A MDE Framework for semi-automatic development of Web applications

Javier Solis, Hasdai Pacheco, Karen Najera, Hugo Estrada
Fondo de Informacion y Documentacion para la Industria INFOTEC
Av. San Fernando 37, Col. Toriello Guerra, Mexico D.F, Mexico
{*javier.solis, ebenezer.sanchez, karen.najera, hugo.estrada*}*@infotec.com.mx*

Keywords: Semantic Web applications:Ontologies:Ontology-Driven Development

Abstract: One of the current trends in computer science is the use of ontologies in Software Engineering. In this context, several academic and industrial works have used ontologies as mechanism for system requirements representation and automatic code generation. However, most of current works propose generic development platforms regardless the reuse of components for a specific domain. In this paper, a Model-Driven Development Framework, called SemanticWebBuilder (SWB) is presented. SWB provides an agile development platform for the Web application domain, where system requirements are modeled through ontologies and from this knowledge representation, the infrastructure of the system is automatically generated. The resultant system can be extended by reusing of code, thus allowing to build complex systems in a short time. Moreover, system data is represented as RDF triples, making this data available in the Semantic Web. SWB has been widely used to develop Web applications for several government dependencies in Mexico.

1 INTRODUCTION

Currently, there is a consensus, in academia and industry, about the importance of analysis and design phases in developing large scale software systems. In this context, most of current software development environments are based on traditional UML analysis and design techniques, and on the use of Domain Specific Languages for modeling system requirements. This approach, even when it gives partial solution to the automatic code generation following the Model-Driven Development paradigm, has some issues that need to be addressed. For instance: the need to develop several models to specify system requirements, such as, architecture, behavior and data model; the static structure of the data model; and the mechanism to store system information. To solve this, other mechanisms to model a system have been proposed. One of the most promising trends in this are area is the use of ontologies.

In this context, research works have proposed the use of ontologies to drive the process of software applications development, since ontologies brings advantages to the software development cycle (Happel and Seedorf, 2006) enabling analysts to better face the evolution of systems. However, most of the current works in this area have proposed generic software development platforms which do not consider the reuse of components for a specific domain.

In this paper, a Model-Driven Development Framework called SemanticWebBuilder (SWB) is presented. SWB provides an agile development platform for Web applications. It uses domain knowledge as a starting point by modeling system requirements through ontologies. Then, the infrastructure of the system (in object-oriented source code) is automatically generated from ontologies. The resultant infrastructure contains the basic functionality for the system, however, it could be extended to add specific functionality by reusing of code, thus allowing to build complex systems in a short time. SWB aims the automatic generation of interoperable software systems to be feasible for several application domains by representing system and business information in a common format, that is as RDF triples. Moreover, SWB provides the mechanisms to expose business information in the Semantic Web. SWB has been widely used in practice to develop real semantic applications for several government dependencies in Mexico.

The paper is structured as follows. Section 2 presents the definition of ontologies in Computer Science. Section 3 describes the motivation of using ontologies as models in our proposal. Section 4 explains the SWB Framework. Section 5 presents the ontology driven development process. Section 6 describes related works. And finally, section 7 presents conclusions and future works.

2 ONTOLOGIES

The term *ontology* is defined as 'an explicit representation of conceptualization' (Gruber, 1993), as well as 'a formal specification of a shared conceptualization' (Borst, 1997). '*Conceptualization* refers to an abstract model which represents the relevant concepts of some phenomenon in the world; *Explicit* means that the type of concepts used, and the constraints on their use are explicitly defined; *Formal* indicates that the ontology should be machine readable. And *Shared* denotes that an ontology captures consensual knowledge accepted by a group' (Studer et al., 1998).

3 MOTIVATION

We aim to provide a Framework based on the MDE approach to automatically generate Semantic Web applications by exploiting ontologies. Following we describe our motivation for using ontologies and the Web application domain. Model-Driven Engineering (MDE) is an approach to software development in which abstract models of software systems are created and systematically transformed to concrete implementations. In this context, modeling languages such as the Unified Modeling Language (UML) and Domain Specific Languages (DSL) have been proposed to represent these abstract models. However, to cover the complete software specification, these modeling languages use of several models, for instance, to represent system structure, behavior, data model, etc. The use of ontologies as models in MDE allows, rather than UML and DSLs, to represent system structure, behavior and data model in a single ontology, together with business logic and software restrictions that would be commonly hard coded in most current systems. Furthermore, ontologies bring the following advantages: they provide a shared understanding of the problem domain and formal specification of system requirements, allow reuse of existing domain knowledge definitions, guide automatic generation of source code through an API that maps concepts of the ontology to classes in an object-oriented language, provide better support for logical inference, integration and interoperability with other components or applications and improve maintenance and update tasks because the source code could be automatically re-generated from ontology definitions if requirements change (Uschold, 2008; Happel and Seedorf, 2006).

