2011 35th IEEE Annual Computer Software and Applications Conference

Diagnosis of Software Models with Multiple Levels of Abstraction Using Ontological Metamodeling

Nophadol Jekjantuk*, Jeff Z. Pan*, and Yuzhong Qu†
*Department of Computing Science, University of Aberdeen, United Kingdom
†State Key Lab. for Novel Software Technology, Nanjing University

***Abstract*—Ontology reasoning and reasoning-based search expansion are powerful tools for semantic applications. In this paper, we investigate how to apply ontology reasoning, with different ontology metamodeling approaches, to diagnose software models, in particular on models with multiple levels of abstraction.**

***Keywords*-Metamodeling; Reasoning; Ontological Metamodeling; Diagnosis; Software Engineering**

I. Introduction

Ontology reasoning and reasoning based semantic search are useful tools for not only semantic applications, but also for software engineering. An ontology consists of a set of important vocabulary (including concepts, properties and individuals) for certain domain, as well as their definitions and background constraints. Ontology reasoning is useful to complete implicit connections among the vocabulary, which are useful for knowledge based systems as well as semantic search, e.g. for search expansion.

In this paper, we investigate how to apply ontology reasoning, with different ontology metamodeling approaches, to diagnose software models with multiple levels of abstraction. Metamodeling has received increased attention over the years. There are two kinds of metamodeling: linguistic metamodeling and ontological metamodeling. While linguistic metamodeling is used to define or extend the modelling language [1], ontological metamodeling is used to model a complex domain model where the distinction between classes and individuals is not clear-cut. We use ontological metamodeling in our investigation.

A. Why do we need ontological metamodeling?

In what follows, we first give a simple motivating example before discussing some real world examples.

Let's take a knowledge base about animals as an example: "Ted is an Eagle" and "Eagles are an EndangeredSpecies". Therefore, this ontology should be modelled by stating the individual Eagle to be an instance of the class EndangeredSpecies. Hence, the symbol Eagle is used to refer to both a class as well as to an individual. This style of modelling is often called metamodeling.

In other situations, the distinction between properties and individuals is not clear-cut either. For example, we may want to represent a property as an individual of a class, which can found in the Collaborative Environment (Wiki) "The property is_located_in is a member of the class Deprecated_Properties".

Indeed, ontological metamodeling is useful in not only specific application areas, such as medical science (e.g., SNOMED-CT and FMA ontology) [17] , but also software engineering in general, such as Model Driven Architecture [4].

B. Metamodeling Support in the OWL Standard

However, the lack of support in the standard Web Ontology Language OWL discourages users from making use of metamodeling, and forces users to use subsumption relations instead of instanceOf relations, which can lead to unintended inferences. In the above example about Eagle. either of the two statements can be represented in OWL DL by asserting the individual Ted to be an instance of the class Eagle. The two statements cannot be modelled in an OWL DL ontology, because OWL DL does not allow a single symbol to refer both to a class and an individual. One may model the class Eagle as a sub class of the class EndangeredSpecies. However, an eagle is a species and not a set of all living eagles. Thus this could lead to reasoners making unintended or incorrect inferences (e.g., Ted is an endangered species).

The current OWL standard supports metamodeling in two flavours. Metamodeling in OWL Full allows treating classes as individuals. However, reasoning in OWL Full is not decidable. OWL 2 [14] supports a very basic, but decidable approach to metamodeling called punning or contextual semantics [13]: one symbol can be used to refer both to a class as well as to an individual; the decision whether a symbol is interpreted as class, property, or individual is context-dependent.

In this paper, we investigate and compare three exisiting ontological metamodeling approaches, namely Contextual semantics [13], OWL FA [15] and classed-based metamodeling [7]. The paper is organised as follows. Section II outlines a metamodeling application and requirements in network device configuration management which demonstrates dependencies between multiple modelling layers. The detail of syntax and semantics of OWL 2, Class-based Metamodeling, and OWL FA are given in Section III. Section IV introduces three small test cases in order to make comparison between the three approaches. Section V provides an evaluation and discussion of the proposed test cases, while a comparison with related work is presented in Section VI. Finally, we conclude in Section VII.

II. Motivating Example and Requirements

This section outlines a metamodeling application and requirements in network device configuration management, which demonstrates dependencies between multiple modelling layers.

