Extending Graphical Part of the Interaction Flow Modeling Language to Generate Rich Internet Graphical User Interfaces

Sarra Roubi[1], Mohammed Erramdani[1] and Samir Mbarki[2]
[1]*High School of Technology, Mohamed First University, Oujda, Morocco*
[2]*Department of Computer Science, Ibn Tofail University, Kenitra, Morocco*

Keywords: Model Driven Engineering, Interaction Flow Modeling Language, Transformation, Model, Meta Model, User Interface.

Abstract: Rich Internet Applications (RIAs) combine the simplicity of the hypertext paradigm with the flexibility of desktop interfaces. 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 an approach for the model driven generation of Rich Internet Application using the Interaction Flow Modeling Language (IFML). The approach exploits the new language IFML recently adopted by the Object Management Group by extending first the graphical part of the Meta Model to fit the RIAs' needs. We used frameworks and technologies known to model-driven engineering, such as Eclipse Modeling Framework (EMF) for Meta Models, Query View Transformation (QVT) for model transformations and Acceleo for code generation. The approach allows to quickly and efficiently generating a RIA focusing on the graphical aspect of the application.

1 INTRODUCTION

HTML-based Web applications have shown 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.

The Model Driven Engineering has been introduced to master complexity and ensures consistency of applications and has shown its efficiency. Besides, it helps improving the quality of applications as well as time savings and increasing the productivity.

Indeed, several researches have applied model-driven techniques to the specification of software application and precisely interfaces and user interaction. Among them, the ones focusing on Web interfaces like OOH-Method (Gmez et al., 2001), WebML (Ceri et al., 2002), HERA (Vdovjàk et al., 2003) and RUX-Model (Linaje et al., 2007). Furthermore, some approaches apply model driven techniques for multi-device UI modeling, such as TERESA (Berti et al., 2004), MARIA (Paterno et al., 2009), IFML (Brambilla et al., 2014)…

However, none of them specifically addresses the needs of RIAs in terms of Graphical aspects and its connection with the application layers respecting a given Design Pattern such as Model View Controller, Model View Presenter...

In this paper, we propose a model-driven approach and the idea is to focus 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 a MVP pattern and provides well designed graphical user interfaces. To do this, we used the OMG standard language called Interaction Flow Modeling Language (IFML). We propose an extension of the IFML suitable for graphical characteristics of RIAs, respecting the MVP pattern, and we describe the process and the proposed Meta Models, the extensions added and the transformation process with QVTo standard.

The paper is organized as follows section 2 reviews the related work; Section 3 summarizes the

Roubi, S., Erramdani, M. and Mbarki, S.
Extending Graphical Part of the Interaction Flow Modeling Language to Generate Rich Internet Graphical User Interfaces.
DOI: 10.5220/0005650601610167
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 161-167
ISBN: 978-989-758-168-7

Model Driven Engineering and introduces the IFML. Section 4 and 5 present respectively the work elaborated and the running example and Section 6 concludes.

2 RELATED WORK

This work is related to several works that dealing with conceptual modeling of software applications. Among these works, there are those focusing on the Web: The Web Modeling Language (WebML) (Ceri, 2002), defined as a conceptual model for data-intensive Web. Also, we find the OO-HDM (Schwabe and Rossi, 1995), a UML-based approach for modeling and implementing Web application interfaces. Moreover, WebDSL (Groenewegen et al., 2008) is a domain-specific language consisting of a core language with constructs to define entities, pages and business logic. In addition, we find HERA (Vdovjàk et al., 2003) which is a model-driven design approach and specification framework focusing on the development of context-dependent or personalized Web information system.

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 Rich Internet Applications 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

Other recent proposals in the Web Engineering field represent the RIA foundations (Urbieta et al., 2007) by extending existing Web engineering approaches. We also find 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 (Preciado et al., 2008).

Also, an MDA approach for AJAX web applications (Gharavi et al., 2008) 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.

3 MODEL DRIVEN ENGINEERING (MDE)

3.1 Transformation Process in MDE

In MDE, every artefact, 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. This brings up the three different layers of abstraction that can be described as follow:

- **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.

