

Modelling and Validating Multilevel Models with OWL FA

Nophadol Jekjantuk¹, Gerd Gröner², Jeff Z. Pan¹, and Yuting Zhao¹

¹ University of Aberdeen, United Kingdom

² University of Koblenz-Landau, Germany

Abstract. Ontologies and its reasoning services are expected to play an important role in many application domains, as well as in software engineering in general. In model-driven engineering (MDE), models, like UML models, represent and specify software systems. One problem with using ontologies within software engineering is that while model-driven engineering realises a four-layer metamodelling architecture, 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. However, this can lead to non-intuitive results. In this paper, we demonstrate multilevel (meta-) modelling using ontologies described in OWL FA, which has a well defined fixed-layered architecture and semantics. We present an approach for modelling and validating multilevel models in OWL FA.

1 Introduction

Ontologies and the Web Ontology Language (OWL) are well established for model descriptions and model management tasks. However, metamodelling is not supported by tractable OWL sublanguages like OWL Lite, OWL DL. More complex languages like OWL Full provide metamodelling facilities, but OWL Full is not decidable. Therefore, no tractable reasoning support is available. The restrictions of the decidable languages either impede metamodelling or restrict the modelling possibilities, so that it is not possible to validate models with respect to their metamodels.

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 (MDS). Metamodels appear in application areas such as UML [22], Model Driven Architecture [7] and E-Commerce. In model-driven software development, software developers would like to improve their software development processes by using reasoning mechanisms such as inconsistency checking of models. In order to do so they need to transform the software models which have a well defined four-layered architecture into an OWL DL ontology which does not support modelling and reasoning over a layered architecture. In OWL DL, we can present classes and objects in two layers. Therefore, developers need to sacrifice some meta-layers or compress it into two

layers which lead to an incomplete representation of the model. Thus, this could be the main reason why software developers ignore to use OWL ontologies and their mechanisms.

A common user complaint about OWL is that it does not provide a decidable sub-language which supports metamodelling. OWL 2 provides simple metamodelling which corresponds to the contextual semantics defined in [15]. However, this modelling technique is mainly based on punning. It has been shown in [19] that this can lead to non-intuitive results, since the interpretation function is different based on the context.

There are various works which consider validation of UML models with OCL constraints like in [6, 8, 12, 20, 14] (cf. Section 8). However, none of these approaches account for a validation across multiple layers, i.e. validate models with respect to their metamodels. They validate models with model constraints and instances of the models, but they do not account for metamodels in their validation. However, for many applications, a two-layer validation without a metamodel is not adequate (cf. Section 2).

There are different categorisation of model consistency and consistency checking as specified in [21]. The focus of our approach is to provide an ability for OWL on validation and consistency checking of models covering multiple modelling layers. This is a generic approach and can be applied in metamodelling and multi-dimensional metamodelling like mega-modelling [10] for linguistic and ontological metamodelling (cf. [13, 4]). Our approach is demonstrated on an example of four-layer linguistic (physical) metamodelling. The main contribution of OWL FA is to enable modelling in OWL with more than the standard two layers in order to handle the class/object duality [3], independent whether linguistic, ontological or combined metamodelling is considered.

In this paper, we use OWL FA [19] to validate models covering multiple modelling layers. Obviously, this requires the consideration of constraints in both models and metamodels. The paper is organised as follows. Section 2 outlines a metamodelling application which demonstrates dependencies between multiple layers. The need and requirement is described in Section 3. The syntax and semantics of OWL FA is given in Section 4. In Section 5 we show the way to model a multilevel metamodelling architecture with metamodelling enabled ontology, followed by an evaluation from experimental results in section 6. Then, we compare our approach with existing approaches in Section 7. Finally, we discuss about the future direction of OWL FA.

2 A Motivating Example

This section gives two examples to demonstrate the need for metamodelling enabled ontologies. Models are depicted in UML-like notations [1]. 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 [17].

Example 1. 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 (cf. [3]). However, this leads to undecidability problems in model validation w.r.t the complexity of the model. In ontology engineering, ontologies for metamodeling like OWL FA separate classes and objects into different layers in order to maintain the decidability of the language. In the rest of the paper, we use ontologies for metamodeling to improve software development processes by validating, consistency checking and classifications of models with respect to their metamodels.

In order to use these benefits, a transformation of software models and metamodels which are mainly graphical models to OWL DL ontologies is not sufficient, since OWL DL does not support reasoning in a layered modelling architecture. This may introduce incomplete meaning of the model or does not provide a valid semantic base at all. OWL FA provides more expressive power for metamodeling and the semantics of OWL FA are well defined.

