Model-Driven Rich User Interface Generation from Ontologies for Data-Intensive Web Applications

Joaquín Cañadas[1], José Palma[2] and Samuel Túnez[1]

[1] Dept. of Languages and Computation. University of Almeria. Agrifood Campus of International Excellence, ceiA3. Spain
jjcanada@ual.es, stunez@ual.es
[2] Dept. of Information and Communications Engineering. University of Murcia. Spain
jtpalma@um.es

Abstract. Building data-intensive Web applications is a complex task widely explored during the last decade. Many approaches have been proposed, mainly based on conceptual models as well as on domain ontologies and knowledge models. This work describes a method for rich user interface 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. A tool supporting the ideas presented this paper has been developed.

Keywords: user interfaces, model-driven engineering, ontologies

1 Introduction

Data-intensive Web applications are Web-based information systems for accessing and maintaining large amounts of structured data, typically stored in database management systems [4]. Its development involves the definition of data models representing the information structure of the problem domain, as well as the design of user interfaces (UIs) to enable end-users to properly interact with the system when managing the collection of data, e.g. UIs for data presentation, data acquisition and data querying.

In general, the usage of data models for domain specification suffers of poor expressivity. To address this problem, ontologies are typically applied to domain modeling in which a conceptualization of a particular domain is given. Ontology formalisms such as OWL2 (Web Ontology Language 2) [22] are the backbone of Semantic Web and are growing in importance in software development [9]. Ontologies can describe the relevant concepts, relations and properties of an application domain adding assertions and constraints, increasing the amount of knowledge that can be represented by data models.

In this work we describe a model-driven method for deriving rich Web-based UIs for data management from domain models based on domain ontologies.

We apply Model-Driven Architecture (MDA) [15], or using the general concept, Model-Driven Engineering (MDE) [20], as the software development approach in which models are used as first class entities and transformations between models and from models to code can be defined and executed. MDA/MDE is currently being applied in many domains, such Web Engineering [14], Ontology Engineering [8], and UI development [11].

Recent technologies for improving end-user experience in Web 2.0 include the so called Rich Internet Applications (RIAs) [7] which provide advanced and more interactive UIs, similar to desktop applications, while minimizing network traffic overhead and increasing user usability and efficiency [23]. In our approach, the UI derived from OWL2 is based on frameworks of reusable components for RIAs. To enrich the UI generation, a presentation model tightly coupled to the domain ontology provides modeling support to customize presentation features of UIs for data-intensive Web applications. For practical implementation, two different Java-based frameworks for RIAs have been considered.

A tool supporting the method presented has been developed using MDE tools provided by the Eclipse Modeling Project[3], and it is supported by the TwoUse Toolkit [17] for ontology creation and management.

The rest of this paper is organized as follows: Section 2 reviews related work. Next, Section 3 introduces RIAs frameworks for Web-based UI implementation. The presentation model and the proposed mapping from OWL2 to UI components are detailed in Section 4. Finally, main conclusions and future work are summarized.

2 Related work

There have been many earlier approaches on UI generation based on models and MDE. We can distinguish between those from the field of Web Engineering and those from the field of Human-Computer Interaction (HCI).

Web Engineering approaches the design and development of Web applications based on conceptual models [3], focusing on content, navigation and presentation models as the most relevant concerns in Web applications design [19]. Based on these models, the full Web application can be developed applying a model-driven approach, including the presentation layer composed of web pages, web forms, links, and so on.

Web Engineering offer rather mature and established methodologies for traditional Web applications, and the UI layer has been explicitly addressed in most approaches. But when we move to Semantic Web information systems, methodologies are still in a development phase [1]. Examples like SHDM [12], Hera [21] and WebML+Sem [1] offer a wide support for ontology languages, basically RDF (Resource Description Framework) and OWL, and focus on different semantic web technologies such as semantic model description, advanced query support, flexible integration, ontology reasoning, and more, but leaving UI aside. In this paper we address that open issue.

[3] http://www.eclipse.org/modeling/

In HCI field a number of model-driven approaches for UI development have also arisen [18]. They are commonly based on models created with extensions of UML (Unified Modeling Language) for UIs modeling [5], but can also use textual formats based on XML (eXtensible Markup Language), as is widely explored in [11].

