Model Driven Architecture

by Richard Soley and the OMG Staff Strategy Group

Object Management Group
White Paper
Draft 3.2 – November 27, 2000

Preface: OMG's Accomplishments

It's about integration. It's about interoperability. For eleven years, the Object Management Group (OMG) has focused on making sure that you can integrate what you've built, with what you're building, with what you're *going* to build. As the pace of technology continues to quicken, and the demands of integrating your existing legacy systems, your new intranet, and your e-business fall on your shoulders, you need an architecture that makes interoperability central to your infrastructure. The bad news is that there will never be a single operating system, single programming language, or single network architecture that replaces all that has passed; the good news is that you can still manage to build systems economically in this environment.

Infrastructure Standards

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

Fred Waskiewicz, OMG Director of Standards

The OMG has led the way in providing vendor- and language-independent interoperability standards to the enterprise. Our mission started with a focus on CORBA which took its most recent shape in 1999 with the adoption of the CORBA 3.0 specification. To the 1995 Internet-based architecture of CORBA/IIOP, CORBA 3.0 added component-model technology that pulled together the heterogeneous computing platforms of the end of the century: the Internet, Java, Windows, SQL back ends, and a thousand other technologies. While this work was going on, other additions to CORBA extended it into environments that require specialized Real-Time, Fault-Tolerant, and Embedded systems. CORBA is a huge success in these markets, and continues to be expanded down (into automotive and aerospace embedded, smart transducer markets) and up (into fault-tolerant, failover, worldwide integrated enterprise customer solutions).

CORBA, the CORBAservices, and Domain Facilities are an integral part of computing today, playing a key role in millions of installations everywhere. Nearly all the application server products on the market are based on OMG standards, and many of the thousands of OMG success stories are documented on our website at www.corba.org. CORBA programming skills are in demand; a search of the on-line job market returns thousands of job listings requesting CORBA knowledge and experience.

Since its inception, the goal of the OMG has been to enable a *global information appliance* – that is, to make it just as easy to plug into information as it is to plug into power. Global travelers know that, although they may need an adapter or converter to plug into the mains as they move from country to country, no fundamental difference will keep them from using the power that

passes through the cord. By providing transparency to operating system, programming language and even network protocol, CORBA provides the adapter for the information plug, breaking down barriers to interoperability at enterprises around the world.

Modeling Standards

UML, the Unified Modeling Language, allows a model to be constructed, viewed, developed, and manipulated in a standard way at analysis and design time. As blueprints do for an office building, UML models allow an application design to be evaluated and critiqued before it is coded, when changes are easier and less expensive to make. In addition, the Common Warehouse Meta-model (CWM) standardizes how to represent database models (schema), schema transformation models, OLAP and data mining models, etc. The Meta-Object Facility (MOF) standardizes a facility for managing models in a repository. And finally, XML Metadata Interchange (XMI) is an interchange format for models, based on the MOF and the popular language XML.

UML is the product of years of experience in how to model software systems.

Andrew Watson, OMG Vice President of Architecture and Technical Director

The OMG's modeling specifications complement but not limited to CORBA. They are being used in virtually every development environment including Java/EJB, message-oriented middleware, DCOM/COM+/.NET and XML/SOAP.

The World's Best Standardization Process

In addition to its technologies, OMG has the *world's best standardization process*. Executing our process, building consensus as they converge on technical details, OMG members produce each new specification in nine to seventeen months. Proven in the marketplace with the standardization of CORBA, the CORBAservices, Domain Facilities, UML, the MOF, and OMG's other modeling standards, the process is one of our group's major assets. OMG's technology adoption process will play a major role in the extended standardization work proposed in this paper.

The Problem We Face: Middleware Proliferation

What's middleware? Middleware environments started out providing interoperability using architectures that are standard (i.e., CORBA), proprietary (e.g., DCOM or MQseries), or somewhere in the middle (e.g., JMI or HTTP/XML/SOAP). Essential services such as transactions, directory, persistence, event handling, and messaging, were added over time. Recently, more powerful middleware has emerged that builds on the basic interoperability environments to make it easier to construct transactional enterprise components (CORBA Component Model, EJB, MTS/COM+) that execute on application servers.

