What is the Model Driven Architecture?

MDA stands for Model Driven Architecture. It is a framework for software development, made by the OMG. Key to MDA is the importance of models in the software development process. Within MDA the software development process is driven by the activity of modeling your software system.

- The MDA process
- Automation of the Transformation Steps
- Modeling Becomes Programming, and Vice Versa
- MDA Building Blocks
- Does it Work Now?
- What You Must Do to Implement the MDA Process
- The MDA Revolution

The MDA process

The MDA process defines three steps:

1. First, you build a model with a high level of abstraction, that is independent of any implementation technology. This is called a Platform Independent Model (PIM).

2. Next, the PIM is transformed into one or more Platform Specific Models (PSMs). A PSM is tailored to specify your system in terms of the implementation constructs that are available in one specific implementation technology, e.g. a database model, an EJB model.

3. The final step is to transform a PSM to code. Because a PSM fits its technology very closely, this transformation is rather trivial. The complex step is the one in which a PIM is transformed to a PSM.
Automation of the Transformation Steps

The MDA transformations are executed by tools. Many tools have been able to transform a platform specific model to code, there is nothing new to that. What's new, is that the transformation from PIM to PSM is automated as well. This is where the obvious benefits of MDA lie. How much time has been spend in your projects with the painstaking task of building a database model from an object-oriented design? How much (precious) time was used by building a COM component model, or an EJB component model from that same design? It is indeed about time that the burden of IT-workers is eased by automating this part of their job.

Modeling Becomes Programming, and Vice Versa

Because the whole MDA process is driven by the PIM, and the PIM is automatically transformed into a PSM, and from there to code, modeling in effect will become programming on a higher level. The PIM specifies the code that needs to be produced. There will be far less programming, or coding (as we like to call it) that remains to be done by hand. Programming, in the sense of building software systems, will eventually become modeling. All software development effort will be focused on producing a good, high level, independent model of the system.

MDA Building Blocks

Now what do we need to implement the MDA process? The following are the building blocks of the MDA framework:

- High level models, written in a standard, well-defined language, that are consistent, precise, and contain enough information on the system.
- One or more standard, well-defined languages to write high level models.
- Definitions of how a PIM is transformed to a specific PSM that can be automatically executed. Some of these definition will be ‘home-made’, that is made by the project that works according to the MDA process itself. Preferably, transformation definitions would be in the public domain, perhaps even standardized, and tuneable to the individual needs of its users.
- A language in which to write these definitions. This language must be used by the transformation tools, therefore it must be a more or less formal language.
- Tools that implement the execution of the transformation definitions. Preferably these tools offer the user the flexibility to tune the transformation step to their specific needs.
- Tools that implement the execution of the transformation of a PSM to code.

Does it Work Now?

MDA is a new development framework. Are all of the MDA building blocks complete, and ready for use? Well, most of them are. And if you choose your toolkit with care, the benefits of MDA are yours to pick from this day on.
• High level models: must be written by you. You are still the only one that knows what your systems must do.
• Modeling languages: use UML in combination with the OCL. This is the only option that ensures consistent and precise models, while preserving the readability.
• Transformation definitions: a few tool vendors have made their own. Look for those, they are the ones that offer the best tools.
• Transformation definition language: the OMG is working on this. In November 2003 the first proposals are expected.
• PIM to PSM - tools: many are becoming available, although for some the level of abstraction of the PIM is not much higher than the PSM. Look for those that offer the highest level of abstraction. Most of these tools are combined with a PSM to code component.
• PSM to code - tools: many were and are available.

What You Must Do to Implement the MDA Process

When you are ready to adopt the MDA process as your own software development process, there are two things that you must do.

1. Build solid PIMs. MDA puts demands on PIMs that are not met by a simple diagram and some text. The PIM must be processed by automated tools. It must therefore be more formal, more precise, consistent, and it must contain as much information about the system as possible.

2. Choose the right toolkit. The aim of MDA is to automate as much of the implementation effort as possible. Of the available tools some give more support than others.

The MDA Revolution

The benefits of MDA include productivity gain, by automating part of the IT-worker's job, portability, and interoperability. All of these have been identified and described (see the OMG website). But there is another promise hidden in the MDA that up till now few have seen.

In the early 1960s our industry was in the middle of a revolution. The use of existing assembly languages was substituted by the use of procedural languages, or third generation languages (3GLs) as they were called later. Algol, Simula were made in those early days. Programs written in procedural languages were compiled (or could we say transformed) into programs written in an assembly language. The compilers were not that good at the time. Algol, for instance, had a possibility to give hints to the compiler on how to translate a piece of code. Maybe you lost a bit of efficiency, speed, expressiveness, but the advantages of a higher level language are obvious to all of us right now.

What MDA can bring us is another revolution. PIMs can be compiled into PSMs, which are compiled into 3GLs (which themselves are compiled into assembly languages). The compilers (read: transformation tools) will not be that good for a few years. Its users will need to give hints on how to transform part of a model. But eventually the advantage of working on a higher level of abstraction will be clear to anyone in the business. It is this promise that, although it complete realisation may be some time ahead, makes the MDA really worthwhile.