On the other hand, the engineering of Web applications is a fairly mature field, however, although there is an abundance of libraries and Frameworks supporting the construction of Web applications, current works do not consider directly the tendencies of the Web domain, such as Semantic Web and Linked Data approaches. In order to provide a Framework that takes into account this tendencies, we propose the use of ontologies as models in the MDE approach to enable Semantic Web application development. In this way, business data is represented as Semantic Web data, that is as RDF triples, therefore, it can be exposed in the Semantic Web.

4 SWB FRAMEWORK

We present our Model-Driven Development Framework called SemanticWebBuilder (SWB)[1]. SWB provides semi-automatic system development. The starting point of the development process is the representation of the system requirements of a particular domain in terms of an ontology. Then, transformations rules are provided in order to obtain, in an automatic way, object-oriented source code according to concepts and properties defined in the ontology. As an output, the SWB Framework provides the infrastructure of the domain system which manages the persistence of system and business information as RDF triples. Although SWB supports the development of different domain systems, it is specialized in the Web application domain. Therefore, an ontology with Web applications requirements has been developed and specific components capable to manage Web applications features have been integrated to the SWB Framework.

Following, we describe the main components of the SWB Framework (Fig. 1):

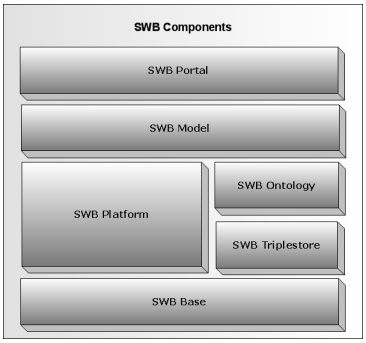

Figure 1: SWB Framework components.

SWB Ontology. A predefined OWL ontology called SWB ontology is provided by the Framework. It has been described by the Web Ontology Language OWL[2]. The SWB ontology contains a base structure

[1] SemanticWebBuilder website. http://www.semanticwebbuilder.org.mx

[2] OWL Reference. http://www.w3.org/TR/owl-ref/

with predefined classes and properties for the transformation from the ontology to object-oriented source code. The classes are OWL class extensions, i.e., subclasses of owl:Class. For instance, swb:Class, to map ontology concepts to Java classes and swb:Interface, to map shared behaviors as Java interfaces; SWB ontology accelerates the development of specific domain ontologies by reusing its base structure. For the Web applications domain, it has been extended with classes to describe Web components and Web sites features.

SWBBase. SWBBase corresponds to a set of libraries implementing generic code utilities, such as string formatting, e-mail sending, database query management, error handling, file manager, etc.

SWBPlatform. It is the main component of the SWB Framework. SWBPlatform implements a Model-Driven Development platform which generates object-oriented source code in the Java language. Its main elements are: the *Application layer* and the *Code Generator*.

The *Application layer* corresponds to the Java representation of RDF and OWL concepts. RDF and OWL concepts (resource, class, property, literal, etc.) has been mapped in a set of of Java classes as is shown in the first two columns of Table 1. Each Java class has defined a set of methods that allow to represent some of the relations considered in RDF and OWL such as: subclass relation (isSubClassOf) and subProperty relation (isSubPropertyOf). These relations are listed in Table 2.

The *Code generator* makes use of the the SWB ontology and the *Application layer* to carry out the automatic code generation. It consist of transformation rules and a transformation engine. By running the transformation engine, the *Code generator* classifies each concept and property defined in the SWB ontology to generate its corresponding Java implementation according to the transformation rules. For instance: each subclass of swb:SWBClass is mapped as a Java class where each class contains a set of accessors methods (getters and setters) to manipulate the bounded properties defined for the SWBClass concept in the ontology; RDF properties or literals are mapped as Java properties or literals as presented in the last column in Table 1; and, each subclass of swb:SWBInterface is mapped as a Java interface since swb:SWBInterface represents specific behaviors that require to be inherited by more than a concept. After running the *Code generator*, two Java layers are generated: Java Base layer and Java Extended layer. The Java Base layer implements the wiring among classes, properties and interfaces defined in the ontology with their supporting Java objects. Classes in this layer must not be modified, due they are automatically regenerated accordingly to the SWB ontology if code generator is re-executed. The Java Extended layer allows developers to implement specific functionality required by the system, such as specific objects behavior.