Over the past decade or more, companies have endured a succession of middleware platforms.

Jon Siegel, OMG Director of Technology Transfer

It's difficult – in fact, next to impossible – for a large enterprise to standardize on a single middleware platform. Some

enterprises found themselves with more than one because their different departments have different requirements, others because mergers or acquisitions created a mix. Even the lucky enterprise with a single middleware choice still has to use other technologies to interoperate with other enterprises and B2B markets.

The middleware environments that are most visible today are CORBA, Enterprise JavaBeans, message-oriented middleware, XML/SOAP, COM+ and .NET. However, over the past decade or so, the middleware landscape has continually shifted. For years we've assumed that a clear winner will emerge and stabilize this state of flux, but it's time to admit what we've all suspected: The sequence has no end! And, in spite of the advantages (sometimes real, sometimes imagined) of the latest middleware platform, migration is expensive and disruptive. (We know an industry standards group that, having migrated their standard infrastructure twice already, is now moving from their latest platform *du jour* to XML.)

At the beginning of the new century, OMG has a middleware solution that works: CORBA. Today CORBA is widely used to communicate over the Internet, integrating applications 24 hours a day, seven days a week. OMG's layered services and vertical market specifications build on CORBA, and are strongly established through the OMG community process. CORBA is the only middleware platform that is both vendor- and language-independent.

Nevertheless, enterprises have applications on different middleware that simply *have* to be integrated even though this process is time-consuming and expensive. Furthermore, the middleware they use continues to evolve.

To make matters even more complicated, different technologies have been used depending on whether communication is within or beyond the firewall. A component built on the assumption that it communicates within a firewall might have to be exposed beyond the firewall for B2B e-commerce, and a component exposed via an extranet might be moved behind the firewall because of an acquisition or merger. Thus, in addition to the basic integration problem, the IT organization must find a way to preserve the development investment made in new components as enterprise boundaries shifts and the underlying technology must change accordingly.

Addressing the Problem: Model Driven Architecture

Companies that adopt the MDA gain the ultimate in flexibility: the ability to derive code from a stable model as the underlying infrastructure shifts over time. ROI flows from the reuse of application and domain models across the software lifespan.

Dr. Richard Soley, OMG Chairman and CEO

There is a way to manage this situation, and it's based on the core modeling standards from OMG. What we have is the ability to *apply modeling technology to pull the whole picture together*.

Figure 1 lays out the **Model Driven Architecture** (MDA), which is language-, vendor- and middleware-neutral.

The core of the architecture, at the center of the figure, is based on OMG's modeling standards: UML, the MOF and CWM. There will be multiple *core models*[1]: 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 but the total number will be small. Each core model will be independent of any middleware platform. The number of core models stays small because each core model represents the common features of *all* of the platforms in its category.[2]

Whether your ultimate target is CCM, EJB, MTS, or some other component or transaction-based platform, the first step when constructing an MDA-based application will be to create a platform-independent application model expressed via UML in terms of the appropriate core model. Platform specialists will convert this general application model into one targeted to a specific platform such as CCM, EJB, or MTS. Standard *mappings* will allow tools to automate some of the conversion. In Figure 1, these target platforms occupy the thin ring surrounding the core.

