INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2016) Vol. 5 (4) 593−604

# SOFTWARE ENGINEERING STANDARDIZATION SHIFT FOR ENTERPRISE SYSTEMS DEVELOPMENT PROCESS

PIOTR ZABAWA

*Department of Physics Mathematics and Computer Science, Institute of Computer Science, Cracow University of Technology*

The paper is focused on the processes of software development of enterprise systems. It is related to the new concept of software development paradigm named Context-Driven Meta-Modeling Paradigm (CDMM-P) introduced by the author. The CDMM-P can be applied to define modeling or meta-modeling languages, to construct enterprise systems data layer. The CDMM-P concept is based on application open ontologies in the form composed of notions characteristic for software engineering and it constitutes the first implementation and the first application of open ontologies in software engineering domain. The paper presents the concept of a shift of existing OMG standardization approach. It explains why the CDMM-P graph representation and its API should be the subject of standardization in place of MOF-based close ontology structures.

Keywords: software engineering, software development process automation, modeling, meta-modeling, UML, MOF, MDA, application context

## 1. Introduction

There is one well known and commonly accepted standardization approach to software engineering domain managed by Object Management Group (OMG) [9, 10]. However, this approach has several important limits. This paper introduces new, more general approach to standardization in this domain. The approach proposed in the paper is based on the new software engineering paradigm named Con-

text-Driven Meta-Modeling Paradigm (CDMM-P) [24] and thus constitutes new alternative approach to software engineering domain, according to author's best knowledge and according to [8]. From the careful literature analysis contained in [24] it results that both traditional approaches to software engineering [14–16, 19, 20] and works related to ontology and its RDF and OWL standards [1, 3–7, 11–13, 18] differ significantly from the paper's research thread, as they are built on different assumptions. The paper contains a discussion focused on the identification of fields for future standardization efforts, thus – the standardization shift.

The paradigm is attractive from economical point of view and has also many application domains beside modeling and meta-modeling – this aspect is discussed shortly in section 2. The illustrative case study for the CDMM-P was presented in [23] on diagramming abstraction level in Context-Driven Meta-Modeling Meta-Modeler tool [22].

Two standardization paths mentioned above are characterized and revisited in the next subsections.

### 1.1. UML-related standards

At the end of the 90s there were more than 30 modeling languages used for automating software development process and supported by appropriate methodologies. The short state-of-the-art from 1997 (the year of UML standard first publication) in software engineering discipline is shown in Figure 1.

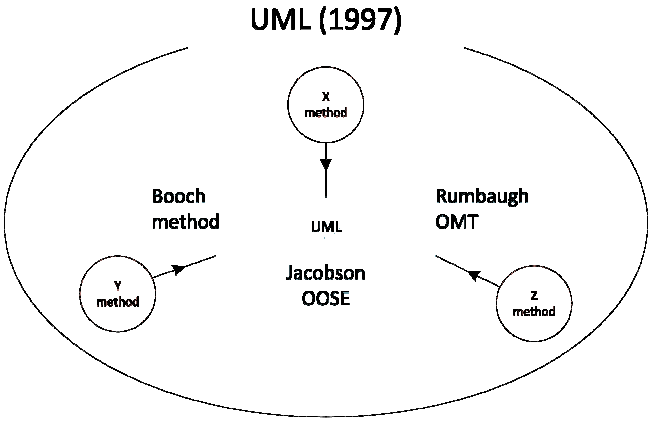


**Figure 1.** Software engineering methodologies in 1997 and UML creation process

In 1996 the process of standardization of software methodologies has been initiated by Booch and Rumbaugh and has been continued with Jacobson later [2, 9, 10, 17]. From the perspective of this paper it is very important to realize how the standardiza-

tion process was initiated. Two essential questions (Q) were asked at the beginning and answered (A) soon:

Q1: How to combine existing modeling languages to obtain a standard modeling language?

A1: UML as the **only** combination of former modeling languages.

Q2: How to define the standard modeling language and other modeling languages?

