Verifying and Validating Multi-layered Models with OWL FA Toolkit

Nophadol Jekjantuk[1], Jeff Z. Pan[1], and Gerd Gröner[2]

[1] University of Aberdeen, United Kingdom
[2] University of Koblenz-Landau, Germany

Abstract. This paper details the use of OWL FA Toolkit for verifying and validating multi-layered (meta-) modelling using ontologies described in OWL FA. We will show how OWL FA and its reasoner (OWL FA Toolkit) could benefit the software modeller on leveraging the software development life cycle through a practical use case.

1 Introduction

Metamodelling, i.e. modelling across multiple modelling layers, dealing with concepts and meta-concepts, is a key issue in model management and especially in model-driven software development (MDSD). Metamodels appear in application areas such as UML [14], Model Driven Architecture [4] and E-Commerce.

Ontologies and the Web Ontology Language (OWL) are well established for model descriptions and model management tasks. However, the standard OWL Web Ontology Language does not support modelling and reasoning over a layered metamodelling architecture. OWL 2 provides simple metamodelling which corresponds to the contextual semantics defined in [9]. However, this modelling technique is mainly based on punning. It has been shown in [12] that this can lead to non-intuitive results, since the interpretation function is different based on the context.

Indeed, for many applications, a validation without a metamodel is not adequate. There are various works which consider validation of UML models with OCL constraints like in [3,5,7,13,8]. However, none of these approaches account for a validation across multiple layers, i.e. validate models with respect to their metamodels. Although they validate models with model constraints and instances of the models, they do not account for metamodels in their validation. We intent to show how OWL FA Toolkit could be used to help close the gap.

2 Motivating Example

This section gives an example to demonstrate the need for metamodelling enabled ontologies. Models are depicted in UML notations. Metamodels are more than a syntactic language description of a modelling language; a metamodel is a description of the concepts of a modelling language specifying the structure and the kind of information that can be handled [10].

L. Aroyo et al. (Eds.): ESWC 2010, Part II, LNCS 6089, pp. 391–395, 2010.

Models and metamodels are commonly used in model-driven software engineering (MDSE). In order to improve software development processes, new technologies which provide reasoning support like consistency checking of models and metamodels are beneficial. In MDSE, each model layer can contain both class and object definitions. However, this leads to undecidability problems in model validation w.r.t the complexity of the model. In ontology engineering, ontologies for metamodelling like OWL FA separates class and object into different layers in order to maintain the decidability of the language.

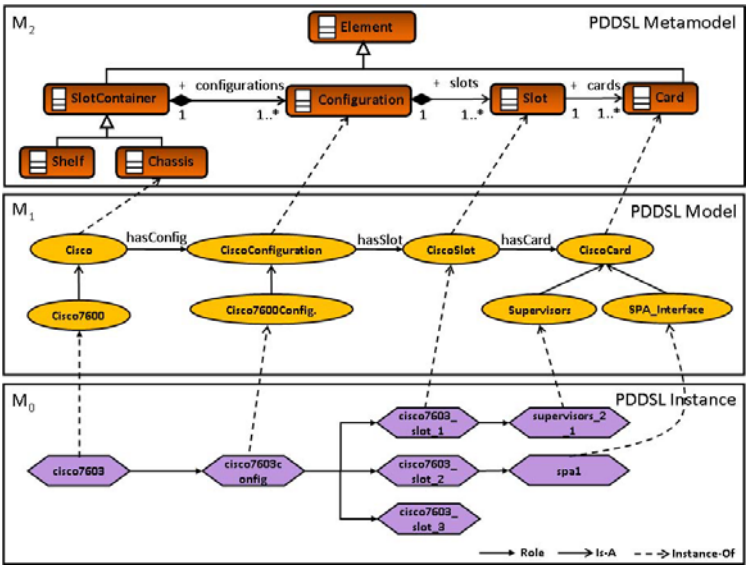

Fig. 1. Layered Architecture for a Physical Device Model

In Fig. 1, a layered modelling architecture is demonstrated for physical device modelling (an application of configuration management). The figure depicts three layers M_0, M_1 and M_2. M_3 is a Meta-metamodel layer which is not included in the example. The arrows between the layers demonstrate instance relationships, the arrows within a layer are concept relations (like object properties and subclass hierarchies). Moreover, the modeller requires that the concepts Chassis and Shelf in M_3 have to be disjoint from each other.