Example 1 *A physical device domain specific language (PDDSL) is a domain specific language (DSL) for physical devices, which is used in business IT system modelling.*

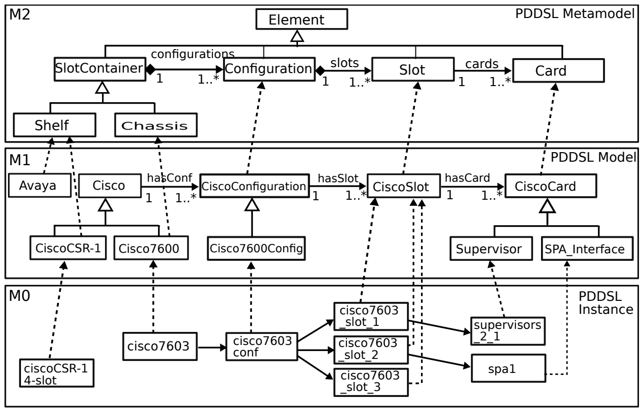

Figure 1. Layered Architecture for a Physical Device Model

In Figure 1, a layered modelling architecture is demonstrated for physical device modelling (an application of configuration management). The figure depicts three layers M_0, M_1 *and* M_2. M_3 *is a Meta-metamodeling layer which is not included in the example. The arrows between the layers demonstrate instance relationships, the arrows within a layer are concept relations (i.e., object properties and subclass hierarchies).*

A crucial task in model-driven engineering is the validation of models and metamodels. A valid model refers to its metamodel and satisfies all restrictions and constraints. However, the validation of multiple layers may lead to inconsistency, even if the consistency is satisfied between all adjacent layers.

For the requirements, we first define basic modelling requirements for using OWL in a layered modelling architecture, in combination with reasoning to validate models and metamodels. Secondly, we describe the functional user requirements form a PDDSL modeller and meta-modeller point of view.

A. Modelling Requirements

In order to improve the software development processes, new technologies which provide reasoning support, like consistency checking of models and metamodels, are beneficial. In order to use the modelling capabilities of OWL and the potential of DL reasoning in a layered architecture, the OWL-based modelling language has to satisfy two requirements. Firstly, the language has to support dealing with concepts and meta-concepts. Secondly, a tractable reasoning complexity and tool support is necessary for the OWL-based modelling language.

B. Functional Requirement

The PDDSL user models physical devices and their configurations in layer M_1 and the data and objects of these models are represented in layer M_0. In usual OWL (-DL) conceptual two-layer models, this modelling layer would be the TBox. In this case, the user benefits from modelling and reasoning support for the models and the instances on the layer below. However, it is not possible to combine modelling and reasoning support covering more than two layers.

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

- Planning of Networks: The modelling frameworks enable restrictions and suggestions of components that can be used in the current model configuration.
- 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(M_2).
- Data quality analysis: The user checks whether a configuration on M_0 is instance of a modelled configuration on M_1.
- Explanations and Justifications for the PDDSL modeller, in order to provide meaningful advice to the modeller of inconsistencies and contradictions in the model and to detect causes of such errors.

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 often appropriate. However, if more of these requirements have to be realised simultaneously, two layers are not enough.

III. Ontology and Metamodeling

A. OWL 2

An OWL 2 DL vocabulary$\mathcal{V}_{\mathcal{O}} = (\mathcal{V}_{cls}, \mathcal{V}_{op}, \mathcal{V}_{dp}, \mathcal{V}_{ind}, \mathcal{V}_{dt}, \mathcal{V}_{lt}, \mathcal{V}_{fa})$ is a 7-tuple over a datatype map D where $\mathcal{V}_{cls}$ is the set of IRIs denoting class names, $\mathcal{V}_{op}$ is the set of IRIs denoting object properties, $\mathcal{V}_{dp}$ is the set of IRIs denoting datatype properties,$\mathcal{V}_{ind}$is the set of IRIs denoting individuals,$\mathcal{V}_{dt}$ is the set of IRIs denoting all datatypes of D, the datatype rdfs:Literal, and possibly other datatypes, is the set of IRIs denoting datatype names, $\mathcal{V}_{lt}$ is the set of well-formed RDF literals and $\mathcal{V}_{fa}$ is the set of pairs (F, lt) for each containing facet F, and literal lt.

