SemanticWebBuilder: A Framework for Semantic Web Applications Development

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

***Abstract*—Nowadays, the use of models in software analysis and design is a common practice, moreover, most of the development projects use UML models to capture the dynamic and static view of the system to-be. With the emergence of the Semantic Web and standards such as RDF and OWL, new opportunities for the use of ontologies in software development have arisen. In this context, several works have used ontologies as a mechanism for the capture of system requirements and as an input for automatic code generation. However, most of current works propose generic development platforms that focus only in the code generation phase. In this paper, an Ontology-Driven Development Framework, called SemanticWebBuilder (SWB) is presented which provides a defined methodology and an agile platform for the development of semantic Web applications. The SWB approach encourages ontology and code reuse to reduce development time and has been widely used to develop Web applications for several government agencies and private companies in Mexico.**

***Keywords*—Semantic Web Technologies, Ontology-Driven Development, Model-Driven Development**

I. Introduction

Nowadays, it is recognized the importance of using models in the software development life-cycle, in particular for large software systems [1]. Thus, in the analysis and design phases, models play an important role as a mechanism to capture the static and dynamic views of the system to-be, allowing to describe the problem domain, to create the documentation of system requirements and to estimate the complexity of the whole system implementation; in the development phase, models are used for the automatic generation of source code through model to code transformations; finally, in the maintenance phase, the source code could be automatically regenerated from model definitions if requirements change.

In this context, Model-Driven Development (MDD) approaches have emerged to support software development life-cycle. For instance, the Model-Driven Architecture (MDA), an approach in which models are first-class artifacts, integrated into the development process by means of a chain of transformations from Platform-Independent Models (PIM) through Platform-Specific Models (PSM) to coded application [2]. Commonly, the PIM and PSM are defined in the standard Unified Modeling Language (UML). For example: AndroMDA [3], a general purpose development tool that generates components in different programming languages starting from UML models; and Acceleo [4], a tool that provides code generators (JEE, .Net, Php) and template editors for Eclipse. MDA is used for the development of systems in several domains, whilst, there are other MDD approaches focused in specific domains that use Domain Specific Languages (DSL) for the specification of models, for example: the WebDSL generator [5], a DSL for web-applications which provides modeling languages for the specification of data models and custom pages, and for the manipulation of data.

On the other hand, with the emergence of the Semantic Web [6] and the ontology models, as well as the creation of new standards (such as RDF [7] and OWL [8]) and tools (such as ontology editors, reasoners and RDF APIs), several research works have addressed the incorporation of ontology models in software development [9]–[12]. From this, a new MDD approach have arisen, that is, Ontology-Driven Development (ODD) where ontologies are used as models to address system development [9]. ODD approaches (besides the advantages of MDD) provide through ontologies a formal mechanism to capture system requirements in a higher abstraction level for a common understanding of the problem domain and they enable the reasoning over generated models to achieve model validation or new knowledge discovery [10]. Examples of ODD works are: RDFReactor [13], a tool that automatically generates Java classes from OWL ontologies by means of an intermediate model (called JModel) to simplify the ontology model and generate the Java classes, resolving the multiple inheritance issue; and Jastor [14], which is focused on producing object-oriented specifications from ontological models in a semi-automatic way, therefore, it needs a manual Java Class extension for each one of the generated interfaces in order to achieve the implementation of the defined system.

Until recent years, the existing ODD approaches covered only part of the entire software development life-cycle, but there was no clear efforts to integrate semantic technologies to drive the whole process. To address this issue, a new approach called Ontology-Driven Information Systems (ODIS) [11] was presented. Conceptually, ODIS does not limit the application of ontologies to analysis and design, but enforces its use as part of the final runtime components of the system. However, there is still the need to provide a comprehensive framework for the development of semantic software systems using ODIS insights.

In this paper we present SemanticWebBuilder (SWB), a practical framework for the agile development of Semantic Web applications and information systems. SWB implements the ideas of ODIS and uses ontologies and Semantic Web standards throughout the software development process. Accordingly, ontologies are part of the system analysis and design, but also serve as input for code generation and as a mechanism to provide data management for running applications. To achieve this, SWB provides developers four main elements: a) An extensible ontology for the domain of Web applications and systems that represents system structure, behavior and data model (all in a single model, rather than MDA and DSL approaches that need several models), together with business logic and software restrictions that would be commonly hard coded in most current systems; b) a defined methodology to drive the life-cycle of systems development using ontology models; c) a transformation mechanism to convert ontology models into source code; and d) a software platform to accelerate the development of applications where runtime data is persisted in RDF format.