Once the Meta Models developed, MDE provides the transition between the CIM, PIM and PSM models through the execution of models transformation. A transformation converts models with a particular perspective from one level of abstraction to another, usually from a more abstract to less abstract view, by adding more detail supplied by the transformation rules. There are two types of transformations in the MDA approach:

- Model To Model: it concerns the transition 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 specific programming language as a target.

For the **Model To Model** transformation, using the modeling approach is designed to have a sustainable and productive models transformation, independently of any execution platform. This is why the OMG has developed a standard for this transformation language which is the MOF 2.0 QVT (OMG - Object Management Group (MOF, MDA,

XMI, QVT, UML, MOFM2T) - http://www.omg.org/.) standing for Query View Transformation. QVT is hybrid character (declarative / imperative) consisting of three languages: QVT-Relation, QVT-Operational and QVT-Core.
1

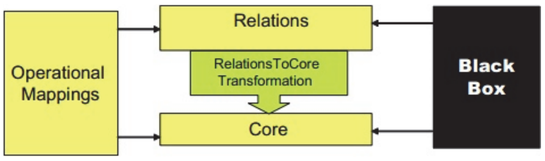


Figure 1: Query View Transformation Architecture.

The declarative part is defined by the Relation and Core languages with different levels of abstraction, while the Imperative part is defined by the operational language. Finally, a black box is defined in the MOF 2.0 QVT standard that enables escaping the whole transformation/library or its parts that are difficult or impossible to implement in pure QVT. See Figure 1.

For this work, we used the QVT-Operational mappings language implemented by Eclipse modeling Framework (www.eclipse.org).

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 being to parse the representation of the model in XML file format Metadata Interchange (XMI) (Schwabe and Rossi, 1995) and apply a number of templates for transforming models conforming to a MOF metamodel into text; source code. 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.

3.2 The Interaction Flow Modeling Language

The Interaction Flow Modeling Language (IFML) is designed for expressing the content, user interaction and control behaviour of the front-end of software applications. In other words, it supports the platform independent description of graphical user interfaces for software applications that can be accessed or deployed on various systems as desktop computers, laptop computers, PDAs, mobile phones, and tablets. Besides, an IFML model should supports several design perspectives, among them:

- The view structure specification, which consists on the definition of view containers, their nesting relationships, their visibility...
- The view content specification that treats the content and data entry elements contained within view containers.
- The events specification that can be produced by the user's interaction, by the application, or by an external system and may affect the state of the user interface.
- The parameter binding specification, which consists on the definition of the input-output dependencies between view components and between view components and actions.

Figure 2 shows a simple example of IFML model where the user can search for a product by entering some criteria in the Product Search Form. The matching items are shown in a list. The selection of one item causes a delete action to be triggered on it. When the deletion is completed, the updated list of products is displayed again. IFML concepts can be stereotyped to describe more precise behaviours (Brambilla et al, 2014).

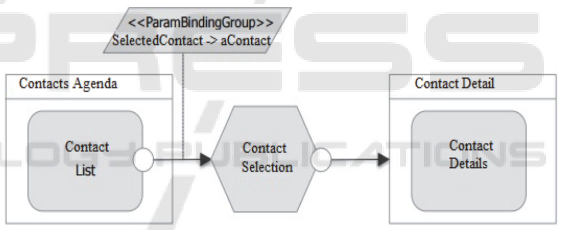


Figure 2: IFML excerpt of the running example: Contact Agenda.

4 THE MODEL DRIVEN PROPOSED PROCESS

Within the MDE context, several frameworks, modeling technologies and transformation languages can be used for different domains and purposes. Among them, we find IFML, EMF, QVT and Acceleo.

In this section, we describe first the extension of the IFML to fit the RIA modeling. After that, we defined an MVP Meta Model for RIA, then we present the transformation rules to generate the final application. Indeed, we propose a model-driven development process to enable the automatic generation of a fully working RIA starting from a simple model.

