

## MDA Journal

February 2004



**David S. Frankel**  
David Frankel Consulting

[df@DavidFrankelConsulting.com](mailto:df@DavidFrankelConsulting.com)

MDA, UML, and CORBA are Registered Trademarks of the Object Management Group. The logo at the top of the second page is a Trademark of the OMG.

[www.bptrends.com](http://www.bptrends.com)

Last month's article by Steve Cook of Microsoft has, not surprisingly, evoked some strong reactions. I said in my introduction to the article that the entry of Microsoft into the model-based systems field is overall a positive development, although I also said that MDA Journal would publish responses to the rejection by Steve and Microsoft of MOF and XMI. This month's article, a response to Steve by Mike Guttman, is less positive, but certainly reflects the attitude of some in the MDA camp. His opinions are his own and thus do not constitute a statement by the OMG. I should also add that I am hopeful that Steve will be available to comment on the responses.

Before I turn the podium over to Mike, I'd like to comment on a couple of specific points in Steve's article. Steve says that MOF is about how models are stored and interchanged. He goes on to say that this territory that MOF covers is a "rather minor aspect of the whole job..."

Firstly, I prefer to characterize what MOF does a bit differently. MOF provides the means to define the abstract syntax of a language, and the MOF technology mappings make it possible to use the abstract syntax to generate an XML syntax for the language, a Human Usable Textual Notation (HUTN) for the language, and Java or CORBA APIs for representing models expressed via the language. The mappings also make it possible to generate code that implements the syntaxes and APIs. Nevertheless, despite my different way of expressing what MOF does, Steve is on reasonably solid ground when he characterizes these capabilities as being about how to store and interchange models.

However, before dismissing the significance of the fact that the modeling industry has started to coalesce around a standard for defining abstract syntax, I would ask Steve to consider some additional factors. He is absolutely correct that we need to go beyond what MOF does, i.e. we need to have standard ways to formally define graphical syntaxes and means to define or to plug in editors and debuggers for languages, so that generic modeling tools can "become" the modeling environment for any formally defined language. However, since all of these extra capabilities must build on top of abstract syntax definitions, an agreement on how to define abstract syntax is an important foundation. By declining to participate in the accord on abstract syntax definition, Microsoft is ignoring some solid progress already made and is starting over from scratch.

More importantly, an increasing number of enterprise architects are becoming alarmed at the balkanization of metadata—i.e., business process models, workflow models, data models, component descriptors, product configuration and tuning parameters, instrumentation, and so on—into disconnected islands. They see the resulting inconsistency in storing and interchanging the different kinds of metadata, as a serious problem.<sup>1</sup> In a lot of these cases there is no option to design a new language; that is, the language for some aspect of the system—workflow or runtime management or whatever—already exists implicitly or explicitly in a tool, and MOF models of the language (i.e. of the metadata) are done retrospectively. The tool vendors are not going to change their surface syntax or the basic environment of their tools, but we would like to motivate these vendors to render their metadata in a MOF-compliant fashion. That is why I'm disappointed that Microsoft won't build on this aspect of agreement rather than discard it, especially since we're now getting significant people in the Semantic Web, business rules, business modeling, and runtime management





MDA Journal

February 2004

communities to define MOF models for their respective areas. This new traction has the potential to drive the various tool vendors to conform with MOF so that we can begin to de-balkanize the enterprise. As I said, I'm happy that Microsoft intends to push beyond abstract syntax and abstract syntax technology mappings. I just wish they would build upon the momentum we have established for having a recognized way to define abstract syntax, rather than having a second way of doing it. We can manage the difference through interworking standards, but it could be better.

Regarding XMI, Steve points out, correctly, that the interoperability story for XMI is far from smooth. He cites a tendency toward combinatorial explosion given that there are different versions of MOF and XMI. This is a valid point. However, a great deal of this is due to immaturity, much as the early attempts to get CORBA language mappings working were bogged down by instability in the CORBA IDL and language mapping standards. Once IDL and the mappings matured and settled down, they proliferated. In the enterprise computing world, much of that proliferation has been invisible under the covers of J2EE plumbing, but the CORBA-C++ language mappings are being used quite heavily in the realtime/embedded world.

