Ontological Metamodeling with Explicit Instantiation

Alfons Laarman, Ivan Kurtev

Department of Computer Science, University of Twente, the Netherlands
{a.w.laarman, kurtev}@ewi.utwente.nl

Abstract. Model Driven Engineering (MDE) is a promising paradigm for software development. It raises the level of abstraction in software development by treating models as primary artifacts. The definition of a metamodel is a recurring task in MDE and requires sound and formal support. The lack of such support causes deficiencies such as conceptual anomalies in the modeling languages. From philosophical point of view metamodels can be seen as metaconceptualizations. Metalanguages have to provide constructs for building ontological theories as a base for modeling languages. This paper describes a new metalanguage derived from the study of Formal Ontology. This metalanguage raises the level of abstraction of metamodels from pure abstract syntax to semantics descriptions based on ontologies. Thus, the language developers can make conscious choices for their modeling concepts and can explicitly define important relations such as instantiation and generalization. With this metalanguage, we aim at a precise conceptual and formal foundation for metamodeling.

Keywords: Metamodeling, ontologies, **instantiation semantics**

1 Introduction

Model Driven Engineering (MDE) relies on models and model transformations for development of software systems. We perceive models are symbolic entities expressed in a modeling language. The traditional way for defining a language is to define first its grammar. In MDE, the central concept is the language *metamodel*. Metamodels are often regarded as definitions of the language abstract syntax. Currently, language developers are supported by several, mainly object-oriented metamodeling languages: ECore, MOF, KM3 [13].

Concerning metamodeling, several problems with a conceptual nature are still open. Atkinson and Kühne distinguish between *linguistic* and *ontological* metamodeling based on linguistic and ontological instantiation respectively [2]. They argue that both types of metamodeling are equally important. The linguistic instantiation is used by the modeling tools builders. The ontological instantiation is used by domain experts. However, it is difficult to build ontological models and metamodels because the current metalanguages support mainly linguistic metamodeling and do not provide first-class constructs for ontological metamodeling.

Furthermore, Guizzardi [9] evaluates UML for its suitability to perform conceptual modeling. The base for evaluation is a *foundational ontology* elaborated by the author and incorporating previous work presented in [11] and [7]. The foundational ontology

specifies the structures in the world recognized by the study of *formal ontology*. Guizzardi shows that the ontological meaning of models based on formal ontology cannot be retained when these models are expressed in UML. The UML language has several anomalies that decrease the quality of models. Examples of such anomalies are *construct overloading*, *construct redundancy*, and *construct incompleteness*.

The current metamodeling practice demonstrated by the metalanguages from the MOF family does not consider the ontological foundations of (meta-) modeling. Since the MOF corresponds to UML infrastructure, any modeling language (domain-specific or general-purpose) can potentially suffer the same anomalies found in UML.

The described problems emerge due to two reasons: *lack of clear understanding* of the metamodeling activity regarding its ontological foundation and *lack of constructs* in the current metalanguages to express required explicit information.

We address the problems described above by proposing a view on the content of metamodels and a new metalanguage. In our approach, metamodels are lifted from pure abstract syntax definitions to expressions of *metaconceptualizations* based on a *foundational ontology*. We retain the structural definition of a language and enhance it with ontological meaning. The philosophical justification of our approach comes from the statement of Quine that in every language an ontology can be found [19]. Thus, the metamodeling activity is a task that identifies and specifies the world structures that are of an interest to solve a given problem. The metalanguage has to be capable to express such structures.

We use a simple foundational ontology *Four-category Ontology* to build a new metalanguage. We propose Ontology Grounded Metalanguage (OGML) as an experimental language for studying the definitions of metamodels based on ontological principles. In OGML, linguistic and ontological instantiations are treated uniformly from technical perspective. Both are defined on the basis of the explicit *instanceOf* definition construct in OGML.

The paper is organized as follows. Section 2 clarifies the meaning of the concepts used in the paper. Section 3 presents the Four-category ontology and compares it with existing foundational ontologies. Section 4 describes OGML by examples. Section 5 discusses the main open issues and positions our approach within the existing work. Section 6 concludes the paper.

2 Conceptual Background

The title of this paper refers to terms that are interpreted in different ways in the literature: *ontology* (ontological), *metamodel(-ing)*, and *instantiation.* We give a short background on these terms and give our understanding.