4.1 Extending IFML for RIA

Using IFM for modelling the graphical user interface makes it possible to clarify the interaction between the pages and navigation. However, we found that the part defining the graphical part of the application in terms of components and layouts did not provide sufficient information. Also, the RIA propose rich graphical interfaces that bring the richness of desktop applications to the Web. We notice also that the interactive dimension and speed of execution are particularly taken into account in these Web applications.

That's why we thought of extending the IFML language to suit the needs of the development of RIAs and respect the Design Pattern on implementing platforms.

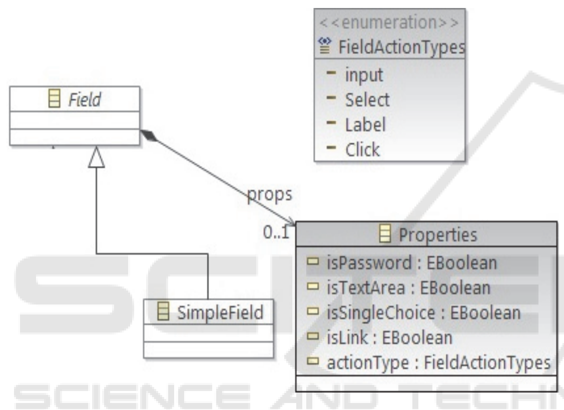


Figure 3: IFML Extension: Adding properties to Simple fields to identify the suitable RIA component.

Indeed, these extensions reflect specifically the graphic part of RIAs, that is to say, the components and their properties. As shown in Figure 3, we added to "SimpleField" properties to identify which component type suits the best and describes efficiently the user-defined action based on the user's "operating" with the interface. Our major purpose is to keep the task as simple as possible and abstract any technical specification so the approach can be easily used by any stakeholders with or without technical knowledge of the platform target. So the FieldActionType gives the information about the action to be performed on the component; Click, Input, Selection... Then, properties help to pick the right component, a button or a link, a text field or a text area...

4.2 RIA Proposed Meta Model

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 the Figure 4, 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.

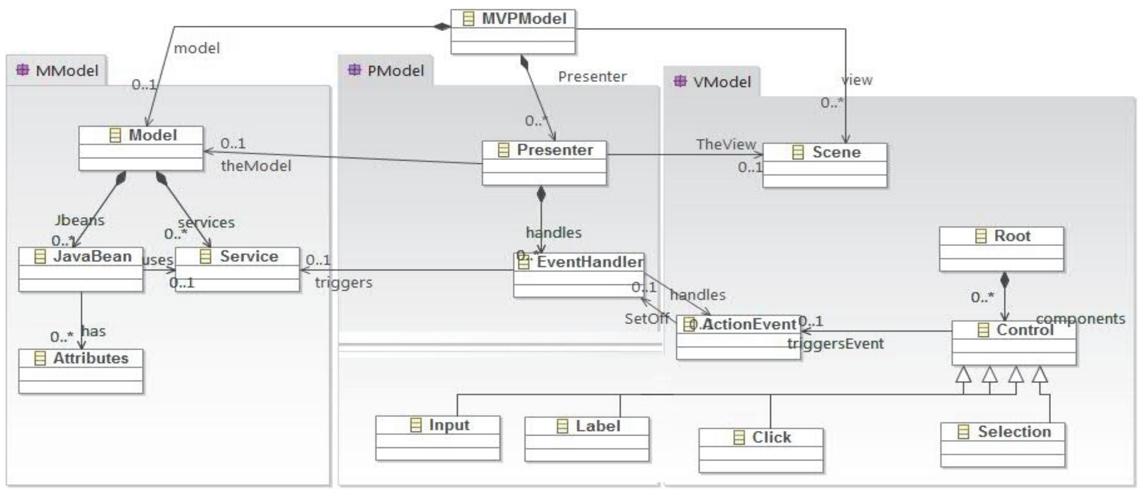


Figure 4: MVP proposed Meta Model for RIA.

