

# On Metamodeling in Megamodels

Dragan Gašević<sup>1</sup>, Nima Kaviani<sup>2</sup>, and Marek Hatala<sup>2</sup>

<sup>1</sup> Athabasca University, Canada

<sup>2</sup> Simon Fraser University Surrey, Canada

dragang@athabascau.ca, {nkaviani, mhatala}@sfu.ca

**Abstract.** Model-Driven Engineering (MDE) introduced the notion of metamodeling as the main means for defining modeling languages. As a well organized engineering discipline, MDE should also have its theory clearly defined in terms of the relationships between key MDE concepts. Following the spirit of MDE, where models are first class citizens, even the MDE theory can be defined by models, or so called megamodels. In this paper, we use Favre's megamodel that was already used for defining linguistic metamodeling. Starting from the premise that this megamodel can also be used for defining other MDE concepts, we use it to specify the notion of ontological metamodeling. Here, we show that in order for this megamodel to be able to fully capture all the concepts of ontological metamodeling, some refinements should be applied to its definition. We also show how these new changes are in the same direction with the work of Kühne in defining linguistic and ontological metamodels.

## 1 Introduction

The idea of Model Driven Engineering (MDE) stems from software engineering, and more specifically, from the recent research in software development. MDE evolved as the paradigm shifted from the object-oriented approach where the main principle is that *everything is an object* into the model engineering paradigm based on the principle that *everything is a model* [6]. The object-oriented technology is about classes and objects, and main relations are *instantiation* (an object is an instance of a class) and *inheritance* (a class inherits from another class). MDE is about models, but it is also about relations between a model and the system under study (which can be a software artifact or a real world domain), metamodels, and model transformations. Similar to the object-oriented technology, MDE can be characterized by two main relations, namely, *representation* (a model represents a software artifact or real world domain) and *conformance* (a model conforms to a metamodel). Generally speaking, MDE is a field of system engineering in which the process heavily relies on the use of models and model engineering. Here, model engineering is considered as a disciplined and rationalized production of models [11].

For MDE to be popularized and accepted by the software engineering community, the theory behind it should be precise and easy to grasp. Furthermore, this theory must be comprehensive enough to address all the phenomena related to the languages

and metamodels used in MDE (e.g. OWL, XML, and UML). It makes the study of MDE’s theory critical. Modeling, as the spirit of MDE, is arguably the best method to study the theory of MDE, especially because of its natural ability in simplifying the specification and description of the intended goal in mind [7].

Atkinson & Kühne [1, 2, 15, 16] and Favre [11, 12] have taken important steps in clarifying the theory of MDE through using models. Atkinson & Kühne have distinguished between two types of metamodeling, namely *ontological* and *linguistic* (cf. Section 3), and Favre has used a megamodel to represent the concepts of modeling and metamodeling in MDE. Our aim in this paper is to compare these two research works and bring clarification inputs, in the form of metamodeling concepts, into both of them. Relying on the generality of Favre’s megamodel to cover the concepts of both linguistic and ontological metamodeling, we apply his megamodel, currently only applied to linguistic metamodels, to ontological metamodels and discuss the raised problems. We try to solve these problems by expanding the megamodel, so that it can support the modeling of ontological metamodels as well. Since MDE theory should be able to cover any modeling language, in this paper we are exercising Favre’s megamodel on the example of the Web Ontology Language (OWL) [23]. As both Favre’s megamodel and OWL are based on *set theory*, applying the megamodel to OWL can help with refining and increasing the generality of the megamodel and thus the MDE theory. The composition of the worlds represented by Semantic Web ontology languages and MDE is followed as a goal by the Object Management Group (OMG) through defining the Ontology Definition Metamodel [4, 8, 13, 20]. Our work can also be regarded as another step towards reconciliation of these two worlds.

The rest of the paper has been organized as follows. In Section 2, we review the notions of megamodel and also describe ontologies and models and how they are inter-related. Section 3 further describes the definition of Favre’s megamodel which is used to provide linguistic metamodeling and ontological metamodeling for OWL as a modeling language in MDE. Section 4 discusses the problems of the current megamodel with modeling the concepts of OWL and gives suggestions on how it can be improved. Finally, Section 5 will be a conclusion to this work followed by planning the future research in this area.

## 2 Megamodels and Models in Model Driven Engineering

Favre has introduced the notion of *megamodel* as a way to formally define the theory of MDE [10]. The term megamodel is selected to avoid the confusion with the basic

meanings of the terms *model* and *metamodel*. Defining the formal theory in the form of a model representing the basic MDE concepts and their relations helps us infer new facts about MDE that are not explicitly represented in the megamodel. From this perspective the megamodel can also be regarded as an ontology of MDE.

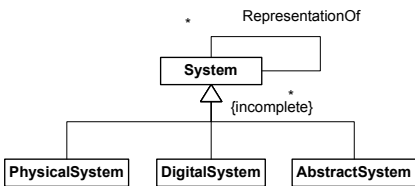


Fig. 1. The megamodel: Classification of systems

Fig. 1 shows an excerpt of Favre’s megamodel [11] as a UML class diagram. The diagram defines the most abstract concept of discourse in MDE – *System*. The (incomplete) classification of systems shown in Fig. 1 distinguishes between physical systems, digital systems, and abstract systems. *PhysicalSystem* represents things from the reality such as ‘a travel agency’. *AbstractSystem* is an abstraction in human minds that can be processed by human brains, for example, concepts and their relations from the biological domain. Finally, *DigitalSystem* is a digital representation that can be processed by computers, such as an XML document with an OWL representation of biological classes and their properties.

