World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:2, 2016

A Model-Driven Approach of User Interface for MVP Rich Internet Application

Sarra Roubi, Mohammed Erramdani, Samir Mbarki

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

Abstract—This paper presents an approach for the model-driven generating of Rich Internet Application (RIA) focusing on the graphical aspect. 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. During the development of the approach, we focused on the graphical aspect of the application in terms of interfaces while opting for the Model View Presenter pattern that is designed for graphics interfaces. The paper describes the process followed to define the approach, the supporting tool and presents the results from a case study.

Keywords—Code generation, Design Pattern, metamodel, Model Driven Engineering, MVP,Rich Internet Application, transformation, User Interface.

I. Introduction

WEB applications concentrated around a client-server architecture where the processing is done on the server side, while 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). 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, design and implantation of graphical user interface for RIAs is known for its complexity and difficulty in using existing tools which can become tedious, time-consuming and requires additional efforts which effect negatively the quality of the resulting RIA. That is why such a model-driven approach adopted in the process of development of RIAs [8] can significantly reduce the risk of rework, improve the quality of the application in general and the UI in particular. In this context, the paper presents an approach based on the MDE paradigm [1] that proposes a complete development process based on a set of models and transformations upon the Eclipse Modeling Project [13] that allows obtaining the implementation of RIAs with JavaFX platform as a target adopting a Model-View-Presenter (MVP) 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 discuss related work dealing with Model-Driven development approaches. In Sections III and IV, we present respectively the Model Driven Engineering Approach and the Model View Presenter Pattern. Section V describes the proposed approach and technologies used to develop it, while in Section VI we report the result of the case study of designing and generating the RIA to validate the approach.

S. Roubi and M. Erramdani are with the MATSI Laboratory, Oujda Institute of Technology, Morocco, 60000 (e-mail: roubi.sarra@gmail.com).

S. Mbarki is with the Computer Science Department of Ibn Tofail University, Kenitra, Morocco (e-mail: samirmbarki@hotmail.com).

II. Related Work

In the Web Engineering context, MDE principles are being used to successfully address the development and evolution of web applications. In [7], it is shown how MDD/MDA principles are applied in the Web Domain to define models, metamodels and transformations, to manage interchangeability issues and to build tools that support the development process. Also, a Rich Internet Application for web based product development was presented in [8]. Besides, [9] proposes approach called OOH4RIA which proposes a model driven development process that extends OOH methodology employing the OOHDM conceptual and navigational scheme of a web application as the basic PIM for the MDA process, using any UML-based design tool which produces an XMI-file as output. Besides, 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 [10]. These methods use models to separate the platform independent model (PIM) design of web systems from the platform-dependent (PSM) implementations as much as possible.

UWE [11] follows the MDA principles and uses the OMG standards [1]. The process makes use of model transformations defined at metamodel level and specified by general purpose transformation languages, such as QVT [2] and graph transformations. Furthermore, an MDA approach for AJAX web applications [12] was the subject of a study that proposes a UML scheme by using the profiling for modeling AJAX user interfaces, and report on the intended approach of adopting AndroMDA for creating an AJAX cartridge to generate the corresponding AJAX application code, in ICEFACES, with back-end integration. A meta model of AJAX has been defined using the AndroMDA tool.

The main difference between the proposed approach and the considered related ones is the focus that we put on the graphical aspect of the application on the one hand, and the complete abstraction of the input model from any technical knowledge of the targeted platform on the other hand. This guarantees the translation of the user's expectation from a simple model to an automatically generated RIA that respects

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

a MVP pattern and provides well designed graphical user interfaces.

III. The OMG Approach and Acronyms

In November 2000, the OMG (The Object Management Group), a consortium of over 1,000 companies, initiated the MDA (Model Driven Architecture) approach [6]. The OMG presents this approach as a way to develop systems that offer greater flexibility in the evolution of the system while remaining true to customer needs and satisfaction. According to Model-Driven Engineering (MDE) principles [3] - [5], software development is focused on the modeling of the application behavior, structure and requirements using formal modeling languages on the one hand and the transformation engines on the other. The aim is to increase productivity, simplify the design process and promote the communication between the stakeholders.