SWB has been widely validated in practice through the development of several products, as well as Web applications and systems for Mexican government agencies and private companies.

II. Overview of the SemanticWebBuilder Framework

The SemanticWebBuilder framework (figure 1) is devoted to address agile development of applications and systems. To achieve this, an ontology is used to model, in a methodological way, the system requirements specification (objects architecture, behavioral aspects and display properties). To accelerate requirements modeling, a base ontology (called SWBOntology) with the definition of reusable concepts and behavioural aspects for Web applications is extended.

After the modeling phase, the resulting requirements ontology is automatically processed by a code generator to obtain the source code for the core of the system to-be. This source code constitutes a domain specific API that encapsulates classes and properties from the requirements ontology and reuses implemented functionality for the concepts defined in the base ontology. In this case, Java is the preferred output language for the code generator. The generated API is supported by a set of software libraries comprised in the SWBPlatform, intended to accelerate software development and to provide a standardized mechanism to manage object persistence in RDF format in a transparent way for the developer.

Finally, with the generated API and the SWBPlatform, developers proceed to code the application logic for the system using a high level set of Java classes and methods that encapsulate all the complexity of RDF data management. In this final phase, SWBPlatform provides mechanisms to automatically develop Web UI components for data capture and display taking as schema the display properties defined in the requirements ontology.

In the following subsections we describe the main components and the development process of SWB, except for the associated methodology, which is described in section III.

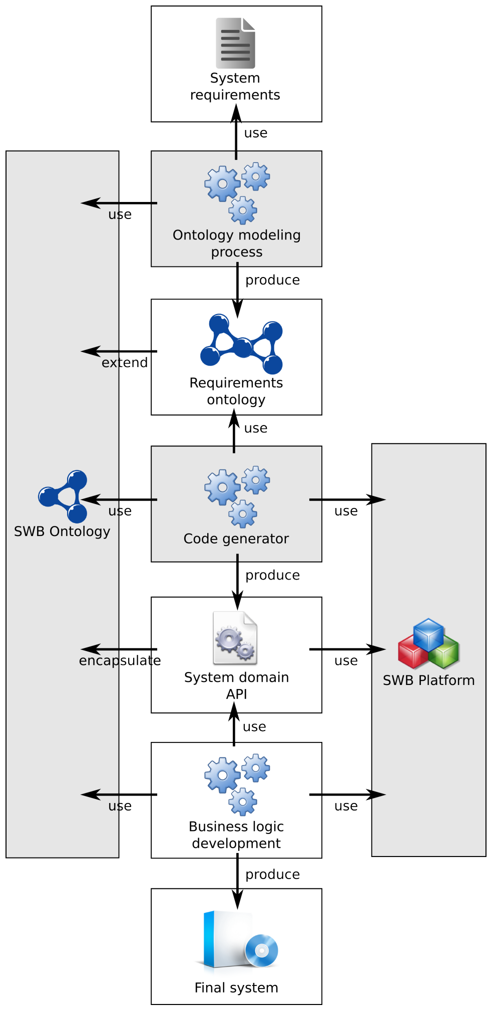

Fig. 1. SemanticWebBuilder Framework

A. SWBOntology

To achieve an agile and standardized way to capture system requirements, the definition of the SWB Ontology[1] was performed using OWL. This ontology is intended to serve as a base for extending new requirements ontologies.

Due to the multiple inheritance restrictions of the Java language and to enforce a correct model to source code transformation, two OWL classes were defined in the SWBOntology. The first one is the SWBClass, which corresponds to a system object definition in Java language (Java class). And the second one is the SWBInterface, which corresponds to object behaviours in Java language (Java interface). Accordingly, concepts for Web applications development were defined as a subclasses of the SWBClass, such as the concepts of WebPage, User, Role, PresentationRule, Form, WebSite, etc. In the same way, common behaviors for the concepts were defined as subclasses of the SWBInterface, such as its capacity to be activated, described, scheduled, traced or deleted. Additionally, OWL concepts were defined to describe Web components,

[1]http://www.semanticbuilder.com/SemWB4/SWB4/swb/web/WEB-INF/owl/swb.owl

code generation information and UI elements for the generated system, such as the concepts of SemanticResource, CodePackage, FormElement and DisplayProperty.

It is important to point out that the SWBOntology is not a generic domain ontology that describes the domain of Web applications. It is instead a specific ontology that contains itself the basic requirements for Web applications development and code generation.