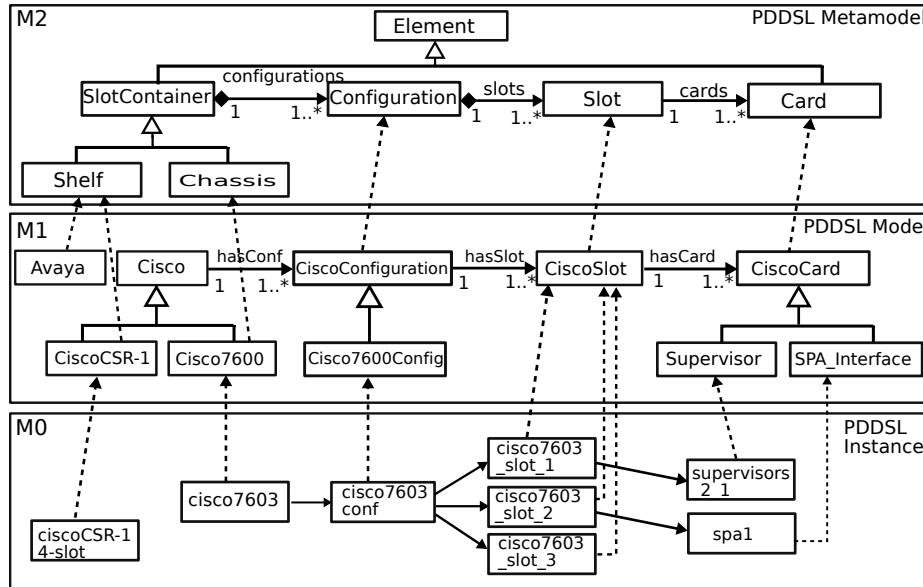


Fig. 1. Layered Architecture for a Physical Device Model

In Figure 1, a layered modelling architecture is demonstrated for physical device modelling (an application of configuration management). A physical device domain specific language (PDDSL) is a domain specific language (DSL) for physical devices which is used in business IT system modelling. The figure

depicts three layers M_0 , M_1 and M_2 . M_3 is a Meta-metamodelling 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).

`Cisco`, `CiscoConfiguarion`, `CiscoSlot` and `CiscoCard` are instances of some classes from the M_2 layer. At the same time, these artifacts are concepts in the layer M_1 and each of these concepts can contain some instance from layer M_0 . The rest of the relationships in a layer like concept subsumption and object properties are represented by the arrows.

There are additional modelling constraints imposed on different layers. In layer M_2 , a model designer requires that each `SlotContainer` has a `Configuration`, expressing that the `ObjectProperty configurations` is a functional property. Each `Configuration` may contain `Slot`, which is expressed by the `ObjectProperty slots` as a functional property. And each `Slot` may contain `Card`, this is also expressed as `ObjectProperty cards`. Moreover, the modeller requires the disjointness of the two classes `Chassis` and `Shelf`. There are also three `ObjectProperty` assertions from the layer M_2 to M_1 : `configuration(Cisco, CiscoConfiguration)`, `slots(CiscoConfiguration, CiscoSlot)` and `cards(CiscoSlot, CiscoCard)`.

In layer M_1 a model designer requires that each `Cisco` have at least one `Configuration`. Each `CisoConfiguration` may contain one or more `CiscoSlot` and each `Slot` may contain `CiscoCard`. `hasConfig`, `hasSlot` and `hasCard` are expressed as `ObjectProperty`.

Additionally, in a concrete model in M_1 , the modeller requires the disjointness of the classes `Supervisor` and `SPA_interface`. Moreover, `Cisco7600Config`. can have only three `CiscoSlot` and `Cisco7600_Slot.1` can contain only card from `Supervisor` class.

A crucial task in model-driven engineering is the validation of models and metamodels. 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.

For instance in the previous described scenario, the model on M_1 and also the corresponding metamodel on layer M_2 are consistent. The inconsistency occurs when they are combined, i.e. consider the modelling restrictions like equivalence or disjointness for the whole model, covering multiple layers simultaneously instead of only two adjacent layers. If one would like to add `Avaya` is a `Shelf` and `Avaya` is equivalent to concept `Cisco` in M_1 . Here, `Avaya` and `Cisco` are concepts in layer M_1 . This equivalence condition does not cause any inconsistency of the modelling layer (M_1 and M_0 for instances). However, combined with the constraints on the metamodel layer M_2 which requires the disjointness of `Shelf` and `Chassis`, this leads to a contradiction and therefore to an inconsistent ontology. Without capturing multiple layers, this inconsistency is not detected since the adjacent layers M_1 , M_0 and M_2 , M_1 are consistent on its own.

Example 2. A difficult task in model-driven software development is the integration or weaving of two different models, for instance the integration of two UML models. Both models are instances of the same metamodel. Both models

remain valid models, but the target is to define mappings between these models. In order to define mappings between the entities of both models, a common metamodel is required which defines concept properties and their relations. The concepts (entities) of the different models are connected based on the definition in the metamodel which are satisfied by these concepts.

There are different specialisations of mappings like model transformation and model refinements, e.g. from a platform independent model to a platform specific model in model-driven software engineering. All these mapping tasks are specified with respect to their corresponding metamodel(s) and metamodel constraints. It is obvious that in all these applications more than two layers are involved.

3 User Requirement

In this section, we describe the requirements for the PDDSL modelling architecture from Section 2 for physical devices from a PDDSL user point of view. The PDDSL user models physical devices and their configurations in layer M_1 . In usual OWL (-DL) conceptual two-layer models, this could be the TBox in order to account for modelling and reasoning in the modelling layer and instance layer. Another possibility is to represent the M_1 layer as ABox and the metamodel layer M_2 in the TBox. In this case, the user benefits from modelling and reasoning support for the models in the modelling layer with respect to the metamodels. However, it is not possible to combine modelling and reasoning support covering more than two layers, i.e. to combine both of the described modelling forms. With OWL FA, the user is able to consider the instance layer below and simultaneously the metamodel layer above in order to profit from a comprehensive modelling framework in OWL.

