Combining Ontologies and IFML Models Regarding the GUIs of Rich Internet Applications

Naziha Laaz(✉) and Samir Mbarki(✉)

MISC Laboratory, Faculty of Science Kenitra, Ibn Tofail University, Kenitra, Morocco
laaznaziha@gmail.com, mbarkisamir@hotmail.com

Abstract. Rich Internet Applications (RIAs) is a new kind of web applications. These applications provide more effective graphical components and promote the fusion of traditional applications and client-server applications. They also furnish convivial and interactive interfaces similar to desktop applications. However RIAs designing and implementation are time and cost consuming. To meet RIAs requirements, we propose a new approach based on Model Driven Engineering methodology to generate GUIs from abstract models. The structural and dynamic aspects of GUIs are modeled to represent complete RIA interfaces. Our model driven development process is based on Ontology and IFML; The logical description of UI components is presented by the ontology domain and their interactions are captured by IFML. The proposed process takes as input abstract models. Then, we apply transformations on these models to produce a code representing Flex rich interfaces. Our approach is illustrated by an example of an e-commerce web site interface.

Keywords: Ontology Definition Metamodel (ODM) · Ontologies · Interaction Flow Modeling Language (IFML) · Platform Independent Model (PIM) · Model Driven Engineering (MDE) · Graphical User Interface (GUI) · Rich Internet Application (RIA)

1 Introduction

Last decade and half ago, systems have been equipped with sophisticated GUIs, and their complexity increases in time. Powerful interaction functionalities are implemented on top of variety of technologies and platforms whose boundaries are becoming less distinguishable: client-server applications, Web applications, rich Internet applications, mobile applications. Our proposal is focused on GUIs of RIAs; these applications have combined the richness and interactivity of desktop interfaces into the web distribution model.

However, software development needs to be more abstract with developed practices [1]. So, researches in software development have focused on abstract models of user interfaces and new modeling language standards have appeared. They have become more powerful in expressing requirements at a high abstraction level. Indeed, the Object Management Group (OMG) launched an effort known as the Model Driven Architecture [2] to align with these changes in technology, and raise the level of abstraction of

C. Dichev and G. Agre (Eds.): AIMSA 2016, LNAI 9883, pp. 226–236, 2016.
DOI: 10.1007/978-3-319-44748-3_22

physical systems. Consequently, many solutions have been emerged to describe and generate graphical interfaces; most of them respect MDA approach. In parallel, many ideas have been proposed which are based on ontologies by integrating it in the description and generation of graphical interfaces. In order to make these ideas functional and evolve to better ontology-driven development practices using, OMG took the initiative to define the Ontology Definition Metamodel [3] by marrying ontologies with metamodeling. Most of works based on ODM are focused on the database and business layers [4], while some works was done to generate presentation form of GUIs. In the other hand, IFML was recently defined to describe the elements and behavior of user interfaces in order to generate code implementation of these interfaces [5].

To meet these requirements, we present a new MDE approach combining the two new standards IFML and ODM to benefit from each other in order to generate rich user interfaces of RIA platform. We used known frameworks and technologies of 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 generate a RIA focusing on the graphical aspect of the application. It can be replicated for different target technologies and platforms.

The rest of this paper is organized as follows: Sect. 2 describes our contribution; we have two sub-sections in this part: The formal ontology of GUIs respecting the syntax OWL 2.0 and IFML. Section 3 is dedicated to the related work. In Sect. 4, we explain the process of Ontology-Driven UIs; it is divided into tree sub-parts: The definition of PIM models, PIM to PSM transformation and code generation from PSM model. Finally, the last part will present a conclusion indicating the status of the objectives and describing future work.

2 Methodology and Contribution

Nowadays, the GUIs are deployed in heterogeneous and interactive spaces: They are spread over a different kind of platforms. This allows thinking of adapting new ways of developing the presentation layer of the application. In this work, we present an approach based on MDA, which proposes a solution for the GUIs abstract representation and their automatic generation to a specific platform.

