The expansion of enterprise computing infrastructures to embrace new platforms and applications—to say nothing of the Internet—has driven the goal of achieving greater software portability, cross-platform interoperability and platform independence. The price for achieving these goals, however, is higher costs of software development. The Object Management Group (OMG) has introduced Model Driven Architecture (MDA), a new approach for achieving these important goals. MDA is a concept that will change the way an organization designs and develops software by separating an application’s business logic from the infrastructure on which it runs.

Traditional software design and development processes create applications for deployment to a specific platform. MDA introduces a higher level of abstraction, enabling organizations to create models that are in essence application “blueprints,” independent of any particular hardware, operating system, platform or middleware. MDA bridges what has been a significant gap between business modeling and software development by ensuring that business models drive application development, not the other way around.

MDA has three cornerstones:

- multiple models at various abstraction levels
- transformations between these models
- support for the MDA set of standards.

Franco Flore, Senior Product Manager for OptimalJ at Compuware, outlines how important automatic transformations are to realizing the benefits of MDA.
Modeling with MDA exploits the OMG’s Unified Modeling Language (UML) to create two key MDA model architectures:

- a Platform Independent Model (PIM), representing business functionality and behaviors without technological detail
- a Platform Specific Model (PSM), which uses a defined standard to map from the PIM to the target platform or platforms for the application modules (for example, J2EE, .NET, CORBA and so on). Code based on the PSM is mapped to create a viable application.

There is a problem, however; though OMG has defined the mapping standards between the PIM, the PSM and the resulting code, it has yet to define how to implement the models. This task has been left to more than 40 software development tool vendors currently supporting the MDA initiative. Though many of these vendors have implemented portions of the MDA, few have done so in its entirety. For users to fully benefit from MDA, vendors need to implement all of MDA, meaning implementing all three cornerstones, and ensure that their tools are standards-based and business model-driven. In addition, tools should automatically transform higher-level business- and platform-independent models into lower-level platform-specific models and generate code automatically.

Until now, creating transformations between the PIM, PSM and the code have been primarily manual, labor-intensive and costly tasks, requiring more experienced IT staff to complete successfully. The unfortunate result is a realization that, without the automatic transformation between models, the effort needed to manually transform the models becomes prohibitive. Until the transformations from the PIM to the PSM, and from the PSM to code are fully automated, organizations using MDA will not get a full return on MDA’s promise of faster, less costly software development.

MDA will become a viable, long-term architecture—and deliver on its productivity promise—when MDA-compliant vendors automate the transformations between models and code. Automatic transformation is the key for turning the MDA vision into reality. Organizations wishing to implement MDA should ensure the MDA-based tools they consider using fully implement these transformations automatically, or they risk not gaining the full benefits of MDA.

Moving from models to code

A variety of organizations are using MDA-based or other models and modeling tools today, at the business process level. But for most of these organizations, modeling is still a paper-based exercise. This creates a
synchronization problem between the model, which is the application blueprint, and the application itself, because the application is updated but the model isn’t. Therefore, from a documentation perspective, the model becomes useless. MDA addresses this problem by merging the modeling and coding processes. However, nearly all of the transformations from a PIM to a PSM and from a PSM to the code itself are being done by hand. This has forced organizations to assign the most skilled IT staff to manually transform the models into code. And their level of skill at doing these transformations directly affects the success of the application.

Standardizing automatic transformations within the MDA framework is currently being addressed by some of OMG’s key members, including Compuware and Sun Microsystems. It’s not a trivial undertaking. Their goal is not only to automate the transformations in a standard way, but also in a way that is easily understood. If IT professionals don’t understand how to use transformation tools, the buy-in from the user community will be negligible at best.

**Patterns are the key**

Patterns are the key to automating transformations between models. As defined by Grady Booch in his book *Unified Modeling Language User Guide*, patterns are “a common solution to a common problem in a given context.” Created by software architects to describe a general solution to reoccurring design problems, patterns capture specific knowledge about the architectures, platforms and technologies that are applied to the application. The use of patterns ensures that the application incorporates best practices, standards and guidelines in a consistent and reliable way—maximizing quality and minimizing risk.

Transformation patterns roughly fall into two categories:

- Technology patterns transform a PIM into a PSM, where the PSM is the target platform chosen by an organization, such as the J2EE platform. Technology patterns apply the technology-specific knowledge to the PIM.

- Implementation patterns transform the PSM into actual code by implementing the specification. For the J2EE specification, for example, the following is generated: SQL scripts for a DBMS tier, Enterprise JavaBeans for the EJB tier, and the Servlets and Java Server Pages for the web tier. Implementation patterns implement an organization’s chosen platform in the PSM through code creation.

OMG is currently working with several of its members, including Sun and Compuware, to define standards for Transformation patterns. For these standards to succeed, widespread support from vendors and end
users alike is needed. Vendors and end users also need to embrace UML, together with standards such as Common Warehouse Metamodel, Meta Object Facility, XML Metadata Interchange and others.

**Conclusion**

With 600 member companies, the OMG is the largest software standards organization of its kind. More than 40 vendors currently support the MDA concept, and the support for MDA is growing—both among vendors that are developing tools for MDA development, and organizations that are adopting MDA as a set of standards for software development.

In October 2002, in a research piece entitled “Development Efficiency Drives Adoption of New Development Paradigms for J2EE and .NET,” Giga Information Group states, “As initiatives like MDA increase in popularity within large enterprises, especially among companies that have a formal architecture in place, the adoption and use of MDA-related tools is growing at a much more rapid pace. Companies that want to maintain or increase their future competitive edge will need to begin evaluating, planning for and migrating development staff to these more efficient forms of development.”

Tool vendors must ensure that they adopt and implement the MDA concept in its entirety. Before choosing an MDA-compliant tool set, end users need to determine a vendor’s level of compliance with MDA—the whole vision, and not just bits and pieces. Specifically, end users need to find vendors that are using a pattern-based approach to automate transformations between models. Pattern-based development offers the best approach to automated transformations, a key cornerstone to implementing MDA in its entirety and the only way to fully realize its benefits.

**Compuware OptimalJ**

Compuware OptimalJ is an enterprise application development environment designed for organizations that are adopting J2EE standards and Model Driven Architecture (MDA), as standardized by the Object Management Group (OMG), in order to accelerate the development, integration and maintenance of applications for competitive advantage.

OptimalJ uses patterns to automatically translate business models into working applications by implementing OMG’s MDA in its entirety. OptimalJ enables organizations to rapidly respond to change, increase development efficiency and dramatically decrease maintenance costs.