MICROSOFT'S NEXT FRONTIER

Jean-Jacques Dubray
Architect, Attachmate

Predicting Microsoft's next moves is always an entertaining activity. Since I was able to predict in January 2003 that Orchestration would be implemented at the operating system level, that the BizTalk Server had nothing to do in the Jupiter project, and that WSE concepts prefigured the future of distributed computing, I will take the risk and predict Microsoft's future in the BPM space and beyond.

Service Oriented Computing (SOC) is a new paradigm for building systems or components so that they become independent of their utilization context--much the same way that human services can be used and combined by service consumers with little knowledge of the context in which they are to be used. As components are often "stateless," services are often "context independent." If loose coupling is enabled by technologies like XML, message oriented interactions, or discovery mechanisms, then, for me, the notion of context independence is the very definition and necessary condition for loose coupling, separating services from components.

Interestingly enough, Business Process Management (BPM) is about modeling and controlling the context in which enterprise resources are being used. Enterprises are constantly monitoring this context and adapting it to exceptions and possible improvements. As such, SOC represents an ideal substrate for BPM.

I wrote my first paper about BPM in the fall of 1997 and worked on my first process engine in the winter of 1999. I have, of course, been following Microsoft's BPM technologies ever since the BizTalk server was announced in the spring of 1999. At the time, Microsoft coined a term that is still sticking: Orchestration. The first version of Microsoft's Orchestration language (XLang) was pretty rough. By the fall of 1999, Matthew Fuchs, who was working for CommerceOne, introduced me to Robin Milner's Pi-Calculus theory, which was the foundation of XLang and, since then, has also become a foundation of SOC. By 2002, IBM and Microsoft had joined forces and created the BPEL4WS 1.0 specification, which was later donated to an OASIS Technical Committee with royalty-free rights to advance it to the 2.0 stage, following OASIS's open development process. We are now in 2004, and this specification is still being worked on.

The development of the specification is going very slowly, and, to date, less than a handful of companies have products that support BPEL (1.1 or 2.0 drafts). I believe that the number of customers is still small. Yet major maneuvers, which started around the end of the summer 2003, are positioning Microsoft for a complete domination of the SOC space and, with it, domination of BPM and, ultimately, IT, Microsoft's next frontier. Just as BizTalk's architecture was pure innovation then, Microsoft is re-inventing itself now as we collectively start to understand how SOC will impact software engineering.

It all started with a very simple but subtle change: Before the summer of 2003, Microsoft had touted that the next big thing would be "Web Services." Then, quietly, the marketing message shifted to talk about "message exchanges," rather than Web Services (which are a particular kind of message exchange), as the foundation of "connected systems." And leveraging ubiquitous connectivity is precisely the driving factor behind SOC; without connectivity, SOC would be irrelevant. Actually, if you look back over the past 20 years, you will see that every new level of connectivity has sparked the emergence, or driven the convergence of, technologies: client/server, the web, EAI, B2B, Web Applications, BPM, Web Services, and now Service Oriented Computing have all been marked by bandwidth and network topology improvements. (Microsoft's marketing message is now managed with nanometric precision —They get it.)

About the same time, Indigo was announced in the shadow of .NET Remoting, creating confusion about the future of this just released distributed object technology. With Indigo, Microsoft moves away from the old concept of "enterprise bus," while creating the perfect foundation of SOC, a "policy driven" framework to securely and reliably exchange messages between processes. After reading the latest white papers published in March on MSDN, I can only hope that Indigo comes sooner rather than later. This is, again, a major innovation, not necessarily in concepts but rather in promoting message exchange services as a first class citizen of the Longhorn operating system.

Yet Microsoft has indicated an even bigger shift--for a company which relied on "code" as a business model: They have announced, via an article from Steve Cook, in BPTrends.com, that software engineering will include Domain Specific Languages; Microsoft is stepping on toes in the Model Driven Architecture space. This approach was recently reiterated in an excellent presentation by Keith Short on MSDN, showing how Whidbey will allow us to manage metadata and metamodels just as Visual Studio does for code and programming languages. To further its commitment to DSL, Microsoft published XAML, an XML language that represents Windows user interfaces. If the idea is not new (I gave the project of building a model-driven Interface Builder on Solaris in 1993 to the students in my Object Oriented Programming class at the Faculte des Sciences de Luminy, home of Prolog), it becomes newsworthy again as Microsoft finally adheres to the idea. Of course, it does not mean at all that the Redmond giant is giving up what made its fortune and fame; on the contrary, they have announced a new language, XEN, as a revival of X# (a now dead semi-stealth project). XEN provides definite advances compared to existing programming languages -including C#. In particular, XEN bridges the gap between Objects and Documents by utilizing XML concepts at its core. It also deals with other problems like object-relational mapping with language constructs rather than APIs. With message handling capabilities and long running context management, XEN could well become the SOC community's favorite language, and, for once, a language built for an architecture and not an architecture on top of a language.