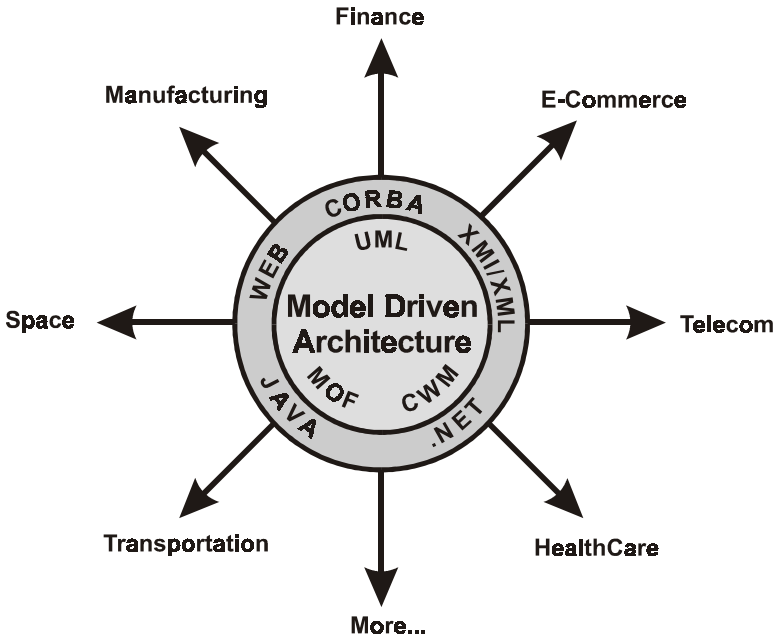

Figure 1: OMG's **Model Driven Architecture**

[1] In OMG's terminology, these models are *UML Profiles*. A number of these profiles are already well along their way to standardization at OMG.
[2] In technical terms, it is a *metamodel* of the category.

When we map a platform-independent model to a particular platform, we produce not only artifacts native to that platform (IDL, deployment descriptors, etc.) but also a platform-specific UML model. We do this because the UML model can express the semantics of the platform-specific solution much more richly than IDL or XML.

David Frankel, Chief Scientist, IONA/Genesis and
OMG Architecture Board Member

A separate core model (UML profile) for real time applications is in the works right now and I think that the mappings from that to platforms will be quite different than mappings of the Enterprise Computing model will be. An entity in a model based on the Enterprise Computing core model will map to a persistent, transactional component, whereas an entity in a real time model may map to something quite different.

C. Douglass Locke, VP-Technology, TimeSys Corp. and
OMG Architecture Board Member

Maximizing automation of the mapping step is a goal; however, 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, the amount of intervention required will decrease.

The platform-specific model faithfully represents both the business and technical run-time semantics of the application. It's still a UML model, but 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.

We can generate a lot of the code and reduce the need for hand programming by an order of magnitude, and in some restricted cases we will be able to generate all of the code.

David Frankel, Chief Scientist, IONA/Genesis and
OMG Architecture Board Member

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 will be substantial or, perhaps in some cases, even complete. 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, because they will be using a consistent architecture for managing the platform-independent and platform-specific aspects of their applications.

Modeling as much of the application semantics as precisely as possible enables a greater degree of code generation.

Sridhar Iyengar, Fellow, Unisys Corporation and
OMG Architecture Board Member

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 provides the best middleware choice because it is vendor- and language-neutral, and bridges easily to all of the other middleware environments. But, to accommodate those enterprises with multiple middleware platforms on their network, many non-CORBA platforms will be incorporated into the MDA. One of the first will be the Java-only EJB.

Because the MDA is platform-independent at its core, adding new middleware platforms to the interoperability environment is straightforward: After identifying the way a new platform represents and implements common middleware concepts and functions, OMG members will incorporate this information into the MDA as a mapping. Various message-oriented middleware tools, XML/SOAP and .NET will be integrated in this way; going one step further, by rationalizing the conflicting XML DTDs that are being proposed in some industries, the MDA can even help you interoperate across them.

> *Some of the transformation standards in the CWM can be used with quite an effect to help solve the DTD transformation problems that the XML-specified interface standards will inevitably run into.*
>
> Jishnu Mukerji, System Architect, HP Corporation and OMG Architecture Board Member

As representations of multiple middleware platforms are added to the MDA and mature over time, 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 basis on a separate core model for each category; differences between application categories prevent us from basing all of our applications on a single core model. By identifying and exploiting concepts common to two or more categories, even these boundaries can be smoothed somewhat.