Since, our ultimate goal is to raise the abstraction level of the GUIs definition that include the structural and dynamic aspects. These two aspects cover all information related to the nature of graphical components, their properties and interactions. Instead, we have developed a process to automatically map from a high-level representation to a lower-level language. The proposed approach specifies two transformations during the development cycle, models becoming more and more concrete until obtaining the code by successive transformations, We have two PIM; two models of the most abstract level, are transformed into a PSM, then a second transformation is established to generate code from the specific model to the Flex platform. After a study on the various GUIs PIM, we detected that the difficult resides in the choice of abstract input models. The two chosen PIM represent two specifications proposed by the OMG. The first PIM, ODM,

is the metamodel that defines ontologies which supports several different ontology representation systems. With this metamodel, we developed our ontology of graphical components. The approach exploits the new language IFML as a second PIM by extending the graphical part of the MetaModel to fit the RIAs' needs. We choose IFML because it allows obtaining all information of interactions between components represented in the GUI ontology.

We established an analysis to represent different aspects of GUIs with these two specifications as presented in the following sub sections.

2.1 GUIs Ontology Respecting OWL2.0 Syntax

We present a semantic approach to the problem by defining ontology for graphical user interfaces. In this section, we show how the GUI domain concepts are presented in OWL 2.0 using OWL DL. The GUI ontology is formed by three concepts: Declarations, Axioms and Assertions. The basic elements of user interfaces domain are expressed by Declarations. There are five declaration types: Classes, ObjectProperty, DataProperty, DataTypes and Individuals. The GUI contains elements divided into: containers and controls; The containers provide a space where controls can be located, and the controls are the elements that display content or accept user input in interfaces (buttons, fields, lists e.g.). These two concepts are represented as classes to group items with similar characteristics resources (buttons, Menus, Field, Window e.g.), and individuals represent instances of classes.

Object and data properties can be used to represent relationships in the domain [6]. The relationship between individuals of the two classes is represented by ObjectProperty, we use this property to link a container with controls or a container with another container. For example, we have defined an objectProperty named **"composedOf"** to say that the class Menu consists of several MenuItem classes, or, to define the relationship between the two classes List and ListItem, etc. Howerver, datatypes are sets of literals such as strings or integers. All these declarations are grouped by axioms, in order to form complex descriptions from the basic entities.

There are three kinds of axioms: Class Axioms, ObjectProperty Axioms and DataProperty Axioms. Each axiom is associated with an expression. Properties Expressions are characterized by a domain and range; a domain is represented by classes and ranges can be classes or dataTypes. However, there are other types of axioms linked to ClassExpression. We consider two classes: Control and List. Control is more general than List, which means that every time we know that an individual is a list, this individual must be a control. In OWL 2, this is done by a so-called axiom subClassOf, that has List as subClass Expression and Control as superClass Expression.

To distinguish between the different types of containers and controls, we defined DataProperties Axioms which have Boolean datatype as Range. The sets of this data Properties are represented in Table 1. For example, we chose isDefault for container when it is a default container such as homepage or welcome interface. In addition to this data properties that give semantics to widgets, there are another dataproperties that define several characteristics such as a geometry (x, y, width, height) and "hasText", "hasname"; text and name attaching to widgets.

Table 1. Nature of GUI Concepts

GUI concept	DataProperties	Description
List	isSimple	Scrolls elements vertically, on a single column.
	isCombo	list items vertically by displaying only the selected item.
Field	isStatic	the components that display text
	isEntry	the controls that allow the user to enter text
Button	isPush	graphical control element that provides the user a simple way to trigger an event
	isRadio	Represents a selection of one item from a list of items. Occur only in groups. Selecting one radio button deselects the others.
	isCheck	Possibility of multiple selections
Window	isModal	Designed to block any user interaction in all other previously active containers.
	isDefault	A home page or welcome default container.
	isXOR	the Container comprising child or containers that are displayed alternatively
	isLandmark	container is reachable from any other element of the user interface without having interactions with other containers

We can add some subPropertyOf Axioms by attaching them with dataProperty and their subProperty Expressions (subPropertyOf Axiom has "hasSize" as dataProperty and "width", "height" as subProperties). After detailed declarations and axioms, it remains to talk about the role of the assertions in the concepts definition of GUI domain. The assertions can be Class Assertions, ObjectProperty Assertions, or DataProperty Assertions. The assertions are intended to clarify how individuals relate to other individuals. In the section reserved to the ontology-driven UI development process, we will present the logic model of GUIs as result of this analysis, respecting the ODM metamodel that meets the syntax OWL2.0.

