Chapter 31
An Ontological Analysis of Metamodeling Languages

Erki Eessaar and Rünno Sgirka

1 Introduction

Metamodeling systems (meta-CASE systems) are used to create new modeling systems (CASE systems), which allow developers to model systems by using one or more modeling languages; to test the models; and possibly to generate program code based on the models. These modeling languages are usually domain-specific languages (Kelly and Tolvanen 2008). Each metamodeling system provides a *metamodeling language* (metalanguage) for specifying modeling languages. There are many different metamodeling systems and hence also metamodeling languages. It raises a question, whether some of the metamodeling languages are better than others and how to find it out. Specification of a formal language should specify semantics, abstract syntax, concrete syntax, and serialization syntax of the language (Greenfield et al. 2004). In this chapter, we concentrate our attention to the evaluation of the abstract syntax of languages.

Livingstone (2008) argues that a programming language should be conceptually simple and hence have characteristics like parsimony, straightforwardness, generality, orthogonality, and uniformity. Siau and Rossi (1998) have proposed a set of evaluation methods of information modeling methods. This kind of methods can be used to analyze different modeling languages – not only information modeling languages. One of the proposed nonempirical methods is *ontological analysis*. Guizzardi (2005) presents the framework for performing the ontological analysis of artificial modeling languages. There exist examples of ontological analysis of modeling languages. These analyses use *foundational ontologies* (also known as upper ontologies or top-level ontologies), which specify domain-independent categories and are theoretically well founded (Guizzardi et al. 2008). For instance, Bunge-Wand-Weber (BWW) ontology (Wand and Weber 1990) has been used to

E. Eessaar (✉) • R. Sgirka
Department of Informatics, Tallinn University of Technology, Tallinn, Estonia
e-mail: eessaar@staff.ttu.ee; runno.sgirka@gmail.com

© Springer Science+Business Media New York 2013

analyze and redesign UML 1.3 (Opdahl and Henderson-Sellers 2002) and Architecture of Integrated Information Systems (ARIS) (Green and Rosemann 1999). More recently, Unified Foundational Ontology (UFO) has been used to analyze and redesign Software Process Ontology, which is a domain ontology (Guizzardi et al. 2008), and UML 2.0 (Guizzardi and Wagner 2010). Guizzardi and Guizzardi (2010) use UFO to design an agent-oriented engineering language for the ARKnowD methodology.

In this chapter, we apply the ontological analysis method (Guizzardi 2005) to the metamodeling languages. The *first goal* of this chapter is to investigate how to perform ontological analysis of metamodeling languages and whether the results of the analysis give language designers sufficient information to improve the quality of metamodeling languages. Therefore, we present the results of a small experiment, during which we analyzed two metamodeling languages by using a foundational ontology. One of the languages is used in a web-based and database-based metamodeling system WebMeta (Eessaar and Sgirka 2010), which we have developed over time. Hence, the *second goal* is to find out whether the metamodeling language of WebMeta needs improvement and, if it does, then propose improvements that are ontologically well founded.

The rest of this chapter is organized as follows: In Sect. 2, we discuss how to perform ontological analysis of metamodeling languages. In Sect. 3, we present the results of an ontological analysis of two metamodeling languages and suggest improvements to one of the languages. Finally, we conclude and point to the further work with the current topic.

2 Ontological Analysis of Metamodeling Languages

It is possible to specify the abstract syntax of a modeling language by using a metamodel (Greenfield et al. 2004). If we use UML class diagrams to represent a metamodel, then modeling constructs are represented as classes and relationships between the modeling constructs are represented as associations or generalizations. In this chapter, we call the metamodel of a metamodeling language a meta-metamodel, based on the MOF metamodeling architecture (Meta Object Facility).