In MDE, every artifact, including the source code, is considered as a model element, and the whole development process can be seen as a set of related transformations from one model to the next one in order to automate the system's implementation from its requirements. Which brings up the three different layers of abstraction that can be described:

Computing Independent Model (CIM): It represents a high level specification of the system's functionality. It shows exactly what the system is supposed to do, but hides all the technology specifications.

Platform Independent Model (PIM): It allows the extraction of the common concept of the application independently from the platform target.

Platform Specification Model (PSM): It combines the specifications in the PIM with the details required of the platform to stipulate how the system uses a particular type of platform which leads to include platform specific details.

Fig. 1 shows the three layers of the MDA approach and the and the transformation necessary for the transition from one to the other.

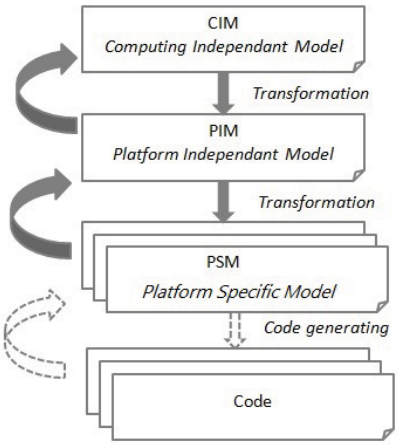


Fig. 1 The levels separation in the Model Driven Engineering approach

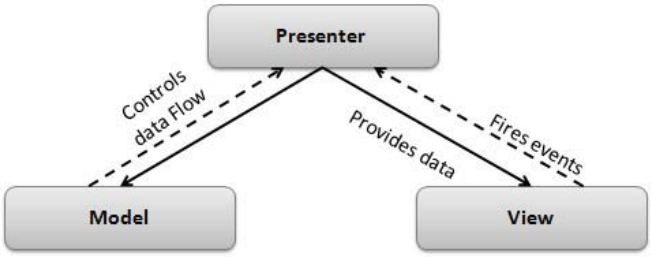


Fig. 2 The MVP design pattern

IV. The MVP Design Pattern

The Model View Presenter pattern is a derivation of the MVC pattern. Its architecture is used mostly for building User Interface since it provides a cleaner implementation of the Observer connection between Application Model and view. The MVP pattern is based on the principle of separation of concerns in presentation logic and carry the three layers (see Fig. 2):

- **The model:** It represents an interface defining the information and data that the user wishes to view or handle within the views of the application.
- **The view:** It is a passive interface that displays the necessary information of the use case to which it is related. It also responds to the user actions and routes the commands to the presenter.
- **The presenter:** It is a control component that acts upon the model and the View and orchestrate the operation of the web layer of the application.

Based on this pattern, many frameworks were designed to assist and help developers build their application focusing on the presentation layer. With JavaFX, we find the clear separation of roles between the view and the control with the FXML (EFF-ects eXtended Markup Language) that is used only for layout. We can say that the tow does compromise and give a better results for developing RIAs.

V. The Model Driven Proposed Process

The approach described in this paper adopts MDE technologies in order to enable the automatic generation of a MVP Rich Internet Application starting from a simplified designed model. When we adopt a model driven development approach, a variety of technologies and frameworks are available. These tools and technologies can be effectively used to achieve the final goal which is the automatic generation of code. In this section, we present the steps followed and the technologies and tools used to elaborate our approach.

