component development strategies

The Monthly Newsletter from Cutter Information Corp. on Managing and Developing Component-Based Systems

Executive Summary
January 2002

It gives me great pleasure to welcome back Paul Harmon, my predecessor as editor of **CDS***, in the role of guest author this month. I can think of no one better equipped to focus on this month's topic: the OMG's Model Driven Architecture (MDA).*

While today's software projects share much in common with their ancestors, one key difference is that projects today are seldom pure development projects: investments in existing systems must be realized and advantage must be taken of an increasing array of commodity software available on the open market. Mr. Harmon explains that the significance of MDA is in providing much needed help in addressing these challenges by supplying an approach to integrating all of the various middleware approaches, languages, and application types that companies have to support.

We begin with a historical overview, clarify the main principles of MDA, and then go on to explain how MDA is heralding a renaissance in the modeling tool space. Mr. Harmon then describes a case study in the uptake of MDA from Wells Fargo bank and reviews the emerging MDA tools market, before wrapping things up with a survey of future trends.

Contents

The OMG's Model Driven Architecture. . . . 1
MDA. 2
MDA and CASE. 5
Wells Fargo's BOS Framework 7
MDA Tools 8

[illegible]

Paul Allen, Editor
E-mail:
pallen@cutter.com

THE OMG'S MODEL DRIVEN ARCHITECTURE

The Object Management Group (OMG) was founded to encourage the use of object technology and to establish standards that would facilitate the creation of distributed systems. Throughout the 1990s, the member companies that make up the OMG labored to create a set of standards, collectively known as the Object Management Architecture (OMA). The centerpiece of the OMA was Common Object Request Broker Architecture (CORBA), which defined how messages from diverse languages could be translated into a common intermediary language — the OMG's Interface Definition Language (IDL) — moved from client to target platforms, and then retranslated into the language of the target object or component.

When most people think of CORBA, they probably visualize the diagram that the OMG popularized during the 1990s. It shows a thick, double-headed arrow that represents the CORBA object request broker (ORB) and the services that keep track of the locations and names of the various distributed objects or components that might need to interact with one another. Above and below the CORBA arrow are the distributed applications and services (objects and components) that companies could use to create systems. In effect, CORBA defined and popularized a middleware-based approach to distributed systems development.

In 1989, when the OMG was established, its members hoped that they could create a single object-oriented (OO) middleware standard that everyone would embrace, thus greatly simplifying the problems involved in creating distributed software systems. Unfortunately, it never happened. Microsoft developed and promoted a proprietary alternative, DCOM. Sun developed Java, which came with its own ORB, Remote Method Invocation (RMI). And, more recently, the Internet has resulted in yet another set of middleware standards based on the Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP), which Microsoft, IBM, and Sun have all embraced.

At the same time that middleware options were proliferating, the move toward e-business has placed companies under increasing pressure to integrate their systems. To create Web-based applications that allow customers to access a variety of corporate databases and applications, developers must find a way to link all of their databases and applications together in a way that can provide Web users with almost instantaneous responses. Consider a banking application that lets a user check his or her checking account. Once the balance is checked, the same user may create electronic transfers that move money from the account to a variety of payees. In the next instant, the same user may decide to move money to or from a savings account, or shift money to pay a bank credit card balance. In a matter of minutes, our user has accessed three or four major bank databases and used three or four different legacy applications to update accounts and transfer funds. None of these applications was originally designed to work with the Internet, and they certainly weren't designed on the assumption that hundreds or thousands of users might seek to use them more or less simultaneously.

To create new Web-based e-business systems of this kind — and business-to-business (B2B) supply chain systems can be much more complex — companies find themselves struggling to integrate diverse databases and applications written in different languages. At the same time, they also find themselves struggling to get different middleware systems to talk with one another in an efficient manner. Enterprise application integration (EAI) is one of the key challenges of this decade, and, in many cases, various middleware APIs are part of the problem.

In 1999, the 800 member companies that make up the OMG began to consider moving beyond CORBA and contributing to a better solution of the integration problem. It was clear that CORBA would never be able to replace all of the other middleware standards companies were using. Companies will have to learn to live with and integrate a variety of operating systems, languages, and databases, as well as a variety of middleware standards. Faced with this reality, the OMG began to consider a new approach to architecting distributed systems.

In the late 1990s, the OMG had sponsored the development of an integrated software modeling approach, the Unified Modeling Language (UML). Unlike CORBA, with its dependence on IDL, UML is completely independent of any programming language or middleware conventions. Using UML, you can create a high-level diagram that describes how two applications could be linked together without concerning yourself with any of the details involved in implementation. Starting from that high-level UML architecture, a developer could proceed to create a design that implemented the linkage in DCOM or in CORBA. Similarly, you could just as well specify that the linkage be implemented in Java, Visual Basic, Enterprise JavaBeans (EJB), or XML. The key lies in the fact that UML doesn't rely on APIs but can be used to generate any given set of APIs.

MDA

After considering some of the pioneering work by member organizations, the OMG decided to move toward adopting a new architecture, which it named the Model Driven Architecture (MDA). In effect, the OMG decided to raise the level of abstraction and focus on describing how the system should be integrated.

This is not to suggest that the OMG is abandoning CORBA. Rather, it will continue to support CORBA and its OMA, while simultaneously developing a second, more comprehensive, UML-based architecture. A developer using the MDA approach begins by creating a UML model that describes how the system should be integrated. (MDA terms this a platform-independent model, or PIM.) Later, this model is used to structure the actual development of code to facilitate the integration described in the PIM. (The actual design is referred to as a platform-specific model, or PSM.) The OMG plans to define a set of profiles to ensure that a given UML model can consistently generate each of the popular middleware APIs.

Editorial Office: Paul Allen, Tel: +44 1737 813911; Fax: +44 1737 814279; E-mail: pallen@cutter.com.
Circulation Office: ***Component Development Strategies***® is published 12 times a year by Cutter Information Corp., 37 Broadway, Suite 1, Arlington, MA 02474-5552. Tel: +1 781 641 9876 or, within North America, +1 800 492 1650; Fax: +1 781 648 1950 or, within North America, +1 800 888 1816; E-mail: service@cutter.com; Web site: www.cutter.com/consortium/.
Editor: Paul Allen. Publisher: Karen Fine Coburn. Group Publisher: Kara Letourneau, +1 781 641 5126, E-mail: kletourneau@cutter.com.
Production Editor: Pamela Shalit, Tel: +1 781 641 5116. Subscriptions: $497 per year; $567 outside North America. ©2002 by Cutter Information Corp. ISSN 1522-7391. All rights reserved. Unauthorized reproduction in any form, including photocopying, faxing, and image scanning, is against the law. Reprints, bulk purchases, past issues, and multiple subscription and site license rates are available on request.

When the OMG began to work on CORBA in 1990, it started by describing a set of general principles. Over the course of a decade, the principles were converted into specific standards and then implemented in a variety of tools. The OMG is a standards organization and doesn't involve itself in the creation or sale of code. Instead, the companies that are members of the OMG convert OMG standards into actual tools and applications.

