

## ▶ Model Driven Architecture Targets Middleware Interoperability Challenges

by [Richard Soley](#)

Chairman and Chief Executive Officer

Object Management Group  
and the OMG Staff Strategy Group

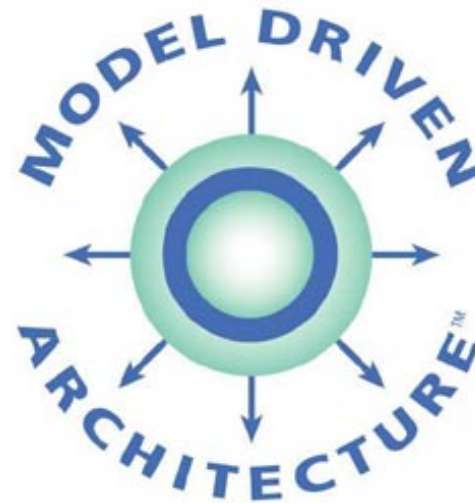
***"CORBA was a powerful first step, but we have more steps to take."***

-Fred Waskiewicz, OMG Director of Standards

*Since its inception, the Object Management Group (OMG) has provided vendor- and language-independent interoperability standards to the enterprise. The CORBA (Common Object Request Broker Architecture) standard has recently been modified for -- and embraced by -- environments that require specialized real-time, fault-tolerant, and embedded systems. The OMG's complementary core modeling specifications include the Unified Modeling Language (UML), the Common Warehouse Meta-model (CWM), the Meta-Object Facility (MOF), and XML Metadata Interchange (XMI).*

*Now, the OMG is building on the success of CORBA and the UML to take aim at the problems arising from the proliferation of enterprise middleware. Its Model Driven Architecture (MDA) initiative champions the extensibility and reliability of CORBA while acknowledging that enterprises cannot simply abandon their investment in other technologies. (More on MDA can be found at [www.omg.org/mda](http://www.omg.org/mda))*

*Rational Software has been particularly active, along with many other OMG member companies, in contributing ideas and principles toward the creation of MDA. If you are a Rational customer interested in modeling or software development infrastructure, I think you'll be interested in the OMG's concepts, goals, and plans for MDA, as described in the following article.*



- ▶ [subscribe](#)
- ▶ [contact us](#)
- ▶ [submit an article](#)
- ▶ [rational.com](#)
- ▶ [issue contents](#)
- ▶ [archives](#)
- ▶ [mission statement](#)
- ▶ [editorial staff](#)

Over the past decade or so, the middleware landscape has continually shifted. For years we've assumed that a clear winner would emerge and stabilize this state of flux, but the time has come to admit openly what many of us have suspected all along: The string of emerging contenders will never end! And, despite the advantages (sometimes real, sometimes imagined) of the latest middleware platform, migration is almost always expensive and disruptive.

OMG's layered services and vertical market specifications are built on CORBA -- which we regard as the optimum middleware -- and strongly established through the OMG community process. We do recognize, however, that enterprises often have applications on other middleware that simply *have* to be integrated into new or modified systems, even though this process is time-consuming and expensive. Furthermore, the middleware these enterprises use continues to evolve.

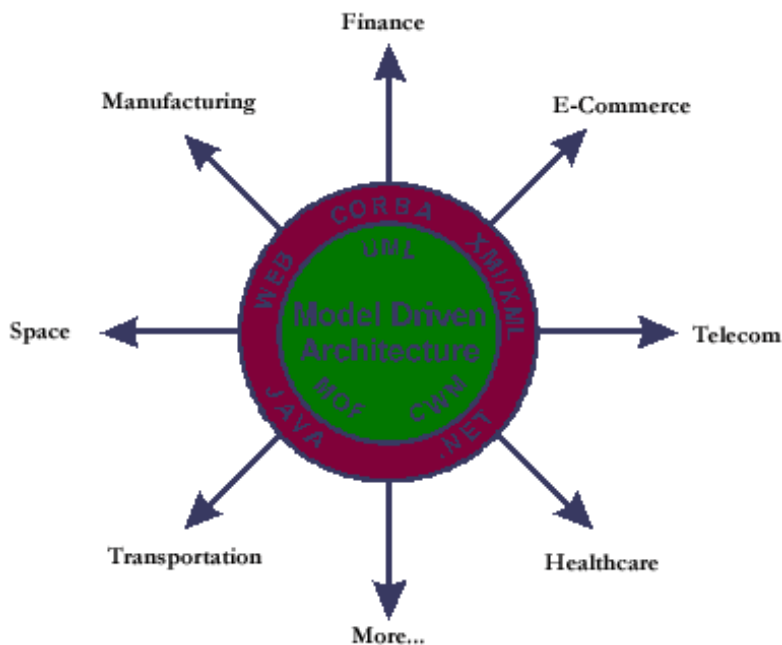
And to make matters even more complicated, before the Internet evolved into an enterprise marketplace, organizations often used different technologies for communication within and beyond their firewall. Now, some businesses want to expose components they built for internal communication out beyond the firewall -- for business-to-business e-commerce, for example. Others want to move components off their extranets and place them behind their firewalls because of an acquisition or merger. So in addition to resolving basic integration problems, IT organizations must find a way to preserve their development investment in new components as enterprise boundaries shift -- and the underlying technologies change.