Moreover, once the MOF 2.0 transformation standard (known as Query, View Transformation or QVT) is promulgated, it will be possible to "purpose-build" an XML schema for a language and use QVT to formally encode the transformation from the abstract syntax to the XML schema, as an alternate way of handling XML serialization within the MDA standards. We sure could use the help and influence of the talented people in Steve's group at Microsoft to get the MDA standards finished and right. Steve's article suggests that we won't get that help, and that would be a pity.

See you next month,

David Frankel

#### Footnotes

<sup>1</sup> For example, as Charlie Betz of Best Buy has pointed out on OMG mail lists, there is a complete disconnect between development metadata and runtime management metadata.





MDA Journal

February 2004

**Michael Guttman**  
**CEO**  
**Miriam Institute**  
**Director**  
**OMG's MDA FastStart**  
**Program**

**Arguing with UML's Success**  
**Making Up for Lost Time**  
**Is Microsoft Against**  
**Interoperability?**  
**Learning the Right Lessons**  
**from History**

## A Response to Steve Cook

*Michael Guttman is CEO of the Miriam Institute and Director of the OMG's MDA FastStart program. Mr. Guttman's views expressed here are his own and do not necessarily represent the opinions of the OMG or its membership —editor.*

According to Mr. Cook, OMG's UML standard has succeeded in popularizing a kind of modeling vernacular that is useful, but only up to a point. Ultimately, he believes that the open UML standard as currently defined by the OMG is simply not an appropriate basis for creating the kind of 'agile' systems and development tools that he and Microsoft envisage. He further claims that OMG's MDA and MOF standards, rather than truly improving or extending UML, primarily act to compound UML's fundamental flaws.

Of course, the modeling community has already put years of work into developing, extending, and improving UML, MDA, MOF and their various derivatives, using the OMG's open process. During Mr. Cook's previous professional incarnation at IBM, he was himself a major contributor to a number of those improvements, and an evangelist of UML, MDA, OMG, and open standards. The industry's massive investment in these open OMG standards doesn't prove they are the only possible approach, but, when a major goal is tool and platform interoperability, it certainly counts for something.

Moreover, UML and its derivatives have already been incorporated into a wide range of popular commercial software development products from many different vendors. Even more importantly, these products have in turn already been used to successfully implement and deploy a wide range of sophisticated, mission-critical systems across a range of industries and computing environments. The OMG's website is filled to the brim with detailed examples of both vendors and end-users who have successfully embraced its open standards.

### Arguing with UML's Success

Despite this clear evidence of the commercial success of UML and its derivatives, Mr. Cook asserts that UML itself has to date played only a minor role in software development. He characterizes this role as "UMLAsSketch," borrowing the term from Martin Fowler, a well-known UML expert. That is, Cook claims that UML as defined is useful only as a kind of drawing tool, and can never be used reliably to fully specify or generate systems.

Mr. Cook goes on to denigrate Fowler's more robust "UMLAsBlueprint" paradigm as fundamentally flawed. Why? Cook's only example is that, according to him, some of Microsoft's current programming constructs don't map as neatly as he would like to UML. "UMLAsProgrammingLanguage," Fowler's highest form, is simply dismissed by Cook as "unlikely to gain headway commercially." There's absolutely no discussion whatsoever of its technical merits, or the fact that a number of vendors are pursuing this approach aggressively.

Mr. Cook's characterization of UML only makes sense if your context is limited entirely to Microsoft's own offerings. To date, Microsoft itself offers no tools to



move from UML to code or anything else. So, if you want to use UML in a Microsoft-only world, you are indeed likely to be limited to "UMLAsSketch."

