ONTOLOGICAL MULTILEVEL MODELING LANGUAGE

Sergey Shavrin

Abstract: This paper presents ontological multilevel modeling language O_2ML, aimed at using with metadata driven information systems. The first part of this paper briefly surveys existing modeling languages and approaches, while the last part proposes a new language to combine their benefits.

Keywords: Metamodeling, information systems, modeling languages.

ACM Classification Keywords: H.0 Information Systems - General.

Introduction

Information systems development comprises a diversity of artifacts creation, e.g. domain model, users guide, code, set of tests, etc. Short term company productivity depends on availability of tools that can ease or automate the process of artifacts creation and usage. However, medium and long term productivity in many respects depends on universality of these artifacts.

Rising of abstraction level is a common way of universalization and therefore a way of artifacts life prolonging. However, abstracting increases semantic gap between an artifact and a machine, thus leading to translation necessity. As is well known there are two types of translators – compilers and interpreters. Overwhelming majority of contemporary CASE-tools utilize compiler approach. Benefits are obvious: translation process executes once, before system exploitation, thus saving target machine resources. On the other hand, interpreter-based systems exhibit great flexibility. The last property appears to be more valuable in the modern circumstances.

Given the interpreter-based information system, domain model is the natural candidate for the role of "control program". In this case system must have capabilities to understand and execute models described in some modeling language. The most widespread modeling language nowadays is UML [7]. At the moment, OMG (Object Management Group) is working on the second version of the language and concomitant standards. Not all specifications had been published yet, but we can already say that a huge amount of work had been done to formalize UML semantics. Completely formalized semantics sets the stage for building unified UML virtual machine. Figure 1 shows a UML-model example.

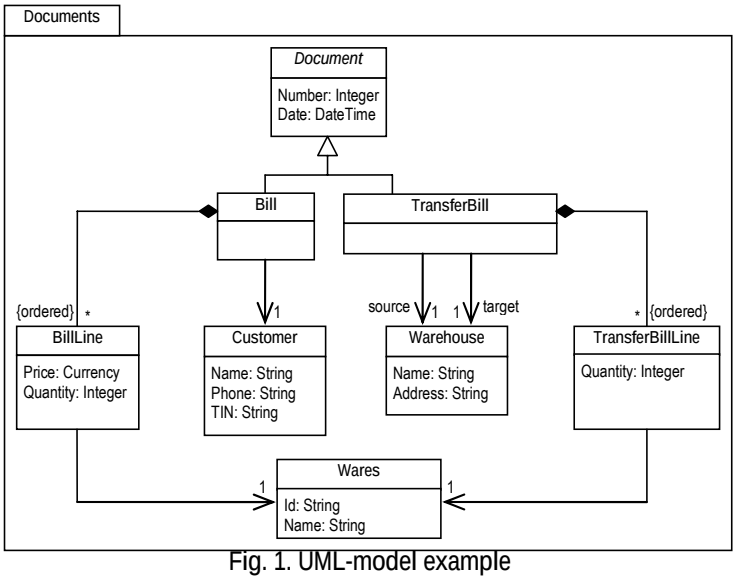

Fig. 1. UML-model example

Given an appropriate tool support, exploiting domain model as a primary artifact significantly increases short term company productivity. However, domain model is prone to become out of date. On the other hand, adjacent domains can be described using similar models differing in details. In this case company can increase its medium and long term productivity by exploiting metamodels which describe more stable metaaspects, common to a set of domains.

UML offers far from complete metamodeling capabilities. These capabilities include stereotypes and tagged values. Powerful metamodeling language has to be able to operate with full-value metaentities at arbitrary number of metalevels.

OMEGA Project

OMEGA [4] – Ontological Metamodeling Extension for Generative Architectures – is a MOF [6] (Meta Object Facility – UML metamodel) extension that introduces ontological metamodeling. OMEGA is aimed at code generation.

OMEGA project introduces a series of notions that enable full-value ontological metamodeling. These notions include metaclasses, metaattributes and metaassociations. It is essential that metaattribute in this case isn't just a metaclass attribute, but a full-value metaentity. An instance of metaattribute is a conventional attribute. This allows one to model such domain features as "Document of each type has exactly one numeric attribute (document number), not less than one date attributes and several property attributes" (Fig. 2 and Fig. 3).