## **Addressing the Problem: Model Driven Architecture**

Fortunately, there is a way to manage this situation. Building on OMG's core modeling standards, we created a Model Driven Architecture (MDA) that is language-, vendor- and middleware-neutral.

As Figure 1 shows, the core of this architecture is based on the UML, the MOF and CWM. Multiple *core models*<sup>1</sup> are currently under development: One will represent enterprise computing with its component structure and transactional interaction; another will represent real-time computing with its special needs for resource control; more will be added to represent other specialized environments. Each core model will be independent of any middleware platform. The total number, however, will be small, because each core model will represent the common features of *all* the platforms in its category.<sup>2</sup>

Whether your ultimate target is the CORBA Component Model (CCM), Enterprise JavaBeans (EJB), the Microsoft MTS and the new .NET architecture, or some other component- or transaction-based platform, the first step in constructing an MDA-based application will be to create a platform-independent application model -- using the UML -- that is consistent with the appropriate core model. Then, platform specialists can convert this general application model into one targeted to a specific platform such as CCM, EJB, or .NET. Figure 1 shows these target platforms in the thin ring surrounding the core.



**Figure 1: OMG's Model Driven Architecture**

Although standard *mappings* will allow tools to automate some of the conversion, in most cases some hand coding will be required, especially in the absence of MDA tools. As users and tool builders gain experience, and techniques for modeling application semantics become better developed, less human intervention will be needed.

The platform-specific model faithfully represents both the business and technical run-time semantics of the application. It's still a UML model, but it is expressed (because of the conversion step) in a dialect (i.e., a profile) of UML that precisely mirrors technical run-time elements of the target platform. The semantics of the platform-independent original model are carried through into the platform-specific model.

The next step is to generate application code itself. For component environments, the system will have to produce many types of code and configuration files, including interface files, component definition files, program code files, component configuration files, and assembly configuration files. The more completely the platform-specific UML dialect reflects the actual platform environment, the more completely the application semantics and run-time behavior can be included in the platform-specific application model, and the more complete the generated code can be. In a mature MDA environment, code generation -- provided through tools from vendors such as Rational Software and their competitors -- will be substantial or perhaps even complete in some cases. Early versions are unlikely to provide a high degree of automatic generation, but even initial implementations will simplify development projects and represent a significant gain, on balance, for early adopters; they will be using a consistent architecture for managing the platform-independent and platform-specific aspects of their applications.

As Figure 1 shows, many of today's connection technologies will be

integrated by the MDA, with room for tomorrow's "next best thing." CORBA represents the best middleware choice because it is vendor- and language-neutral, and bridges easily to all of the other middleware environments. To accommodate those enterprises with multiple middleware platforms on their network, however, many non-CORBA platforms will be incorporated into the MDA. One of the first will be the Java-only EJB.

## **Adding New Middleware Platforms**

Because the MDA is platform-independent at its core, adding new middleware platforms to the interoperability environment will be straightforward: After identifying the way a new platform represents and implements common middleware concepts and functions, OMG members can incorporate this information into the MDA as a mapping. Various message-oriented middleware tools, plus XML/SOAP (Simple Object Access Protocol) and .NET will be integrated in this way; in fact, by rationalizing the conflicting XML document type definitions (DTDs) that are being proposed in some industries, the MDA can even help organizations interoperate across them. And, as representations of multiple middleware platforms are added to the MDA and mature over time, the generation of integration tools -- bridges, gateways, and mappings from one platform to another -- will become more automated.