Fig. 2 represents the transformation from the SWB ontology to Java source code according the MDE approach. On the left side is situated the OWL metamodel in the M2 layer and the SWB ontology in the M1 layer. On the right side is situated the *Application layer* at the same level than the OWL metamodel (M2) since it represents OWL language in Java code. The transformation rules have been defined in this layer. The transformation engine applies the transformation rules in the layer M1, transforming the SWB ontology into Java code.

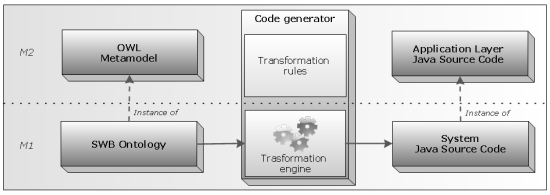

Figure 2: Code generation through the MDE approach.

Additionally, SWBPlatform contains classes that implement the communication mechanism with different triple stores and encapsulates the complexity associated with the direct manipulation of RDF data.

SWBModel. It is a set of classes automatically generated by the SWBPlatform for the Web applications domain. It contains the implementation of Web page, Web site, User and Role concepts. The classes provide methods to manipulate RDF data in a high level of abstraction through Semantic managers, Semantic Vocabularies and native Java objects.

SWBTripleStore. It is a set of classes to encapsulate methods for RDF management with the Jena API. The classes provide features to improve the performance when accessing triple stores in several database managers through different built-in connectors.

SWBPortal. It is a set of classes to develop Web applications for a Java application server. It implements a set of embeddable Web components, an HTML template processor, a request distributor and utilities for user session management, data validation, security issues, a content management system for testing components, an extension point for the rapid development of new Web components, etc.

The Ontology-Driven Development process through the SWB Framework and a descriptive example are presented in following sections.

Table 1: Application layer.

RDF/OWL Concept	Application layer object	Java Concept
RDF Resource	Semantic object	Object instance
OWL Class	Semantic class	class
RDF Property	Semantic property	property
RDF Literal	Semantic literal	literal

Table 2: OWL relationships representation.

Application layer object	Relationship	Application layer object
Semantic class	isSubClassOf	Semantic class
Semantic class	hasProperty	Semantic property
Semantic property	isSubPropertyOf	Semantic property
Semantic object	hasDataTypeProperty	Semantic class
Semantic object	hasObjectProperty	Semantic object

5 DEVELOPMENT PROCESS

We present the Ontology-Driven Development process which automatically generates the source code of Web applications and extensions to existing ones by exploiting the SWB Framework. The process consists of three steps:

1. Problem conceptualization: Problem requirements, such as: concepts and interactions, properties, and behaviors of each concept needed in the problem domain are modeled by means of OWL ontologies.
2. Automatic source code generation: By running the *Code generator*, each element in the ontology is transformed into Java code according to the transformation rules. After the code generation, developers have these elements available: a) Java classes for all defined concepts in the ontology, b) methods for objects communication as defined in the ontology properties and relations, including: association, aggregation, inheritance and composition, c) methods for object persistence in different triple-stores and databases, d) view modes to display and manipulate properties of each object instance and e) cache system of semantic objects for system and data access performance.
3. Specific functionality implementation: Additional Java source code could be developed to accomplish specific functionalities required by the system, such as, specific behavior of each object, data validation or user interface implementation.

SWB provides a software requirement traceability mechanism which ensures that the model is consistent with the resultant source code. Changes in the model affect the implementation directly, however, extensions to the implementation do not affect the model. This allows to easily modify and to previously extend developed functionality, to reduce code maintenance and to increase reuse of code.

Following we describe an example of the development process where a Web application is extended with a specific functionality.

5.1 Extending a Web application

Web applications generated through the SWB Framework can be extended with several specific functionalities depending on the domain of the Web application, for instance, tourism, economics, health, etc.

In order to illustrate the development process a basic example is presented. The example is related to the development of an online contact directory. The contact directory manages personal contacts as part of a social Web site. It allows to add, edit and delete contacts. Each contact entry holds the name, address, phone and email. The contact directory should provide four views: a view with the list of all registered contacts, a view to visualize detailed information of each contact; a view to add new contacts, and a view to modify the information of a contact.

The application of the proposed development process for the contact directory is presented below.