A commonly accepted notion of metamodel is that it is a model of models expressed in a given language. Thus, a metamodel defines the constraints for all the admissible models expressed in the language. Often, the metamodel is regarded as a definition of the abstract syntax of the language.

The term ontological metamodeling (and ontological instantiation) was introduced by Atkinson and Kühne in [2]. They distinguish between linguistic and ontological metamodeling. Fig. 1 illustrates the distinction between them in the context of the three-levels MOF architecture.

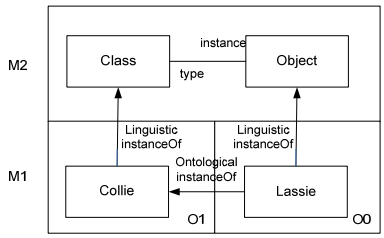

Figure 1. Linguistic and ontological instantiation

Linguistic metamodeling is used to define metamodels of languages. The instances of metamodels are models at M1 obtained by linguistic instantiation. Linguistic metamodeling defines the *form* that a statement (model) in a language may take. Linguistic instanceOf delimits metalevels (e.g. M1 and M2). Ontological metamodeling allows the type/instance relation to exist within a single metalevel. In Fig. 1 the object Lassie is an instance of the class Collie. The *instanceOf* relation is called ontological and it is concerned with the *content* that a statement (model) has by representing a particular domain. The ontological *instanceOf* partitions models into ontological levels (e.g. O1 and O2) within a single linguistic level. The linguistic instanceOf is defined by the metalanguage used to define metamodels (for example, MOF) and the ontological instanceOf is defined by a particular modeling language (for example, UML).

Guizzardi [10, 8] studies the relation between metamodels and ontologies. He recognizes two distinct purposes of metamodels: as a definition of the abstract syntax and as a definition of the *world view* underlying the language. Assume that we would like to define a language that describes state of affairs in a given domain (Fig. 2).

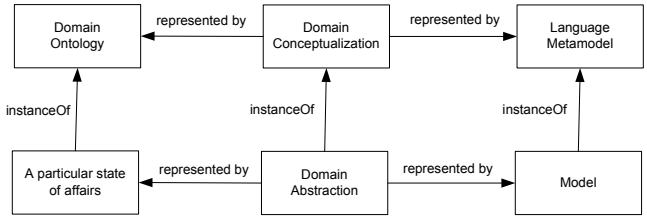

Figure 2. Domain conceptualization and metamodel

The middle column in Fig. 2 represents domain abstractions and a domain conceptualization. They are conceptual entities in the modeler's mind. In order to communicate them we define a language to be used to specify models. In Fig.2, we show the language metamodel. Guizzardi understands the term metamodel as a specification of the world view of the language, that is, the description of what a language can describe in terms of real world phenomena. The capability of the language to express certain domain is measured by comparing the elements of the metamodel to the elements of the representation of the domain conceptualization called domain ontology. Here the domain ontology is supposed to be the best possible representation of the domain conceptualization. The smaller the gap between the domain ontology and the metamodel is the more precisely the models can represent the real world phenomenon in the domain.

Unfortunately, current practice of metamodeling in MDE mostly treats metamodels as definitions of the abstract syntax. Metamodelers are not aware of the real world meaning of the language constructs. The result is decreased quality of the models due to anomalies in the modeling languages. Metalanguages such as MOF are not expressive enough to articulate the difference between various modeling constructs. Consider for example the model elements Collie and Lassie in Fig. 1. They are instances of MOF classes, that is, they are MOF objects. However, the real world meaning is rather different. *Lassie* represents an individual, a concrete collie. *Collie* represents the characteristics of all the dogs of this breed, that is, it captures the universal properties of the collies. In a MOF-like architecture, this difference is not expressible. Furthermore, the metatypes Class and Object classify types and individuals respectively, so they are different. Both are instances of MOF Class and consequently indistinguishable by the MOF-based tools. Finally, the definition of the ontological *instanceOf* in UML is just a MOF association and is treated as any other association in the UML metamodel.

We aim at retaining ontological properties of the metamodels by treating them as representation of the language underlying world view. Therefore, metamodels become more than descriptions of the abstract syntax of a language. They are enriched with explicit knowledge of the ontological nature of their constructs. When we talk about explicit instantiation, we mean that a metamodeling language provides us with a first-class construct for defining ontological instantiations according to the understanding of Kühne.