In early 2001, the OMG described a set of general principles for MDA. Unlike CORBA development, which started from scratch, many of the elements of the new MDA architecture have already been defined by the OMG and implemented by vendors. The UML specification, for example, was first released in 1997. Several dozen companies have produced software tools that support some or most of the diagrams defined by the UML specification.

Since UML was first released, the OMG has continued to refine and extend it. UML is currently in release 1.4 and will move to version 2.0 sometime in 2002. In addition to extending and refining the UML diagramming conventions, the OMG has done a number of other things to improve UML. Foremost among them, the OMG has created a metamodel for modeling languages, which it called Meta Object Facility (MOF).

UML describes diagrams that are used to model specific software applications. Thus, for example, UML defines a class diagram that can be used to illustrate how classes can be assembled. When you use the class diagram notation to describe an actual class model of an accounting application, you have created a model of the accounting application. The higher-level UML description of a generic class diagram is a metamodel. It's a model that can be used to describe any of thousands of specific software models. UML supports about a dozen different diagrams; for example, class diagrams, use case diagrams, sequence diagrams, activity diagrams, object instance diagrams, and package diagrams. Each of these diagram types is a different type of metamodel. MOF is a meta metamodel, which is used to describe all of the models making up UML. Broadly, MOF is a very abstract model that can describe any other type of model, including non-UML models (see Figure 1).

MOF is necessary for UML to ensure that each of the specific UML model types is defined in a consistent way. Thus, MOF ensures that a "class" in a class diagram has a precise relationship to an "activity" or to a "use case" in their respective diagrams. (In effect, each UML diagram is really just a "view" of the common, underlying UML metamodel.) At the same time, MOF is actually defined by a subset of the elements used in the UML metamodel. This can all seem a little overwhelming, but the net result is that MOF maintains that UML is a more precise software development modeling system than any of the earlier software methodologies. IBM, Oracle, Sun, and Unisys have all begun to incorporate MOF elements into their major systems to ensure that their various products can communicate with each other.

In the process of developing UML and MOF, several OMG members were impressed with the ability of these technologies to bring a greater degree of consistency to many other aspects of the OMG's standardization efforts. One task force redefined the OMG's CORBA IDL as a MOF-compliant model. This, in turn, means that a software tool could automatically convert a UML diagram into IDL code. In addition to generating MOF-compliant models, the OMG is also developing extensions of UML, called profiles. UML can be extended with new diagrammatic elements according to rules adopted by the UML and MOF task forces. Specific extensions are called stereotypes, and a set of extensions and tag values is called a profile. Thus, the OMG has defined a profile for CORBA and for IDL. Profiles can be designed to help translate UML diagrams into code and to assist in reverse engineering the code into UML in a consistent manner.

In a related effort, an OMG task force developed a standard for data warehouses, called the Common Warehouse Metamodel (CWM). CWM is a metamodel for data warehouses. At the same time, CWM is a MOF-compliant metamodel. In other words, CWM is defined in terms of MOF.

In still another effort, the OMG's MOF task force defined a XML system that uses rules to generate XML

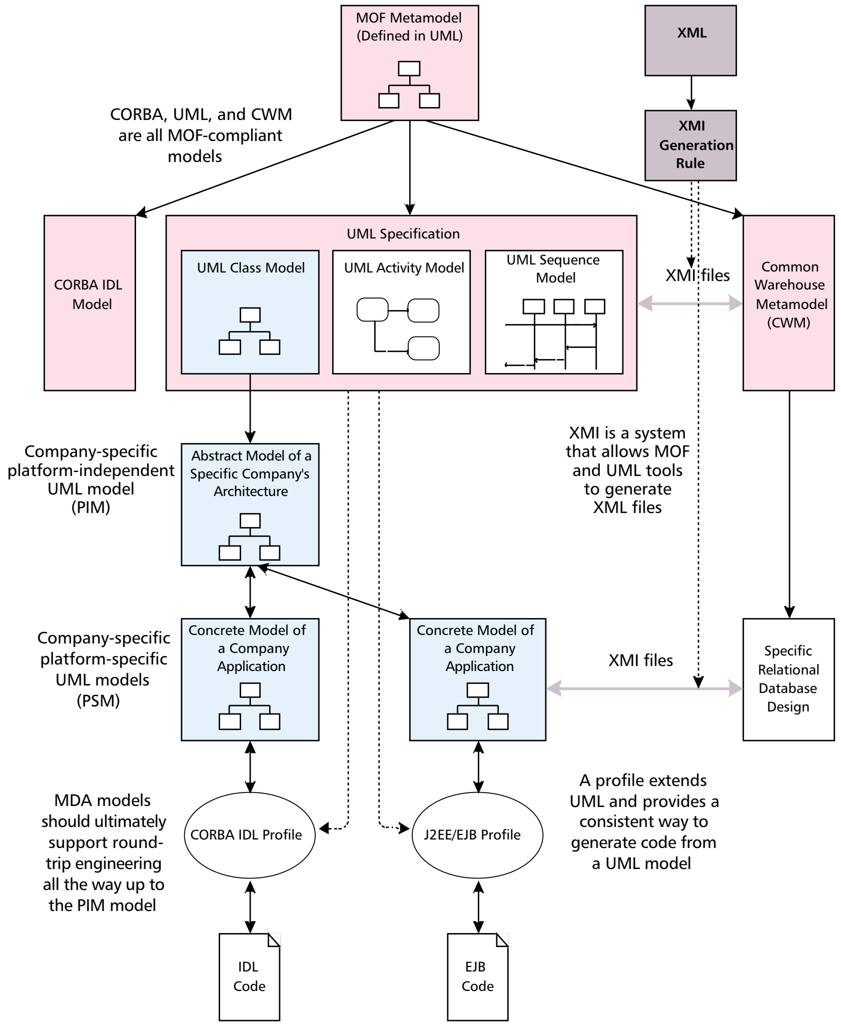
MOF Metamodel (Defined in UML)
XML
CORBA, UML, and CWM are all MOF-compliant models
XMI Generation Rule
CORBA IDL Model
UML Specification
UML Class Model
UML Activity Model
UML Sequence Model
XMI files
Common Warehouse Metamodel (CWM)
Company-specific platform-independent UML model (PIM)
Abstract Model of a Specific Company's Architecture
XMI is a system that allows MOF and UML tools to generate XML files
Company-specific platform-specific UML models (PSM)
Concrete Model of a Company Application
Concrete Model of a Company Application
XMI files
Specific Relational Database Design
MDA models should ultimately support round-trip engineering all the way up to the PIM model
CORBA IDL Profile
J2EE/EJB Profile
A profile extends UML and provides a consistent way to generate code from a UML model
IDL Code
EJB Code

Figure 1 — Some of the key relationships between models used in Model Driven Architecture (MDA).

files from any MOF-compliant model. The system is called the XML Metadata Interchange (XMI). In the next release of UML — once XMI support is added to a UML tool or a database — UML diagrams and the information contained within the diagrams can be moved from one tool or database to another via the Internet, using XMI. By the same token, using XMI, CWM products can generate XML files and pass them to UML tools, and vice versa.