A2: MOF as the **only** UML abstraction.

So, what was done during standardization process? Best common three parts of modeling languages were combined with several other modeling language best concepts according to Figure 1. After that the abstraction (meta-model) named Meta-Object Facility (MOF) for defining this modeling language was introduced and later applied to other languages.

As the result the IT community obtained **one** standard modeling language (UML) defined in **one** meta-modeling language (MOF).

From ontology point of view all these standards represent close ontologies, that is closed systems of notions. And, in contrast to some knowledge representation techniques, these close ontologies are applied for (meta-)modeling in software engineering discipline. However, (meta-)models also constitute a kind of knowledge.

Close ontology based standards already exist but do not work as intended. First two samples illustrating the nature of the limit introduced into the whole approach to standardization (close ontology) are AspectJ (2002) and Scala trait relationship (2003). Notions introduced by Aspect-Oriented Programming (AOP) and the notion of Scala trait were not reflected in the discussed standardization thread so far. This conflicts with the Model-Driven Architecture (MDA) concept which is oriented on fast accommodation of new technologies. The standardization process of OMG standards is long lasting, which is the consequence of the assumption of close ontologies. The system of notions from the OMG standards is difficult to change, like each systems of notions designed as closed and to remain closed. Moreover, the standardization process tends to be longer year by year due to the number of new concepts (eg. more than 120 programming languages for JVM in 2014) and due to the growing size of the UML.

### 1.2. CDMM-related standards

In contrast to the official standardization process and in spite of its huge added value, the alternative standardization thread was initiated by the author in 1993, that is before introduction of close ontology standards characterized above. The research way initiated that time resulted from different questions asked at the beginning.

The first question and answer were:

Q1: How to create a tool for modeling in any (existing or not) modeling language?

A1: Through application of the new paradigm.

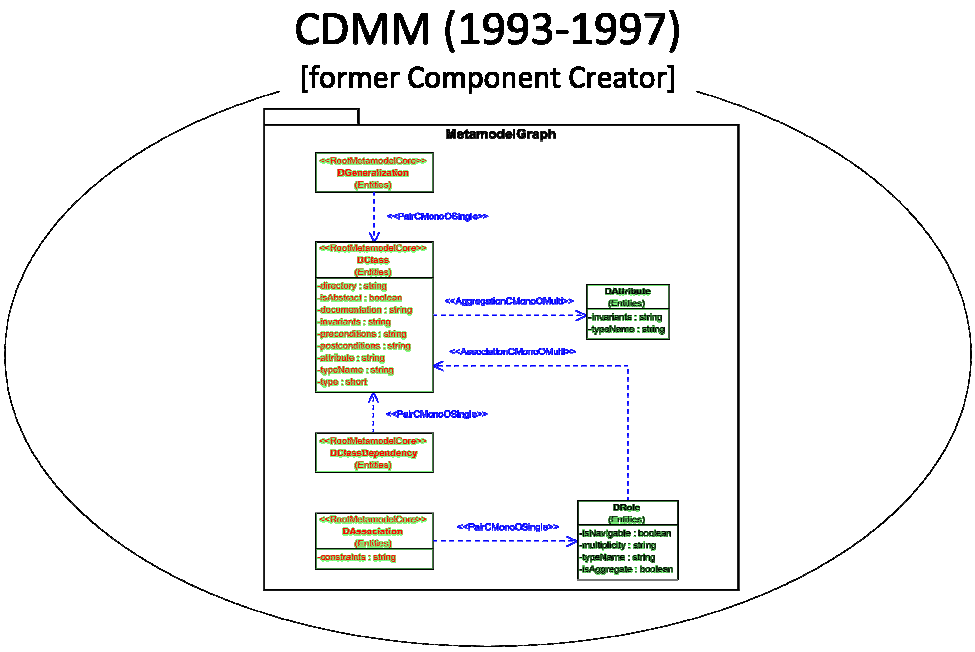


**Figure 2.** CDMM-P applied to (meta-)modeling in 1997
MetamodelGraph package was published in [23]

The new value was introduced to (meta-)modeling via application of the new paradigm. The paradigm changed the way classes are interrelated as the result of application of the Inversion of Control (IoC) architectural pattern. Moreover, the meta-model classes were divided into classes representing meta-model graph nodes (meta-model entities) and classes representing meta-model graph edges (meta-model relations). In effect, both classes and relationships of the meta-model were decoupled in source code and were dynamically interrelated into meta-model graph at run time.
The next question was asked and answered then:

Q2: Is the abstraction of CDMM itself needed?