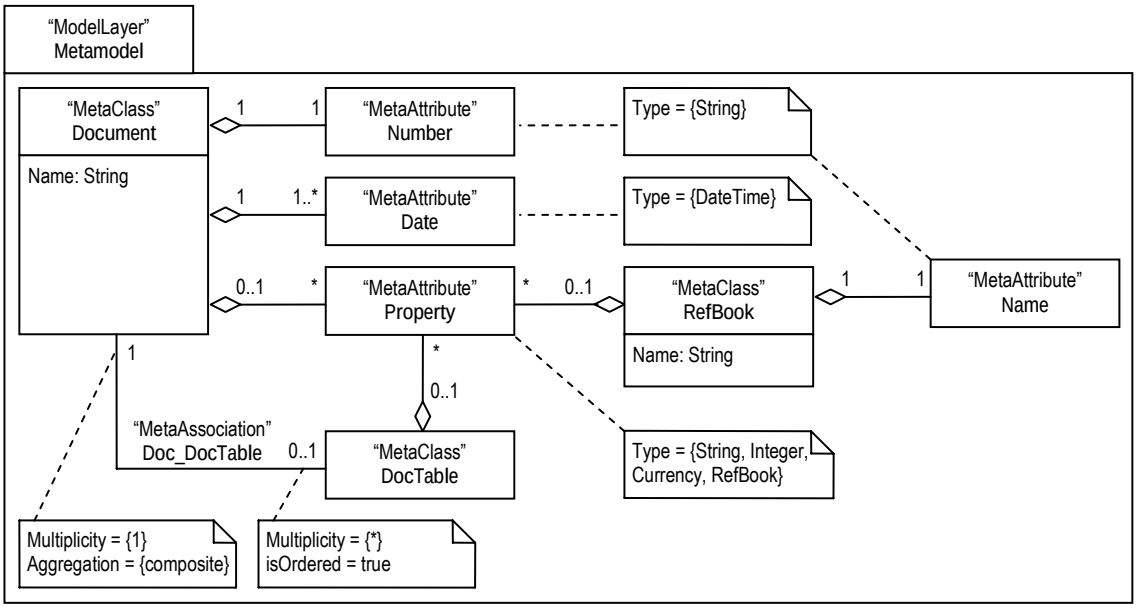

Fig. 2. OMEGA-metamodel example

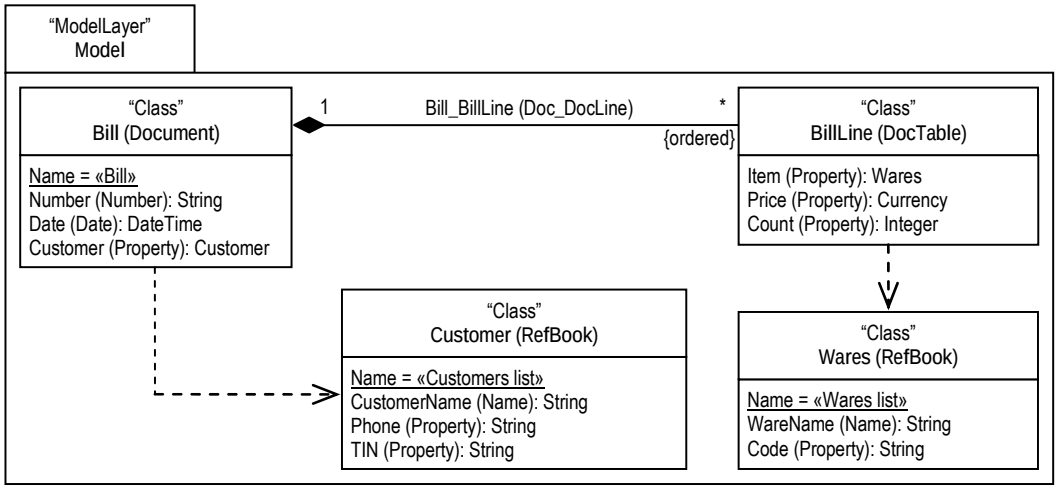

Fig. 3. OMEGA-model example

However OMEGA has two disadvantages. First of all, OMEGA is based on MOF and therefore inherits all its features. In particular MOF is aimed at describing languages like UML and CWM [5] doesn't have some capabilities that are useful in information systems' domain modeling. Namely MOF (and therefore OMEGA) doesn't support plural multidimensional classification – a very convenient modeling tool in author's opinion.

