TECHNICAL TRANSACTIONS 5/2017
CZASOPISMO TECHNICZNE 5/2017
COMPUTER SCIENCES
DOI: 10.4467/2353737XCT.17.071.6428

Piotr Zabawa (pzabawa@pk.edu.pl)
Department of Physics, Mathematics and Computer Science, Cracow University of Technology

Notions of context-driven meta-modeling (cdmm)

Pojęcia metamodelowaniasterowanego kontekstem (cdmm)

Abstract

The paper focuses on the meta-model notions and introduces new terminology to the meta-modeling discipline. This terminology refers to the new concept of Context-Driven Meta-Modeling (CDMM), which is more general then other approaches to defining modeling languages. In the result it extends the meta-modeling domain vocabulary. All notions in the paper were introduced on the basis of the decomposition of responsibilities identified in meta-models. Till now modeling languages were assumed to be compact and monolithic structures, in contrast to good design practices, not being a subject of any decomposition. The system of notions introduced in the paper is general enough to be sufficient when Context-Driven Meta-Modeling Paradigm (CDMM-P) is combined with the traditional class-object paradigm.

Keywords: meta-model, decomposition of responsibilities, notion, term, modeling language, paradigm

Streszczenie

Artykuł ten skoncentrowany jest na pojęciach metamodelu i wprowadza nową terminologię do dyscypliny metamodelowania. Terminologia ta odnosi się do nowej koncepcji Contex-Driven Meta-Modeling (CDMM), bardziej ogólnej niż inne podejścia do definiowania języków modelowania. W efekcie rozszerza ona słownictwo dziedziny metamodelowania. Wszystkie pojęcia zostały wprowadzone na podstawie dekompozycji odpowiedzialności zidentyfikowanych w metamodelach. Dotąd języki modelowania były traktowane wbrew dobrym praktykom projektowym jako zwarte monolityczne struktury niepoddawane żadnej dekompozycji. System pojęć wprowadzonych w artykule jest wystarczająco ogólny do zastosowania w łączeniu paradygmatu Context-Driven Meta-Modeling Paradigm (CDMM-P) z tradycyjnym paradygmatem klasowo-obiektowym.

Słowa kluczowe: metamodel, dekompozycja odpowiedzialności, pojęcie, termin, język modelowania, paradygmat

1. Introduction

In the paper new meta-modeling notions are introduced. In contrast to standard approaches to defining modeling languages managed by Object Management Group (OMG) like MOF, CMOF and EMOF [11], UML [4, 15], BPMN2 [5] or managed by IBM, like eCore, the notions presented below result from the analysis of meta-model responsibilities. All mentioned meta-models are statically defined as compact monolithic structures at compile-time, like in [1–3, 6–9, 12–14]. As the result the introduction of a structural change to such meta-model is difficult. In consequence, this traditional standard-based approach to defining modeling languages significantly slows down the whole Model-Driven Design market. First, a change introduction to a meta-model is a long-lasting process. A good example of this problem is an Aspect-Oriented Programming (AOP) released in 2002 which has not been reflected in the UML standard [15] yet. Second, strangely enough, all standards mentioned above are specified against good design practices introduced and popularized just by the OMG-related community.

The lacking element that helps to introduce or rather identify and differentiate already existing but hidden responsibilities in these standards is the concept of the Context-Driven Meta-Modeling (CDMM) related to the Context-Driven Meta-Modeling Paradigm (CDMM-P) [21]. The paradigm is implemented in the form of the Context-Driven Meta-Modeling Framework (CDMM-F) [17] and supported by some tools [19, 20] to prove the feasibility of the CDMM-related concepts. Also other problems are published [16, 18].

2. CDMM overview

Meta-models in CDMM are composed of classes which are not related at compile-time - there are no references from one class to the other. These classes are divided into classes that should be placed in the meta-model graph nodes and classes that should be placed in the meta-model graph edges. This is different from other approaches as they do not have class representation of relations. The meta-model graph is created at run-time from these classes. Application context plays a special role in this approach [16]. The more extended description of the CDMM is presented in [16, 17, 21].

The CDMM-F has been implemented in Java, Spring and AspectJ technologies. The inversion of control (IoC) architectural pattern and the aspect-oriented concept of injecting default implementation of an interface is crucial in the CDMM-F.

The CDMM-Meta-Modeler [19, 20] is an Eclipse PlugIn for creating CDMM-compliant meta-models. Both cited papers constitute a good introduction to the CDMM concept as they are focused on the graphical modeling of meta-models, so they are illustrative.

3. Meta-modeling notions

There are several reasons for specificity of the meta-modeling terminology:

- nature of the CDMM elements characterized shortly in section 2,
- the decomposition of the meta-model into graph and graph classes responsibilities,
- the possibility of embedding responsibilities in the meta-model graph through the CDMM-P based mechanisms like in [18].