2.2 Extending Interaction Flow Modeling Language

The new OMG Interaction Flow Modeling Language standard (IFML) is defined in March 2014 [7]. IFML is a platform independent model (PIM) that can be used to express interaction design decisions independently of the implementation platform. It allows to capture the user interaction and content of the front-end (user interface) and model the control behavior of that system's user interface.

In the approach presented in this paper, we used IFML as PIM-level interaction flow modeling, it brings several benefits to user interfaces development process of web applications. It permits the formal specification of the different perspectives of the user interface such as interface composition, interaction and navigation options. This work uses one of four technical artifacts defined by the present specification [5], which is the IFML metamodel.

IFML metamodel is composed of three packages: The Core package, the Extensions package and the DataTypes package. The Core package contains the concepts that build up the interaction infrastructure of the language in terms of InteractionFlowElements, InteractionFlows, and Parameters. The Extension package extends the concepts defined by Core package by concrete concepts with more complex behaviors. While The DataTypes package contains the basic data types defining in the UML metamodel, and specializes a number of UML metaclasses as the basis for IFML metaclasses, and presumes that the IFML DomainModel is represented in UML. After studying the various packages of IFML metamodel, we noted that the part defining the graphical application components and their characteristics did not provide sufficient information. IFML is extensible, in fact, we thought to expand it to meet the needs of RIA development and implementation of overall output platforms.

We have completed the IFML metamodel by defining the GUI ontology. In addition, we modify the core package of the metamodel. So, we added Ereference to ViewComponent metaclass which has isContainment() as a method because ViewComponent can be composed of viewcomponents like a form composed of List As seen in (Fig. 1).

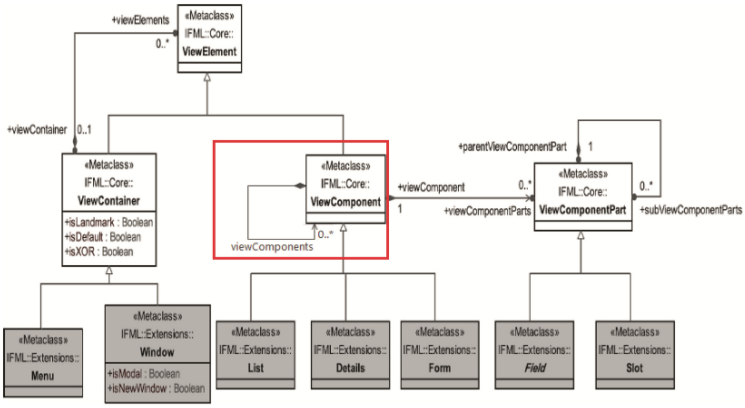

Fig. 1. IFML extension.

3 Related Work

Ontologies provide a formal representation of knowledge and the relationships between concepts [8]. Recently, a number of use cases have been proposed that employ ontologies for modeling user interfaces and their components, Examples are automatic generation of explanations for user interfaces, adaptation of user interfaces for different needs and contexts, and integration of user interface components [9].

Those use cases require a strongly formalized ontology of the domain of user interfaces and interactions. In this regard, many works have been developed. In [10], the authors discuss the differences between the UI description languages and formal ontologies and how they can benefit from each other. Their goal is to define a formal ontology

of user interfaces and interactions domain. The formal ontology will not replace user interface description languages, but will be a valuable enhancement.

With the appearance of MDE, great research effort has been dedicated to this methodology marrying it with ontology. Lot of them are focused on conceptual models. The most relevant are: [11–15]. In the other hand, few UI generation approaches based on MDE have been defined in recent years. In [16], the authors describe a method for rich UI development for data-intensive Web applications based on OWL2 ontologies, which applies model-driven engineering to derive a user interface from the domain ontology, incorporating modern rich components for Web-based interfaces. The model-driven process proposed is supported by the TwoUse Toolkit [19] for OWL2 authoring and management. In our approach we also use ontologies as domain model, but we combine it with IFML model to derive a complete presentation of Web UIs.