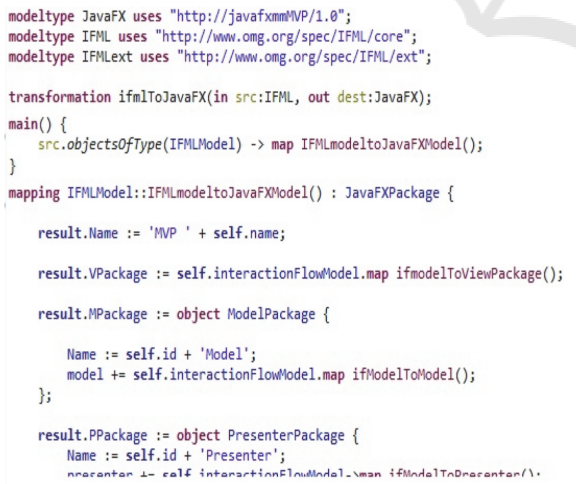
In the view part, the scene is composed of graphical components, named controls that could be containers as Roots. From the IFML input and the extension added, we can have more specific information about the component to add and which container it belongs to. We added a hierarchical relationship between graphical components base on their type. We also thought about the composite relationship that can connect containers with simple components.

4.3 Transformation Process

4.3.1 Model to Model Transformation

Once the Meta Modeling phase established, we defined the transformation rules. Since we have defined the extended IFML and PSM for MVP RIA Meta Models, we need to define the Model To Model transformation using the QVTo standard respecting a defined algorithm.

The entry point of the transformation is the main method. This method makes the correspondence between all elements of the IFML model of the input model and the elements of type JavaFXPackage output model. For instance, for each interactionFlowModel we create the equivalent ViewPakage that will gather all the graphical aspect of the RIA. To that, we connect the Preseter and the Model also creates for the IFML model.The Figure 5 below shows an excerpt of the Transformation program:



```
modeltype JavaFX uses "http://javafxmmMVP/1.0";
modeltype IFML uses "http://www.omg.org/spec/IFML/core";
modeltype IFMLext uses "http://www.omg.org/spec/IFML/ext";

transformation ifmlToJavaFX(in src:IFML, out dest:JavaFX);
main() {
    src.objectsOfType(IFMLModel) -> map IFMLmodeltoJavaFXModel();
}
mapping IFMLModel::IFMLmodeltoJavaFXModel() : JavaFXPackage {

    result.Name := 'MVP ' + self.name;

    result.VPackage := self.interactionFlowModel.map ifmodelToViewPackage();

    result.MPackage := object ModelPackage {

        Name := self.id + 'Model';
        model += self.interactionFlowModel.map ifModelToModel();
    };

    result.PPackage := object PresenterPackage {
        Name := self.id + 'Presenter';
        presenter += self.interactionFlowModel->map ifModelToPresenter();
```

Figure 5: Query View Transformation Code excerpt.

4.3.2 Model to Text Transformation

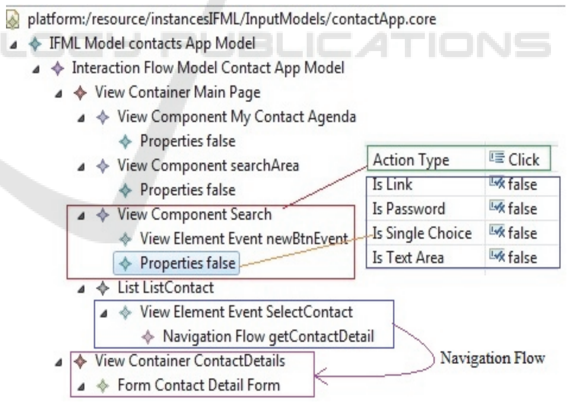
When the model file for RIA is generated, the next step is to be able to generate the whole code of the application. To do this, we used the OMG standard Acceleo, whose operating principal is to use templates to generate code from a model. The realization of a new generation module therefore requires the creation of templates. So, we defined templates 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.

5 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 and edit its information.

We provide the design model, instance of the extended IFML, as described above, which is basically the only input file to our generator. The Figure 6 shows the input model.