The whole terminology is addressed to meta-model classes and was divided into two main categories:

- node-related classes,
- arc-related classes.

The new terms introduced in the paper according to the assumptions presented above are contained in Tables 1–2. All terms are shortly described and named. The system of stereotypes is also introduced in Tables 1–2 for later use in meta-models and succeeding papers. Some stereotypes are abstract (they cannot be used in meta-models) – printed in italics and some are concrete - printed in bold in Tables 1–2.

It is worthy of notice that there is only one responsibility integrated to the meta-model class as long as the CDMM paradigm is taken into account: entity - a class responsible for storing and giving access to data and some responsibilities are taken dynamically from the role a class plays regarding meta-model. Entity classes are so natural elements of CDMM-compliant meta-models as the CDMM meta-models are equivalent to data layer models applied to meta-modeling as the application domain.

Another classification dimension is connected to the fact that a meta-model graph may be constructed from empty classes just to represent the pure structure. This structure is then a subject of embedding other classes as meta-model responsibilities. So, there are the following terms shown in Tables 1–2:

- skeleton class – an empty class dedicated to being a meta-model graph structure,
- entity class – a class representing data that can be placed in a meta-model graph node or edge; other classes that is classifiers or domain classes are not allowed to be placed in the meta-model graph structure as they influence negatively the Application Programming Interface (API) dedicated to traversing meta-model graph.

The same classification of meta-model responsibilities is shown in the Figure 1 in the form of the UML class diagram, but it is limited there to stereotypes grouped into packages. There is also a coloring convention introduced in the Figure 1: green classes are specific to the nodes and blue classes are specific to the arcs. It is worth noticing here that there may be however also some classes of responsibilities embedded to both node classes and edge classes of the meta-model graph.

Table 1. Node-related meta-model classes

Notion description	Names			Range	
	full	abbreviated	stereotype	name	stereotype
A class in a meta-model graph node	meta-model graph node class	node class	<<n>>	Meta-model graph node classes	<<n>>
A meta-model entity class embedded in a meta-model graph node class	meta-model graph node entity class	node entity class	<<ne>>		
An empty meta-model graph node class	meta-model graph skeleton node class	skeleton node class	<<sn>>		
A non-empty meta-model graph node class	meta-model graph entity node class	entity node class	<<en>>		
A structural responsibility class embedded in a meta-model graph node class	meta-model node responsibility class	node responsibility class	<<nr>>	Meta-model node responsibility classes	<<nr>>
A meta-model entity class embedded in an empty meta-model graph node class	meta-model skeleton node entity class	skeleton node entity class	<<sne>>		
A meta-model domain class embedded in a non-empty meta-model graph node class	meta-model entity node entity class	entity node entity class	<<ene>>		

Table 2. Arc-related meta-model classes

Notion description	Names			Range	
	full	abbreviated	stereotype	name	stereotype
A class in a meta-model graph node	meta-model graph arc class	arc class	<<a>>	Meta-model graph arc classes	<<a>>
A meta-model entity class embedded in a meta-model graph node class	meta-model graph arc entity class	arc entity class	<<ae>>		
An empty meta-model graph node class	meta-model graph skeleton arc class	skeleton arc class	<<sa>>		
A non-empty meta-model graph node class	meta-model graph entity arc class	entity arc class	<<ea>>		
A structural responsibility class embedded in a meta-model graph node class	meta-model arc resposnibility class	arc responsibility class	<<ar>>	Meta-model arc responsibility classes	<<ar>>
A meta-model entity class embedded in an empty meta-model graph node class	meta-model skeleton arc entity class	skeleton arc entity class	<<sae>>		
A meta-model domain class embedded in a non-empty meta-model graph node class	meta-model entity arc entity class	entity arc entity class	<<eae>>		

4. Illustration

In order to show how concrete stereotypes shown in the Figure 1 can be applied in meta-models for the meta-model elements classification purposes a diagram was presented in Figure 2. This figure helps to explain why such terms like the ones presented in Tables 1-2 and in the Figure 1 were introduced in the paper.

There are some conventions in Figure 2:

- CDMM-compliant relationships are represented in the form of dotted line directed graph edges,
- meta-model graph node classes are depicted in green while their responsibilities are displayed in grey,
- there is only one layer of responsibilities taken into account in the examples.