A. Steps Followed and Adopted MDE Technologies to Develop the Approach

The process followed to define our modeling approach includes three main steps as described in Fig. 3 and listed below:

- *Defining PIM and PSM Metamodel.* First, for the input model, we focused on keeping the models as simple as

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

possible. We also wanted the input model to describe the user's expectation from the application to be in terms of activities and operations. That is why we started from a the basic notion of the use case that descries the main goal of the view to be developed. Second, for the PSM metamodel, we chose a reference architecture which is the Model View Presenter Design Pattern. To define the two meta models, we used the Eclipse Modeling Framework (EMF).

- *Developing a Graphical Modeling Tool:* A graphical editor was developed to enable the developer to create, view, and edit the models and instances of the defined PIM meta model. To this aim, we used the Eclipse Graphical Modeling Framework (GMF).
- *Developing transformations:* Model to Model transformation was defined to generate automatically the MVP RIA model from the simple model input. Then, Model To Text rules were defined to automatically generate the RIA code source using Acceleo Templates. See Figs. 7 and 8.

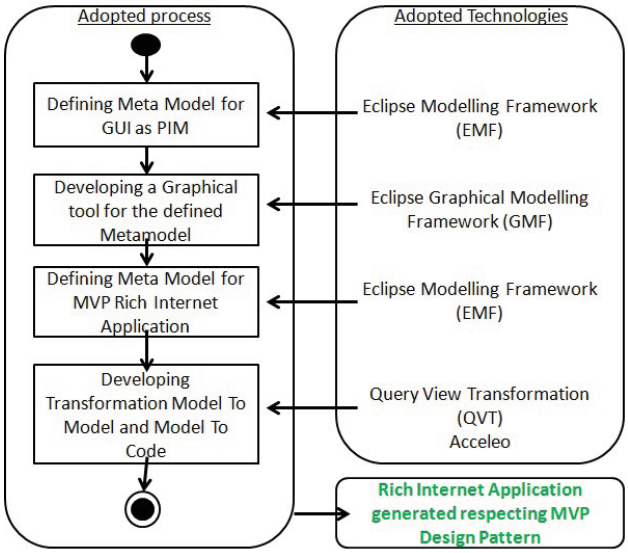


Fig. 3 Process and technologies adopted to define our model-driven approach

B. The Defined Design Meta Models and Tool

1) PIM and PSM Meta Models: First, we developed the PIM 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. We also wanted to keep the focus on the graphical aspects of the application. Indeed, the meta model translates the users vision of the application in terms of actions and interactions, by describing what is expected from the application from a graphical point of view. On the one hand, 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**. Finally, we added a Disposition for each action to be able to give each component the appropriate position in the view. Besides, the user can define the **model elements** that will define the business layer of the RIA that are made up of attributes and methods. Fig. 4 shows the proposed PIM meta model and the relationship between its elements.

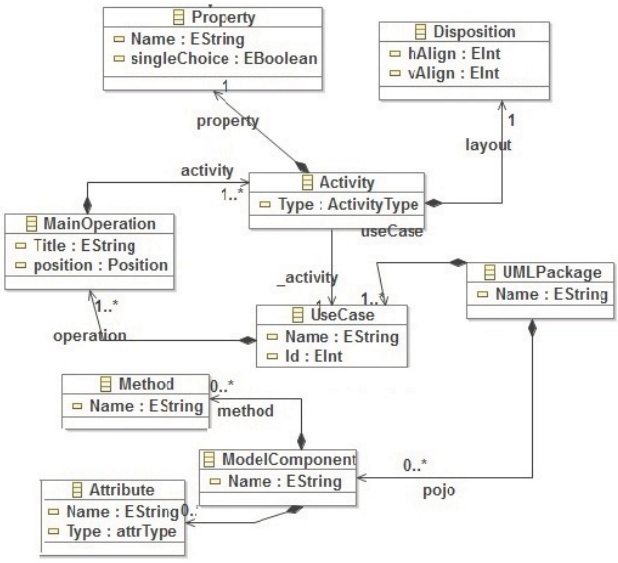


Fig. 4 Proposed GUI PIM meta model for RIA

Second, we defined the PSM metamodel for the RIA adopting the MVP as a core architectural pattern (e.g., the JavaFX implementation platform in the case of this article). As shown in Fig. 5, we have the three packages:

- **ViewPackage**, containing meta-classes to represent Views and the graphical components taking into account the hierarchy in it.
- **ModelPackage**, representing the domain or the business layer of the application and is made up of Methods and Beans.
- **PresenterPackage**, containing the presenters to ensure the connection between the tow layers of the MVP pattern.