However, [17] presents an approach to the problem of porting graphical user interfaces by aligning representations of user interfaces in different technologies to an abstract semantic web model for graphical user interfaces. Our proposal has similarities with this approach in sense that we also assemble ontology concepts to give a high abstract representation of UIs.

4 The Model Driven Development Process

We present an approach deriving User Interface (UI) of Rich Internet Applications from the combination of OWL2 ontology and IFML to automatically generate UI according to models specifications. Semantics of UI elements and their characteristics can be induced from the GUIs ontology's domain. However, IFML is responsible for capturing the interactions and actions related to concepts defined in the logical model of UIs. The model-driven process proposed was implemented using MDE tools of Eclipse Modeling Project.

This process is projected on a case study shown in (Fig. 2). It starts with abstract Models, in order to produce a flex Model as a target model. The choice of RIA as destination was not arbitrary because the design and implantation of GUI for RIAs is known for its complexity and difficulty in using existing tools. The chosen case study represents an interface containing a form for billing in e-commerce website. We will see later, how

Fig. 2. Case study: billing form.

the elements composing this interface are represented with their various types of interaction. The process is divided into three steps: The definition of PIM Models, the definition of PSM Model and M2M Transformation. Figure 3 shows the Model-driven process combining ODM and IFML to generate UIs of Rich Internet Applications.

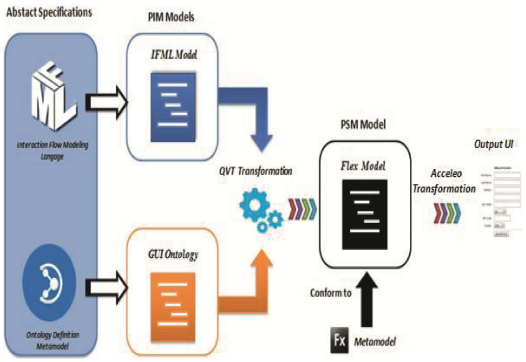

Fig. 3. Process overview.

4.1 PIM Models

In a first step, we define two PIM models: Logic model respecting the syntax of ODM metamodel, and Interaction model deriving from IFML metamodel. The two metamodels are defined with EMF (Eclipse Modeling Framework) in ecore format.

To define the logic model, we applied the analysis detailed in Sect. 2 to our case study. As Depicted in (Fig. 2). The interface comprises two containers; Window and Form which contain controls; six static fields, seven entry fields, two lists and a submit button. The representation of these elements in the logic model is defined as follows: containers and controls are represented by Individuals instanced from owl: Class distinguished by their data Properties as an example see (Fig. 4).

The second abstract model respects the syntax of IFML metamodel. We analyzed the IFML metamodel to select the required packages for an independent intermediate representation of interactions related to UIs. In our process, we will focus on the two packages: Core and extensions. These packages represent abstractly the structure of user interfaces, and dependencies between their elements in terms of interactions.

According to IFML metamodel, An IFML model is the top-level container of all the rest of the model elements. It contains an InteractionFlowModel, which contains all the elements of the user view of the application represented by the InteractionFlowModelElement. InteractionFlowModelElement has seven direct subtypes: InteractionFlowElement, InteractionFlow, Expression, The elements of an IFML model that are visible at the user interface level are called ViewElements, which are specialized in ViewContainers and ViewComponents. ViewContainers like windows, menus are containers of other ViewContainers or ViewComponents, while ViewComponents are elements of the interface that display content or accept input from the user. The extension package includes concrete examples of

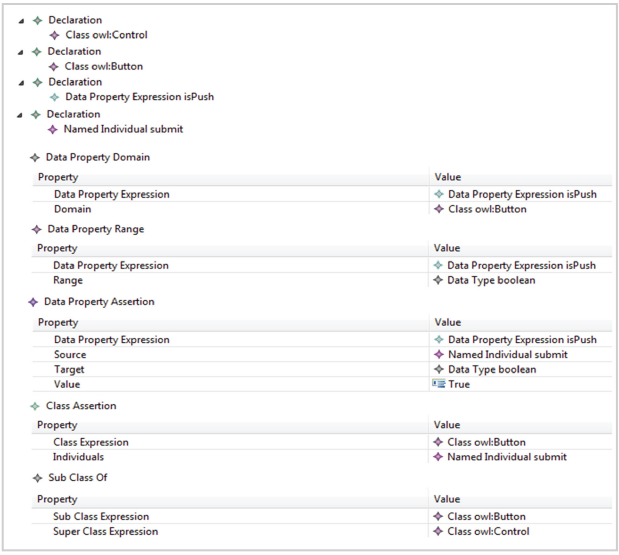

Fig. 4. Defining submit button with ODM.

