What's Holding Back OMG's Model Driven Architecture?

Immaturity, UML complexity and lack of personnel to create implementations are drawbacks

BY CAROL WEISZMANN AND SUSAN MESSENHEIMER

These days, information technology implementations need to cost less to create, deploy and manage. And they need to work better with what organizations already have in place.

Ideally, the focus of an IT implementation should be on business function rather than on the esoterica of the technology itself. Most often this is best achieved by abstracting—that is, modeling—business functionality.

"All forms of engineering rely on models as essential to understand complex real-world systems," noted Alan W. Brown, IBM distinguished engineer. Yet most software developers don't use separately defined models, opting instead for a code-only approach, making it tough to distinguish between important elements and bells and whistles—and much more costly to manage change as programs evolve and become more complex.

Given that modeling can help address several of developers' "top of mind" issues and concerns—notably application life-cycle management, agile development, business integration and service-oriented architectures—it's curious that so few enterprise development teams have embraced the best-known approach to model-driven software development: Object Management Group's Model Driven Architecture (MDA), a standards-based effort to build programs from models using model transformations.

MDA has been around since late 2001 and is supported by development tools from more than 50 vendors as well as open-source groups such as OMEX AG (openmdx.org).

Achieving agility. The idea is to move, via transformations, from business models (called computation-independent models, or CIMs) to platform-independent models (PIMs) to platform-specific models (PSMs) to code.

"The requirements must be traceable from the CIM to the PSM and vice versa," said Francisco Javier Lomas Beltran, a Microsoft .NET certified application developer working for Ecuador-based software development company Kruger, which is adopting MDA. "This is one of the keys to knowing that the abstraction of the problem is well done. Another key is the record of the transformations, including which element was transformed to which element in each model and what mappings have been used to accomplish that."

Higher quality, lower costs. Harold Teets, senior vice president and CTO/CIO at Xspedius Communications, said that his firm's experience with MDA (using tools from Kabira Technologies and IBM to build customer resource management capability) "has fostered a more global look at data replication and helped provide more focus on systems integration."

This, he pointed out, has produced rapid improvements in internal data quality, improvements in productivity, and reductions in data duplication, rework and application support needs. "We learned," Teets said, "that leveraging existing data stores and building an integration services layer that insulated implementations via adapters resulted in reduced software development and integration costs."

Fast too. "Some of the highest-speed systems in the world are MDA systems," asserted Dirk Epperson, vice president of product strategy at Kabira, a real-time software development firm committed to MDA. "We have customers with applications that are 100 percent model-driven—that is, there is no C++ or Java in the application models—and the systems handle well over 10,000 transactions per second in production and have been tested at many times that rate."

IMMATURITY, FUD AND THE FUTURE

So why hasn't MDA been more widely embraced? "The major weakness in MDA is that it is immature," said Epperson. "It's unlikely that models based on one vendor's tools could be used on another vendor's MDA with no changes." What's more, there is as yet no single compliance test that determines MDA compatibility.

Tom Gullion, Borland Software's product manager for Together products, pointed to another indication of MDA as a work-in-progress: "There's a general perception that UML was already bloated at version 1.5," he noted. "UML 2.0 has added a significant amount of complexity to the specification."

Although UML is standardized for defining program structure, observed Epperson, program logic has come to UML only recently in the form of "Action Semantics," which serves as a high level of abstraction for this logic. "But, there is not, to date, a single syntax for the Action Semantics," Epperson said. "Instead, vendors comply with different syntaxes for the same semantics. A single syntax will make more IT organizations comfortable with an MDA approach."

Paresh Yadav, chief technology officer at Saint Technologies, a Canadian development company following MDA but yet to commit to it, cited several additional problems: "There's a lack of qualified professionals to implement MDA, a steep learning curve, and the upfront cost prohibits wide-scale adoption—even though MDA can provide huge cost savings over the life cycle of an application due to savings in supporting

MDA: SEPARATING THE PARTS TO STREAMLINE THE WHOLE

Addressing the complete software development life cycle—designing, deploying, integrating and managing applications as well as data—MDA incorporates key OMG-specified services (including directory services, event handling, persistence, transactions and security) and leverages several OMG standards:

UML: The widely embraced Unified Modeling Language.

