# Whatever Happened to CORBA?

MDA Journal

May 2005

David S. Frankel
David Frankel Consulting

df@DavidFrankelConsulting.com



## Introduction

I just returned from the weeklong meeting of the Object Management Group (OMG) Technical Committee, which took place in Athens, Greece. The OMG holds five Technical Committee meetings per year. These meetings are not conferences; rather, they are working meetings where OMG member companies propose, review, revise, and cast up or down votes on standards, all subject to approval by the Architecture Board and Board of Directors.

The OMG has two main areas of activity: CORBA and MDA®. CORBA was the original work of the OMG, before MDA came into being. I have focused most of my attention in recent years on MDA, but watching the OMG in action this past week motivated me to turn the spotlight onto CORBA this month. Some of the things I have to say about it may surprise you.

## Distributed Services – The Original CORBA Vision

Technically, CORBA defines standard mechanisms for providing services that clients access across a network, and for publishing the interfaces to such services. It does so in a way that is independent of the programming language used to construct the service, meaning that a client service written in one programming language can access a service written in another language.

When CORBA emerged in the early and mid 1990s, its proponents intended it to be the technical underpinning for electronic commerce. Two of the fathers of CORBA, Jason Matthews and Michael Guttman, wrote a book in 1994[1] that laid out a vision of what CORBA would mean for industry. They foresaw a "mega net" that would serve as the communications backbone for an electronic economy, with CORBA as the protocol for publishing and offering services over the mega net. (At that time the internet had not quite entered the popular consciousness, even within the computer industry.)

They got it right in many respects. Their vision was startlingly similar to the buzz one hears today about internet-based, service-oriented architecture, and B2B commerce. However, due to some accidents of history, web service protocols have come to dominate B2B commerce.

[1] Michael Guttman and Jason Matthews, *The Object Technology Revolution*, John Wiley & Sons, 1994.

## Uses of CORBA Today

Nevertheless, CORBA still is an important technology, even though it does not get much play in the trade press. Use of CORBA is still increasing. It is used in air traffic control technology, airline reservation systems, financial services, and more. The *users* of these technologies do not see CORBA because, when CORBA does its job correctly, it works silently, under the covers, making widely distributed parts of systems work together as though they were co-located.

CORBA is not used for web-based B2B commerce, however, because it is not a web protocol. The web imperative has forced system administrators to open up their systems to contact with the outside world, and the job of keeping their systems secure is daunting. They are reluctant to allow use of CORBA for B2B because letting yet another protocol cross their corporate firewalls is just too scary. Thus, CORBA is used for coordination among parts of systems that lie within company firewalls or among parts that communicate with each other over special, closed networks.

It is too bad that CORBA's use is limited in this way. Many CORBA experts lament – with some foundation – that web services standard bodies are re-inventing much technology that CORBA had nailed down quite well ten years ago.

You may hear voices that say that CORBA is inherently unsuited to B2B commerce. For the technical among you, the claim is that CORBA is connection-oriented, while B2B commerce must be connection-less. The truth is that CORBA was designed to be able to function in either manner. It has mostly been used with connection-oriented protocols, but that is not due to an inherent limitation.

This argument is academic at this point, however. Web service protocols have won out for the current and next generation of B2B commerce, and there is little use fighting it. Behind the firewall, though, elements of CORBA continue to run.

## J2EE and CORBA

J2EE application servers use CORBA for behind-the-firewall communication over the network. But CORBA is buried in the plumbing, invisible to the application programmer, because the J2EE programming model hides some of the complexity of the CORBA programming model.

There is some interesting background to this. In 1996, work began on component technology for CORBA. The CORBA community was aware of the complexity of the CORBA programming model and the problem that posed for application programmers. CORBA certainly made programming-distributed applications easier than it would have been without CORBA, but it was time to take the next step in complexity hiding. Furthermore, CORBA provided little to address the laborious tasks of packaging services as independently deployable units, and for assembling small packages of functionality into larger ones.

The OMG had a false start chasing the component dream. It issued an RFP for a "business object facility" that appeared at first to be on a path to achieve the desired objectives. Unfortunately, the OMG membership bogged down in squabbling over the nature of such a facility, and eventually the RFP was withdrawn. An alternative RFP was issued, and, by 1998, the OMG released the *CORBA Component Model (CCM).*

Unfortunately for the OMG, the delay had cost precious time. The Enterprise Java Beans (EJB) component model, a part of J2EE, beat CCM to market and was sufficiently entrenched at that point to render CCM irrelevant to corporate IT.

That was a shame, because CCM has some distinct advantages over EJB. One of these advantages is that CCM has a more powerful model for wiring components together. The manifest for a CCM component includes a declaration of the services that the component requires from other components, in addition to a declaration of the services that the component offers to clients. An EJB component manifest declares only the interfaces that the component offers.

For example, with CCM, an order fulfillment component could declare that it offers a *FulfillOrder* service, and declare that it accesses a *PostToLedger* service that a general ledger component offers. This capability makes it possible to build sophisticated tools that read the manifest and automate the assembly of multiple components into larger components.

Another important distinction between CCM and EJB is that CCM's CORBA basis means that is not hard wired to one programming language, whereas EJB is inextricably intertwined with Java. For enterprise IT, the restriction to Java is not a problem, and may in some sense be an advantage, given the rise of powerful, Java-based enterprise software tools that work with EJB and J2EE as a whole. However, for real-time and embedded systems the restriction to Java is a problem; and therein lays the story of how obituaries for CCM turned out to be premature.