Note that the View/Presenter layers are responsible for describing the structure and content of views in terms of behavioral elements while the navigation flow is ensure through the presenter's handler that are connected to the specified services from the model layer. In the view part, the scene is composed of graphical components, named controls that could be containers as Roots. We added a hierarchical relationship between graphical components base on their type. We also thought about the composite relationship that can connect containers with simple components. Besides, 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. All those elements give us a well defined RIA respecting the MVP

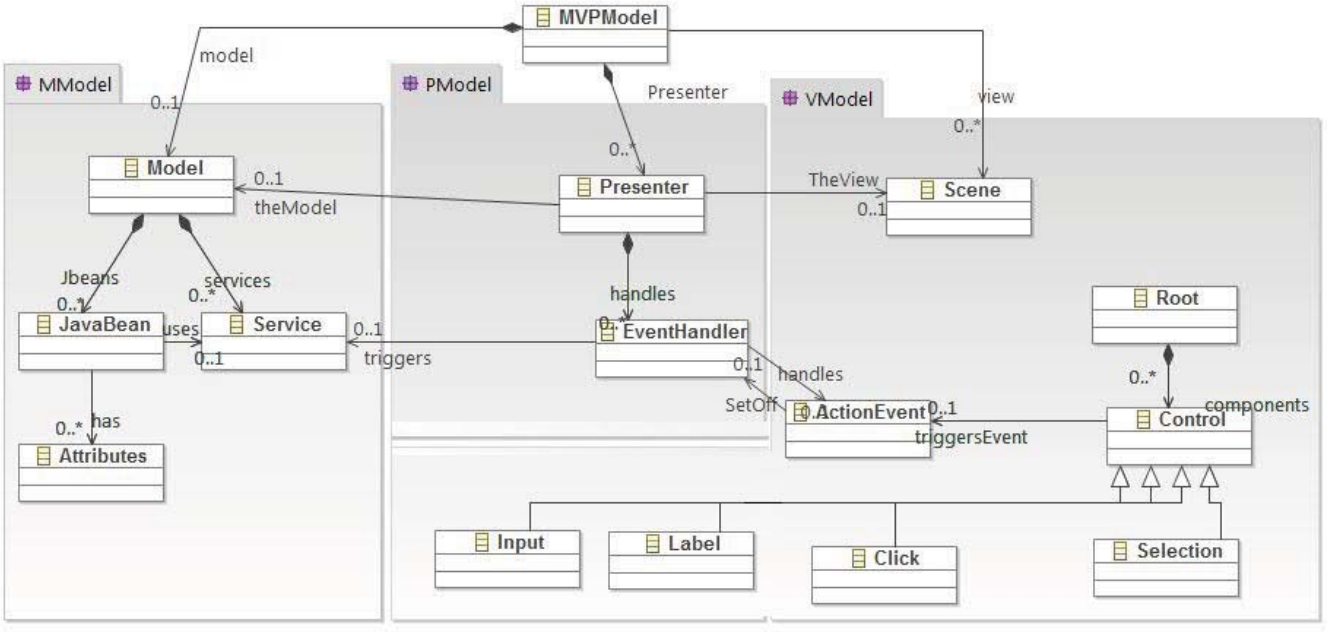


Fig. 5 A simplified version of the MVP design metamodel for RIA

pattern and taking into account the richness of the graphical aspects of these applications.