3 Approach

In MDE, metamodels are expressed in a language called metalanguage. Current metalanguages are mainly object-oriented due to pragmatical reasons such as familiarity to the developers and tool support. If we perceive a metamodel as something more than a structural definition, then we need to study the requirements for a suitable metalanguage.

Consider the upper layer in Fig. 2. The domain ontology is an artifact expressed in a language. What is the domain conceptualization of this language? What is the "ideal" ontology that captures this conceptualization? According to Guizzardi, we can apply the pattern in Fig. 2 by treating domain conceptualizations as a domain of study. The result of the application is shown in Fig. 3.

The set of various domain-specific conceptualizations is conceptualized in a domain-independent metaconceptualization. The representation of this metaconceptualization as an ontology is called *Foundational Ontology*. It is derived from the study of *Formal Ontology*. Several authors provide concrete versions of Fig. 3. Wand [20] uses the Bunge-Wand-Weber (BWW) ontology as a foundational ontology and UML as a language for expressing domain models. Guizzardi performs a similar study on UML by using Unified Foundation Ontology (UFO) as a foundational ontology. The two approaches study the ontological correctness of UML metamodel.

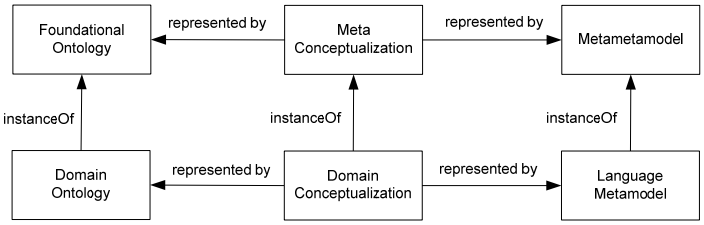

Figure 3. Ontologies and metaconceptualization

We aim at formulating language metamodels by using a vocabulary derived from a foundational ontology. In this way, the constructs of metamodels become instances of the most fundamental and domain-independent ontological categories. For example, UML Class and ER Entity are classified as constructs that are used to represent classifiers (or universals). Although they belong to different languages, they have a similar ontological nature. In this way, metamodels carry additional ontological information that can be used to align and compare metamodels with each other as well as to a given foundational ontology.

The approach for treating metamodels as representation of metaconceptualizations leads to the following interpretation of the metalevels:

- M1: models that represent reality. They are expressed in a modeling language;
- M2: metamodels of modeling languages that represent the real world view embodied in the language;
- M3: a metametamodel of a metalanguage. The metalanguage is used to express various worldviews. It is derived from a metaconceptualization, which in turn is derived from a foundational ontology;

To proceed with this approach we need to select a Foundational Ontology. We examined several existing foundational ontologies: UFO, DOLCE [5], BWW. We used the following criteria for selecting a foundational ontology:

- The ontology should be simple;
- The constructs should be familiar to the developers;
- The ontology should allow expressing the metamodels of the major existing programming, data description, and modeling languages, both general purpose and domain specific;

Considering these requirements we opt for a descriptive minimalistic ontology, like in the approach of Guizzardi and Wand et al. Also because our work can be considered as an initial experiment in applying formal ontology theory in metamodeling, we chose a small foundational ontology called Four-category Ontology (FCO). For the sake of minimality, we did not include the refined concepts of universals such as *sortal*, *role*, *category*, etc. found in UFO.

In FCO, the basic distinction is between *individuals* and *universals* as the most fundamental entities of being. Figure 4 depicts the concepts in this ontology. Individuals are classified as *Substantial* and *Moment* individuals. Substantial individual or just *substance* is something that can exist by itself without depending on the existence of other individuals. In the programming languages and modeling languages, substantial individuals are usually represented as objects (e.g. Java object and UML object).

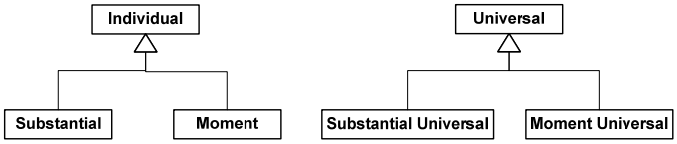

Figure 4. The Four-Category Ontology

Moments are individuals that exist in other individuals. Moments cannot exist standalone, they are existentially dependent on at least one individual (called *bearer*). The relation between a moment and its bearer(s) is called *Inherence* relation. Moments may inhere in more than one individual. In programming and modeling languages, moments are called in various ways: *slot* and *link* in UML, *field* in Java, etc.