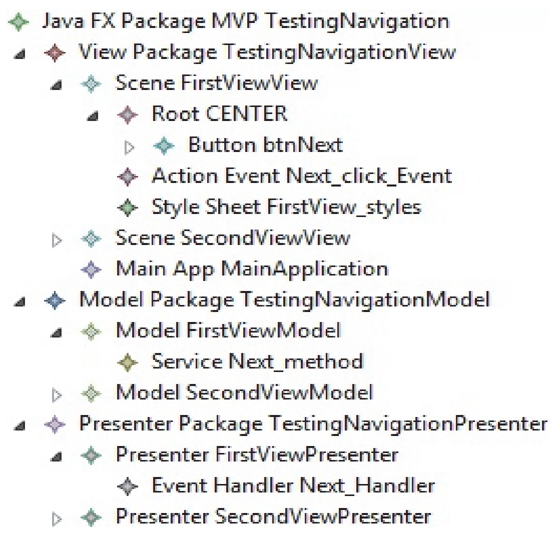


```
platform:/resource/instancesIFML/InputModels/contactApp.core
  IFML Model contacts App Model
    Interaction Flow Model Contact App Model
      View Container Main Page
        View Component My Contact Agenda
          Properties false
        View Component searchArea
          Properties false
        View Component Search
          View Element Event newBtnEvent
          Properties false
        List ListContact
          View Element Event SelectContact
            Navigation Flow getContactDetail
      View Container ContactDetails
        Form Contact Detail Form
```

Action Type	Click
Is Link	false
Is Password	false
Is Single Choice	false
Is Text Area	false

Figure 6: Input model for the contact application.

Once the application has been sufficiently modelled using the extended IFML, the generated file, as shown in Figure 7, is a model respecting the Meta Model for RIA implementing the MVP Design Pattern. This file will be considered as the input for the code generator and will provide all the source code files needed for the application to be almost running. The graphical aspect of the application and the user interaction are the major focus on this approach.



```
Java FX Package MVP TestingNavigation
  View Package TestingNavigationView
    Scene FirstViewView
      Root CENTER
        Button btnNext
      Action Event Next_click_Event
      Style Sheet FirstView_styles
    Scene SecondViewView
    Main App MainApplication
  Model Package TestingNavigationModel
    Model FirstViewModel
      Service Next_method
    Model SecondViewModel
  Presenter Package TestingNavigationPresenter
    Presenter FirstViewPresenter
      Event Handler Next_Handler
    Presenter SecondViewPresenter
```

Figure 7: The generated file for the MVP RIA of the Contact Application.

The generated file represents the input file for the code generator that was developed using Acceleo. The idea is to generate the files for a MVP Rich Internet Application focusing on the graphical aspect. Here after in Figure 8, the main view of the generated application based on the IFML extended model only.

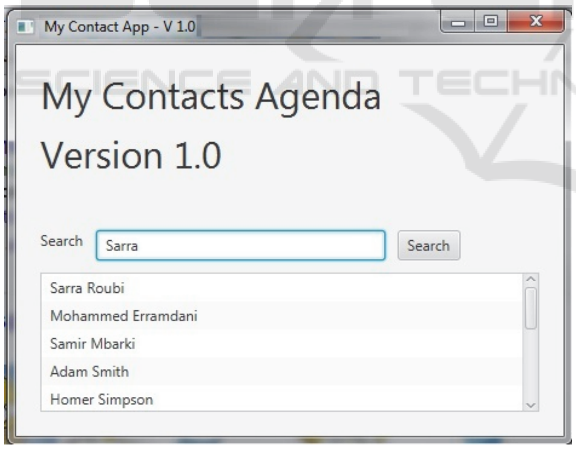


Figure 8: The main view of the generated MVP RIA Contact Application.

The second view of the application consists on editing the contacts detail as a result of the click event on the list. Information sent are relative to the contact selected. Figure 9 shows the view of the edition of the contact details.