A crucial task in model-driven engineering is the validation of models and metamodels [3,5,7,13,8]. A valid model refers to its metamodel and satisfies all the restrictions and constraints. However, the validation of multiple layers may lead to inconsistency even if the consistency is satisfied between all adjacent layers, i.e. model and instance layer. For instance, in the previous described scenario, if one would like to add Avaya is an instance of Shelf in M_2 and Avaya, considered as a concept in M_1, is equivalent to the concept Cisco in M_1. Combined with the constraints on the model layer M_2 which requires the disjointness of

class Shelf and Chassis leads to a contradiction and therefore to an inconsistent ontology. Without capturing multiple layers, this inconsistency is not detected since Avaya is either considered as a concept on layer M_1 or as an instance. However, modelling with more than two layers has to account for concepts on layer M_i that are at the same time instances of concepts of the layer M_{i+1}.

3 Technical Background

OWL FA [12] enables metamodelling. It is an extension of OWL DL, which refers to the description logic $\mathcal{SHOIN}(\mathcal{D})$. Ontologies in OWL FA are represented in a layered architecture. This architecture is mainly based on the architecture of RDFS(FA) [11]. OWL FA specifies a layer number in class constructors and axioms to indicate the layer they belong to. For example, `SubClassOf(Annotation (Layer "2"^^xsd:int) Chassis SlotContainer)` belongs to layer 2. Since OWL FA is based on OWL DL all language features from OWL DL are also available in OWL FA in contrast of RDFS(FA) like transitive `TransitiveProperty` and inverse `owl:inverseOf` property restrictions.

The syntax is adopted from OWL DL. The semantics of two layers which can be considered as TBox and ABox are same as in OWL DL. The idea of OWL FA is that the interpretation depends on the layer but is still an OWL DL interpretation. Therefore in each layer, or from one layer to the next layer standard OWL DL reasoning capabilities can be used. For more details about the reasoning in OWL FA refer to [6].

4 OWL FA Toolkit

In this section, we introduce the OWL FA Toolkit, a simple graphic user interface for modeller to create an OWL FA ontology and perform reasoning over it. The OWL FA Toolkit contains features as following:

- Editor - for checking the OWL FA ontology before perform the reasoning. In this version it supports only functional syntax.
- Ontology Consistency - for checking whether a given multi-layered models is valid. This enables modeller to validate models with their metamodels.
- Concept Satisfiability - for verifying whether a concept A is a non-empty set in a given OWL FA ontology $\mathcal{O}$. A modeller can verify a particular concept that might leads to contradiction of the model.
- Query Answering - for accessing information form a given multi-layered models by using SPARQL query.
- Reasoning with OWL DL ontologies - for reasoning with a multi-layered models that described in OWL DL syntax. A modeller can easily create model ontology and metamodel ontology separately.

Figure 2 shows screen capture of OWL FA Toolkit loaded OWL FA ontology for validating multi-layered models. More details about the OWL FA Toolkit are described in [6].

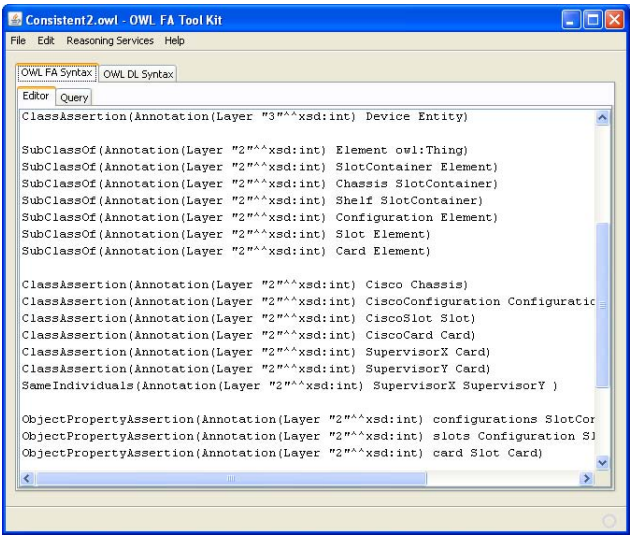

Fig. 2. OWL FA Toolkit

5 Related Work

In [15] spanning objects are used in order to have different interpretations for objects that are instances and classes simultaneously. Compared to OWL FA one spanning object refers to one ontology $\mathcal{O}_i$.