B. *SWBPlatform*

SWBPlatform is a software package with a set of Java classes and libraries developed at the same time as the SWBOntology. The main goal of SWBPlatform is to encapsulate management and persistence of RDF information in a generic way, using OWL definitions as data schemas to store and retrieve RDF graphs. This allows to isolate the data layer from the application layer through the definition of Java wrappers for OWL concepts. This includes wrappers such as SemanticClass, SemanticProperty, SemanticObject and SemanticLiteral, that correspond to an OWL Class, OWL Property, RDF Resource and RDF Literal respectively. Additionally, the definition of a wrapper for the SWBClass and the SWBInterface classes was done.

Internally, the wrappers for OWL concepts in SWB Platform implement basic data validations and make use of common RDF libraries such as Apache Jena[2] to achieve RDF management through several triple stores and databases.

C. *Code Generator*

Once the SWBOntology and the SWBPlatform were developed, a code generator (SWBCodeGen) was built that uses the wrappers of SWBPlatform. This code generator takes as input a requirements ontology extending the SWB Ontology and provides as output a set of Java classes and methods to manage object properties and persistence. The generated code constitutes a high level Java API that hides the complexity associated to RDF management.

The transformation process from the ontology model to Java code is performed following two simple rules: 1) a concept definition in the requirements ontology, whose type is SWBClass, is mapped to two Java Classes. The first one (base class) is named adding the prefix "Base" to the class name and implements methods to get and set object properties through SWBPlatform wrappers and utilities. The second one is named the same as the the original OWL class and extends the base class; 2) a concept definition in the requirements ontology whose type is SWBInterface is mapped to the corresponding wrapper in SWBPlatform (the wrapper is called GenericObject). This interface declares the methods to get and set object properties.

Hierarchy of concepts in the requirements ontology is preserved in the generated code using the Java "extends" mechanism. In the case that concepts subclass the SWBInterface, the generated code implements the method definitions in the extended classes making use of the Java "implements"

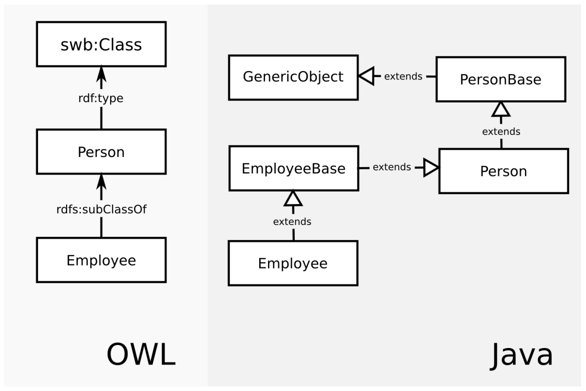

Fig. 2. Mapping of OWL concepts to Java Classes in the code generation

mechanism. An schematic example of the transformation of SWBClass concepts is presented in figure 2.

III. The SWB Software Development cycle

The previously described components of the SWB framework constitute its operational base. The methodological approach to manage the development cycle, which constitutes the other part of the framework, is exposed in this section.

Our approach reuses several aspects of the traditional object-oriented methodologies, considering the same phases of software development. However, our phases have different meaning and also generate different types of outputs, mainly in the requirements elicitation and knowledge modeling phases. Figure 3 shows the iterative context of the approach, where the domain knowledge is refined using ontology models. In the early stage of the development (the first one or two cycles), the participation of experts of the problem domain is required to help in the capture of relevant knowledge and system requirements, as well as to detect possible ontologies to reuse. Each subsequent iteration involves a refinement of both the software system implementation and the associated knowledge model, accelerating the development as the captured knowledge for

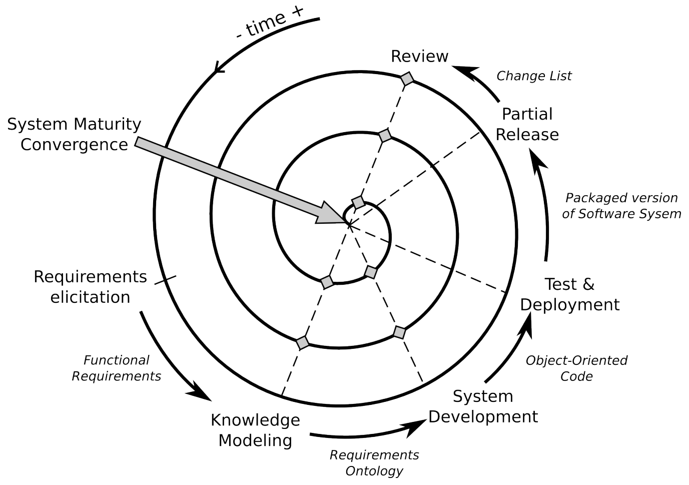

Fig. 3. SWB Software development cycle

[2]http://jena.apache.org/

the system converges to cover the solution domain and the system fulfills all the defined requirements.