A2: No – CDMM can be applied recursively as its abstraction, if needed.

This way, through the application of CDMM-P to meta-modeling, we are able to obtain many modeling languages defined each in one of many meta-modeling languages.

Moreover, the application of CDMM-P resulted in creation of (meta-)modeling languages based on open ontologies.

Now, the question about the status of standardization may be asked, like in case of the approach presented in section 1.1: what is the subject of standardization while open ontologies are applied?

## 2. CDMM-P characteristics from business perspective

Before the standardization shift will be discussed, the value of the CDMM-P is presented. The paradigm is very important from business perspective. One reason is software development process automation, but another one results from the impact to contemporary IT technologies.

The CDMM-P is a paradigm which can be addressed to such software engineering disciplines like:

- design
- programming
- modeling
- testing
- other… .

Another point of view to the CDMM-P shows that the paradigm can be applied to constructing such software system elements like:

- (meta-)modeling languages
- applications data layer and mapping this layer to objects
- graph libraries
- knowledge models
- other… .

In the nearest future the CDMM-P paradigm can influence:

- approaches to constructing Virtual Machines (JVM, .Net)
- data mapping technologies
- programming languages
- other… .

It is worth noticing, that CDMM-P paradigm main design criterion was to maximize ease of change introduction. This is one of the most demanded properties of software systems – especially enterprise systems. The same criterion should be also applied to (meta-)models construction, but it was not.

As the result of the mentioned criterion, the modeling languages as well as target domain-specific software products have improved Return on investment (ROI) metric due to:

- lowering software development cost

- shortening of time-to-market
- increase of product life-time.

These improvements impacting ROI directly can be achieved via:

- extensive automation of software development process
- simplification of enterprise systems change introduction
- meta-modeling on demand.

Application of CDMM-P should break current limits of close ontology base approaches to (meta-)modeling and enterprise systems data layer implementation.

## 3. Standardization shift subjects

Several questions can be asked regarding the standardization shift. This section is focused on: **What to shift?**

The close ontology based standards were standardization subject so far. However all business advantages mentioned in section 2 can be achieved when open ontologies are applied, according to the conclusion from section 1. That is open ontologies should be promoted in place of their instances (close ontologies). And open ontologies should be the subject of standardization.

However, in order to define precisely what should be the subject for standardization in open ontologies based approaches the specific features of CDMM approach must be known. The most important specific features of CDMM approach which differentiate it significantly from close ontology based approaches presented in section 1.1 are as follows:

- the modeling (not necessarily diagramming) tool is automatically generated from meta-model
- the software project artifacts generator tool is automatically generated from meta-model
- special role of the application context XML file in self-organizing tools [21].

As the result there is no one universal modeling tool for a particular modeling language, say for UML, because there is no one standard modeling language. There is also no one meta-modeling tool for one existing meta-modeling language – MOF.

But, obviously, there must be something “fixed” which helps to make use of this flexible open ontology based approach. This is the Application Programming Interface (API).

So, what are the problems caused by moving standardization focus to the APIs? The most important problems are as follows:

- how to construct tool APIs
- how to scan/query meta-model in order to find all required model elements from the tool’s client source code through the tool’s API.

For the purpose of software artifacts generating the (meta-)model elements must be accessed from generators. The following two access modes should be provided by the API:

- scanning/traversing meta-model graph
- querying meta-model graph.