Interoperability will be most transparent within an application category: enterprise applications with other enterprise applications; real-time applications with other real-time applications. This follows from our approach of a separate core model for each category; differences between application categories prevent us from basing all applications on a single core model. But identifying and exploiting concepts common to two or more categories can smooth over the boundaries to some extent.

## **Working with Legacy Applications**

Our discussion so far has assumed that we were building an application -- and its model -- from scratch. Legacy applications present different challenges: Many were built before component environments were even conceived and do not fit neatly into any of our core models. Legacy applications may be brought into the MDA, however, by wrapping them with a layer of code that is consistent with an MDA core model. If we build an MDA model of the wrapper first, then the outer portion of that wrapper -- the one that faces the network and interoperates with our other applications and services -- may be generated automatically, at least in part. The other side of the wrapper -- the one that invokes and returns from the legacy application itself -- typically must be hand coded.

## **An Internet ORB**

As a next-generation OMG standard currently in development, the MDA can serve as an Internet *Object Request Broker* (ORB), integrating across *all* middleware platforms, past, present, and future. OMG, the organization

that knows ORBs better than any other, is ideally suited to extend this concept beyond middleware standards to a middleware-neutral, model-driven approach, offering users these specific advantages:

- Organizations will be able to build new MDA-based applications using the middleware of their choice. They will have the security of knowing that the essential semantics of their application have been systematically distilled into a platform-independent model, and that any future migrations they might need to make to different middleware (or even new versions of the same middleware) will be reasonably manageable. In addition, they can produce interoperability bridges and gateways to other MDA-based applications within an enterprise as well as interconnections with customers, suppliers, and business partners in a methodical way, using a consistent architecture and some degree of automatic generation.
- Legacy applications -- the ones that keep your business in business - - will interoperate with an organization's current applications once they wrap them as we described and incorporate their functions into the MDA. They can remain on their established platforms; the MDA will help automate construction of bridges from one platform to another.
- Industry standards for all verticals will include platform-independent models defined in terms of the MDA core models: standard facilities performing standard functions, that you can buy instead of build, with interoperability and evolvability improved by their MDA roots. We'll describe these facilities and their role below.
- As new middleware platforms emerge, the OMG's rapid, consensus-based standardization process will incorporate them into the MDA by defining new standardized mappings. MDA tools will thus be able to target additional platforms for conversion to a platform-independent model. These tools will also be able to support bridges to the new platforms.
- Developers will gain the ultimate in flexibility: the ability to regenerate code from a stable, platform-independent model as the underlying infrastructure shifts over time. ROI will rise from the reuse of application and domain models across the software lifespan, especially during long-term support and maintenance, the most expensive phase of an application's life.
- Models are built, viewed, and manipulated via UML, transmitted via XMI, and stored in MOF repositories.

- Formal documentation of system semantics (through modeling) will increase software quality and extend the useful lifetime of designs (thereby increasing ROI) .

Taking advantage of our standards and tools that exploit them, OMG members have this integration task well underway. They are defining the Enterprise Computing Core Model and mapping it to the most widely-used middleware platforms. They are also defining a core model for real-time computing.

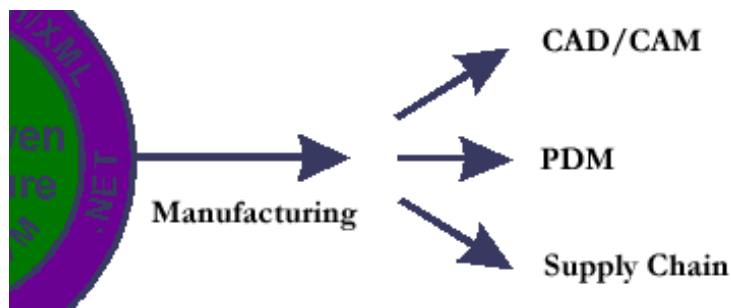
## **Standardizing Domain Models**

Since January 1996, a sizeable percentage of OMG members have been meeting in Domain Task Forces (DTFs), communities focused on standardizing services and facilities in specific vertical markets. Until now these specifications have consisted of interfaces written in OMG Interactive Data Language (IDL), with accompanying semantic descriptions in English text. Standardizing components at a platform level, as we have done with CORBA, is certainly a viable contribution to solving the integration and interoperability problem, but we are now prepared to go a step beyond that.

