Enough is enough in the field of BPM: We don’t need BPELJ: BPML semantics are just fine

Or is it back to the 3GLs?

A response to a white paper by IBM and BEA entitled
BPELJ: BPEL for Java, March 2004

Howard Smith

Why has IBM and BEA proposed a new business process standard called BPELJ? The new paper, whose authors include key players behind the move by IBM and Microsoft to combine WSFL and XLANG to create BPEL (widely viewed as a move against the then accepted open standard BPML), argues that:

“This white paper proposes a combination of BPEL with Java, named **BPELJ**, that allows these two programming languages to be used together to build complete business process applications.”

So BPMJ is WSFL, plus XLANG, plus Java.

But many in the burgeoning community developing process standards thought that the intent of developing BPEL at OASIS was precisely this: a language to build complete business processes. Indeed, many vendors were led to the conclusion that they should abandon development of BPMI.org’s BPML, which was the accepted approach prior to the publication of BPEL. So what’s going on?

A short history lesson

On November 13th 2002, BPMI.org released BPML 1.0 following a two-year *open development process* that began with the release to the community of an important idea, BPML 0.1 in 1999. BPML 1.0 is an open, complete, formal and royalty free standard for the expression of business processes. Sixteen months on, March 2004, OASIS have not yet completed BPEL, and BPELJ is just a white paper (unless IBM and BEA are going to swing another surprise on the industry and release BPELJ as they as did BPEL.

Originally BPEL was BPEL4WS. Before it became input to a treacle-like OASIS standardization process, it was released to an unsuspecting public in August 2002, just 3 months before BPMI.org released BPML 1.0 and completed that part of its work. And this was strange, because IBM was a member of BPMI.org at the time, having joined the initiative 18 months before and just 4 days before BPMI.org released BPML 0.4 in Las Vegas. Everyone got in on that press release.

BPML 0.4 was widely regarded as an innovation, unifying (not merging) previously disparate computing models being pursued by different vendors. By contrast, informed observers noted at the time that BPEL4WS was an unhappy merger of two quite different models, Microsoft’s XLANG (still being developed by them today some believe) and IBM’s WSFL, a workflow inspired language for using workflow engines to drive Web services.

Since IBM was a BPMI.org insider, they could have adopted BPML, but chose not to. And what is even stranger was that Microsoft, their partner in BPEL, had solid engineers and theorists that understood the BPMI.org work and their interest in Pi Calculus models. Indeed, Microsoft was working on similar ideas in 1999 when BPMI.org instigators met for similar discussions. Many of us were going in similar directions. Sixteen companies founded BPMI.org mid-2000 to create that focal point. Within 18 months the group had 200 members. Yet there were always doubts about the commitment of some large players. While this is somewhat understandable, I believe that genuine opportunities to accelerate the realization of BPM were lost.

How did OASIS get involved?

OASIS runs technical committees (TCs) for a living. Many OASIS TCs are concerned with the standardization of XML tag sets to support various horizontal and vertical industry schemas and applications. More recently they have become the host for vendor led technical committees. OASIS imposes good process on TC work, but the large number of TCs that OASIS host, with sometimes divergent, overlapping or contradictory aims, leaves OASIS with no clear focus. This is in contrast to W3C, BPMI, OMG, OAGIS and others, who have clearer overall missions. BPMI.org itself is highly focused on BPM and BPMS. They view PROCESSes as a new technology as OBJECTs were when first proposed. While BPMI.org is nowhere near as established as the OMG and the W3C, BPMI.org is thought of as an innovator in the processes field, creating new industry momentum in the area of BPM. So how did OASIS end up doing BPEL?

Simply, IBM and Microsoft needed a host for the merged XLANG-WSFL BPEL TC it intended to lead, and a standards process that would give the resulting BPEL specification some credibility. This could also have been achieved at BPMI.org or W3C. The choice of OASIS is shrouded in mystery. In fact, BPEL could have been submitted to BPMI.org or W3C.