For modelling of physical network devices in PDDSL, the user (PDDSL modeller) requires the following modelling support.

- Planning of Network: The modelling frameworks enable restrictions and suggestions of components that can be used in the current model configuration. These restrictions and suggestions are based on the corresponding metamodel in layer M_2 .
- Consistency checking of devices and configurations: The consistency of the devices and configurations which are modelled in layer M_1 are checked and validated with respect to the corresponding metamodels. The metamodels are defined in the layer M_2 .
- Data quality analysis: The user checks whether a configuration on M_0 is instance of a modelled configuration on M_1 , or for a given configuration in M_0 the most specific configuration model in M_1 is searched.

The realisation of these service requirements depends on the modelling possibilities and on the reasoning services that are available for the model. To realise a single requirement, a two-layered model is appropriate. However, if more of these requirements have to be realised simultaneously, two layers are not enough.

For instance, we consider the second and third requirement. The consistency of a model is checked with respect to its metamodel in which all model constraints are defined. In this case, the layer above has to be considered in order to check the consistency of the modelling layer M_1 . The analysis of the data quality covers the layer M_1 and the instance layer M_0 . In this case, the user is interested in finding a configuration model for a configuration instance. The user might be interested for instance in finding the most specific or the most general model. In order to satisfy both requirements, at least three layers might be useful for modelling and reasoning.

4 OWL FA syntax and semantics

OWL FA [19] enables metamodelling. It is an extension of OWL DL, which refers to the description logic $\mathcal{S}\mathcal{H}\mathcal{O}\mathcal{I}\mathcal{N}(\mathcal{D})$. Ontologies in OWL FA are represented in a layered architecture. This architecture is mainly based on the architecture of RDFS(FA) [18].

OWL FA specifies a stratum number in class constructors and axioms to indicate the strata they belong to. Let $i \geq 0$ be an integer. OWL FA consists of an alphabet of distinct class names \mathbf{V}_{C_i} (for stratum i), datatype names \mathbf{V}_D , abstract property names \mathbf{V}_{AP_i} (for stratum i), datatype property names \mathbf{V}_{DP} and individual (object) names (\mathbf{I}); together with a set of constructors (with subscriptions) to construct class and property descriptions (also called *OWL FA-classes* and *OWL FA-properties*, respectively).

The semantics of OWL FA are a model theoretic semantics, which is defined in terms of interpretations. Given an OWL FA alphabet \mathbf{V} , a set of built-in datatype names $\mathbf{B} \subseteq \mathbf{V}_D$ and an integer $k \geq 1$, an *OWL FA interpretation* is a pair $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, where $\Delta^{\mathcal{J}}$ is the domain (a non-empty set) and $\cdot^{\mathcal{J}}$ is the interpretation function, which satisfy the the following conditions below (where $0 \leq i \leq k$):

1. $\Delta^{\mathcal{J}} = \bigcup_{0 \leq i \leq k-1} \Delta_{A_i}^{\mathcal{J}} \cup \Delta_{\mathbf{D}}$, where $\Delta_{A_i}^{\mathcal{J}}$ is the domain for stratum i and $\Delta_{\mathbf{D}}$ is the datatype domain;
2. $\Delta_{A_{i+1}}^{\mathcal{J}} = 2^{\Delta_{A_i}^{\mathcal{J}}} \cup 2^{\Delta_{A_i}^{\mathcal{J}} \times \Delta_{A_i}^{\mathcal{J}}}$ and $\Delta_{\mathbf{D}} \cap \Delta_{A_i}^{\mathcal{J}} = \emptyset$;
3. $\forall \mathbf{a} \in \mathbf{V}_I : \mathbf{a}^{\mathcal{J}} \in \Delta_{A_0}^{\mathcal{J}}$ and $\forall \mathbf{C} \in \mathbf{V}_{C_{i+1}} : \mathbf{C}^{\mathcal{J}} \subseteq \Delta_{A_i}^{\mathcal{J}}$;
4. $\forall R \in \mathbf{V}_{AP_{i+1}} : R^{\mathcal{J}} \subseteq \Delta_{A_i}^{\mathcal{J}} \times \Delta_{A_i}^{\mathcal{J}}$ and $\forall T \in \mathbf{V}_{DP} : T^{\mathcal{J}} \subseteq \Delta_{A_0}^{\mathcal{J}} \times \Delta_{\mathbf{D}}$;
5. $\bigcup_{\forall d \in \mathbf{B}} V(d) \subseteq \Delta_{\mathbf{D}}$, where $V(d)$ is the value space of d ;
6. $\forall d \in \mathbf{V}_D$, if $d \in \mathbf{B}$, then³
 - (a) $d^{\mathcal{J}} = V(d)$, where $V(d)$ is the value space of d ,
 - (b) if $v \in L(d)$, then $(“v” \wedge d)^{\mathcal{J}} = L2V(d)(v)$, where $L(d)$ is lexical space of d and $L2V(d)$ is the lexical-to-value mapping of d ,
 - (c) if $v \notin L(d)$, then $(“v” \wedge d)^{\mathcal{J}}$ is undefined;
otherwise, $d^{\mathcal{J}} \subseteq \Delta_{\mathbf{D}}$ and $(“v” \wedge d) \in \Delta_{\mathbf{D}}$.

³ To simplify the presentation, we do not distinguish datatype names and datatype URIs here.