The abstract syntax for an OWL 2 class definition is:

$$C \leftarrow \top \mid \bot \mid CN \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \{o\} \mid \exists R.C \mid \\ \forall R.C \mid \exists R : Self \mid \leq nR.C \mid \geq nR.C \mid$$

where $C \in \mathcal{V}_{cls}$, $a \in \mathcal{V}_{ind}$, $R, T \in \mathcal{V}_{op}$ and it are simple roles, and n is a non-negative integer.

The semantics of OWL 2 ontologies are given by means of interpretations. An interpretation $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ called the domain, together with a function $\cdot^{\mathcal{I}}$ mapping individual names to elements of $\Delta^{\mathcal{I}}$, class names to subsets of $\Delta^{\mathcal{I}}$, and role names to subsets of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function is extended to complex class expressions in the usual way as explained in detail in [14].

An interpretation $\mathcal{I}$ satisfies an axiom φ if we find that $\mathcal{I} \models \varphi$:

- $\mathcal{I} \models S \sqsubseteq R$ if $S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$,
- $\mathcal{I} \models S_1 \circ \ldots \circ S_n \sqsubseteq R$ if $S_1^{\mathcal{I}} \circ \ldots \circ S_n^{\mathcal{I}} \sqsubseteq R^{\mathcal{I}}$,
- $\mathcal{I} \models Dis(R, S)$ if $R^{\mathcal{I}}$ and $S^{\mathcal{I}}$ are disjoint,
- $\mathcal{I} \models C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq C^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ satisfies $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $R(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies an ontology $\mathcal{O}$ if it satisfies all axioms of $\mathcal{O}$. An ontology $\mathcal{O}$ is satisfiable if it has a model. An ontology $\mathcal{O}$ entails an axiom φ, if every model of $\mathcal{O}$ is a model of φ.

B. Classed-based Approach

The classed-based approach [7], is ontology-inherent metamodeling for classes in OWL based on an axiomatisation of class reification. We can obtain the a metamodeling-enabled version $\mathcal{O}^{meta}$ from a given ontology $\mathcal{O}$ with function $\mathsf{bound}(\cdot)$,$\mathsf{SepDom}(\cdot)$,Typing $(\cdot)$ and $\mathsf{MatSubClass}(\cdot)$.

Let $\mathcal{O}$ be a domain ontology with vocabulary $\mathcal{V}_{\mathsf{O}_C, \mathsf{O}_R, \mathsf{O}_I}$ The vocabulary of the metamodeling-enabled version is

$\mathsf{O}_C^{meta} := \mathsf{O}_C \cup \{\mathsf{Inst}, \mathsf{Class}\}$
$\mathsf{O}_R^{meta} := \mathsf{O}_R \cup \{\mathsf{type}, \mathsf{subClassof}, R_{\mathsf{Inst}}\}$
$\mathsf{O}_I^{meta} := \mathsf{O}_I \cup \{o_{\mathsf{C}} \mid \mathsf{C} \in \mathsf{O}_C\}$

where all newly introduced names are fresh where they are not part of $\mathcal{V}_{\mathcal{O}}$.

The function $\mathsf{bound}(\cdot)$ returns its input after rewriting it as follows: first, every occurrence of X having one of the forms $\top$, $\neg\mathsf{C}$, $\forall R.\mathsf{C}$, $\leqslant nR.\mathsf{C}$, $\exists U.\mathsf{Self}$ is substituted by $\mathsf{Inst} \sqcap X$, where we can express complex classes C. Next, the universal role is localized by substituting every $\forall U.\mathsf{C}$ by $\forall U.(\mathsf{Inst} \sqcup \mathsf{C})$ and every U occurring on the left hand side of a role chain axiom by $R_{\mathsf{Inst}} \circ U \circ R_{\mathsf{Inst}}$ where R_{Inst} is axiomatized via $\exists R_{\mathsf{Inst}}.\mathsf{Self} \equiv \mathsf{Inst}$. The functions SepDom, Typing, and MatSubClass return a set of axioms as specified in [7]. The metamodeling-enabled version $\mathcal{O}^{meta}$ of $\mathcal{O}$ is $\mathsf{bound}(\mathcal{O}) \cup \mathsf{SepDom}(\mathcal{O}) \cup$ $\mathsf{Typing}(\mathcal{O}) \cup \mathsf{MatSubClass}(\mathcal{O})$.

