An Ontology-based Approach for Enabling Linked Data Capabilities to MOF Compliant Models

Fernando Silva Parreiras[1], Gerd Gröner[2] and Tobias Walter[2]
[1]*Faculty of Business Sciences, FUMEC University, Av Afonso Pena 3880, 30130009 Belo Horizonte, Brazil*
[2]*Institute for Web Science and Technologies – WeST, University of Koblenz-Landau, 56072 Koblenz, Germany*

Keywords: OWL, Linked Data, Software Engineering, UML, MOF, Model Driven Engineering.

Abstract: In the software development process, there are standards for general-purpose modeling languages and domain-specific languages, capable of capturing information about multiple views of systems, e.g., static structure and dynamic behavior. In a networked and federated development environment, modeling artifacts need to be linked, adapted and analyzed to meet the information requirements of multiple stakeholders. In this paper, we present an approach for linking, transforming and querying MOF-compliant modeling languages on the web of data. We define structural mappings between MOF and OWL and propose the usage of semantic web technologies for linking and querying software models.

1 INTRODUCTION

In a model-driven architecture, software engineers rely on a variety of languages for designing software systems. As different stakeholders need different views of the information, the software development environment needs to encompass a myriad of general-purpose and domain-specific languages with complementary and overlapping applications.

Since it is hard to capture all aspects of software into one single model, contemporary model-driven architectures include numerous notations to be used according to the software development task. The inevitable usage of multiple languages leads to unmanageable redundancy in developing and managing the same information across multiple artifacts and, eventually, information inconsistency. With the growing demand for networked and federated environments, the question arises about what and how existing web standards can help existing modeling standards in fulfilling the requirements of a *web of models*.

Semantic web technologies (Antoniou and van-Harmelen, 2004) and linked open data (LOD) principles (Bizer et al., 2008) allow any kind of data to be represented, identified, linked and formalized on the web. The same data can be adapted for use with another purpose, according to the software engineer's perspective.

The interest on this topic has motivated the Object Management Group (OMG) to issue a request for proposal aiming at defining a structural mapping between Meta Object Facility (MOF) models and Resource Description Framework (RDF) representations (OMG, 2009b). This mapping should make possible to apply LOD principles into MOF compliant models and to publish MOF compliant models as LOD resources.

In a collaborative environment, developers create architectures with information expressed in multiple modeling languages. These languages should be grounded with stronger semantics than MOF can provide today.

The Web Ontology Language (OWL) (W3C OWL Working Group, 2009) provides a powerful solution for formally describing domain concepts in networked environments. OWL is part of the semantic web stack, is compatible with RDF and with LOD principles. OWL's objective is to provide evolution, interoperability, and inconsistency detection of shared conceptualizations.

Although transformations from the MOF-like metamodels to OWL have been proposed before (Gasevic et al., 2004), addressing the aforementioned problems requires a coherent framework comprising techniques not only for transforming but for extending, linking and queries MOF compliant models.

In this paper, we describe such a framework for supporting the interaction and interrelationships of modeling languages in distributed software modeling environments. We present our contribution as fol-

Silva Parreiras F., Gröner G. and Walter T..
An Ontology-based Approach for Enabling Linked Data Capabilities to MOF Compliant Models.
DOI: 10.5220/0004006201260131
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 126-131
ISBN: 978-989-8565-11-2

lows: Section analyzes the requirements to be addressed. Section 3 describes the building blocks of our approach. We analyze the related work in Section 4. Section 5 finishes the paper.

2 REQUIREMENTS

Based on demand identified in OMG's request for proposal (OMG, 2009b), we identify three fundamental requirements for realizing a linked-open data environment in model-driven software development:

RQ1: Extend Modeling Languages. Metamodeling languages based on EMOF do not provide capabilities for integrating conforming modeling languages, or for defining constraints for these languages. Here, an additional (constraint- and expression-) language that provides more expressiveness than usual modeling languages is required. It should provide constructs to integrate languages (e.g. by asserting that concepts are equivalent).

RQ2: Identification of Equivalences. For semi-automatically integrating modeling languages, a mechanism is required that allows for identifying concepts that are equal over languages and that are then candidate for merging. The result is a combined metamodel.

RQ3: Integration Management. To achieve an integrated representation of several modeling languages, we need more that one technique. Firstly, languages and conforming models must be *transformed* into the same representation. Further they must be *linked* for *reasoning* and *querying* over languages and models.