Each metamodeling language is a domain-specific modeling language that is used to specify modeling languages. These modeling languages will be used to create models, which specify static and/or dynamic characteristics of very different subject areas. For instance, Kelly and Tolvanen (2008) present five examples of domain-specific languages, which were implemented by using the same metamodeling language: IP telephony and call processing, insurance products management, microcontroller applications, mobile phone applications, and digital wristwatch applications.

The modeling constructs in a metamodeling language are very generic, because they must facilitate definition of very different languages, which are used in many different domains. Foundational ontologies describe very general concepts like

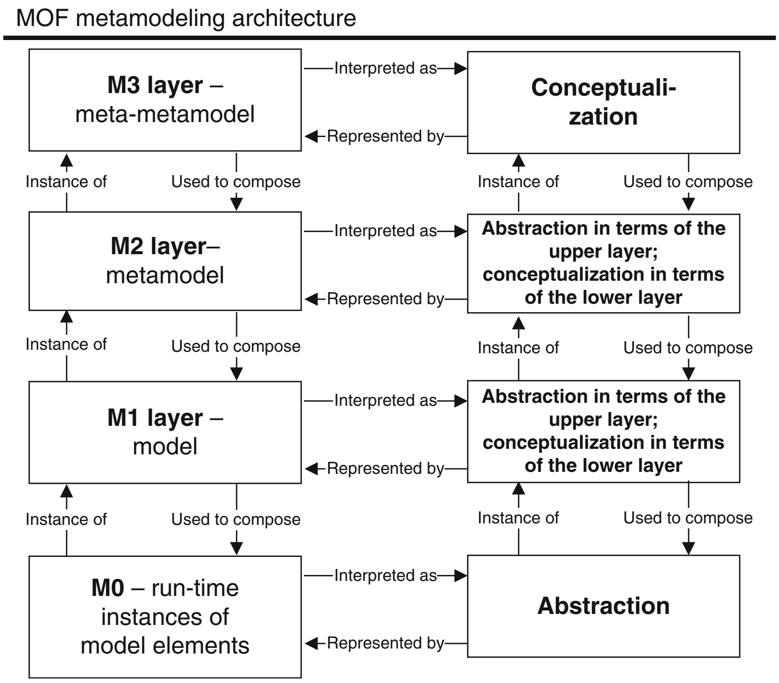
MOF metamodeling architecture
M3 layer – meta-metamodel
Interpreted as
Represented by
Conceptualization
Instance of
Used to compose
M2 layer– metamodel
Abstraction in terms of the upper layer; conceptualization in terms of the lower layer
M1 layer – model
M0 – run-time instances of model elements
Abstraction

Fig. 31.1 Correspondence between four layers of the MOF metamodeling architecture and conceptualization and abstraction

Entity, *Particular*, and *Universal*. Therefore, foundational ontologies are the suitable basis for the ontological analysis of metamodeling languages.

The elements that make up a *conceptualization* of a domain D are used to articulate *abstractions* of certain state of affairs in reality (Guizzardi and Wagner 2010). Each ontology is a formal specification of a domain conceptualization.

Any number of meta-layers, greater than or equal to two, are permitted by the MOF 2.0 metamodeling architecture (Meta Object Facility). For instance, UML infrastructure specification (OMG Unified Modeling Language™ uses a four-layer metamodel hierarchy, where M3 is the highest layer. Figure 31.1 presents the correspondence between four layers of the metamodeling architecture and conceptualizations and abstractions. For instance, a certain conceptualization of the domain of modeling language design can be constructed by considering concepts such as *Monadic Universal* and *Relation*, among others. These concepts are represented by the modeling constructs in a metamodeling language.

By using these concepts, it is possible to articulate a domain abstraction that a certain modeling language allows us to model *Actors*, *Use cases*, and relationships between *Actors* and *Use cases*.