In constructing the megamodel, the following definition of models is used: A *model is a set of statements about some system under study* [22]. This definition introduces the notions of a model, the system under study, and their relations. The fact that being a model or a system under study (aka subject [15, 16]) is a relative notion, not an intrinsic property of an artifact, is represented by the *non-transitive* relation *RepresentationOf* (or simply  $\mu$ ) between different systems. This means that, one system can be a model of another system, and this can be represented by *RepresentationOf*. For example, an ontology of the biological domain (abstract system) is a representation of the conceptualization used by a biologist. A file in an OWL XML format (digital system) is a representation of the ontology of the biological domain. The same ontology can also be represented in an XML format of UML (i.e., UML XML Metadata Interchange, XMI).

## 2.1 On Meaning of Models and Ontologies

Since in this paper we are using the OWL language to experiment with Favre’s megamodel, in this subsection, we discuss the relations between ontologies and models in order to motivate why the MDE theory should be applicable to the ontology languages. We have already defined a model as a set of statements about some system under study [22]. Looking strictly from the software development perspective, models are used to *specify (or prescribe)* a software system being developed. Having models formally defined, we can use them to check the validity of developed systems with respect to these models. Furthermore, we can check the consistency of well-defined models. In addition to specifying systems, models can also be used to *describe* systems under study. This is the role of conceptual models that model concepts and their relations within the system under study. In a megamodel, the role of models is defined relative to the roles that reflect relations between systems they represent. A model can be an abstract system that is a *RepresentationOf* another system (that can be physical, abstract, or digital, see Fig. 1), expressed in some modeling language. As one system is a *RepresentationOf* another system, the meaning of a model is always relative. For example, a UML model is an abstract system that is used to describe a specific domain (either physical or abstract system) or to specify a software system (a digital system).

On the other hand, we have ontologies based on the knowledge representation languages. Initially, ontology was defined as a formal specification of a conceptualization. More recent definition states that an ontology is a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic [14]. This definition implies the

*descriptive* nature of ontologies, which is equivalent to the descriptive nature of models. Thus, the *RepresentationOf* relation from the megamodel can be applied to ontologies as well. However, nothing can prevent us from building specification models (e.g., of a software system) using ontologies, as nothing can prevent us from representing ontologies using UML constructs and later transform that representation into ontology languages [2]. One of the main advantages of ontologies over UML is that ontologies have formally defined semantics which makes them better suited for checking the validity of the systems they specify. This also proves the usefulness of the *RepresentationOf* relation from the megamodel for ontologies when they are used as specification models.

Based on the above discussion, we can conclude that although UML and OWL originate from different domains they have a lot in common. Furthermore, we should not neglect the fact that the design of UML and object-oriented programming languages is generally based on frame-based systems. In addition, object-oriented technology has had an important impact on ontology development methodologies and the construction of ontology languages. Some types of description logics adopted the basic object-oriented concepts, such as instantiation and inheritance<sup>1</sup>. This resulted in a proposal for unification of software modeling and knowledge representation technologies [2]. Therefore, in the rest of the paper, we use OWL to exemplify the metamodeling phenomena that are defined by using the megamodeling approach.

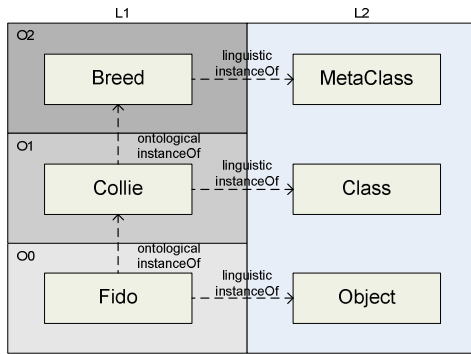
### 3 Metamodeling

We start our discussion on metamodeling with the definition of metamodels given in the OMG's MOF specification [19]: *A metamodel is a model that defines the language for expressing a model.* This definition emphasizes the linguistic nature of metamodeling where metamodeling is used for defining modeling languages. Seidewitz [22] gives the most commonly used definition of metamodels in MDE. *A metamodel is a specification model for a class of systems under study where each system under study in the class is itself a valid model expressed in a certain modeling language.* In fact, a metamodel is a specification model, which leads us to the conclusion that metamodels should be used to validate models represented in a specific language. That is, a metamodel makes statements about what can be expressed in the valid models of a certain modeling language. Generally speaking, a metamodel is any language specification written in English, such as the W3C's OWL language specification or the OMG's UML specification. However, in order to automatically check the validity of models w.r.t. their metamodels, we need to formalize the definition of the metamodel. In the scope of the MDA's pile of standards, MOF is defined as a metamodeling language for defining other languages (see [17] for further details about the MDA). In common understanding of the MDA, metamodels and models are connected by the *instanceOf* relation meaning that a metamodel element (e.g. the Class metaclass from the UML metamodel) is instantiated at the model level (e.g. a UML class Collie).

---

<sup>1</sup> The discussion about different design practices, such as the use of inheritance in software designs and knowledge representation is out of scope of this paper – see [3] for details.