Both functionalities should be fundamental for model-driven software project artifacts generating tools. The second access mode is not handled by contemporary tools – this results from different properties of close and open ontology based approaches.

## 4. Standardization shift strategy

The next question, which is discussed in this section, is: **How to shift?** Possible answers to this question are collected and shortly analyzed from the benefits perspective. The discussion is focused on the "standardization decision tree" presented in Figure 3.

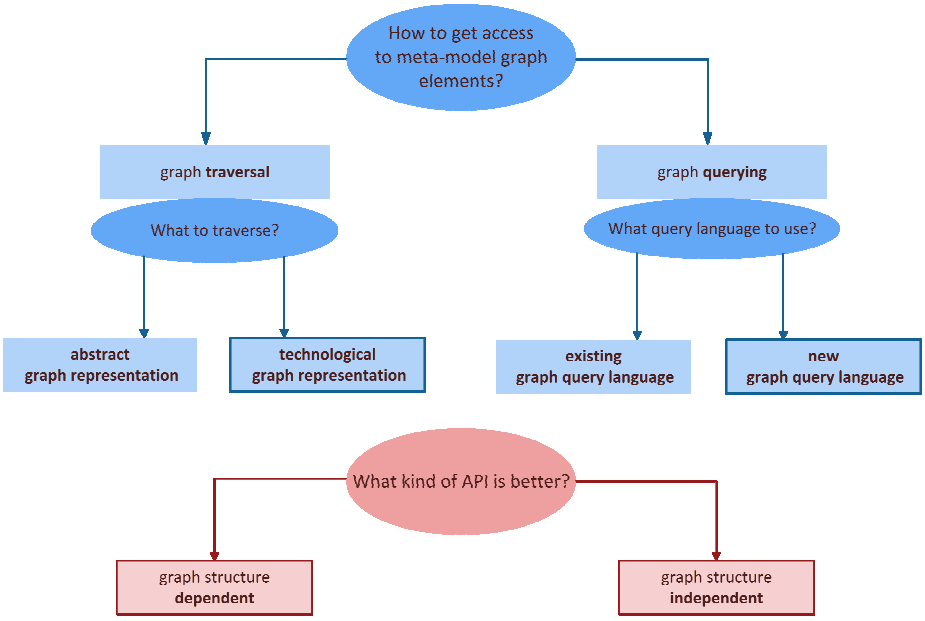


**Figure 3.** Standardization decision tree- author's suggestions in frames

The decision tree from Figure 3 is composed of two parts. The blue part is focused on possible ways of accessing meta-model graph elements. The red part is concentrated on the availability of meta-model graph structure information.

Below the character of influence of each decision from the graph presented in Figure 3 on the possible benefits is shortly discussed. The kind of influence can be positive (pros) or negative (cons).

### 4.1. Graph traversal

Graph traversal pros and cons are as follows:

- Pros:
  - the same approach that is known from existing modeling tools can be applied – easier accommodation of the change by designers/developers community
- Cons:
  - no access to a part of the graph – the meta-model graph can be accessed by scanning all its elements

Abstract representation of the graph has the following pros and cons:

- Pros:
  - ease of implementation – any form of graph representation may be chosen
- Cons:
  - ambiguity of representation and APIs – it results from the number of possible graph representations

Another option, not very obvious to the community, is to base the graph implementation on available technologies. This is just what the Context-Driven Met-Modeling Framework (CDMM-F) offers. This concept is named in the paper the “technological representation of the graph”. It also has its own pros and cons:

- Pros:
  - unambiguous representation of the graph supported by technologies
  - no need for implementing graph structure by the (meta-)modeling tool vendor
- Cons:
  - lack of appropriate technologies available

The problem is the lack of technologies that can support this approach. Implementation of such technologies is a huge challenge both from technical and business perspectives. But it offers:

- a chance of development for many existing technologies
- a chance of elaboration of new technologies.

This positive influence on technologies constitutes a kind of added value from the technological representation of the graph approach. This is very innovative and realistic.