MOF: The Meta-Object Facility, which unifies UML, UML profiles and MDA platform-specific models.

CORBA: The Common Object Request Broker Architecture, OMG's middleware standard.

CWM: The Common Warehouse Meta-model, standardizing metadata interchange in data-warehouse and business-analysis environments.

XMI: The XML Metadata Interchange, a standard for using XML to store and exchange models.

TWO SCHOOLS OF MDA PHILOSOPHY AND PRACTICE

The Executable, or Translationist, School:

MDA is basically programming with pictures.

Mike Burba, marketing director of Compuware's application development solutions, explained: "All application information should be described in models, and those models should be 'compiled' into the target platforms (J2EE, .NET, CORBA, etc.)."

The Pragmatic, or Elaborationist, School:

Achieving 100 percent code generation from models is impractical.

Mike Sawicki, product manager of Compuware's OptimalJ, spelled it out: "Models should be a high-level business abstraction of the application, pure in the sense that they contain no meta-information of the technology platform on which the application is to be deployed." But since it's impossible to achieve that level of abstraction with the exclusive use of models, pragmatic implementations of MDA (including Compuware's OptimalJ) "allow [platform-independent models] to be translated into [platform-specific models], tweaked and enriched and then translated into code that can also be elaborated upon."

Modeling, MDA and Microsoft

BY CAROL WEISZMANN AND SUSAN MESSENHEIMER

and enhancing the applications."

Thus vendor-neutral MDA makes it easier for IT architects to efficiently design systems that can integrate legacy environments with the constantly changing parade of new standards, such as XML, .NET, EJB/J2EE and CORBA.

"The application of open industry standards as the basis for automation of transformation between models is an important additional value to businesses and developers," said IBM's Brown. "It means that there can be rigor to the process being used, the transforms can be developed and managed as reusable assets, and the efficiency and quality of the solutions being developed can be increased."

THEN THERE'S MICROSOFT

Not surprisingly, Microsoft does not explicitly support MDA. But, according to Keith Short, architect for the company's Visual Studio Enterprise tools, Microsoft developers are embracing their own version of modeling.

The trouble with OMG's MDA, contends Microsoft in a document called "Modeling Strategy and FAQ," first released to SD Times, is that it addresses "only a small subset of what we believe are the real issues that must be addressed to enable effective model-driven development." For instance, no single set of models can describe all possible systems effectively. "A single PIM and a single PSM per target platform, all developed using a general-purpose modeling language, as prescribed by MDA, are insufficient to support the significantly greater levels of automation promised by model-driven development," the document reads.

In a document called "Modeling Strategy and FAQ," dated May 20 and obtained by SD Times, Microsoft lays out its view of model-driven development and the ways in which Visual Studio 2005 Team System, now in beta, will fulfill its vision. Among the key elements of the Microsoft perspective:

"A model should be a first-class artifact in a project—not just a piece of documentation waiting to become outdated. Models have a precise syntax, are often best edited and viewed using a graphical tool, and embody semantics that determine how domain-specific concepts in models map to other implementation artifacts, such as code, project structures and configuration files. In this way, a model is a lot like a source code file, and the mechanisms that synchronize it with other implementation artifacts are a lot like compilers."

While models can be "compiled," they are more often implemented by a combination of generated and hand-edited artifacts. "In such cases, it is critically important to carefully manage the way the generated and hand-edited artifacts fit together." Thus Microsoft supports "techniques to ensure that generated and hand-edited artifacts are kept separate, and that code added by a developer is never disturbed when boilerplate code required by a tool is generated. These techniques include the use of class delegation and inheritance, and particularly the use of 'partial classes.'"

Microsoft calls these sorts of modeling languages Domain Specific Languages (DSLs)—a DSL is "a small, highly focused language for solving some clearly identifiable problem." Examples include SQL for data manipulation and XML for document structure definition. "Good ways to find candidate DSLs are to identify the patterns used by developers, and encapsulate them into a modeling language, or to surface the concepts in a software framework as abstractions in a modeling language that can then generate small amounts of code that extend the framework. These techniques allow us to control the amount and complexity of generated code."