Earlier research in MDE recognized two different types of instantiation relations in various metamodeling architectures: i) instantiation of elements residing at different modeling layers; and ii) instantiation of elements residing at the same modeling layer [5]. This is further explored in [1] where the example shown in Fig. 2 is used. We can say that *linguistic metamodeling* is equivalent to the definitions cited above [22] and [19]. In fact, linguistic metamodeling is a way for defining modeling languages and their primitives (e.g., Object, Class, MetaClass) on the layer of a metamodel (L2 in Fig. 2 or M2 in terms of the MDA). These language concepts are *linguistically* instantiated on the layer of models (L1 in Fig. 2 or M1 in terms of the MDA).



**Fig. 2.** Two types of metamodeling [1]: ontological and linguistic

*Ontological metamodeling* is concerned with domain definition and uses ontological instantiation. For example, a UML object (Fido) is an ontological instance of a UML class (Collie). It is important to say that a UML class and a UML object are on different ontological layers (O0 and O1, respectively), but still they are on the same linguistic metamodeling layer (L1 or M1 in terms of the MDA). We can go further and provide more abstract domain types or domain metaclasses (Breed) on higher ontological layers (O2) without the need to leave the model layer L1.

This distinction of metamodeling types had two important implications on the MDA:

1. The existence of the M0 layer in the classical four-layer MDA architecture has been discarded, as there is no need to separate the M0 and M1 layers. In fact, M0 layer comprising UML objects which are instances of UML classes is now a part of the M1 layer, but UML objects and UML classes are still on different ontological layers (O0 and O1).
2. The UML2 specification enables designing domain concepts whose instances can be both classes and objects.

Looking at the OWL language, we can identify both types of metamodeling. The OWL specification can be regarded as a model of the OWL modeling language, thus it is a linguistic metamodel. To allow for the automatic checking of ontology validity, this linguistic metamodel of OWL has to be specified by using a metamodeling language. Indeed, this is the role of the RDF schema of the OWL language that is a part of the official OWL specification. This schema can be regarded as a linguistic metamodel of the OWL language. To bring OWL ontologies to the MDA, a linguistic metamodel of OWL needs to be represented by an MDA language (i.e. MOF). This, at a more general level, is the goal of the ODM initiative [20], namely, to use MOF as a metamodeling language to develop a linguistic metamodel for representing ontological languages.

In terms of ontological metamodeling, OWL Full allows an instance of a class to be another class, while Boris Motik recently described one OWL Full semantic that is decidable [18].

Taking into account the above examples, we can identify an important issue that should be answered: “How can we formally specify the difference between the ontological and linguistic metamodeling?” In the remainder of this section, we will try to answer this question by using the megamodel [12].

### 3.1 The Megamodel: Metamodeling Relations

Having in mind the purpose of metamodels as means for defining modeling languages, Favre suggests analyzing metamodels in terms of the language theory that is based on sets [12]. The language theory allows us to formally represent relations between models and metamodels, so that one can reason over modeling languages, and thus deduce facts about modeling languages and their characteristics that are not explicitly asserted. In a very broad definition borrowed from the language theory where a *language is a set of systems (or a set of sentences)*, it is emphasized that a

language itself is an abstract system without any materialization. Additionally, we need a way to verify the validity of statements in modeling languages. Models seem as one way to do so due to their specification nature.

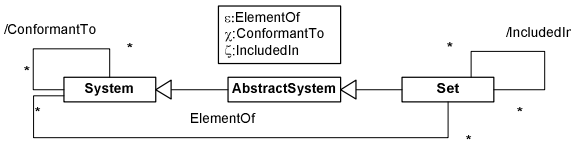


Fig. 3. Elements of the megamodel to define metamodeling

Based on these facts, the megamodel is extended with new elements in order to explain the meaning of metamodeling (Fig. 3). Besides adding the concept *Set* into the megamodel, the relation *ElementOf* ( $\epsilon$ ) is added as a means to assert that a system belongs to a set. In terms of OWL, this relation allows us to state that an individual (i.e. an Abstract System) is an *ElementOf* a class (i.e., a Set). This relation can also relate two sets stating that a set is *ElementOf* (or subset of) another set. However, *ElementOf* is not a transitive relation. This implies that the elements of a set’s subset are not elements of its superset. It is obvious that this relation can not be used to model inheritance. To enable inheritance, the megamodel defines the *IncludedIn* ( $\zeta$ ) relation which is transitive. The transitivity of this relation means that the elements of a set’s subset are also elements of its superset, which is the common meaning of this relation in UML (generalization) and OWL (subClassOf). However, verifying the validity of statements in modeling languages can not be done by using only sets and their relations. For this purpose, the megamodel defines the relation *ConformantTo* ( $\chi$ ): a non-transitive relation between systems in the megamodel. In the rest of this section, we use these relations to define two types of metamodeling.

