A Practitioner's Perspective

Alec Sharp
Senior Consultant
Clariteq Systems Consulting

asharp@clariteq.com

Weighing in on BPMN – What it's Good for, What it's Not

I put off writing about this topic for a long time because of some less-than-happy experiences discussing Business Process Modeling Notation (BPMN.) As I've observed before in a BPTrends Column, "like a smoldering fire, these discussions generate smoke and heat, but little illumination." I reasoned that I have enough stress in my life already without deliberately attracting anyone's ire, but I came to feel a bit cowardly, so I started sketching out this Column. Just as I got it organized, there was a burst of discussion on the topic in the BPM blogs, apparently triggered by Jim Sinur's observation (apt, in my opinion) that BPMN wasn't going to catch on with a non-technical audience. I've read some of the subsequent commentary, but I've left the thrust of this Column as it was – I want to get my thoughts out before responding to anyone else's. A tip of the hat and a big "thank you!" to Sandy Kemsley and Adam Deane, the bloggers I most rely on. I'd probably have missed the latest round of BPMN discussions entirely if it weren't for the great service they provide by summarizing and linking to other blogs.

BPMN – to each his own?

People who have attended one of my conference presentations or workshops on working with business processes know I'm not a big fan of BPMN, but that doesn't mean I'm a big detractor. I'm not a big fan of country music either, but I don't make a point of lecturing country music fans about their taste in music. You play country in your home, I'll stick with BritPop in mine. Unfortunately, that "live and let live" attitude doesn't prevail with some of BPMN's more vocal proponents. They aren't content with "playing" BPMN in their homes – they want me to "play it" in mine, and don't understand why I don't have any interest in it. Occasionally, the commentary gets a bit tetchy, with those who don't use BPMN characterized as "losers" or professionally lacking because they "can't be bothered to learn a few symbols" that fit on a one-page chart. (Yes, these statements have actually been made.)

BPMN – First Exposure

The first time I saw BPMN, I was being interviewed by a professor and some researchers from a large US university. The professor used my book "Workflow Modeling" as a text in the MBA program he headed, and told me the students (typically with 15+ years of business experience) liked the book's practical methods. Following on from this, the interview was intended to get a better understanding of why I'd chosen to do certain things the way I recommended in the book.

It went smoothly until a question that I initially misunderstood: "When did you achieve your breakthrough realization of the value of simplicity?" one of the researchers asked. I thought she was joking, so I probably answered with something flip like "Well, I'm a simple guy…" She persisted, and when I realized it was a real question, I answered that there was no "breakthrough" – it seemed self-evident that simplicity (not to be confused with "simplistic") was necessary to fully engage the people whose processes we were studying. "Everyone gets that, don't they?" I asked. Evidently not – the professor pulled out the OMG's BPMN 1.1 (I think) specification and passed it to me. He explained that this was giving his MBA students heartburn, and was a big reason they liked my much simpler methods.

I scanned the document, over 100 pages, and my first thought was "This will never catch on." If a notation took that many pages to explain, how was it ever going to get traction with a business audience? After looking at it a little more closely, though, I had my "aha" moment – it wasn't a *business process modeling* notation – it was a technical workflow specification notation, or, more to the point, a workflow programming language.

A Programming Language?

Even though it was called a "modeling notation," that's not how it struck me – BPMN was a *programming language*, and it seemed clear that it had been designed to specify workflow with the precision and rigor needed for implementation in an automated environment. It was a *visual* programming language, as opposed to the *text-based* programming languages of my technical past, but a programming language nonetheless.

"I've been here before…"

In fact, I remember thinking about APL (A Programming Language) when I saw the rich palette of symbols for events, gateways, activities, and the like, offered in BPMN. For those of you not familiar with it, APL was a powerful, array-oriented language that used a visually dense notation made up of mathematical symbols. Entire programs could be written in one right-to-left line, hence, a favored pastime of APL programmers: creating "one-liners." Here's an example I found on the web, Michael Gertelman's one-line APL implementation of Conway's "Game of Life" (catpad.net/michael/apl):

I used to teach APL, and loved coding in it (I liked Lisp even more), but I would never claim that it was suitable for a wide-ranging business audience even if they loved what APL programs could do for them. Some specialists – scientists, actuaries, and financial analysts, for instance – were very comfortable working in APL, but the general business population? – not a chance. And so it is with BPMN – useful at the technical end of the spectrum, at what I'll describe as the "specification level," but not at all relevant to the work I do in the business process arena.