Constructor	DL Syntax	Semantics
top	\top_i	$\Delta_{A_{i-1}}^{\mathcal{J}}$
bottom	\perp	\emptyset
concept name	CN	$\text{CN}^{\mathcal{J}} \subseteq \Delta_{A_{i-1}}^{\mathcal{J}}$
general negation	$\neg_i C$	$\Delta_{A_{i-1}}^{\mathcal{J}} \setminus C^{\mathcal{J}}$
conjunction	$C \sqcap_i D$	$C^{\mathcal{J}} \cap D^{\mathcal{J}}$
disjunction	$C \sqcup_i D$	$C^{\mathcal{J}} \cup D^{\mathcal{J}}$
nominals	$\{o\}$	$\{o\}^{\mathcal{J}} = \{o^{\mathcal{J}}\}$
exists restriction	$\exists_i R.C$	$\{x \in \Delta_{A_{i-1}}^{\mathcal{J}} \mid \exists y. \langle x, y \rangle \in R^{\mathcal{J}} \wedge y \in C^{\mathcal{J}}\}$
value restriction	$\forall_i R.C$	$\{x \in \Delta_{A_{i-1}}^{\mathcal{J}} \mid \forall y. \langle x, y \rangle \in R^{\mathcal{J}} \rightarrow y \in C^{\mathcal{J}}\}$
atleast restriction	$\geq_i mR$	$\{x \in \Delta_{A_{i-1}}^{\mathcal{J}} \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{J}}\} \geq m\}$
atmost restriction	$\leq_i mR$	$\{x \in \Delta_{A_{i-1}}^{\mathcal{J}} \mid \#\{y \mid \langle x, y \rangle \in R^{\mathcal{J}}\} \leq m\}$
datatype exists restriction	$\exists_1 T.d$	$\{x \in \Delta_{A_0}^{\mathcal{J}} \mid \exists t. \langle x, t \rangle \in T^{\mathcal{J}} \wedge t \in d^{\mathcal{J}}\}$
datatype value restriction	$\forall_1 T.d$	$\{x \in \Delta_{A_0}^{\mathcal{J}} \mid \forall t. \langle x, t \rangle \in T^{\mathcal{J}} \rightarrow t \in d^{\mathcal{J}}\}$
datatype atleast restriction	$\geq_1 mT$	$\{x \in \Delta_{A_0}^{\mathcal{J}} \mid \#\{t \mid \langle x, t \rangle \in T^{\mathcal{J}}\} \geq m\}$
datatype atmost restriction	$\leq_1 mT$	$\{x \in \Delta_{A_0}^{\mathcal{J}} \mid \#\{t \mid \langle x, t \rangle \in T^{\mathcal{J}}\} \leq m\}$

Table 1. OWL FA classes

In the rest of the paper, we assume that i is an integer such that $1 \leq i \leq k$. The interpretation function can be extended to give semantics to OWL FA-properties and OWL FA-classes. Let $RN \in \mathbf{V}_{\text{AP}_i}$ be an abstract property name in stratum i and R an abstract property in stratum i . Valid OWL FA abstract properties are defined by the abstract syntax: $R ::= RN \mid R^-$, where for some $x, y \in \Delta_{A_{i-1}}^{\mathcal{J}}$, $\langle x, y \rangle \in R^{\mathcal{J}}$ iff $\langle y, x \rangle \in R^{-\mathcal{J}}$. Valid OWL FA datatype properties are datatype property names.

Let $\text{CN} \in \mathbf{V}_{C_i}$ be an atomic class name in stratum i , R an OWL FA-property in stratum i , $o \in \mathbf{I}$ an individual, $T \in \mathbf{V}_{\text{DP}}$ a datatype property name, and C, D OWL FA-classes in stratum i . Valid OWL FA-classes are defined by the abstract syntax:

$$\begin{aligned}
C ::= & \top_i \mid \perp \mid \text{CN} \mid \neg_i C \mid C \sqcap_i D \mid C \sqcup_i D \mid \{o\} \mid \exists_i R.C \\
& \forall_i R.C \mid \leq_i nR \mid \geq_i nR \\
& (\text{if } i = 1) \exists_1 T.d \mid \forall_1 T.d \mid \leq_1 nT \mid \geq_1 nT
\end{aligned}$$

The semantics of OWL FA-classes are presented in Table 1 (page 7).

C is *satisfiable* iff there exist an interpretation \mathcal{J} s.t. $C^{\mathcal{J}} \neq \emptyset$; C subsumes D iff for every interpretation \mathcal{J} we have $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$.