Among these approaches, the PEGASUS method [13] presents an effort to supply end-users with mechanisms for authoring Web-based applications using ontologies to specify knowledge for building data models together with presentation models. Moreover, it enables the generation of a Web UI from the ontology in basic HTML (HyperText Markup Language) and JSP (JavaServer Pages) code. In our approach we also use ontologies as domain models and a presentation model, but we focus on current rich Web UI generation using modern components of RIA frameworks which are richer in functionality than basic HTML and JSP.

In a recent work [6], the same authors provide a way of modeling UIs based on semantic models of domain problem, deriving a Web application for displaying content. The method is based on document transformation through a set of XSLT (Extensible Stylesheet Language Transformations) applied to XML files to generate documents for the UI. The result can incorporate AJAX (Asynchronous JavaScript And XML) components to have a better interactive result. Our proposal has similarities with this approach in sense that we also focus UI development with rich AJAX components, but has important differences with respect to the model-driven approach applied since we use MDE transformation languages and tools for defining the approach instead of XSLT transformations, as well as we derive a UI not only for displaying content but also for content acquisition and querying.

3 Frameworks for RIAs in JavaEE

Modern UI development requires the usage of extensive software libraries and frameworks, and code becomes rather platform-specific [11]. In this work we focus on Java Platform Enterprise Edition (JavaEE[4]) technology and the JavaServer Faces framework (JSF) [10] as implementation platform. To fit once of the most important characteristics of RIAs, a richer interaction is achieved adding AJAX technology to provide improved user experience. Several rich UI frameworks for JSF applications are available, some of them are well established such as RichFaces[5] (JBoss project), ICEfaces[6] (ICEsoft project), MyFaces[7] (Apache project), ADF Faces[8] (Oracle project) and Google Web Toolkit[9] (Google project). In this

[4] http://java.sun.com/javaee
[5] http://www.jboss.org/richfaces
[6] http://www.icefaces.org/
[7] http://myfaces.apache.org/
[8] http://www.oracle.com/technetwork/developer-tools/adf/
[9] http://code.google.com/webtoolkit/

work, only RichFaces and ICEfaces have been considered, although the approach can easily adapted to other frameworks.

JBoss RichFaces is an advanced JSF based framework that provides a complete range of rich AJAX enabled UI components. RichFaces is made up of two component tag libraries: *a4j:* represents core AJAX functionality, and *rich:* represents self contained and advanced components such as calendars, datatables, trees and more (see the RichFaces showcase[10] for details). Current version, RichFaces 4.0, can be used in any container compatible with JSF 2.0.

ICEsoft ICEfaces is an integrated AJAX application framework that enables JavaEE application developers to easily create and deploy thin-client RIAs in Java. ICEfaces 2 is the current version of the open-source framework based on the JSF 2.0 standard. It offers a vast set of rich components included in the *ice:* tag library, to create rich advanced UIs (see the ICEfaces showcase[11] for details).

4 Model-driven rich user interface generation

4.1 Process overview

Our approach is based on the assumption that a UI can be induced from the ontology classes, properties and assertions. Since OWL2 semantics is richer than semantics of UI components, only a part of OWL2 can be represented in the UI and supported by the proposed mapping.

Fig. 1 shows the model-driven schema proposed for deriving rich Web UI from OWL2 ontologies. The process starts with the specification of an OWL2 ontology of the problem domain. In a first step, a presentation model with default presentation values is derived from the ontology applying a model-to-model (M2M) transformation. Developers can customize this presentation model to drive a better UI generation. Later, a model-to-text (M2T) transformation produces the final code.

The model-driven process proposed was implemented using MDE tools of Eclipse Modeling Project, and it is supported by the TwoUse Toolkit[12] for OWL2 authoring and management. The TwoUse Toolkit is a free, open source tool bridging the gap between Semantic Web and MDE, that supports OWL2 authoring based on the Ontology Definition Metamodel (ODM) [16]. In this environment, metamodels are defined with EMF[13] (Eclipse Modeling Framework) in ecore format. The ODM metamodel is provided by TwoUse whereas the Presentation metamodel has been designed by the authors of this work.