Our discussion so far has assumed that we were building our application – and its model – from scratch. Legacy applications do not fit this pattern. With many built before component environments were even conceived, applications in this category typically will not fit smoothly into any of our core models. However, legacy applications may be brought into the MDA by wrapping them with a layer of code consistent with the MDA Enterprise component core model, or another core model if appropriate. If we build an MDA model of the wrapper first, the outer portion – that faces the network and interoperates with our other applications and services – may be generated automatically at least in part, although the other side of the wrapper – that invokes and returns from the legacy application itself – will typically have to be hand coded.

> *Legacy wrappers will have to do all sorts of things that require programmer intervention and can't be rolled into a generic mapping rule, such as making LU 6.2 calls to mainframe transactional systems.*
>
> Tom Rutt, DMTS, Global Strategic Standardization, Lucent, and OMG Architecture Board Member

As the next generation of OMG standards, the MDA plays the role of the next generation *Internet ORB*, integrating across *all* middleware platforms – past, present, and future. OMG, the organization that knows ORBs better than any other, is the ideal organization to extend this concept beyond middleware standards to a middleware-neutral, model-driven approach. MDA users gain these specific advantages:

- You will be able to build new MDA-based applications using the middleware of your company's choice. You will have the security of knowing that the essential semantics of your application have been systematically distilled in the form of a platform-independent model and that any future need to migrate to different middleware (or even new versions of the same middleware) will thus be reasonably manageable. Interoperability bridges and gateways to other MDA-based applications within your enterprise as well as interconnections with customers, suppliers, and business partners can be produced methodically, using a consistent architecture and some degree of automatic generation.
- Your legacy applications – that is, the ones that keep your business in business – will interoperate with your current applications once you wrap them as we described and incorporate their functions into the MDA. You can leave them on their established platforms; the MDA will help automate the construction of bridges from one platform to another.
- Industry standards – in your industry and others – 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 in the next section of this paper.
- 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 expand the list of platforms that they can target when they assist in converting a platform-independent model. The tools will also be able to expand their support for bridges to incorporate the new platforms.
- *Developers gain the ultimate in flexibility, the ability to regenerate code from a stable, platform-independent model as the underlying infrastructure shifts over time. ROI flows 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 declaration of semantics of systems (through modeling) will increase software quality and extend the useful lifetime of designs (and thus return on investment).

Taking advantage of our standards and tools that exploit them, OMG members have this integration task well underway. The Enterprise Computing core model is being defined and mapped to the most-used middleware platforms. A core model is also being defined for Real Time computing.

There is another dimension of instability that the relatively stable platform-independent model protects you from: *Shifting enterprise boundaries*. The IT manager's challenge is how to preserve the development investment made in new components when a boundary shifts and the underlying technology must change. Stable models address this challenge.

Andrew Watson, OMG Technical Director and Vice President of Architecture

Standardizing Domain Models

Since January 1996, a sizeable percentage of OMG members have been meeting in Domain Task Forces, communities focused on standardizing services and facilities in specific vertical markets. Until now these specifications have consisted of interfaces written in OMG IDL with accompanying semantic description in English text. Standardizing components at a platform level, in terms of standards such as CORBA, is certainly a viable contribution to solving the integration and interoperability problem, but we can do something additional.

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 distilled explicitly. OMG's domain specifications fall into this category because the model for virtually every one is *not* expressed separately from its 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 Healthcare Resource Access Decision Facility, already implemented in Java and EJB in addition to CORBA, is an example. There are more.

Thus, in order to maximize the utility and impact of OMG domain facility specifications in the MDA, they will be in the form of normative, platform-independent UML models augmented by normative, platform-specific UML models and interface definitions for at least one target platform. The common basis on MDA will promote partial generation of implementation code as well, but implementation code of course will not be standardized.

Today OMG has ten Domain Task Forces with several more "in the chute;" more are added from time to time. Rather than show them all in a static diagram, we've only included a representative sample in Figure 1 where they appear as rays emanating from the center.