But of course there are plenty of UML and MDA products from other vendors that do generate code and other artifacts, including for Microsoft's own tools and platforms. Many of these products already realize much of "UMLAsBlueprint," and some go a good ways towards "UMLAsProgrammingLanguage." In a circular, self-fulfilling way, Cook's article simply ignores these sophisticated UML tools, whose existence – not to mention commercial success - are inconvenient to his line of reasoning.

So what's the real reason why Microsoft itself doesn't already offer such tools? It's clearly not because they aren't technically feasible. As far as I can tell, Microsoft has no such UML tools mainly because it has up until now chosen to ignore the rest of the industry's steady march towards model driven development, and missed the formative years of its evolution. That is, while the OMG and its members were busy developing UML, MOF and MDA – and creating supporting tools - Microsoft largely sat on the sidelines.

Sadly for Microsoft, it is now clear that this was a serious mistake, and the company has a lot of catching up to do. Microsoft's recent hiring of Steve Cook and a number of other ex-OMG software-modeling experts is no doubt indicative of this painful and belated realization. In this respect, I'm glad to see Microsoft finally waking up, and applaud its move toward what is by now mainstream thinking.

### **Making Up for Lost Time?**

Unfortunately, though, Microsoft has apparently chosen to make up for its lost time, not by enthusiastically joining the MDA party, but at least in part by trying to slow down everyone else. Cook effectively takes this approach by arguing that the rest of the modeling community's current approach to modeling is fundamentally flawed, and won't ever be able to deliver the goods that developers really want and need. The corollary is, of course, that Microsoft has the soon-to-be-released correct approach, yet again coming to the rescue of the software development community.

Now let's get Mr. Cook's message straight: no open standards, no open process - just new proprietary Microsoft products that are supposedly going to be much more powerful and 'agile' than anything the OMG's supporters can ever hope to produce. According to Cook, these products will look a bit like UML, but improve it with a number of necessary – and, of course, proprietary — revisions and extensions.

What's better about these as-yet-unspecified revisions and extensions? One of the improvements Mr. Cook cites is that they will match up more directly to Microsoft's existing proprietary development tools and deployment platforms. That's certainly a reasonable goal for Microsoft, but, without an open standard or an open process, what does it really mean? Cook can criticize UML, MOF,

and MDA because they are open, published standards. In the meantime, we just have to trust Microsoft that their approach will be better when it's released.

Now I'm perfectly able to believe that Mr. Cook's team can produce some interesting products, and if they use some flavor of UML and MDA, so much the better. However, I strongly suspect that most of what he suggests that Microsoft needs to 'fix' MDA really could be accomplished by building upon existing or planned OMG standards. And, even if there really are some deeper changes required, there has always been plenty of opportunity for any vendor – and certainly one the size of Microsoft - to influence the direction of any of OMG's standards. But of course I can't prove my case, because Microsoft has yet to provide us with any specs of its own, and currently declines to participate in the OMG.

If there are indeed fundamental flaws in UML and MOF, currently being used in hundreds of products and thousands of systems, why doesn't Microsoft offer to work with the rest of the development community in the OMG to correct them? After all, won't these problems affect Microsoft's own customers who use those products? This certainly wouldn't stop – or even significantly slow down - Microsoft from introducing any of its own new products. In fact, we'd all be just that more ready to embrace those new Microsoft products. Microsoft takes exactly this approach when it comes to some other standards, such as WSDL and XML.

So what's the real reason that Microsoft doesn't just bite the bullet and finally embrace UML, MDA and the OMG? I'm sure Mr. Cook and his team at Microsoft could continue to dissemble on this question from a technical perspective, citing the various flaws in the current OMG standards. However, I believe the real issue has nothing to do with technical deficiencies in UML or MOF, many of which are already well known and under discussion within the OMG anyway.

Instead, maybe Microsoft's real reason has something to do with the way that it has decided to do business in this arena. The nearest thing I can imagine is that since Microsoft has finally determined that some form of MDA is the way to go, and has started too late to utterly dominate the OMG's MDA sandbox, the company is now busy planning to set up its own MDA-like sandbox across the street. I hope this isn't really true, but Mr. Cook's article seems to drift heavily in this direction.

