Int J Software Informatics, Vol.4, No.3, September 2010, pp. 277–290 E-mail: ijsi@iscas.ac.cn
International Journal of Software and Informatics, ISSN 1673-7288 http://www.ijsi.org
 Tel: +86-10-62661040

Modelling and Reasoning in Metamodelling Enabled Ontologies

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

[1] (University of Aberdeen, United Kingdom)

[2] (University of Koblenz-Landau, Germany)

Abstract Ontologies are expected to play an important role in many application domains, as well as in software engineering in general. One problem with using ontologies within software engineering is that while UML, a widely used standard for specifying and constructing the models for a software-intensive system, has a four-layer metamodelling architecture, the standard Web Ontology Language (OWL) does not support reasoning over layered metamodels. OWL 2 provides simple metamodelling by using a punning approach, however, the interpretation function is different based on the context, which leads to non-intuitive results. The OWL FA Language has a well defined metamodelling architecture. However, there is no study and tool for supporting reasoning over OWL FA. In this paper, we discuss some reasoning tasks in OWL FA. We also introduce the OWL FA Tool kit, a simple tool kit for manipulating and reasoning with OWL FA.

Key words: metamodelling; ontology; OWL FA; reasoning

Jekjantuk N, Gröner G, Pan JZ. Modelling and reasoning in metamodelling enabled ontologies. *Int J Software Informatics*, **2010, 4(3): 277–290. http://www.ijsi.org/1673-7288/4/i57.htm**

1 Introduction

Metamodelling appears in many applications areas (such as UML[13], Model Driven Architecture[2], XML[15] and E-Commerce). It is not only the underpinning of modelling languages such as UML, but also central to OMG's MDA-based computing.

The W3C Web Ontology Language (OWL)[11] in combination with reasoning is already used in various other research areas like in model-driven software engineering in order to exploit the expressiveness of OWL and the usage of inference. However, the lack of a formal OWL language or OWL extension which supports metamodelling is an obstacle for the usage of OWL in other complex application areas.

The Resource Description Framework (RDF) and OWL Full support metamodelling by allowing users to use the built-in vocabulary without restrictions, which introduces an undecidability problem. OWL[11] provides formal semantics focused on conceptual modelling and adaptability of inference using DL reasoners and reasoning algorithms, but OWL does not support layered reasoning. OWL 2 provides simple

* This work is sponsored by the European Project Marrying Ontologies and Software Technologies (MOST ICT 2008-216691)
Corresponding author: Nophadol Jekjantuk, Email: n.jekjantuk@abdn.ac.uk
Received 2010-09-13; accepted 2010-10-29

metamodelling with semantics which correspond to the contextual semantics defined in Ref.[6]. However, it has been shown in Ref.[9] that these can lead to non-intuitive results.

For example, the following axioms state that *Eagle* is an *Endangered* species, and that *Harry* is an *Eagle*:

$$\mathsf{ClassAssertion(Endangered\ Eagle)} \tag{1.1}$$

$$\mathsf{ClassAssertion(Eagle\ Harry)} \tag{1.2}$$

The axioms 1.1, 1.2 could be interpreted by DL reasoner as follows:

$$\mathsf{ClassAssertion(Cls-Endangered\ Ind-Eagle)} \tag{1.3}$$

$$\mathsf{ClassAssertion(Cls-Eagle\ Ind-Harry)} \tag{1.4}$$

The names of classes and individuals do not interact with each other even if they are sharing the same name, e.g., *Eagle* is represented as individual by the name $\mathsf{Ind-Eagle}$ and as class by the name $\mathsf{Cls-Eagle}$. This kind of metamodelling is often referred to as punning. Let us consider the following axioms:

$$\mathsf{SameIndividuals(Aquila\ Eagle)} \tag{1.5}$$

$$\mathsf{ClassAssertion(not(Aquila)\ Harry)} \tag{1.6}$$

The axioms 1.5, 1.6 could be safely added to the ontology in contextual semantics, but under layered semantics this ontology is inconsistent because 1.5 indicates the meta-individual equality($\mathsf{owl:sameAs}$) since the axiom $\mathsf{Eagle} \approx \mathsf{Aquila}$ indicates the equivalence of the two classes Eagle and Aquila. However, axiom 1.6 describes that Harry is not in Aquila which leads to the contradiction in combination with axiom 1.4.

In this paper, we present modelling and reasoning algorithms for OWL FA knowledge bases. For the reasoning service, an OWL FA ontology is transformed to a set of OWL DL ontologies, then existing DL reasoners are applied to the transformed knowledge base. The syntax and semantics of OWL FA is described in Section 1. Modelling in OWL FA is demonstrated in Section 3. In Section 4, reasoning in OWL FA is described. This contains a reduction to OWL DL knowledge bases and reasoning algorithms in order to propagate conditions between different modelling layers. Features of the OWL FA Tool Kit are described in detail in Section 5. The early evaluation is presented in Section 6. Then, related work and the direction of OWL FA are discussed in Section 7 and Section 8 respectively.

2 OWL FA Syntax and Semantics

OWL FA[9] enables metamodelling. It is an extension of OWL DL, which is equivalent 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)[8].