And the innovation does not stop there: Applications blocks, like the User Interface Process and Asynchronous Services, and frameworks like ShadowFax all point to the same direction: Microsoft is quietly weaving a fabric upon which BPM solutions will flourish.
The most visible aspect of this strategy is maybe the decision to move "Orchestration" out of Biztalk and into its Longhorn operating system to strengthen its SOC platform. Technically, this is, again, a perfect move; Orchestration is a very important, but low level, language. Like transactions in its time, it is particularly well suited for developing, for instance, the services model-oriented business logic to manage the events affecting business entity states in long running activities. Actually, Orchestration is part of a larger family of concepts called "Coordination." Composition, transaction, and choreography are all forms of coordination. The concept even has its standard: WS-Composite Application Framework (OASIS TC WS-CAF). This standard is architected very precisely and will become the keystone of constructing software for service oriented architectures. I would really not be surprised if Longhorn features a "Microsoft Coordination Server" as a replacement for Microsoft Transaction Server (MTS).

Of course, this move casts a shadow on the future of BPEL as "the BPM standard," but not as an orchestration language. I was shocked to hear both Norm Judah and Keith Short's presentation implying that BPEL is focused too much on EAI and is not at the right level to describe Business Processes; Keith concludes that new Domain Specific Languages will be needed! Could it be that Microsoft is already working on a new and true BPM language? I don't know, but it would be a crime not to! Anybody who has looked seriously at BEPL can only come to Norm and Keith's conclusion that this language is not well suited for a comprehensive BPM approach.

We could also try to predict what such a language would entail. In my humble opinion, there are three irreducible concepts, *event*, *activity,* and *state,* at the foundation of a business process definition metamodel. Most approaches so far have focused on events (Pi-Calculus) or Activities (Petri Nets), but hardly any have focused on state--business entity state, that is. In the best case,

parts of an XML document are managed in the data flow of a business process definition, but no business entity semantic is available to the business process.

Another key characteristic of a modern Business Process Definition language should be its inability--yes, inability-- to describe a business process from A to Z with all explicit exception paths: I was talking to a large company's chief architect at a conference last November. His conclusions were stunning: his company had just gone through a complete mapping of all their processes. I said, "That must be a great asset. You are ready for BPM." "Actually, no," he answered. They had come to the conclusion that because of the way they had modeled the processes, the resulting definitions were far too complex and, consequently, brittle. Nobody could comprehend the impact of any changes, let alone move the enterprise architecture to a BPMS.

Microsoft seems to be patiently establishing technical services, tools, an application model, programming languages, and domain specific languages for service oriented computing, with an appetite for innovation that is unmatched in the industry. Microsoft understands that the future belongs to those who can master "connected systems." Today, the value of an application is not defined by functionality any longer but, rather, by connectivity to its environment. All these new ripples on the computing landscape, combined with a new Business Process Definition Language, could well prove to be the *Furculac Caudinae* of its competitors. In the mean time, these changes will have enabled BPM to spread its wings, at last, after five long years of various trials and errors.

_____

[1] Steve Cook's article, Domain Specific Modeling and Model Driven Architecture, was published as part of Dave Frankel's MDA Journal and appeared in January 2004. To access the article, go to www.bptrends.com, then publications/columns/MDA Journal.

Jean-Jacques Dubray is the Technical Architect and a Senior Member of the Technical Staff at Attachmate, where he is working on SOA and BPM technologies. Jean-Jacques was the editor of ebXML BPSS 1.1 specification and has contributed to the design of many specifications at W3C, OASIS, OAGI, and BPMI. He also maintains a website on which he publishes information and insights into BPM technologies. See www.ebPML.org.