Universals are entities that can be instantiated in individuals. The individuals that exemplify a universal have something in common. For example, things that consist of matter have a mass. In this case mass is a universal.

Universals are classified into *substantial* and *moment universals*. Substantial universals are exemplified by substantial individuals and moment universals are exemplified by moment individuals. *Instantiation relation* is the relation between an individual and a universal. Universals have their representatives in the existing computer languages. UML classes correspond to substantial universals. UML attributes and associations correspond to moment universals.

4 Ontology Grounded Metalanguage

OGML is our experimental metalanguage based on FCO. It helps the language developers to make conscious choices for their modeling concepts and enforces the definition of important relations such as instantiation and generalization. In the current section, we introduce OGML by defining the metamodel of a tiny subset of UML, called *Simple UML*. The metamodel of the language is shown in Fig. 5 (left part) together with an example model (right part) and *instanceOf* relations. The upper part represents class diagrams and the lower part object diagrams.

A metamodel expressed in OGML consists of definitions. Definitions describe how a particular language conceptualizes the world by defining the structure of universals and individuals. In addition to this, a metamodel may define explicitly the instantiation and generalization relation of the language.

UML classes, for example, the Crocodile**,** are *substantial universals* from ontological point of view. We instantiate the OGML construct *SubstantialDefinition* to express that the element Class in the UML metamodel defines the structure of substantial universals (lines 1-2).

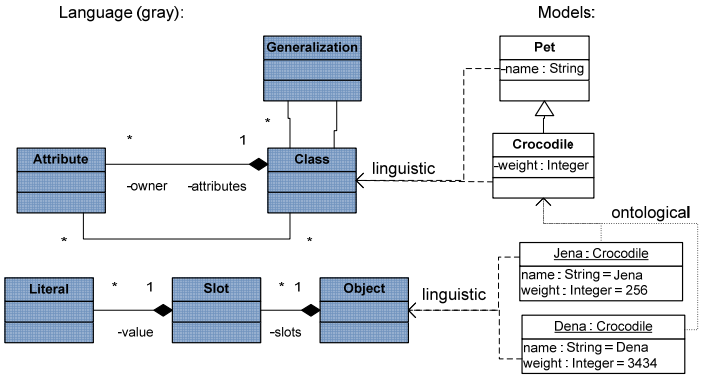

Figure 5. The example language SimpleUML

Classes have attributes, which are in turn *moment universals*, expressed as instances of the OGML construct *MomentDefinition* (lines 4-9). The relation between a moment definition and the substantial definition(s) is called c*haracterization relation.* The definition of Attribute states the fact that concrete attributes are attached to a single class. Since characterization relation connects two constructs, it has two roles: the *universalDefinitionRole* and a *momentDefinitionRole*. To define UML Association then we could instantiate *MomentDefinition* with two characterization relations. This expresses the fact that the instances of associations (called *links* in the context of UML) are moments that inhere in two individuals. It should be noted that OGML allows a moment definition to characterize another moment definition. This ultimately allows a moment to inhere in another moment. This is a major difference with the BWW ontology where properties do not have properties.

The definition how UML represents individuals follows a similar structure. Substantial individuals are defined by instantiating *ObjectDefinition* (lines 11-12) and moments are defined by a *PropertyDefinition* (lines 14-17). The fact that UML slots inhere in UML objects is expressed by the *dependsOn* clause.

```
1. SubstantialDefinition Class {
2. }
3.
4. MomentDefinition Attribute {
5.       attribution universalDefinition = "Class"
6.                   universalDefinitionRole = "owner"
7.                   momentDefinitionRole = "attributes"
8.                   multiplicity = 1-*;
9. }
10.
11. ObjectDefinition Object {
12. }
13.
14. PropertyDefinition Slot {
15.       value : String;
16.       dependsOn Object role = "slots" multiplicity = *;
17. }
```