### Is Microsoft Against Interoperability?

But if it is true, this can only mean one thing – that Microsoft is planning to compete pretty much across the board with the OMG. What does a huge, for-profit company like Microsoft stand to gain by trying to compete with the OMG, a non-profit industry consortium dedicated to industry-wide tool and platform interoperability? How can you compete with the goal of industry-wide interoperability? It seems to me like a shortsighted policy that will ultimately hurt Microsoft's own customers.

But Mr. Cook and I do agree about one thing: this is ultimately about the developer. Thankfully, Cook doesn't argue with the OMG's assertion that developers desperately need something like MDA. However, to the extent that his current

view prevails at Microsoft, developers will once again have to contend with two fundamentally similar MDA-like approaches to the exact same set of problems, expressed in annoyingly different vernaculars. It's COM vs. CORBA again, but on a bigger scale.

I'm quite sure that Mr. Cook would argue, as some others at Microsoft have done before, that this is not necessarily a bad thing, because such 'competition' will bring choice to the market, like Coke and Pepsi. But to the typical developer, this is unfortunately not about choosing between Coke and Pepsi. Practically speaking, this is about being forced to continuously mix Coke with Pepsi, and hoping the result doesn't blow up in your face.

That's because most developers already live in a world of multiple tools, platforms, languages and generations of software – and multiple vendors. Moreover, they usually have little choice in the matter. Therefore, proprietary improvements that impact only one vendor's tools and platforms won't solve their problems. Only improvements that reach across their entire set of tools and platforms – now and into the future - will truly be useful. But this is exactly the problem that an open standard like OMG's MDA is designed to address. MDA is already about choice – the ability for users to choose any set of 'best in breed' solutions, while knowing in advance that they will interoperate.

So, if Microsoft insists on trying to market some sort of proprietary MDA, developers will simply find it progressively harder to integrate Microsoft's future tools and platforms smoothly into the emerging standards-compliant industry mix. Some form of universal MDA will ultimately prevail anyway, since developers will need and demand it, but it will just come later rather than sooner, and will cost more rather than less to implement and support. Microsoft can't ultimately stop or control this process, and these negative consequences will prove most costly to its own customers – that is, those who choose (or are stuck with) Microsoft's tools and platforms.

### Learning the Right Lessons from History

Interestingly, Cook begins his article talking about the failure of CASE, claiming that "What went wrong with CASE was that it ... included a large and irreversible code-generation step, such that the code that was generated could not be modified by the developer without invalidating the entire approach..." This arcane technical explanation widely misses the mark.

The primary reason that CASE failed was because CASE vendors were unable to agree on a common approach and to develop supporting standards. As a result, each vendor had to solve the entire problem of CASE by itself – from the business model to the deployment platform. Not only was this proprietary approach a terrible disservice to customers, but also it ultimately sank the CASE vendors themselves.

The OMG was created – and it created UML and MDA – specifically to avoid this kind of pitfall in the future. In reality, Microsoft's best way to avoid the previous pitfalls of CASE would be to join the OMG, and become the biggest



booster for MDA. Blasphemy in some circles, perhaps, but I ask again – why not? If MDA or UML has problems, why not help the OMG fix them, rather than fight the OMG?

After all, Microsoft's problem can't really be with the OMG itself, which is, by its very nature, an open standards group – open to change, open to scrutiny, and open to everyone, including Microsoft. And it certainly can't be really with UML or MDA per se, which comprise an open architecture more than capable of including Microsoft's tools and platforms. The many MDA tool vendors who already support a range of Microsoft products already prove that assertion.

The bottom line is that, contrary to Mr. Cook's article, there is plenty of room in the OMG's sandbox for Microsoft to play. But apparently Cook and some of his colleagues just aren't ready for this. Is this a good business strategy for Microsoft? Maybe, or maybe not – it's ultimately not for me to say. But it's certainly no good for anyone else in the software business, and least of all for Microsoft's own customers.