A well-conceived service or facility is always based on an underlying semantic model that is independent of the target platform, even if that model is not documented explicitly. OMG's domain specifications fall into this category because the models for them are *not* expressed separately from their IDL interfaces. Since their models are hidden, these services and facilities have received neither the recognition nor the widespread implementation and use that they deserve outside of the CORBA environment, especially considering the quality of their underlying models. Extending these implied models outside of CORBA just makes sense. The OMG has already implemented The Healthcare Resource Access Decision Facility, for example, in Java and EJB as well as CORBA. And there are more underway, as shown in Figure 1.

Basically, each DTF will produce standard frameworks for standard facilities in their application space. These will be formulated as normative, platform-independent UML models augmented by normative, platform-specific UML models and interface definitions for at least one target platform. Their common basis in MDA will also promote partial generation of implementation code, but that code, of course, will not be standardized.

For manufacturing, for example, the DTF could produce normative MDA UML models, IDL interfaces, Java interfaces, XML DTDs, etc. for CAD/CAM interoperability, PDM (Product Data Management), and supply chain integration (see Figure 2). Once these models are completed and adopted, their implementation can be partially automated in any middleware platform supported by the MDA.



**Figure 2: UML-Based Model Frameworks for Manufacturing**

The three facilities in our example -- CAD/CAM, PDM, and Supply Chain -- would benefit from the interoperability that only the MDA can provide. Because CAD/CAM and PDM applications are tightly integrated, they are likely to be implemented by an individual enterprise or software vendor in, for example, CORBA or EJB. Supply chain integration, by contrast, is more of an inter-enterprise function, so we might expect an XML/SOAP-based implementation supported by an industry market-maker or trade organization to become popular. It will be essential to interoperate among the three, however: CAD/CAM designs feed into PDM production facilities that drive the supply chain; in turn, the supply chain will refer back to CAD/CAM for details on a particular part. If all three functions start out as UML models in the MDA, we may eventually be able to generate a significant portion of the implementation for each on its preferred platform, as well as the bridges we need to integrate each of the facilities with the other two.

## **Including Pervasive Services**

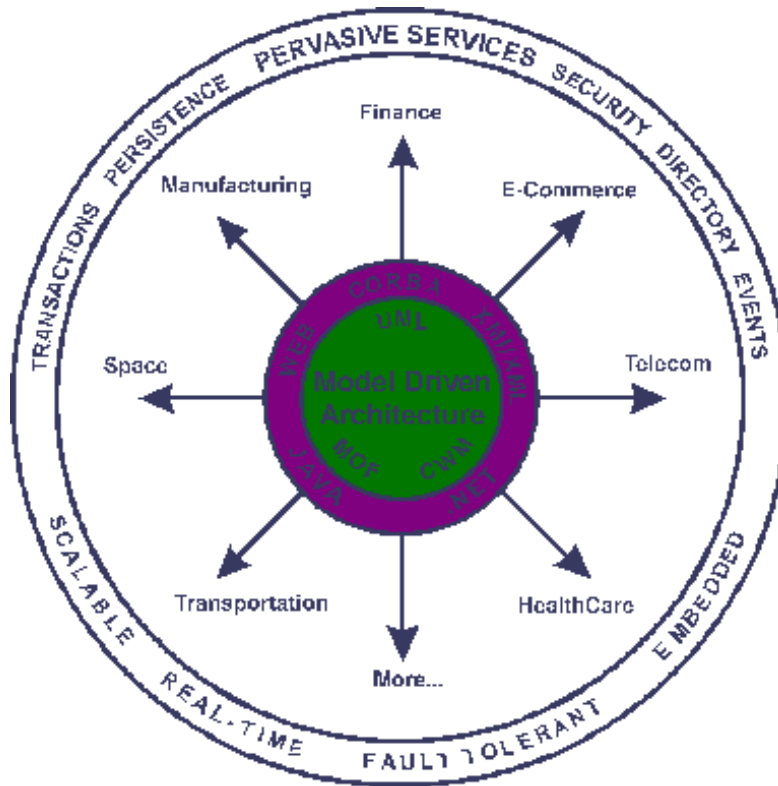
Enterprise, Internet, and embedded computing rely on a set of essential services. The list varies somewhat depending on the source but typically includes directory services, event handling, persistence, transactions, and security. In addition, computing systems or applications may take on specialized attributes in either their hardware or software -- that is, they may be scalable, real-time, fault-tolerant, or designed to fit into a confined (embedded) environment.

