

# Developing in OMG's New Model-Driven Architecture

Jon Siegel  
Director, Technology Transfer  
Object Management Group

In this paper, we're going to describe the application development process supported by OMG's Model Driven Architecture (MDA) – the model that you build, the artifacts that you produce, how information flows from one set of artifacts to the next, and how the MDA process ultimately yields an application running on virtually *any* target middleware platform. OMG has introduced the MDA in several papers, all accessible via a single click from our main MDA web page [www.omg.org/mda](http://www.omg.org/mda). We're not going to review the MDA or do any other introductory stage-setting here. If you need an introduction, check these references on the web.

## Developing in the MDA – Single Target Platform

Although a primary advantage of MDA-based development is the ability to produce applications for virtually every middleware platform from the same base model, we're going to start with a simple example – generating a server on a single platform. Once we've completed this and traced the routes all of the various code artifacts through the process, we'll show how the MDA re-uses the *same* mechanism for multiple targets. We've picked the CORBA Component Model (CCM) as our example target platform but the differences among platforms, although crucial in detail, are not significant at the introductory level of this paper.

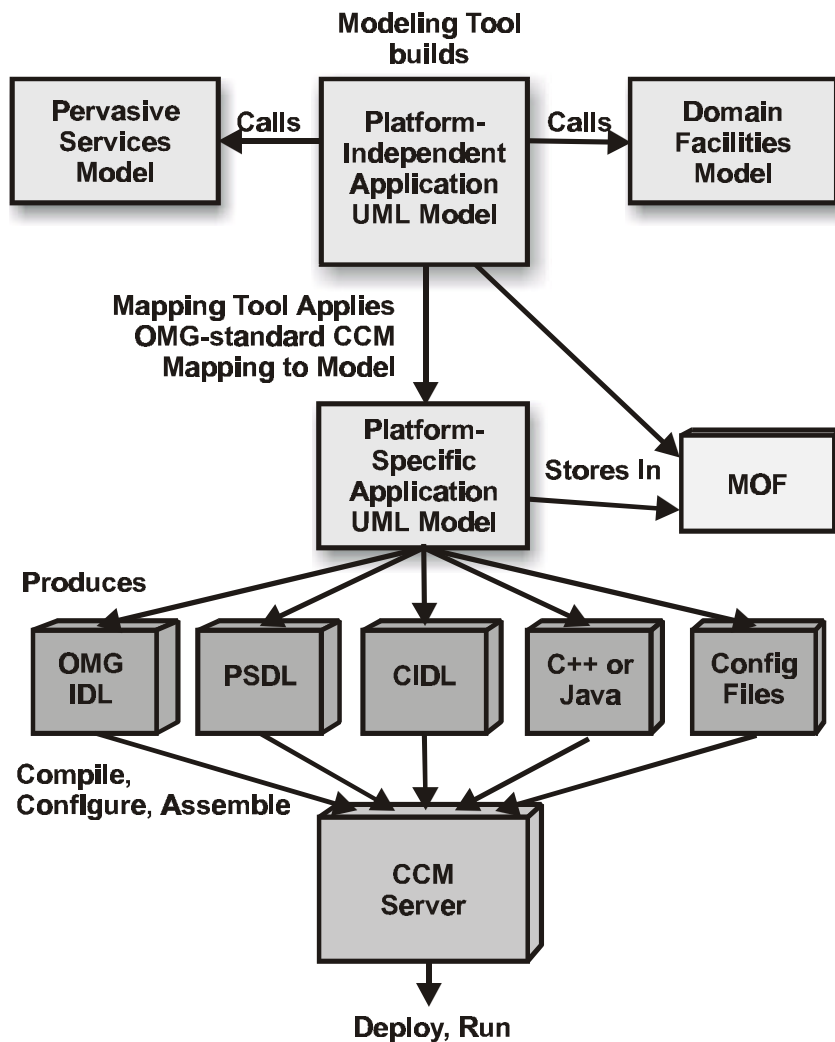
### Step 1: The Platform-Independent Model (PIM)

All MDA development projects start with the creation of a *Platform Independent Model* (PIM), expressed in UML and shown at the top of Figure 1. Reflecting business functionality and behavior undistorted by technology, MDA models at this highest level can be constructed by business experts rather than systems programmers. PIMs exist at several levels; more refined PIMs include some behavior reflecting their general platform type (a component activation pattern, for example) although they never specialize to an individual platform.