### 3.2 Linguistics Metamodeling

Let us consider now how we can express relations between some constructs of the OWL language in terms of the extended megamodel (see Fig. 4). As an illustration, we use the concepts from Fig. 2. In OWL, we can use an individual to represent a

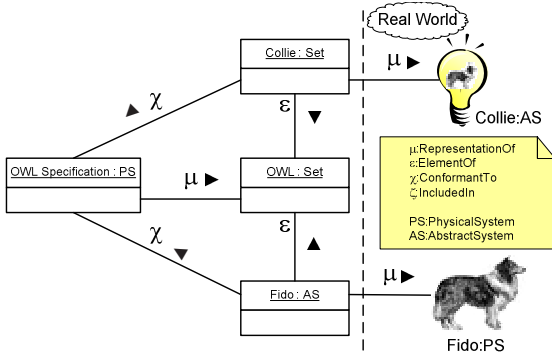


Fig. 4. Relations between models, metamodels, and modeling languages

physical system. In Fig. 4, an OWL individual is an abstract system (Fido:AS) that is a representation of ( $\mu$ ) a physical system (our pet collie Fido:PS). Additionally, we can use an OWL class, e.g. Collie, to generalize the characteristics of all collies. According to the definitions of the megamodel the class Collie should be defined as a set in order to be a representation of the set of all individuals. Thus, the Collie:Set class is

a representation of (or a model of) the abstract system Collie:AS. Note that we leave the analysis of the relation between the class Collie:Set and the individual Fido:AS for the next subsection. According to the definition borrowed from the language theory that a *language is a set of systems*, we infer that both the individual Fido:AS and the class Collie:Set are elements of ( $\epsilon$ ) the OWL language (i.e. OWL:Set). Furthermore, we can say that a *modeling language* (in this case OWL:Set) *is a set of models*. Since a set is an abstract system, we need to represent ( $\mu$ ) a modeling language by a physical system in order to develop tools for checking the validity of expressions in that language. In fact, such a representation of ( $\mu$ ) a modeling language (OWL:Set) in the form of a physical system (OWL Specification:PS) is actually *a model of a modeling language*, i.e. a *metamodel*. This is why in this case we can say that the official W3C's OWL specification can play the role of a metamodel. In the same way, the ODM specification is a metamodel of OWL but specified in MOF. Finally, we can conclude that a metamodel is *a model of a set of models* [12].

There is one important relation between elements of (e.g. Collie:Set and Fido:AS) a modeling language (e.g. OWL:Set) and the metamodel of the modeling language (e.g. OWL Specification:PS). This relation is *ConformantTo* ( $\chi$ ), which means that all elements (classes and individuals) of a modeling language have to conform to ( $\chi$ ) the metamodel of the modeling language. It is very important to note that the relation between models (Collie:Set and Fido:AS) and their metamodel *is not ElementOf*, hence the metamodel does not contain them. Another observation is that the relation *ConformantTo* is equivalent to the *linguistic instanceOf* relation from Fig. 2 and this type of metamodeling is *linguistic metamodeling*. In terms of the OWL language this

relation is represented by the *rdf:type* property relating the OWL language construct *owl:class* with a concrete ontology class, such as the class Collie:Set. We refer readers to [12] where the notion of linguistic metamodeling is defined for the UML language in a similar way.

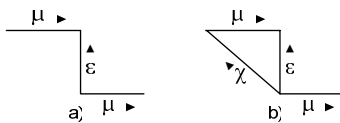


Fig. 5. The  $\mu\epsilon\mu$  (a) and conformant (b) patterns

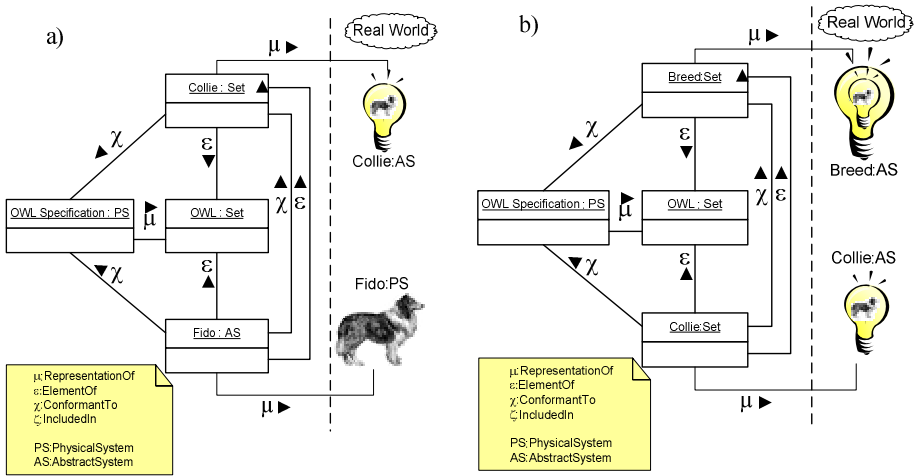
From the above analysis, one can infer an interesting pattern for defining linguistic metamodeling. This pattern is shown in Fig. 5a, and Favre [12] named it the  $\mu\epsilon\mu$  pattern or a pattern representing *a model of a set of models*. Furthermore, we used the conformance pattern (Fig. 5b) to define linguistic metamodeling. The number of times we apply this pattern (i.e. the number of meta-steps) determines whether we are dealing with a model, a metamodel, a metametamodel, etc. For example, repeating this pattern two times leads us to the notion of a meta-metamodel. This is how MOF is defined in the MDA architecture: MOF is a model of a metamodeling language, i.e. a language for defining modeling languages. Looking back at the notion of metamodeling used in [18] to assert that *Breed* from Fig. 2 is a metamodel (i.e. a metaclass) of the *Collie* class, it is obvious that in this case the conformance pattern (Fig. 5b) is not applicable. Although the *ConformantTo* relation exists in this case as well, the *ElementOf* relation between *Breed* (metamodel) and *Collie* (model) is not present, although it should be defined according to the definition of the OWL class from the OWL specification. The existence of the *ElementOf* relation would imply that the metamodel (*Breed*) contains specific elements of a modeling language (in this case the class *Collie*), which is not the case in linguistic metamodeling. Instead, it is the case in ontological metamodeling as explained in the next subsection<sup>2</sup>.