At the end of each iteration a review of the development status must be performed by the development leader in order to analyze current conditions and take the decision to proceed to the next iteration or to an intermediate step (possible intermediate steps are marked as diamonds in Figure 3). For example, if as a result of the review it is determined that the knowledge domain remains without changes according to previous cycles, and only certain aspects of the functionality need to be adjusted, then the next cycle will start from the development phase of the process instead of the knowledge modeling phase.

It is important to point out that even if the approach presented in Figure 3 has similarities with the spiral model of Boehm [15], there are several conceptual and functional differences in modeling and development phases. To be precise, the input of our approach is a set of concepts and requirements (defined in ontologies) closer to the final users way of thinking, instead of objects that are usually only understood by software engineers. By the other hand, the output is an ontology-driven information system, which considers and implements data persistence in RDF format.

IV. Using SWB for the development of a Semantic Content Management System

To prove the applicability of SWB in industrial development projects, the definition and implementation of a Semantic Content Management System (Semantic CMS) was achieved using the presented ideas. The Semantic CMS (called SWBPortal[3]) was intended to be the new major version of the flagship CMS of INFOTEC. This new version should provide a powerful and flexible Web platform for the management of several Web sites, Web pages, and Web components giving context to the published information through the use of Semantic technology, thus allowing the subsequent deployment of Semantic Web applications. The phases in the development of SWBPortal, conducted during several iterations are described in the following subsections. These phases served as well to validate the systematic achievement of results using the SWB framework.

A. Requirements elicitation

This activity was executed using traditional Requirements Engineering methodologies and taking as baseline the implemented functionality in the previous version of the flagship CMS, as well as new requirements coming from final users. A main difference of our approach of doing this activity is that the main goal, besides the gathering of functional requirements, was the discovering of the knowledge (concepts, hierarchical and associative relations) of the CMS domain, hidden along the different UML models, documentation and source code of previous system. The result of this activity was an entity-relationship diagram of the domain and the list of functional requirements for SWBPortal.

[3]http://www.semanticwebbuilder.org.mx/es/swb/SWB_Portal_Productos

B. SWBPortal Knowledge modeling

Following our approach, the concepts of the CMS domain and the SWBPortal requirements were modeled in an ontology. In this particular case, the SWBOntology was not extended, but rather complemented with SWBPortal knowledge. This means that the SWBOntology also contains the SWBPortal requirements. The decision of modeling requirements this way was made for the following two reasons: a) the SWBOntology already defined many of the CMS domain concepts; b) to allow all Web components developed using SWB to be included seamlessly in Web pages deployed using SWBPortal.

Thus, to complete this activity, new concepts such as those of hierarchical relation and hierarchical node were added to the SWBOntology along with behaviors and properties needed to implement SWBPortal. The final result was the SWBOntology enriched and containing the definition of classes, properties and relations needed for the development of Semantic Web applications to be deployed mainly through SWBPortal.

C. Code Generation and SWBPortal development

In this activity, the SWBCodeGen was configured and executed taking as input the SWBOntology in order to get the source code of the core SWBPortal. The result was the Java API of SWBPortal (SWBPortal API) with all classes and methods needed to manage the persistence of Web content in RDF format. Using the SWBPortal API and the SWBPlatform, functions to manage portlet-like components and a mechanism to build generic Web forms (SWBFormManager) to capture and present the information of SWBPortal objects were developed as a first step. As a second step, all the components of SWBPortal were developed on top of a Java Web application architecture.

These components included modules for the management of user sessions, publish flows, user devices, user rules, Web sites and Web pages, as well as modules to edit HTML templates, insert Web components, count hits, generate reports, log user activity, index and search content, etc. Finally, the design and integration of Web UI components was achieved using the SWBFormManager to provide a generic UI environment, customizable for many Web sites and components. The result of this activity was SWBPortal itself.

D. SWBPortal test, deployment and release

This final activity was done using traditional testing and deployment methodologies. An advantage of using SWB, confirmed in testing, was the decreased rate of bugs injection in the code of objects persistence (because of the automatic code generation). Also, this code reduced testing time because it had to be tested only once. In fact, we can make sure that if the generated code works for a particular case, it works the same for all other cases.