## The Rebirth of CORBA Component Model

The software and firmware that drives real-time and embedded systems, such as instrumentation for aircraft and myriad other kinds of machinery, often has high performance requirements that make Java too slow. Java's relative slowness stems from the fact that it executes over a "virtual machine" that provides the same abstraction for all kinds of computer hardware. This is also what makes Java so portable; but where processing speed is at a premium, the extra layer at runtime introduces an unacceptable drag on performance. Most IT systems are not processing-intensive, and thus Java does not introduce a performance bottleneck. For IT systems, bottlenecks arise from applications that hog network resources and access databases inefficiently. But most real-time and embedded systems are processing-intensive.

The bottom line is that EJB is a non-starter for many real-time and embedded systems. Most new development in this sector is in C++ (or even sometimes in

plain C). Yet these systems are highly componentized, and the lack of appropriate software component technology has been a substantial restraint on productivity.

Thus, in the early part of this decade, some of the more advanced real-time and embedded systems development companies decided that CCM is well suited for real-time and embedded systems. Not only can CCM support C++, but its facilities for wiring components together make it quite powerful as a general component model.

These systems companies became active in the OMG and introduced some revisions to CCM to satisfy some of their requirements, and now CCM is a going concern. There are a number of commercial CCM implementations, with more to come, including some advanced component assembly tools.

For example, Thales, a France-based, multinational aerospace company with 66,000 employees is building a robust CCM implementation as the component runtime technology for their next generation systems. Thales's central architectural organization in Paris is rolling out the CCM runtime in conjunction with an ambitious MDA undertaking, using MDA modeling tools for designing components, and producing model compilers that generate and assemble CCM components from the models.

Energetic work on the part of the real-time and embedded systems community continues in the OMG to produce ancillary CCM standards and to maintain the core CCM standard.

### Real-time/Embedded Systems, SOA, and BPM

As this discussion moved to the subject of real-time and embedded systems, aficionados of service-oriented architecture (SOA) and business process management (BPM) may have found their attention wandering, assuming that runtime and embedded systems are not relevant to SOA and BPM. The focus of the BPM community has been on using BPM tooling to drive corporate IT systems, not to drive aircraft and ships.

However, I believe it would be a mistake to dismiss runtime and embedded systems from the SOA and BPM discussion. When the Coast Guard has to coordinate ships, helicopters, and fixed-wing aircraft for search and rescue missions, it has a business process management problem. From a certain point of view, search and rescue is just another business, like retail banking, travel reservations, and so on, requiring the orchestration of a number of services.

Recently, I was called in as part of a panel of experts to assess the model-driven architecture of a system somewhat similar to the Coast Guard example. The colonel in charge of the highly complex project had a remarkably well-developed vision of autonomous service components working together. He used his own terminology to articulate his ideas, but the implicit ideas were similar to the ideas of components, service-oriented architecture, value chains, and business process management. I told him he was essentially talking about

value chains, and I was surprised that he had *never heard* of the term "value chain," despite the fact that he had described the concept in so many words. That he had not heard of the term is not a reflection on his intelligence and capabilities – he has these in large proportions. Rather, I think it is an indication that there is an untapped market there that has not drawn the attention of the bulk of the SOA and BPM thinkers yet. I can imagine real-time and embedded systems that exploit a combination of CCM and conventional SOA and BPM technology to good effect.

## MDA and CORBA

MDA and CORBA are not simply two unrelated sides of the OMG. There are some important relationships between the two.

### *Middleware as a Key Enabler for MDA*

Just as a robust runtime framework for applications makes the job of *hand coding* applications easier, such frameworks make the job simpler for a model compiler that *generates* such applications. That is why advanced middleware, such as J2EE and the CORBA Component Model, is a key enabler for MDA. The Thales MDA strategy takes this into account, understanding that a strong middleware implementation is important to its success.

### *Metadata Integration*

I have written extensively in the MDA Journal about the fact that MOF is core to MDA's goal of breaking down the disparate meta-data silos in our information systems. The traditional CORBA Interface Repository, which manages CORBA meta-data, has been one of these silos. However, CCM includes MOF metamodels for the CORBA metadata and for the extended meta-data that CCM uses. This paves the way for integrating CCM meta-data in MOF-based meta-data frameworks.

Also, the original MOF 1.x specification had a MOF-to-CORBA technology mapping, whose goal was to make it possible to use CORBA to manage meta-data in a MOF-compliant fashion. However, the mapping had significant scalability limitations. The real-time and embedded system vendors took charge of the MOF 2.0-to-CORBA mapping, and they did an excellent job of addressing the problem.

### *UML Profiles for CORBA*

Several years ago, the OMG ratified the UML Profile for CORBA, which defines how to use UML to model the CORBA aspects of a system. The real-time and embedded system vendors have extended that profile for CCM, so that now the OMG has the UML Profile for CCM.

## Summary

CORBA has survived the lack of attention from the trade press and the ascendancy of J2EE in the corporate IT world. It would be stretching credulity

to say that it is more than a niche player at this point, but it is a significant player in three sizeable niches:

1. As a distributed computing programming model and infrastructure for applications that consist of parts communicating over behind-the-firewall networks or other kinds of networks that are not part of the public internet.
2. As plumbing for J2EE-based application servers—plumbing that is invisible to the application programmer.
3. In applications of the CORBA Component Model in the real-time and embedded space.

Real-time and embedded systems are also an unexploited market for SOA and BPM.

MDA and CORBA are related in a general sense, in that middleware is an enabler for MDA. There are also OMG standards that directly connect the two; specifically, the UML Profile for CCM (which includes the UML profile for the base CORBA) and the MOF 2.0-to-CORBA mapping.