An important construct in OGML is the definition of *instanceOf* relations. In the terminology of Fig. 1, OGML metamodels defines instantiation relation as a first class construct. Let us consider the definition of UML instanceOf. We need to express the facts that (a) classes are instantiated to objects and attributes to slots; (b) an individual

can be queried for the values of its moments and the values obey certain constraints. The concrete syntax is illustrated in the following listing. Line 2 states that every class is instantiated to an object. In this case, substantial universals are instantiated to substantial individuals.

```
1.  Relations UMLInstanceOfAssociationsOnLinks {
2.    c : Class -> o : Object {
3.    }
4.    a : Attribute -> s : Slot  {
5.          attribution {
6.                naming name <- a.name;
7.                valuing [a.lowerbound .. a.upperbound] s.value;
8.                typing a.type;
9.          }
10.   }
11. }
```

OGML allows substantial universals to be instantiated to other universals, thus achieving a multilevel ontological metamodeling according to Fig. 1. Line 4 states the attribute moment universals are instantiated to slots. If an UML object has a set of slots then the object may be queried by using the name of the slot, which is obtained as the name of the defining attribute (line 6). The value of the slot is stored in its value property (line 7). Lines 7 and 8 also specify multiplicity and typing constraints. Querying the value of a moment is based on the concept of attribute function used in BWW ontology. For each moment, at least one attribute function is defined. In our example, slots are unary moments and only one attribute function is needed. If a moment inheres in more than one individual then an attribute function is defined per characterization relation. Note that line 5, explicitly names the characterization to which the attribute function is assigned (*attribution*).

OGML explicitly defines its own *instanceOf* relation following the same idea illustrated in the *SimpleUML* example. Hence, from the perspective of the tools using models, there is no technical difference between the linguistic and ontological instantiations. We built a tool [18] that allows expressing OGML metamodels and conforming models in a concrete syntax. By having two models that are related either by linguistic or ontological instanceOf, and the metamodel of their language, the tool is capable of checking the conformance between the models by using a single algorithm. The tool provides full OCL support with an extension for dealing with multiple classifications of a given model element (for example, the crocodile Jena is an instance of Object from the point of view of OGML and an instance of Crocodile from UML point of view).

5 Discussion and Related Work

The design of OGML raises multiple questions. The first question is the choice of a foundational ontology. We opted for FOC due to its simplicity and the observation that its constructs are usually represented in some form in many computer languages. However, in the current version of OGML it is not possible to treat properly primitive data types such as integers, booleans, etc. They are equalized to substantial individuals, which is ontologically debatable. OGML needs to be extended with constructs for defining abstract entities, for example, mathematical structures.

The second question is how to incorporate a full-fledged Foundational Ontology such as UFO. One possibility is to extend OGML. This will result in a large

metametamodel with many constructs needed for conceptual modeling only. Another possibility is to define a foundational ontology as a metamodel. In any case, committing to a certain foundational ontology as a theoretical base for OGML poses an immediate limitation that all the models in the modeling space become more or less aligned with the world view of one ontology. However, there may be other foundational ontologies that are perfectly possible alternatives.

The third question is about how OGML relates to the existing self-reflective metametamodels. OGML is defined as a self-reflective metametamodel [17]. This definition poses interesting challenges that deserve a separate paper, and is intentionally omitted here due to a lack of space.

We claim that technically the linguistic and ontological instantiations are the same, at least because they are all expressed by a single OGML construct. On the other hand, the work by Kühne [14, 15] and Gasevic [6] indicate the opposite. We have to clearly state that we do not claim conceptual equivalence between the two types of instantiations. In [14, 15, 6] they are distinguished mainly on the basis of the nature of the represented systems by the so-called *represents* or μ relation. In our work, we do not represent this relation, hence this difference is not apparent. Furthermore, our understanding of *instanceOf* is a shortcut similar to the *conformantTo* relation used by Bezivin, Favre, and Gasevic. When we say that an object *o* is an instance of class *C* according to a certain definition of the *instanceOf* relation, we mean the following. *o* is a member of the extension of *C*, where the membership is checked on the basis of the semantics of OGML instanceOf definition construct (encoded in the tool) and the intensional representation of *C*. On the other hand, the intensional representation of *C* perceived simply as an expression in a given language may be a member of the extension of another class. Clearly, Gasevic made this same distinction.