OWL FA specifies a layer number in class constructors and axioms to indicate the layer they belong to. Let $i \geqslant 0$ 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 $\mathbf{V}_{\mathrm{APi}}$ (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).

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 $\mathbf{B} \subseteq \mathbf{V}_{\mathrm{D}}$ and an integer $\mathrm{k} \geqslant 1$, 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 $0 \leqslant \mathrm{i} \leqslant \mathrm{k}$):

1. $\Delta^{\mathcal{J}} = \bigcup_{0 \leqslant \mathrm{i} \leqslant \mathrm{k}-1} \Delta_{A_\mathrm{i}}^{\mathcal{J}} \cup \Delta_{\mathbf{D}}$, where $\Delta_{A_\mathrm{i}}^{\mathcal{J}}$ is the domain for layer i and $\Delta_{\mathbf{D}}$ is the datatype domain;
2. $\Delta_{A_{\mathrm{i}+1}}^{\mathcal{J}} = 2^{\Delta_{A_\mathrm{i}}^{\mathcal{J}}} \cup 2^{\Delta_{A_\mathrm{i}}^{\mathcal{J}} \times \Delta_{A_\mathrm{i}}^{\mathcal{J}}}$ and $\Delta_{\mathbf{D}} \cap \Delta_{A_\mathrm{i}}^{\mathcal{J}} = \emptyset$;
3. $\forall \mathsf{a} \in \mathbf{V}_{\mathrm{I}} :\ \mathsf{a}^{\mathcal{J}} \in \Delta_{A_0}^{\mathcal{J}}$ and $\forall \mathsf{C} \in \mathbf{V}_{\mathrm{C}_{\mathrm{i}+1}} :\ \mathsf{C}^{\mathcal{J}} \subseteq \Delta_{A_\mathrm{i}}^{\mathcal{J}}$;
4. $\forall R \in \mathbf{V}_{\mathrm{APi}+1} :\ R^{\mathcal{J}} \subseteq \Delta_{A_\mathrm{i}}^{\mathcal{J}} \times \Delta_{A_\mathrm{i}}^{\mathcal{J}}$ and $\forall T \in \mathbf{V}_{\mathrm{DP}} :\ T^{\mathcal{J}} \subseteq \Delta_{A_0}^{\mathcal{J}} \times \Delta_{\mathbf{D}}$;
5. $\bigcup_{\forall d \in \mathbf{B}} V(d) \subseteq \Delta_{\mathbf{D}}$, where $V(d)$ is the value space of d;
6. $\forall d \in \mathbf{V}_{\mathrm{D}}$, if $d \in \mathbf{B}$, then[1]
   (a) $d^{\mathcal{J}} = V(d)$, where $V(d)$ is the value space of d,
   (b) if $v \in L(d)$, then $(\text{“}v\text{”}\hat{}\hat{}d)^{\mathcal{J}} = L2V(d)(v)$, 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 $(\text{“}v\text{”}\hat{}\hat{}d)^{\mathcal{J}}$ is undefined;

   otherwise, $d^{\mathcal{J}} \subseteq \Delta_{\mathbf{D}}$ and $\text{“}v\text{”}\hat{}\hat{}d \in \Delta_{\mathbf{D}}$.

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

Let $\mathsf{CN} \in \mathbf{V}_{\mathrm{C_i}}$ be an atomic class name in layer i, 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:

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

The semantics of OWL FA-classes are presented in Table 1 (page 4).

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 $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$.

[1] To simplify the presentation, we do not distinguish datatype names and datatype URIrefs here.

Table 1. Database characteristics

Constructor	DL Syntax	Semantics
top	$\top_{\mathrm{i}}$	$\Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}}$
bottom	$\bot$	$\emptyset$
concept name	CN	$\mathsf{CN}^{\mathcal{J}} \subseteq \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}}$
general negation	$\neg_{\mathrm{i}} C$	$\Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}} \setminus C^{\mathcal{J}}$
conjunction	$C \sqcap_{\mathrm{i}} D$	$C^{\mathcal{J}} \cap D^{\mathcal{J}}$
disjunction	$C \sqcup_{\mathrm{i}} D$	$C^{\mathcal{J}} \cup D^{\mathcal{J}}$
nominals	$\{\mathsf{o}\}$	$\{\mathsf{o}\}^{\mathcal{J}} = \{\mathsf{o}^{\mathcal{J}}\}$
exists restriction	$\exists_{\mathrm{i}} R.C$	$\{x \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}} \mid \exists y.\langle x,y\rangle \in R^{\mathcal{J}} \wedge y \in C^{\mathcal{J}}\}$
value restriction	$\forall_{\mathrm{i}} R.C$	$\{x \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}} \mid \forall y.\langle x,y\rangle \in R^{\mathcal{J}} \rightarrow y \in C^{\mathcal{J}}\}$
atleast restriction	$\geqslant_{\mathrm{i}} mR$	$\{x \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}} \mid \sharp\{y \mid \langle x,y\rangle \in R^{\mathcal{J}}\} \geqslant m\}$
atmost restriction	$\leqslant_{\mathrm{i}} mR$	$\{x \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}} \mid \sharp\{y \mid \langle x,y\rangle \in R^{\mathcal{J}}\} \leqslant m\}$
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	$\geqslant_1 mT$	$\{x \in \Delta_{A_0}^{\mathcal{J}} \mid \sharp\{t \mid \langle x,t\rangle \in T^{\mathcal{J}}\} \geqslant m\}$
datatype atmost restriction	$\leqslant_1 mT$	$\{x \in \Delta_{A_0}^{\mathcal{J}} \mid \sharp\{t \mid \langle x,t\rangle \in T^{\mathcal{J}}\} \leqslant m\}$