### 3.3 Ontological Metamodeling

To explain ontological metamodeling, let us first clarify the relations between the elements of the modeling language. Fig. 6a shows an extended UML static structure diagram from Fig. 4 with the relations between the set *Collie:Set* and the abstract system *Fido:AS* shown. The notion of a set for *Collie* has been chosen correspondent to the definitions of Favre's megamodel in which ontological concepts have been considered as sets [12]. The set *Collie:Set* in modeling languages (such as OWL and UML) can be best modeled by using the term *class*. Such a set is typically composed of elements that are just abstract systems (or facts in terms of the OWL specification). This explains why we have the *ElementOf* relation between *Collie:Set* and *Fido:AS* with the meaning that *Fido:AS* is an *ElementOf* *Collie:Set*. However, the set of abstract systems groups them also based on their common characteristics. Consequently, according to Favre's megamodel, abstract systems have to be *ConformantTo* their sets. This implies having a *ConformantTo* relationship from the *Fido:AS* to *Collie:Set*, as it has been shown in Fig. 6a.

In Fig. 6b, we analyze the relation between the meta-class *Breed:Set* and the class *Collie:Set* from Fig. 2 using the megamodel. The meaning of the set *Collie:Set* is the same as above (Fig. 6a): *RepresentationOf* ( $\mu$ ) the abstract system *Collie:AS*; *ConformantTo* ( $\chi$ ) the OWL specification; *ElementOf* ( $\epsilon$ ) the OWL:Set (i.e. OWL language); and *Fido* is *ElementOf* ( $\epsilon$ ) and *ConformantTo* ( $\chi$ ) the set *Collie*. Note that *Breed* also has been chosen as a set by following the same rationality that was expressed for *Collie*. This way we can keep up with the notions of megamodel that has been suggested by Favre.

<sup>2</sup> This also reflects the strict separation (a basis for strict metamodeling architectures) of a set of elements or symbols belonging to models from a set of elements belonging to the metamodel, i.e., having strict separation between meta-layers in metamodeling architectures [22]. This idea has also inspired [19] to develop their metamodeling architecture for the Semantic Web.



**Fig. 6.** Relations between abstract systems and sets of abstract systems represented in a modeling language (a); Ontological metamodel in the megamodel: Relations between a set and sets of models (b)

Let us look at Breed. Breed:AS is the real meaning of Breed in human’s mind, while the set Breed:Set (according to Favre’s megamodel) is considered as a *RepresentationOf* ( $\mu$ ) the abstract system Breed:AS, i.e Breed:Set is a model of the abstract system Breed:AS. Furthermore, the Breed:Set contains the Collie:Set meaning that the Collie:Set is an *ElementOf* ( $\epsilon$ ) the Breed:Set. In terms of OWL, the Breed:Set is a class that contains another class, i.e. the Collie:Set. Since the set Breed:Set defines characteristics of its elements (instantiation relation from Fig. 2), the Collie:Set should be also *ConformantTo* ( $\chi$ ) the Breed:Set. Due to the non-transitive definition of the *ElementOf* relation, the elements of the Collie:Set (i.e. Fido:AS) are not elements of the Breed:Set. In this case, we can say that Breed:Set is a superset of the Collie:Set whose element is the abstract system Fido:AS. In the OWL language, it can be said that the class Collie is of type Breed, which means that the Breed class is a meta-class for Collie. The individual Fido is of type Collie, but Fido is not of type Breed.

Here we have a case of metamodeling, which is not linguistic metamodeling as defined in the previous section; it is ontological metamodeling. In this case, we can identify another pattern that is based on the  $\chi\epsilon$  relation between two sets, and both sets are models of abstract systems. However, the *ConformantTo* relationship between Fido as an abstract system and Collie as a set, and also Collie as a set and Breed as another set, opens some points for discussion. Although it is reasonable to consider an abstract system conformant to another abstract system, it is not reasonable to consider an abstract system or a set conformant to another set. Looking at Collie as a set of all collies in the world, there is nothing in common between our Fido and a community of Collies, but the commonality comes from the properties that all these dogs share as Collies. Similarly the representation and definition of the concept Breed makes Collie a Breed, but the Breed set is just to identify the individuals that have been instantiated

from Breed. In Section 4, we argue how the megamodel can be modified in order to have a more comprehensive representation of the ontological metamodeling.