An OWL FA ontology \mathcal{O} consists of $\mathcal{O}_1, \dots, \mathcal{O}_k$. Each \mathcal{O}_i consists of a TBox \mathcal{T}_i , an RBox \mathcal{R}_i and an ABox \mathcal{A}_i . An OWL FA *TBox* \mathcal{T}_i is a finite set of class inclusion axioms of the form $C \sqsubseteq_i D$, where C, D are OWL FA-classes in stratum i . An interpretation \mathcal{J} satisfies $C \sqsubseteq_i D$ if $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$. Let R, S be OWL FA abstract properties in stratum i . An OWL FA *RBox* \mathcal{R}_i is a finite set of property axioms; namely, property inclusion axioms ($R \sqsubseteq_i S$), functional property axioms ($\text{Func}_i(R)$) and transitive property axioms ($\text{Trans}_i(R)$). An interpretation \mathcal{J} satisfies $R \sqsubseteq_i S$ if $R^{\mathcal{J}} \subseteq S^{\mathcal{J}}$; \mathcal{J} satisfies $\text{Func}_i(R)$ if, for all $x \in \Delta_{A_{i-1}}^{\mathcal{J}}$, $\#\{y \in \Delta_{A_{i-1}}^{\mathcal{J}} \mid \langle x, y \rangle \in R^{\mathcal{J}}\} \leq 1$ ($\#$ denotes cardinality); \mathcal{J} satisfies

$\text{Trans}_i(R)$ if, for all $x, y, z \in \Delta_{A_{i-1}^{\mathcal{J}}}$, $\{\langle x, y \rangle, \langle y, z \rangle\} \subseteq R^{\mathcal{J}} \rightarrow \langle x, z \rangle \in R^{\mathcal{J}}$. The semantics for datatype property inclusion axioms and functional axioms can be defined in the same way as those in OWL DL. Like in OWL DL, there is no transitive datatype property axioms.

Let $\mathbf{a}, \mathbf{b} \in \mathbf{I}$ be individuals, C_1 a class in stratum 1, R_1 an abstract property in stratum 1, l a literal, $T \in \mathbf{V}_D$ a datatype property, X, Y classes or abstract properties in stratum i , E a class in stratum $i+1$ and S an abstract property in stratum $i+1$. An OWL FA *ABox* \mathcal{A}_1 is a finite set of individual axioms of the following forms: $\mathbf{a} :_1 C_1$, called *class assertions*, $\langle \mathbf{a}, \mathbf{b} \rangle :_1 R_1$, called *abstract property assertions*, $\langle \mathbf{a}, l \rangle :_1 T$, called *datatype property assertions*, $\mathbf{a} = \mathbf{b}$, called *individual equality axioms* and $\mathbf{a} \neq \mathbf{b}$, called *individual inequality axioms*. An interpretation \mathcal{J} satisfies $\mathbf{a} :_1 C_1$ if $\mathbf{a}^{\mathcal{J}} \in C_1^{\mathcal{J}}$; it satisfies $\langle \mathbf{a}, \mathbf{b} \rangle :_1 R_1$ if $\langle \mathbf{a}^{\mathcal{J}}, \mathbf{b}^{\mathcal{J}} \rangle \in R_1^{\mathcal{J}}$; it satisfies $\langle \mathbf{a}, l \rangle :_1 T$ if $\langle \mathbf{a}^{\mathcal{J}}, l^{\mathcal{J}} \rangle \in T^{\mathcal{J}}$; it satisfies $\mathbf{a} = \mathbf{b}$ if $\mathbf{a}^{\mathcal{J}} = \mathbf{b}^{\mathcal{J}}$; it satisfies $\mathbf{a} \neq \mathbf{b}$ if $\mathbf{a}^{\mathcal{J}} \neq \mathbf{b}^{\mathcal{J}}$. An OWL FA *ABox* \mathcal{A}_i is a finite set of axioms of the following forms: $X : E$, called *meta-class assertions*, $\langle X, Y \rangle : R$, called *meta-property assertions*, or $X =_{i-1} Y$, called *meta individual equality axioms*. An interpretation \mathcal{J} satisfies $X : E$ if $X^{\mathcal{J}} \in E^{\mathcal{J}}$; it satisfies $\langle X, Y \rangle : R$ if $\langle X^{\mathcal{J}}, Y^{\mathcal{J}} \rangle \in R^{\mathcal{J}}$; it satisfies $X =_{i-1} Y$ if $X^{\mathcal{J}} = Y^{\mathcal{J}}$.

An interpretation \mathcal{J} satisfies an ontology \mathcal{O} if it satisfies all the axioms in \mathcal{O} . \mathcal{O} is *satisfiable* (*unsatisfiable*) iff there exists (does not exist) such an interpretation \mathcal{J} that satisfies \mathcal{O} . Let C, D be OWL FA-classes in stratum i , C is *satisfiable* w.r.t. \mathcal{O} iff there exist an interpretation \mathcal{J} of \mathcal{O} s.t. $C^{\mathcal{J}} \neq \emptyset$; C subsumes D w.r.t. \mathcal{O} iff for every interpretation \mathcal{J} of \mathcal{O} we have $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$.

5 Representing Multilevel Models with Ontologies

In this section, we present the way of expressing multiple-layered model with OWL FA. Although the layer numbers can/should be encapsulated by tools, there are two rules of thumb to help users to get the number right. Firstly, the subscript numbers are only used to indicate a sub-ontology (e.g. \mathcal{O}_2), a constructor (e.g. \exists_2), or axiom symbols (e.g. \sqsubseteq_2 , $:_2$) in a sub-ontology. Secondly, subscript numbers for constructors and axiom symbols indicate the sub-ontology that the class descriptions constructed by these constructors and axioms belong to.

The following example shows how to model a physical device ontology with OWL FA notation. The main reason for the functional syntax is that it is obvious to see which layer or stratum they belong to. Below we shown how to express multilevel model with an OWL FA notation.

According to PDDSL modelling architecture from Section 2, we can represent the class, property and instance relations in the M_2 layer. Example 3 shows how to describe class constructs by using OWL FA. Example 4 and 5 show how to describe constraint, class and property assertions from the layer M_2 to M_1 respectively.