C. Extended Class-based Approach

Although the class-based approach is complete w.r.t OntoClean methodology, where meta ontologies can make use of subsumption constraints from the domain ontology, it cannot be reused in the Software Model case. For a better understanding, let's consider a simple example. Let class C be equivalent to D in $\mathcal{O}$ and we generate the meta version from $\mathcal{O}$ then add the three meta statements. o_C is an instance of Meta-class E and o_D is an instance of Meta-class F. Then, we make E and F disjoint from each other. Please note that class C and individual o_C refer to the same symbol (punning style).

Given an ontology $\mathcal{O}$ that has only one axiom $\mathsf{C} \equiv \mathsf{D}$ then we could generate $\mathcal{O}^{meta}$ as follows:

bound$(\mathcal{O})$	$\mathsf{C} \equiv \mathsf{D}$
SepDom$(\mathcal{O})$	$\mathsf{c\_Class} \equiv \neg\mathsf{c\_Inst}$ $\mathsf{c\_Class(o\_C)}$ $\mathsf{c\_Class(o\_D)}$ $\top \sqsubseteq \forall\mathsf{r\_subClassOf.c\_Class}$ $\top \sqsubseteq \forall\mathsf{r\_type.c\_Class}$ $\exists\mathsf{r\_subClassOf}.\top \sqsubseteq \mathsf{c\_Class}$ $\exists\mathsf{r\_type}.\top \sqsubseteq \mathsf{c\_Inst}$
Typing$(\mathcal{O})$	$\mathsf{C} \equiv \exists\mathsf{r\_type}.\{\mathsf{o\_C}\}$ $\mathsf{D} \equiv \exists\mathsf{r\_type}.\{\mathsf{o\_D}\}$
MatSubClass$(\mathcal{O})$	$\mathsf{c\_Class} \sqcap \forall\mathsf{r\_type}^{-}.\exists\mathsf{r\_type}.\{\mathsf{o\_C}\} \equiv$ $\mathsf{c\_Class} \sqcap \exists\mathsf{r\_subClassOf}.\{\mathsf{o\_C}\}$ $\mathsf{c\_Class} \sqcap \forall\mathsf{r\_type}^{-}.\exists\mathsf{r\_type}.\{\mathsf{o\_D}\} \equiv$ $\mathsf{c\_Class} \sqcap \exists\mathsf{r\_subClassOf}.\{\mathsf{o\_D}\}$

Table I
$\mathcal{O}^{meta}$ OF $\mathcal{O}$

Then we add following axioms to $\mathcal{O}^{meta}$: $\mathsf{E(o\_C)}$, $\mathsf{F(o\_D)}$ and $\mathsf{E} \equiv \neg\mathsf{F}$. However, $\mathcal{O}^{meta}$ does not entail that C = D. Thus, the ontology is still consistent.

If we add following two axioms: $\mathsf{E} \sqsubseteq \forall\mathsf{subClassOf}.\neg\mathsf{F}$, $\mathsf{F} \sqsubseteq \forall\mathsf{subClassOf}.\neg\mathsf{E}$, the ontology becomes inconsistent.

Therefore, we have extended the class-based approach by following definition:

Definition 1 *For every occurrence of* C *and* D *in the form of* $\mathsf{C} \sqcap \mathsf{D} \sqsubseteq \bot$ *in the meta layer, we add* $\mathsf{C} \sqsubseteq \forall\mathsf{subClassOf}.\neg\mathsf{D}$ *and* $\mathsf{D} \sqsubseteq \forall\mathsf{subClassOf}.\neg\mathsf{C}$. *Thus, we can make use of subsumption constraints in the meta layer that can be entailed from instance layer.*

Then, the class-based approach is able to detect the contradiction present in the ontology. The results are presented in column 5 of table II.

D. OWL FA

OWL FA [15] enables metamodeling. 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.

OWL FA specifies a layer 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 $\mathbf{V}_{\mathrm{C_i}}$ (for layer i), datatype names $\mathbf{V}_{\mathrm{D}}$, abstract property names $\mathbf{V}_{\mathrm{AP_i}}$ (for layer i), datatype property names $\mathbf{V}_{\mathrm{DP}}$ 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 $\mathsf{CN} \in \mathbf{V}_{\mathrm{C_i}}$ 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, $T \in \mathbf{V}_{\mathrm{DP}}$ a datatype property name, and C, D OWL FA-classes in layer i. Valid OWL FA-classes are defined by the abstract syntax:

$$\begin{array}{rcl} C & \leftarrow & \top_{\mathrm{i}} \mid \bot \mid \mathsf{CN} \mid \neg_{\mathrm{i}} C \mid C \sqcap_{\mathrm{i}} D \mid C \sqcup_{\mathrm{i}} D \mid \{o\} \mid \\ & & \exists_{\mathrm{i}} R.C \mid \mid \forall_{\mathrm{i}} R.C \mid \leqslant_{\mathrm{i}} nR \mid \geqslant_{\mathrm{i}} nR \mid \\ & & \text{(if i = 1) } \exists_1 T.d \mid \forall_1 T.d \mid \leqslant_1 nT \mid \geqslant_1 nT \end{array}$$

The semantics of OWL FA are model theoretic semantics, which are 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. The interpretation function is explained in detail in [15].

IV. Software Model Test Cases

In this section, we detail the software model test cases that we have shown in Figure 1 from Section II. In the following example ontology, we focus on the relationship between classes and their meta-classes. Moreover, this ontology is consistent.

$$\begin{array}{rcl} \mathsf{SlotContainer} & \sqsubseteq & \mathsf{Element} \\ \mathsf{Shelf} & \sqsubseteq & \mathsf{SlotContainer} \\ \mathsf{Chassis} & \sqsubseteq & \mathsf{SlotContainer} \\ \mathsf{Chassis} & \sqsubseteq & \neg\mathsf{Shelf} \\ \mathsf{CiscoCSR}-1 & : & \mathsf{Shelf} \\ \mathsf{Cisco7600} & : & \mathsf{Chassis} \\ & \dots & \\ \mathsf{CiscoCSR}-1 & \sqsubseteq & \mathsf{Cisco} \\ \mathsf{Cisco7600} & \sqsubseteq & \mathsf{Cisco} \\ \mathsf{CiscoCSR}-1-\mathsf{Slot} & : & \mathsf{CiscoCSR}-1 \\ \mathsf{Cisco7603} & : & \mathsf{Cisco7600} \\ & \dots & \end{array} \qquad (1)$$

Figure 2. Example of complex software model that require multiple layered

A. Test Case 1

In this test case, we add the following axiom to the ontology in Figure 2.

$$\mathsf{CiscoCSR}-1 \ \sqsubseteq \ \mathsf{Cisco7600} \qquad (2)$$

We expect that the ontology will become inconsistent because axiom (1) states that Chassis and Shelf are disjoint from each other but axiom (2) states that $\mathsf{CiscoCSR}-1$ and Cisco7600 share some elements which contradict with axiom (1).

B. Test case 2

In this test case, we add the following axiom to the ontology 2.

$$\mathsf{CiscoCSR}-1 \ \equiv \ \mathsf{Cisco7600} \qquad (3)$$

We expect that the ontology will become inconsistent due to the same reason as Test case IV-A, because $\mathsf{CiscoCSR}-1$ and Cisco7600 should not share any elements.

C. Test case 3

In this test case, we remove axiom (1) from the ontology and add the following axioms into it.

$$\mathsf{CiscoCSR}-1 \ = \ \mathsf{Cisco7600} \qquad (4)$$
$$\mathsf{CiscoCSR}-1 \ \equiv \ \neg\mathsf{Cisco7600} \qquad (5)$$

We expect that the ontology will become inconsistent because axiom 4 states that meta-individual $\mathsf{CiscoCSR}-1$ is same as meta-individual Cisco7600 in the meta layer, while Class $\mathsf{CiscoCSR}-1$ is disjoint with class Cisco7600 in the instance layer.

V. Evaluation and Discussion

We used a fragment of the software model ontology (c.f. Figure 1) to test each approach. This leaves us with 3 settings as described in Section IV: we first perform consistency checking for OWL 2 metamodeling, OWL FA, the class-based approach and the extension of the class-based approach. Then, we chose OWL FA and the extension of the class-based approach to find modelling mistakes and their explanations. The main reason is that there are only two approaches that are able to detect the contradiction from our test cases. In this evaluation we use black box explanation framework[1] together with the Hermit reasoner[2] The consistency checking results are shown in Table II, and the comparison between OWL FA and the extension of class-based approach is shown in III.