An OWL FA ontology Σ consists of $\Sigma_1, \ldots, \Sigma_{\mathrm{k}}$. Each Σ_{i} consists of a TBox $\mathcal{T}_{\mathrm{i}}$, an RBox $\mathbb{R}_{\mathrm{i}}$ and an ABox $\mathcal{A}_{\mathrm{i}}$. An OWL FA *TBox* $\mathcal{T}_{\mathrm{i}}$ is a finite set of class inclusion axioms of the form $C \sqsubseteq_{\mathrm{i}} D$, where C, D are OWL FA-classes in layer i. An interpretation $\mathcal{J}$ satisfies $C \sqsubseteq_{\mathrm{i}} D$ if $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$. Let R, S be OWL FA abstract properties in layer i. An OWL FA *RBox* $\mathbb{R}_{\mathrm{i}}$ is a finite set of property axioms; namely, property inclusion axioms ($R \sqsubseteq_{\mathrm{i}} S$), functional property axioms ($\mathsf{Func}_{\mathrm{i}}(R)$) and transitive property axioms ($\mathsf{Trans}_{\mathrm{i}}(R)$). An interpretation $\mathcal{J}$ satisfies $R \sqsubseteq_{\mathrm{i}} S$ if $R^{\mathcal{J}} \subseteq S^{\mathcal{J}}$; $\mathcal{J}$ satisfies $\mathsf{Func}_{\mathrm{i}}(R)$ if, for all $x \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}}$, $\sharp\{y \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}} \mid \langle x,y\rangle \in R^{\mathcal{J}}\} \leqslant 1$ ($\sharp$ denotes cardinality); $\mathcal{J}$ satisfies $\mathsf{Trans}_{\mathrm{i}}(R)$ if, for all $x, y, z \in \Delta_{A_{\mathrm{i}-1}}^{\mathcal{J}}$, $\{\langle x,y\rangle, \langle y,z\rangle\} \subseteq R^{\mathcal{J}} \rightarrow \langle x,z\rangle \in R^{\mathcal{J}}$. 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 are no transitive datatype property axioms.

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

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

interpretation $\mathcal{J}$ of Σ we have $C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$.

3 Modelling of Metamodelling Enabled Ontologies

In this section, we present the way to express metamodelling enabled ontologies in OWL 2. The layer information is encapsulated in custom annotation property called "Layer". This is different from[5] because we realised that creating a new syntax for OWL FA is unnecessary since we could store layer information as annotation properties. Moreover, this ontology conforms still to the OWL 2 syntax like the punning style which is another way to capture a simple modelling in OWL 2. 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 of the 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 an Endangered Species ontology with the DL syntax and then convert it into OWL 2 functional syntax. The main reason for using functional syntax is that it is obvious to see which layer they belong to.

Example 3.1. Endangered Species ontology expressed in DL syntax as follow:

$$\mathsf{Eagle} \;:_2\; \mathsf{Endangered} \tag{3.7}$$
$$\mathsf{Aquila} \;:_2\; \mathsf{Endangered} \tag{3.8}$$
$$\mathsf{Aquila} \;\approx_2\; \mathsf{Eagle} \tag{3.9}$$
$$\mathsf{Eagle} \;\sqsubseteq_1\; \mathsf{Bird} \tag{3.10}$$
$$\mathsf{Aquila} \;\sqsubseteq_1\; \mathsf{Bird} \tag{3.11}$$
$$\mathsf{Harry} \;:_1\; \mathsf{Eagle} \tag{3.12}$$

Example 3.2. Endangered Species ontology expressed in OWL 2 syntax as follow:

$$\mathsf{ClassAssertion(Annotation(Layer"2")Endangered\ Eagle)} \tag{3.13}$$
$$\mathsf{ClassAssertion(Annotation(Layer"2")Endangered\ Aquila)} \tag{3.14}$$
$$\mathsf{SameIndividuals(Annotation(Layer"2")Eagle\ Aquila)} \tag{3.15}$$
$$\mathsf{SubClassOf(Annotation(Layer"1")Eagle\ Bird)} \tag{3.16}$$
$$\mathsf{SubClassOf(Annotation(Layer"1")Aquila\ Bird)} \tag{3.17}$$
$$\mathsf{ClassAssertion(Annotation(Layer"1")Eagle\ Harry)} \tag{3.18}$$

4 Reasoning in OWL FA

Now we briefly discuss some reasoning tasks in OWL FA. According to the layered architecture, the knowledge base Σ in OWL FA is divided into a sequence of knowledge bases $\Sigma = \Sigma_1, \ldots, \Sigma_k$, whereas k is the number of layers. Since individuals in layer $i+1$ can be classes and properties in layer i, this also affects the axioms of the layer below. Hence, individual axioms in the knowledge base Σ_{i+1} can be considered as class axioms in the knowledge base Σ_i.