V. Conclusion

We have presented SemanticWebBuilder, an agile framework for the development of Semantic Web applications that implements the ODIS ideas. To build the framework, one needed step was the definition of a base ontology for requirements elicitation. This extensible ontology, called SWBOntology captures the definition of concepts from the domain of Web applications and systems, as well as behavioral aspects for the defined concepts, defining then in a single model, system structure, behavior and data model (rather than MDA and DSL approaches that need several models). A second step consisted in the development of a software package, called SWBPlatform, with Java classes to encapsulate RDF data manipulation in a high abstraction level set of Java objects. Common RDF libraries such as Apache Jena were used for this purpose. With the SWBOntology and the SWBPlatform, a third step consisted in the development of a code generator that applies simple transformation rules to generate Java source code from OWL ontologies.

With the use of the SWBOntology, the SWBPlatform and the code generator, SWB covers the operational aspects of Ontology-Driven Development, however, a methodological approach to manage the entire software development cycle was defined to cover all aspects of the ODIS ideas. Thus, SWB provides the following advantages: a) it applies an iterative methodology to address Ontology-driven software development; b) it contains flexible and agile mechanisms to adapt and extend system development to new business needs; c) it reduces code maintenance issues and enforces code reuse; d) it increases the reliability of source code since human errors are reduced through automatic code generation; e) reasoners can be run over requirements ontologies and the system run-time data to make proof of assertions about business information; f) simple mechanisms can be applied to expose the information of the developed systems in the Semantic Web and to apply paradigms such as Linked Data to semantically enrich the information.

SWB components and methodology have been widely validated through the development of two Open-source products. The first one is a Semantic Content Management System for the creation and management of Web portals, called SWB-Portal and described in previous sections. The second one is named SWBProcess[4] and is a Semantic Business Process Management System supporting process modeling, execution and monitoring in a Web-based environment. Additionally to these products, several Web applications and systems deployed in the portals of major Mexican government agencies have been developed using SWB and SWBPortal. Some examples are VisitMexico[5], the portal of the Secretary of labor and social welfare[6], and the portal of the Secretary of public education[7]. In the private sector, it has been used, for example, to build applications for MedicaSur[8].

References

[1] B. Hailpern and P. Tarr, "Model-driven development: the good, the bad, and the ugly," *IBM Syst. J.*, vol. 45, no. 3, pp. 451–461, Jul. 2006.

[2] OMG, "Model Driven Architecture. Online," http://www.omg.org/mda/, 2003, last access: 05/09/2013.

[3] M. Bohlen *et al.* (2012) AndroMDA. http://www.andromda.org/.

[4] J. Musset *et al.* (2012) Acceleo. http://www.acceleo.org/.

[5] Z. Hemel, L. C. L. Kats, D. M. Groenewegen, and E. Visser, "Code generation by model transformation: a case study in transformation modularity," *Software and System Modeling*, vol. 9, no. 3, pp. 375–402, 2010.

[6] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, pp. 34–43, 2001.

[7] *Resource Description Framework (RDF): concepts and abstract syntax*, World Wide Web Consortium Std. [Online]. Available: http://www.w3.org/TR/rdf-concepts/

[8] *Web Ontology Language*, World Wide Web Consortium Std. [Online]. Available: http://www.w3.org/2004/OWL/

[9] P. Parrend and B. David, "Use of ontologies as a way to automate mde processes," in *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, vol. 1, 2005, pp. 567–570.

[10] H.-J. Happel and S. Seedorf, "Applications of ontologies in software engineering," in *SWESE'06*, Athens, USA, 2006.

[11] M. Uschold, "Ontology-driven information systems: Past, present and future," in *FOIS'08*, 2008, pp. 3–18.

[12] N. K. Dragan Gasevic and M. Milanovic, "Ontologies and software engineering," in *Handbook of Software Engineering and Knowledge Engineering: Fundamentals*, 1st ed. World Scientific Press, 2009, vol. 3.

[13] M. Volkel, "RDFReactor - From Ontologies to Programmatic Data Access," in *Jena User Conference*. HP Bristol, May 2006.

[14] A. Kalyanpur, D. J. Pastor, S. Battle, and J. Padget, "Automatic Mapping of OWL Ontologies into Java," in *Sixteenth International Conference on Software Engineering and Knowledge Engineering*, G. R. Frank Maurer, Ed., June 2004, pp. 98–103.

[15] B. W. Boehm, "A spiral model of software development and enhancement," in *ACM SigSoft Software Engineering Notes*, vol. 2, no. 4, 1986, pp. 22–42.

[4] http://www.semanticwebbuilder.org.mx/SWBProcess

[5] http://www.visitmexico.com/en/

[6] http://www.sedesol.gob.mx/

[7] http://www.sep.gob.mx/

[8] http://www.medicasur.com.mx/