On the other hand, these abstractions also constitute a domain conceptualization of the domain of use-case modeling. These concepts are represented by the modeling constructs in a modeling language (in this case, the language for creating

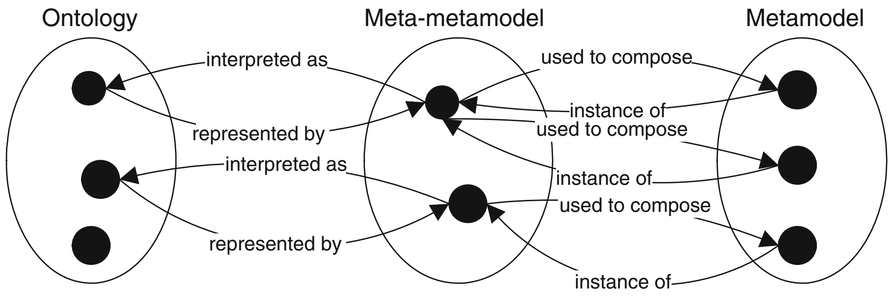

Fig. 31.2 Example of mappings between the elements of an ontology, a meta-metamodel, and a metamodel, which has been created based on the meta-metamodel

use-case models). By using these concepts, it is possible to articulate a domain abstraction that, for example, an actor *Client* is associated with a use-case *Make an order* in a particular model. On the other hand, these abstractions also constitute a domain conceptualization of the domain of a particular information system. These concepts are represented by the model elements in a model. By using these concepts, it is possible to articulate a domain abstraction that, for example, an actor *John Smith* makes an order with number *O110234*.

For each layer of the metamodeling architecture, there are corresponding languages, which are used to create models that correspond to this layer. It is possible to perform the ontological analysis of all these languages. In this chapter, we are interested in the ontological analysis of metamodeling languages, which correspond to the M3 level of the architecture.

Ontological analysis of a metamodeling language L means that one has to compare the metamodel of L with an ontology O to find possible violations of the following desired properties of L: completeness, soundness, lucidity, and laconism (Guizzardi and Wagner 2010). If L has these properties, then the metamodel of language L and ontology O is isomorphic, and it should reduce the problems of using L.

A metamodeling language L is *complete* in terms of a domain D if and only if every concept in the ontology O of that domain is represented in a modeling construct of L (Guizzardi 2005; Guizzardi and Wagner 2010). Each metamodeling language is used to specify zero or more modeling languages (see Fig. 31.2). Hence, in case of evaluating completeness of metamodeling languages, one could *also* investigate whether it is possible to represent every concept in O in the metamodel of at least one language, which is created by using the metamodeling language. For instance, one of the concepts in the UFO ontology is *Role* (Guizzardi and Guizzardi 2010). During the analysis of completeness of a metamodeling language L based on the UFO ontology, one could investigate, whether L allows language designers to create a modeling language L' where one of the modeling constructs represents the concept *Role* in the UFO. We do not conduct this analysis in this chapter.

A metamodeling language L is *sound* in terms of a domain D if and only if every modeling construct in L has an interpretation in terms of a domain concept in the ontology O (Guizzardi 2005; Guizzardi and Wagner 2010). A modeling construct *c* in a modeling language L', which is created by using L, might not have a corresponding concept of O that provides its interpretation. Metamodeling languages should not be too restrictive and should allow language designers to define this kind of modeling constructs in modeling languages. However, it is a violation of the soundness property in case of L', and one has to perform ontological analysis of L' to find the problem.

A metamodeling language L is *lucid* in terms of a domain D if and only if every modeling construct in L represents at most one domain concept in O (Guizzardi 2005; Guizzardi and Wagner 2010).

A metamodeling language L is *laconic* in terms of a domain D if and only if every concept in the ontology O of that domain is represented at most once in the metamodel of L (Guizzardi 2005; Guizzardi and Wagner 2010).

In addition, one has to evaluate, whether the meta-metamodel of L follows all the constraints, which have been specified in the ontology O.

3 An Experiment

In this section, we present the results of an ontological analysis of two metamodeling languages. A problem, which limits the selection of metamodeling languages for the analysis, is that complete specifications of the abstract syntax of some of the languages are not publicly available.

