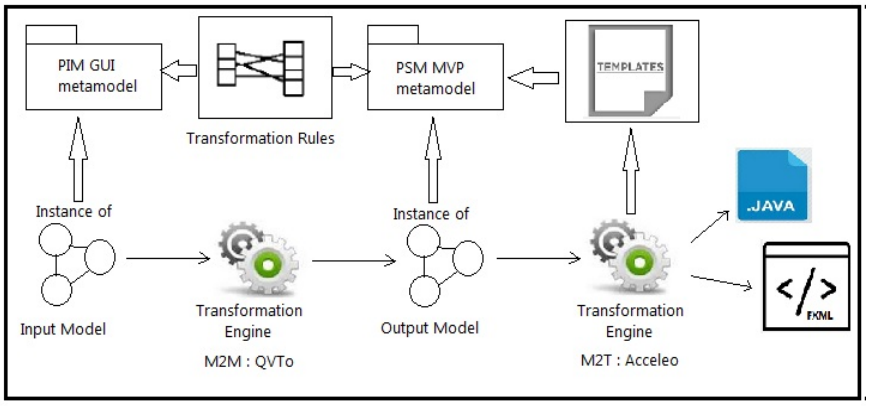

Figure 4. Transformation process in the proposed approach

Indeed, we defined the transformation rules that we developed into a transformation engine using the standard QVTo. This engine takes as input a model that is an instance of the PIM defined previously, and gives a MVC JavaFX model that could be easily used to generate the whole code of the application.

The main method is the entry point of the transformation and takes the input and output model references as arguments. It makes the correspondence between all elements under root element UmlPackage of the input model and the elements of target root element JavaFXPackage, figure 5.

```
modeltype GUI uses "http://guimm/1.0";
modeltype JAVAFX uses "http://javafxmm/1.0";

transformation umlToJavaFX(in src:GUI, out dest:JAVAFX);
main() {
        src.objectsOfType(UMLPackage)->map umlPackToMVCPack();
}
mapping UMLPackage::umlPackToMVCPack () : JavaFXPackage {
        result.Name := 'MVC ' + self.Name;
        result.VPackage := object ViewPackage {
                Name := self.Name + 'View';
                views += self.useCase->map useCaseToScene();
        };
        result.MPackage := object ModelPackage {
                Name := self.Name + 'Model';
                model += self.useCase.map useCaseToModel();
        };
        result.CPackage := object ControllerPackage {
                Name := self.Name + 'Controller';
                controller += self.useCase->map useCaseToController();
        };
}
mapping UseCase::useCaseToScene() : Scene {
    result.Name := self.Name + 'View';
    result.events+=self.operation.activity
    [not(Type=ActivityType::label)].map activityToEvents();
    ….
```

Figure 5. QVT portion of the transformation engine to JavaFX MVC

When the transformation rules defined and the engine is sufficiently working, the second step consists on generating the code from the obtained PSM model instance. The template of the target JavaFX for MVC is developed using the OMG standard Acceleo that automatically transform models obtained in the first transformation phase into Code for JavaFX. An excerpt of the template is presented in figure 6:

```
[template public createViewFiles(theRoot : Root)]

    <[theRoot.position.toString().toLowerCase()/]>
    <GridPane>
        [for (widget : Control | theRoot.hasWidgets)]
            <[if (widget.oclIsTypeOf(label))]Label text="[widget.oclAsType(label).Text/]"
            [elseif(widget.oclIsTypeOf(TextField))]TextField onAction = "#[widget.triggersEvent.Name.escapeSpecialChar()/]"
            [elseif(widget.oclIsTypeOf(PasswordField))]PassordField onAction = "#[widget.triggersEvent.Name.escapeSpecialChar()/]"
            [elseif(widget.oclIsTypeOf(Button))]Button text="[widget.oclAsType(Button).Text/]"
                          onAction = "#[widget.triggersEvent.Name.escapeSpecialChar()/]"
            [elseif(widget.oclIsTypeOf(Link))]HyperLink text="[widget.oclAsType(Link).Text/]"
                          onAction = "#[widget.triggersEvent.Name.escapeSpecialChar()/]"
            [elseif(widget.oclIsTypeOf(TextArea))]TextArea onAction = "#[widget.triggersEvent.Name.escapeSpecialChar()/]"
            [elseif(widget.oclIsTypeOf(List))]ListView
            [elseif(widget.oclIsTypeOf(ComboBox))]ComboBox
            [elseif(widget.oclIsTypeOf(CheckBox))]CheckBox
        [/for]
             [/if]

    </GridPane>
    </[theRoot.position.toString().toLowerCase()/]>
```