### 4.2. Graph querying

The concept of querying meta-model graph can be based on existing graph query languages or may trigger elaboration of new ones.
Existing graph query languages pros and cons are as follows:

- Pros:
  - reusability – the software available on the market may be reused
- Cons:
  - lacking support for multidimensional problems characteristic for CDMM

In the case of elaborating new graph query language(s) the following pros and cons may be observed:

- Pros:
  - support for multidimensional problems characteristic for CDMM – this problem may be addressed if this option is chosen as contemporary graph query languages assume flat graph structure
- Cons:
  - lack of such languages, lack of base technologies for such languages

This is again a big challenge to design such concepts and supporting technologies.

### 4.3. Graph structure dependent API

In order to traverse or query the meta-model graph from the API client's source code the developer may assume that the graph structure is known and is not subject of change. This assumption is correct in case of close ontologies. However, in the case of open ontologies the meta-model structure is assumed to be subject of relatively frequent changes and the same is about structure-dependent API.
Graph structure dependent API pros and cons are as follows:

- Pros:
  - API for traversing graph structure is generated automatically
- Cons:
  - the graph structure must be known by API's client

If the graph structure-dependent API is applied, then as many APIs as (meta-) modeling languages are obtained.

### 4.4. Graph structure independent API

The problems mentioned in section 4.3 can be eliminated if the meta-model API is fixed. It means that the API must be structure-independent. According to currently available technologies it should be implemented in reflexive approach (ex. Java reflection) or should be based on meta-programming (ex. Groovy meta-programming).

Graph structure independent API pros and cons are as follows:

- Pros:
  - the graph structure may be unknown by API's client
- Cons:
  - unknown (none?)

As the result of this approach one universal API for all (meta-)modeling languages is obtained. This way the old problem of porting a client code from one modeling tool to the other is solved because having one standard API, the tools may follow this standard and still they can compete in the form of extending the standard.

## 5. Conclusion

In the paper it is shown that application of open ontologies to software engineering domain may bring progress in several IT branches, among the others in modeling (creating model for a particular project in a modeling language), in metamodeling (defining modeling languages), in technologies related to enterprise systems data layer, in virtual machines, programming languages and in software development processes. It is also very attractive from economical point of view.

Some current problems and technological limits were also identified. The problems are driving the development and their solutions are intended to be published in succeeding papers.

So, what are the benefits resulting from application of open ontology based approach(es) to software engineering domain? The fixed modeling languages or metalanguages with tool-specific APIs are exchanged by user-defined languages with standardized API. As the result, tool vendors competition fields will migrate from their traditional positions to the new ones.

It is worth noticing, that, generally speaking, the proposed standardization shift is a shift of current compromises. This kind of "renegotiation" is typical for engineering industry domain. In the paper the ease of (meta-)modeling tools implementing is sacrificed to ease of introducing or customizing modeling languages. In consequence, new (meta-)modeling tools are much more complex than before. On the other hand, the (meta-)modeling process is significantly simplified and well known barriers in modeling domain are broken.

There are several challenges for future research in software engineering resulting from the proposed standardization shift. They are connected to such problems like multidimensionality, new programming languages, new virtual machines, mapping new software engineering notions to all MDA notions. But the most difficult challenge is to implement the approach in enterprises.

***REFERENCES***


[1] Aßmann U., Zschaler S., Wagner G. (2006) *Ontologies, meta-models, and the model-driven paradigm.* In C Calero, F Ruiz, and M Piattini, editors, Ontologies for Software Engineering and Software Technology, pages 249–273. Springer.

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

[3] Calero C., Ruiz F., Piattini M. (2006) *Ontologies for Software Engineering and Software Technology*. Springer.

[4] Djurić D., Devedžić V. (2010) *Magic Potion: Incorporating New Development Paradigms through Meta-Programming*. IEEE Softw., 27(5):38–44.

[5] Djurić D., Jovanović J., Devedžić V., Šendelj R. (2010) *Modeling Ontologies as Executable Domain Specific Languages.* presented at the 3rd Indian Software Eng. Conf.