2) Transformation Process: Once the meta modeling phase established, we defined the transformation rules. Since we have defined the PIM and PSM metamodels, we need to define both the Model To Model transformation using the QVTo standard and the Model To Text transformation with Acceleo to generate the final code source for the RIA first modeled.

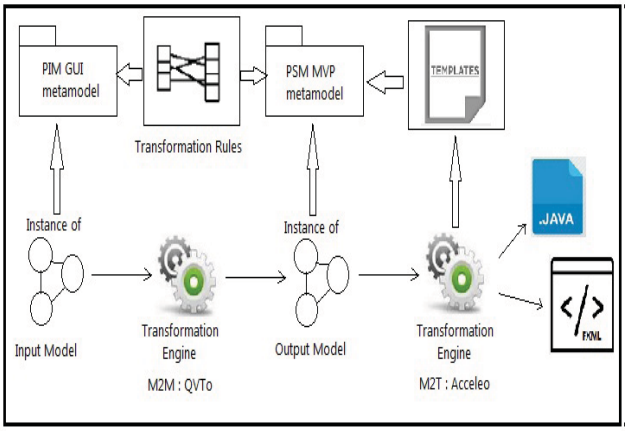


Fig. 6 Transformation process in the MDE approach

- **Model To Model:** The entry point of the transformation is the main method. This method makes the correspondence between all elements of type UmlPackage of the input model and the elements of type JavaFXPackage output model. For instance, for each use case from the input model we create a presenter, a model and a view. The view components are defined from the main operations and activities in each use case.

Finally we get as a result of this transformation a model corresponding to the JavaFX meta model defined.

```
modeltype GUIMVP uses "http://guimmMVP/1.0";
modeltype JAVAFXMVP uses "http://javafxmmMVP/1.0";

transformation guiToJavaFXmvp(in src:GUIMVP, out dest:JAVAFXMVP);

main() {
    src.objectsOfType(UMLPackage)->map umlPackToMVCPack();
}

mapping UMLPackage::umlPackToMVCPack() : JavaFXPackage {
    result.Name := 'MVP ' + 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.PPackage := object PresenterPackage {
        Name := self.Name + 'Presenter';
        presenter += self.useCase->map useCaseToPresenter();

    };
```

Fig. 7 QVTo portion of the transformation engine

- **Model To Text:** we defined templates, Fig. 8, with Acceleo to automatically transform models obtained in the first transformation phase. The execution of these templates we developed gives the source code of the application with Java files for the views, the presenters and the models. With these generated files we are able to create an MVP JavaFX project that give us the graphical interface with all the components as desired and also all the connections with the application's three layers.

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

```
[comment encoding = UTF-8 /]
[module generateJFX('http://javafxmmMVP/1.0')]

[template public generateJFXCode(aJavaFXPackage : JavaFXPackage)]
[comment @main/]
[for (theScene : Scene | aJavaFXPackage.VPackage.views)]
[file ('/view/'+theScene.Name.toUpperFirst().replaceAll(' ', '')+'.fxml',
    false, 'UTF-8')]
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane xmlns:fx="http://javafx.com/fxml/1"
    xmlns="http://javafx.com/javafx/2.2"
    fx:controller="controller.[theScene.hasPresenter.
          Name.escapeSpecialChar().trim()/]">
  <children>
    <BorderPane>
    [for (theRoot : Root | theScene.roots)]
        [createViewFiles(theRoot)/]
    [/for]
        </BorderPane>
    </children>
```