ViewComponents such as List, Details and Form, and ViewComponentParts such as Fields and Slots [5].

In the example shown in (Fig. 2), a view container is tagged as «window» and marked as Default. The Form is composed of OnSubmit Event. The effect is represented by a navigation flow connecting the event associated with the OnSubmit Event. The navigation flow expresses a change of state of the user interface. The occurrence of the event causes a transition from a Form (source Interaction Flow element) to other windows (targets Interaction Flow elements). We associated the form with two Expressions; an activation Expression that denotes the condition that must be satisfied by the current interaction context for the event that triggers an action to become active and Interaction Flow Expression that determines which InteractionFlow is followed after an event occurrence. These expressions are expressed by javascript Language. The model instance is an abstract form to our case study respecting the IFML metamodel syntax [5]. Thus, with this two ODM and IFML models, we can easily have a UI target model in multiple platform as desktop, web or mobile.

4.2 Model to Model Transformation (PIM 2 PSM)

Once the Meta Modeling phase established, we defined the transformation rules. For this work, we used the QVT-Operational mappings language implemented by Eclipse modeling Framework [18]. The PIM, the most abstract level model is transformed into PSM Model. Since we have defined PIM models and PSM Metamodel for RIA, we define the Model To Model transformation using the QVTo standard respecting a defined algorithm. The entry metamodels are ODM and IFML metamdels, and the target one is Flex metamodel.

The entry point of the transformation is the main method. This method makes the correspondence between all elements of the IFML and Ontology models of the input models and the elements of type Flex output model.

For instance, each NamedIndividual which is an instance of "Window" class that has a dataProperty "isDefault", it would be mapped to BorderContainer. Form is obtained from "Form" Individual, and the NamedIndividuals of "Button" and "Field" and "List" classes, are mapped as Button, Text Input and List in Flex model. The data-properties (width, height, name, size, etc.) are mapped to properties figured in the Flex metaclasses. The (Fig. 5) below shows an excerpt of the Transformation program:

```
modeltype MyOWL uses "http://owl2/1.0";
modeltype MyIFMLCORE uses "http://www.omg.org/spec/IFML/core";
modeltype MyIFMLEXT uses "http://www.omg.org/spec/IFML/ext";
modeltype MyFlex uses "http://Flex/1.0";
//INPUT AND OUTPUT MODELS
transformation Transforms(in owl : MyOWL, core : MyIFMLCORE, out flex : MyFlex);
main() {
owl.objectsOfType(Ontology) -> map OntologyToFlexModel();
}
//MAPPING DES RACINES
mapping Ontology::OntologyToFlexModel() : FlexModel
{
bordercontainers += self.declaration.entity[NamedIndividual]->map
individual2borderContainer();
}
//MAPPING NAMEDINDIVIDUALS which have class window as class expression TO Border
Containers
mapping NamedIndividual:: individual2borderContainer() : BorderContainer

when {self.classAssertion.classExpression[Class].entityURI="owl:Window"->asBag();}
{
 name:=self.value;

self.dataPropertyAssertion->forEach(element)
      {
      if element.dataPropertyExpression.entityURI.toString().indexOf("width")-
```

Fig. 5. Query view transformation code excerpt.

4.3 PSM Model and Code Generation

This step describes the gradual refinement from a higher to a lower level of abstraction [19]. By applying the transformation rules mentioned before, each element figured in ODM and IFML models will be transformed to an element of Flex metamodel. It includes the generation of the target model in compliance with Flex metamodel. This model contains all the elements, properties, interactions collected from the two PIM models. This file is used to produce the necessary code by applying M2T transformation.