In an OWL FA knowledge base Σ, Σ_2, ..., Σ_k are $\mathcal{SHIQ}$ knowledge bases, i.e. nominals are not allowed. A nominal in a higher layer can lead to unsatisfiability of the knowledge bases. An interesting feature of Σ is that there could be interactions between Σ_i and Σ_{i+1}.

4.1 Preprocessing

In this section, we discuss how to reduce the reasoning problem in OWL FA into a reasoning problem in OWL DL.

Definition 4.1. Let $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ be an OWL FA knowledge base, where each of $\Sigma_1, \ldots, \Sigma_k$ is consistent. $\Sigma^* = \langle \Sigma_1^*, \ldots, \Sigma_k^* \rangle$, called the *explicit knowledge base*, is constructed by making all the implicit atomic class axioms, atomic property axioms, individual equality axioms explicit.

As we have a finite set of vocabulary, we have the following Lemma.

Lemma 4.1. Given an OWL FA knowledge base $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$. The explicit knowledge base Σ^* (OWL DL knowledge base) can be computed from Σ in finite steps.

Proof: When $k = 1$, we can calculate the explicit knowledge base Σ_1^* in finite steps because the sets of names of classes (in layer 1), properties (in layer 1) and individuals are finite. When $k > 1$, let us assume that we can calculate the explicit knowledge bases $\Sigma_1', ..., \Sigma_i'$ (where $1 \leqslant i < k$) from $\Sigma_1, \ldots, \Sigma_i$ in finite steps. We add all the class and property equality axioms in Σ_i' to Σ_{i+1}. If the updated Σ_{i+1} is consistent. Then,we can make the implicit individual equality axioms (if any) explicit and add new class and property equality axioms into Σ_i'. Thus, we can calculate Σ_1'', ..., Σ_i'' in finite steps. As the individual names in Σ_{i+1} are finite, we can calculate the explicit knowledge bases $\Sigma_1^*, \ldots, \Sigma_{i+1}^*$ in finite steps.

Note that if a class description is not defined in Σ_i (i.e., if it is not equivalent to any atomic class), it is not represented by any meta-individual in Σ_{i+1}. This suggests the connections between Σ_i and Σ_{i+1} are atomic classes and properties in Σ_i, which are meta-individuals in Σ_{i+1}.

We now present the algorithm `Reduce`, that will reduce an OWL FA knowledge base Σ into a set of OWL DL knowledge bases $\langle \Sigma_1^*, \ldots, \Sigma_k^* \rangle$. This algorithm is based on Definition 4.1 and Lemma 4.1. The algorithm takes an OWL FA KB Σ as input and returns a set of OWL DL KB $\langle \Sigma_1, \ldots, \Sigma_k \rangle$. The Algorithm `Reduce` is shown in Algorithm 1.

The following theorem shows the termination of the algorithm `Reduce`, applied to an OWL FA KB Σ.

Theorem 4.1. Given an OWL FA knowledge base $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$, then `Reduce`($\Sigma$) terminates.

Proof: Termination of algorithm `Reduce` is straightforward from Lemma 4.1, which we can construct $\langle \Sigma_1^*, \ldots, \Sigma_k^* \rangle$ from Σ in finite step and a sets of class, property and individual equality axioms are finite. Thus, algorithm `Reduce` always terminates.

Here is the result from applying the Algorithm `Reduce` to the OWL FA KB Σ:
$\Sigma_2 = \{\mathsf{Endangered} : \mathsf{Eagle}, \mathsf{Endangered} : \mathsf{Aquila}, \mathsf{Eagle} = \mathsf{Aquila}\}$
$\Sigma_1 = \{\mathsf{Eagle} : \mathsf{Harry}, \mathsf{Eagle} \sqsubseteq \mathsf{Bird}, \mathsf{Aquila} \sqsubseteq \mathsf{Bird}, \mathsf{Aquila} \equiv \mathsf{Eagle}\}$

4.2 Consistency checking

In this section, we present the algorithm `Consistent`, that will check the consistency of an OWL FA knowledge base $\mathcal{O}$. We can reduce an OWL FA knowledge base to a collection of OWL DL knowledge bases, therefore existing DL reasoner capabilities can be used. Consistency checking for OWL FA is done in two steps: First, we check the syntax of OWL FA. For example, $\Sigma = \{C \sqsubseteq_2 D, C \sqsubseteq_3 E\}$ is non-well formed because in OWL FA we do not allow OWL class construct between layer except an instance-of relationship. Secondly, we check the consistency of each OWL DL-knowledge base that is computed from the OWL FA knowledge base with an existing DL reasoner. The Algorithm `Consistent` is shown in Algorithm 2.

Algorithm 1 Reduce

Input: OWL FA KB Σ
Output:satisfiable and a set of OWL DL KB$\langle\Sigma_1, \ldots, \Sigma_k\rangle$.