All of these developments provide an idea of how the OMG was already working to provide more extensive support for UML and to standardize techniques that would make UML diagrams more dynamic and useful.

Most large companies throughout the world already use UML to design new enterprise applications. In most cases, the software development team begins by creating a use case model of the application and one or two generic business models, described via class diagrams or activity diagrams. Then the team moves on and elaborates these initial diagrams into a detailed design by adding more and more specificity to the UML diagrams.

The idea behind the OMG's MDA is simplicity. What the OMG proposes is that companies create high-level UML descriptions of how applications will be structured and integrated (PIMs). These descriptions will be independent of any actual implementation details. From the PIM model, a more constrained UML design, or PSM, can be generated. A PSM design can then be converted into language code designed for a specific platform. The final product, code for a specific platform, is termed an Enterprise Deployment Model (EDM). Put another way, PIMs can be used to generate any of several different PSMs, each of which could generate a different specific application (EDM). Essentially, MDA assumes that companies will create UML descriptions of applications and middleware links and then use those graphical descriptions to generate code.

Consider a concrete example: Each middleware language models a transaction in a slightly different way. But every transaction has a start and an end. At the PIM level, we use UML to specify a generic transaction. Then, at the PSM level, we refine the transaction so that it's appropriate for whatever language and platform we intend to use. Using this approach, a company could create a PIM and then generate several different PSMs, one for the European division that wants to use EJBs, one for the Japanese division that's using Java, and one for the US division that's using XML and .NET middleware.

Conrad Bock of Kabira Technologies explains it this way: "Model-driven development frees users from being bound to any particular vendor's APIs. For example, a UML-driven application server does not have a set of APIs to use; it has a model. Since the model is platform independent, it will work on any UML-driven application server. Users are not committing to a specific server when they put their mission-critical business knowledge into an MDA PIM model."

Obviously, if MDA is to fulfill Bock's ambitions, it requires that the OMG not only refine UML and its associated technologies, but also revise existing CORBA services and domain frameworks to provide MDA support. In each case, the existing standard, which is defined in CORBA IDL, will need to be redefined as an abstract UML model. This work has begun and will certainly take two or three years to complete. The MDA approach also assumes that MDA products will support reverse or round-trip engineering, so that any code modified by a developer can be used to change the UML diagram in the appropriate PSM. It further assumes that the revised PSM will be used to revise the PIM, assuring enterprise-wide consistency. The OMG has begun to refer to its MDA approach as a "20-year architecture," stressing that, while middleware and APIs may come and go, a company should be able to maintain a PIM for many, many years.

MDA AND CASE

Developers who have been around for a while will immediately recognize how similar MDA is to the CASE movement of the late 1980s. They are alike in the sense that they both seek to improve application development by automating some code generation tasks. But consider the differences: In the mid-1980s, developers were working with much less sophisticated modeling languages. Most software development was done in COBOL and was designed to run on IBM-compatible mainframes. The best of the CASE tools let developers create diagrams of applications and then generated the COBOL code and the relational database code necessary to implement the design. Because the models were less precisely defined, it was very hard to move from code to model; therefore, in most cases, once code was generated, and then modified by hand, the model was out of date.

Similarly, although there were several software modeling notations, no one had tackled the difficult problem of building a repository that could store the various modeling elements or translate one product's diagrams into models in another product's format. IBM started such an effort — the AD/Cycle Repository — at the end of the 1980s, but it abandoned that effort just as the CASE movement lost steam at the beginning of the 1990s. The CASE movement declined because companies were switching from COBOL and mainframe platforms to new languages, like C and C++; new platforms, like Unix servers; and new designs that relied on object and client-server architectures.

In effect, MDA proposes that companies revive the CASE concept, with several important differences. First, a much more precise software methodology is available. The widespread adoption of UML ensures that tools will use a common notation. At the same time, UML is so precise that the reverse engineering of code into UML diagrams is a common feature of several UML tools. Indeed, some products eliminate reverse engineering completely by generating a runtime application from UML and providing debugging capabilities at the model level. Another difference is that

MOF has already resulted in several repositories that can store UML diagrams and data. IBM, Oracle, and Unisys all offer MOF-compliant repositories, for example. At the same time, XMI makes sure that UML tools can pass diagrams back and forth over the Internet, between tools, or to repositories or databases without loss of meaning. In other words, most of the problems companies faced with CASE in the late 1980s have already been overcome.

What's more important, however, is that in most instances, today's companies will not use the MDA approach to actually model a complete application from scratch. Component reuse or framework-based solutions, including the use of packaged enterprise resource planning (ERP) applications and e-commerce component frameworks, provide a more efficient way of building large portions of any new application. At the same time, the PIM model generally does not need to be concerned with which objects are distributed, parallelism and the queuing of events, transaction boundaries, or security. MDA-based application servers can support these in lower-level models or, better yet, as deployment configurations. Instead of creating complete applications, today's companies will use UML models to define how subsystems are to be linked together into an integrated whole, and then they will generate the middleware code needed for their applications. Since UML is middleware-independent, the same high-level diagrams can be used to generate different middleware solutions for different aspects of a large-scale integration project. When you consider that the development of middleware and EAI code often takes more than half the time it takes to create a new application, automatic generation of middleware code could be a huge boon to enterprise developers.

Even though much has been done and systems have already been generated using MDA principles, there is still work for the OMG to do. In effect, MDA depends on software tools that will model distributed architectures and, in some cases, generate code from high-level specifications. To maintain consistency, and hence the ability to move a model from one tool to another, the code generation process will need to be standardized. In other words, a number of profiles will need to be defined. Task forces are working on profiles that will define standard ways to generate code for various popular languages. Sun and various other companies supporting Java are working on generating a profile that would generate EJB components from UML diagrams. The OMG and the Workflow Management Coalition (WfMC) are working on a profile that would allow UML to generate input for workflow tools.

Similarly, while UML currently provides a complete description of structure, it lacks one for behavioral semantics. Today's UML developers must currently rely on the UML Object Constraint Language (OCL), which is an incomplete way of describing what one might want a system to do. Without complete behavioral semantics, UML cannot generate all the code needed for any possible type of application. The OMG already has a task force working on this problem. The new specification, which is termed UML Action Semantics, will be included in UML 2.0 when it is released in the spring of 2002.

The OMG adopted the MDA in early 2001. It described the new architecture in terms of its existing standards. At the same time, the OMG has committed itself to create new standards that will extend the existing standards and make the MDA approach even more useful in the future.

Figure 2 illustrates the graphic the OMG uses to describe its new architecture. In the center are UML, MOF, and CWM: these are the core systems used to describe the high-level models. Using profiles, you can generate code for technologies described in the next ring, including CORBA (IDL), Java, .NET, XML, XMI, and other Web protocols like HTML. The OMG is also defining profiles for the various middleware services it developed during the 1990s. And, various OMG domain task forces will be creating industry-specific frameworks of objects and components that developers can use to generate specific types of applications.