Addressing these requirements allows for achieving the following features:

Consistent View over Multiple MOF Models. Based on an integration of multiple (MOF-based) languages and constraints describing dependencies between languages, we get a consistent view over multiple models conforming to those languages.

Integrated Well-formedness Constraints. Based on the extended modeling language, we are able to define integrated well-formedness constraints in an embedded manner. This means that language designers should be able to use the metamodeling language (e.g. EMOF) of choice and still annotate elements of language metamodels to define constraints.

Dependency Checking. Once languages are integrated, apart from consistency checking (checking if models conform to the language metamodel), it is possible to check dependencies across language models.

Query Answering. Based on underlying formal semantics and constraints, it is possible to query models and inferences. It is possible to ask and answer the following question: What is the effect of updating the plug-in pellet? How much do I have to test? Furthermore we are able to query for impacts of model elements to others (Impact Analysis) and thus can identify e.g. cyclic dependencies or other unexpected consequences.

3 INTEGRATING MOF AND OWL

In this section, we describe how we exploit ontology technologies to represent the semantics of modeling languages with OWL. We present how we link modeling languages with OWL, how we transform MOF-based models in OWL and how we query and manage links between modeling languages.

Our approach consists of (1) optionally extending MOF metamodels with OWL annotations, (2) translating MOF metamodels and models into OWL concepts, properties, individuals and property assertions, (3) using ontology matching to link same classes and individuals across metamodels and (4) write and execute queries over multiple MOF models.

3.1 OWL for Conceptual Modeling

Despite features of OWL like shared conceptualizations, support to evolution, interoperability and consistency detection, OWL provides multiple means for describing classes. It relies on class and property expressions to construct class descriptions. These descriptions represent sets of individuals by specifying conditions on the individuals' properties.

In the following list, we present examples using OWL2 syntax of constructs used to link MOF metamodels with OWL. In Line 1, we describe the equivalence of a UML Activity and BpmnDiagram. The equivalence of the set of individuals of the class OpaqueAction and the set of individuals of the class Activity where the property activityType is set to Task in the BPMN metamodel is defined in Line 2. Lines 3 and 4 characterize the property general of the UML metamodel as irreflexive and transitive. In Line 5, we derive a new property in the BPMN metamodel based on a property chain, i.e., a composition of the properties outgoingEdges and target are properties of sucessorActivities. For instance, if individual x outgoingEdges y and y target z than x sucessorActivities z. Similarly, a property chain ancestorNodes for the UML metamodel is defined in Line 6. The equivalence of the defined property chains is expressed in Line 7.

Listing 1: Linking MOF metamodels with OWL.

```
1 EquivalentClasses (uml:Activity
      bpmn:BpmnDiagram)
  EquivalentClasses (uml:OpaqueAction
      ObjectSomeValuesFrom
      (bpmn:activityType bpmn:Task))
  IrreflexiveObjectProperty
      (uml:general)
  TransitiveObjectProperty (uml:general)
5 SubObjectPropertyOf(
      ObjectPropertyChain
      (bpmn:outgoingEdges bpmn:target)
      bpmn:sucessorActivities)
  SubObjectPropertyOf(
      ObjectPropertyChain (uml:outgoing
      uml:target) uml:sucessorNodes)
  SubObjectPropertyOf(
      ObjectPropertyChain (uml:include
      uml:addition) uml:includeUseCases)
  SubObjectPropertyOf(
      ObjectPropertyChain
      (ObjectInverseOf(uml:addition)
      uml:includingCase)
      uml:includingUseCases)
  EquivalentObjectProperties
      (uml:sucessorNodes
      bpmn:sucessorActivities)
```

At the model level, developers can link models (metamodel instances) with OWL constructs. The SameIndividual axioms allow to define the equality of individuals in order to assert that instances of different metamodels are the same. For example, if we have a UML package called west.twouse.backend, we can assert that this package is the same as the Java package with the same name – SameIndividual(uml:west.twouse.backend java:west.twouse.backend).

Additionally, OWL2 provides other standard constructs that can help to enrich MOF metamodels and to extend its expressiveness (RQ1). Object property axioms aim at characterizing object properties like the definition of subproperty relations and the expression of reflexive, irreflexive, symmetric, asymmetric and transitive properties. Moreover, equivalent and disjoint axioms allow explicit statements on the equivalence and disjointness of classes and properties.

Another benefit of using OWL for metamodel annotations is the monotonicity of OWL, i.e., adding further axioms to a model does not falsify existing extailments. Thus, OWL provides a non-invasive way to integrate same or similar concepts over modeling languages (RQ3).