Example 3. Class construct in M_2 layer:

- SlotContainer \sqsubseteq_2 Element (1)
- Self \sqsubseteq_2 SlotContainer (2)
- Chassis \sqsubseteq_2 SlotContainer (3)
- Configuration \sqsubseteq_2 Element (4)
- Slot \sqsubseteq_2 Element (5)
- Card \sqsubseteq_2 Element (6)

Example 4. Constraint in M_2 layer:

- SlotContainer $\sqsubseteq_2 \exists_2$ configurations.Configuration (7)
- Configuration $\sqsubseteq_2 \exists_2$ slots.Slot (8)
- Slot $\sqsubseteq_2 \exists_2$ cards.Card (9)

Example 5. Classes and properties assertion from the layer M_2 to M_1 :

- (Cisco, CiscoConfiguration) $:_2$ configuration (10)
- (CiscoConfiguration, CiscoSlot) $:_2$ slots (11)
- (CiscoSlot, CiscoCard) $:_2$ cards (12)
- CiscoConfiguration $:_2$ CiscoConfiguration (13)
- CiscoCard $:_2$ Card (14)
- CiscoSlot $:_2$ Slot (15)
- Cisco $:_2$ Chassis (16)

This layered representation, i.e. representing the OWL FA knowledge base in sub-ontologies allows to represent entities that are classes and instances at the same time. Due to limitations of space, we could not show the complete OWL FA ontology in this paper. However, the class, property and instance relations in M_1 layer can be described in the same manner.

6 Experimental Result

In this section, we compare our approach with OWL 2; OWL 2 is the only OWL language that supports metamodelling and has tool support. For OWL FA, we use the OWL FA toolkit that is introduced in [11].

6.1 Use case 1: Validating Multilevel Model

OWL 2 provides simple metamodelling with semantics which correspond to the contextual semantics defined in [15], however, it has been shown in [19] that these can lead to non-intuitive results. For example, the following axioms state

that AvayaP450 is a Avaya, and Avaya is a Shelf:

$$\text{AvayaP450} : \text{Avaya} \quad (17)$$

$$\text{Avaya} : \text{Shelf} \quad (18)$$

The axioms 17 and 18 could be interpreted by DL reasoner as follows:

$$\text{Ind} - \text{AvayaP450} : \text{Cls} - \text{Avaya} \quad (19)$$

$$\text{Ind} - \text{Avaya} : \text{Cls} - \text{Shelf} \quad (20)$$

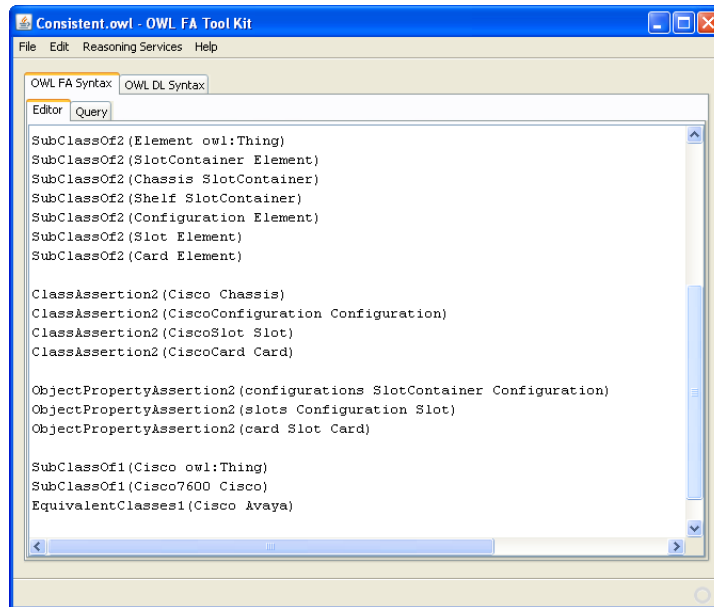


Fig. 2. Validating Multilevel Model with OWL FA Toolkit

Because the names of concepts and individuals do not interact even if they are the same, this type of metamodeling is often referred as punning. Let us consider the following axioms:

$$\text{Avaya} \equiv \text{Cisco} \quad (21)$$

$$\text{Shelf} \sqsubseteq \neg\text{Chassis} \quad (22)$$

The axioms 21 and 22 could be safely added to the ontology in contextual semantics, but under layered semantics this ontology is inconsistent because 21 indicates two concepts *Avaya* and *Cisco* are equivalent but 22 indicates that concept *Shelf* and *Chassis* are disjoint. This leads to a contradiction and therefore to an inconsistent ontology.

Figure 2 shows how to use OWL FA toolkit to validate multilevel model. More details about the OWL FA toolkit are described in [11].

6.2 Use case 2: Instance Retrieval for Multilevel Model

In OWL 2, if we do not provide the exact interaction between layers, it is difficult for any existing DL reasoner to discover this information. Let us consider the following axiom that defines the equivalence of *Avaya* and *Cisco* and the following query:

$$\text{Avaya} \equiv \text{Cisco} \tag{23}$$

```
SELECT *
WHERE {
    ?x a ont:Chassis.
}
```

From the query above, a user would like to retrieve all instances of *Chassis*. The result from the query will contain only *Cisco*. Without adding an axiom to indicate that *Avaya* is a *Chassis*, *Avaya* is missing in the answer set when we would like to retrieve all objects that belong to *Chassis*. However, OWL FA can return the more complete answer set because OWL FA can discover the missing relation in the reasoning process. For more details, we refer to [11].