## 4 Expanding the Megamodel for Ontological Metamodeling

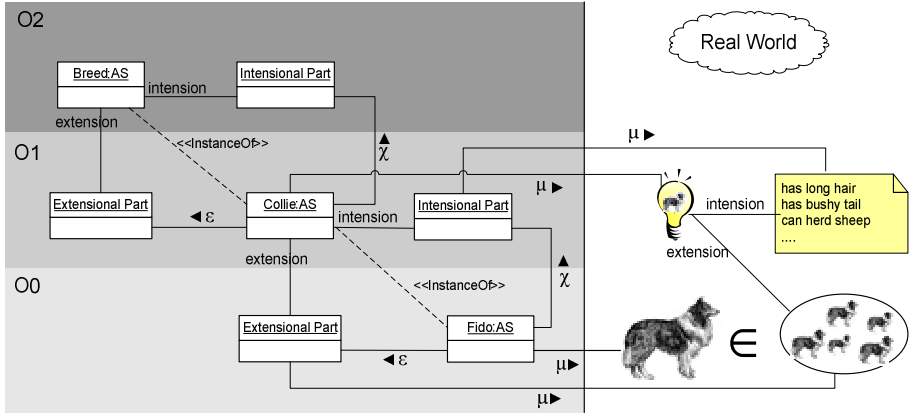
So far, we have discussed ontological and linguistic metamodeling through the use of megamodeling notions. According to the discussions in the previous section, Favre's megamodel [12] provides a suitable definition for linguistic metamodeling in MDE, however, once we apply it to ontological metamodeling some issues arise. In Section 3.3, we have shown that, as a result of expressing ontological metamodels by using Favre's megamodel, an abstract system or a set is considered conformant to another set, which is not reasonable. For example, Fido, from our previous example, is not *ConformantTo*( $\chi$ ) the set of Collies, but it is *ConformantTo*( $\chi$ ) the characteristics that differentiate between Collies and other breeds of dogs. At the same time, the same Fido is an *ElementOf* ( $\epsilon$ ) the set of Collies. Knowing that each ontological concept can be modeled with a class, the question is how we can precisely define the sets and the characteristics of a class, and its correspondent concept, in the megamodel.

Let us start with the definition of a class in the OWL specification. According to W3C's specification for OWL [23], "Classes provide an abstraction mechanism for grouping resources with similar characteristics. Every OWL class is associated with a set of individuals, called the *class extension*. The individuals in the class extension are called the *instances* of the class. A class has an *intensional* meaning (the underlying concept) which is related but not equal to its class extension. Thus, two classes may have the same class extension, but still be different classes."

The main problem with the definition of Favre's megamodel is that there is no distinction between the *intension* and the *extension* of a class. While our Fido is *ConformantTo*( $\chi$ ) the characteristics of the class Collie, i.e. the intension of the class Collie, it is an *ElementOf* ( $\epsilon$ ) the extension of this class. In fact, in ontological metamodeling the instantiation relationship (*instanceOf*) happens between two concepts when one concept is *ConformantTo*( $\chi$ ) the intension and is an *ElementOf* ( $\epsilon$ ) the extension of the other concept. Consequently, the relations between Breed, Collie, and Fido of Fig. 2 should be modeled similar to Fig. 7, which is different from what we obtained in Fig. 6. Note that the  $\llinstanceOf\gg$  relationship of Fig. 7 is not part of the megamodel and we are using the link between the super concept and its instance only for the sake of clarification.

We claim that a set is equivalent to the extensional part of a concept, and considering the concept equal to a set means dropping the intensional part of that concept. Especially, it has been also pointed out in the OWL definition of a class that the extensional part might be shared among a couple of classes with each class representing different intensions. Referring to a class as a set makes the desired intensional meaning ambiguous. Also, according to the definition of a class in OWL, the extension of a class is a set of its instances or individuals. It means that this extension is in the same ontological level as one single individual of the class (see Fig. 7). Additionally, for each of the intensional and extensional properties of a concept, there is a description in the form of an abstract system in the real world

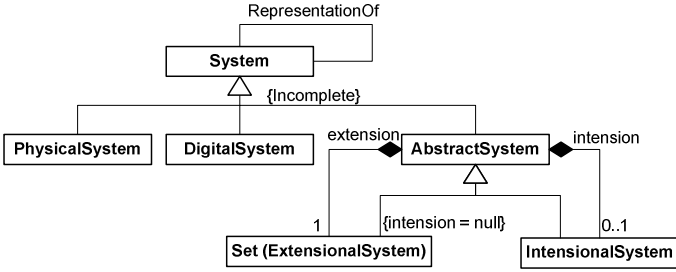
which can be connected to the intension or extension of the system with a *RepresentationOf* ( $\mu$ ) relation as shown in Fig. 7. In OWL, the intensional part of a concept is defined in the form of class properties, also each object instantiated from a class is an element of the extensional part of the class. Consequently, a class can be considered an abstract system composed of intensional and extensional parts which are also abstract systems.



**Fig. 7.** The relations between the instantiated elements and the intensional and extensional properties of their super classes

Based on the discussions that we have made to this point, the megamodel introduced by Favre should be further refined to cover the intensional and extensional meanings of a concept. We argued that intensional and extensional properties of a class are also abstract systems. We believe, the term *Set*, coined in Favre’s megamodel as an abstract system, is nothing but the extensional meaning of the ontological concept, and thus this extensional meaning of the concept is an abstract system as well. Moreover, the intensional meaning of a concept is fully in line with the definition of the abstract system in Favre’s megamodel (cf. Section 2) and can be considered an abstract system. This leads us to improving Favre’s megamodel by extending the abstract system according to Fig. 8. It is worth mentioning that Favre in his megamodel represented the identified types of a system as *incomplete*, which has left the room for further expansion and improvement. Our expansion of Favre’s system is yet an incomplete representation of the system as well.

As Fig. 8 shows, an abstract system composes a set (as the extensional part) and an intensional part which both are also abstract systems. However, a set is an abstract system for which there is no intensional part, while the intensional system is an abstract system that can have both the extensional and intensional parts. This definition of an intensional system makes it possible to have a set of intensions for an abstract system, which is generally the case when defining the ontological concepts.