We selected the following languages, which have been developed in-house by our university: a metamodeling language that is used in our web-based and database-based metamodeling system WebMeta (ver. 0.5) (Eessaar and Sgirka 2010) (see Fig. 31.3) and a metamodeling language, which is proposed to use in the context of evolutionary information systems (Roost et al. 2007) (see Fig. 31.4).

We selected Unified Foundational Ontology (UFO) as the foundational ontology, which is used as the basis in the ontological analysis. A reason is that it has been recently used in other ontological analysis as well. Another reason is that the ontology is documented by using diagrams that resemble UML class diagrams, and it simplifies the ontological analysis process.

Rosemann et al. (2004) suggest that one has to set the scope of an ontological analysis of a language L by selecting a subset S of concepts of the ontology O. Only the concepts that belong to S will be used in the analysis. The subset must contain only these concepts of O that are relevant in terms of the language L metamodel. Metamodeling languages are used to specify the structure of modeling languages. Metamodeling languages are not used to specify behavior or social concepts. Hence, we used in the ontological analysis a subset (a compliance set) of UFO, namely, *UFO-A: An Ontology of Endurants*. However, if one wants to analyze whether it is possible to represent every concept in the foundational ontology O in the metamodel of at least one language, which is created by using the metamodeling language, then one has to use O completely and not only a subset of O.

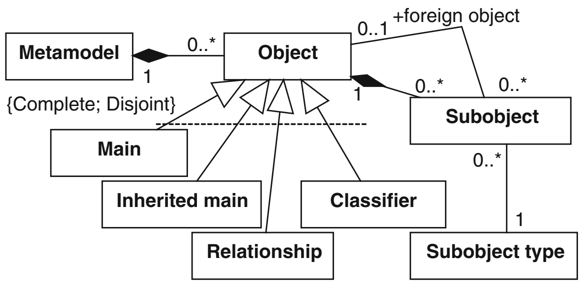

Fig. 31.3 Meta-metamodel of the metamodeling system WebMeta (ver. 0.5)

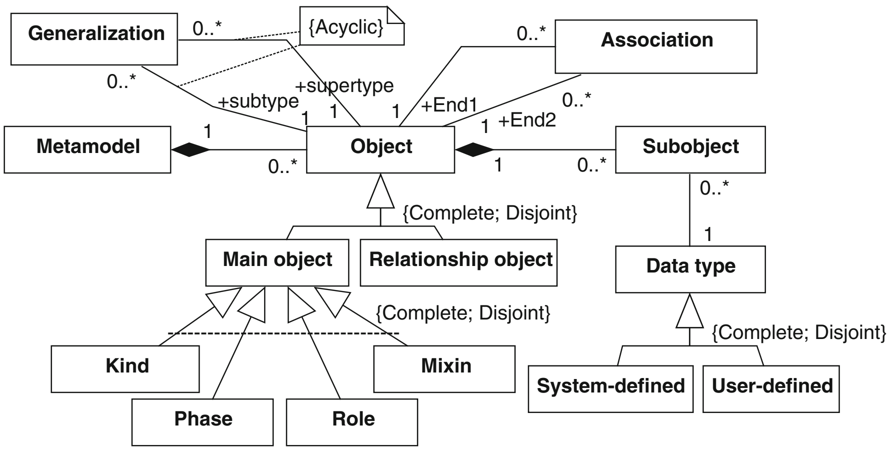

Fig. 31.4 A redesign meta-metamodel of the WebMeta metamodeling system

A problem is that the UFO ontology is still in active development and has not yet completely stabilized. Therefore, the descriptions of UFO elements are somewhat different in different articles. However, UFO-A is more mature and stable compared to other parts of UFO (Guizzardi and Wagner 2010). The current analysis is based on the UFO-A specification, which is presented in (Guizzardi and Wagner 2010). We do not present in this chapter a formal characterization of UFO-A due to the lack of space.

