

Towards Hybrid Reasoning for Verifying and Validating Multilevel Models

Nophadol Jekjantuk¹, Gerd Gröner², Jeff Z. Pan¹, and Edward Thomas¹

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 realizes a four-layer metamodelling architecture, the new version of OWL Web Ontology Language, called OWL 2, it supports only simple metamodelling. Moreover, the semantics of metamodelling in OWL 2 corresponds to the contextual semantics which leads to non-intuitive results. Another issue is that the Open World Assumption (OWA) assumes a model is incomplete. Therefore, we could not validate some constrains in OWA. In this paper, we demonstrate multilevel (meta-) modelling using ontologies described in OWL FA, which has a well defined fixed-layered architecture and semantics. As well as an approach to integrate Closed World Assumption(CWA) with OWA in order to use both assumptions for verifying and validating multilevel model.

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 sub languages 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 [21], Model Driven Architecture [5] 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. However, OWL DL 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 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 [4, 6, 12, 20, 13] (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).

Another issue is that description logics rely on open world assumption (OWA) which assumes incomplete information as default and allows for validating incomplete models. Therefore, we could not validate some constraints in OWA (cf. Section 4.3). The closed-world assumption (CWA) assumes that the elements in the model are known and unchanging. However, we would like to use the benefits from both assumptions to validate multilevel model.

There is work on closed world reasoning or local closed world reasoning (cf. [9]) for OWL knowledge bases like using epistemic operators as described in [10, 8]. Hybrid knowledge representations are investigated in [14] to integrate both assumptions. However, none of those approach consider metamodelling.

In this paper, at first, we use OWL FA [19] to describe a multilevel model. Then, we use OWL FA reasoner [11] together with Local Closed World Assumptions (LCWA) engines to validate models covering multiple modelling levels. Thus, this will leverage the software development life cycle.

The paper is organised as follows. Section 2 outlines a metamodelling application which demonstrates dependencies between multiple levels. The need and requirement is described in Section 3. In the subsequent sections, we demonstrate metamodelling based on description logics including the syntax and semantics of OWL FA, followed by a detailed description of how to represent multilevel models with metamodelling in ontologies as well as the need of closed world assumption are given in Section 4. In Section 5, we detailed the local closed world assumption, followed by an evaluation from experimental results in Section 6. Then, we discuss the future direction of OWL FA, while a comparison with related works is contained in Section 7.

2 A Motivating Example

This section gives an example to demonstrate the need for metamodeling enabled ontologies. Models are depicted in UML 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 descriptions, i.e. instances in layer M_i are treated as classes in the layer M_{i-1} below. However, representing such a layered modelling architecture directly in an OWL ontology (e.g. in an OWL Full ontology) leads to undecidability of reasoning in order to validate models.