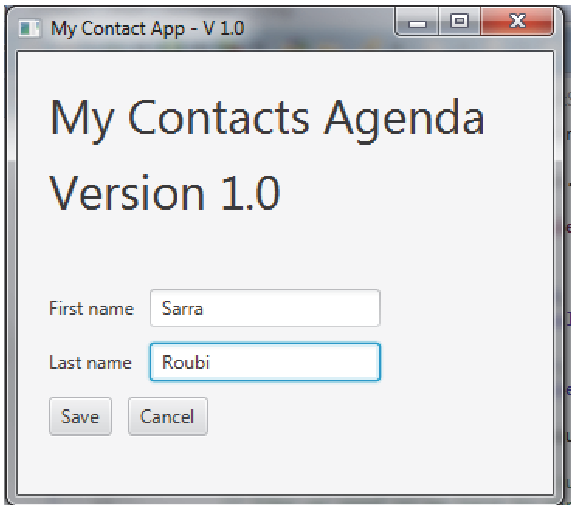


Figure 9: The edition view of the generated MVP RIA Contact Application.

6 CONCLUSION AND PERSPECTIVES

In this paper we presented an extension of OMG's standard IFML (Interaction Flow Modeling Language) for Rich Internet Application development based on a Model Driven Development process. Furthermore, we defined a Meta Model for RIA that respects the Model View Presenter pattern and developed the transformation engine that allows the automatic generation of the output model. This generated modeled represents an input to a Model To Text generator and give an almost complete RIA ready to be deployed, focusing on the graphical aspect of the application on one hand, and the user event handling on the other.

The major contribution in the proposed approach is the simplification of the IFML and the abstraction of technical details, besides the addition of the extension part that describes efficiently the graphical components to be chosen to accomplish the user's expectation of the application. By using this Model Driven method the RIA can be easily generated without having to know all the technical specification of the execution platform.

Future works will cover the implementation of more refined code generator and the application of the proposed method to estimate how this approach scales in large projects. Also, we aim at enrich the IFML and target several platform for mobile, desktop and Web application starting from the same input model.

REFERENCES

Berti, S., Correani, F., Mori, G., Paterno, F., and Santoro, C., 2004. Teresa: a transformation-based environment for designing and developing multidevice interfaces. *In CHI Extended Abstracts*, pages 793–794.

Brambilla, M. et al., 2014. Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End To cite this version : Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End.

Brambilla, M., Fraternali, P., et al, 2014. *The interaction flow modeling language (ifml), version 1.0. Technical report*, Object Management Group (OMG), http://www.ifml.org.

Ceri S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M., 2002. *Designing Data-Intensive Web Applications. The Morgan Kaufmann Series in Data Management Systems*. Morgan Kaufmann Publishers Inc.

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* (Page: 38, Year of publication, ISBN: 978-80-227-2899-7).

Gmez, J., Cachero, C., Pastor, O., 2001. *Conceptual modeling of device-independent web applications*. pages 26–39.

Groenewegen, D., Zef Hemel, Lennart C. L. Kats, and Eelco Visser, 2008. Webdsl: a domain-specific language for dynamic web applications. *In Gail E. Harris, editor, OOPSLA Companion*, pages 779–780. ACM.

Linaje, M., Preciado, J. C., and 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, pages 226–241.

Martinez-Ruiz, F. J., Arteaga, J. M., Vanderdonckt, J., and 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*, pages 3238. IEEE Computer Societ.

Paterno, F., Santoro, C., and 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).

Schwabe, D. and Rossi, G., 1995. The object-oriented hypermedia design model. *Communications of the ACM,* 38(8), pp.45–46.

Schwabe, D., Rossi, G., 1995. *The object-oriented hypermedia design model*. pages 45–46.

Urbieta, M., Rossi, G., Ginzburg, J., and Schwabe, D., 2007. Designing the Interface of Rich Internet Applications. *In Proc. LA-WEB'07*, pages 144–153.

Vanderdonckt, J., 2005. A MDA-compliant environment for developing user interfaces of information systems. *In CAiSE*, pages 16–31.

Vdovjak, R., Frasincar, F., Houben, G. J, and Barna, P., 2003. *Engineering Semantic Web Information Systems in Hera*. Journal of Web Engineering, 1(1-2):3–26.