W3C has huge credibility due to its work on WWW from the past, and had developed an interest in a particular aspect of process, namely choreography. W3C had convened a group, WS CHOR, still active in a small way today, and BPMI.org sought to submit some aspects of BPML to them, through a specification called WSCI (Web services choreography interface). They believed that the whole BPML story (big spec) would be too much at that stage of W3C’s interest. Maybe WSCI played a role in frightening off IBM and Microsoft from submitting BPEL to W3C. There is the story of Microsoft resigning from the WS CHOR group after only two days of interaction. Once again, the decision is shrouded in mystery. In any case: OASIS got BPEL4WS as a result, and because of IBM and Microsoft presence in leading the TC, smaller vendors flocked to the show, despite having previously been supporting BPML. As a result, some vendors who had been working toward implementations of BPML switched tracks to BPEL, or put things on hold. But there was also confusion about what a BPEL, or BPML, implementation actually meant. (It’s about whether you are implementing an engine for individual BPEL or BPML threads, *or* providing the Pi Calculus machinery to create end-to-end processes consisting of a multitude of deeply nested BPEL or BPML threads). Most vendors in the OASIS BPEL TC are in fact extending existing technologies rather than creating new BPMS products. Either way, the industry ended up with BPEL, BPML and WSCI, several other minor specs and

… not to be forgotten in all this, XPDL and other work by the WfMC.org. (Some press coverage through the later phases of this at www.bpm3.com/standards/)

So what's with BPELJ?

The authors of the BPELJ white paper argue that there are "*programming* tasks best suited to BPEL and *programming* tasks best suited to Java." While there is no doubt that Java has a long life ahead, and will be used for a host of programming projects for which the use of business process languages such as BPML are inappropriate, I don't see BPELJ as a positive step for those building or using Business Process Management Systems (BPMS). BPM, not programming, is the core mission of BPMI.org.

The questions addressed in my paper are therefore these:

1. Using IBM and BEA's *software development* terminology, what are the "programming requirements" needed within a business process modeling language?

2. What is missing from BPEL in this regard?

3. Are the items missing in BPEL (as evidenced by the BPELJ proposal) present in BPML? Have they been there all along?

4. Is extending BPEL with conventional programming language constructs (Java) sensible for those doing Web services and composite application development?

5. Is extending BPEL with conventional programming language constructs (Java) sensible for those building or using BPM systems?

   Note: A lot has been written in the past about the formal basis for BPML and its significance to process management. Does the extension of BPEL to include Java further distance BPEL from the requirements for a formal representation for BPMS? The analogy is with RDBMS that rests on its formal representation, the relational data model of data.

6. Is there another way to achieve the same ends as intended by BPELJ, other than by introducing new Java elements?

7. Will BPEL be extended with other languages? What does that mean for portability?

8. Did the vendors who stopped developing BPML in favor of BPEL imagine that to make it complete it would need to be extended with Java? Or did they imagine the BPEL language would be completed and, eventually, rival BPML? Did the vendors that switched track from BPML to BPEL believe there was good faith by themselves and BPMI.org and that everyone was working toward a merger of BPEL and BPML? Or were they thinking in an entirely different direction?

9. Does the publication of a proposal for BPELJ indicate a divide between those who are building new *programming* systems (application server vendors for example) and those who are building BPMS? Does this divide matter and how viable is it to link the development of BPM standards with those intended as programming standards?

10. What is the status of BPML and can it be used to build Business Process Management Systems? With the publication of proposals for BPELJ, what is the safe path for those developing BPMS?

11. Do merged computer languages, like BPELJ, create a BPM capability for business people? Isn't a BPMS needed that provides, in one capability, not one standard, a unification of the world of software engineering and the world of process engineering, improvement and reengineering? Has this something to do with the difference between BPMI.org's DDA (Design Driven Architecture) and the OMG's MDA (Model Driven Architecture)? Does Java in BPEL reduce its formality required for BPM (Pi Calculus, Join Calculus, Petri Nets etc)?