It should be noted that the difference between the ontological and linguistic instantiations and the nature of a metamodel are still debatable [12]. A language with at least three levels of ontological instantiation may allow representation of MOF, MOF metamodels, and MOF models in a single level. Then, the linguistic instantiation in the context of MOF becomes ontological. Thus, these two concepts appear to be relative. It is beyond the scope of this paper (and the space does not permit) to discuss this issue.

Atkinson and Kühne [1] propose an approach for multilevel metamodeling in which a modeling construct is assigned with a *potency* that indicates how many times it can be instantiated. Although this seems reasonable from technical point of view, there is no guidance to the modeler how to assign the potency value. We believe that considering the ontological nature of a modeling construct is a clearer way to reason about the instantiations.

6 Conclusions

In this paper, we proposed a view on metamodeling that treats metamodels as specifications of the world view embodied in a modeling language. This view is regarded as a metaconceptualization and is expressed in a metalanguage called OGML built upon a foundational ontology. As such, metamodels are more than just a definition of the abstract syntax of a language. In addition, we provide a construct for explicit definition of instantiation relation for the modeling languages and it is applied

to the OGML itself. This enables support of ontological metamodeling based on formal ontology theory and uniform treatment of linguistic and ontological instantiation in the modeling tools.

We envision at least two promising applications of this approach: interoperability in the line of [3] and enhancing the set of transformation scenarios in MDE as described in [16]. These two applications together with a proper formalization of OGML are the main directions for a future research.

References

1. Atkinson, C., Kühne, T. The Essence of Multilevel Metamodeling. UML 2001: 19-33
2. Atkinson, C., Kühne, T. Model-driven development: a metamodeling foundation. *IEEE Software*, 20(5), pp. 36-41, 2003
3. Atzeni, P., Cappellari, P., Torlone, R., Bernstein, P.A., Gianforme, G. Model-independent schema translation. VLDB J. 17(6): 1347-1370 (2008)
4. Degen, W., Heller, B., Herre, H., Smith, B. GOL: toward an axiomatized upper-level ontology. FOIS 2001: 34-46
5. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A. Sweetening WORDNET with DOLCE. AI Magazine 24(3): 13-24 (2003)
6. Gasevic, D., Kaviani, N., Hatala, M. On Metamodeling in Megamodels. MoDELS 2007: 91-105
7. Guarino, N., Welty, C.A. A Formal Ontology of Properties. EKAW 2000: 97-112
8. Guizzardi, G. Ontological Foundations for Structural Conceptual Models. PhD thesis. University of Twente, 2005. ISBN 90-75176-81-3
9. Guizzardi, G., Ferreira Pires, L., van Sinderen, M. An Ontology-Based Approach for Evaluating the *Domain Appropriateness* and *Comprehensibility Appropriateness* of Modeling Languages. MoDELS 2005: 691-705
10. Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. DB&IS 2006: 18-39
11. Heller, B., Herre, H. Ontological Categories in GOL. Axiomathes 14: 71-90, Kluwer Academic Publishers, 2004
12. Hesse, W. More matters on (meta-)modelling: remarks on Thomas Kühne's "matters". Software and System Modeling 5(4): 387-394 (2006)
13. Jouault, F., Bézivin, J. KM3: a DSL for Metamodel Specification. FMOODS 2006, Bologna, Italy, 14-16 June 2006
14. Kühne, T. Matters of (Meta-)Modeling. Software and System Modeling 5(4): 369-385 (2006)
15. Kühne, T. Clarifying matters of (meta-) modeling: an author's reply. Software and System Modeling 5(4): 395-401 (2006)
16. Kurtev, I. van den Berg, K. MISTRAL: A Language for Model Transformations in the MOF Meta-modeling Architecture. MDAFA 2004: 139-158
17. Laarman, A.W. (2009) An Ontology Based Metalanguage with Explicit Instantiation. Master's thesis, University of Twente.
18. OGML website, http://wwwhome.cs.utwente.nl/~laarman/ogml/, retrieved at 15-9-09
19. Quine, W.V.O. 'Ontological relativity' and other essays. New York: Columbia University Press, 1969
20. Wand, Y, Storey, V., Weber, R.: An Ontological Analysis of the Relationship Construct in Conceptual Modeling. ACM Trans. DB Syst. 24(4): 494-528 (1999)