Eschewing a one-size-fits-all approach to modeling, Microsoft instead prefers to gather custom collections of DSLs into "Software Factories," which encompass and extend MDA "where MDA is defined in a broader sense than the official definition based on PIMs and PSMs." ▮

Microsoft also sees UML as overly complex and failing to address in a natural way "critical issues for modern application development, such as database design, testing, deployment, service orientation, component-based development, and user interface construction." Still, it recommends using UML—which will ship with Visual Studio 2005—for sketching, whiteboarding, napkins, documentation and conceptual drawings not directly related to code. It prefers precisely defined DSLs and DSL-based tools for:

- Precise abstractions from which code is generated.
- Precise abstractions that map to variation points in frameworks and components.
- Precise mappings between DSLs.
- Conceptual drawings that have precisely specifiable mappings to other DSLs or to code artifacts.

MDA VERSUS DSL

Andrew Watson, vice president and technical director at OMG, pointed out what he regards as problems with Microsoft's approach.

Microsoft is not using relevant standards, though Watson said its position is evolving, since it now may "support interchange with popular UML and MOF tools" by using OMG's standard XML Metadata Interchange (XMI) format.

Microsoft doesn't support platform independence. Watson noted that all real-world customers actually use multiple platforms. "Because supporting multiple platforms is an overriding concern for most IT customers," he said, "MDA places a strong emphasis on it."

Watson continued, "Microsoft doesn't provide much support for long-term maintenance of legacy applications. In contrast, MDA emphasizes long-term support and maintenance"—an important factor for any business that must sustain the value and use of its legacy apps.

"Despite UML's popularity with users, Microsoft is avoiding using it where it would be applicable," said Watson. "Instead, Microsoft advocates use of Domain-Specific Languages, which in

▸ continued on page 30

Why Hasn't MDA Seen Wide-Scale Adoption?

◂ continued from page 29

many cases are very reminiscent of UML diagrams."

A substantial collection of MDA success stories (www.omg.org/mda/products_success.htm) in a wide range of sectors—including banking, aeronautics, electrical engineering, government, railroads, software development, logistics and online education (some with details about cost savings and productivity improvements)—challenge the contention that MDA is unable to handle modeling requirements across distinctly different business sectors.

As Watson sees it, Microsoft is embracing MDA without actually saying so. Their point of agreement seems to be the Meta-Object Facility (MOF) standard, which, Watson said, "is the foundation of MDA."

Opposing the brave new world that MDA and its competitors represent is no small amount of fear, uncertainty and doubt. "Paradigm shifts require cultural change, and change is difficult for any organization," said Mike Burba, marketing director of Compuware's application development solutions.

"One change that development teams must undergo is organizational in nature: Some members of the team are going to be asked to play new roles," said Mike Sawicki, product manager of Compuware's OptimalJ. "Because less time is spent on the implementation phase of development projects, more emphasis, proportionally, is placed on application design. So some developers need to adopt new technical skills, such as UML modeling, and others must develop new 'soft' skills, like being able to effectively communicate with the business."

But Saint Technologies' Yadav is optimistic. "Recent widespread adoption of tools that help 'visualize' code is a good indication that eventually MDA will gain a larger following." ▮

MOF AT A GLANCE

The collection of standard interfaces called the Meta-Object Facility constitutes a common, abstract language that can be used to define, construct and manipulate interoperable metamodels, thus enabling dissimilar metamodels to reside in the same metadata repository (which stores data models and domain object models) as long as they all conform to MOF.

At the heart of OMG's Model Driven Architecture, the MOF specification defines a small set of constructs for object-oriented information modeling that can be:

- Extended by inheritance and composition to define a richer information model supporting additional constructs.
- Or used as a meta-meta-model (or an ontology)—that is, as a model for defining information models, even those differing (in ways both small and large) from the MOF itself. Among the meta-models the MOF has been used to define are the Unified Modeling Language (UML) and the Common Warehouse Meta-model (CWM).