In a sense, MDA is a return to earlier ideas, yet with a whole host of new standards, tools, and more modest goals. More important, it addresses one of the key problems every company faces as it struggles to adapt to the rapidly changing world of new technologies, new business process models, and new applications. Rather than trying to standardize on one language or middleware approach — in effect, on one set of APIs — the OMG suggests that companies standardize on a high-level architecture that will support the widest possible range of applications the company might want to build. After that, applications and middleware can be generated from the core architecture. As specific modules or components change, the interfaces can be regenerated from the core model in the same language or in a new language. When companies decide to adopt the next hot thing after XML — and there will undoubtedly be yet another new API system — the OMG will prepare a profile for the new API system. Meanwhile, however, the UML architecture should last for many years.

WELLS FARGO'S BOS FRAMEWORK

To provide a concrete idea of how MDA might work, consider how Wells Fargo has been generating middleware for the past few years. Wells Fargo is the fourth-largest bank in the US with a market value of US $85 billion. The organization has been using CORBA for almost a decade and has reaped significant results from a systematic, continuing commitment to a single, comprehensive middleware approach. Recently, however, it has begun to evolve a new architectural approach to ensure that it can integrate its CORBA middleware with other types of middleware.

Eric Castain, Wells Fargo's senior vice president for its Business Object Services (BOS) group, recently described Wells Fargo's architecture integration strategy during an OMG press briefing held in conjunction with the OMG's announcement of its new Model Driven Architecture. Castain explained that Wells Fargo was already "using some of the concepts that are in the principles of the MDA process" in the Wells Fargo BOS group. The Wells Fargo BOS organization was also described at a recent UML conference by Tony Mallia, a senior architect with CIBER, Inc., a consulting company that helped Wells Fargo develop its model-driven development approach. (For more information, check www.omg.org/mda or e-mail tmallia@ciber.com.)

Wells Fargo's BOS group is responsible for maintaining all of the middleware and the middle-tier servers that are used to link specific applications to legacy applications and data. To deal with the complexities it faces, Wells Fargo's BOS group has created a single, UML-based business model that describes its architecture. This PIM model is maintained by UML modeling tools that allow Wells Fargo developers to generate implementation code as needed. By maintaining its business model independent of any specific middleware technologies, Wells Fargo can integrate various technologies or shift from one technology to another as required, without the messy redesign that is required when the architectural design is an integral part of the middleware implementation.

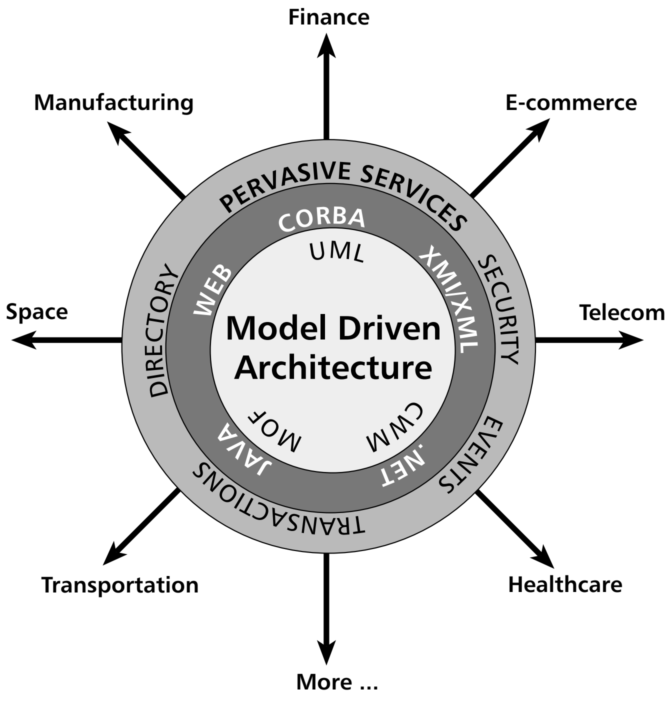

Figure 2 — The OMG's diagram of its Model Driven Architecture.

Figure 3 provides an overview of how the models developed by the BOS group are used to generate the middleware needed for new Wells Fargo applications. Wells Fargo's BOS business model is specified in UML and represented in Rational Software's Rational Rose modeling tool, supplemented by a set of extensions to Rose built by the BOS team. The business model provides a detailed, logical specification of the major software components and how they interact, including the messages passed between components. Wells Fargo is capable of generating OMG IDL structure code for

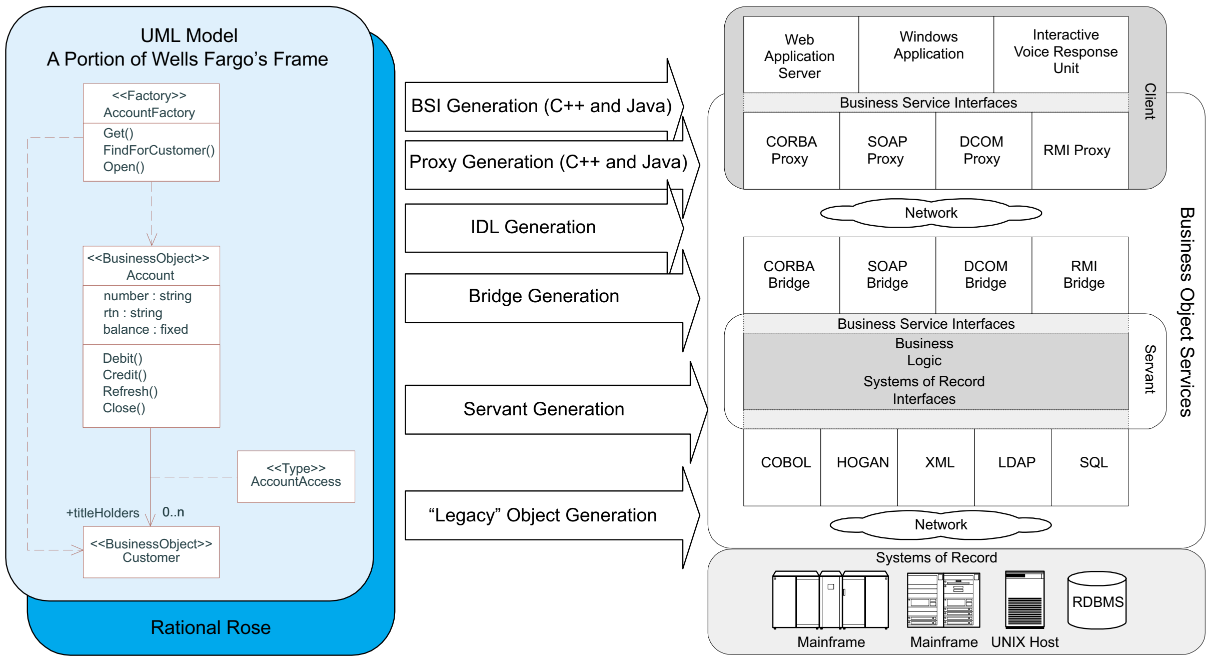

Figure 3 — Code is generated from the UML framework.
(After a figure used by Eric H. Castain during the OMG's MDA press briefing.)

any message in the model. In other words, Wells Fargo's BOS team has provided flexible interfaces in the face of inflexible systems of record (SOR), rapidly changing top-layer technology, and rapidly changing customer demand.