[6] Gašević D., Djurić D., Devedžić V. (2009) *Model Driven Engineering and Ontology Development*. Springer-Verlag.

[7] Gašević D., Kaviani K., Milanović M. (2009) *Ontologies, software engineering*. In Handbook on Ontologies. Springer-Verlag.

[8] Goczyła K. (2011) *Ontologies in Information Systems* (in Polish). Akademicka Oficyna Wydawnicza EXIT.

[9] Object Management Group (2015) *Meta object facility (mof) core specification version 2.0*. URL: http://www.omg.org/spec/MOF/2.0.

[10] Object Management Group (2015) *Unified modeling language (uml) superstructure version 2.2*. URL: http://www.omg.org/spec/UML/2.2.

[11] Guizzardi G. (2005) *Ontological foundations for structural conceptual models.* Telematica Instituut Fundamental Research Series, 15.

[12] Guizzardi G. (2007) *On ontology, ontologies, conceptualizations, modeling languages, and (meta)models*. In Frontiers in Artificial Intelligence and Applications Volume 155, pages 18–39, Amsterdam. Conference on Databases and Information Systems IV, IOS Press. Selected Papers from the Seventh International Baltic Conference DB and IS 2006.

[13] Holanda O., Isotani S., Bittencourt I., Elias E, Tenório T. (2013) *Joint: Java ontology integrated toolkit*. Expert Systems with Applications, 40:6469–6477.

[14] Kern H., Kühne S. (2007) *Model interchange between aris and eclipse emf*. In 7th OOPSLA Workshop on Domain-Specific Modeling, Montreal.

[15] Krahn H., Rumpe B., Völkel S. (2007) *Efficient editor generation for compositional dsls in eclipse*. In Proceedings of the 7th OOPSLA Workshop on Domain-Specific Modeling DSM' 07, Jyväskylä University, Jyväskylä, 2007. Technical Report TR-38.

[16] Kalnins A., Vilitis O., Celms E., Kalnina E., Sostaks A., Barzdins J. (2007) *Building tools by model transformations in eclipse*. In Proceedings of DSM 2007 workshop of OOPSLA 2007, pages 194–207, Montreal, University Printing House.

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

[18] Peng X., Zhao W., Xue Y., Wu Y. (2006) *Ontology-based feature modeling and application-oriented tailoring*. In Reuse of Off-the-Shelf Components, pages 87–100. Springer-Verlag, New York.

[19] Sprinkle J., Mernik M., Tolvanen J.-P., Spinellis D. (2009) *What kinds of nails need a domain-specific hammer?* IEEE Software, 26:15–18. Guest Editors' Introduction: Domain Specific Modelling.

[20] Silingas D., Vitiutinas R., Armonas A., Nemuraite L. (2009) *Domain-specific modeling environment based on uml profiles*. In Information Technologies 2009: Proceedings of the 15th Conference on Information and Software Technologies, IT 2009, pages 167–177, Kaunas. Kaunas University of Technology.

[21] Zabawa P. (2015) *Context-Driven Meta-Modeling Framework (CDMM-F) - Context Role*. Technical Transactions of Cracow University of Technology, 112(1-NP), pages 105-114.

[22] Zabawa P., Fitrzyk G. (2015) *Eclipse Modeling Plugin for Context-Driven Meta-Modeling (CDMM-Meta-Modeler)*. Technical Transactions of Cracow University of Technology, 112(1-NP), pages 115-125.

[23] Zabawa P., Fitrzyk G., Nowak K. (2015) *Context-Driven Meta-Modeler (CDMM-Meta-Modeler) Application Case-Study*. Information Systems in Management, accepted for publication.

[24] Zabawa P., Stanuszek M. (2014) *Characteristics of the Context-Driven Meta-Modeling Paradigm (CDMM-P)*. Technical Transactions of Cracow University of Technology, 111(3-NP), pages 123–134.