The second disadvantage concerns OMEGA semantics - its description is mainly informal. This fact complicates OMEGA virtual machine creation.

Deep Instantiation

Speaking about instantiation one usually have in mind shallow instantiation. This implies that an instance is created in accordance with its class definition. In other words defining a class we make assertions about its immediate instances. Obviously, it is the only possible interpretation of instantiation in two-level "class-instance" model. However, exploitation of this notion in multi-level case can result in a series of problems. In particular, ambiguous classification and replication of concepts arise [1, 2].

In order to solve shallow instantiation problems Atkinson and Kühne proposed to use a new notion of deep instantiation [2]. This notion allows one to make assertions not only about immediate instances, but instances of instances and so on. This capability is gained by introduction of potency notion – a number that defines allowed instantiations quantity. For example, an instance of class with potency 2 (metaclass) is a class with potency 1 (ordinary class). And an instance of class with potency 1 is a class with potency 0 (object). Similarly, an attribute with potency 2 becomes an attribute with potency 1 (ordinary attribute) that, in its turn, becomes an attribute with potency 0 (slot). Figure 4 shows an example of potency exploitation.

Aside from potency, Atkinson and Kühne introduced a *dual field* notion – an object possessing attribute and slot semantics [2]. In terms of potencies dual field is a slot with non-zero potency. Figure 4 shows some dual field examples, namely "EntityName" and "Description".

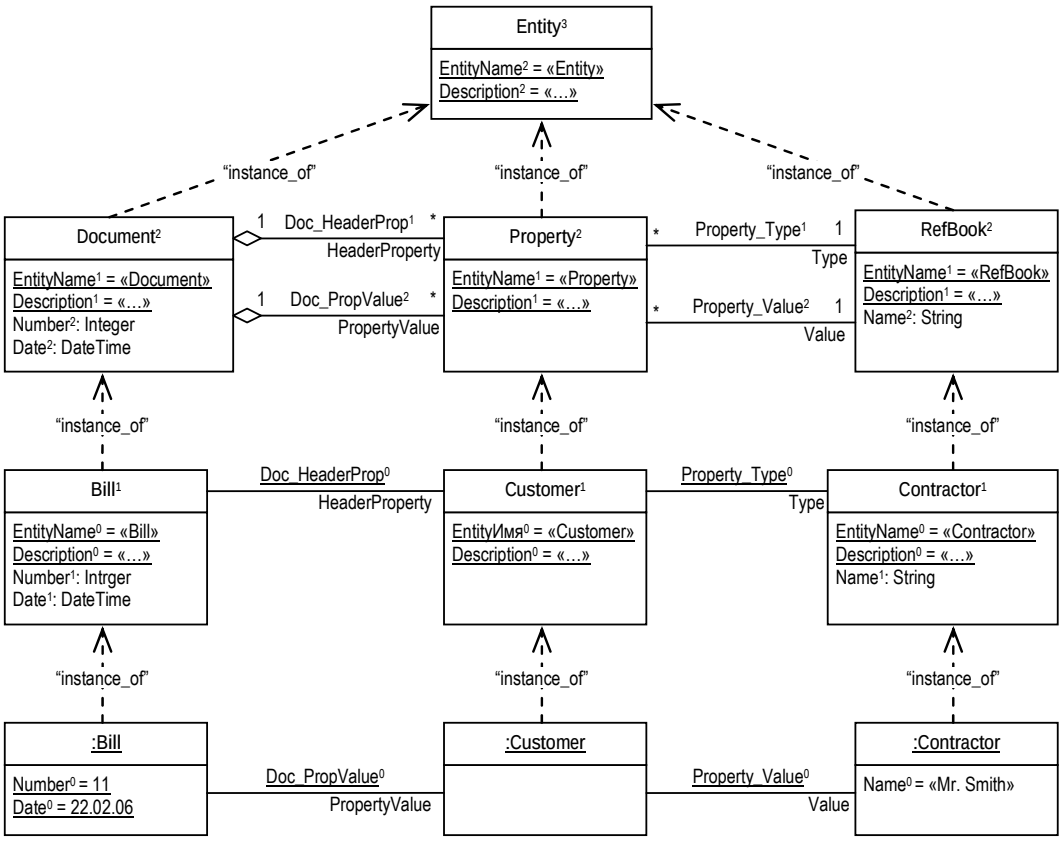

Fig. 4. Potency usage example

In spite of the fact that potency allows to avoid multilevel modeling problems mentioned above, it is obviously insufficient to solve real-world problems. The language with potency support proposed in [2] is too simple to be used in practice. There is a need in additional metamodeling tools like metaattributes and metaassociations.