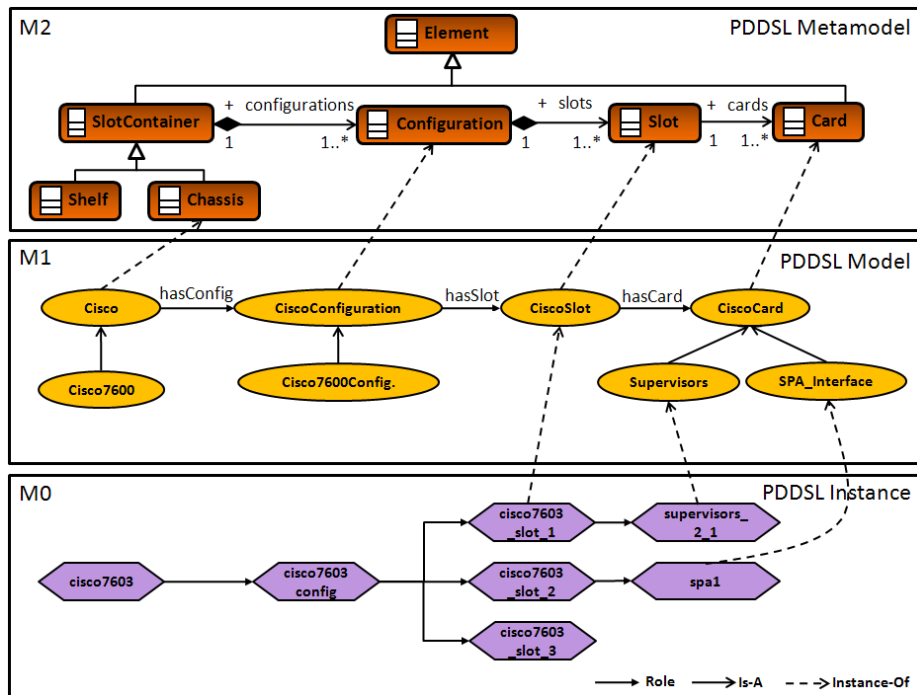


Fig. 1. Layered Architecture for a Physical Device Model

In Figure 1, a layered modelling architecture is demonstrated for physical device modelling (an application of configuration management). The models and

metamodels are described in a domain specific language (DSL) that is called physical devices domain specific language (PDDSL). It is germane in metamodelling to distinguish between different roles (or actors) on different model or metamodel layers. A language or model designer (or DSL designer) describes the metamodel that is represented by a model on layer M_2 and its instances on M_1 . The second role (or actor) is the language user (or DSL user) that builds models on layer M_1 for a certain configuration which is instantiated on layer M_0 . These models on M_1 are focused on concrete and detailed configurations while the models built by the designer on layer M_2 are more general and abstract. Obviously, the model is build with respect to the given metamodel on layer M_2 . However, these two actors are working independently.

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 in the M_2 layer. Obviously, in a layered modelling architecture, these instances in the layer M_1 are treated as concepts with respect to the layer M_0 below. The instance relations between adjacent layers are represented by the dotted arrows. The rest of the relationships like subclass relations and object properties are displayed by the continuous arrows, e.g. `Shelf` is a subclass of `SlotContainer`.

There are additional modelling constraints imposed on different layers. In layer M_2 a model designer requires that each `SlotContainer` has a `Configuration`, which is expressed by the functional ObjectProperty `configurations`. And each `Configuration` has to contain at least one `Slot`, which is expressed by the slots as a functional ObjectProperty `slots`. And each `Slot` contains at least one `Card`, this is also expressed as a functional ObjectProperty `cards`. The main reason to expressed `configurations`, `Slot`, and `Card` as a functional ObjectProperty is that a relationship between two concept can define only specific direction. Moreover, the modeller requires the disjointness of the `Chassis` and `Shelf`. The associations in layer M_2 are described by three ObjectProperty assertions form 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 the ObjectProperty.

Additionally, in a concrete model in M_1 , the modeller requires the disjointness of the two classes `Supervisor` and `SPA_interface`. Furthermore, `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 for the whole model, covering multiple layers simultaneously. If the language designer would like to add `CiscoCRS1 – 4Slot` is a `Shelf`. Hence, `CiscoCRS1 – 4Slot` is an instance in layer M_1 of `Shelf` in layer M_2 . At the same time, `CiscoCRS1 – 4Slot` is also a concept in layer M_1 with respect to the next layer M_0 below, i.e. there are instances in layer M_0 of the concept `CiscoCRS1 – 4Slot`. For the concrete configuration, the language user (or system modeller) requires for the concept `CiscoCRS1 – 4Slot` the equivalence to the concept `Cisco` in M_1 . Combined with the constraints on the metamodel layer M_2 which require the disjointness of `Shelf` and `Chassis`, the model is inconsistent, since it is required that two instances that are considered as identical are instances of disjoint classes.

This inconsistency is not detected in a pure OWL DL model since `CiscoCRS1 – 4Slot` in layer M_1 is either an instance of `Shelf` or a concept with instances on the layer M_0 . However, `CiscoCRS1 – 4Slot` cannot be both at the same time. OWL FA separate classes and objects into different layers, i.e. model and instance layer as in OWL DL, to maintain the decidability of the language. 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 metamodelling and the semantics of OWL FA are well defined.

Although, OWL FA allows us to capture a multilevel model and validate it with its reasoning mechanism, in some cases we cannot validate the multilevel model suitably because OWL FA realizes the open world assumption (OWA). We will discuss this through an example in Section 4.3.

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 (language 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.