A UML diagram with OCL constraints is transformed to a constraint satisfaction problem. In [13] UML/OCL models are modelled with the constraint language Z in order to validate class diagrams. The language Alloy is used in [1] as a representation of UML/OCL models. The Alloy Analyser verifies the model properties. Constraint logic programming (CLP) is applied in [8] to validate UML models and model constraints. Also the metamodels are translated to CLP and validated based on defined metamodel specifications. However, properties of a layer are not considered in the next layer.

Berardi et al. [3] apply DL Reasoning to UML class diagrams. The expressiveness of UML diagrams and constraints are restricted to the expressiveness of the DL $\mathcal{ALC}^-$. Basic conceptual modelling including model constraints is demonstrated for UML diagrams in OWL The consistency check of a UML class diagram is then reduced to concept satisfiability in $\mathcal{ALC}^-$. However, the verification is only performed on the conceptual level, without accounting for a metamodelling architecture.

First-order logic (FOL) is used in [7] for consistency checks of UML class diagrams. The main contribution are different algorithms to perform the consistency check and the analysis of inconsistency triggers. The transformation from UML class diagrams with OCL constraints to FOL is also described in [2] in order to enable consistency check.

6 Conclusion

This paper focuses on showing how to use OWL FA Toolkit for verifying and validating multi-layered models. We also compare our tool with OWL 2 modelling and reasoning, since OWL 2 is the only OWL language that supports metamodelling and has tool support. For demonstration purposes, we use some case studies from the MOST project (http://www.most-project.eu), in particular, the physical device modelling (an application of configuration management). We will show how OWL FA and its reasoner (OWL FA Toolkit) could benefit software modellers on leveraging the software development life cycle.

References

1. Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: UML2Alloy: A challenging model transformation. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 436–450. Springer, Heidelberg (2007)
2. Beckert, B., Keller, U., Schmitt, P.H., et al.: Translating the Object Constraint Language into first-order predicate logic. In: Proceedings, VERIFY, Workshop at Federated Logic Conferences (FLoC), Copenhagen, Denmark, Citeseer (2002)
3. Berardi, D., Calvanese, D., De Giacomo, G.: Reasoning on UML class diagrams. Artificial Intelligence 168(1-2), 70–118 (2005)
4. Brown, A.: An introduction to Model Driven Architecture. IBM Technical Report (2004), `http://www-128.ibm.com/developerworks/rational/library/3100.html`
5. Cabot, J., Clariso, R., Riera, D.: Verification of UML/OCL Class Diagrams using Constraint Programming. In: Software Testing Verification and Validation Workshop, pp. 73–80 (2008)
6. Jekjantuk, N., Gröner, G., Pan, J.Z.: Reasoning in Metamodeling Enabled Ontologies. In: Proceeding of the International Workshop on OWL: Experience and Directions, OWL-ED 2009 (2009)
7. Kaneiwa, K., Satoh, K.: Consistency checking algorithms for restricted UML class diagrams. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 219–239. Springer, Heidelberg (2006)
8. Malgouyres, H., Motet, G.: A UML Model Consistency Verification Approach based on Meta-Modeling Formalization. In: SAC 2006: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 1804–1809. ACM, New York (2006)
9. Motik, B.: On the properties of metamodeling in owl. J. Log. Comput. 17(4), 617–637 (2007)
10. Ober, I., Prinz, A.: What do we need metamodels for?
11. Pan, J.Z., Horrocks, I.: RDFS(FA) and RDF MT: Two Semantics for RDFS. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 30–46. Springer, Heidelberg (2003)
12. Pan, J.Z., Horrocks, I., Schreiber, G.: OWL FA: A Metamodeling Extension of OWL DL. In: Proceeding of the International Workshop on OWL: Experience and Directions, OWL-ED 2005 (2005)
13. Roe, D., Broda, K., Russo, A.: Department of Computing. Mapping UML models incorporating OCL constraints into Object-Z. In: Imperial College of Science, Technology and Medicine, Department of Computing (2003)
14. UML. Unified Modeling Language, `http://www.uml.org/`
15. Welty, C.A., Ferrucci, D.A.: What's in an instance? Technical report, RPI Computer Science (1994)