Wells Fargo's online applications are created by development teams that require the cooperative work of three groups: a client group (e.g., Internet banking), the BOS group, and the systems organization that maintains back-end systems and records. In effect, the BOS group contributes the UML-based business model that defines the core business objects, the interfaces, and the middleware infrastructure for the project. The actual application is created by reusing objects defined in the BOS infrastructure and, in some cases, extending those objects. The application developers create the UML diagrams that describe the specific applications they are developing. They then generate the IDL message code, which is then used to generate the CORBA stubs and repository entries that will be used in actually passing messages between application components.

The work that has been done at Wells Fargo provides a good example of a model-driven approach to software development. The BOS group focuses on maintaining the business object (PIM) model. Actual development groups use the BOS business object model to generate the specific or PSM models they need for their specific applications. Later, they use the design to generate the messages for the actual components they will use. BOS architects and application developers depend on UML to provide the common language for their efforts.

Castain explains it as such: "Our application development environment has been tailored to support the model-centric approach. First of all, we define the application in what we call a design model in UML. From that, we generate the implementation models that are actually needed for the application. These platform-specific models are then augmented with additional code and any additional logic that may be needed. We are not trying to generate a complete application, but only those portions that can be efficiently generated by a model. Lastly, we generate the actual application programs."

MDA TOOLS

So far, we've discussed MDA standards without considering how companies will actually implement

MDA. The Wells Fargo BOS system was developed before MDA was adopted by the OMG. Indeed, it is the success of several large systems, like the one developed by Wells Fargo, that convinced OMG members that the MDA approach was viable. Wells Fargo developed its BOS system using Rational's UML diagramming tool, Rose, and supplemented Rose with a component framework that Wells Fargo developers created in conjunction with consultants from CIBER. Some companies will be happy to create their own MDA systems in this manner. Most, however, will wait for software vendors to generate MDA products to assist them.

Just as dozens of vendors developed CORBA tools once the OMG published its CORBA standards in the early 1990s, today we are witnessing the beginning of a new software market for MDA tools. Moreover, just as the early CORBA tools varied in what elements of the CORBA standard they supported and how completely they implemented the standard, the early MDA tool vendors vary quite a bit in their support of MDA. This will change as time passes, as the OMG generates more complete MDA standards, and as companies become more experienced and more demanding in what they ask of MDA tools.

Looking at the MDA tools market as it currently stands, just six months after the OMG announced its MDA architecture, we can discern several different types of MDA tools. Some focus on the entire MDA lifecycle, from PIM to code generation. Some focus on just code generation and others on code reengineering. Some are essentially repository-based, while others are based on application servers. It's too early to offer a definitive definition of what constitutes an MDA tool. In essence, an MDA tool supports UML and associated technologies like MOF, XMI, CWM, and profiles. More important, however, since there are many graphic tools to create UML diagrams, MDA tools support the distinction between PIM and PSM models and provide users with the ability of UML to generate code. A number of companies offer products that have functioned as MDA tools for several years. In effect, they were using UML and other OMG standards to help clients generate applications before the OMG decided to officially name and formalize the MDA process. Others have just developed tools or modified tools to provide MDA support.

Predictably, as users become more demanding and as vendors see what is popular, the current tools will be enhanced to become more comprehensive. At this point, it isn't surprising that the tools are only first approximations of what MDA tools will be in upcoming years. What is impressive is that so many vendors have already announced MDA tools. When I started to research MDA in June, I could identify only three MDA vendors; today I know of 15, and a new vendor seems to appear each month. Clearly, MDA has stirred up a huge amount of interest and excitement among tool vendors, and the race is on to create good MDA tools.

Table 1 provides a list of MDA tool vendors, many of which I briefly describe later in this article. The list also includes some vendors that have announced MDA support but that have not provided me with a description of how their tools will function. IBM, for example, just announced its support for MDA and indicated that it would support MDA via WebSphere. I've been unable to obtain more specific information, however.

It's too early to evaluate MDA tools in any detail, but the following descriptions should provide readers with an overview of the early entries in the MDA tools sweepstakes. Hopefully, later in the year, I'll be able to write an issue of ***CDS*** that focuses on three or four of the tools in more depth and offers a more systematic analysis of the features an ideal MDA tool should offer.

Adaptive's Adaptive Framework

Adaptive sells a suite of products collectively known as Adaptive Framework. Adaptive Framework is a repository-based framework that builds on MOF and can manage information from all the MDA-related metamodels discussed. The specific elements include Adaptive Repository, based on the Unisys Universal Repository (UREP) technology, which Adaptive has taken a source license for and extended; Adaptive Workshop, an environment that provides the tools to customize a repository solution; and Adaptive Portal, a completely Web-based user interface for interacting with and communicating the models.

Adaptive also provides metamodels for linking MDA development into an enterprise-wide "big

Table 1 — Vendors That Are Selling or Have Announced MDA Tools

Company Web Site	MDA Product	Contact
Adaptive Bournemouth, UK www.adaptive.com	Adaptive Framework	Pete Rivett, CTO Pete.rivett@adaptive.com
Codagen Technology Montreal, Quebec, Canada www.codagen.com	Gen-it Architect	Michel Brassard, CTO Mbrassard@codagen.com
EBuilt, Inc./Codigo Solutions Irvine, California, USA www.ebuilt.com www.codigoxpress.com	CodigoXpress	Phillip Lindsay, Vice President Plindsay@ebuilt.com
Embarcadero Technologies San Francisco, California, USA www.embarcadero.com	Describe (next generation of GDPro)	Joy Lowry, Media Relations joy.lowry@embarcadero.com
Firestar Software (Formerly ONTOS) Andover, Massachusetts, USA www.firestarsoftware.com	ObjectSpark	Mark Eisner eisner@objectspark.com
Headway Software Somerville, Massachusetts, USA www.headwaysoftware.com	Headway reView	Tom O'Sullivan, Vice President Marketing Info@headwaysoftware.com
IBM www.ibm.com	WebSphere	Stephen A. Brodsky, WebSphere MDA Architect sbrodsky@us.ibm.com
Interactive Objects Software GmbH Freiburg, Germany www.io-software.com	ArcStyler (being incorporated in Borland's Enterprise Studio 2)	Richard Hubert, CEO Richard.Hubert@io-software.com
Kabira Technologies, Inc. San Rafael, California, USA www.kabira.com	ObjectSwitch, Kabira Business Accelerator, and others	Grover Righter,Vice President Tech Strategy Grover.righter@kabira.com
Kennedy Carter Ltd. Guildford, Surrey, UK www.kc.com	iUML and I-CCG	Ian Wilkie, Tech Director Ian@kc.com
ObjeXion Software Vieux-Thann, France www.objexion.com	Netsilon	Pierre-Alain Muller pa.muller@objexion.com
Metanology Alpharetta, Georgia, USA www.metanology.com	MDE	Jeff Bergstrom, Vice President Marketing JeffBergstrom@msn.com
Project Technology Tucson, Arizona, USA www.projtech.com	BridgePoint and DesignPoint	Stephen J. Mellor, CTO and Chairman Steve@projtech.com
Secant Technologies Cleveland, Ohio, USA www.secant.com	ModelMethods	Keith Rupnik, Product Management sales@secant.com
Softeam Paris, France www.objecteering.com	Objecteering/UML	Joan Le Bris, Marketing and Communication Manager joan.lebris@softeam.fr
TogetherSoft Raleigh, North Carolina, USA www.togethersoft.com	Together ControlCenter 5.5	Alison Freeland, Director Public Relations alison.freeland@togethersoft.com
TechOne, Inc. Oakland, California, USA www.techone.com	ACE	Vipul Singhal, President vipul@techone.com