5 Conclusion and Future Work

With this paper, we have given a new approach MDE assembling two important abstract specifications defined by the OMG to derive UIs Rich Internet Apps. Our approach is based on the assumption that a UI for rich internet applications can be induced from IFML and ontologies. IFML allows representing abstractly the structure of user interfaces, and dependencies between its elements in terms of interactions. However, ODM

captures presentation features related to the UI and represent the basic concepts constituting a user interface. The major contribution in the proposed approach is the addition of the extension part to IFML and the definition of the GUI ontology that describe efficiently the graphical components of the application. Our challenge is to generate RIAs by using this Model Driven method without having to know all the technical specification of the execution platform.

Future works will cover the implementation of more refined code generator. Also, to obtain an enhanced result, this work can be extended to supplementary platforms like mobile Plateform starting from the same input models. Moreover, we can consider integrating other frameworks like JavaFX and GWT for Rich Internet Application.

References

1. Schmidt, D.C.: Guest editor's introduction: model-driven engineering. Computer **39**(2), 25–31 (2006)
2. OMG, MDA. Guide Version 1.0. 1. Object Management Group (2003)
3. Object Management Group: Ontology Definition Metamodel. Version 1.0, OMG (2009). http://www.omg.org/spec/ODM/1.0/
4. Paulheim, H., Probst, F.: A formal ontology on user interfaces yet another user interface description language. In: 2nd Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS) (2011)
5. Object Management Group: Interaction Flow Modeling Language. Version 1.0, IFML (2015). http://www.omg.org/spec/IFML/1.0/
6. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax W3C Editor's Draft 14 September 2009 This version: http://www.w3.org/2007/OWL/draft/ED-owl2-syntax-20090914/
7. Brambilla, M., Fraternali, P.: Interaction Flow Modeling Language: Model-driven UI Engineering of Web and Mobile Apps with IFML. Morgan Kaufmann, San Francisco (2014)
8. Pan, J.Z., Staab, S., Aßmann, U., Ebert, J., Zhao, Y.: Ontology-Driven Software Development. Springer Science & Business Media, Heidelberg (2012)
9. Paulheim, H., Probst, F.: Ontology-enhanced user interfaces: a survey. Semantic-Enabled Advancements on the Web: Applications Across Industries: Applications Across Industries, p. 214 (2012)
10. Paulheim, H.: Ontology-Based Application Integration. Springer Science & Business Media, New York (2011)
11. Gaševic, D., Djuric, D., Devedžic, V.: Model Driven Engineering and Ontology Development. Springer Science & Business Media, Heidelberg (2009)
12. Parreiras, F.S., Staab, S., Winter, A.: TwoUse: Integrating UML models and OWL ontologies. Inst. für Informatik (2007)
13. Brockmans, S., Haase, P., Hitzler, P., Studer, R.: A Metamodel and UML profile for rule-extended OWL DL ontologies. Springer, Heidelberg (2006)
14. Bumans, G.: Mapping between Relational Databases and OWL Ontologies: an example. Comput. Sci. Inf. Technol. **756**, 99–117 (2010). Scientific Papers, University of Latvia
15. Parreiras, F.S., Staab, S.: Using ontologies with UML class-based modeling: the TwoUse approach. Data Knowl. Eng. **69**(11), 1194–1207 (2010)
16. Canadas, J., Palma, J., Túnez, S.: Model-Driven Rich User Interface Generation from Ontologies for Data-Intensive Web Applications (2011)

17. Wysota, W.: Porting graphical user interfaces through ontology alignment. In: Ryżko, D., Rybiński, H., Gawrysiak, P., Kryszkiewicz, M. (eds.) Emerging Intelligent Technologies in Industry. SCI, vol. 369, pp. 91–104. Springer, Heidelberg (2011)
18. OMG, QVT. Meta Object Facility 2.0, Query/View/Transformation Specification, OMG (2008). http://www.omg.org/spec/QVT/1.0/PDF
19. Wagner, C.: Model-Driven Software Migration: A Methodology: Reengineering, Recovery and Modernization of Legacy Systems. Springer, Wiesbaden (2014)