Fig. 8 Acceleo portion of the transformation templates

C. GMF Graphical Tool

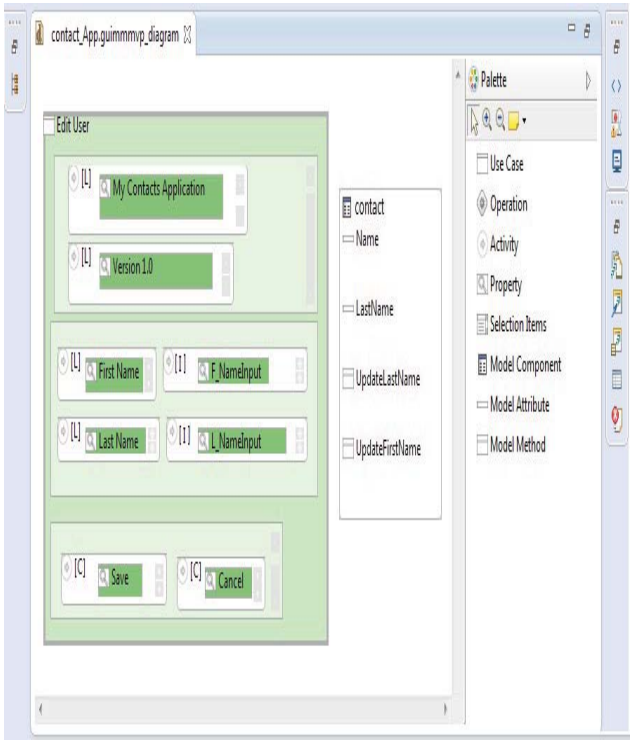


Fig. 9 The graphical Modeling Editor showing an excerpt the design model of one view of the Contacts Application

In order to simplify the task for the user, a graphical editor was developed to enable creation, edition and view of the models in a graphical way that reminiscent of the typical structure of an application view. Here after a screen shot of the input model generated using the graphical tool.

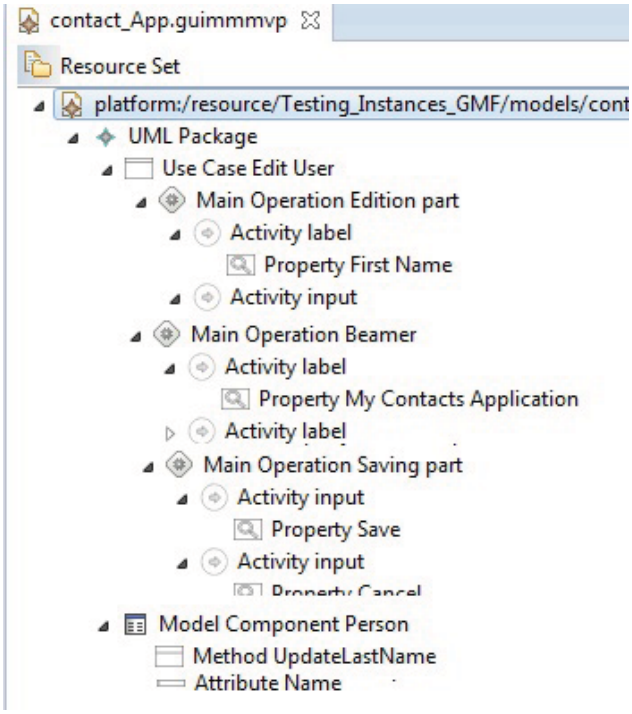


Fig. 10 The input model file generated from the developed Graphical Tool

VI. From a Model to a Running Example

In order to validate our approach, we applied it to generate a simple Contact application for searching and editing contacts information. The application enables its users to: View the contacts in a list, select a contact, edit a contact, update a contact's information. We provide the design model, instance of the proposed PIM meta model, as input for the transformation engine and using the graphical tool, as described in Fig. 9. The code source and files were then produced to implement the application first designed.

A. The Contact Application Design Model

Fig. 10 shows the input model designed using the graphical tool developed for the approach. We defined the two views that we want for our application with all the operation and activities that the user expect to find in the final application. Also, we find the model elements that will be used in the processing of information in the application. For instance, we have the Edit view that have tow main operations that are the view of the user's information and the saving/canceling part. The first operation needs inputs and labels components, while the others needs tow buttons. In addition, a model component Person is needed with first name and last name attributes is needed. Once the input model of the contacts application is sufficiently modeled respecting the PIM meta model, the resulting file is used as an input for the transformation engine developed for the approach. Indeed, we will first generate a MVP model for the contact Application that respects the PSM defined metamodel. Then, it will became the input for the code generation part.

