The Third Wave

February 2005

Howard Smith
Chief Technology Officer
(Europe) of
Computer Sciences
Corporation (CSC)

Peter Fingar
Executive Partner
The Greystone Group

co-authors
Business Process Management: The Third Wave
and
IT Doesn't Matter -- Business Processes Do.

authors@bpm3.com

Industry analysts agree that the next generation is upon us. They only disagree about what it looks like. Sound suspicious? Are analysts unsure of what the next-generation innovation is?

Everyone is talking about service-oriented applications, but that's just distributed computing, like CORBA, reborn on the Web, using XML APIs. The analysts also agree that the new Web services-based infrastructure will support lots of new applications, of which BPM is one. They position BPM as the application that knits together independent services into cohesive solutions. All well and good. But there is less agreement about how to do that.

One camp sees the next generation as the next generation of software development, i.e., programming, with the objective of forming composite applications from services. That will work, of course, and it's really no different in kind to what we've had for a decade, only now it's easier to achieve because of ubiquitous Internet and Web –standards. Companies involved in producing integrated development environments (IDEs) for this type of distributed programming are trying to make it easier by including in their technology stack a workflow-based or BPEL-based process engine. The engine does some of the work that would have to be done by programmers to code the interactions between services. These vendors have realized that engines are a good idea in software design. The same goes for rules programming with the inclusion of a separate rules engine. But some vendors are looking at this next-generation programming a little differently.

What if, they surmise, instead of just supplying a programming language, compiler, and computer to run it on, they could supply some kind of server, in addition to the process engine, that provides all kinds of pre-built services and interfaces into the network infrastructure, and into other software products? Wouldn't that speed up development? The idea is similar to the use of a J2EE application server, but because the objective of the new server is not just Java programming, but distributed service stitching, they have called their new concept—to pick up on a hot term—a BPM server.

The idea of a BPM server is this: To get useful composite applications built, you need this new process server thing. It would feel much like a J2EE app server in use, and its development environment would feel much like today's best of breed Java IDEs. The only thing that's changed, fundamentally, is that the programming language is BPEL.

We can see that BPEL is envisaged to be similar to programming because some vendors, to fill in gaps in BPEL (which was never designed as a programming language), have found it necessary to pen BPEL-J—Java extensions for BPEL. That's only natural, for most BPEL servers are really extensions of existing technologies such as Java, J2EE, .Net etc. Many analysts see these developments as a natural evolution in software development, and so they are. The objective is still the development of software. But in making this software transition to the process layer, analysts report that BPM servers—given the diverse technologies they rely on and their inherently distributed, Web services-based nature—can be challenging to maintain.

The Third Wave

by
Howard Smith
and
Peter Fingar

February 2005

Analysts point out, correctly, that performance of BPM servers will be a concern in very high volume scenarios, and this is borne out in practice. Some question whether these BPEL-programming servers will ever be as resilient as Java or C# programming environments have become. At the very least, it will take some time (years) before they are as good. Look at how long it took J2EE to mature. Some say there are still issues in the Java development space. Looking at these BPM server products, and all the different pieces of technology required to make them work, is there a better architecture? After all, new BPEL-programming IDEs and their underlying servers are trying to do something very complex. Developing a composite app from multiple distributed Web services is not the same as developing a piece of single-threaded Java code. Processes cross systems, many services, run for hours, days, or weeks, implement complex rules, interact with people, and more. And, they must provide processes that are persistent and transactional. Making all this work reliably is a formidable undertaking both for the vendor, and the programmer. Just look at what is happening underneath: messages flying about, distributed transactions, data transformation, service invocation via complex APIs into other applications, and the like.

To make it easier for the developer of today's next-generation distributed programming servers, the vendors are adding graphical development tools and simple notations. The idea is that a simple notation, a diagram, can specify both individual services and specify their interactions to create a business process.

Most of you will now have seen a BPEL-style process-programming tool. And vendors, knowing how complex this is to support, are using different approaches to turning those BPEL diagrams into real code. Some convert the diagrams to Java and then rely on the Java virtual machine for execution, or the equivalent in .Net. The program then executes like any other program. Other vendors embed a process engine, as we mentioned earlier. Here, portability is a real issue, given the fact that there is no reference architecture for these products. They are very different from one another, with vendors trying to extend their existing technology in the best way they see fit to achieve what the business needs, a simple approach to creating and deploying new business functionality. It's very early for such a reference architecture to appear, because even the core standards involved, such as BPEL, have not yet been standardized. And BPEL servers are more complex than application servers because the latter have not had to address long-running transactions associated with processes. It is these advanced forms of functionality that BPEL servers will have to tackle, and we have no doubt they will. But, let's not forget that viewing the world of technology in terms of technical transactions may not reflect the types of transactions needed in a world where long-running persistent processes are the norm—in fact, where the process itself can be considered a kind of transaction whose impact needs to be considered in the round.

