Multi-Level Platform Descriptions

Colin Atkinson
Software Engineering Group
University of Mannheim
L15, 16
68131 Mannheim
atkinson@informatik.uni-mannheim.de

Thomas Kühne
FG Metamodeling
Darmstadt University of Technology
Hochschulstr. 10
64289 Darmstadt
kuehne@informatik.tu-darmstadt.de

1 Introduction

The MDA approach to software development is based on the notion of automating the mapping of platform independent models into implementation-level platform specific models based on abstract models of platforms [OMG03]. The notions of "platform" and "platform model" are thus central to the whole MDA vision. However, most research on MDA to date has focused on the "transformation" problem [SPG03], and there are few explicit definitions of the notion of "platform" or "platform model" in the MDA literature. Moreover, those that do exist are rather vague and incomplete.

This is a problem, because the input to and output from an MDA tool is directly dependent on the way platforms are used and described. We believe MDA will only realize its full potential if more general notions of the role and nature of platforms and platforms models are adopted. We believe that a more general notion of "platform model" is needed based on a multi-level approach to metamodeling, distinguishing between linguistic and logical metalevels.

2 Describing Platforms

Platforms are typically regarded as supporting the execution of software systems. For example, the MDA guide defines a platform as "a set of subsystems and technologies that provide the capabilities needed to support the execution of a software application." This in turn leads to the idea that PSMs are models of an application that are represented in an executable technology while PIMs are models of an application that are represented in a non-executable technology. However, this distinction is irrelevant for MDA tools. In order to automate mappings between application models it is the nature of the description vehicles used to represent the models which is important, not the issue of whether specific application models are executable. Thus, to support the development of sophisticated MDA tools we need a notion of "platform" which is relevant to the problem which MDA tools are essentially charged with solving–inter-model transformation. Such a notion of platform fully captures the characteristics of the description vehicle used to render an application model. Our litmus test for deciding whether a platform description is complete is the question of whether transformations would be affected if a certain platform feature were changed.

2.1 Language Definition

The most concrete definition of "platform model" available today comes from the school of thought that characterizes MDA in terms of transformations between Domain Specific Languages (DSLs) [CK03, GS03]. According to this school of thought, the essential difference between input and output models in MDA transformations is that they are written in different languages (or language dialects). In other words, the information that has to be input into a model transformation tool to effect the transformation is a description of the languages that the models are written in. Therefore, although it is not stated explicitly, platform models are essentially thought of as being language definitions. However, this approach fails to capture any predefined language usages over and above the core language definition.

2.2 Multi-Level Description

As illustrated in Fig. 1, in addition to a *language* level (L_1) which describes the basic concepts with which applications designed to use the platform can be constructed (e.g., documents the choice of Java) we suggest to also include the level logically located below it, containing language usages (L_0). As elaborated below, a language has many standardized usages which, if altered, would cause a complete platform description to change.

However, it is important to recognize that the language usage level (L_0) is itself structured, i.e. contains a number of metalevels of its own. While the relationship between levels L_1 and L_0 is a linguistic instance-of relationship, the O_x levels within L_0 are related by ontological instantiation [AK03]. The O-levels in particular host *predefined instances* (O_0), *predefined types* (O_1), and possibly even *predefined metatypes* (O_2), etc.

Predefined instances describe the pre-instantiated objects that are found in some environments such as Java's standard I/O streams "in, out, err" or Smalltalk's "true" and "false" instances, but also preexisting system timers, J2EE application containers, etc.

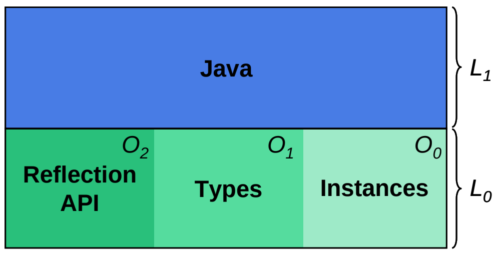

Figure 1 : General Platform Model

Predefined types (e.g., classes) augment the core language capabilities with additional services, typically found in standard libraries. This includes types coming from the language support layer (e.g., Java's "Object" class) but also regular classes, such as Java's "List" class, which can be considered part of a standardized library package.

Unless one includes the above features within a platform description, any dependent model or application will be rendered invalid once any of these features change. Therefore, a complete platform description must span all logical levels ($O_{0..n}$), in addition to a standard description of the L_1 language level.

Note that the distribution of elements to the L- and O-levels is purely based on their logical position within this two-dimensional instantiation framework. Whether an element may be used more frequently than another one, who its creator is, and how likely it may vary when evolving a platform does not play a role in this organization which purely includes language usages (L_0) in addition to language definition (L_1) and structures the former into ontological sublevels.

3 Conclusion

Platforms models play a vital role in the MDA approach but so far have been neglected in terms of how they need to be represented. With respect to the question of whether a platform description is complete we propose a litmus test based on the notion that a platform description must change whenever any associated transformations would be affected by a platform feature change. Hence, we argue that it is necessary to describe platform features at more than just one level. Indeed, this is true for several standard usages of the language which should be organized at several distinct ontological metalevels within the language usage level. Predefined instances, types, and even metatypes, are just as vital to a full platform description as the language itself.

References

[OMG03] OMG. MDA Guide Version 1.0.1, 2003. Version 1.0.1, OMG document omg/03-06-01.
[SPG03] S. Sendall, G. Perrouin, N. Guelfi., O. Biberstein, Supporting Model-to-Model Transformations: The VMT Approach, IEEE Software 2003.
[CK03] S. Cook and S. Kent, "The Tool Factory" OOPSLA2003 Workshop on Generative Techniques in the context of Model Driven Architecture. 2003.
[GS03] J. Greenfield and K. Short, Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools, OOPSLA'03, October 26–30, 2003.
[AK03] C. Atkinson and T. Kühne, Model-Driven Development: A Metamodeling Foundation, IEEE Software, 20 (5), pp. 36—41, September 2003