M2M transformation is designed with ATL[14] (Atlas Transformation Language), where as final code is generated by a M2T transformation implemented in JET[15] (Java Emitter Templates). As a result, the code for the rich Web UI

[10] http://richfaces.org/showcase
[11] http://component-showcase.icefaces.org
[12] http://code.google.com/p/twouse/
[13] http://www.eclipse.org/modeling/emf/
[14] http://www.eclipse.org/m2m/atl/
[15] http://www.eclipse.org/modeling/m2t/?project=jet

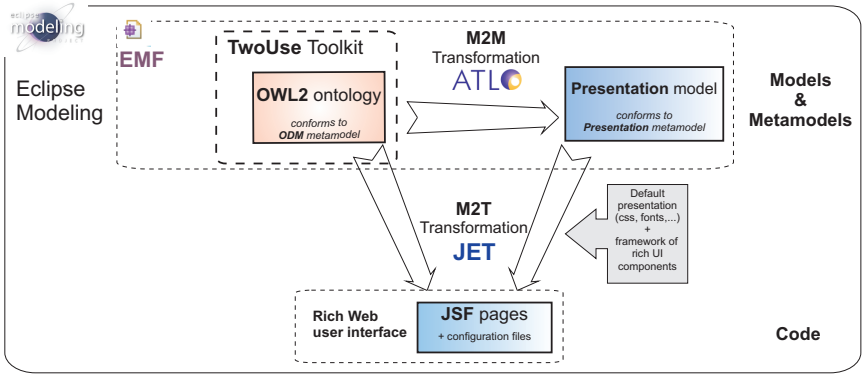


Fig. 1. MDE schema for rich Web user interface generation

is obtained as a set of JSF web pages based on the selected framework of rich components, RichFaces or ICEfaces. Default configuration is injected to provide presentation templates for pages, styles and css files.

4.2 The Presentation metamodel

To drive a powerful UI generation, the Presentation model captures features of UI components used in data-intensive applications, for example, in what other they are to appear, their visual appearance and layout, and more. Basically we propose three main types of presentation elements in the UI for data-intensive Web applications:

- a *menu*, including a hierarchy or tree with the class taxonomy
- a *list* page per ontology class, listing all instances of the class,
- a *form* page per class for viewing and editing instances of the class.

Fig. 2 shows the Presentation metamodel using a simplified UML class diagram notation. It defines the primitives that can be used in the modeling language, that is, in presentation models. Metaclasses in this metamodel are designed to allow the extension of presentation models by adding new features or modifying existing ones, enabling the process to evolve. For that purpose, a *Class* is related to a *MenuItem* which stores the feature(s) for displaying the class in the menu page, and to a *TableList* which stores the feature(s) for displaying the class in the list page. Similarly, a *Property* is related to a *FormField* which stores the features for displaying the property as a field in the form page, and to a *ListColumn* which stores the features for displaying the property as a column in the list page.

Tables 1 and 2 describe the main metaclasses and attributes that can be specified in a presentation model. How they are used in the generation of UI elements is explained in following section.