Problem conceptualization. The conceptualization consists of modeling the problem domain to extract important entities and relations. In this way, a first understanding of the knowledge implicit in the problem domain is obtained. The first analysis of the contact directory leads to the creation of the draft diagram of Fig. 3. By using the draft diagram and knowledge of domain experts, we can start with the contact directory ontology definition by reusing and extending the basic structure of the SWB ontology. Therefore, concepts, properties, relations, object behavior and restrictions related to the contact directory are modeled in the SWB ontology via an ontology editor such

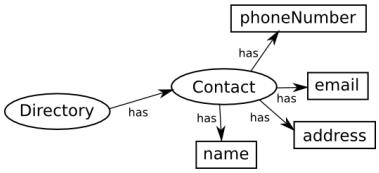

Figure 3: Contact directory conceptualization.

as Protege[3] or TopBraid[4]. Properties to display and to format the contact directory are added to let the SWBPlatform automatically build the necessary Web forms for the manipulation of the contact directory in the Web application. The Contact directory ontology is described in Fig. 4.

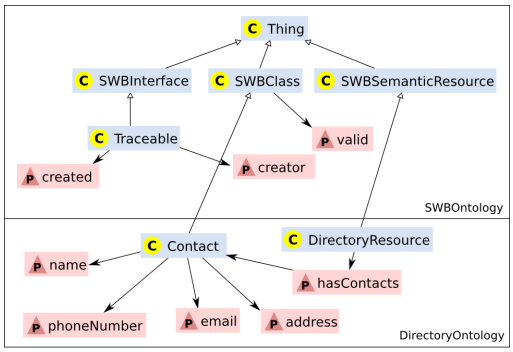

Figure 4: Contact directory ontology.

Code generation. A script is created for code generation through SWBPlatform. The script applies the transformation rules embedded in the *Code generator* and, from de SWB ontology, it generates the base source code for the contact directory. This script allows to regenerate the source code if the ontology changes. The generated source code represents a high level domain Java API that hides all complexity of RDF data management (Fig. 5). The contact directory is represented as the DirectoryResource class. It contains all the methods needed to implement the contact directory and its behavior using SWBPortal.

Specific functionality implementation. An 'All contacts' view has to be implemented to accomplish the contact directory requirements. The development is carried out in the Java Extended layer by using the methods and properties encapsulated in the Java Base layer. Each mode and action is created through the SWBPortal to manage http requests, render HTML code for the user view and to achieve user interaction. Object persistence and retrieval is done through SWBPlatform, which manages all RDF data.

[3]http://protege.stanford.edu/

[4]http://www.topquadrant.com/products/TB_Composer.html

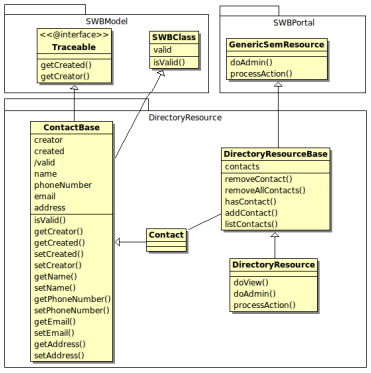

Figure 5: Contact directory classes.

The final contact directory source code is incorporated to the Web application source code by the content management system provided by SWBPortal. Fig. 6 shows the 'All contacts' view and Fig. 7 the 'Add contacts' view.

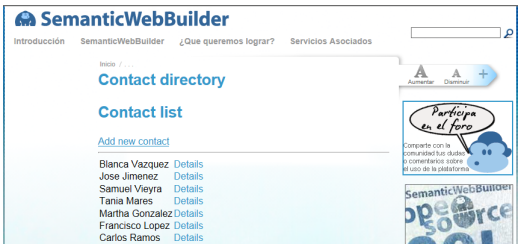

Figure 6: 'Contact list' view.

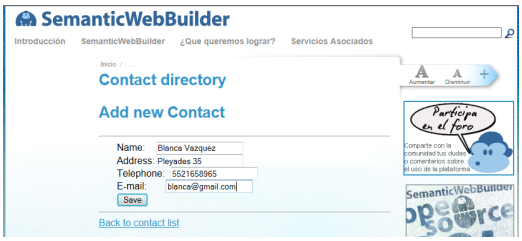

Figure 7: 'Add new contact' view.