Figure 6. Acceleo templates for the transformation engine to JavaFX Code

The execution of these templates gives the source code of the application with Java and XML files for the views, the presenters and the models. Those generated files are loaded in an IDE as a JavaFX project respecting the MVC pattern. It gathers the three layers and specifically the graphical interface with all the components as desired with all the connections with the application's layers.

5.3 Results

We applied this approach to a running example of an online product search application. We defined the input model, respecting the proposed PIM meta model and sufficiently describing the view elements that should compose the generated view.

Figure 7. represents the result after applying the Model To Model transformation through the engine. The first element in the generated PSM model is the JavaFXPackage that includes the view Package, the model Package and the controller package standing for the MVC pattern. The controller package contains the controller for each scene to handle the user events. It is also connected to the model to call the appropriate method for the event raised.

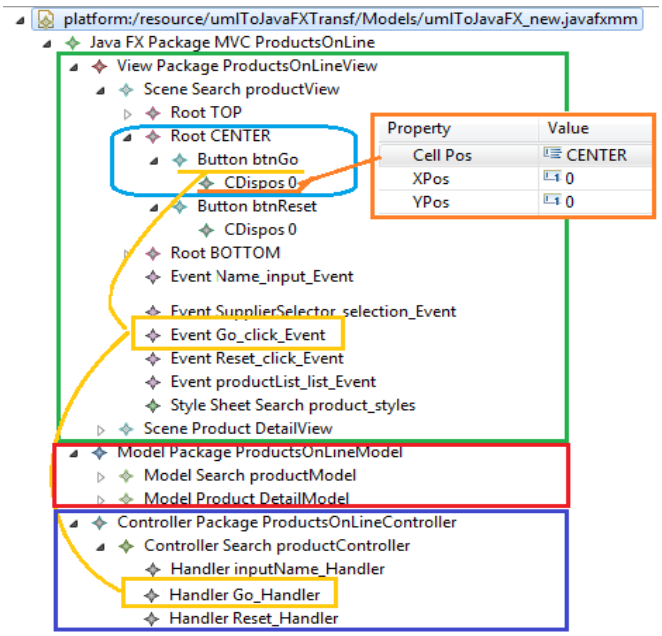

Figure 7. Generated model for MVC Rich Internet Application

The generated file has the entire description to generate the code application. Indeed, each component is typed and has its position in the scene. This file gathers all the information required to easily generate java code sources, FXML files for the views, packages and relationships between all of the layers of the MVC pattern.

After the design and generation of the application, the set of artifacts resulting from the whole process constitute a RIA ready to be deployed and can be easily loaded into an IDE. Figure 8 shows the screenshot of the generated view the application: the search product view with the form for the criteria of the research and a list that will give all the available products.

