Model-Based Development - Beyond Model Transformation

Peter Tabeling
Hasso-Plattner-Institute for Software Systems Engineering
P.O. Box 90 04 60, 14440 Potsdam, Germany
tabeling@hpi.uni-potsdam.de

Abstract

The Model Driven Architecture (MDA), as propagated by the Object Management Group, is often considered to be a milestone towards model-based development. This paper argues that MDA should merely be seen as an intermediate phase towards a truly architecture-based approach to software development. Model-based development will only succeed if we improve the communication within development projects. Architectural models must not be seen only as input of generator tools but as the primary means of knowledge transfer and as the basis of the division of labor within a project. In this context, even the object oriented paradigm might be questioned in the future.

1. Near-Term Objectives of MDA

In 2000, the Object Management Group began working on Model Driven Architecture [1][2], and later announced MDA as the new strategy to develop large industrial systems. According to M. Fowler [3], “MDA is a standard approach to using the UML as a programming language”. While this statement arguably simplifies the overall direction of MDA, it surely reflects one of MDA’s major objectives, namely to provide standardized tools which allow UML diagrams to be used as input for (at least partial) code generation.

As a standardization body, the OMG also intends to define a new integration level above CORBA [5], which did not really succeed to integrate middleware platforms such as J2EE [6] or .NET [7] - today, the OMG consideres CORBA to be a “platform” *beside* J2EE and .NET [2]. In a few words, the near-term objectives of MDA can be summarized as (1) *platform integration* and (2) use of *UML as a “programming language”*.

In MDA-related publications [1][2][4], “model transformation” is often presented as the basic approach to achieve these goals. The vision is that software architects or developers use a modeling language, usually UML, to create high-level (“platform independent”) models (PIM) which are refined - mostly by MDA tools - towards lower-level (“platform specific”) models (PSM) and finally executable code.

2. Long-Term Perspectives

It is expected that first generation MDA tools generate only parts of the application code, i.e. manual work by application programmers is still needed. However, the declared goal of MDA proponents is to minimize or completely avoid human assistance in code generation - “In a mature MDA environment, code generation will be substantial or even complete.” [1] The vision goes even further, i.e. not only code generation but also model refinement from PIM-level to PSM-level will eventually become a mostly automated and „uninteresting“ step.[1]

As a consequence, platforms being covered by MDA will increasingly become invisible because they will be superseded by higher-level platforms, namely MDA environments.

3. Implications

In essence, MDA will only repeat and continue what has already been done in the past by introducing assemblers, compilers, virtual machines and middleware platforms - namely raising the level of abstraction for software developers. This achievement will be at least as important as platform integration.

However, MDA addresses mostly the technical side of model-based development. A developer switching from Cobol to Java does not necessarily become a good OO designer - similary, using MDA tools is not sufficient for a transition to truly model-based development. Personnel

1. “As ’Forward Engineering Only’ development processes advance [...] the need for PSMs [...] will tend to decrease, much as the need for 3GL compilers to save intermediate assembly code decreased when 3GL source-level debuggers matured.” [4], p. 242.

involved in requirements engineering, project planning or architectural design must also *think* and *communicate* in high-level models and modeling elements. This applies especially to experts of different fields, like GUI developers, database designers, the project manager, business process modelers and so on. Hence, these experts must not confine themselves to using just domain-specific models. Conceptual architectural models are needed which *integrate* the different views at the high level(s) and foster project-wide communication [8]. Once software projects are organized around such models, model-based development could show its full potential [9]. For example, work packages for teams can better be identified on the basis of components or subsystems of the architecture, instead of feature sets.

In this context, the question about the „right“ modeling concepts arises. At present, this seems to be mostly determined “bottom-up”, i.e. by existing platforms to be integrated and by UML, with its roots in object oriented development. However, since the 90s it has repeatedly been stated - especially by proponents of architecture description languages - that objects and classes are no longer suitable as building blocks for large systems. Instead, „components“ and „connectors“ should be the prevalent modeling and design elements.[1] Advocates of object orientation might argue that inheritance, encapsulation and polymorphism are valuable benefits which should not be sacrificed. This is true, but these features are not necessarily tied to objects and could also be achieved in the context of „architecture oriented methods“, e.g. component types can be defined utilizing inheritance. It seems promising to revive concepts of architecture description languages in the context of model-based development.

While the upcoming UML 2.0 standard is influenced by these ideas, it is still tailored primarily to object oriented languages. In addition, the „design-by-commitee-approach“ of UML has lead to an compilation of (too) many diagram types with subtle interdependencies, making UML too complicated for human communication [11] and giving the users too much choice. A recent study [12] among major software companies indicates that ad-hoc-notations are still preferred over UML when doing architectural design. It seems that UML has made good progress as the basis for tools, but is not well-prepared for the non-technical side of model-based development.

1. “[...] separating components form connectors, raising them both to visibility as top-level abstractions [...] also raises them in the conciousness of the designer.” [10] p. 19.

4. Conclusion

The success of model-based development heavily depends on the implementation of its non-technical aspects. Communication and project managment must be adapted to an architecture-centric approach - otherwise, tools for model-based development will have little impact.

5. References

[1] Richard Soley et. al. *Model Driven Architecture,* Object Management Group White Paper, ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf, November 2000

[2] OMG Architecture Board, *Model Driven Architecture (MDA),* Object Management Group, http://www.omg.org/docs/ormsc/01-07-01.pdf, July 2001

[3] Martin Fowler, *UML Distilled - A Brief Guide to the Standard Object Modeling Language,* 3rd ed., Addison Wesley, 2003

[4] David S. Frankel, *Model Driven Architecture - Applying MDA to Enterprise Computing,* Wiley, 2003

[5] OMG, *The Common Object Request Broker Architecture,* Specification vers. 3.0.2, Object Management Group, December 2002

[6] SUN, *Java 2 Platform - Enterprise Edition,* Specification, vers. 1.4, Sun Microsystems, 2003

[7] Microsoft, *The .NET Framework,* msdn.microsoft.com/net-framework, Microsoft, 2004

[8] Frank Keller, Peter Tabeling et. al. *Improving Knowledge Transfer at the Architectural Level: Concepts and Notations,* International Conference on Software Engineering Research and Practice, Las Vegas, June 2002

[9] Frank Keller, Siegfried Wendt, *FMC: An Approach Towards Architecture-Centric System Development,* IEEE Symposium and Workshop on Engineering of Computer Based Systems, 2003

[10] Nenad Medvidovic, Richard N. Taylor, *A Classification and Comparision Framework for Software Architecture Description Languages,* Dept. of Information and Computer Science, University of California, Irvine, 1997

[11] Joaquin Miller, *What UML Should Be,* Communications of the ACM - article series, vol. 45, no. 11, November 2002

[12] Frank Keller, *Über die Rolle von Architekturbeschreibungen im Software-Entwicklungsprozess,* PhD thesis, Hasso-Plattner-Institute at the University of Potsdam, Germany, 2003