When these services are defined and built on a particular platform, they necessarily take on characteristics that restrict them to that platform, or ensure that they work best there. To avoid this, OMG will define Pervasive Services at the platform-independent model level in UML. Only after the services' features and architecture are fixed will platform-specific definitions be generated for all of the middleware platforms supported by the MDA.

At the abstraction level of a platform-independent business component model, services are depicted at a very high level (similar to the view the component developer has in CCM or EJB). When the model is mapped to a particular platform, developers will use their development tools of choice to generate code (or dynamically invoke it) that makes calls to the native services of those platforms. The pervasive services will be visible only to

lower-level applications, i.e., those that write directly to services.

Hardware and software attributes -- scalability, real-time, fault tolerance, or embedded characteristics -- will be modeled as well. By defining UML representations for these attributes or, in the case of fault tolerance, for an environment that combines the attribute with enterprise computing, OMG will extend the MDA to support and integrate applications with these desirable characteristics.



**Figure 3: MDA Encompasses Pervasive Services and Specialized Computing Environments.**

Figure 3 emphasizes that pervasive services are available to all applications, in all environments. True integration requires a common model for directory services, events and signals, and security. By clarifying that these services are implementable in different environments and easily integrated, MDA represents our goal of universal integration: *it becomes a global information appliance.*

## **An Invitation From the OMG**

Although much of the infrastructure for the MDA is in place or under construction, there is still a lot to do. If your company works at either the modeling or infrastructure level, you can have a voice in defining the MDA. Requests for Proposals (RFPs) have been issued for UML 2.0, and all of the components of the Business Objects Initiative (BOI) except the first are

still in their formative stages in the OMG adoption process. Of the mappings to various middleware environments, only that to CORBA is even in progress; the rest exist only as potential RFPs. UML models for the pervasive services have not yet been constructed or adopted.

Application models defined by the DTFs will form the basis for implementations extending from CORBA to every middleware environment. Whether your company is a provider or user of domain-level applications, now is the time to get involved in their standardization. As a provider, you can maximize your impact on future standards and be recognized as a key player. As a user, you can integrate your company's requirements into the RFP that defines the new standard and influence the models and standards that you will eventually use. You will also enjoy working with the best and brightest in the industry to develop your architecture of choice. One condition: To ensure that OMG standards remain relevant to the marketplace, companies whose submissions are adopted by OMG members as a standard must agree to market or commercially use an implementation of the specification.

With MDA, the OMG is continuing its quest to support integration and interoperability across heterogeneity at all levels. Our first goal -- to enable integration by introducing a distributed object model -- is complete. Today, objects are at the core of every vendor's enabling architecture and all e-businesses. But our integration mission is not yet fulfilled; now, we must evolve from a middleware-centric to a modeling-centric organization.

That does not mean, of course, that we are leaving CORBA behind. CORBA is a foundation of this new architecture. As the only vendor- and language-independent middleware, it is a vital and necessary part of the MDA superstructure; software bridges would be hard to build without it. To give this superstructure maximum extensibility and move the reuse equation up one level, however, we must focus on expressing the architecture completely in terms of modeling concepts.

Another building block of this new architecture is a more concentrated focus on conformance testing, certification of programmers, and certification of products (branding). For this we will leverage the work of our current Analysis and Design Task Force, which has undertaken testing and branding projects relating to the UML, the MOF, XMI, and CWM, and is now working on the BOI and UML representation of Enterprise Application Integration (EAI). Ultimately, of course, the success of our efforts in these areas will depend on strong relationships with outside organizations with relevant expertise.

<sup>1</sup> The OMG calls these models *UML Profiles*. A number of these profiles are already well along their way to standardization.

<sup>2</sup> In technical terms, it is a *metamodel* of the category.