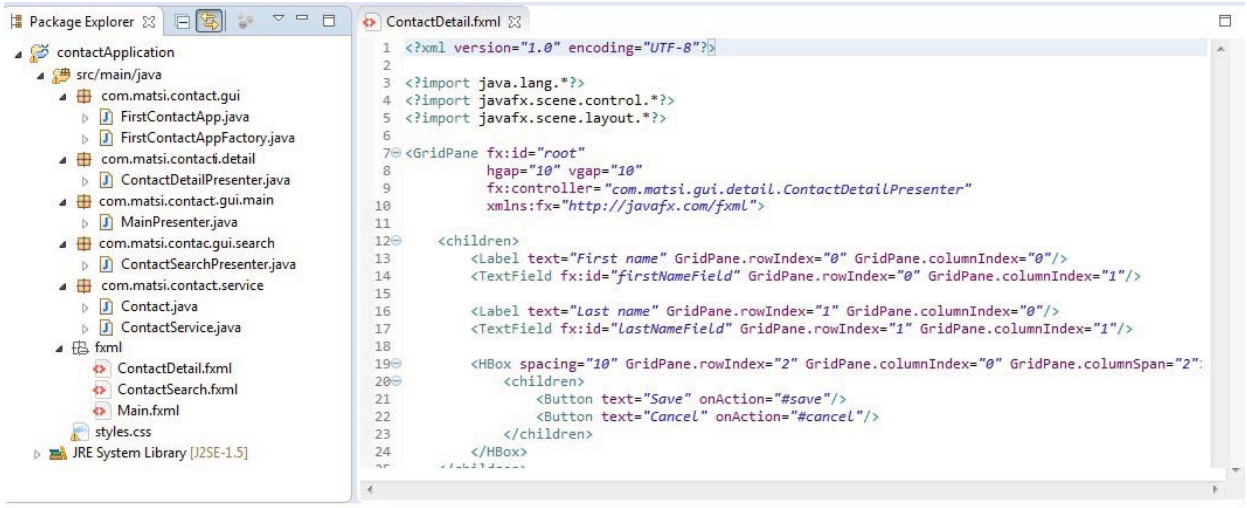


Fig. 11 The contact application RIA as an Eclipse project

B. The Contact Application Generated RIA

After the design and generation of the application, the set of artifacts produced by the code generator tool constitute a RIA ready to be deployed. Fig. 11 shows the generated RIA loaded into Eclipse IDE as JavaFX project. The Eclipse package explorer shows the list of code artifacts produced respecting the MVP pattern and the view editor shows the FXML file responsible for the view part.

The conducted case study confirmed the correct functioning of the tools developed to support our approach. We can say that the fast and automatic generation of RIA based only on the designed input model makes it possible to verify and validate the design itself and to undertake a design refinement process effortlessly.



Fig. 12 The main view of the Contact App automatically generated



Fig. 13 The detail view of the selected contact from the application

VII. Conclusion and Perspectives

In this paper we have presented an approach for the model-driven generation of Rich Internet Application using Eclipse technologies and frameworks such as EMF, GMF, QVTo and Acceleo. The approach consists of first defining tow metamodels as PIM and PSM respecting the MVP pattern. Second, we defined the transformation rules for both Model To Model and Model To Text transformation. Finally, we developed a graphical tool to help manage and use the approach efficiently. A case study conducted on designing and generating a RIA for contact management has shown that the approach is valid and the supporting tools work properly. In particular, the approach enables effortlessly repeating the development cycle ”modeling-generating-validating” to verify and improve the design of the application.

In the future, this work will be extended to allow the generation of other components of RIAs besides the configuration files, we also aim at refining the graphical tool and enlarge the specter of use for other technologies, patterns and supporting other interesting target platforms (e.g., for mobile devices).