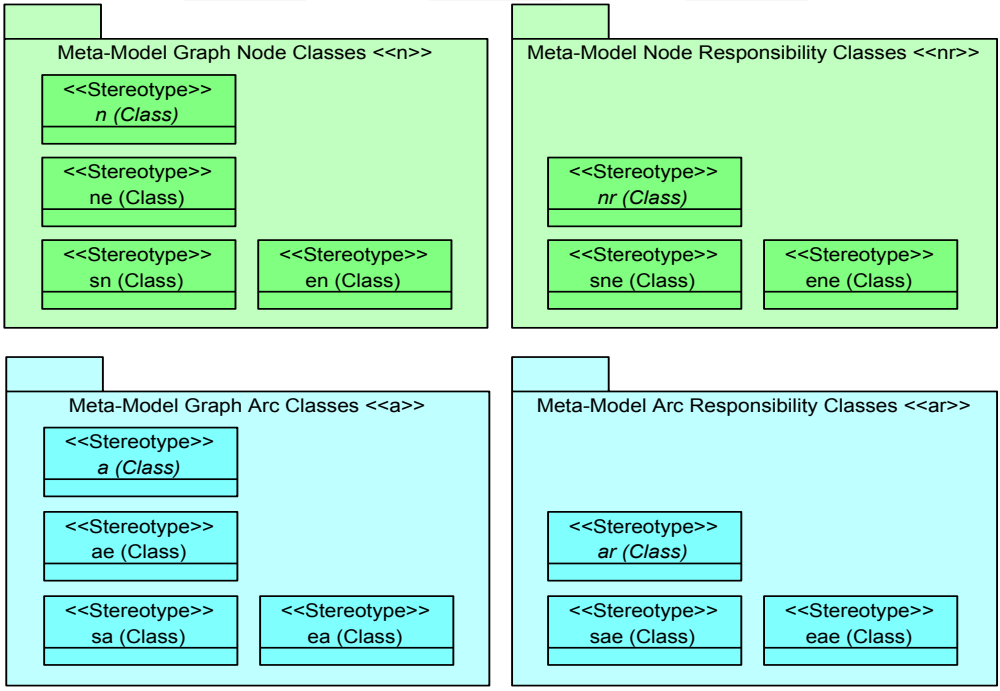

Fig. 1. UML diagram of CDMM meta-model notions in the form of stereotypes classified into packages

Only the direct neighbors of meta-model graphs were taken into account in Figure 2. But the responsibilities added to the meta-model graph may take the form of complex and deep hierarchies. The hierarchies may constitute layers of responsibilities.

The example shown in the Figure 2 is focused on the entity classes. The CDMM-compliant meta-model presented in the figure can be created at run-time and, in the consequence, it could be easily changed. There are no arc classes shown in the diagram (lack of stereotypes associated to the graph arcs), nevertheless they must be used to interrelate node classes.

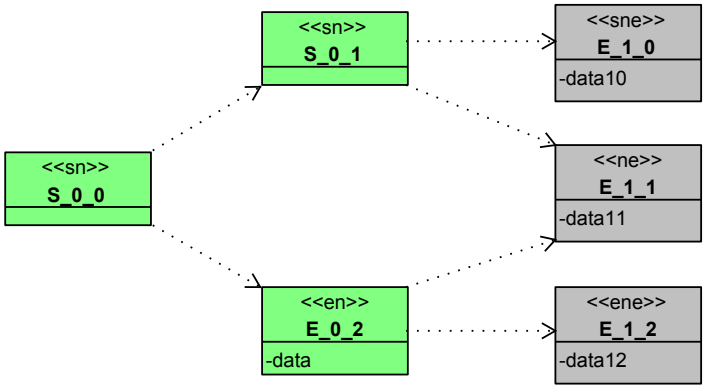

Fig. 2. Node entity classes <<ne>>, <sne>>, <<ene>> embedded in the meta-model graph node classes <<sn>>, <<en>> via CDMM arc class <<a>>

A sample CDMM compliant meta-model graph is presented in the Figure 2. It conforms to the convention: graph nodes in green and additional structural responsibilities in gray. Some stereotypes from the Table 1 and from the Figure 1 are applied in the meta-model graph to specify the role of the graph elements they play regarding the meta-model graph. There are two nodes consisting of empty classes, which are stereotyped by <<sn>> and there is one node class being an entity class, which is stereotyped by <<en>>. There are also entity classes which perform different roles in relation to the meta-model nodes. The class stereotyped by <<sne>> is an entity class which is associated at run-time to an empty node class (stereotyped by <<sn>>). The <<sne>> class is named *skeleton node entity class*. The <<ene>> stereotype is applied to the entity class which is used to introduce the entity hierarchy to the entity class being already a node class (stereotyped by <<en>>). The <<ene>> class is named *entity node entity class*. The last class, the one stereotyped by <<ne>>, is an entity class which is reused by two node classes of different types (roles they play in the current meta-model, which is a current context for these classes regarding this particular meta-model). The <<ne>> entity class is applied to <<sn>> and <<en>> classes and that is why just the <<ne>> stereotype is placed in this class - the stereotype without "s" or "e" prefix. This <<ne>> is named just *node entity class*.