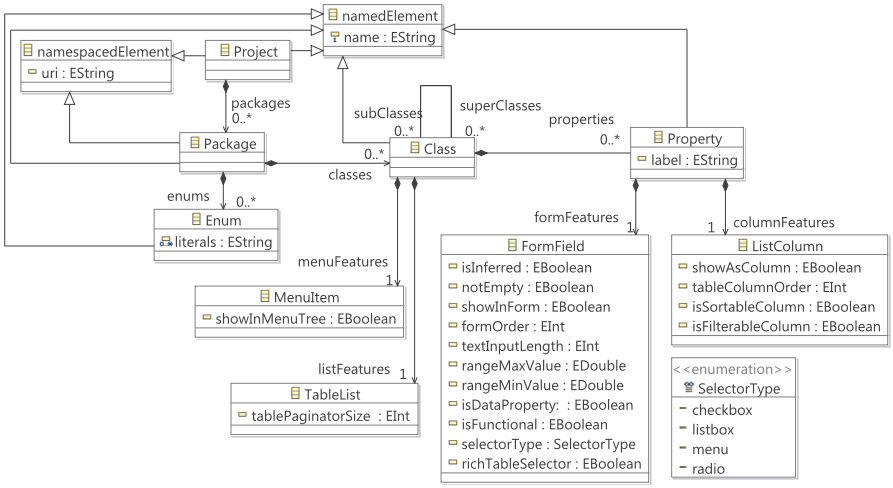


Fig. 2. Presentation metamodel

Table 1. Presentation metamodel: Class features

Metaclass	*Feature*	*Type*	*Used in Page*	*Description*	*Default value*
`Class`	*name*	`string`	all	name of the class	OWL2
	subClasses	`Class[0..*]`	menu	collection of subClasses	OWL2
	superClasses	`Class[0..*]`	menu	collection of superClasses	OWL2
	properties	`Property[0..*]`	form, list	collection of properties having the class as domain	OWL2
	menuFeatures	`MenuItem`	menu	link to presentation features for a class in the menu page	
	listFeatures	`TableList`	list	link to presentation features for a class in the list page	
`MenuItem`	*showInMenuTree*	`boolean`	menu	true if the class is showed in the menu (tree)	true
`TableList`	*tablePaginatorSize*	`integer`	list	rows per page in the list	10

4.3 Mapping OWL2 to user interface components

This section describes how the menu tree, list and form pages are derived.

Menu tree. Domain concepts are represented in OWL2 as named classes, which can have subclasses, conforming a hierarchy of classes. To display such a hierarchy a tree is commonly used. For that purpose, RichFaces provides a *rich:tree* component which renders a tree control on the page. Similarly, ICEfaces has the *ice:tree* component that displays hierarchical data as a tree of branches and leaf nodes. Only classes with the presentation feature *showInMenuTree* true-valued are shown in the tree.

Form page. For displaying and editing instances of a particular class, a rich form with the properties of the class is generated. In OWL2 each property has a domain and a range, and two types of properties are distinguished: datatype properties, relations between instances of classes and primitive data

Table 2. Presentation metamodel: Property features

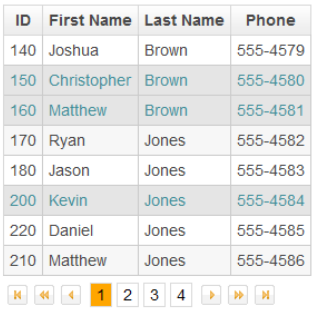
Metaclass	*Feature*	*Type*	*Used in Page*	*Description*	*Default value*
Property	*name*	string	form, list	name of the property	OWL2
	label	boolean	form, list	text to show in the property field (form) or the table column (list)	prop. name
	fromFeatures	FormField	form	link to presentation features for a property in a form field	
	columnFeatures	ListColumn	list	link to presentation features for a property as a column of the table list	
FormField	*isInferred*	boolean	form	true if the value is not editable	false
	notEmpty	boolean	form	true if the value can not be empty	true
	showInForm	boolean	form	true if the form includes a field for the property	true
	formOrder	integer	form	field order in the form	null
	textInputLegth	integer	form	field size for the property value	30
	rangeMaxValue	real	form	maximum value allowed for number type properties	null
	rangeMinValue	real	form	minimum value allowed for number type properties	null
	isDataProperty	boolean	form	true if the property type is primitive, false otherwise (object property)	OWL2
	isFunctional	boolean	form	true if the property cardinality is as much one, false otherwise	OWL2
	selectorType	SelectorType	form	for object properties, kind of selector showing related instances	menu
	richTableSelector	boolean	form	for object properties, use a rich datatable for selecting related instances	false
ListColumn	*showAsColumn*	boolean	list	true if the property is showed as a column in the list	true
	tableColumnOrder	int	list	column order of the property in the table	null
	isSortableColumn	boolean	list	true if the table can be sorted by column values	true
	isFilterableColumn	boolean	list	true if the table can be filtered by column values	false

types (e.g. integer or string); and object properties, relations between instances of two classes. The default behavior in OWL determines that a property can relate an instance of the domain to multiple instances of the range, but defining the property as *functional* the relation is from an individual to only one primitive value (functional datatype property) or individual (functional object property). Similarly, a cardinality constraint can set the property range to 1. The form page for a particular class contains fields for all the properties having the class as domain and with a true value in the presentation feature *showInForm*.