The Domain Task Forces (DTFs) produce standard frameworks for standard functions in their application space. For example, a Finance DTF standard for an accounts receivable facility might include a platform-independent UML model, a CORBA-specific UML model, IDL interfaces, a Java-specific UML model, and Java interfaces. XML DTDs or schema generated via XMI-based mapping rules could be included as well. All of these artifacts would be normative. Such a standard would have broad impact, in that the platform-independent model would be useful even in middleware environments other than those targeted by the platform-specific parts of the specification. Since accounts receivable is an Enterprise Computing application, the normative, platform-specific artifacts would be derived at least partially via standard mappings of the Enterprise Computing core model to the platforms.

As another example, the Manufacturing DTF could produce normative MDA UML models, IDL interfaces, Java interfaces, XML DTDs, etc. for (Figure 2) CAD/CAM interoperability, PDM (Product Data Management), and a Supply Chain integration facility. Once the MDA models for these have been completed and adopted, their implementation can be partially automated in any middleware platform supported by the MDA.

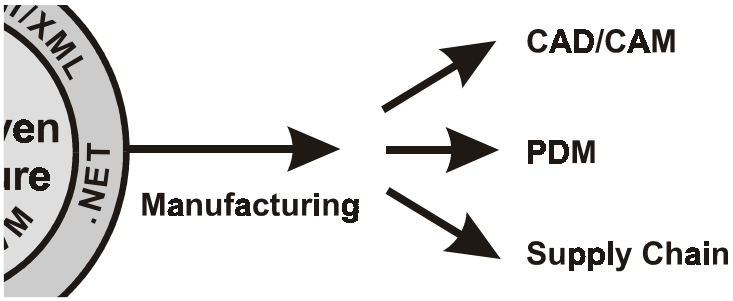

Figure 2: UML models of frameworks for vertical facilities

The three facilities that we're using for this example – CAD/CAM, PDM, and Supply Chain – demonstrate a benefit that only the MDA can provide: CAD/CAM and PDM applications are tightly integrated and so will probably be implemented by an individual enterprise or software vendor in, for example, CORBA or EJB. Supply Chain Integration, on the other hand, is more of an inter-enterprise facility so we might expect an XML/SOAP based implementation supported by an industry market-maker or trade organization to become popular. But, it will be essential to interoperate among the three: CAD/CAM designs feed into PDM production facilities which drive the supply chain; in turn, the supply chain will refer back to CAD/CAM for details on a particular part. By starting all three out as UML models in the MDA, we may eventually be able to generate a significant portion of the implementation of each on its preferred platform, as well as the bridges we need to integrate each of the facilities with the other two.

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 in a confined environment (Embedded).

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 visible only 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, code will be generated (or dynamically invoked) that makes the calls to the native services of those platforms. The pervasive services would be visible only to lower-level applications, i.e. applications that write directly to services.

Hardware and software attributes – Scalability, Real-Time, Fault Tolerance, or Embedded characteristics – may be modeled as well. By defining UML representations for these environments or, in the case of Fault Tolerance, an environment that combines it with Enterprise Computing, OMG will extend the MDA to support and integrate applications with these desirable characteristics.