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

Open Science Index, Computer and Information Engineering Vol:10, No:2, 2016 publications.waset.org/10003854/pdf

REFERENCES

[1] OMG - Object Management Group (MOF, MDA, XMI, QVT, UML, MOFM2T) - http://www.omg.org/.
[2] D. C. Schmidt. Model-Driven Engineering. Computer, 39:25 31, 2006. IEEE Computer Society
[3] G. M. Kapitsaki, D. a. Kateros, G. N. Prezerakos, and I. S. Venieris, "Model-driven development of composite context-aware web applications," Inf. Softw. Technol., vol. 51, no. 8, pp. 12441260, 2009.
[4] Z. Ahmed and V. Popov, "Integration of Flexible Web Based GUI in I-SOAS," 2010.
[5] S. Meli, J. Gmez, S. Prez, and O. Daz, "A model-driven development for GWT-based rich internet applications with OOH4RIA," Proc. - 8th Int. Conf. Web Eng. ICWE 2008, pp. 1323, 2008.
[6] Miller, J., Mukerji, J., al. MDA Guide Version 1.0.1 (OMG, 2003).
[7] N. Koch, S. Melia-Beigbeder and J. Vara-Mesa. ModelDriven Web Engineering. European Journal for the Informatics Professional - Joint issue with NOVATICA, IX(2):4045, April 2008.
[8] Z. Ahmed and V. Popov, Integration of Flexible Web Based GUI in ISOAS, 2010.
[9] S. Meli, J. Gmez, S. Prez, and O. Daz, A model-driven development for GWT-based rich internet applications with OOH4RIA, Proc. - 8th Int. Conf. Web Eng. ICWE 2008, pp. 1323, 2008.
[10] J. C. Preciado, M. Linaje, R. Morales-Chaparro, F. Sanchez-Figueroa, G. Zhang, C. Kroi, and N. Koch, Designing rich internet applications combining UWE and RUX-method, Proc. - 8th Int. Conf. Web Eng. ICWE 2008, pp. 148154, 2008.
[11] N. Koch and A. Kraus. The expressive power of uml-based web engineering. In Proc. of the 2nd International Workshop on Web Oriented Software Technology, IWWOST2002. Springer Verlag, 2002.
[12] Gharavi, V., Mesbah, A., Deursen, A. V., Modelling and Generating AJAX Applications: A Model-Driven Approach. Proceeding of the7th International Workshop on Web- Oriented Software Technologies, New York, USA (Page: 38, Year of publication: 2008, ISBN: 978-80-227-2899-7)
[13] The Eclipse Modeling Project. http://www.eclipse.org/modeling



Sarra Roubi S. Roubi is a PhD student with the MATSI Laboratory, Oujda Institute of Technology, Morocco, 60000 e-mail: roubi.sarra@gmail.com
She got a degree of engineer in Computer Science from the National School of Applied Science. She is focusing her researches on the Model Driven Engineering approach applied to the automatic generation of the Graphical User Interface.



Mohammed Erramdani M. Erramdani is with the MATSI Laboratory, Oujda Institute of Technology, Morocco, 60000 e-mail: m.erramdani@gmail.com
He is a professor in the Department of Management at the Institute of Technology and teaches the concept of Information System. He got his thesis of national doctorate in 2001. His activities of research in the MATSI Laboratory (Applied Mathematics, Signal Processing and Computer Science) focusing on MDA (Model Driven Architecture) integrating new technologies XML, EJB, MVC, Web Services, etc.



Samir Mbarki S. Mbarki is with the Computer Science Department of Ibn Tofail University, Kenitra, Morocco, e-mail: samirmbarki@hotmail.com
He is a professor in the Department of Computer Science at Faculty of Science Ibn Tofail University. His research interests include software engineering, model driven architecture, software metrics and software tests. He obtained an HDR in computer science from Ibn Tofail University in 2010.