Many of the problems with BPEL BPM servers will be solved, of course, and higher level programming via BPEL, on top of Java etc., will emerge in a rock solid form. But we should not expect this too soon. Indeed, in a cursory look at the technical complexity of combining services provided by diverse software to create composite applications, analysts have noted that today's BPEL servers

The Third Wave

by
Howard Smith
and
Peter Fingar

February 2005

must still be regarded as a form of specialized programming that preserves the dependence on technicians for process change. There is no doubt about this. But what if there was a better way to achieve the same result?

Analysts point out that unless the BPEL engine is very smart indeed, and unless development patterns are provided to the programmer in advance to cover complex tasks, such development tools will not be able to provide the required performance and productivity to cover even typical processes, such as workflow. Workflow, of course, has been around for a decade or more in business. Is it realistic for BPEL servers to be programmed to perform as a best of breed workflow implementation? For most, the BPEL implementation, being extensions of existing technology, simply won't be up to the job. Try writing the logic of a workflow engine in BPEL and see what we mean.

So to get around this limitation in the implementation of BPEL engines[1], some BPEL-server vendors are considering the inclusion, in their product, of a separate "workflow" engine. This engine, an addition to the BPEL engine, is required to make up for the fact that the BPEL-engine itself cannot support the required performance of the complex BPEL patterns required to emulate workflows. For example, concurrency is a real challenge. If any of you have seen a BPEL model of even simple workflow, then you will know that the concurrency requirements are extreme, to say the least. If one then takes into account the functionality of modern process tools, designed for business people, that extend workflow into new rich models that support coordination, negotiation, commitments and many other forms of person-to-person interaction, we must express extreme scepticism that a typical BPEL-process-orchestration approach is likely to be able to replicate those solutions. Indeed, even using a high-end workflow engine, let alone a low level BPEL-engine, these advanced processes cannot be easily replicated and would take, if attempted, a year or two of development. Put simply, most BPEL engines, based on what we have seen, won't have a high enough performance in the near term to replicate existing workflow and collaboration technologies. The BPEL process models to execute today's sophisticated workflow, negotiation and commitment processes across connected loops of interaction among multiple user roels will be too complex for today's first-generation BPEL implementations. Indeed, emulating such functionality is so ambitious that, for many vendors, it doesn't even figure in their product roadmap. Buyer beware: A tick in the BPEL support box may not mean the product is a BPMS.

Thus, it is our conjecture that, over the next years, we will witness a divide forming between those vendors who are primarily focused on programming metaphors and those who are focused on enabling business people to manage their own processes. Indeed, this is already happening.

While the infrastructure vendors will perceive the business-oriented BPM tools as "just applications," business people will nevertheless find great value in them. And while the business-oriented BPM tools will never be general-purpose programming environments, the business will not care. They did not care that a spreadsheet could not solve differential equations; they got on and built needed business numerical models, freeing themselves from technicians. As we have

[1] Not necessarily a limitation in the BPEL language itself

The Third Wave

by
Howard Smith
and
Peter Fingar

February 2005

written before, there is a symbiotic relationship emerging between the business-oriented BPM applications and the infrastructure of Web services.

Whereas it is theoretically viable to build any application in any environment, including using a BPEL-based programming server, some ways of building the solution will be easier than others, and many companies will choose to acquire an application, rather than build it. That application is BPM. Also, let's be clear. BPEL-programming servers, like J2EE, are designed to support the development of solutions, rather than *be* the solution. In fact, a business-oriented BPM solution could itself be developed on one of these new BPEL servers. Indeed, we suspect this is what is happening. Today, the large vendors who are beginning to supply BPEL-IDEs and the associated process servers, will tomorrow offer, based on them, a full blown business-oriented BPM solution that can be used directly by business people. They do not expect BPEL to be used by business people today. They intend to build on
their server designed for programmers to create the business-friendly solutions of tomorrow.

But what if there were another possibility?

We have written much about the possibility for a BPMS. A BPMS is not a BPEL server. The very processes that the BPEL-based programmers want to produce can be described in another way, not using programming, but by using clean, simple, algebraic forms. These forms, called pi calculus processes, can be managed as data on a platform designed for the purpose.