Rosemann et al. (2004) suggest that in order to improve the quality of the ontological analysis of a language L, the metamodel of L and an ontology O, which is used in the ontological analysis, must be represented by using the same language. In case of this analysis, we used the meta-metamodels, which were represented by using UML class diagrams. The ontology (UFO-A) that is used in the analysis is represented in (Guizzardi et al. 2008; Guizzardi and Wagner 2010) by using diagrams, which resemble UML class diagrams.

Table 31.1 Mappings between the modeling constructs of the WebMeta (ver. 0.5) metamodeling language and concepts of the UFO-A ontology

Metamodeling language	Ontology
Classifier object	Quality structure
Inherited main object	Object universal, Basic formal relation
Main object	Object universal
Metamodel	Abstract set
Object	Universal
Relationship object	Material relation
Subobject	Quality universal, Relator universal
Subobject type	Quality structure

3.1 WebMeta Metamodeling Language

The meta-metamodel of the metamodeling language, which is the basis of WebMeta metamodeling system (Eessaar and Sgirka 2010), specifies the following classes: *Metamodel*, *Object*, *Main, Inherited main*, *Classifier*, *Relationship, Subobject*, and *Subobject type* (see Fig. 31.3). Each *Metamodel* specifies the abstract syntax of a modeling language (that belongs to the M2 layer; see Fig. 31.1). *Classifier* is used to specify modeling language constructs that help modelers to characterize some other modeling language constructs. *Relationship* is used to specify modeling language constructs that relate other modeling language constructs. *Main* and *Inherited main* are used to specify modeling language constructs that are not classifiers or relationships. *Inherited main* can be used to specify a modeling language construct based on inheritance from some other *Main* or *Inherited main* modeling language construct. *Subobject* is used to specify the properties of modeling language constructs and associations between modeling language constructs. Examples of *Subobject type* are integer, varchar(100), and boolean.

The metamodeling language of WebMeta *is sound*, because every modeling construct in the metamodeling language has an interpretation in terms of a domain concept in the UFO-A ontology (see Table 31.1). For instance, the ontology element *Abstract set* provides interpretation to the language construct *Metamodel*. Metamodel is a set of model elements, which together specify the abstract syntax of a language.

Each data type is a named, finite set of values (Date 2006). Hence, we agree with Guizzardi and Wagner (2010) that the ontology element *Quality structure*, which is a subclass of *Set* in the UFO-A ontology, is the ontological interpretation of the *Subobject type* construct in the metamodeling language. Each *Classifier object* is conceptually a set of values that are used to characterize model elements at the M1 layer of the MOF metamodeling architecture. For instance, a use-case modeling language could allow modelers to specify the importance of each particular use case. One could define a classifier object *Priority* that belongs to the *Use case* metamodel. The values that belong to the classifier object and can be used to characterize use cases could be “low,” “medium,” and “high.” Hence, we think that the ontological interpretation of the modeling construct *Classifier object* must be the same than the interpretation of the modeling construct *Subobject type*. The modeling construct *Subobject type* represents system-defined (simple) data types. The modeling construct *Classifier object* represents user-defined data types (more precisely, *enumerated types*).

The metamodeling language of WebMeta *is not complete*, because there are concepts in the UFO-A ontology that are not represented by a modeling construct of the metamodeling language.

For instance, there are concepts like *Kind*, *Role,* and *Phase*, as well as *Concrete particular* that are not represented by a modeling construct of the metamodeling language.

The metamodeling language of WebMeta *is not lucid* because there are modeling constructs in the metamodeling language, which represent more than one domain concept in the ontology (see Table 31.1). The modeling construct *Inherited main* represents *Object universal* and *Basic formal relation*. If we use *Inherited main* construct in a metamodel, then it means that there is an inheritance relationship in the metamodel. Inheritance belongs to the category *Basic formal relation* (Guizzardi and Wagner 2010).