Specializations and extensions to UML give it the power to express the detailed models required by the MDA. Termed a *UML Profile*, a standardized set of extensions (consisting of *stereotypes* and *tagged values*) defines a UML environment tailored to a particular use, such as modeling in a specific environment or on a specific platform. PIMs will be modeled using the profile for Enterprise Distributed Object Computing (EDOC) or Enterprise Application Integration (EAI), both near the end of their successful adoption processes. The UML profile for CORBA completed adoption by OMG in 2000; profiles for other platforms are in process.

What sorts of behavior will be incorporated into the PIM? *Object Constraint Language*, a part of UML, lets modelers specify invocation pre- and post-conditions very precisely, so these will surely be included in this category. Setting and getting of parameter values, a common task in business applications, is easy to automate so look for this to be well handled by code generation facilities even in early generation MDA tools. On the other hand, coding of the calculation engines for new algorithms (for financial derivative contracts or scientific applications, for example) may never be fully automated.

MDA application-modeling tools will contain representations of the Pervasive Services and Domain Facilities, allowing them to be used by or incorporated into your application as you model it, via a menu selection. Any facility defined in UML may be imported into the tool as well, and used by the application in the same straightforward way. If the service or facility runs on another middleware platform, MDA tools will generate cross-platform invocations automatically.



**Figure 1: Using the MDA to generate a CCM server.**

## Step 2: The Platform-Specific Model (PSM)

Once the first iteration of your PIM is complete, it is stored in the MOF and input to the mapping step which will produce a *Platform-Specific Model* (PSM) as shown in the second row from the top in Figure 1. To produce your PSM, you will have to select a target platform or platforms (you don't have to run your entire model in the same component environment, as we'll show in the next section) for the modules of your application.

During the mapping step, the run-time characteristics and configuration information that we designed into the application model in a general way are converted to the specific forms required by our target middleware platform. Guided by an OMG-standard *mapping*, automated tools perform as much of this conversion as possible, flagging ambiguities for programming staff to resolve by hand. Early versions of the MDA may require considerable hand adjustment here; the amount will decrease as profiles and mappings mature over time.