3.2 Extending MOF Models

In order to extend the expressiveness of Ecore metamodels, we use a annotation profile. In this paper, we consider Ecore as an implementation of EMOF for the Eclipse Modeling Framework (EMF) (Budinsky et al., 2003). In an Ecore-based metamodel, every element is an EObject and therefore can adopt EAnnotations (Budinsky et al., 2003). The process of building and extending ECore-based metamodels consists of three steps:

1. To build the metamodel by using the Ecore metametamodel.
2. To annotate the metamodel in order to make it more understandable for OWL annotations, which are created in step 3. This means that each element in a metamodel gets an annotation, so that it represents an ontology element. For example: EClasses are annotated by #Class to represent an OWL Class; EReferences are annotated by #ObjectProperty to represent an OWL Object Property, or EAttributes are annotated by #DataProperty to represent an OWL Data Property. This step can be accomplished automatically, since we have to iterate over all elements in the metamodel.
3. To create annotations to define additional constructs that usually are not expressable in Ecore-based metamodels. In general, we can create additional OWL annotations for Class Axioms, Object Property Axioms, Data Property Axioms, Class Expressions, Object Property Expressions and Data Property Expressions.

   For example, we can define the concepts of two classes as equivalent by creating a new annotation #EquivalentWith, which refers (by references) to those annotations of Ecore classes. Alternatively, we can define references as transitive, creating a #Transitive annotation which refers to the annotation of the reference that should be declared as transitive.

3.3 Mapping MOF and OWL

Ecore and OWL have a lot of similar constructs,e.g., classes, attributes and references. To extend the expressiveness of Ecore with OWL constructs, we need to establish mappings between the Ecore constructs and OWL constructs.

Based on these mapping, we develop a generic transformation script to transform any Ecore Metamodel/Model into OWL TBox/ABox – OWLizer.

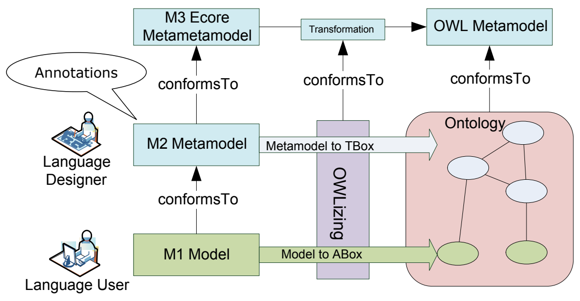

Figure 1: OWLizer.

Fig. 1 depicts the conceptual schema of transforming Ecore into OWL.

The four lanes, Actor, Ecore, Model Transformation and OWL show three modeling levels according to the OMGs Four layered metamodel architecture: the metametamodel level (M3), the metamodel level (M2) and the model level (M1). Vertical arrows denote instantiation whereas the horizontal arrows are transformations from the Ecore modeling space to the OWL modeling space. The ontology consists of the TBox that represents the metamodel and the ABox which expresses the model.

The language designer annotates the existing metamodel with Ecore annotations for OWL. These annotations work in the same way as the UML profile for OWL. Thus, the language designer can create new attributes and references and define axioms for them in OWL by using reference properties. It is possible to add further modeling axioms in OWL for the model elements, e.g., expressing two artifacts as identical.

A model transformation takes the metamodel and the annotations as input and generates an OWL ontology where the concepts, enumerations, properties and data types (TBox) correspond to classes, enumerations, attributes/references and data types in the metamodel. Another transformation takes the model created by the language user and generates individuals in the ABOX of the same OWL ontology. The whole process is transparent for language users.

The structural mapping from Ecore-based metamodels and models to OWL makes MOF models available as federated, accessible and query-ready LOD resources. Models can be transformed into a common representation in OWL ontologies according to this structural mapping. Having models represented in OWL ontologies, one might connect these ontologies and process these ontologies in a federated way. Thus, the resulting OWL representations address the requirement RQ3 (Integration Management: transform, link and query models) defined in Sect. 2.

3.4 Ontology Matching

In a model driven paradigm, resources expressed using distinct modeling languages must be reconciled before usage. In this paper, we illustrate some of the multiple ontology matching techniques. For a deeper understanding of this topic, please refer to (Euzenat and Shvaiko, 2007).