The modeling construct *Subobject* represents the ontology concepts *Quality universal* and *Relator universal*. *Subobjects* are used to represent properties of objects. For instance, if we specify use-case modeling language, then the fact that each use case must have a name would be specified by defining the subobject *name* of the object *Use case*. Each *name* is existentially dependent of one single particular – a *Use case*. *Subobjects* are also used to represent relationships between objects. The fact that each use case should be associated with a primary actor could be specified by defining the subobject *primary actor* of the object *Use case*. The subobject would have the foreign object *Actor*. Each *primary actor* is existentially dependent on a plurality of particulars – a *Use case* and an *Actor*.

The metamodeling language of WebMeta *is not laconic*, because there are some concepts in the UFO-A ontology which are represented by more than one modeling construct in the metamodeling language (see Table 31.1). The ontology concept *Object universal* is represented by the modeling constructs *Main object* and *Inherited main object*. For instance, one can define the modeling construct *Use case* by creating a main object. However, one can also define the modeling construct *Use case* by creating the main object *Model element* and then creating the inherited main object *Use case* based on the main object *Model element*. The concept *Quality structure* is represented by the modeling constructs *Subobject type* and *Classifier object*. Hence, while creating a metamodel, one has to decide whether to specify the possible values of a property of a language construct by using a predefined *Subobject type* or by specifying a new set of values.

Figure 31.4 presents the *first version* of a redesigned meta-metamodel of the WebMeta metamodeling language. Our goal is not to present *detailed* description of the new language but to *illustrate* useful results of ontological analysis.

Quality structure is a subclass of *Particular* according to the UFO-A ontology. However, *Classifier object* is specified as a subclass of *Object* (the ontological interpretation of which is *Universal*) in the original meta-metamodel (see Fig. 31.3). Hence, we propose to redesign the metamodeling language in a way that *Classifier object* will not be a subclass of *Object* any more. Instead, in the new model, we have class *User-defined (type)* that is a subclass of *Data type*. For the sake of clarity, we propose to rename *Subobject type* to *System-defined (type)*.

Main object has in the new model subclasses that represent the ontology concepts, which are the subclasses of *Object universal* in the UFO-A ontology.

It improves the completeness of the metamodeling language and makes it possible to enforce the constraints, which are prescribed by UFO (Guizzardi 2005) and regulate the relations between these different types of universals. For instance, a language designer could define use-case modeling language construct *Actor* by using the metamodeling language construct *Kind*. One could also define use-case modeling language construct *Primary actor* by using the metamodeling language construct *Role*. However, definition of a generalization relationship, according to which *Primary actor* is the supertype of *Actor*, must be prohibited based on a constraint that is defined in the UFO (Guizzardi 2005).

We have added constructs *Generalization* and *Association* to the metamodeling language to increase the lucidity of the language. The construct *Subobject* is now only used to specify the properties of model constructs in the metamodels (*Quality universals* in terms of the UFO-A ontology).

An ontological analysis of the redesigned metamodeling language (based on the UFO-A ontology) shows that the language is sound and lucid. However, the language is not complete and not laconic. The language is not laconic because the concept *Quality structure* in the ontology is represented by three modeling constructs: *Data type*, *System-defined data type*, and *User-defined data type*. The language is not complete because, for instance, the concept *Concrete particular* is not represented in a modeling construct of the metamodeling language.

3.2 A Metamodeling Language Proposed in Roost et al. (2007)

The metamodeling language, which is proposed in Roost et al. (2007), is very generic. The meta-metamodel has only one class – *Element* (see Fig. 31.5).

The metamodeling language *is not complete* because only the most generic concept of the ontology (*Entity*) is represented by a construct (*Element*) in the metamodeling language.

In addition, the relationships in the meta-metamodel represent some (but not all) *Basic formal relations* in the UFO-A ontology.