12. Despite the existence of BPML, IBM and Microsoft combined WSFL and XLANG to create BPEL. Now a related group of authors propose to combine BPEL and Java to create BPELJ. Customer's were already confused enough as a result of spec proliferation. What are they likely to be thinking now? How does all this mashing around with disparate specs compare with the consistent vision of BPML.org that published BPML 0.1 in 1999 and went on to release BPML 1.0 in 2003 through an *open process and public review period*.

13. Where do workflow standards fit into all this?

14. Can we hypothesize any underlying reasons for the publication of a white paper proposing BPELJ that are not stated in the IBM and BEA white paper?

I cannot answer all these questions here, because I don't have all the answers, or even all the questions? But I'd like to set down some of my observations. I make these observations from the perspective of the broad shift I perceive, from data digitization and exploitation enabled by the RDBMS and its formal foundations, towards process digitization and exploitation enabled by the BPMS and its formal foundations.

See Figure 1 below.

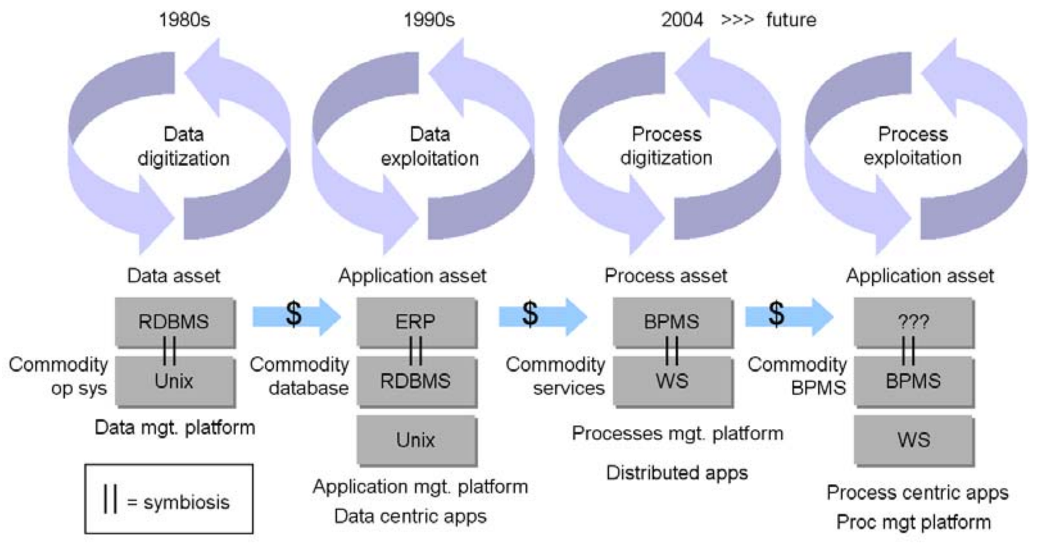

Figure 1 – Successive waves of digitization and exploitation of digital assets create new operating systems, platforms and capabilities – A symbiotic relationship between BPMS and Web services emerges

Programming constructs in BP languages

Business processes inevitably include computation, logic, data manipulation and many other aspects of what has traditionally been called "programming." While languages such as BPML are not intended as general-purpose programming languages, BPML includes all required constructs. (See page 113 of *Business Process Management: The Third Wave*). How could one have, for example, an Order to Cash Process, without computation and data manipulation across numerous systems and services? BP languages have to be complete, and able to do everything a traditional programming language like Java can do, yet this is not the same as saying that such a language will be used for all purposes. Spreadsheets are an example. Although they can theoretically perform all possible calculations, in practice they are used for all appropriate tasks for business people.

Business people understand that when their pure programming requirements exceed that of tools like BPMS, based on languages such as BPML, they ask IT developers to create or find additional code for them that can be included as participants in business processes. This code does not have to be, and should not be, included in the process language exposed to business people. For what would that mean for process management thereafter? If we go down the BPELJ road, won't business processes start to resemble software programs, with all the attendant difficulties that paradigm has created (stovepipes, concrete logic, software development waterfall lifecycle etc) and for which BPM was designed as an antidote?