1: boolean satisfiable = true;
2: Collect axioms from the layer number and store in $L_0, ..., L_n$
3: Create knowledge base $\Sigma_i = (L_0, L_1), ..., \Sigma_k = (L_{n-1}, L_n)$
4: **repeat**
5: $ce_i = \emptyset, pe_i = \emptyset$ *and* $oe_i = \emptyset$
6: Check consistency of Σ_i with DL reasoner
7: **if** Σ_i is consistent **then**
8: **for** each Σ^*_i $(1 \leqslant i \leqslant k)$ **do**
9: Identify the new class equality in Σ^*_i and store it in ce_i
10: Identify the new property equality in Σ^*_i and store it in pe_i
11: Identify the new individual equality in Σ^*_i and store it in oe_i
12: **for** each ce_i, pe_i and ce_i **do**
13: **if** i = 1 **then**
14: Add ce_i and pe_i as individual equality into Σ^*_{i+1}
15: **else if** i = k **then**
16: Add oe_i as property or class equality into Σ^*_{i-1}
17: **else**
18: Add ce_i and pe_i as individual equality into Σ^*_{i+1}
19: Add oe_i as property or class equality into Σ^*_{i-1}
20: **end if**
21: **end for**
22: **end for**
23: **else**
24: satisfiable = false;
25: **end if**
26: **until** ($ce_i = \emptyset$ & $pe_i = \emptyset$ & $oe_i = \emptyset$) || satisfiable = false;
27: **return** $\langle\Sigma_1, \ldots, \Sigma_k\rangle$.

We invite the reader to note that *check-dl-consistent* is a function call to a DL Reasoner.

Algorithm 2 Consistent

Input: OWL FA Knowledge Base $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$
Output: *true* if Σ is consistent, *false* otherwise

```
1: Ont = ∅
2: Check OWL FA syntax
3: Ont = Reduce(Σ);
4: for each Σ_i in Ont do
5:    check-dl-consistent(Σ_i)
6:    if Σ_i is not consistent then
7:       Return false
8:    end if
9: end for
10: return true.
```

Theorem 4.2. Given an OWL FA knowledge base $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$. Σ is consistent iff each Σ_i^* $(1 \leqslant i \leqslant k)$ is consistent.

Theorem 4.2 shows that we can reduce the OWL FA-knowledge base consistency problem to the OWL DL-knowledge base consistency problem.

Proof: The consistency check of an OWL FA KB with Consistent(Σ) is straightforward from Lemma 4.1. We can construct $\langle \Sigma_1^*, \ldots, \Sigma_k^* \rangle$ from Σ in finite steps then we check the consistency for each Σ_i^* with a DL reasoner. Therefore, the OWL FA knowledge base Σ is consistent if and only if each Σ_i^* $(1 \leqslant i \leqslant k)$ is consistent.

Let us consider the following axiom by inserting it into OWL FA KB Σ (cf. Example 3.2):

$$\mathsf{ClassAssertion(Annotation(Layer"1")Not(Aquila)\ Harry)} \tag{3.19}$$

It is obvious to see that this axiom will make Σ_1^* inconsistent, which leads to an inconsistent OWL FA KB Σ because the meta-individual equality axiom $\mathsf{Eagle} \approx_2 \mathsf{Aquila}$ indicates the equivalence of the two classes Eagle and Aquila, and $\mathsf{Harry}^{\mathcal{J}}$ cannot be both in and not in $Eagle^{\mathcal{J}}$.

4.3 Instance retrieval

Instance retrieval in OWL FA is trivial because after the reduction process, we get a set of OWL 2 DL ontology then we could perform instance retrieval against those ontologies. However, without specifying a target ontology, it is not efficient since, we have to go through all ontologies in a set. Therefore, we need a smart algorithm for instance retrieval for OWL FA, in order to select the right ontology that contains a target class. Firstly, we need to search for a target class in each ontology of the set. This step does not require any DL reasoner. Then, we could perform instance retrieval against a selected ontology with a DL reasoner. A formal definition of instance retrieval for OWL FA is given in Definition 4.2.

Definition 4.2. Given an *ABox* $\mathcal{A}_i$ and a query Q, i.e., a class expression, find all individuals a such that a is an instance of Q, i.e., $\{a \mid \forall a \in \mathcal{A}_i, a : Q\}$. $\diamond$

We present the instance retrieval for OWL FA in Algorithm 3. The algorithm *instanceOf* will take an OWL FA ontology $\mathcal{O}$ and a class C as input. The algorithm returns a set containing the instances of class C.

