Principles, Standards and Tools for Model Engineering

Jean Bézivin, Frédéric Jouault, David Touzet
ATLAS Group (LINA & INRIA)
name.surname@univ-nantes.fr

Abstract

We take here a broad view of model engineering as encompassing different approaches such as the OMG MDA™ proposal [9], the Microsoft Software Factories view [5], and many others. We distinguish the three levels of principles, standards and tools to facilitate the discussion. We propose the idea that there may exist a common set of principles that could be mapped to different implementation contexts through the help of common standards. We illustrate our claim with AMMA, a lightweight architectural style for a model-engineering platform that is currently mapped onto the Eclipse Modeling Framework [4].

1. Introduction

The IBM manifesto [3] makes the claim that MDA-based approaches are founded on three ideas: Direct representation, Automation and Standards. Direct representation allows a more direct coupling of problems to solutions with the help of Domains Specific Languages (DSLs). Automations means that the facets represented in these DSLs are intended to be processed by computer-based tools to bridge the semantic gap between domain concepts and implementation technologies. This should be complemented by the use of open standards that will allow technical solutions to interoperate. These three ideas are, in our opinion, key to the development of MDE (Model-Driven Engineering) approaches. However, to make it practical, we need to extend it in two directions. First, we need to build MDE on a sound set of principles. Next, we need to implement MDE on practical, widely used platforms.

This paper proposes a conceptual architecture intended to bridge the gap between principles and implementation. In the second section, we introduce a possible foundation for model engineering. We then proceed, in the third section, to demonstrate how some of these principles may be mapped onto an operational system. We discuss advantages, limitations and possible extensions of our approach in the conclusion.

2. Definitions and concerns

Models are now commonly used to provide representation of real-world situations: a model is said to *represent* a system. Each model is defined in *conformance* to a metamodel. Each metamodel may be associated with a given DSL. It describes the various kinds of contained model elements, and the way they are arranged, related, and constrained. Representation and conformance relations are central to model engineering [2].

As models, metamodels are also composed of elements. The metamodel elements provide a typing scheme for model elements. This typing is expressed by a *meta* relation between a model element and its metaelement (from the metamodel). A model M conforms to a metamodel MM if and only if each model element has its metaelement defined in MM.

The growing number of existing DSLs has emphasized the need for an integration framework for all available metamodels. This integration is achieved by introducing a metametamodel dedicated to the definition of metamodels. In the same way that models are defined in conformance with their metamodel, metamodels are defined by means of the metametamodel language. Metamodels conform to the metametamodel. As an example, we can consider the MOF (Meta-Object Facility), which is the OMG proposal for metamodels definition [7]. As both models and metamodels, the metametamodel is composed of elements. As a consequence, a metamodel conforms to the metametamodel if and only if each of its elements has its metaelement defined in the metametamodel.

Within the MDE area, artifacts of the model-driven architecture are considered as first class entities. Suitable tools are hence required in order to conveniently handle models and metamodels. We describe, in the next section, the set of features provided by AMMA, our MDE architectural style.

3. AMMA: a model engineering platform

AMMA (Atlas Model Management Architecture) is our current proposal for a model engineering platform. We first present an overview of the platform before describing its mapping to the Eclipse/EMF framework.

3.1. Platform overview

AMMA is based on four blocks providing a large set of model processing facilities:

- The Atlas Transformation Language (ATL) is a model transformation language [8] having its abstract syntax defined using a metamodel. An ATL transformation accepts a source model M_a (conforming to a metamodel MM_a) and produces a target model M_b (conforming to MM_b) by means of a transformation M_t that conforms to ATL semantics.
- The Atlas Model Weaver (AMW) enables designers to perform model-weaving operations between either models or metamodels. Weaving operations aim to specify the links, and their associated semantics, between elements of source and target models.
- The Atlas MegaModel Management (AM3) is an environment for dealing with models and metamodels, together with tools, services and other global entities, when considered as a whole. The megamodel is some kid of registry recording all available resources (e.g. models, transformations, tools, services, etc.) and associated metadata in a given scope.
- The Atlas Technical Projectors (ATPs) define a set of injectors and extractors, which can be seen as import and export facilities between the model engineering Technical Space (TS) [6] and other TSs (Databases, Flat files, XML, EBNF, etc).

3.2. Implementation

The AMMA platform is implemented on top of the Eclipse/EMF framework [4]. ATL uses the basic features of EMF to handle both models and metamodels. An Integrated Development Environment (IDE) has been developed for ATL on top of Eclipse [1]. Based on EMF, it makes use of many other features, such as the code editor and code debugging frameworks. AMW uses more advanced EMF features: built as a model editor, it benefits from editing domains facilities for complex model handlings (including undo-redo). Eclipse is mostly used as an IDE for software development. As such, it includes facilities enabling to navigate the code, to keep track of the files that need rebuilding, etc. AM3 is used as a truly model-oriented extension of these abilities.

5. Conclusion

There are many variants of model engineering. Our attitude has been to find the set of basic principles common to the entire dominant model engineering approaches and to make them explicit. We are then in a position to clearly separate the principles, standards, and tools levels. The main contribution of this work is the AMMA conceptual architecture, seen as an intermediary level between model engineering basic principles and executable systems running on operational platforms like EMF/Eclipse. The main advantage of proceeding in this way is that we may more clearly evaluate the gap between principles and implementation. One important ongoing work is the implementation of AMMA on a very different target platform, namely the Microsoft Visual Studio Team System. This experiment is intended to assess the generality of the four functional AMMA blocks.

6. References

[1] Allilaire, F., Idrissi, T., "ADT: Eclipse Development Tools for ATL", EWMDA-2, Kent, September 2004.

[2] Bézivin, J., "In search of a Basic Principle for Model Driven Engineering", Novatica/Upgrade, Vol. V, N°2, April 2004, pp. 21-24.

[3] Booch, G., Brown, A., Iyengar, S., Rumbaugh, J., Selic, B. "An IBM Manifesto". MDA Journal, May 2004.

[4] Budinsky, F., Steinnberg, D., Merks, E., Ellersick, R., Grose, T.J., "Eclipse Modeling Framework", EMF, The Eclipse series, ISBN 0-13-142542-0, 2004.

[5] Greenfield, J., Short, K., Cook, S., Kent, S., "Software Factories", Wiley, ISBN 0-471-20284-3, 2004.

[6] Kurtev, I., Bézivin, J., Aksit, M., "Technical Spaces: An Initial Appraisal", CoopIS, DOA'2002 Federated Conferences, Industrial track, Irvine, 2002.

[7] OMG/MOF, "Meta Object Facility (MOF) Specification", OMG Document AD/97-08-14, September 1997.

[8] OMG/RFP/QVT, "MOF 2.0 Query Views Transformations RFP", OMG Document AD/2002-04-10.

[9] Soley, R., and the OMG staff, "Model-Driven Architecture", OMG Document, November 2000.