picture," which also includes goals and objectives, organization structure, software deployment, and the management of reusable assets, like components. By linking the MDA artifacts into a complete development lifecycle, Adaptive's software can provide traceability and end-to-end impact analysis. It also supports central cataloging and searching of artifacts, versioning, access control, and management of the development process.

Adaptive Framework provides an open platform based on MOF, XMI, and the forthcoming Java Metadata Interface (JMI — a Java version of MOF). It functions as an integration hub to bridge different tools that might not otherwise be able to communicate directly. It allows analysis and transformations to be carried out directly in the repository.

Codagen Technologies' Gen-it Architect

Codagen's Gen-it Architect provides an architecture-centric solution that supports the construction of software applications for the .NET and Java platforms. Code of a delivered system can be classified as either value-added — fulfilling functional requirements — or as structural — supporting the system architecture, technologies, framework, or the execution platform choices. Structural code, providing rich, powerful, and robust services to programmers, represents large and complex portions of a system. The Codagen Gen-it Architect software tool addresses this emerging MDA-driven market for a given UML PIM of an application by creating an architectural blueprint that provides a loosely coupled approach between the architecture decisions and their implementation.

Changes can be made to the implementation of a given technology or platform without having to modify the UML PIM. Gen-it Architect allows the architect to capture the best practice implementation for each service by creating XML-based templates and storing them within a blueprint. UML-tagged values capture implementation data needed for different technologies or platforms. Gen-it Generator applies the template-based generation rules within the blueprint to the model. As pure source code, the result can be deployed on any platform without constraints of runtime licenses or helper components. Gen-it Architect provides adapters for UML modeling tools like Rational Rose, Together/J, and Visio-UML, and for integrated development environment (IDE) tools like VisualAge for Java and Visual Studio .NET. The commercial version of Gen-it Architect generates Java code and XML schema, while a beta version for the .NET platform supports C# and Visual Basic. Derivative products let developers visually edit or create XML schema and B2B workflows for XLANG, ebXML BPSS, and WSFL.

EBuilt/Codigo Solutions — CodigoXpress

CodigoXpress is a UML-centric code automation tool designed to be utilized in highly iterative development environments. Consistent with the OMG's MDA approach, CodigoXpress starts with a PIM in XMI format and then generates high-quality, technology-specific implementations. CodigoXpress provides comprehensive support for the construction step in MDA by generating, structuring, building, and deploying code. The code generated by CodigoXpress is platform-specific and, therefore, corresponds to the MDA notion of an EDM. Moreover, the generated code is optimized for each of the targeted platforms. For example, specific patterns are followed for object-to-relational mapping, creating optimized SQL/stored procedures as well as deployment scripts for an Oracle database; other patterns are followed for an IBM DB2 implementation. Other patterns are used to generate code for BEA WebLogic or IBM WebSphere Application Server, and so on.

CodigoXpress is an MDA tool that supports UML, is associated with other standards such as XMI and Java 2 Enterprise Edition (J2EE), and is able to generate code from UML models. It is well on its way to becoming a full-scale MDA tool in that it distinguishes between PIM and EDM models and has support for model reverse engineering and round-trip engineering. CodigoXpress does not currently have a distinct representation of a PSM. Rather, it uses an innovative meta-object language to create Implementation Pattern Language (IPL) to describe platform-specific implementation patterns that are parsed and applied during code generation. This creates deployable components that are specialized and optimized toward user-selected target platforms. Thus, IPL essentially provides a vehicle to capture and immortalize high-quality, real-world implementations of technology. Given IPL, it's straightforward to accommodate a new implementation technology, or other target platform, by simply extending an existing or defining a new IPL pattern. The net result of leveraging IPL is the elimination of any direct coupling of enterprise models to implementation technology.

FireStar Software's ObjectSpark

FireStar Software (formerly ONTOS) supports the OMG's MDA initiative with its ObjectSpark product. ObjectSpark creates, executes, and manages a data/message service layer for accessing internal and external databases and applications with XML-enabled APIs. A key element of FireStar's approach to data services is to provide a service interface that is defined and controlled by the applications' own object model.

ObjectSpark consists of the ObjectSpark Designer and the ObjectSpark Universal Adapter. The ObjectSpark Designer is a visual programming environment in which developers create or modify a map that defines the data/message service layer by importing their application interface definition (UML or XMI files), importing the schemas of the target databases (data definition language formats), importing the message formats (XML schemas and document type definitions), and setting the appropriate behavior based on business logic requirements. These maps are stored in XML to support iterative engineering. The ObjectSpark Universal Adapter transforms these maps at runtime into a data service that is a specific platform implementation for COM+ or J2EE. The ObjectSpark-created data service provides high-performance optimizations (e.g., marshalling, caching, and attribute shadowing for composition, aggregation, and associations) that are optimized and leverage the specific platform.

Within the context of the OMG's MDA, ObjectSpark uses OMG's standards and initiatives of UML, XMI, and storing of models. Specifically, ObjectSpark uses UML or XMI to specify the data services interface. ObjectSpark data services fully implement UML and XMI Data Object Class Diagram Specifications. ObjectSpark supports round-trip (iterative) engineering. Finally, ObjectSpark supports the distinction between an implementation-independent PIM and platform-specific PSM. ObjectSpark has the ability to accept a PIM based on OMG standards and let the developer create the specific PSM implementation.

Headway Software's Headway reView

Headway Software sells Headway reView, a software visualization tool that allows developers to reverse engineer software code. In effect, reView lets a developer convert Java and C++ code into UML diagrams. Headway reView only uses a subset of UML, the minimal subset required for programmers to develop a strong understanding of the overall structure and design of the code. The tool does more than simply display the UML; it automatically and intelligently lays out the model in such a way that the position of the icons on the screen in itself conveys meaning. Headway reView is capable of visually illustrating the dependencies in the code at all levels and, interestingly, between all levels: package, class, method, and data member. Unlike traditional pan and zoom user interfaces, the model itself can then be "surfed" to identify the areas of interest, in a manner not dissimilar to the way a user surfs the Web.