Algorithm 3 instanceOf

Input: OWL FA Knowledge Base $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ and a class C
Output: A set contains instance of class C

```
1: ind = ∅
2: Ont = ∅
3: Ont = Reduce(Σ);
4: for each Σi in Ont do
5:    if Σi contains C then
6:       ind = get-dl-instance-of(Σi, C)
7:    end if
8: end for
9: return ind.
```

4.4 Justification on OWL FA

A justification for an entailment in an OWL FA ontology can be extended from justification for an entailment in an OWL DL ontology because we can reduce the reasoning problem in OWL FA to a reasoning problem in OWL DL. However, in the reduction process, a new axiom can be added to an OWL DL ontology. Therefore, if a justification for an entailment axiom in $\mathcal{O}_i$ contains those new axioms which have been added during the reduction process, we need to store an information for that axiom from another ontology in the lookup table. From an upper ontology, if the new axiom has been added as class or property equality axioms, we can map those axioms from class or property equality axioms to individual equality axioms in the $\mathcal{O}_{i+1}$. Hence, we can retrieve the further justification from the upper ontology if needed. From a lower ontology, if a justification contains individual equality axioms, we can map those individual equality axioms to class or property equality axioms in the $\mathcal{O}_{i-1}$ then we can retrieve the further justification from the lower ontology if needed.

Definition 4.3. For an OWL FA ontology $\Sigma = \langle \Sigma_1, \ldots, \Sigma_k \rangle$ and an entailment η_i where i is a layer number, a set of axioms $\mathcal{J}_i$ is a justification for η_i in Σ i. $\mathcal{J}_i$ may contain further justifications from $\mathcal{O}_{i+1}$ and/or $\mathcal{O}_{i-1}$ if the ontology Σ has added class or property equality axioms and/or added individual equality axioms, respectively. The further justification can be retrieved from the information stored in the lookup table.

In order to keep trace of the new axioms that have been added in the reduction process, we need extend the algorithm `Reduce` in Section 4.1. The algorithm takes an OWL FA KB Σ as input and returns a set of OWL DL KB $\langle \Sigma_1, \ldots, \Sigma_k \rangle$ and a lookup table between new axioms and the original axiom. The lookup table is indexed by the layer, i.e. the axioms in the table refer to the layer which contains additional axioms after the reduction. These additional axioms affects also the justifications for this layer. Hence, we will later exploit the information from the lookup table about added axioms to compute the justification. We now present the algorithm `Justification`, that will retrieve a justification for an entailment in an OWL FA ontology. The algorithm takes an OWL FA KB Σ and an entailment η_i as input and returns a set of justification axioms $\mathcal{J}$. The algorithm `Justification` is shown in Fig. 4.

We invite the reader to note that *compute-dl-justification* is a function call to a

Algorithm 4 The algorithm for computing a single justification for FA

Algorithm Justification(Σ)
Input: OWL FA KB Σ and an entailment η_i
Output: a set of justification axioms $\mathcal{J}$.
begin
  $\Sigma^* = \text{Reduce}(\Sigma)$;
  set $\mathcal{J}_i$ = compute-dl-justification(Σ_i^*, η_i)
  if $\mathcal{J}_i$ contains axioms in MP_i **then**
    if axiom in MP_i is class or property equality **then**
      $\mathcal{J}_{i+1}$ = compute-dl-justification(Σ_{i+1}^*, axiom)
    end if
    if axiom in MP_i is individual equality **then**
      $\mathcal{J}_{i-1}$ = compute-dl-justification(Σ_{i-1}^*, axiom)
    end if
  end if
end

DL Reasoner. The mapping table MP_i indicates added axioms to the ontology by the reduction. In this case, the adjacent ontology Σ_{i+1}^* or Σ_{i-1}^* is also relevant for the justifications of the ontology Σ_i^*.

Due to space limitations, we cannot describe all reasoning tasks for OWL FA in this paper, however, since we can reduce OWL FA into a set of OWL DL ontologies then all existing DL reasoner's capabilities can be used.

5 OWL FA Tool Kit

In this section, we reintroduce the OWL FA Tool Kit, a simple graphic user interface for modeller to create an OWL FA ontology and perform reasoning over it. The screen capture of OWL FA Tool Kit is shown in Fig.1. The OWL FA Tool Kit contains features as follows:

- Editor - for checking the OWL FA ontology before performing the reasoning.
- Ontology Consistency Checker - for checking whether a given metamodelling enabled ontology is consistent.
- Class Satisfiability Checker - for verifying whether a class A is a non-empty set in a given OWL FA ontology $\mathcal{O}$.
- Query Answering - for accessing information from a given metamodelling enabled ontologies by using SPARQL queries.
- Export a collection of OWL DL to files - for separating the domain knowledge from its meta knowledge.