O_2ML

This part proposes a new modeling language – O_2ML (Ontological Multi-Level Modeling Language). This language combines the best modeling features considered above. Namely, O_2ML is based on:

- UML – reach intra-level modeling capabilities (plural inheritance, plural multi-dimensional classification);
- OMEGA – reach inter-level modeling capabilities (metaclasses, metaattributes, metaassociations);
- Deep Instantiation – multi-level modeling support (potency values).

Figures 5 to 7 show O_2ML usage example. One can see on these figures that potency values allow reducing metaattributes quantity. This leads to more simple and compact models. Formally, an attribute with potency value of $n > 1$ is a metaattribute with potency value of $n - 1$ that satisfies following constrains:

- set of allowed types is constrained to only one type;
- instance quantity in each owner-class instance is precisely one;
- instances names replicate their parent name.

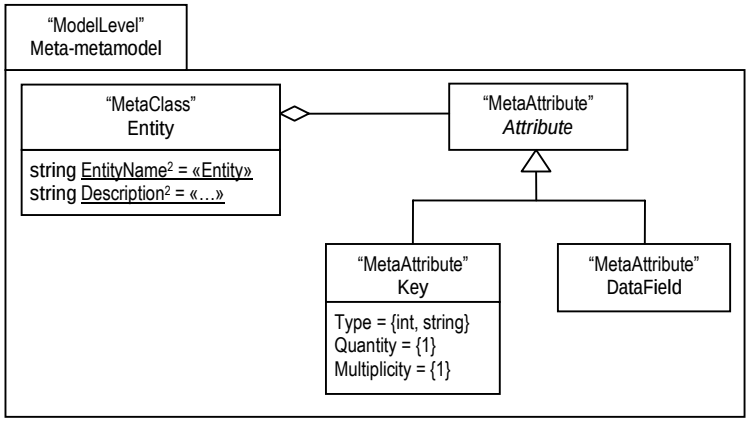

Fig. 5. O_2ML-meta-metamodel example

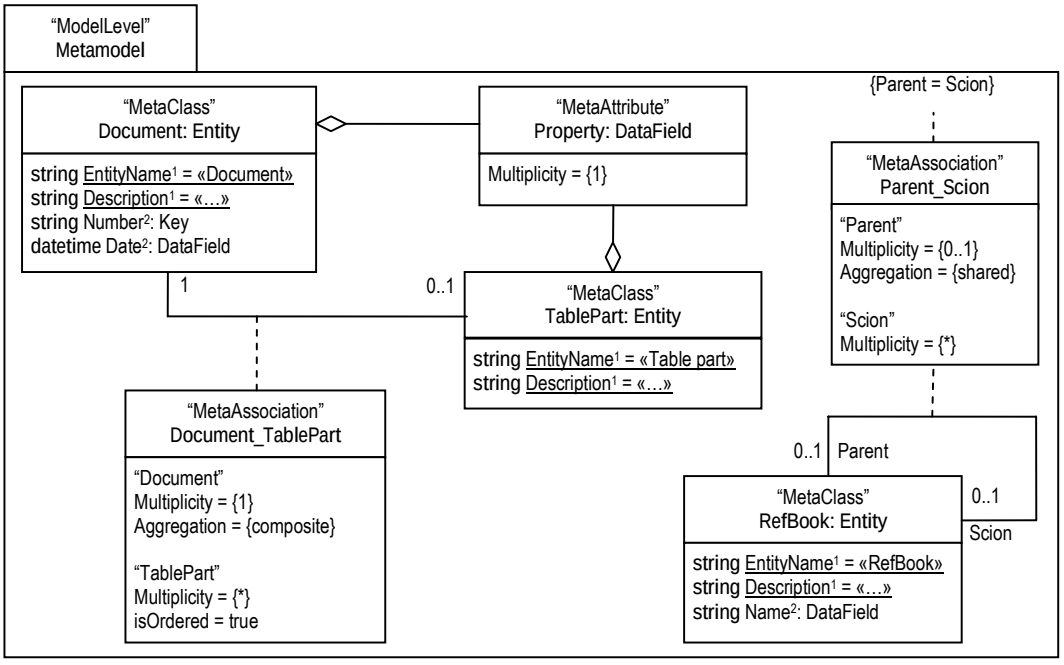

Fig. 6. O_2ML-metamodel example

O_2ML graphical notation uses attribute definition syntax that differs from the one used in UML. The syntax is as follows: <Type> <Attribute Name><Potency value>: <Metaattribute Name>. This approach conforms to the fact that metaattribute is a classifier for corresponding attributes.

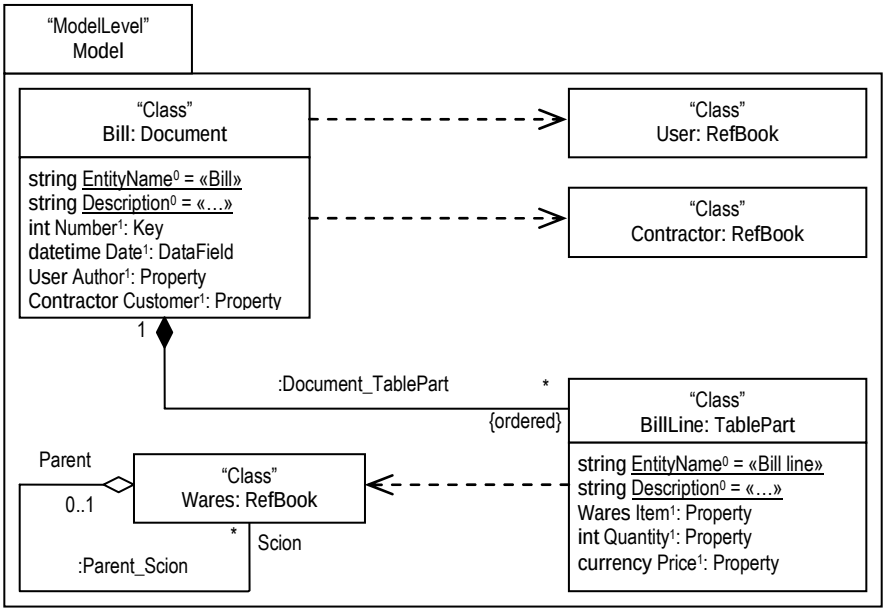

Fig. 7. O_2ML-model example