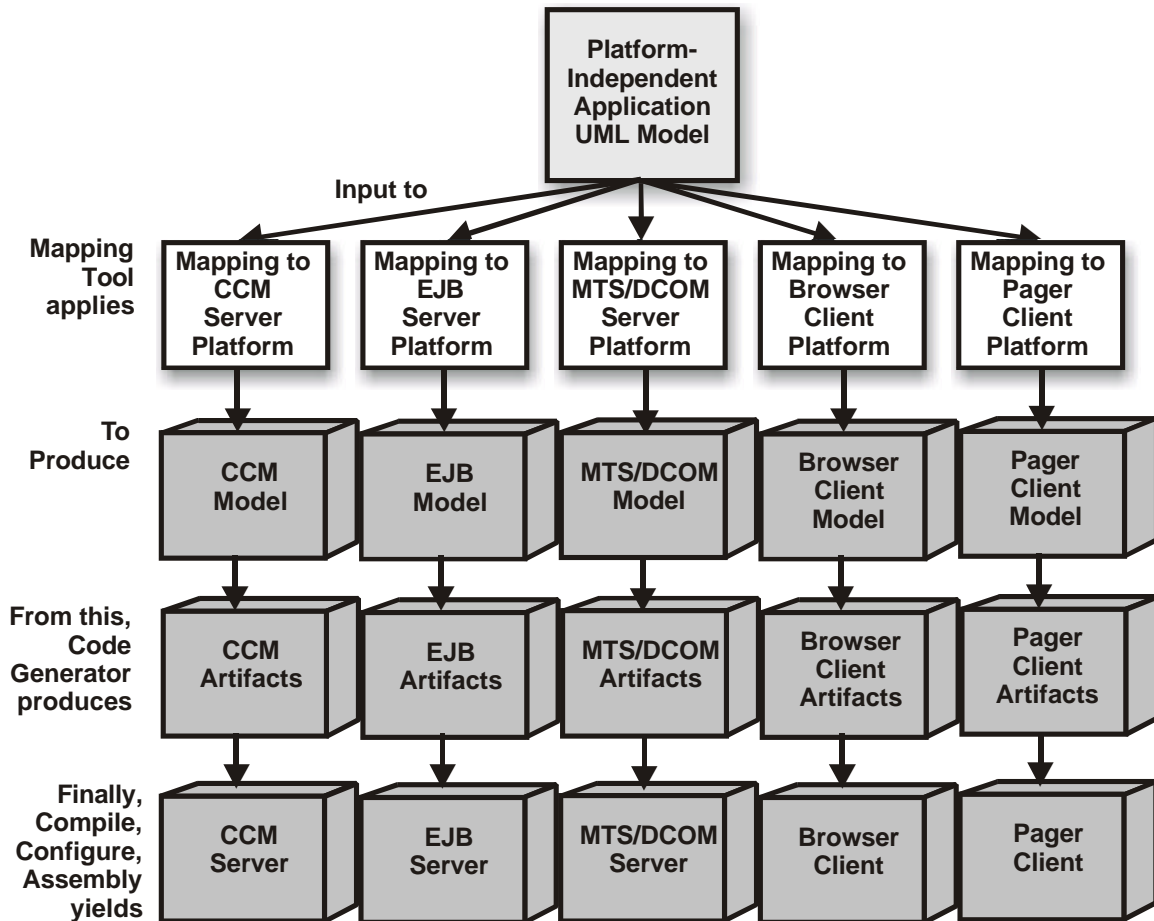
## Step 3: Generating the Application

As Figure 1 shows in its third row, an MDA tool for CCM must generate all of the files that our platform requires. Artifacts for other middleware targets will be different. Each MDA mapping will produce the file types that its middleware platform requires.

Hand-coding, when required, will be applied to the output of the code generation step. Then a middleware-specific tool will compile all of the various code elements, and create executable modules for deployment and installation.

## Developing in the MDA – Multiple Target Platforms

Although development of a model-to-code capability for a single target platform is significant enough, benefits multiply when the MDA extends to multiple targets as we show in Figure 2.



**Figure 2: Leveraging Other Middleware and Client Platforms**

OMG will define mappings to many middleware targets. By adding various client as well as server platforms, the MDA can be made to generate code for these targets as well, regardless of differences in calling pattern from server to any number of client types.

The first three columns of Figure 2 show MDA mappings to a number of middleware server platform targets. We've chosen the CCM, EJB, and MTS/DCOM for the figure, but these are only examples: you can expect mappings to XML/SOAP, .NET, and other new environments as well as to legacy systems such as mainframe-based TP applications. We admit, it would be unusual for an enterprise to generate implementations of the same application on an assortment of servers but ISVs will surely do this to support customers on a range of platforms.

The final two columns illustrate mappings to sample client platforms. Again, we've picked two only as examples; there is nothing in the MDA that would prevent any particular client platform from being included. Because clients can differ markedly even at the model level – consider a voice-activated telephone client as compared to a browser-based web version – we expect that some different client platforms will have to be modeled individually. Nevertheless, many of these will share a common server interface with differences accounted for automatically by the MDA.

## Conclusion

There's no such thing as the "best" target platform: what's best for you may not be best for me, because of the hardware, or network, or developer teams that our enterprises already have in place. In this world, the "best" development environment is one that lets you choose the target platform *after* you've modeled your application, and even move from one to another with relative ease. Even more important, you need to interoperate with *every* platform out there, regardless of the one you've chosen, to take advantage of business opportunities. That's what the MDA gives you.