Filling the Gap between Semantic Web and Model Driven Engineering: The TwoUse Toolkit

Fernando Silva Parreiras1, Tobias Walter1, and Gerd Gröner1

ISWeb — Information Systems and Semantic Web,
Institute for Computer Science, University of Koblenz-Landau
Universitätsstrasse 1, Koblenz 56070, Germany
(parreiras, walter, groener}@uni-koblenz.de

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.

1 Motivation

The interest in integrating ontology technologies and software engineering has gained more attention with commercial and scientific initiatives. The Semantic Web Best Practice and Deployment Working Group (SWBPD) in W3C included a Software Engineering Task Force (SETF) to explore how Semantic Web and Software Engineering can cooperate. The Object Management Group (OMG) has an Ontology Platform Special Interest Group (PSIG) aiming at formalizing the semantics in software by knowledge representation and related technologies. The concrete results of such initiatives are the specification of the OMG Ontology Definition Metamodel, the Ontology Web Language (OWL) Metamodel, the introduction to Ontology Driven Architectures and a Semantic Web Primer for Object-Oriented Software Developers.

However, practitioners interested in integrating ontology technologies and software engineering lack on tool support. Although guidelines for model transformations and implementations of these transformations exist, a comprehensive framework dedicated to fill the gap between model-driven engineering and ontology technologies is lacking so far. For example, the IBM Integrated Ontology Development Toolkit 1 does not provide support to instance editing or multiple ontologies. Moreover, ontology editors like the NeOn Toolkit 2 support exclusively ontology development and do not provide tools for software developers.

Providing a framework for integrating MDE and ontology technologies requires dealing with the following challenges:

- Seamless integration between class-based modeling languages and OWL. Developers should be able to import/export models seamlessly in different formats like Ecore, UML, XML and OWL.
- Modeling Design Pattern. Integrated frameworks should provide developers with capabilities for reusing exiting knowledge from other projects in the form of design patterns.
Integration with other semantic web and modeling languages such as the Semantic Web Rule Language (SWRL) and OCL. Developers should be able to work with semantic web languages (OWL, SWRL and SPARQL Query Language for RDF) as well as with software languages (UML and OCL) in accordance to the task to be solved.

The TwoUse Toolkit, a framework for development of model-driven applications with ontology technologies, addresses these challenges. It comprises a set of model transformations, graphical and textual editors and reasoning services. In this demo, we summarize the main features of the TwoUse toolkit on Section 2 and, in Section 3, we illustrate the TwoUse toolkit with two use cases.

2 TwoUse Toolkit Main Features

The TwoUse Toolkit\(^3\) is an implementation of current OMG and W3C standards for developing ontology-based software models and model-based OWL ontologies. It is a model-driven tool to bridge the gap between Semantic Web and Model Driven Software Development.

TwoUse toolkit building blocks are:

- A set of model transformations. Generic transformations like Ecore2OWL and XML2OWL allows developers to transform any software language into OWL. Specific transformations like UML2OWL and BPMN2OWL allow developers to OWL representations of software models.
- A set of textual and graphical editors. TwoUse relies on textual and graphical editors for editing and parsing W3C standard languages like OWL2 and SPARQL, OMG standards like UML and OCL as well as other domain-specific languages.
- A set of reasoning services like classification, realization, query answering and explanation.

The TwoUse Toolkit has two user profiles: model-driven software developer and OWL ontology engineer. In this demo, we concentrate on the functionalities for model-driven software developers:

- Describe classes in UML class diagrams using OWL class descriptions.
- Semantically search for classes, properties and instances in UML class diagrams.
- Design business rules using the UML Profile for SWRL.
- Extend software design patterns with OWL class descriptions.
- Make sense of UML class diagrams using inference explanations.
- Link software engineering artifacts by transforming software languages into OWL (OWLizing).
- Write OWL queries using SPARQL or the OWL-like languages using query editors with syntax highlighting.

\(^3\) \url{http://twouse.googlecode.com/}. TwoUse toolkit is an open source tool developed by the WeST Institute at the University of Koblenz-Landau.
Validate refinements on business process models.

In the following, we illustrate the synergy between these features in two use cases: transforming software languages into OWL and integrating UML and OWL.

3 Demonstration

3.1 OWLizer: Transforming Software Languages into OWL

Software development consists of multiple phases, from inception to production. During each software development phase, developers and other actors generate many artifacts, e.g., documents, models, diagrams, code, tests and bug reports. Although some of these artifacts are integrated, they are usually handled as islands inside the software development process.

Many of these artifacts (graphical or textual) are written using a structured language, which has a defined grammar. In a model-driven environment, concepts of software languages are represented by metamodels, whereas the artifacts written in those software languages are represented by models, which are described by the language metamodel. Thus, by transforming software metamodels and models into OWL and by aligning the OWL ontologies corresponding to software languages, we are able to link multiple data sources of a software development process, creating a linked-data repository for software development.

Let us consider an example of integrating two data sources: UML diagrams and Java Code. Regardless of generating Java code from UML diagrams, developers would like to have a consistent view of corresponding classes and methods in UML and Java, i.e., developers might want to consult UML diagrams looking for a corresponding Java class. In this scenario, OWL and ontology technologies play an important role.

Fig. 1 (A) depicts the usage of M3 transformations together with ontology technologies [1]. UML metamodel and model as well as Java grammar (metamodel) and java code (model) are transformed into OWL ontologies. Ontology alignment techniques [2] might identify some concepts in common between the two ontologies (UML and Java), e.g., package, class, method. Moreover, individuals with the same name in these two ontologies are likely the same.
Once the two ontologies are aligned, queries against the Java ontology also retrieve elements defined in UML diagrams. Now it is possible to retrieve sequence diagrams including a given Java class, since the two artifacts (UML diagrams and Java code) are now linked. This is only one example of the great potential provided by linking software engineering artifacts using OWL technologies.

3.2 Integrating UML and OWL

In general, software design patterns deal with variation and delegation. However, as already documented by [3], software design patterns present drawbacks. For example, in the Strategy Pattern, the clients must be aware of variations and of the criteria to select variations at runtime. Hence, the question arises of how the selection of specific classes could be determined using only their descriptions rather than by weaving the descriptions into client classes.

The basic idea lies in decoupling class selection from the definition of client classes by exploiting OWL-DL modeling and reasoning. We explore a slight modification of the Strategy Pattern that includes OWL-DL modeling and that leads us to a minor, but powerful variation of existing practices: the Selector Pattern [4] (Fig. 1 (B)).

The Context has the operation select, which uses OWL-like query operations to dynamically classify the object according to the logical descriptions of the variants. A Variant is returned as result. Then, the Context establishes an association with the Concept, which interfaces the variation.

The application of the Selector Pattern presents the following consequences:

**Reuse.** The knowledge represented in OWL-DL can be reused independently of platform or programming language.

**Flexibility.** The knowledge encoded in OWL-DL can be modeled and evolved independently of the execution logic.

**Testability.** The OWL-DL part of the model can be automatically tested by logical unit tests, independently of the UML development.

The application of TwoUse can be extended to other design patterns concerning variant management and control of execution and method selection. Design patterns that factor out commonality of related objects, like Prototype, Factory Method and Template Method, are good candidates.

References