Combining the power of meta-programming and meta-modeling in the OMG/MDA framework.

Jean Bézivin & Nicolas Ploquin

University of Nantes,
2, rue de la Houssinière, BP 92208
44322 Nantes cedex 3, France

As systems become more complex, more evolutive and more open on their environments, the software crisis seems to be worsening. Many end-users are reacting about the annoyance of iteratively moving their information system from one platform to another one, at a high cost, with limited ROI and with little hope that this conversion process ever halts. In theory computer systems should have lives measured in decades but unfortunately in practice the obsolescence period of technical support platforms is usually much shorter. The only solution seems to lie in separating the stable parts from the variable parts of the information systems.

In reaction to this new software crisis, the OMG is proposing the MDA framework. In this proposed vision of the way to construct and maintain information systems, the regular evolution and obsolescence of technical platforms is taken as a normal process and no more as an exceptional event. Since there is virtually no hope that an ideal and universally accepted middleware platform will prevail, solutions have to be found to integrate this new parameter in modern software engineering practices. The obvious conclusion that comes out from this observation suggests to rapidly switch from code-centered to model-centered approaches. Fortunately the OMG has a set of readily available standards that could provide the backbone for the new vision: UML, MOF, XMI, CWM, SPEM, etc. These specifications guarantee the basic interoperability for the multiple tools that are needed to transform the abstract MDA vision in commercial workbenches for industrial software development and maintenance.

Tooling the MDA is thus the next important step. This means that existing technology has to be identified and that its compliance with current standards has to be assessed. This also means that new techniques may have to be invented to facilitate the emerging model engineering practices. CASE tools (e.g. Rational Rose) and Application Development tools (e.g. IBM Visual Age) will certainly have to closely interoperate if they keep separate identities. An important number of other more specialized tools will rapidly become available to populate the framework. A classification of the necessary functionalities is needed. Among the more obvious we may quote assisted model transformation, model testing, model metrication, parameterized code generation, reverse engineering, prototypers for UML specification, verifiers for OCL statements, model and meta-model management including version management and repository support, meta-model alignment, merging and comparison, process-centered environments, collaborative systems, and much more.

We are mainly interested here in software maintenance and reverse engineering. The contribution presents an original software evolution scheme, proposed to become part of the MDA framework. An initial prototype has been built in our lab. It implements a first version of the proposal as a simple C# program, using the basic reflective mechanisms of the language. The main idea is to allow the C# program to automatically generate a XMI model of itself, based on a given specific MOF-compliant meta-model. In the first prototype, the correspondence between the MOF meta-model and the generation code is hand-synchronized. Building on this preliminary work, a more automatic way of proceeding is suggested. If we are able to provide facilities for meta-model-driven reverse engineering, the resulting model may be later used in a number of other operations available within the MDA framework (model transformation, verification and validation, installation on other platforms, separation of the business entities from the technical entities, etc.)

We will show in the presentation how modern meta-modeling and meta-programming techniques may be combined to implement a new generation of software maintenance tools. The proposal deals with the future maintenance of software written in the C# programming language, supported by the DotNet platform and uses all support available in the MDA framework (UML, MOF, XMI). In addition to this, we take advantage of the introspection properties of C#. A similar approach could also be used with other modern programming languages like Smalltalk or Java. Complementarity of these with more conventional approaches of reverse engineering legacy systems (e.g. Cobol programs) will also be discussed.