Case	Expected	OWL 2	OWL FA	Class-based	Class-based2
1	No	Yes	Yes[2]	Yes	No
2	No	Yes	No	Yes	No
3	No	Yes	No	Yes	No

Table II
Consistency Checking Result

[1]http://owl.cs.manchester.ac.uk/explanation/
[2]http://hermit-reasoner.com

From Table II, OWL 2 metamodeling failed to detected the contradiction from all cases because we used one symbol to refer both to a class as well as to an individual. However, under contextual semantics, classes and individuals are interpreted independently, even if they refer to the same symbol.

OWL FA failed to discovered the contradiction in the first test case, because under fixed layered semantics, the subsumption relations in the lower layers make meta- individual different but do not make them the same as each other.

The class-based approach also failed to detect the contradiction in all test cases. Although, we can entail that $\mathsf{r\_subClassOf(o\_CiscoCSR - 1\ ,\ o\_Cisco7600)}$ and $\mathsf{r\_subClassOf(o\_Cisco7600\ ,\ o\_CiscoCSR - 1)}$, do not make $\mathsf{o\_CiscoCSR - 1 = o\_Cisco7600}$.

The extension of the class-based approach is able to find the contradiction in all cases, because we added the necessary constraints into the meta layer in order to make use of the subsumption constraints from the instance layer.

Case	OWL FA	Class-based 2
1	0[2]	1
2	1	4
3	2[3]	4

Table III
NUMBER OF JUSTIFICATION

As shown in Table III, OWL FA could discover fewer justifications than the extension of class-based approach because the extension of class-based approach compresses all three metamodeling layers into two layers. Additionally, it preserves all connections between layers and constraints as described in Section IV. This preservation increased the complexity of the model by adding extra axioms. Thus, the model and justifications in OWL FA are more straight forward than the class-based approach. We invite the reader to be note that we do not evaluate the performance in this section.

$$
\begin{aligned}
\mathsf{c\_Class} \sqcap \forall \mathsf{r\_type}^{-}.\exists \mathsf{r\_type}.\{\mathsf{o\_CiscoCSR} - 1\} &\equiv \\
\mathsf{c\_Class} \sqcap \exists \mathsf{r\_subClassOf}.\{\mathsf{o\_CiscoCSR} - 1\} \\
\mathsf{CiscoCSR} - 1 \equiv \exists \mathsf{r\_type}.\{\mathsf{CiscoCSR} - 1\} \\
\mathsf{Cisco7600} \equiv \exists \mathsf{r\_type}.\{\mathsf{Cisco7600}\} \\
\mathsf{Shelf}(\mathsf{o\_CiscoCSR} - 1) \\
\mathsf{Chassis}(\mathsf{o\_Cisco7600}) \\
\mathsf{c\_Class}(\mathsf{o\_Cisco7600}) \\
\mathsf{CiscoCSR} - 1 \equiv \mathsf{Cisco7600} \\
\mathsf{Chasis} \sqsubseteq \forall \mathsf{subClassOf}.\neg \mathsf{Shelf}
\end{aligned}
$$

Figure 3. Justification from Test Case 2 with the class-based approach

[2]The ontology is consistent based on OWL FA Semantics.

[3]One justification from $\mathcal{O}_2$ and another one from $\mathcal{O}_1$

Figure 3 and 4 show the justifications from Test Case 2. It is obvious to see that justifications in OWL FA are rather simpler than the class-based approach.

$$
\begin{aligned}
\mathsf{Chassis} \sqsubseteq \neg \mathsf{Shelf} \\
\mathsf{o\_CiscoCSR} - 1 = \mathsf{o\_Cisco7600} \\
\mathsf{Shelf}(\mathsf{o\_CiscoCSR} - 1) \\
\mathsf{Chassis}(\mathsf{o\_Cisco7600})
\end{aligned}
$$

Figure 4. Justification from $\mathcal{O}_1$ in Test Case 2 with OWL FA

In the class-based approach, generating $\mathcal{O}^{meta}$ form given $\mathcal{O}$ is cheaper than the reduction process in OWL FA, because OWL FA uses existing DL reasoners to reduce an OWL FA ontology into a set of OWL DL ontologies. This technique is more expensive than the first approach, which syntactically rewrites type relationships as type roles and class subsumptions by the $\mathsf{subClassOf}$ role, which is then axiomatically synchronised with the actual valid subclass relation in the considered model.