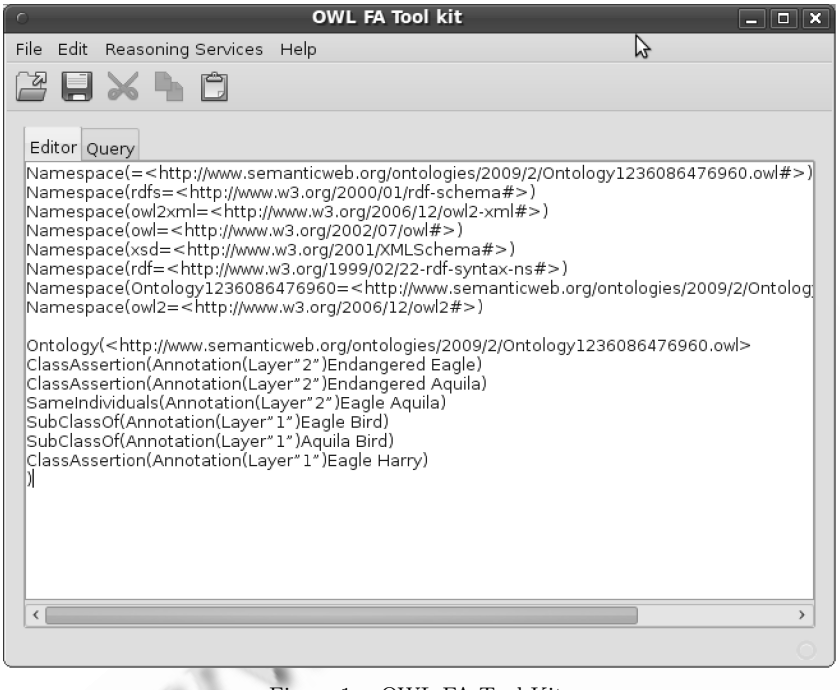

Figure 1. OWL FA Tool Kit

6 Evaluation

In this section, we compare the metamodelling in OWL FA with OWL 2 as OWL 2 is the only OWL language that can support metamodelling and it has tools support.

6.1 Use case 1: Consistency checking

OWL 2 provides simple metamodelling with semantics which correspond to the contextual semantics defined in Ref.[6], however, it has been shown in Ref.[9] that these can lead to non-intuitive results.

Let us consider an ontology from Example 3.2 in Section 3 and the axioms 4.19. This ontology is consistent when we perform consistency checking with any existing DL reasoner. The existing DL reasoner does not take layer information into account which are described as annotation property and it interpret this ontology with contextual semantics.

This ontology is inconsistent based on layered architecture in OWL FA as we described in Section 4. OWL FA Tool Kit takes layer information into account and maintain a relationship between elements that share the same URIs.

6.2 Use case 2: Instance retrieval

In OWL 2, if we do not provide the explicit instantiation between class and individual, it is difficult for any existing DL reasoner to discover those information because in contextual semantics, classes and individuals are interpreted independently. Let's consider an ontology from Example 3.2 and remove axioms 3.14 and 3.15 from the

ontology. Then, we would like to retrieve all individuals that belong to *Endangered*. Without adding an axiom to indicate that *Aquila* is an *Endangered* then, *Aquila* is not included in the answer set. Although, the class *Aquila* is equivalent to the class *Eagle* but the interpretations of the class *Eagle* and the individual *Eagle* are independent from each other. Therefore, the existing DL reasoners could not found the relation between *Aquila* and *Endangered*.

For OWL FA and our tool kit, it returns more complete answer sets as shown in Fig.2, because in the reduction process, we propagate all class and property equality axioms to be individual equality axioms in the higher layer and propagate all individual equality axioms to be class or property equalities in the lower layer.

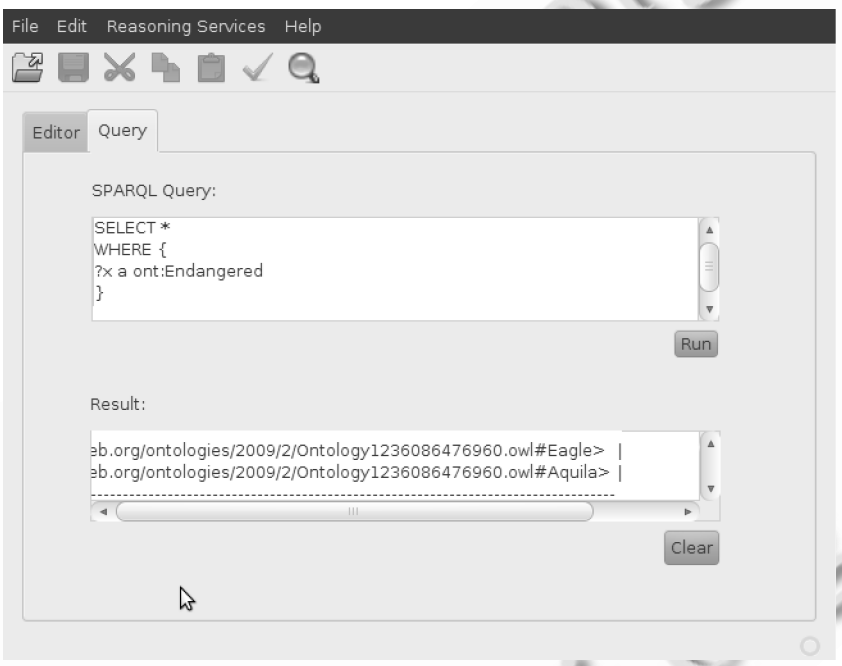

Figure 2. Instance retrieval with OWL FA Tool Kit

7 Related Work

OWL FA was introduced in Ref.[9] as a metamodelling extension of OWL. Motik[6] 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 classes, individuals and roles. This is like the different representation of an object in the OWL DL ontologies Σ_i in OWL FA. OWL2[7] 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 interpretation of class and individual are not independent.