The mapping of object properties to the UI implies that for non-functional properties, common selectors are used, such as *selectManyCheckbox*, *selectManyListbox* and *selectManyMenu* widgets, whereas for functional properties, *selectOneListbox*, *selectOneMenu* and *selectOneRadio* widgets can be used.

The widget used for an object property is set in the presentation feature *selectorType*. In case that a property has a true value in the presentation feature *isInfered*, a selector widget is not generated for it because its value can not be modified by users.

When an object property has the feature *richTableSelector* to true, then the values to be selected are shown in tables instead of in selectors. RichFaces provides the *rich:extendedDataTable* component for a powerful selection of one or many items. Similarly, ICEfaces includes the *ice:rowSelector* tag in the *ice:dataTable* component to provide that functionality. Fig. 3 shows an example of multiple data table selection.



ID	First Name	Last Name	Phone
140	Joshua	Brown	555-4579
150	Christopher	Brown	555-4580
160	Matthew	Brown	555-4581
170	Ryan	Jones	555-4582
180	Jason	Jones	555-4583
200	Kevin	Jones	555-4584
220	Daniel	Jones	555-4585
210	Matthew	Jones	555-4586

1 2 3 4

Fig. 3. Multiple data table selection (3 rows selected)

To fulfill the list of values to be selected, individuals of the range type are listed. When an object property has the same class as domain and range types, then reflexive or irreflexive property axioms must be considered. In a irreflexive object property, the self individual (domain) is not included in the list of values to be selected (range), whereas in a reflexive one, all individuals are listed.

OWL2 provides several class extension constructs to define unnamed anonymous classes. The "oneOf" expression enables the definition of an enumerated class through the list of individuals that constitute the instances of the class. When the range of a property is an enumerated defined through a "oneOf" anonymous class, the enumeration literals obtained from the individuals linked to the enumeration are used as the list of allowed-values that can be selected. An example is the *hasSex* functional property from the *Person* class to the enumerated class with values {*female, male*}. In this case, one of the three *selectOne* selectors can be used in the interface.

It is possible to further constrain the range of a property with property restrictions. The *"has-Value"* restriction specifies an anonymous class based on the existence of particular property value. Other classes can be a subclasses of such a property restriction. As example, the *Woman* class is a subclass of *"hasSex has female"* property restriction, and similarly, the *Man* class is subclass

of *"hasSex has male"*. In the UI, this is mapped to default values that can not be editable by end-users.

List page. Finally, for listing all the instances or individuals of each named class, a page with a rich data table component is generated. Table columns are those properties of the class with a true value in the *showAsColumn* feature of the presentation model, and the column position in the table is established by the *tableColumnOrder* value. Both RichFaces and ICEfaces provide a *datatable* component with advanced functionality including a *paginator* widget for viewing the table as multiple pages of rows instead of as one large table, a *sortColumn* feature allowing the user to sort of data in the table, and *filterValue* feature for filtering data rows (only available in RichFaces). The corresponding features in the presentation model allow to customize these elements.

5 Conclusions and future work

In this work a model-driven method for generating rich Web UIs from OWL2 domain ontologies was presented, continuing a research focused on model-driven development of Web applications from ontologies and rules [2]. Our approach is based on the assumption that a UI for a data-intensive application can be induced from the domain ontology classes, properties and axioms. To obtain an enhanced result, a presentation model captures presentation features related to the UI. Since UI development is platform-specific task, JavaEE and JSF technologies for Web application development were chosen as target implementation in our research. Two frameworks of rich UI components were considered, although the approach can be extended to other frameworks. The proposal is tested with a proof of the concept tool.

The extension of the proposal to cover a larger set of OWL2 elements is considered as future work, as well as enhanced UI functionality to provide full Semantic Web information system generation from ontologies. Enriching the ontology with SWRL rules and analyzing how rules can affect to the UI is also considered as future work, focusing on how rules can provide UI adaptivity.