The metamodeling language *is sound*. The concept *Entity* in the ontology provides the interpretation to the modeling construct *Element* in the metamodeling language. The metamodeling language *is lucid* because the modeling construct *Element* in the metamodeling language represents exactly one concept (*Entity*) in

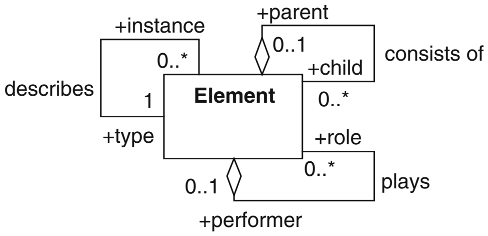

Fig. 31.5 Meta-metamodel of a metamodeling language, which is proposed to use in the context of evolutionary information systems (Roost et al. 2007)

the ontology. The metamodeling language *is laconic* because every concept in the ontology is represented at most once in the meta-metamodel of the language.

Based on the meta-metamodel, an *Element* can be an instance of itself. It violates a disjointness constraint, which is specified in UFO-A, according to which an *Entity* cannot be a *Particular* and a *Universal* at the same time.

3.3 Discussion

It is hard (and in our view unnecessary) to achieve completeness in terms of a complete foundational ontology in case of metamodeling languages. The reason is that each metamodeling language has to provide only a small number of generic modeling constructs. The constructs have to be generic because they will be used to specify different modeling constructs (with different semantics) of different modeling languages. Hence, we used in the analysis only a subset of a foundational ontology – *UFO-A: An Ontology of Endurants*. However, UFO-A specifies the concept *Concrete particular* and its subclasses. A metamodeling language L does not have to contain the constructs that represent these concepts because most of the constructs of L (except *Data type* and its subclasses) represent *universals*, which can be *instantiated* at the lower layer of the metamodeling architecture.

It might be a good idea to change the representation of the meta-metamodel of a language, without changing the semantics of the model, to facilitate the ontological analysis. For instance, in WebMeta (ver 0.5), the database schema, which implements the meta-metamodel, is created according to the model (a) of Fig. 31.6. However, for the analysis, we presented the existence of different types of objects by using subclasses (see part b of Fig. 31.6) and a constraint {Complete; Disjoint}.

The constraint shows that each instance of a superclass (*Object*) belongs to exactly one of the subclasses of *Object*. In case of model, (b) it is easier to map constructs of a metamodeling language with the concepts in an ontology.

The ontological analysis detected only few problems in a very generic and flexible meta-metamodel (see Sect. 3.2). However, in case of this very generic meta-metamodel, one could more easily violate the constraints that are specified in the foundational ontologies, while defining a metamodel. Ontological analysis is used to evaluate only one aspect of a language (its abstract syntax), and it should not

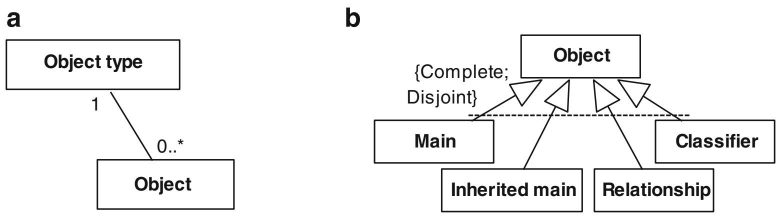

Fig. 31.6 Different representations of a meta-metamodel

be the only evaluation method of metamodeling languages. A problem of the ontological analysis is that it is somewhat subjective in nature and depends on the selection of ontology as well as on the understanding of the ontology and metamodeling languages by the evaluators. However, even in this way, it can give useful information about the deficiencies of a particular language (see Sect. 3.1).

Laarman and Kurtev (2010) use a simple foundational ontology (four-category ontology) to construct an ontologically well-founded metamodeling language (ontology-grounded metalanguage). This chapter complements the paper (Laarman and Kurtev 2010) by suggesting the use of ontological analysis to improve the *existing* languages.