For BPMS vendors, the objective is not to overload their process language with a programming language, but to allow for the reuse of existing IT assets—packages or services—within the process definition through appropriate participation and projection. And do so safely.

The result is a hybrid environment. A BPMS allows the business to selectively digitize those aspects of end-to-end processes that are relevant to their current business objectives, and place those under business control in terms of their exploitation and lifecycle of improvement and optimization. This is similar to what RDBMS enabled in terms of data management and data analysis.

BPELJ acts in the opposite direction to BPMS. Whereas a BPMS lets the business take control of the process and leverage existing IT systems to a depth and breadth limited only by the capability of BPMS and BPMS users, BPELJ is for programmers who want to extend process languages with programming features.

What's missing in BPEL?

BPMI.org has previously encouraged its members to look critically at BPEL as compared with BPML, yet at the same time to participate fully in the OASIS BPEL TC. Why? It has been obvious from the start that BPEL omits features of BPML necessary for process digitization and management, and since BPEL is not in the control of BPMI.org, it's unclear of whether these features will ever be added. Why? The members of the OASIS BPEL TC are developing BPEL for many *different reasons*. It's not a BPM group per se. From the outset, and this is still true today, there are virtually no BPM or Workflow companies in the BPEL TC committee. The group is dominated by software infrastructure vendors, typically application server, EAI and distributed computing vendors, as well as startups developing new BPEL engines (presumably to sell or OEM to other companies). All of the discussions in the TC mailing lists are around esoteric technical aspects of Web services technology. By contrast, BPMI.org discussions are about how to represent, model, manage and improve business processes. Web services are important, some believe them to be the next operating system environment replacing Unix as the platform for business capabilities, but let's not confuse these technical discussions on the plumbing with the new process-oriented applications that will exploit the new Web services operating system. BPMS is the killer app for Web services, as RDBMS was for Unix. The purpose of BPMI.org is to guide people implementing BPMS, and its interface to the Web services substrate. The objective of BPMS is to extend a BPM service to the business, with tools able to discover, design, deploy, execute, operate, analyze and optimize business processes.

Of the BPMS vendors, only Intalio is significantly participating and contributing to the OASIS BPEL TC discussion. A quick look at the BPEL mailing list over the last few months reveals this to be the case. So why is Intalio so deeply involved in BPEL? As chair of BPMI.org, and an author of BPML, they are making a genuine attempt to ensure that BPEL will serve the needs of BPMS vendors. But the matter is not in their full control, as you can understand. Is BPML on hold, pending the outcome with BPEL?

The very fact that BPELJ has been proposed, and based on what can be inferred from the BPELJ paper, it is possible that in early implementations of BPEL by companies like IBM and BEA, and possibly others, they have found the need to call out to Java programming in order to create complete processes. If true, this is not a surprise. To give to a process language the full power of a conventional language (remembering that this is different from saying it will be used for all purposes) is a formidable challenge. It's not just about adding the right tags. They have to be added in the right way and the language has to be interpreted with an appropriate execution model. It needs the expressive capability to be able to combine multiple communicating and cooperating threads of distributed execution (mobile processes) and this in term requires a new computational paradigm to support that, a new engine. This is called a process virtual machine (PVM). Such a PVM is not the combination of two engines, one for what IBM and BEA call "programming the large" (BPEL) and one for what they call "programming in the small" (Java). A PVM is a unification of two separate paradigms, the computational logic typical of past and modern programming languages (3GLs like Java), and the types of distributed interactions that occur through communication in messaging middleware (BPEL). It was Robin Milner's great insight to unify computation and communication in the Pi calculus, and Pi calculus machinery lies at the heart of a BPMS. (Read Milner's book, and selected articles at www.bpm3.com/picalculus/)

> Think about a Java program executing. And then think of a COBOL program executing. The Java and COBOL programming languages perfectly describe what is happening in each thread, deterministically. But connect the two programs together to create a process, interacting through communication in a distributed computing environment and what's the result? How can that be described deterministically? That's what BPML is for.