Just as an ERP system is rock solid because of its underlying data model, so a BPMS can have a rock solid underpinning. Just as a spreadsheet is rock solid because it has a finite state machine so can a BPMS be rock solid. And by adopting the science of processes, many technical issues associated with producing a reliable process management system evaporate. RDBMS designers learned this a long time ago. Once this science is appreciated, new options arise. For example, it is possible to build not just a way to program such processes, but to manage them before, during, and after execution. Workflow management system designers also understood this two decades ago, and have been trying to impress upon the industry and their customers the power of the approach to development practices ever since. Everyone knows that with a good workflow product, deploying a new flow is more like a configuration activity than it is development. Now, those same ideas can play a greater role in complete systems design, since, unlike workflows, processes are not limited to information flow, but extend to calculations, data manipulation, and more. Processes, in this sense, can be represented and executed, and products to enable their management can emerge, as occurred with data. For companies don't just want process development that use BPEL, they want a capability to manage all processes, however developed, as business assets, as they do with data today, using the RDBMS. The BPMS is our name for such a platform.

While the BPMS is compliant with BPEL-standards where it touches the infrastructure to invoke services—including other applications, etc., the BPMS, like an RDBMS, can expose to the business, reliable tools for top-down process design. And a BPMS is compliant with many other standards internally, so that,

The Third Wave

by
Howard Smith
and
Peter Fingar

February 2005

for example, if one wished to use a different rules engine or security server to the one provided with the BPMS, they could be interchanged. Thus, the BPMS is more of a *capability*, or a *platform*, than an IDE for software development. A BPMS is more like an RDBMS in use than a J2EE application development server.

So while a BPMS is a new and different approach to a BPEL-server, and while it may not be used for all programming tasks, as far as the business is concerned, its discrete process model is sufficient and, with its process management capabilities, it serves business needs more fully than just as a new way to develop code. Indeed, some analysts are beginning to see that the BPMS is an important contribution to the development of the reference architecture for that "next generation" that the analysts say is here. For what is at stake is whether the "next generation" is just a new way to develop distributed software processes, or a new platform for business process management. Such a capability differs from a programming server in many ways. The primary role of a programming environment is to create new programs. While a BPMS can be used to create new processes, it will often be used to describe existing processes in the business environment so as to bring them under management, just as an RDBMS is used to define a consolidated data model. So when IT industry analysts talk about the "next generation," not only do they disagree on details associated with the next generation of distributed programming, they may have missed or have chosen to omit this other possibility, a BPMS.

There will, of course, also *be* a next level of programming. After all, until BPML, and now BPEL, we had no way to describe, in a language script, distributed processing. We could describe single computing threads, using Java, C++ or other similar languages, but once those programs sent messages to each other we could only stand and watch. We had no way to describe the end-to-end business process, let alone, manage it. Now, with a BPMS, we can do those things, and let the BPMS do the wiring up of what is required to make it happen. The BPMS turns process design (not programming) models into implementations via a single step: deployment on the BPMS.

The BPMS is a significant *innovation,* whereas, IT analysts report some implementations of BPEL (which may be marketed as BPM) to be an *evolution* of programming. Analysts looking for the next generation won't find it in evolutions of programming and would do better to look towards the pi calculus: a genuine revolution in computer science.

The BPMS is an innovation in the same sense that the RDBMS was an innovation. It changes the game, enabling new business-led functionality and freeing business people from further dependence on programmers. Now that the BPMS exists, programmers and technicians need not see every problem as a programming problem. IT service providers now have the option to acquire and implement a BPMS capability in the enterprise, and to deliver a process management *service* to process owners, just as they deliver an email *service* today. Indeed, technicians who understand this possibility welcome it, for it allows them to delight their business sponsors. The CIO with a BPMS reduces process development to deployment time and resource costs dramatically. By doing so, IT staff can

The Third Wave

by
Howard Smith
and
Peter Fingar

February 2005

move on to more interesting work, instead of forever running to keep up with every business-led request for support from IT. Creating and deploying a process should be, in an ideal world, no different than creating and sending an email.

Business people don't care about pi calculus in the BPMS any more than they care about Codd's relational model in the RDBMS. But to technicians developing a BPMS in practice, process calculi form the inspiration and theoretical basis for their work. As a result of these foundations, not only does the BPMS provide a solid platform for business processes comparable to using the RDBMS for data management for ERP systems, the BPMS provides a platform for the *real* next generation of applications: process-oriented applications.

This *BPMS-centered* next generation will be based on moving BPQL into the process development environment, not unlike the movement of SQL into the data environment in the past. In this sense, a BPMS is very different and alien to Java or BPEL programming, and so should it be, designed as it is for building tools to support end users, not just technicians. Yet even to technicians, the BPMS is intuitive, because they too know the value of SQL in relation to the RDBMS for the management of today's business critical assets—data.

The RDBMS combined with SQL and 4th Generation Languages (4GLs) were once the next generation, but now we must move on to the *next* next generation that enables *process-based competition* in today's fierce global economy. Companies don't want more software development, they want results—meaning they want a powerful business process management *capability* that gives them direct manipulation of their business-critical assets—processes.

For more details of the BPMS, see Howard Smith's paper also published this month in BPTrends, entitled "What A BPMS Is."