In order to support also the language designer on the next layer above with ontologies and reasoning services, the metamodelling layer M_2 would be represented in the TBox and the instances of layer M_1 are modelled in the ABox. However, it is not possible to combine modelling and reasoning support covering more than two layers in OWL (-DL), 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 meta-model 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 realization of these service requirements depends on the modelling possibilities and on the reasoning services that are available for the model. To realize a single requirement, a two-layered model is appropriate. However, if more of these requirements have to be realized 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 are required for modelling and reasoning.

4 Metamodelling Based on Description Logics

In the following we present OWL FA and semantics together with how to use OWL FA to represent a multilevel model. Then we discuss why the Open World Assumption is not enough for validating the multilevel model.

4.1 OWL FA syntax and semantics

OWL FA [19] 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) [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 V_{C_i} (for stratum i), datatype names V_D , abstract property names V_{AP_i} (for stratum i), datatype property names 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).

Let $CN \in \mathbf{V}_{C_i}$ be an atomic class name in layer i ($i \geq 0$), R an OWL FA-property in layer i , $o \in \mathbf{I}$ an individual, $T \in \mathbf{V}_{DP}$ a datatype property name, and C, D OWL FA-classes in layer i . Valid OWL FA-classes are defined by the abstract syntax:

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

The semantics of OWL FA is a model theoretic semantics, which is defined in terms of interpretations. In other words, 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. 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 tuple of the form 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. 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 $RN \in \mathbf{V}_{AP_i}$ be an abstract property name in layer i and R be an abstract property in layer 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. The interpretation function is explained in detail in [19].

4.2 Representing Multilevel Model with OWL FA

In this section, we present the way of express 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. The 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 2 shows how to describe class constructs by using OWL FA notation. Example 3 and 4 show how to describe constraint, class and property assertions from the layer M_2 to M_1 respectively.

Example 2. Class construct in M_2 layer:

- SlotContainer \sqsubseteq_2 Element (1)
- Shelf \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 3. 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 4. 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)

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.

4.3 Closed and Open World Assumption in OWL FA

Knowledge representation in OWL and also in OWL FA in general realizes the open world assumption (OWA). The open world assumption assumes incomplete information. More precisely, we can only use known statements to infer information. In the semantic web, this assumption is beneficial in order to build a knowledge base that can be further extended and enriched. In contrast, in the closed world assumption each statement that is unknown is assumed to be false, i.e. we assume our knowledge base is complete.

In some applications, the closed world assumption is assumed like in database systems and also in model-driven engineering. Hence, for certain validation tasks in multi-level models closed world assumption or closed world reasoning is required.

The following example demonstrates the difference of closed and open world assumption. We consider the following assertions as an excerpt of the ontology

from layer M_1 to M_0 . There are assertions that describe for some `Cisco7600Config` instances the cards (`CiscoCard` instances) that are plugged-in in the slots.

$$(\text{cisco7603_slot_1}, \text{supervisors_2_1}) :_1 \text{ hasCard} \quad (17)$$

$$(\text{cisco7603_slot_1}, \text{supervisors_2_2}) :_1 \text{ hasCard} \quad (18)$$

$$(\text{sicso7603_slot_2}, \text{supervisors_3_1}) :_1 \text{ hasCard} \quad (19)$$

$$(\text{sicso7603_slot_3}, \text{supervisors_3_2}) :_1 \text{ hasCard} \quad (20)$$

$$\text{supervisors_2_1} :_1 \text{ Supervisors} \quad (21)$$

$$\text{supervisors_2_2} :_1 \text{ Supervisors} \quad (22)$$

$$\text{supervisors_3_1} :_1 \text{ Supervisors} \quad (23)$$

$$\text{supervisors_3_2} :_1 \text{ Supervisors} \quad (24)$$

A device modeller is interested whether a certain card of type `SPA_Interface` is used in any of the current configurations of type `Cisco7600Config`. `SPA_Interface` is a subclass of `CiscoCard` in M_1 (cf. Fig. 1). Assumed there is no assertion in the ontology, that explicitly states that there is a configuration with a slot that is connected to a card of type `SPA_Interface`.

Using standard open world assumption we cannot answer this question since the absence of an assertion that assigns such a card of type `SPA_Interface` to a configuration does not imply that there is no such configuration. In contrast, using the closed world assumption, we can conclude from the ontology that there is no such configuration that uses a card of type `SPA_Interface`.