Although appropriate middleware and EAI broker can make some of those BPML processes happen, those tools don't provide a way to *describe* what is happening. And if what's happening cannot be described, deterministically, it cannot be managed.[1] BPML can describe what is happening in real processes in a way that creates a description that can be externalized, digitized and maintained over time. That creates persistent XML data that captures the execution history and lifecycle, and future plan. In this sense, we could describe BPML as the world's first commercial strength language for distributed computation. It allows for the description of PROCESSES (see Do you Grok PROCESS? www.bpm3.com/grok).

The Transaction Giveaway

One of the key innovations in BPML was the support for transactional processes, not just transactions invoked outside of a process. Transactional support, and how to include it in BPEL, has long been a discussion point in the BPEL community. The BPELJ white paper comes clean, I quote:

> "Customers use J2EE application servers for building robust applications. One aspect of robustness is the support of ACID

[1] Actually, some middleware and EAI implementations cannot support some BPML semantics, such as Switch/Choice, due to the fact they are driven by more limited workflow engines. So some real world process behavior cannot be deployed using these technologies.

> transactions. Consequently, BPELJ extends BPEL by adding support of ACID transactions to business processes."

BPML has had transactions, including ACID properties, since the get go. Indeed, its model of transaction supports nested transaction, within nested processes, and across multiple BPML processes (i.e. within an end-to-end process). And if we can believe what the BPEJ spec says (Page 21 onward), even though BPELJ adds transactions, there appear to be limitations in the BPELJ transaction support when compared to BPML.

Does BPML have everything needed for BPMS?

BPML supports the concept of zero-code design-driven deployment. BPML supports transactions. BPML can reuse code in other environments. BPML processes can express complete, end-to-end executable processes.

For BPM, BPML can be judged complete. This is not a theoretical issue. As some of you may know, Computer Sciences Corporation has been involved in BPMI.org from the outset. Perhaps unique among large systems integrators, we have quite a bit of experience with BPMS. We know it works, and in our process work, our engineers have said they can find nothing that cannot be modeled in BPML that they would otherwise have to write in Java. Indeed, BPMS is allowing us to propose a new contract between IT services providers and end users, by extending to them a BPM service and allowing business control over processes within certain domains where this is desirable.

Taking two examples:

Example 1. The BPEJ white paper has a table on page 3, which sets out the types of tasks for which BPEL and Java are "best suited." One example for Java is given as:

"Deconstruct a document that has arrived—finding important values, converting them and then inserting them into other documents"

In one BPMS case study, a BPML process does just that, in the context of supply chain order management processes. It works just fine and does not break the BPML formal model of separation of control flow/data flow.

Example 2. Another programming case best suited to Java, not BPEL, stated by the BPELJ authors is:

"Calculate a value that will be used to affect the flow of control within the business process (e.g. loops and branches)."

Virtually every BPML process does that.

Are there performance limits if one takes a pure process language approach?

Performance is always a question when a new model of computing, such as BPML, is proposed.

A BPMS implements a new model of concurrency. It need not depend on Java threads or other operating system concurrency features. Those things would limit performance.

If BPML is doing things normally done in Java, what does the benchmark look like? The formal benchmark will be published in due course, but right now, in tests, we are amazed at the performance, even on commodity Intel hardware. Yet again it should be stressed: BPML is not intended to replace all *programming.* One could hardly write a BPMS in BPML, for example.

Does BPELJ make sense for programming and programmers?

Hybrid computer languages present real problems. Without good tools, languages such as BPEL are impenetrable to many, and BPELJ adds yet another layer of complexity, syntax and interaction between the BPEL and the Java environment. What's really needed is a simplification step, not an extension, for it's not sufficient to hide complexity under better development tools (a cleaver IDE). BPM requires a consistency and formality for it to be widely adopted as a platform.