For reasoning, in the classed-based approach, an ontology is still a valid OWL 2 ($\mathcal{SROIQ}$) ontology, which existing DL reasoners can directly reason over; in contrast with OWL FA. OWL-FA needs to reduce reasoning to OWL DL reasoning, which is more expensive than reasoning over single ontology. For more details on reasoning in OWL FA please refer to [8].

VI. RELATED WORK

In [19] spanning objects are used in order to have different interpretations for objects that are instances and classes simultaneously.

There are various works which consider validation and consistency checking of UML models with OCL constraints like in [3], [5], [9], [16], [12]. However, none of these approaches account for a validation across multiple layers, i.e. validate models with respect to their metamodels. These approaches 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.

There are different categorisations of consistency and consistency checking as specified in [18]. The focus in this paper is to provide the ability for OWL to validate specification and instance consistency of models covering multiple modelling layers. This is a generic approach and can be applied in metamodeling and multi-dimensional metamodeling like mega-modelling [6] for linguistic and ontological metamodeling (cf. [10], [1]).

First-order logic (FOL) is used in [9] for consistency checking of UML class diagrams. The main contributions are various 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 checking.

VII. Conclusion and Outlook

In this paper, we have applied three different ontology metamodeling approaches to addressing requirements form Software Engineering that demand modelling of a complex domain model at multiple levels of abstraction. We show the pros and cons of the three existing approaches. To address the limitations of the existing approaches, we extend the class based approach to provide more configurable support for ontological metamodeling. Our experiments show that OWL FA and the extended class-based approach could benefit software modellers on leveraging the software development life cycle based on their requirements.

In the future, we would like to investigate how to combine the benefit from OWL FA and the extended class-based approaches. For instance, by syntactically reducing an OWL FA ontology to OWL DL ontology while preserving all connections between layers and generalise the approach to be used in other domains.

Acknowledgements

This work has been partially supported by the EU Marrying Ontologies and Software Technologies (MOST) project and by the State Key Laboratory for Novel Software Technology, Nanjing University. We would also like to thank Birte Glimm for her helpful comments.

References

[1] C. Atkinson and T. Kühne. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):36–41, 2003.

[2] 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.

[3] D. Berardi, D. Calvanese, and G. De Giacomo. Reasoning on UML class diagrams. *Artificial Intelligence*, 168(1-2):70–118, 2005.

[4] A. Brown. An introduction to Model Driven Architecture. IBM Technical Report. URL http://www-128.ibm.com/developerworks/rational/library/3100.html, 2004.

[5] 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.

[6] J. 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.

[7] B. Glimm, S. Rudolph, and J. Vlker. Integrated metamodeling and diagnosis in owl 2. In *Proceedings of the 9th International Semantic Web Conference*, volume 6496 of *LNCS*, pages 257–272. Springer, November 2010.

[8] N. Jekjantuk, G. Gröner, and Jeff. Z. Pan. Modelling and reasoning in metamodelling enabled ontologies. In *Knowledge Science, Engineering and Management (KSEM) 2010*, Belfast, 2010.

[9] K. Kaneiwa and K. Satoh. Consistency checking algorithms for restricted UML class diagrams. *Lecture Notes in Computer Science*, 3861:219, 2006.

[10] T. Kühne. Matters of (Meta-) Modeling. *Software and Systems Modeling*, 5(4):369–385, 2006.

[11] M. Lind and Ulf Seigerroth. Multi-layered process modeling for business and it alignment. *Hawaii International Conference on System Sciences*, 0:1–10, 2010.

[12] 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.

[13] B. Motik. On the Properties of Metamodeling in OWL. *Journal of Logic and Computation*, 17(4):617–637, 2007.

[14] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.

[15] J. Z. Pan, I. Horrocks, and G. Schreiber. OWL FA: A Metamodeling Extension of OWL DL. In *Proceeding of the International workshop on OWL: Experience and Directions (OWL-ED2005)*, 2005.

[16] 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.

[17] S. Schulz, B. Suntisrivaraporn, and Franz Baader. SNOMED CT's problem list: Ontologists' and logicians' therapy suggestions. In *Proceedings of The Medinfo 2007 Congress*, Studies in Health Technology and Informatics (SHTI-series). IOS Press, 2007.

[18] R. Van Der Straeten. *Inconsistency Management in Model-Driven Engineering*. PhD thesis, Vrije Universiteit Brussel, 2005.

[19] C. A. Welty and David A. Ferrucci. What's in an instance? Technical report, RPI Computer Science, 1994.