**Fig. 8.** The extended megamodel to cover intensional and extensional systems

Given the definitions in Fig. 7 and Fig. 8, our representation of ontological metamodel becomes so close to what Kühne has provided in [15, 16]. Kühne considers element  $e$  an ontological instance of a type (or super class)  $T$  where:

$$\mu(e) \in \text{extension}(\mu(T)) \text{ and}$$

$$\text{intension}(\mu(T))(\mu(e))$$

The second formula is equal to considering the *RepresentationOf* element  $e$  *ConformantTo* the intension of the *RepresentationOf* type  $T$ . The main difference, however, is that in Kühne’s representation of ontological metamodel there is no intensional or extensional system in the ontological layers (e.g.  $O_0$  or  $O_1$ ) and these two systems are only considered as the meanings in the real world (referred to in the form of *RepresentationOf*). Nonetheless, we believe that extensional and intensional meanings also exist as systems within the modeling space [9]. The witness to this claim is the representation of the properties for a class which can be considered as the intension of the class. Furthermore, referring to the definition of a class in OWL, provided in the beginning of this section, one can recognize that the set of individuals for an OWL class are not judged based on their presence in the real world, but these individuals are the objects instantiated from the OWL class in the knowledge base. We can even consider the representation of a class in OWL as a *DigitalSystem* which is a model of the *AbstractSystem* intended in the program. However, as the notion of model is relative; this chain of finding a model for another model will never stop unless we decide to stop it at a certain point. As a result, we consider the definition of a class in OWL as an *AbstractSystem*.

Now that we have made a clear understanding of the ontological metamodeling by refining the definitions of the megamodel, we can provide a definition for the ontological metamodel, based on our notions of megamodel. We define *an ontological metamodel as a set (or extension) of models of abstract systems classified based on some instentions, and as an element of a modeling language*. At the same time, *the ontological metamodel is a model of an abstract system*. Given the definition above, the next section compares the implications of ontological and linguistic metamodeling.

### 4.1 Ontological and Linguistic Metamodeling: Implications

There are several important consequences of the definitions above:

1. An ontological metamodel is a model of an abstract system, which means that ontological metamodel can only model abstract systems (not physical and digital systems). This further means that an ontological metamodel can only be a representation of systems that are sets of either other sets or abstract systems. Consequently, an ontological metamodel in the OWL Full language can not contain individuals, only classes.
2. Being an ontological metamodel means being only a model of an abstract system, but not a model of models of systems like it is the case with linguistic metamodel. Consequently, an ontological metamodel *is not* a model of a modeling language, and thus it is not supposed to define a set of valid constructs of a modeling language.
3. The definition of an ontological metamodel is surprising at first, since an ontological metamodel is not just a model, but it is also a set. Having in mind that the main purpose of ontological metamodel is to define libraries of high level domain types [1] it seems quite natural that an ontological metamodel is indeed a set of (predefined domain specific) models of abstract systems. That is to say, a set of classes, meta-classes, meta-meta-classes and so forth in terms of OWL or UML.
4. An ontological metamodel should be conformant to the linguistic metamodel of the modeling language like any other element of the modeling language. This is again compliant with the notion of ontological metamodeling from Fig. 2 where the libraries of high level domain types are linguistic instances of the linguistic metamodel.

There are also some inferred facts about relations between ontological and linguistic metamodeling.

5. The first concern with regards to the linguistic metamodel is whether the change to the megamodel affects the definition of linguistic metamodeling or not. Fig. 9a shows the exact pattern that we introduced in Fig. 5b. Fig. 9b is the rotation of that same pattern, and finally, Fig. 9c is the pattern that we obtained for ontological metamodeling based on intensional and extensional expansion of abstract system, which is really close to Fig. 9b. It can be said that in ontological metamodeling, because we are mostly dealing with the *AbstractSystems*, the notions of intensional and extensional systems become more important, so that they need to be clearly defined. However, in linguistic metamodeling, because the metamodel is usually a *PhysicalSystem* or *DigitalSystem*, it can play as both intensional and extensional systems and thus these two systems can be omitted (in Fig. 8 we consider intensional and extensional systems only for *AbstractSystems*). The possibility of having an *AbstractSystem* as a linguistic metamodel is open for future research.

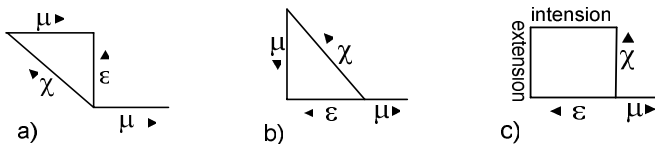


Fig. 9. The effect of the change to the megamodel on the pattern of Fig. 5

6. Linguistic metamodeling is used to specify a set of valid statements of a modeling language, and thus linguistic metamodeling (a linguistic metamodel of the OWL or UML languages) specifies whether ontological metamodeling is supported by a modeling language. Defining a modeling language by a linguistic metamodel does not necessarily mean that the modeling language supports ontological metamodeling.
7. Metamodeling languages, such as MOF, may have also support for ontological metamodeling according to the meta-step definition for linguistic metamodeling. This means that we can define meta-classes that are ontological instances of other meta-classes on the same linguistic layer. However, the pragmatics for the presence of such features in metamodeling languages is out of scope of this paper (see discussion on minimal reflective metamodel [22]).

