Engineering Software Engineering

by Hans van Thiel

MDA pioneer and author Anneke Kleppe:

"The essence of Model Driven Architecture is its potential for automated model transformation."

Anneke Kleppe has been involved in OMG standardization of the Unified Modeling Language (UML), in particular the Object Constraint Language (OCL), since 1995. She has co-authored several books on UML and OCL and recently one of the world's first books on the Model Driven Architecture (MDA). MDA is a framework, introduced by the OMG, to facilitate software development by utilizing abstract hierarchical models. Eventually, Kleppe predicts, it will become possible to transform generic models into working code by 'just pressing a button'.

Maybe because UML is a graphical language and OCL is not – it's a formal notation for stating object properties - OCL is less known and less widely used in the modeling of software systems. However, it has been part of the UML standard from the beginning and OCL 2.0 has officially been adopted by the OMG in 2003.
Two main contributors to OCL are Dutch software engineers Jos Warmer and Anneke Kleppe, who have jointly published a book about OCL in 1999 and a new one about the 2.0 version in late 2003.
In daily life Anneke Kleppe runs her own company, Klasse Objecten, which specializes in object oriented consultancy, training and design.
Also in 2003, Kleppe and Warmer co-authored - this time also with Wim Bast of Compuware – 'MDA Explained, The Model Driven Architecture: Practice and Promise'.
MDA has been introduced in 2001 by the OMG staff as a framework for software development, using a hierarchy of software models with differing degrees of genericity and specificity. It has been and still is actively promoted by the OMG organization and it has been successfully deployed in several large software development projects.
'MDA Explained' is one of the first books to appear about this new paradigm which, according to Anneke Kleppe, will dramatically change software engineering in the decade to come.

Can you say something about the history and how you became involved in OMG standardization?

Until 1995 I was employed at the research department of Dutch telecoms company KPN, and I figured the future would be in object orientation. So, in 1995 I started my own company, Klasse Objecten. At that time Jos Warmer was working on the first version of the UML standard on behalf of IBM. We cooperated on this and that, and the result was our book on OMT (*in Dutch, HvT*) on the one hand and my contribution to that first UML version on the other. We also wrote a book about UML and its use (*also in Dutch, HvT*). It's little known that we were involved in defining that first UML version but the standard carries our names to prove it. Jos has worked at Klasse Objecten as well, from 1999 till 2002.

Throughout 'MDA Explained' you use a small catering company as an example. The modeling hierarchy consists of one platform independent model (PIM) with three platform specific models (PSM) beneath it. The first PSM is relational, the second is EJB (Enterprise Java Beans) oriented and the third is web based.
The PSMs transform to respectively SQL (System Query Language), EJB and JSP (Java Server Pages) source code. How does this explain MDA?

In MDA you always start with a platform independent model. In the example this is a model which is transformed into a system with a three tier architecture, i.e. three platform specific models and underneath them three 'source code models'. The code of a system or subsystem is essentially also a model. The database communicates with the middle tier and this in turn with the web server, so you have to model these communications in your PSMs as well.
The system for the 'breakfast service company' in the book has been implemented in OptimalJ from Compuware. The tool and the example are available on the Klasse web site.
The promise of MDA is that you can automate such model transformations, including all kinds of variations and intermediate models. Think of tools that can accept parameters for transformations.
MDA is about automating automation and its value lies in such tools for model transformations.

Do you mean automatic transformations between e.g. Java and C++ and vice versa?

No, not in particular, because such transformations are not enough. It's not about Java or C++ or SQL but about different PSMs that are required to work together. It's about a whole of platform specific models that together result in a particular architecture. A platform independent model will have to be transformed into several PSMs. The better tools now are able to do this, partially.
The word 'platform' is often understood to mean 'implementation' but in my view it should be used more in the sense of 'architecture'. So, a PIM could specifically be modeled into a three tier or a five tier architecture, or something else altogether.

UML tool providers often mention 'round trip engineering'. When you look at it more closely, it's usually limited to simple stubs or signatures of operations.

But that's not the tool provider's fault – it stems from the limited expressiveness of the modeling language. The OMG is now working on an executable version of UML, an extension within which you can specify bodies of operations. Furthermore, work has started on QVT (Query, View and Transformation), a language for specifying transformations. Neither will be available in the near future, though.
There is a UML Action Semantics, based on the Action Specification Language designed by Kennedy Carter in the UK. This is also an official OMG standard which allows for executable UML models, but in my view ASL is too close to the hardware to use with MDA.

How about OCL?

The second version of OCL has been greatly extended and it has now become a complete query language. With OCL you can express values in a system and compare them to other values. So you can state pre- and postconditions, initial values, how values are derived from others, the body of query operations and so on. In UML it's often not clear how objects relate to each other and for this OCL is quite useful.
Actually you'd need two OCL extensions to be able to specify actions on a sufficiently high abstraction level and these are assignment and the creation of a new object.

Late 2003 you wrote a plug-in for the open source Eclipse development environment. Is this 'Octopus' open source as well?

Yes, we're taking some time to clean up the code and document it thoroughly, but as of June 2004 it will be open source. Octopus is a tool which supports the new version of OCL. We found that most tools don't support OCL very well and by working in an environment like Eclipse we can promote world wide use of OCL ourselves. Additionally, we're now working on code generation based on OCL expressions.

Getting back to the relation between different MDA models, working code included – how can you ever get from a generic description to a specific one automatically? Isn't that almost a self-contradiction?

Admittedly, every model is specific to some extent. If you choose to use some class in your model, then you do so because this is specifically useful for this system. Even if only the name of the class is known, the model is already specific. But that doesn't mean MDA has no merit. While implementing a UML model, today, programmers have to do a lot of things that are repeated again and again. Like implementing an association, for example. With MDA tools this dull work can be automated.
MDA has just started but I really believe the time will come when you can translate a platform independent model into working code with just one press of a button.

MDA is about models of models and this seems to fit in well with the generic layering structure of the OMG standards. Your book has an entire chapter about those standards.

The OMG standards are indeed similarly structured. With the MOF (Meta Object Facility) you specify UML and CWM (Common Warehouse Meta-model), with UML you specify a UML model and that model specifies the system

you're designing. The four layers, which are denoted M0 to M3 within the OMG, allow for transformations between all OMG specifications.
It may seem complicated, but it's just like a class and an object. As an object is an instance of a class, so a system is an instance of a model. You can regard the model as an instance of the language and the language itself is an instance of the MOF.
If you want to transform a platform specific model in MDA - more accurately a platform specific architecture - into another one, then you can do so through the platform independent model above it.
Such tools don't exist right now, but that's the principle.

How do you see the future of MDA?

In my opinion the significance of MDA is comparable to that of object orientation, or database technology before that, or compiler construction even earlier.
MDA is a real paradigm shift in software development and engineering. Likewise it will, like those others mentioned, take ten years or more to fully reach its potential.

Anneke Kleppe may be reached at a.kleppe@klasse.nl

This article is an adaptation of an interview which appeared in Dutch language magazine Computable, 37 - 6, p 14, 15 (6 February 2004) and is currently archived at http://www.computable.nl/artikels/archief4/d06ms4dq.htm

Hans van Thiel is a technical writer and journalist who contributes regularly to IT magazines in the Benelux. He can be reached at hthiel@compuserve.com