How to Choose a Metamodeling Approach

Arnon Sturm

Department of Information Systems Engineering, Ben-Gurion University of the Negev
Beer-Sheva, Israel
sturm@bgu.ac.il

Abstract. Nowadays as the trend of model driven development is evolved, the use of metamodeling has gain massive attention. Multiple metamodeling approaches have been developed for various usages. However, neither comparison nor an evaluation of these approaches has been suggested. In this paper, we propose a framework for comparing among metamodeling approaches.

Keywords: metamodel, comparison, evaluation

1 Introduction

Metamodeling refers to the analysis, construction and development of rules, constraints, and models for the purpose of modeling a predefined class of problems. Mostly, metamodeling is used for defining modeling languages. However, it is also used for defining domain specific languages, for transforming artifacts among various languages, for interoperability among technologies, and for representing ontologies. Over the last decade, metamodeling has become a main concern for both the academic and the industrial communities. This has led to the development of many approaches for metamodeling. Among the various metamodeling approaches, one can find the Meta Object Facility [1] that was initiated by Object Management Group (OMG)[1] and is implemented within the Eclipse framework[2], namely Ecore, the GOPPRR approach which was initiated by [2] and is implemented within a commercial tool of MetaEdit[3], the Meta Programming System (MPS)[4] [3], Gmodel [5], and MetaL [4]. Although many metamodeling approaches exist, no comparing framework has been proposed.

The purpose of this paper is to stimuli the discussion about the need to compare metamodeling approaches and to set the basis for such a framework. In the following existing comparisons are presented and discussed, followed by a proposal for a

[1] http://www.omg.org/
[2] http://www.eclipse.org/modeling/emf/
[3] http://www.metacase.com/
[4] http://www.jetbrains.com/mps/

comparison framework for metamodeling techniques. The paper concludes with a short summary and plans for future research.

2 Related Work

Metamodeling approaches exist for many years. However, to the best of our knowledge a comparison or an evaluation of these approaches is missing. Yet, there are a few works that compare a proposed metamodeling approach to others. For example, in [6], the author compared the Eclipse EMF/GEF and MetaEdit for the purpose of developing a DSL in terms of productivity based on experimental settings. Another example appears in [4], in which the authors criticize existing metamodeling languages with the following: (1) lacking in defining links between semantically related constructs in different metamodels; (2) limited support on multiple levels of the metamodel; and (3) rigid separation of metamodel levels.

3 The Comparison Framework

To allow a comparison among various metamodeling approaches we gathered a set of criteria from various sources, such as object-oriented techniques comparison [7], the various work in the area of metamodeling, and various comparison and evaluation frameworks. We adjust the selected criteria to be fitted for the purpose of comparing metamodeling approaches. In the following we present the set of criteria along with their classification.

General

This class of criteria refers to the general purpose and description of the metamodeling approach.

1. **Purpose**: The purpose of the specific metamodeling approach: enabling transformation, constructing CASE tools, supporting MD*, etc.
2. **Lifecycle coverage**: What are the lifecycle stages supported by the metamodeling approach and associated tool? Enforcing analysis and design? Enable code generation? Etc.

Modeling Technique

This class of criteria refers to the technique used for defining the metamodels.

1. **Accessibility**: The ease of use and understanding the metamodeling approach.
2. **Expressiveness**: The capability of specifying:
   a. The main concepts.
   b. Relationships among concepts: association, reference, aggregation, generalization-specialization, instantiation.
   c. Relationships among models: refinement, reference.

   d. Constraints

3. **Multiple-level support**: The number of levels supported by the specific metamodeling approach, and the relationships among the levels supported by the approach (instantiation, reference, uni- or bi-directional association, generalization-specialization).
4. **Complexity management**: The ability to deal with various levels of abstraction (i.e., various levels of detail). Sometimes, high-level description is needed, while in other situations, more detail is required.
5. **Analyzability:** The capability to check the internal consistency or implications of metamodels.

Pragmatics

Pragmatics refers to dealing with practical aspects of deploying and using a metamodeling approach. In particular, the framework suggests examining the following issues:

1. **Resources**: What resources are available in order to support the metamodeling approach? Is a textbook available? Are users' groups established? Is training and consulting offered by the vendor and/or third parties? In addition, are automated tools (CASE tools) available in support of the approach (e.g., graphical editors, code generators, and checkers)?
2. **Required expertise**: What is the required background of those learning the metamodeling approach? Does the approach assume knowledge in some discipline?

4 Summary

In this paper we propose a framework for comparing metamodeling approaches. We aim at stimuli the discussion of what the relevant set of criteria for that framework is. Future work refers to extending and refining the framework and applying it to various metamodeling approaches for enabling the selection of the appropriate approach for specific tasks and for reasoning about the missing gaps in these approaches.

References

1. OMG, Meta Object Facility, http://www.omg.org/mof/, 2009.
2. Kelly, S., Towards a Comprehensive MetaCASE and CAME Environment: Conceptual, Architectural, Functional, and Usability Advances in MetaEdit+, Phd Dissertation, University of Jyväskylä, 1997.
3. Demitriev, S., Language Oriented Programming: The Next Programming Paradigm, JetBrain, white paper, 2004.
4. Englebert, V. and Heymans, P., Towards more extensible metaCASE tools. In A.L. Opdhal J. Krogstie and G. Sindre, editors, International Conference on Advanced Information Systems Engineering (CAiSE'07), LNCS 4495, 454-468, 2007.

5. Bettin, J. and Clark T., GModel – A Language for Modular Meta Modeling, KISS Workshop at ASWEC, 2009.
6. Kelly, S., Comparison of eclipse EMF/GEF and metaEdit+ for dsm. In Workshop on Best Practices for Model Driven Software Development, OOPSLA, 2004.
7. The Object Agency, A Comparison of Object-Oriented Development Methodologies, Technical report, http://www.toa.com/smnn?mcr.html, 1995.