Programmers will decide if BPELJ is a good programming language. Experienced programmers take the "horses for courses" approach. A good tool, like a hammer, needs a dedicated purpose. Overloading features in one language doesn't work for many programmers. Is there such a thing as a Holy Grail programming language? I doubt it. This might sit oddly having just read the comments about BPML, but there is a key difference. BPML is dedicated and single minded. In fact, its focus is discrete, transactional processes that are widely applicable. BPML is precisely what's needed for business people to create process flows that fix problems in their business. It's not overloaded with features. Quite the opposite, the number of primitives is tiny. CSC developers learn it in hours. The primitives and processes created from BPML are easily used to compose reusable processes that can be used to create larger end-to-end processes. Introspection over existing IT systems during process design permits a Lego-building-block approach to process consolidation. The same simplicity is true of the essential elements of Web services. Service orientation and process orientation are complementary and simultaneously two sides of the same coin. This is why BPELJ is not welcome. It fudges the issue, even if some development tool can hide the complexity.

Does BPELJ make sense for BPM? Does formality matter?

The emergence of a BPMS platform can only arise if it is underpinned by a formal and complete model that creates a step change in the capability exposed to the business, as occurred with the relational model (E. F. Codd) and RDBMS. Looking at BPELJ reveals a mix of models that cannot be reconciled as the basis for BPMS.

Looking at the white paper, the reader cannot help but be reminded of Stored Procedures and SQL extensions in RDBMS technology. Indeed, the paper states "A BPELJ process will execute on any platform that supports the BPELJ extensions to BPEL."

While BPELJ allows Java code to be directly represented and reused in a BPEL process definitions, so does BPML, but by different means. Rather than a fusion of two languages, BPEL and Java, BPML is (and BPEL could be) a single language that can describe (and therefore include) descriptions of functionality from many other languages. Industry commentator *The Register* has already noted that:

> "Given the reasons for BPELJ are laudable it would seem that BPEL.net (or maybe it should be called BPELC#) will be developed probably with a slightly different construct. If this happens then most people will use BPELJ or BPELC#, rather than just the base BPEL. Without the benefit of being able to share code the benefit of a shared BPEL will disappear and the two camps will diverge once more. Maybe what is needed is to generalize the concept of BPELJ to support any suitable language. So that the BPEL would remain standard and C# or Java, or another language could be included, or generated, in the relevant place.
> www.theregister.co.uk/content/53/36490.html

And that's just two languages! In business, the reality is that companies want to *reuse* code from many existing IT assets, not just do new development. In fact, the majority of what happens in business is applications *maintenance, consolidation and extension.* (Think process *improvement)* New development is rare. In one project where a BPMS was used, five different technologies were projected into the single BPML language environment to create a single BPML end-to-end process. And this is a simple case.

Can BPEL be extended to do BPELJ-like things without the addition of Java?

Of course—it's called BPML.

Indeed, the OASIS TC is making progress to extend BPEL towards BPML. This should continue. Some as satisfied with the progress of the TC. Others are frustrated at slow progress and the 18 month to 2 year delay introduced by BPEL over BPML. But there is progress in a BPML direction. A few of the recent TC issues seem to be lifted straight off the BPML spec (*foreach,* multiple input/output parameters, XPath variables, etc). Wise BPMS vendors are moving on, beyond the execution engine, to future developments in the field of BPM.

Care must be taken lest BPEL is overloaded with duplicate features during the OASIS standardization process. For example, in the original BPEL4WS specification, people noted that in some cases the spec provided alternative ways of achieving the same semantic. The addition of Java to BPEL compounds this issue in several ways. For example, according to the white paper, Join Conditions can now be specified in BPEL and BPELJ as XPath or also in BPELJ as a Java "snippet".

Are there implications now for BPM vendors who switched track to BPEL from BPML?