Ontology matching is the discipline responsible for studying techniques for reconciling multiple resources on the web. It comprises two steps: match and determine alignments and the generation of a processor for merging and transforming. Matching identifies the correspondences. A correspondence for two ontologies A and B is a quintuple including an id, an entity of ontology A, an entity of ontology B, a relation (equivalence, more general, disjointness) and a confidence measure. A set of correspondences is an alignment. Correspondences can take place at the schema-level (metamodel) and at the instance-level (model).

Matchings use multiple criteria: name of entities, structure (relations between entities, cardinality), background knowledge, e.g., existing ontologies or wordnet. Techniques can be string-based or rely on linguistic resources. Furthermore, matchings are established according to the structures that are compared: (i) Internal structure comparison: this includes property, key, datatype, domain and multiplicities comparison. (ii) Relational structure comparison: the taxonomic structure between the ontologies is compared. (iii) Extensional techniques: extensional information is used in this method, e.g., formal concept analysis

The correspondences depend on the applied criteria and technique. For example, if we apply string matching, it generates a false positive correspondence between the UML Activity and the BPMN Activity. However, if we apply structure-based techniques and analyze the structure of the UML class Action and the BPMN class Activity, we see that both have similar structures (both have one superclass with two associations with the same cardinalities). However, the UML class Action is abstract and the BPMN class Activity is concrete. So, we could assert that the class Activity is a subclass of class Action.

Automatic matching techniques should be assisted by domain experts because of existence of false positive matches. For example, the correspondence between BpmnDiagram and UML Activity is hard to catch automatically. With the ontology matching capabilities, we address the requirements RQ2 (Identification) and RQ3 (Integration Management: transform, link and query models) by identifying correspondences for creating links between (meta) models.

3.5 Querying MOF-models

SPARQL is the W3C standard query language for RDF graphs, which is a triple-based language (Prud'hommeaux and Seaborne, 2007). Writing SPARQL queries for OWL can be time-consuming for those who work with OWL ontologies, since OWL is not triple-based and requires reification of axioms when using a triple-based language. Therefore, we proposed SPARQLAS, a language that allows for specifying expressions that rely on inferences over OWL class descriptions (Schneider, 2010). It is a seamless extension of the OWL syntax for querying OWL ontologies. SPARQLAS enables using variables wherever an entity (Class, Datatype, ObjectProperty, DataProperty, NamedIndividual) or a literal is allowed.

SPARQLAS queries are translated into SPARQL queries and executed by any SPARQL engine that supports graph pattern matching for OWL2 entailment regime (Glimm and Parsia, 2010). SPARQLAS queries operate on both the schema level (M2 metamodeling level) and on the instance level (M1 modeling level). For example, Listing 2 show a SPARQLAS query about use cases that include other use cases. In this example, we ask about the individuals ?x whose type is an anonym class where the transitive property includeUseCase has as value some use case.

Listing 2: Use Cases that includes some other use case.

```
Namespace: uml = <...uml2/3.0.0/UML#>
Select ?x
Where:
    ?x type (UseCase and
         includeUseCase some UseCase)
```

With SPARQLAS, we cover requirement RQ3 (Integration Management: transform, link and query models) by providing distributed query facilities for models and metamodels represented in OWL.

3.6 Proof of Concept

We have realized our approach in the TwoUse Toolkit and it is available for download on the project web site[1]. The TwoUse toolkit uses the Eclipse Modeling Framework (Steinberg et al., 2009) and comprises (a) a set of model transformations, (b) a set of textual and graphical editors, and (c) a set of reasoning services, e.g., classification, realization, query answering and explanation.

[1]http://twouse.googlecode.com/.

3.7 Limitations

Depending on the strategy for matching and aligning ontologies there is the possibility of false positive matches. For example, if we use a string based matching technique, we match OWL classes with the same name as equivalent, although the two concepts might not be semantically the same. Thus, domain experts must be involved to proof the results of matching and alignments.

A further limitation of our approach regards the quality of metamodels. There are metamodels that are designed for generating code. We have seen that, in order to ease the implementation, references in ECore can be declared as containment (instead of implementing reference resolvers). This leads to the situation where multiple objects represent the same model elements. Here ontology matching techniques are powerless, since models and the ontology extracted from these models consist of elements having same names or properties.

A further issue in evaluating our approach is the missing implementations of technical spaces. In our case, all implementations are under the roof of Ecore, an implementation of EMOF. In the future we would like to do similar tests under (C)MOF.

Moreover, linking and aligning metamodels and models means firstly transforming (owlizing) everything into one ontological representation and secondly processing ontology matching algorithms, which can cause performance issues.