4 Conclusions

In this chapter, we investigated how to conduct an ontological analysis of metamodeling languages. As an example, we performed an ontological analysis of the metamodeling language of our own metamodeling system WebMeta (ver. 0.5) (Eessaar and Sgirka 2010) and a metamodeling language described in (Roost et al. 2007). We used a subset of the Unified Foundation Ontology (namely, UFO-A) as the basis of the analysis. We found several problems of the metamodeling languages and presented a redesigned meta-metamodel of the metamodeling language of WebMeta. We conclude that it is possible and useful to conduct ontological analysis of metamodeling languages. It would help language designers to find problems of languages and compare languages in terms of the number and severity of the problems. However, the investigation of completeness of a metamodeling language would probably lead to the conclusion that the language is incomplete.

Future work should include similar analysis based on the bigger set of metamodeling languages. In addition, it would be necessary to find out whether the ontological analysis of the same metamodeling languages based on other foundational ontologies would give the same results. We also have to continue the improvement of the metamodeling language of WebMeta.

Acknowledgments This research was supported by European Social Fund's Doctoral Studies and Internationalisation Programme DoRa.

References

Date CJ (2006) The relational database dictionary. A comprehensive glossary of relational terms and concepts, with illustrative examples. O'Reilly, Sebastopol

Eessaar E, Sgirka R (2010) A database-based and web-based meta-CASE system. In: International conference on systems, computing sciences and software engineering. Springer, Dordrecht, pp 379–384

Green P, Rosemann M (1999) An ontological analysis of integrated process modeling. In: CAiSE'99. LNCS vol 1626. Springer, Berlin, pp 225–240

Greenfield J, Short K, Cook S, Kent S (2004) Software factories: assembling applications with patterns, models, frameworks, and tools. Wiley, Indianapolis

Guizzardi G (2005) Ontological foundations for structural conceptual models. Telematica Instituut Fundamental Research Series No. 15. Ph.D. thesis, University of Twente

Guizzardi RSS, Guizzardi G (2010) Applying the UFO ontology to design an agent-oriented engineering language. In: ADBIS'10. LNCS vol 6295. Springer, Berlin, pp 190–203

Guizzardi G, Wagner G (2010) Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages. In: Theory and applications of ontology: computer applications. Springer, Dordrecht, pp 175–196

Guizzardi G, Falbo R, Guizzardi SS (2008) Grounding software domain ontologies in the Unified Foundational Ontology (UFO): the case of the ODE software process ontology. In: XI Iberoamerican workshop on requirements engineering and software environments, Recife, Brazil

Kelly S, Tolvanen JP (2008) Domain specific modeling enabling full code generation. A Wiley-Interscience Publication, Hoboken

Laarman A, Kurtev I (2010) Ontological metamodeling with explicit instantiation. In: SLE 2009. LNCS vol 5969. Springer, Heidelberg, pp 174–183

Livingstone D (2008) Simplicity. Systemist 30:16–39

Meta Object Facility (MOF) Core Specification. Version 2.4 Convenience, ptc/2010-12-08

OMG Unified Modeling Language^{TM} (OMG UML), Infrastructure. Version 2.4, ptc/2010-11-16

Opdahl AL, Henderson-Sellers B (2002) Ontological evaluation of the UML using the Bunge–Wand–Weber model. Softw Syst Model 1(1):43–67

Roost M, Rava K, Veskioja T (2007) Supporting self-development in service oriented information systems. In: 7th WSEAS international conference on applied informatics and communications, Athens, Greece, pp. 52–57

Rosemann M, Green P, Indulska M (2004) A reference methodology for conducting ontological analyses. In: ER 2004. LNCS vol 3288. Springer, Heidelberg, pp 110–121

Siau K, Rossi M (1998) Evaluation of information modeling methods-a review. In: Thirty-first Hawaii international conference on system sciences, 5th edn. IEE Computer Society, Los Alamitos, pp 314–322

Wand Y, Weber R (1990) An ontological model of an information system. IEEE Trans Softw Eng 16(11):1282–1292