Figure 3 shows the results from the OWL FA toolkit of the given SPARQL Query. OWL FA toolkit is able to return complete answer set.

7 Related Work

OWL FA was introduced in [19] for metamodelling in OWL. Motik [15] addressed metamodelling in OWL with two different semantics. The contextual semantics (or π -semantics) uses punning, i.e. names are replaced by distinct names for concepts, individuals and roles. This is like the different representation of an object in the OWL DL ontologies \mathcal{O}_i in OWL FA. OWL2 [16] provides simple metamodelling features which is based on the contextual approach. The other semantics is the HiLog semantics (or ν -semantics). The HiLog semantics is stronger than the π -semantics. The concepts and individual interpretations are not independent.

In [23] 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 .

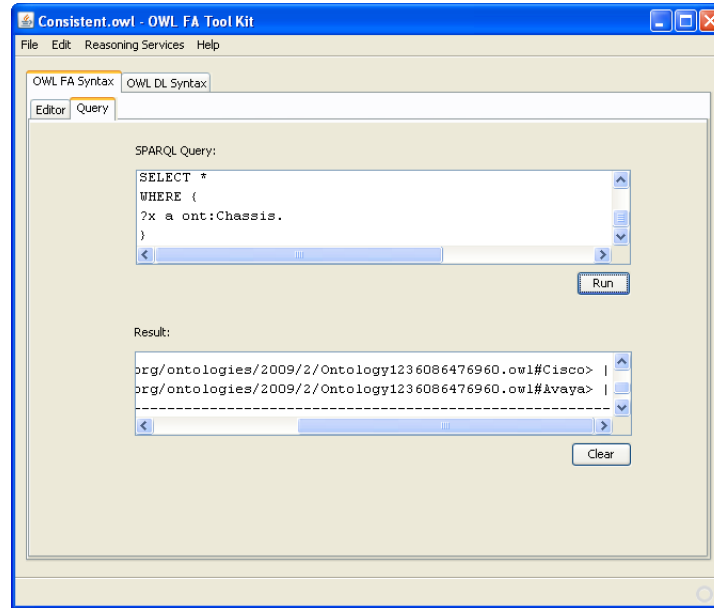


Fig. 3. Instance retrieval with OWL FA Toolkit

De Giacomina et al. [9] proposed the HiDL – Lite language, which adds one layer on top of the DL-Lite \mathcal{R} language. This supports meta-classes and meta-properties and presents the query answering algorithm by reduce HiDL – Lite to DL-Lite \mathcal{R} with the intention of using an existing DL-Lite reasoner. In OWL FA the semantics of meta-level are same as domain knowledge, unlike in HiDL-Lite semantics of the meta-level need to be re-defined.

Constraint programming is proposed for an automated verification of UML/OCL models in [8]. A UML diagram with OCL constraints is transformed to a constraint satisfaction problem. In [20] UML/OCL models are modelled with the constraint language Z in order to validate class diagrams. The formal language Alloy is used in [2] as a representation of UML/OCL models. The Alloy Analyser verifies the model properties. The modelling constraints do not cross multiple layers. Constraint logic programming (CLP) is applied in [14] 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. [6] 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 [12] for consistency checks of UML class diagrams. The main contributions 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 [5] in order to enable consistency check.

8 Discussion

In this section, we discuss the future direction of OWL FA. Although OWL FA has a well defined metamodelling architecture, OWL does not support cross layer constraints. Let's take the well known Endangered species as an example. If one would like to define a constraint on a meta-concept `EndangerSpecies` that all instances of these meta concept have only 3000 individuals. Therefore, if a concept `Eagle` is an instance of the meta-concept `EndangerSpecies`, the constraint should be applied to the concept `Eagle` as well. We have an idea how to express this kind of constraint by using a `TopObjectProperty` in OWL 2, we can then express `EndangerSpecies` requirement as $\top \sqsubseteq \leq 3000 \sqcup .\text{EndangerSpecies}$ then propagate this constraint to all instances of `EndangerSpecies`. This is beyond OWL FA so we would like to investigate on enriching OWL FA toward OWL 2 FA. Another issue is that in the cross layer constraints or restrictions for metamodelling in ontologies that we described in sections 2-5, we show that OWL FA is able to capture multiple layers better than OWL 2. However, the constraints or restrictions in OWL FA are relations between two layers. Let's consider the `ObjectProperty` assertion in layer M_2 , `SlotContainer` have `Configuration`, which is expressed by the `ObjectProperty` configurations. This constraint can only be used to validate the model between the layers M_2 and M_1 . We would like to investigate that it is possible to propagate the constraints across multiple layers. We are thinking about to use a meta prefix (meta-) like `meta_hasConfig(Cisco, CiscoConfiguration)` that would be an object property assertion in M_2 layer and `hasConfig(cisco7603, Cisco7603Config.)` in M_1 layer. Then we could propagate the property assertion in layer M_2 to layer M_1 semantically. For instance, `meta_hasConfig(Cisco, CiscoConfiguration)` in M_2 to become `Cisco \sqsubseteq hasConfig.CiscoConfiguration` in M_1 by discarding meta prefix. This would be very interesting because we could specify all the constraints only in higher layers, then propagate them down to the lower layers automatically. The cross layer constraints are requirements from software engineering.