Most BPM vendors deferred a decision to implement BPML, and then BPEL, pending the BPEL-BPML debate, and, once the OASIS BPEL TC got under way, were still skeptical about implementing BPEL too early in the process. BPELJ might confirm their fears. It is terrifying for small companies to go ahead and implement unstable and evolving specs, which might pop surprises for them as the specs are developed behind the closed doors of IT gorillas. Small vendors don't have the luxury of unlimited resources.

Some cynics perceive that proposals made in advance of implementation by IT gorillas (e.g. BPEL4WS) can have a stultifying effect on innovation and the ability of smaller vendors to get traction for their products and services. Again, with BPELJ, it is signal that, for the majority of vendors, the situation with BP standards is still evolving. Some small vendors will wait and absorb new standards at their own pace. Others, who have developed process foundations in the past, will be first to market with BPEL compliance.

Does BPELJ Matter?

For many vendors not developing BPMS, BPEL, BPELJ and more to come are add-ons, little more than tick boxes in customer RFPs, as opposed to new technical capabilities of significance. But for those vendors who are challenging the status quo, and who are creating new products to compete with application servers and EAI tools, and who are suggesting new ways for businesses to use computers (BPMS), the impact can be significant if they are subjected to unstable and unknown directions set out by IT gorillas. In this case, it is vital that BPMS vendors keep to their own vision and path and not be distracted by incumbents' moves.

Did we need BPEL?

From the outset, BPEL was a two edged sword. It was a set back compared to BPML and, simultaneously, confirmation that BPML was the way forward. Various BPEL-supporting companies, both BPMS vendors and others, have stated that they would have preferred to stay with BPML, judging it technically superior and cleaner in design, and available right now. BPELJ won't help them now, except, perhaps, in one way. If BPELJ goes ahead, then the capability of BPELJ-based implementations will compete more obviously with implementations that grew up in the original domain of BPML. This is good news, for we shall see how these different architectures compare and contrast. Vendors can then compete on function, performance, and quality of service etc, not the underlying model. (As did RDBMS vendors who nevertheless agreed about he fundamentals of relational data management)

Were BPEL and BPML ever on the same path as some may have assumed?

Some observers felt that, when BPEL was announced, its authors had in mind the same basic model of computing as the BPMI.org founders had with BPML. They felt, "OK, so the end-game standard won't be called BPML, but if BPEL is effectively the same, all will be well, and as a small vendor, I'm much happier following IBM and Microsoft than BPMI.org." In light of the proposal for BPELJ, I'm not so sure.

The dialog and participation in the BPEL TC are all about *programming,* not process management. Are the needs of those extending application servers to BP-programming and those building BPMS diverging? They are indeed very different strategies as initial case studies show. The game may be changing, and with it, and direction BPMI.org should take in the future. No doubt new BPMS technologies will also come from IT gorillas. We have always looked forward to a time when several implementations exist (Page 234 BPM3W).

Where do software engineering and process engineering *intersect*?

In the BPMS—that's the whole point.

Where do workflow standards fit in?

The BPEJ white paper does not speak of workflow. Why? Both BPMI.org and the OASIS BPEL TC believe that workflow is a higher-level pattern that can be deployed on a BPEL/BPML environment. Computer Sciences Corporation has demonstrated that working in its labs. Thus, the emphasis going forward is the encoding of workflow semantics in lower level process languages such as BPML and BPEL. (Note: BPMS can also facilitate services from existing workflow technologies to participate in end-to-end process models.)

Are there reasons for the BPELJ proposal other than the technical agenda set out in the white paper?

We can only surmise. Clearly, BPELJ extends BPEL, so the authors presumably feel that BPEL has deficiencies. But why not just extend BPEL, rather than add Java? Perhaps both are planned?

IBM and BEA are the top two application server vendors. Perhaps BPELJ is about preserving and leveraging core application server technologies? Could this limit the success of pure play BPEL implementers? And BPELJ, if fully realized, will certainly create more competition for BPMS providers. Indeed, the paper states "A BPELJ process will execute on any platform that supports the BPELJ extensions to BPEL." Creating a tight link between the emerging BPM capability and the existing Java capability, allows incumbents to create vertical stacks of technology. Is this another attempt to lock in?