4 RELATED WORK

The integration of software artifacts has been the topic of works including (Antoniol et al., 2005; Mockus and Herbsleb, 2002). However, these approaches presented dedicated extractors for specific systems,e.g., bug tracking and version control but not for software models. Moreover, neither of these approaches presents formats for publishing data suitable to the linked-data approach, i.e. they do not share the principles of interoperability for connecting federated software models across the Web.

Kiefer et al. (Kiefer et al., 2007) and Iqbal et al. (Iqbal et al., 2009) explored semantic web approaches for transforming software artifacts such as data from version control systems, bug tracking tools and source code into linked data. Both approaches use artifact-specific extractors and thus work only for a fixed number of software artifacts. We propose a generic approach for transforming and managing any MOF metamodel in a web format.

The OMG ontology definition metamodel (OMG, 2009a) specifies mappings between OWL and UML. In this paper, we present a general approach for mapping arbitrary MOF models into OWL. We provide the means to express any MOF metamodel in its equivalent OWL.

The OMG Request For Proposal for MOF to RDF Structural Mapping in support of Linked Open Data (OMG, 2009b) aims at defining a structural mapping between OMG-MOF models and RDF. This work is a response to this request. We propose an approach to be used as a benchmark for future proposals.

5 CONCLUSIONS

In this paper, we propose an approach to enable analysis, federation, querying and rationalization of models expressed in MOF compliant languages, including OMG standards and domain-specific languages. We demonstrate how our approach addresses the requirements of an architecture ecosystem (OMG, 2009b).

The contribution in this paper shows that the usage of the Ontology Web Language for specifying metamodels is a viable solution to achieve interoperability and shared conceptualizations. The role of OWL is not to replace MOF or the Object Constraint Language because OWL addresses distinct requirements, specially concerning networked environments. OWL should compose the spectrum of software modeling languages in a unified architecture.

REFERENCES

Antoniol, G., Penta, M. D., Gall, H., and Pinzger, M. (2005). Towards the integration of versioning systems, bug reports and source code meta-models. *Electr. Notes Theor. Comput. Sci.*, 127(3):87–99.

Antoniou, G. and vanHarmelen, F. (2004). *A Semantic Web Primer*. MIT Press, Cambridge, MA, USA.

Bizer, C., Heath, T., Idehen, K., and Berners-Lee, T. (2008). Linked data on the web (ldow2008). In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1265–1266, New York, NY, USA. ACM.

Budinsky, F., Brodsky, S. A., and Merks, E. (2003). *Eclipse Modeling Framework*. Pearson Education.

Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*. Springer.

Gasevic, D., Djuric, D., Devedzic, V., and Damjanovi, V. (2004). Converting uml to owl ontologies. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 488–489. ACM.

Glimm, B. and Parsia, B. (2010). SPARQL 1.1 Entailment Regimes. Working draft 26 january 2010, W3C. Available on http://www.w3.org/TR/sparql11-entailment/.

Iqbal, A., Ureche, O., Hausenblas, M., and Tummarello, G. (2009). Ld2sd: Linked data driven software development. In *Proceedings of the SEKE'2009, Boston, Massachusetts, USA, July 1-3, 2009*, pages 240–245. Knowledge Systems Institute Graduate School.

Kiefer, C., Bernstein, A., and Tappolet, J. (2007). Mining software repositories with isparql and a software evolution ontology. In *ICSEW '07: Proceedings of the 29th International Conference on Software Engineering Workshops*, Washington, DC, USA.

Mockus, A. and Herbsleb, J. D. (2002). Expertise browser: a quantitative approach to identifying expertise. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 503–512, New York, NY, USA. ACM.

OMG (2009a). *Ontology Definition Metamodel*. Object Modeling Group.

OMG (2009b). *Request For Proposal MOF to RDF Structural Mapping in support of Linked Open Data*. Object Modeling Group. Available at http://www.omg.org/cgi-bin/doc?ad/2009-12-09.

Prud'hommeaux, E. and Seaborne, A. (2007). SPARQL query language for RDF (working draft). Technical report, W3C.

Schneider, M. (2010). Sparqlas: Writing sparql queries in owl syntax. Bachelor thesis, University of Koblenz-Landau. German.

Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2009). *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional.

W3C OWL Working Group (2009). OWL 2 Web Ontology Language Document Overview. Technical report. Available at http://www.w3.org/TR/2009/WD-owl2-overview/.