Important O_2ML feature is that its semantics is described formally. This fact eases O_2ML virtual machine creation. The semantics is described using XOCL (eXecutable OCL) – an extension of OMG's OCL and a part of XMF (eXecutable Metamodeling Framework) [3].

Bibliography

[1] Atkinson C., Kühne T. Rearchitecting the UML Infrastructure. ACM Transactions on Modeling and Computer Simulation, Vol. 12, No. 4, October 2002.

[2] Atkinson C., Kühne T. The essence of multi-level metamodeling. In Proceedings of the Fourth International Conference on the Unified Modeling Language, M. Gogolla, C. Kobryn, Eds., Lecture Notes in Computer Science, vol. 2185, 19–33, 2001.

[3] Clark T., Evans E., Sammut P., Willans J. Applied Metamodelling: A Foundation for Language Driven Development http://albini.xactium.com/web/downloads/b1a35960appliedMetamodelling.pdf, 2004.

[4] Gitzel R., Ott I., Schader M. Ontological Metamodel Extension for Generative Architectures (OMEGA), Working Paper, University of Mannheim, Department of Information Systems III, http://www.bwl.uni-mannheim.de/Schader/_files/gitzel-omega.pdf, June, 2004.

[5] Object Management Group, Common Warehouse Metamodel, http://www.omg.org/technology/cwm, 2001.

[6] Object Management Group, Meta Object Facility Core v2.0, http://www.omg.org/cgi-bin/doc?formal/2006-01-01, January 2006.

[7] Object Management Group, UML Superstructure Specification v2.0, http://www.omg.org/cgi-bin/doc?formal/05-07-04, July 2005.

Author's Information

Sergey Shavrin – Perm state university, computer science department, senior lecturer; e-mail: shavrin@gmail.com