De Giacoma et al.[4] proposed the $\mathsf{HiDL-Lite}$ 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 reducing $\mathsf{HiDL-Lite}$ 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 $\mathsf{HiDL-Lite}$that semantics of the meta-level need to re-define.

Description Logic reasoning is applied to UML models in Refs.[1, 14, 3]. The models are transformed into DL representations. Reasoning is used to check consistency of models and between models. However, metamodels are not considered.

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. One would like to define a constraint on a meta-class Endangered that all instances of these meta-class have only 3000 individuals. Therefore, if a class Eagle is an instance of the meta-class Endangered, the constraint should be applied to the class 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 this Endangered requirement as $\top \sqsubseteq \leqslant 3000 \bigcup .\mathsf{Endangered}$ then propagate this constraint to all instances of Endangered. 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 an ObjectProperty assertion in layer $\mathsf{M_2}$, Endangerd liveIn Continent, which is expressed by the ObjectProperty liveIn. This constraint can only be used to validate the model between the layers $\mathsf{M_2}$ and $\mathsf{M_1}$. We plan 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_liveIn that would still be an object property in the $\mathsf{M_2}$ layer and there the object property assertions in layer $\mathsf{M_1}$ remain unchanged like liveIn(Eagle, Europe). However, we could also propagate this property assertion from model $\mathsf{M_1}$ semantically to model $\mathsf{M_0}$. For instance the property assertion meta_liveIn(Eagle, Europe) in $\mathsf{M_1}$ becomes the subclass axiom $\mathsf{Eagle} \sqsubseteq \exists \mathsf{lineIn.Europe}$ in the model $\mathsf{M_0}$. 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.

9 Conclusion

In this paper, we reintroduce OWL FA language and demonstrate how to model the metamodelling enabled ontology, followed by a description of reasoning in OWL FA for different reasoning tasks. And the reduction from an OWL FA knowledge base into OWL DL knowledge bases algorithms are describes. These algorithms use standard DL reasoning as a black-box service. Based on the given examples, a metamodelling enables ontology is described in OWL 2 DL.

We have shown that we can make use of the existing DL reasoners to reason over OWL FA knowledge base. As we discussed in section 4, we can compute the explicit OWL DL knowledge base Σ_{i}^* from OWL FA knowledge base Σ^*.

We have implemented the OWL FA Tool Kit for modeller to manipulating and

reasoning over OWL FA standard and plan to incorporate these into the TrOWL[2] reasoning infrastructure.

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. And We plan to use quality guaranteed approximations[10,12] reasoner instead of traditional DL reasoner in order to improve scalability and efficiency of the tool kit.

References

[1] Berardi D, Calvanese D, De Giacomo G. Reasoning on UML Class Diagrams. Artificial Intelligence, 2005, 168(1-2): 70–118.

[2] Alan Brown. An introduction to Model Driven Architecture. IBM Technical Report. http://www-128.ibm.com/developerworks/rational/library/3100.html

[3] Cali A, Calvanese D, De Giacomo G, Lenzerini M. Reasoning on UML Class Diagrams in Description Logics. Proc. of IJCAR Workshop on Precise Modelling and Deduction for Object-Oriented Software Development (PMD), 2001.

[4] De Giacomo, Giuseppe, Maurizio Lenzerini, Riccardo Rosati. Towards higher-order DL-Lite (preliminary report). Proc. of the International Workshop on Description Logic (DL-2008), Dresden, Germany, May 13-16, 2008.

[5] Jekjantuk N, Gröner G, Pan JZ. Reasoning in Metamodeling Enabled Ontologies. Proc. of the International workshop on OWL: Experience and Directions (OWL-ED2009), 2009.

[6] Boris Motik. On the Properties of Metamodeling in OWL. J. Log. Comput., 2007, 17(4): 617–637.

[7] Boris Motik, Peter F. Patel-Schneider, 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.

[8] Pan JZ, Horrocks I. RDFS(FA) and RDF MT: Two Semantics for RDFS. Proc. of the 2nd International Semantic Web Conference (ISWC2003), 2003.

[9] Pan JZ, Horrocks I, Schreiber G. OWL FA: A Metamodeling Extension of OWL DL. Proc. of the International Workshop on OWL: Experience and Directions (OWL-ED2005), 2005.

[10] Pan JZ, Thomas E. Approximating OWL-DL Ontologies. Proc. of AAAI, 2007.

[11] Peter F, Patel-Schneider, Patrick Hayes, et al. OWL Web Ontology Language Semantics and Abstract Syntax. Technical report. W3C, Feb. 2004. W3C Recommendation.

[12] Ren Y, Pan JZ, Zhao Y. Soundness Preserving Approximation for TBox Reasoning. Proc. of the 25th AAAI Conference Conference (AAAI2010), 2010.

[13] UML. Unified Modeling Language. http://www.uml.org/.

[14] Van Der Straeten R, Simmonds J, Mens T. Detecting Inconsistencies between UML Models using Description Logic. Proc. of the 2003 International Workshop on Description Logics (DL2003), Rome, Italy, 2003, 81: 260–264.

[15] W3C. Extensible Markup Language (XML). http://www.w3.org/XML/, 2001.

[2] http://www.trowl.eu