6 RELATED WORKS

Currently, there is a growing trend, in academia and industry, of works that propose the use of MDE for software development. General purpose development tools such as AndroMDA (Bohlen et al., 2012) and Acceleo (Musset et al., 2012) are open source MDA Frameworks. AndroMDA generates components in different programming languages by means of customizable plug-ins that support code generation. It mainly uses UML models stored in XMI. Acceleo provides code generators (JEE, .Net, Php)

and template editors for Eclipse. Web application domain purpose such as WebDSL generator (Hemel et al., 2010) and WebRatio (Brambilla et al., 2010). WebDSL is a DSL for web-applications with rich data models. WebDSL provides sub languages for the specification of data models, for the definition of custom pages, and for the manipulation of data. WebRatio is a an eclipse-based case tool for engineering web applications. It uses the WebML modeling language for designing Web sites. This kind of tools use several models to represent system specification, such as, architecture, behavior and data model. Instead, SWB uses only an ontology to represent those models. Moreover, by using ontologies SWB provides the mechanism to expose system data in the Semantic Web.

Other works are those related with the use of ontologies for code generation, such as: ODASE (Uschold, 2008), an ontology-driven architecture which permits to represent into an ontology the business knowledge needed to build an IT application. ODASE is a general purpose platform, instead SWB has a specialized set of reusable components for the Web applications domain, therefore, in this domain, SWB can be more usable. RDFReactor (Völkel and Sure, 2005) automatically generates Java classes from OWL ontologies through an intermediate model.This intermediate model must be updated as RDF specifications evolve and the Java generated classes must be extended manually in order to keep all changes made during the development process. SWB automatically supports the evolution of models by the modification of the SWB ontology.Jastor (Kalyanpur et al., 2004) generates Java code from OWL ontologies using Java Beans. OWL classes are mapped into Java interfaces to solve the problems of multiple inheritance. Manual extensions need to be addressed for each generated Java interface to achieve the system implementation. With SWB, Java Class extensions are defined automatically.

7 CONCLUSIONS

We have presented an Ontology-Driven Development Framework called SemanticWebBuilder (SWB). The starting point of the development process is the ontological representation of a problem domain by extending the base structure of our proposed ontology *SWB ontology*. As output, SWB automatically generates the infrastructure of the domain system which manages the persistence of system and business information as RDF triples. The semi-automatic development process through SWB has been described, so that, a resultant system could be easily extended, being able to build complex systems. SWB is specialized in the Web application domain. Therefore, the SWB ontology has been supplemented with Web applications requirements. Additionally, specific components have been integrated to the SWB Framework to manage Web applications needs. Through the use of ontologies SWB provides the following advantages: flexible and agile mechanisms to adapt to new business needs, reducing code maintenance issues and increasing reuse of code; increased reliability since human errors are reduced; reasoners can run over ontologies and data to make proof of assertions about business information; and finally, Semantic Web mechanisms can be applied to expose business information in the Semantic Web and to exploit data through paradigms such as Linked Data.

SWB has been successfully applied in the development of several Web applications actually used by government dependencies in Mexico. For instance, Visit Mexico - http://www.visitmexico.com/, Secretary of labor and social welfare - http://www.stps.gob.mx, and Secretary of public education - http://www.sep.gob.mx/.

REFERENCES

Bohlen, M. et al. (2012). AndroMDA. http://www.andromda.org/.

Borst, W. N. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. PhD thesis, Enschede.

Brambilla, M., Butti, S., and Fraternali, P. (2010). Webratio bpm: A tool for designing and deploying business processes on the web. In *Web Engineering*, volume 6189, pages 415–429.

Gruber, T. R. (1993). Toward principles for the design of ontologies used for knowledge sharing. In *International Journal of Human-Computer Studies*, volume 43, pages 907–928.

Happel, H.-J. and Seedorf, S. (2006). Applications of ontologies in software engineering. In *SWESE'06*, Athens, USA.

Hemel, Z., Kats, L. C. L., Groenewegen, D. M., and Visser, E. (2010). Code generation by model transformation: a case study in transformation modularity. *Software and System Modeling*, 9(3):375–402.

Kalyanpur, A., Pastor, D. J., Battle, S., and Padget, J. A. (2004). Automatic mapping of owl ontologies into java. In *SEKE'04*, pages 98–103.

Musset, J. et al. (2012). Acceleo. http://www.acceleo.org/.

Studer, R., Benjamins, R., and Fensel, D. (1998). Knowledge engineering: principles and methods. *Data and knowledge engineering*, 25:161–197.

Uschold, M. (2008). Ontology-driven information systems: Past, present and future. In *FOIS'08*, pages 3–18.

Völkel, M. and Sure, Y. (2005). Rdfreactor - from ontologies to programmatic data access. In *ISWC'05*.