The current version of Headway reView stores its data in XML format and will upgrade this to support XMI in an upcoming release. This will be Headway's first step in integrating with the current modeling tools, such as Rational Rose. Headway sees a good fit between reView and Rose, with reView being capable of broadening the reach and appeal of UML by making it easier to get the model into the hands of the developer. It doesn't yet support PIMs, as such, but is planning on moving in that direction.

Interactive Objects Software's ArcStyler

Interactive Objects (iO) Software was already moving in the same direction as the OMG when MDA was announced. ArcStyler is, in essence, an implementation of MDA concepts in the context of a holistic IT-architectural style. ArcStyler enables a chief architect or lead designer to achieve end-to-end model-driven development based on a tightly integrated suite of tools, known as an Architectural IDE, which embeds and adds architectural intelligence to existing tools such as Rational Rose and JBuilder. ArcStyler generates all four tiers of J2EE/EJB infrastructure, including deployment and test environments, for the leading application servers (e.g., IBM, BEA, Borland) using an extensible generator engine. Aside from out-of-the-box support for MDA-based architectural style, ArcStyler provides a visual development environment enabling a chief architect to extend the style to meet the special needs of a particular project or corporate environment. The capability of tailoring the architectural style while still leveraging a clean MDA approach is known as metaprogramming.

Richard Hubert, CEO of iO Software, describes an MDA tool or tool suite as a product that "enables, supports, automates, and significantly improves the software development process by applying the principles of MDA." He goes on to say that such a tool suite must, at a minimum, "model the business domain and support a trackable mapping to technical UML models and then to an effective IT infrastructure and generate code from the models and an infrastructure tailored for a specific platform. It should also enable the creation, propagation, and maintenance of an IT architectural style. The central issue is to use visibly related, well-formed models to express different aspects of a software system, and to support automatic or semi-automatic transformations between these models to keep them and their generated artifacts in sync."

Borland has announced that it will incorporate ArcStyler in its Borland Enterprise Studio 2. In effect, Borland's Enterprise Studio 2 will become an MDA product as a result.

Kabira Technology's ObjectSwitch and Kabira Business Accelerator

Kabira's MDA product is called ObjectSwitch. It is a UML-driven application server that provides very high-speed transactionality, distribution, persistence, event queuing, and fault tolerance. ObjectSwitch comes with a Rose add-in for creating PIMs and a design tool that imports PIMs from Rose. The design tool supports flexible implementation of objects, for example in databases, CORBA, EJB, SNMP, and so on, and generates code for the runtime server. Users of ObjectSwitch can change implementation of PIMs without modifying their Rose models and can produce complete, very high-speed, production-grade applications without round-trip engineering. Another of Kabira's MDA products is called the Kabira Business Accelerator. It is built on ObjectSwitch. In effect, the Business Accelerator is a suite of frameworks and precomplied modules running on the Kabira ObjectSwitch server platform for business-process-driven applications. Kabira Business Accelerator includes support for UML, XML, DOM, PHP-3 and 4, and Java.

Kabira is currently focused primarily on the telecommunications and e-commerce markets. It provides infrastructure software for the delivery of Next-Generation Services made possible by the convergence of the Internet, traditional enterprise, and telecommunications networks. Kabira's ObjectSwitch product suite is designed for the fast deployment of Next-Generation Services capable of delivering high-performance, reliable, scalable solutions. The delivery of Next-Generation Services requires customers to create something new, while leveraging what they already have. In order to do this, customers must be able to integrate existing software applications, automate business processes, integrate network equipment using high-speed network protocols, and mediate among existing systems. ObjectSwitch provides all of these capabilities within a single, unified architecture, and it streamlines the development process through the use of UML modeling tools.

Kennedy Carter's iUML and I-CCG

iUML comprises a modeler and simulator. The iUML modeler permits the intelligent capture of platform-independent, executable UML models with the UML diagrams being supported by the action semantics-compliant Action Specification Language (ASL). The iUML simulator provides an execution environment in which models can be executed, debugged, viewed, and tested. The iUML family supports predefined mappings to platform-specific implementations, which preserve the semantics of the application.

The I-CCG product supports the definition of user configurable mappings from PIMs to PSMs. The mappings are specified using executable UML models. This flexible approach allows users to define mappings to platform-specific implementations for a range of application categories. Examples include small, embedded real-time systems, using MISRA C and OSEK, to large distributed systems exploiting technologies, such as CORBA, EJBs, and XML.

Colin Carter of Kennedy Carter explains his company's commitment to MDA in this way: "UML models, together with an action semantics-compliant language, allow the full potential of MDA to be exploited. Conventional development processes have made it difficult to retain the value of platform-independent models. Automated support for the mappings to a platform-specific implementation is now available. So businesses can use models as the primary, maintained form of specification of their application and business logic. As a result, an organization's key intellectual property is retained when the implementation technologies change. In fact, changes in technology can be exploited without the worry and cost of respecifying the application."

Metanology's Meta Development Environment

Since 1998, Metanology Corporation's Meta Development Environment (MDE) has enabled application development consistent with OMG's MDA. Using UML, an application is expressed in a PIM and transformed into an implementation of the application on a specific platform. The model transformation is accomplished by using MetaPrograms, MDE's unique code generation technology. MetaPrograms are created in Java, are easy to write, do not depend on existing code repositories, and do not require that developers learn proprietary technology to take advantage of the MDE's capabilities. The MDE produces pure source code, requiring no additional runtime components or specific application server.

MetaPrograms are developed independent of the PIM using MDE's MetaProgramming feature. A single set of MetaPrograms controls the transformation of any PIM onto a specific technology platform, with MetaProgram-controlled architecture. PIMs are MetaProgram-independent.

The combination of application-specific PIMs and model-independent, platform-specific MetaPrograms dramatically decreases the cost of developing, supporting, and migrating multiple applications on multiple platforms. Metanology currently has MetaPrograms that transform a PIM into an enterprise-scalable implementation of the application on Sun's J2EE and Microsoft's DNA. MetaPrograms for Microsoft's .NET will be available in the first quarter of 2002. These MetaPrograms can be licensed and used as is or can form the basis for an organization to quickly create a custom architecture unique to its existing applications, architectures, and legacy systems.

ObjeXion Software: Netsilon

Netsilon is a comprehensive, visual model-driven environment for e-business application automation. Netsilon is promoting agile modeling and Web product line software engineering using MDA concepts and vision. Netsilon combines a UML modeler and an application generator. The various aspects of e-business software systems are captured in business models (business objects and rules), navigation models (the application cinematic), and presentation models (look-and-feel elements, designed with third-party Web authoring tools), while a powerful action language (with OCL and Java-like semantics) allows it to combine all these modeling elements together.

Netsilon ensures consistency and synchronization among these three types of models (which may evolve independently from each other) during the complete lifecycle, and it automates code generation and deployment. Models are totally platform-independent. At generation time, parameters specify the desired deployment configuration. Supported deployment environments currently include the J2EE and PHP application servers and Oracle and MySQL databases.