9 Conclusion

In this paper, we have presented an approach for verifying and validating multi-level models by using OWL FA. Firstly, we described the requirements for modelling and validating of multilevel models through a practical example. Then,

we detailed the need and requirements from software engineering. In the subsequent sections, we demonstrated the syntax and semantics of OWL FA, followed by the detailed description of how to model multilevel models with metamodelling enabled ontologies, in particular with OWL FA. The early experimental results show that OWL FA and its reasoner could benefit software modellers on leveraging the software development life cycle. We also discuss about the future direction of OWL FA.

We have shown how to use the OWL FA Toolkit to verify and validate reasoning across layers in multi-layer model and we plan to incorporate these into the TrOWL⁴ reasoning infrastructure. We will support the creation of OWL FA ontologies with a plug-in for the Protégé⁵ ontology development environment, which is currently in development.

In the future, we would like to enrich OWL FA language toward the direction we described in discussion section in order to increase expressive power of the language such as propagate constraints between layers. Moreover, we would like to apply the fixed-layer architecture to OWL 2 DL which has more expressive power than OWL DL. And we plan to provide tools along with a reasoning mechanism for OWL 2 FA.

Acknowledgements

This work has been partially supported by the European Project Marrying Ontologies and Software Technologies (MOST ICT 2008-216691).

References

1. OMG Unified Modeling Language (OMG UML) Infrastructure. <http://www.omg.org/spec/UML/2.2/Infrastructure>, 2009. Version 2.2.
2. K. Anastakis, B. Bordbar, G. Georg, and I. Ray. UML2Alloy: A challenging model transformation. *Lecture Notes in Computer Science*, 4735:436, 2007.
3. Colin Atkinson, Matthias Gutheil, and Bastian Kennel. A Flexible Infrastructure for Multilevel Language Engineering. *IEEE Trans. Software Eng.*, 35(6):742–755, 2009.
4. Colin Atkinson and Thomas Kühne. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):36–41, 2003.
5. B. Beckert, U. Keller, P.H. Schmitt, 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.
6. D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.
7. Alan Brown. An introduction to Model Driven Architecture. IBM Technical Report. URL <http://www-128.ibm.com/developerworks/rational/library/3100.html>, 2004.

⁴ <http://www.trowl.eu>

⁵ <http://protege.stanford.edu>

8. J. Cabot, R. Clariso, and D. Riera. Verification of UML/OCL Class Diagrams using Constraint Programming. In *Software Testing Verification and Validation Workshop*, pages 73–80, 2008.
9. Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Towards higher-order DL-Lite (preliminary report). In *Proceedings of the International Workshop on Description Logic (DL-2008), Dresden, Germany, May 13-16, 2008*.
10. Jean-Marie Favre. Foundations of Meta-Pyramids: Languages vs. Metamodels - Episode II: Story of Thotus the Baboon. In *Dagstuhl Seminar 0401 on Language Engineering for Model-Driven Software Development*, 2004.
11. Nophadol Jekjantuk, Gerd Gröner, and Jeff Z. Pan. Reasoning in Metamodeling Enabled Ontologies. In *Proceeding of the International workshop on OWL: Experience and Directions (OWL-ED2009)*, 2009.
12. K. Kaneiwa and K. Satoh. Consistency checking algorithms for restricted UML class diagrams. *Lecture Notes in Computer Science*, 3861:219, 2006.
13. T. Kühne. Matters of (Meta-) Modeling. *Software and Systems Modeling*, 5(4):369–385, 2006.
14. H. Malgouyres and G. Motet. A UML Model Consistency Verification Approach based on Meta-Modeling Formalization. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1804–1809, New York, NY, USA, 2006. ACM.
15. Boris Motik. On the Properties of Metamodeling in OWL. *J. Log. Comput.*, 17(4):617–637, 2007.
16. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. Owl 2 web ontology language: Structural specification and functional-style syntax. World Wide Web Consortium, Working Draft WD-owl2-semantic-20081202, December 2008.
17. I. Ober and A. Prinz. What do we need metamodels for? In *4th Nordic Workshop on UML and Software Modelling*, pages 8–28. Citeseer, 2006.
18. Jeff Z. Pan and Ian Horrocks. RDFS(FA) and RDF MT: Two Semantics for RDFS. In *Proc. of the 2nd International Semantic Web Conference (ISWC2003)*, 2003.
19. Jeff Z. Pan, Ian Horrocks, and Guus Schreiber. OWL FA: A Metamodeling Extension of OWL DL. In *Proceeding of the International workshop on OWL: Experience and Directions (OWL-ED2005)*, 2005.
20. D. Roe, K. Broda, A. Russo, and Department of Computing. Mapping UML models incorporating OCL constraints into Object-Z. In *Imperial College of Science, Technology and Medicine, Department of Computing*, 2003.
21. R. Van Der Straeten. *Inconsistency Management in Model-Driven Engineering*. PhD thesis, Vrije Universiteit Brussel, 2005.
22. UML. Unified Modeling Language. <http://www.uml.org/>.
23. Christopher A. Welty and David A. Ferrucci. What’s in an instance? Technical report, RPI Computer Science, 1994.