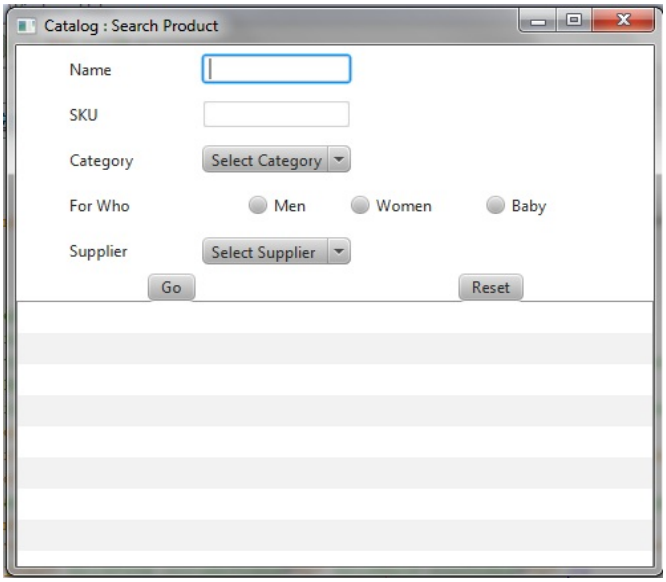

Figure 8. The research product view automatically generated

5. Conclusion

In this paper, we applied the MDE approach to generate a Rich Internet Application. The process involves first defining tow meta models as PIM and PSM respecting the MVC pattern for RIA. Second, we defined the transformation rules for both Model To Model and Model To Text transformation. A case study conducted on designing and generating a RIA for a search movie application has shown that the approach is valid and the supporting tools work properly. In the future, we aim at extending this work to allow the generation of other complex graphical components of RIAs and handling the layout management also. Furthermore, we can consider integrating other frameworks like Flex and GWT and combine the approach with existing ones.

References

Ahmed, Z., & Popov, V. (2010). Integration of Flexible Web Based GUI in I-SOAS. Retrieved from http://arxiv.org/abs/1011.3257

Berti, S., Correani, F., Mori, G., Paterno, F., & Santoro, C. (2004). Teresa: A transformation-based environment for designing and developing multidevice interfaces. *In CHI Extended Abstracts*, 793–794. http://dx.doi.org/10.1145/985921.985939

Gharavi, V., Mesbah, A., & Deursen, A. V. (2008). Modelling and Generating AJAX Applications: A Model-Driven Approach. *Proceeding of the7th International Workshop on Web- Oriented Software Technologies, New York, USA* (p. 38, Year of publication, ISBN: 978-80-227-2899-7).

Kapitsaki, G. M. et al. (2009). Model-driven development of composite context-aware web applications. *Information and Software Technology, 51*(8), 1244–1260. http://dx.doi.org/10.1016/j.infsof.2009.03.002

Linaje, M., Preciado, J. C., & Sanchez-Figueroa, F. (2007). A Method for Model Based Design of Rich Internet Application Interactive User Interfaces. *In Proceedings of International Conference on Web Engineering*, July 16-20, 2007, Como, Italy, 226–241. http://dx.doi.org/10.1007/978-3-540-73597-7_18

Martinez-Ruiz, F. J., Arteaga, J. M., Vanderdonckt, J., & Gonzalez-Calleros, J. M. (2006). A first draft of a model-driven method for designing graphical user interfaces of Rich Internet Applications. *In LA-Web 06: Proceedings of the 4th Latin American Web Congress*, p. 3238. IEEE Computer Societ.

http://dx.doi.org/10.1109/la-web.2006.1

Meliá, S. et al. (2008). A model-driven development for GWT-based rich internet applications with OOH4RIA. *Proceedings - 8th International Conference on Web Engineering, ICWE 2008,* 13–23. http://dx.doi.org/10.1109/ICWE.2008.36

Miller, J., & Mukerji, J. (2003). MDA Guide Version 1.0.1. OMG.

OMG, Q. (2008). Meta Object Facility (MOF) 2 . 0 Query / View / Transformation Specification. Transformation, (January), pp.1–230. Available at: http://www.omg.org/spec/QVT/1.0/PDF/.

Paterno, F., Santoro, C., & Spano, L. D. (2009). Maria: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact., 16*(4). http://dx.doi.org/10.1145/1614390.1614394

Sottet, J. S., Ganneau, V., Calvary, G., Coutaz, J., Demeure, A., Favre, J. M., & Demumieux, R. (2007). Model-driven Adaptation for Plastic User Interfaces. *Proceedings of the 11th IFIP TC 13 International Conference on Human-computer Interaction*, pp.397–410. http://dx.doi.org/10.1007/978-3-540-74796-3_38

Vanderdonckt, J. (2005). A MDA-compliant environment for developing user interfaces of information systems. *In CAiSE*, 16–31. http://dx.doi.org/10.1007/11431855_2