Acknowledgments. The authors wish to thank the Spanish Ministry of Education and Science for funding received under projects TIN2009-14372-C03-01 and PET2007-0033, and the Andalusian Regional Government under project P06-TIC-02411.

References

1. Brambilla, M., Facca, F.M.: Building semantic web portals with WebML. In: Web Engineering, 7th International Conference, ICWE 2007, Como, Italy. Lecture Notes in Computer Science, vol. 4607, pp. 312–327. Springer (2007)
2. Cañadas, J., Palma, J., Túnez, S.: A MDD approach for generating rule-based web applications from OWL and SWRL. In: 3rd Workshop on Transforming and Weaving Ontologies in Model Driven Engineering (TWOMDE 2010). vol. 604. CEUR Workshop Proceedings, Málaga, Spain (June 2010)

3. Ceri, S., Fraternali, P., Matera, M.: Conceptual modeling of data-intensive web applications. Internet Computing, IEEE 6(4), 20–30 (2002)
4. Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, 1 edn. (Dec 2002)
5. Cerny, T., Song, E.: A profile approach to using UML models for rich form generation. In: Information Science and Applications (ICISA), 2010 International Conference on. pp. 1–8 (2010)
6. Chavarriaga, E., Macías, J.A.: A model-driven approach to building modern semantic web-based user interfaces. Advances in Engineering Software 40(12), 1329–1334 (2009)
7. Driver, M., Valdes, R., Phifer, G.: Rich internet applications are the next evolution of the web. Tech. rep., Gartner Research Note. G (2005)
8. Gašević, D., Djurić, D., Devedžić, V.: Model Driven Architecture and Ontology Development. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
9. Gašević, D., Kaviani, N., Milanović, M.: Ontologies and software engineering. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 593–615. Springer Berlin Heidelberg (2009)
10. Geary, D., Horstmann, C.S.: Core JavaServer Faces. Prentice Hall, 2 edn. (2007)
11. Hussmann, H., Meixner, G., Zuehlke, D.: Model-Driven Development of Advanced User Interfaces. Springer-Verlag New York Inc (2011)
12. Lima, F., Schwabe, D.: Application modeling for the semantic web. In: LA-WEB. pp. 93–102. IEEE Computer Society (2003)
13. Macías, J.A., Castells, P.: Providing end-user facilities to simplify ontology-driven web application authoring. Interacting with Computers 19(4), 563–585 (2007)
14. Moreno, N., Romero, J.R., Vallecillo, A.: An overview of Model-Driven Web Engineering and the MDA. In: Web Engineering: Modelling and Implementing Web Applications, pp. 353–382. Springer London (2008)
15. Object Management Group: MDA Guide Version 1.0.1. OMG document: omg/2003-06-01 (2003)
16. Object Management Group: Ontology Definition Metamodel. Version 1.0. OMG (2009), available at http://www.omg.org/spec/ODM/1.0/
17. Parreiras, F.S., Staab, S.: Using ontologies with UML class-based modeling: The TwoUse approach. Data & Knowledge Engineering 69(11), 1194–1207 (Nov 2010)
18. Pérez-Medina, J.L., Dupuy-Chessa, S., Front, A.: A survey of model driven engineering tools for user interface design. In: Task Models and Diagrams for User Interface Design, 6th International Workshop, TAMODIA 2007, Toulouse, France. Lecture Notes in Computer Science, vol. 4849, pp. 84–97. Springer (2007)
19. Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.): Web Engineering: Modelling and Implementing Web Applications. Human-Computer Interaction Series, Springer London, London (2008)
20. Schmidt, D.C.: Guest Editor's Introduction: Model-Driven Engineering. Computer 39(2), 25–31 (2006)
21. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. J. Web Engineering 2(1-2), 3–26 (2003)
22. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (2009), available at http://www.w3.org/TR/owl2-overview/
23. Wright, J.M., Dietrich, J.: Requirements for rich internet application design methodologies. In: Web Information Systems Engineering - WISE 2008, 9th International Conference, Auckland, New Zealand. Lecture Notes in Computer Science, vol. 5175, pp. 106–119. Springer (2008)