Or is it that IBM and BEA, and EAI companies, in their initial implementations of BPEL have hit problems? Performance? It is well known that languages such as BPEL and BPML, in end-to-end processes, create unparalleled requirements for concurrency and, with inappropriate implementation techniques, severe resource implications. Putting it concisely, for those with no BPML process virtual machine, BPEL implementers could hit limits requiring them to call out to Java for relief.

Collaxa: BPEL implementer: speaks out

Collaxa has already written on their Web site blog that, "BPELJ is one step forward and one step back." They go on to say, "My first impression was that that the BPELJ proposal was political maneuvering from BEA to wrap some 'standards look-and-feel' around the millions of dollars they spent on WLI. But I then realized that this proposal could really be a step backward for BPEL and web service orchestration hence this post." They point out six shortcomings of the BPELJ proposal. www.collaxa.com/news.blog.html

As context, Collaxa points out that the company "worked about 3 1/2 years ago on a concept called ScenarioBean that tried to bridge the gap between BPEL and Java.

ScenarioBean suffered from a lot of the same shortcomings." BPMS vendors know this all too well. BPMS rests on its ability to sit between the world of process and the world of existing technologies, and it is at this intersection that BPMS vendors have made significant progress.

LooselyCoupled.com have spoken out as well, dubbing BPELJ an "ill-conceived mongrel specification." www.looselycoupled.com/blog/lc00aa00025.html

And core BPEL must change, again

WSFL … XLANG … BPEL4WS … BPEL …BPELJ …

BPELJ proposes changes to core BPEL to accommodate Java. In what version of BPEL or BPEL dialect will these be included? The voting processes at OASIS will presumably decide this. Or will the BPELJ implementation, in practice, race ahead of the BPEL standard?

At the heart of this rocky roadmap lies a simple truth. A new BPMS technology, and the business discipline of BPM, require a formal model, and cannot be treated like a never-ending series of extensions to an ad-hoc computer language.

An ad-hoc straw poll on BPELJ

Talking to a few people in the community, a pattern emerges. Typical comments are these:

- I haven't looked at BPELJ in too much detail, but I'm not excited about creating variants of BPEL and tying them to specific languages.
- First Java extends BPEL, then perhaps C#, then Visual Basic, PHP, etc.
- The whole point of using Web services and XML technologies is to have platform neutral solutions.
- We're talking about IBM/BEA here. IBM will cooperate with Microsoft and try to differentiate BPELJ.
- BEA needs to keep Java relevant in any way possible, since essentially they're selling software development tools, not a BPMS.

Takeaways

Looking at BPELJ, it's hard to know precisely how it fits into the future of concurrent distributed computing languages. As a Java extension to BPEL it creates additional complexity. One wonders whether it might have been cleaner for Java to be extended with concurrency? After all, Java programmers need concurrency, just as business people do in their BPM tools. But can these really be the same language? Aren't we mixing apples and oranges? Isn't the IBM, BEA, Microsoft led XLANG, WSFL, BPEL4WS, BPEL and BPELJ evolution really about their existing strategies and competition between different programming paradigms? Do these debates really have anything to do with building the reliable foundations for BPMS?

The conclusion to the white paper states, “BPELJ makes it possible to use Java as a language for calculations and data manipulation, so that a single file can contain complete and cohesive description of an entire business process.” BPML accomplished that for business people two years ago. Is BPELJ any more than another experiment by programmers, another fork in the road for BP standards?

There is no doubt that the IBM-BEA white paper will be of interest to programmers. It’s already being discussed at programming sites. It presumably makes some sense to those extending app servers. But I seriously doubt whether it will have any impact on those trying to extend a BPM capability to business people. For BPMS, BPELJ is not the right direction.

+++