Project Technology's BridgePoint and DesignPoint

The heart of Project Technology's toolset is a metamodel of executable semantics for MDA's PIMs, comprising UML diagrams and an action semantics-compliant action language. The Model Builder captures PIMs that conform to MDA in a repository based on the metamodel. The Model Verifier is a PIM interpreter. Developers can establish sets of initial conditions, step through the execution state by state or action by action, and modify the model, if needed, as the model executes.

DesignPoint is a suite of products, each of which is a model compiler. Each model compiler embodies a PSM with a set of translation rules that compiles a PIM into a specific EDM. Some model compilers generate multitasking PSMs with persistence with an EDM target of C++; others generate a single task sitting directly on the silicon without the necessity for an embedded operating system into a C EDM. Every model compiler is completely open, allowing developers the freedom to modify the PSM directly, or to build their own PSM.

Each DesignPoint model compiler relies on BridgePoint's Generator, which accesses the PIM stored in a repository and applies the model compiler's rules to generate applications that can be for any programming language, conforming to any API and any coding standards.

Secant Technologies' ModelMethods

Secant Technologies is an established application server vendor, whose server has traditionally been called the Secant Extreme Server. In conjunction with the OMG's announcement of MDA, Secant announced ModelMethods, a suite that combines the ModelMethods Object Integrator; data integration software; and ModelMethods Enterprise Server, a server designed to support a model-driven approach to software development and maintenance. As Secant explains it: "ModelMethods signifies a new name and packaging for Secant Extreme software and demonstrates our existing and future support for the OMG's Model Driven Architecture."

Secant Technologies has also announced its support for MDA.

Keith Rupnik of Secant's product management group describes an MDA product and Secant's tool as follows: "An MDA tool is a model-based development solution that supports OMG's standard MDA specification and architecture. In a nutshell, such a tool allows a complete system to be specified in UML without regard for the target operating system, programming language, or middleware platform, and then, on demand, generates the software once these variables are specified. Secant provides such a tool that currently generates 80% of CORBA and J2EE applications." Secant currently integrates with Rational Rose and TogetherSoft's Together Control Center to allow the developer to model the application and then, through Secant, generate the code that implements the model. Secant is already working to extend ModelMethods to provide support for XML, Web services, and the ability to build, power, and evolve J2EE, CORBA, and C#/.NET applications.

Softeam's Objecteering/UML

Softeam provides MDA support through the Objecteering/UML CASE tool product line. Built around a repository fed by processors, all of which share the same information, Objecteering/UML is an all-in-one tool, combining modeling, code generation, documentation, and test generation. At the heart of Objecteering are two tools: Objecteering/UML Profile Builder, through which users can implement their own UML profiles; and Objecteering/UML Modeler, which is customized by the UML profiles deployed on it.

Since 1994, Objecteering/UML has added behavior to UML profiles, through the support of the dedicated "J" language. Applying a UML profile to a UML model is an operation carried out by the user, who applies technology-specific knowledge to a technology-independent model. This provides orthogonal separation of a model dedicated to a particular domain from platform-specific knowledge. Users can create their own UML profiles, which include UML extensions specific to a technique: dedicated model consistency checks, model transformation behavior implementing model mapping (such as PIMs to PSMs mapping), model filtering, presentation rules, specific GUIs, and diagram-automated construction or filtering.

Objecteering/UML Profile Builder made it easy to implement the "UML to EJB" standard UML profile, published by the Java Community Process. Objecteering/UML also features test modeling and generation capabilities, thanks to UML profiles. Tests are modeled independently of the UML application model, through the specialization of sequence diagrams, and wizards are used to create a test model independent of the application model. The test model generates 100% of test code, including stubs, using the JUnit library for test execution. Test documentation, including test definition and test execution reports, is also produced.

TechOne, Inc.'s ACE

Accelerated Construction Engine (ACE) is an MDA-based development accelerator to automate the task of architecting, designing, and coding a multitier, OO application, based on proven design and implementation patterns.

ACE works with visual UML modeling tools supporting XMI (e.g., Rational Rose, Select, and StructureBuilder), to access and transform PIMs to PSMs. The ACE design properties are technology-independent and specify, for example, a business object's lifecycle, persistence, transactional, UI/presentation, and EAI aspects. At present, ACE has a collection of more than 200 properties. To further simplify a designer's task, sensible default design property values are supplied by the ACE workbench.

Once design is completed, from a PSM, ACE can automatically generate and assemble a significant portion of the *n*-tier application code using technology-specific implementation templates for EJB, JSP, XML, SOAP, CORBA, and SNMP.

ADDITIONAL RESOURCES

To get additional information on MDA, visit the OMG Web site: www.omg.org/mda. This site keeps changing as new announcements are posted. There are several papers posted on the OMG site that are worth reading. Among the most important are *The OMG's Model Driven Architecture: A Technical Perspective* (OMG document ab/2001-02-04) and Richard Soley's *Model Driven Architecture*, which provides a good explanation for the OMG's move to a new architecture. David Flater of the National Institute of Standards and Technology has written a paper entitled *The Impact of Model-Driven Architecture*, which is also good. In addition, the slide sets by Tom Digre and Cutter Consortium Senior Consultant David Frankel are worth studying. All are available on the OMG site.

Richard Hubert of iO Software has just completed the first book on MDA to hit the market, *Convergent Architecture: Building Model-Driven J2EE Systems with UML* (John Wiley & Sons, December 2001); for more information, check www.convergentarchitecture.com. Kennedy Carter has also announced a book, titled *Executable UML for Model Driven Architectures*, which will be published in early 2002. For more information, check www.dk.com. You can expect to see several additional books on MDA later in 2002.

WHAT HAPPENS NEXT

As suggested, it's encouraging that so many companies have announced support for the MDA concept so quickly. It suggests that MDA is really an idea whose time has come and that many different companies were exploring closely related ideas that they were happy to position under the OMG's MDA banner. I predict we'll hear a lot more about MDA in 2002.

Throughout the 1990s, those who focused on model-driven development didn't have any general concept to rally around. The crash of the CASE vendors in 1990 led most software engineering vendors to take a low profile. At the same time, object methodologies and OO modeling tools flourished, and we got better at generating code from models and reengineering. UML became an international standard. In the meantime, the problems that led to CASE in the first place didn't go away. One has only to read any of the popular computer journals to realize that we still aren't very good at developing large enterprise applications. Every month there are reports of the failure of major enterprise development projects. Developers need help. And EAI has proven to be especially difficult. It is time for a major push to create better tools for large-scale systems development. The OMG has raised the MDA banner and stands ready to generate the standards needed to assure its success. Vendors who have been working to automate enterprise development are rallying to MDA. The question now is whether major companies will also rally. I predict that they will. It will take some time, but the need is great. Soley describes MDA as the basis for a "20-year architecture" and as the solution to the EAI problem. I suspect that's going to appeal to a lot of CIOs who are trying to cut costs while at the same time generating and modifying e-business applications as fast as they can.

This article benefited from feedback from the following individuals: Conrad Bock, Michel Brassard, Colin Carter, Eric Castain, David Frankel, Paul Hickey, Richard Hubert, Phillip Lindsay, Ken Lord, Tony Mallia, Stephen Mellor, Pete Rivett, and Keith Rupnik.