## 5 Conclusion

Favre by representing megamodels and Kühne by identifying linguistic and ontological metamodeling have had important contributions to clarifying the theory of MDE. In this paper, we investigated the generality and comprehensiveness of Favre's megamodel, once it is applied to the ontological metamodels. We have shown that for this megamodel to be able to fully capture the intended meaning of the ontological metamodeling, the concept of *AbstractSystems* should be further expanded to entail *intensional* and *extensional* systems which are also *AbstractSystems*. We also argued that the extensional system is what we know as a *Set*, while the intensional system is the characteristic or the set of characteristics (as an intensional system can entail a set itself as a result of being an abstract system) that classify the objects in the set. As a consequence of applying the change to the megamodel, we have shown how the ontological classification becomes more meaningful by making each concept *ConformantTo* the intensional system of its super class and an *ElementOf* the extensional system of the super class. We have also shown that this change does not really interfere with Favre's representation of linguistic metamodeling using megamodels and the whole systems remains consistent. As a future plan for this research, we decide to consider the relations between different intensional systems that have the share the same extensional system (e.g. a group of planets clustered in one single set based on both their biological and botanical characteristics) and see how the intensional and extensional relations affect the definition of the megamodel.

## References

1. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. *IEEE Software* 20(5), 36–41 (2003)
2. Atkinson, C.: Unifying MDA and Knowledge Representation Technologies. In: Proceedings of the International Workshop on the Model-Driven Semantic Web (At the 8th International Conference on Enterprise Distributed Object Computing), Monterey, CA (2004)
3. Atkinson, C., Kühne, T.: Rearchitecting the UML infrastructure. *ACM Transactions on Modeling and Computer Simulation* 12(4), 290–321 (2002)

4. Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J.E., Letkowski, J., Emery, P.: Extending the Unified Modeling Language for ontology development. *Software and Systems Modeling* 1(2), 142–156 (2002)
5. Bézivin, J., Lemesle, R.: Ontology-Based Layered Semantics for Precise OA&D Modeling. In: *Proc. of the WSh on Object-Oriented Tech.*, Jyväskylä, Finland, pp. 151–154 (1998)
6. Bézivin, J.: On the unification power of models. *Software and System Modeling* 4(2), 171–188 (2005)
7. Bézivin, J., Grebe, O.: Towards a Precise Definition of the OMG/MDA Framework. In: *Proceedings of ASE'01* (November 2001)
8. Bodoff, D., Ben-Menachem, M., Hung, P.C.K.: Web Metadata Standards: Observations and Prescriptions. *IEEE Software* 22(1), 78–85 (2005)
9. Djurić, D., Gašević, D., Devedžić, V.: The Tao of Modeling Spaces. *Journal of Object Technology* 5(8), 125–147 (2006)
10. Favre, J.M.: Towards a Basic Theory to Model Driven Engineering. In: *WISME 2004. Proceedings of the UML2004 International Workshop on Software Model Engineering*, Lisbon, Portugal (2004)
11. Favre, J.M.: Foundations of Model (Driven) (Reverse) Engineering: Models - Episode I, Stories of the Fidus Papyrus and of the Solarus, Dagstuhl Seminar 04101 on Language Engineering for Model-Driven Software Development, Dagstuhl, Germany (2004)
12. Favre, J.M.: Foundations of the Meta-pyramids: Languages and Metamodels - Episode II, Story of Thotus the Baboon, Dagstuhl Seminar 04101 on Language Engineering for Model-Driven Software Development, Dagstuhl, Germany (2004)
13. Gašević, D., Djurić, D., Devedžić, V.: *Model Driven Architecture and Ontology Development*. Springer, Berlin (2006)
14. Hendler, J.: Agents and the Semantic Web. *IEEE Intelligent Systems* 16(2), 30–37 (2001)
15. Kühne, T.: Clarifying matters of (meta-) modeling: an author's reply. *Software and Systems Modeling* 5(4), 395–401 (2006)
16. Kühne, T.: Matters of (Meta)- Modeling. *Software and Systems Modeling* 5(4), 369–385 (2005)
17. Miller, J., Mukerji, J.: MDA Guide Version 1.0., *OMG Document: omg/2003-05-01* (2003), Available: [http://www.omg.org/mda/mda\\_files/MDA\\_Guide\\_Version1-0.pdf](http://www.omg.org/mda/mda_files/MDA_Guide_Version1-0.pdf)
18. Motik, B.: On the Properties of Metamodeling in OWL. In: *Proceedings of the 4th International Semantic Web Conference*, Galway, Ireland, pp. 548–562 (2005)
19. *OMG MOF OMG Meta Object Facility Specification v1.4*, *OMG Document formal/02-04-03* (2002), Available: <http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf>
20. *OMG ODM: Ontology Definition Metamodel*, *OMG Document ad/05-08-01* (2005), Available: <http://www.omg.org/cgi-bin/apps/doc?ad/05-08-01.pdf>
21. Pan, J.Z., Horrocks, I.: Metamodeling architecture of Web ontology languages. In: *Proceedings of the 1st Semantic Web Working Symposium*, Stanford Univ., USA, pp. 131–149 (2001)
22. Seidewitz, E.: What Models Mean. *IEEE Software* 20(5), 26–32 (2003)
23. *W3C Specification for OWL: OWL Web Ontology Language Reference: (February 2004)*, Available: <http://www.w3.org/TR/owl-ref/>