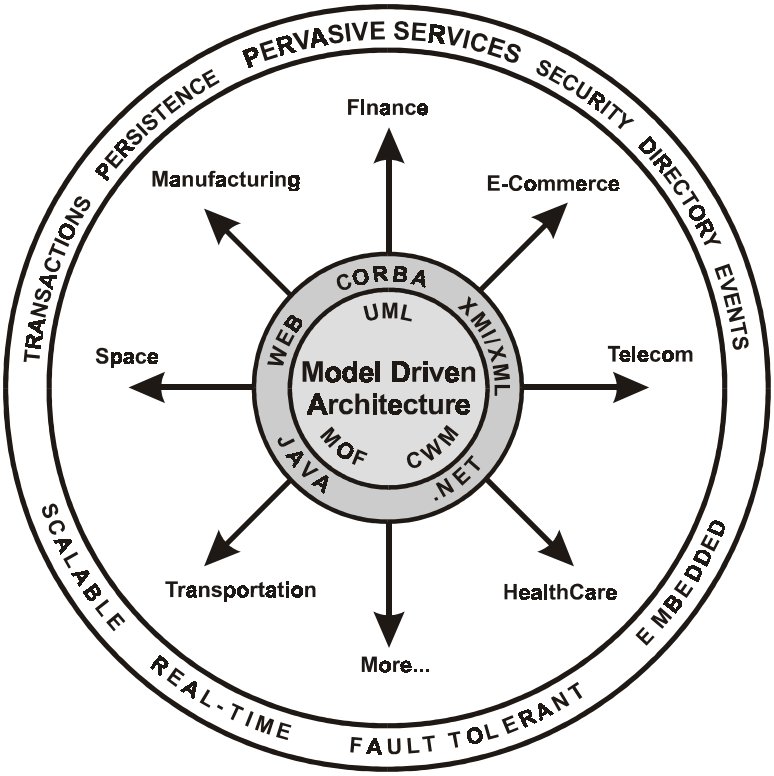

Figure 3: MDA showing the Pervasive Services and specialized computing environments.

In Figure 3, we've shown the Pervasive Services as a ring around the outside of the diagram to emphasize that they're available to all applications, in all environments. True integration requires a common model for directory services, events and signals, and security. By extending them to a generalized model, implementable in the different environments and easily integrated, the Model Driven Architecture becomes the basis of our goal of universal integration: the *global information appliance.*

What You Can Do

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 – RFPs have been issued for UML 2.0, and all of the components of the 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 been neither constructed nor adopted.

Application models defined in this environment by OMG's Domain Task Forces 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 if you're not doing it already. If you are a provider, this maximizes your impact on the future standards at the same time as it identifies you as a key player; if you are a user, this lets you put your company's requirements into the RFP that defines the new standard. This is your opportunity to have your voice heard in the models and standards that you will end up using; you will be working with the best and the brightest in the industry to develop the architecture of choice, which

I think the requirements for business software will continue to evolve faster than the technology solutions and that business developers will continue to have "programming" jobs for the rest of my career.

Carol Burt, 2AB, Inc., and OMG Architecture Board Member

- Integrates what you've built, with what you're building, with what you will build in the future;
- Remains flexible in the face of constantly changing infrastructure; and
- Lengthens the usable lifetime of your software, lowering maintenance costs and raising ROI.

Expanding the Role of the Object Management Group

We are continuing our quest to support integration and interoperability across heterogeneity at all levels. Our first mission, to enable integration through the introduction of the distributed object model, is complete: objects are the way systems are built today, at the core of every vendor's enabling architecture and also at the core of all e-businesses. But the *integration* task is not yet done. To focus on this, OMG will extend our focus from a middleware-centric to a modeling-centric organization.

To support this effort, the OMG must also concentrate extra effort on conformance testing, certification of programmers and certification of products (branding). While OMG has been involved in the past with various testing & branding efforts for its standards, and is currently building programmer certification for the CORBA and UML technologies, the expanded role of the OMG must be built on rock-solid testing, certification and branding. In many cases these efforts will depend on strong relationships with outside organizations with relevant expertise. Focusing on this problem is critical to the success for OMG's new role.

The foundation for the new role of the organization will be the current Analysis and Design Task Force and its work on UML, the MOF, XMI, and CWM, along with work in progress on the BOI and UML representation of EAI.

Today, companies whose submissions are adopted by OMG members as our standard must agree to market or commercially use an implementation of the specification. Under MDA, the commercial availability requirement will remain, ensuring that OMG standards remain relevant to the marketplace.

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. Extending this superstructure, the MDA is expressed completely in terms of modeling concepts, moving the reuse equation up one level.

Conclusion

The need for integration remains strong and OMG's work is not yet complete – we need to build on the success of UML and CORBA. With our experience, cohesive membership, established process, and tradition of consensus-based decision-making, OMG is in the ideal position to provide the model-based standards that are necessary to extend integration beyond the middleware approach. We've outlined the task, and directions of the solution, in this paper. Now is the time to put this plan into effect. Now is the time for the Model Driven Architecture.