5 Local Closed World Assumption

The Local Closed World Assumption (LCWA) allows selective parts of an otherwise open world ontology to be locally closed. This means that the reasoner cannot assume the existence of any additional individuals or property relationships between classes or properties which are within the domain of the closed elements. For example, under the open world assumption, given the axiom $C \sqsubseteq \exists R.D$; the reasoner would assume the presence of a relationship R between every explicit instance of the class C and some instance of the class D , whether or not it was explicitly stated. If we locally close the class D , then the reasoner could only assume such a relationship with an explicit member of D . If the reasoner could not make this assumption, for example, if R was an inverse functional property, and all instances of D were already accounted for in the property R , this would render the ontology inconsistent.

To close a class in an ontology, we simply enumerate all known members of that class, and add a new axiom to the ontology stating that the class is equivalent to that set of individuals. This requires a language that can express nominals, such as OWL DL, or OWL2.

To close a property, we must add a new axiom for each instance of the property which can be inferred from the ontology. For each instance of a property $R(i_1, i_2)$, we add new axioms such that $\exists R.\{i_2\} \equiv \{i_1\}$ and $\exists R^-\{i_1\} \equiv$

$\{i_2\}$. Furthermore, we take the set of all inferable instances of the property $(i_{d1}, i_{r1}), \dots, (i_{d1}, i_{r1})$ and state that $\exists R \equiv \{i_{d1}, \dots, i_{dn}\}$ and $\exists R^- \equiv \{i_{r1}, \dots, i_{rn}\}$.

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:

$$\text{AvayaP450} : \text{Avaya} \quad (25)$$

$$\text{Avaya} : \text{Shelf} \quad (26)$$

The axioms 25 and 26 could be interpreted by DL reasoner as follows:

$$\text{Ind} - \text{AvayaP450} : \text{Cls} - \text{Avaya} \quad (27)$$

$$\text{Ind} - \text{Avaya} : \text{Cls} - \text{Shelf} \quad (28)$$

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

$$\text{Avaya} \equiv \text{Cisco} \quad (29)$$

$$\text{Shelf} \sqsubseteq \neg \text{Chassis} \quad (30)$$

The axioms 29 and 30 could be safely added to the ontology in contextual semantics, but under layered semantics this ontology is inconsistent because 29 indicates two concepts *Avaya* and *Cisco* are equivalent but 30 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].

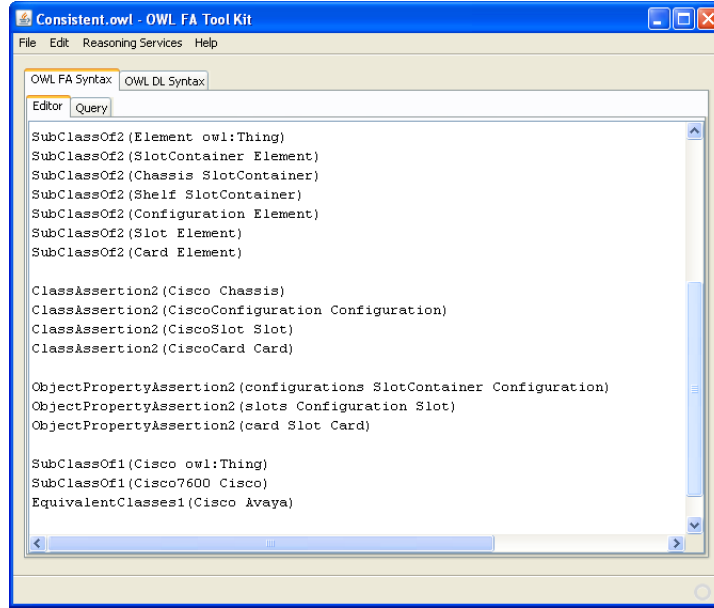


Fig. 2. Validating Multilevel Model with OWL FA Toolkit

6.2 Use case 2: Validating Multilevel Model with Local Closed World Assumption

Without using Local Closed World Assumption, we might not be able to validate a multilevel model. Let us consider the following OWL FA ontology.