5. Conclusions

The consistent system of notions and the terminology for meta-model elements were introduced in the paper. They were presented in the form of tables, stereotypes and an illustration of application. The motivation for introducing these notions was the clarification of terms used in already published papers. It is worth noticing that notions introduced in the paper can be generalized.

Such the terminology is important for further publications. It introduces the clear and systematic system of notions for referencing in papers. Another important role of the terminology is to define the basis for comparative study between the CDMM-compliant meta-models and other modeling languages. The existence of the terminology also constitutes

the necessary condition for transformations between different meta-models, forming the first step to the CDMM interoperability. The generalization of notions may be useful in papers focused on transformations between different modeling languages. Research on the above mentioned subjects are advanced and the results will be presented in next papers.

References

[1] Akehurst D., Howells G., McDonald-Maier K., *Implementing associations: UML 2.0 to Java 5*, Softw Syst Model, Springer-Verlag 2006, DOI 10.1007/s10270-006-0020-1.

[2] Bildhauer D., *On the relationship between subsetting, redefinition and association specialization,* [in:] Proc. of the 9th Baltic Conference on Databases and Information Systems 2010, Riga, Latvia 07/2010.

[3] Bildhauer D., *Associations as First-class Elements*, Proceedings of the 2011 conference on Databases and Information Systems VI: Selected Papers from the Ninth International Baltic Conference, DB&IS 2010, p. 108–121, IOS Press Amsterdam, The Netherlands, The Netherlands 2011.

[4] Booch G., Rumbaugh J., Jacobson I., *The Unified Modeling Language User Guide*, Addison-Wesley, 2005.

[5] Object Management Group (2011), Business Process Model and Notation 2.0, http://www.omg.org/spec/BPMN/2.0 (access: 5.05.2017).

[6] Diskin Z, Easterbrook S., Dingel J., *Engineering Associations: From Models to Code and Back through Semantics*, In: Objects, Components, Models and Patterns, Volume 11, Lecture Notes in Business Information Processing, Proceedings of 46th International Conference, TOOLS EUROPE 2008, Zurich, Switzerland, June 30–July 4, 2008, p. 336–355.

[7] Feinerer I., *A Formal Treatment of UML Class Diagrams as an Efficient Method for Configuration Management*, PhD. dissertation, Vienna, March 2007.

[8] Feinerer I., Salzer G., Software & Systems Modeling, 13(3), 2014, p. 1167–1187.

[9] Génova G., Ruiz del Castillo C., Llorens J., *Mapping UML Associations into Java Code*, Journal of Object Technology, Vol. 2, No. 5, September–October 2003.

[10] Kleppe A.G., Warmer J., Bast W., *MDA Explained: The Model Driven Architecture: Practice and Promise*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[11] Object Management Group (2006), Meta Object Facility (MOF) core specification version 2.0, http://www.omg.org/spec/MOF/2.0 (access: 5.05.2017).

[12] Roques P., *SysML vs. UML 2: A Detailed Comparison*, MoDELS'11 Tutorial, October 16th, Wellington, New Zealand 2011.

[13] Szlenk M., *Formal Semantics and Reasoning about UML Class Diagram*, 2006 International Conference on Dependability of Computer Systems, IEEE, 25–27 May 2006, p. 51–59, DOI: 10.1109/DEPCOS-RELCOMEX.2006.27.

[14] Tan H.B.K., Yang Y., Bian L., *Improving the Use of Multiplicity in UML Association*, Journal of Object Technology, Vol. 5, No. 6, July–August 2006.

[15] Object Management Group (2009), Unified Modeling Language (UML) superstructure version 2.2, http://www.omg.org/spec/UML/2.2 (access: 5.05.2017).
[16] Zabawa P., *Context-Driven Meta-Modeling Framework (CDMM-F) - Context Role*, Technical Transactions 1-NP/2015, p. 105–114, DOI: 10.4467/2353737XCT.15.119.4156
[17] Zabawa P., *Context-Driven Meta-Modeling Framework (CDMM-F) – Internal Structure*, 2016, submitted for publication.
[18] Zabawa P., *NamedElement Revisited in Aspect-Oriented Approach*, Technical Transactions, 1-NP/2016, p. 17–27.
[19] Zabawa P., Fitrzyk G., *Eclipse Modeling Plugin for Context-Driven Meta-Modeling (CDMM)-Meta-Modeler*, Technical Transactions, 1-NP/2015, p. 115–125.
[20] Zabawa P., Fitrzyk G., Nowak K., *Context-Driven Meta-Modeler (CDMM)-Meta-Modeler Application Case-Study*, Information Systems in Management, 5(1), 2016, p. 144–158.
[21] Zabawa P., Stanuszek M., *Characteristics of Context-Driven Meta-Modeling Paradigm (CDMM-P)*, Technical Transactions, 3-NP/2014, p. 123–134.