$$\text{Shelf} \sqsubseteq_2 \neg\text{Chassis} \quad (31)$$

$$\text{Cisco7600} :_2 \text{Chassis} \quad (32)$$

$$\text{CiscoCRS1} - 4\text{Slot} :_2 \text{Shelf} \quad (33)$$

$$\text{Cisco7603} :_1 \text{Cisco7600} \quad (34)$$

$$\text{CiscoCRS1} - 4\text{Slot} - 1 :_1 \text{CiscoCRS1} - 4\text{Slot} \quad (35)$$

$$\text{CiscoCRS1} - 4\text{Slot} - 1 \approx_0 \text{Cisco7603} \quad (36)$$

A Cisco7603 is stop working and it need to be replace. One would like to know that it is possible to use CiscoCRS1 - 4Slot - 1 instead. Then, the modeller immediately make CiscoCRS1 - 4Slot - 1 become equivalent to Cisco7603 and validate it with OWL FA Toolkit. The expected answer should be invalid but the OWL FA Toolkit returns valid according to Open World Assumption(OWA). In OWA, even Cisco7600 has one individual in the knowledge base it does not means that Cisco7603 is only one individual of Cisco7600. Cisco7600 can have infinite individual which is not define yet. The axioms CiscoCRS1 - 4Slot - 1 \approx_0

Cisco7603 does not make Cisco7600 become equivalent to CiscoCRS1 – 4Slot. Therefore, this multilevel model is still valid.

Now, we use new version of OWL FA tool kit with included local closed world assumption engine. Then, We close concept Cisco7600 and CiscoCRS1 – 4Slot. The result of validation will be inconsistent.

7 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 apply 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 relation 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.

8 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 [22] 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 .

De Giacomina et al. [7] 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 [6]. 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 [13] 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. [4] 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 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 [3] in order to enable consistency check.

9 Conclusion

In this paper, we have presented an approach to verify and validate multilevel model that describe in OWL FA by using hybrid reasoning approach. At first, we described the requirements for verifying and validating of multilevel model through a practical example. Then, we detailed the need and requirements from software engineering. In the subsequent sections we demonstrated metamodelling based on description logics including the syntax and semantics of OWL FA, followed by the detailed description of how to represent multilevel model with metamodelling in ontologies and the need of closed world assumption. Later we detailed about local closed world assumption. The early experimental results

show that OWL FA and its reasoner could benefit a software modeller in order to leverage 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 over multilevel model and we plan to incorporate these into the TrOWL³ reasoning infrastructure. We will support the creation of OWL-FA ontologies with a plug-in for the Protégé⁴ 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 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. Anastasakis, B. Bordbar, G. Georg, and I. Ray. UML2Alloy: A challenging model transformation. *Lecture Notes in Computer Science*, 4735:436, 2007.
3. 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.
4. D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.
5. Alan Brown. An introduction to Model Driven Architecture. IBM Technical Report. URL <http://www-128.ibm.com/developerworks/rational/library/3100.html>, 2004.
6. 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.
7. 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*.
8. F.M. Donini, D. Nardi, and R. Rosati. Description logics of minimal knowledge and negation as failure. *ACM Transactions on Computational Logic (TOCL)*, 3(2):225, 2002.

³ <http://www.trowl.eu>

⁴ <http://protege.stanford.edu>

9. O. Etzioni, K. Golden, and D. Weld. Tractable Closed World Reasoning. *Proc. of Knowledge Representation (KR)*, 2004.
10. S. Grimm and B. Motik. Closed World Reasoning in the Semantic Web through Epistemic Operators. In *CEUR Proceedings of the OWL Experiences and Directions Workshop, Galway, Ireland*. Citeseer, 2005.
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. 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.
14. B. Motik, I. Horrocks, R. Rosati, and U. Sattler. Can OWL and Logic Programming live together happily ever after? *The Semantic Web-ISWC 2006*, pages 501–514, 2006.
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-semantics-20081202, December 2008.
17. I. Ober and A. Prinz. What do we need metamodels for?
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. UML. Unified Modeling Language. <http://www.uml.org/>.
22. Christopher A. Welty and David A. Ferrucci. What's in an instance? Technical report, RPI Computer Science, 1994.