Historical perspective

BPMN certainly isn't the first attempt at developing a common language for both developers and business people. Back in the 1980s heyday of CASE tools we saw various versions of this, often championed under the "programming without programmers" banner, but let's go back another 20 years to the grandmother of them all, COBOL – the Common Business Oriented Language. As Michael Coughlan (CSIS, University of Limerick, www.csis.ul.ie) stated, "One of the design goals for COBOL was to make it possible for non-programmers, such as supervisors, managers, and users, to read and understand COBOL code. As a result, COBOL contains such English-like structural elements as verbs, clauses, sentences, sections, and divisions. As it happens, *this design goal was not realized*. (Italics mine.) Managers and users nowadays do not read COBOL programs. Computer programs are just too complex for most laymen to understand them, however familiar the syntactic elements." Sounds like BPMN to me.

For a more succinct statement on COBOL, consider the typically spare words of Tony Hoare, one of my heroes in the computing field: "It aimed at readability but unfortunately achieved only prolixity." (C. A. R. Hoare, "Hints on Programming Language Design," 1974.) Web definitions of "prolixity" include "too long," "unduly prolonged or drawn out," and "marked by or using an excess of words." That also sounds like BPMN to me, at least when it's applied outside its intended domain.

What BPMN means to a non-programmer – me!

I spend somewhere between 200 and 250 days per year on the road, mostly doing one of two things – helping clients improve or reinvent their business processes, or running workshops to teach others the essential techniques for doing the same. As such, business process modeling, analysis, and redesign are central to my work. Programming isn't. The use cases and service specifications that I produce might be programmed in Java, Ruby, or Ajax by a developer, the

data model I produce might be "programmed" using SQL Server DDL by a DBA, and the process flows might be "programmed" in BPMN, thanks to the efforts of a workflow engineer, but that doesn't mean I have to learn Ruby, SQL Server DDL, or BPMN in order to do what I do. Those are no more part of my work than swinging a framing hammer is to an architect's work. Granted, the best architects I've worked with actually understand what it's like on a construction site. In the same vein, I've done my share of programming and database creation, and could surely become competent in BPMN, but those sorts of skills just aren't relevant to engaging business professionals in understanding and redesigning their processes.

Can we leave it at that?

I was happy enough with the conclusion that BPMN is a programming language, but it turned out that many of BPMN's proponents didn't see it this way. They saw it as a universal language – a lingua franca – suitable for use by both technical and non-technical audiences, as the introduction to the spec states. It hasn't worked out that way. This is one more example of a pattern that has been played out many times over the years – tools and techniques that are useful in a technical domain being pushed (or pulled) into the domain of business (or process) analysis where they become more harmful than useful. There are many examples, but UML (Unified Modeling Language) is especially relevant, because some of its principles seem to have made their way into BPMN. UML, a notation originally cobbled together from three separate sources, was intended to "depict the artifacts of a software system." (The 1.0 spec used words to that effect, and over the years wording has been added to make the point that it's good for modeling businesses, too; it isn't, and the updated wording strikes me as more about marketing than "truth in advertising.") Anyway, the fact that UML was designed for software specification didn't stop people from assuming that "modeling" included "business modeling," so we've seen countless attempts to use UML as a business analysis tool. Generally, they have been unsuccessful, and where they have worked, we see that the UML has been watered-down and "stereotyped" to the point that it isn't really UML any more; it's more like other widely used techniques. That's exactly the case with BPMN and the "small subset" argument.

Incidentally, I've been making the "it's a programming language" point for years, a perspective not always cheerfully received by BPMN fans. I've been happy to see other commentators make the same point recently, and I was at first surprised, then very happy, when I heard a senior executive from the OMG state at a recent conference that BPMN could just as well be described as *a programming language* as a modeling language.

Predictable Objections

At this point, I'm confident that some readers are cheering, and others are already considering heated responses to the effect that (a) I'm overstating BPMN's complexity, and (b) business people can quite comfortably understand, and even develop, BPMN diagrams using the oft-cited "small subset" of the language. These are among the arguments that drive me crazy. First, please follow this link and download the excellent BPMN 2.0 poster advertised at www.BPMN.org: http://www.bpmb.de/index.php/BPMNPoster. I'm not sure it's an improvement, but here's a shortened link as well: http://bit.ly/d2LIxa. (You can also find a BPMN 1.1 poster via www.bpmn.org.)

In response to the predictable objections, I'd say (a) there's no conceivable way anyone can look at the BPMN symbol set depicted on the poster and say that it *isn't* complex and *is* suitable for non-technical audiences and (b) the examples that are usually used to demonstrate the "subset" argument barely use 5% of the symbol set – typically a lane, a simple task, a simple sequence flow, and a gateway or two, probably exclusive and maybe a parallel for good measure. Folks, *that's not BPMN* – it's a few boxes and lines, just like we were using for many years before BPMN came along, and certainly not something that requires a poster and a massive specification document.

There might be some other questions or objections, so let's address those, and then run for cover!

Where does BPMN fit?

I've made some of these points before in my June 2009 BPTrends column "Boxes, Lines, Widgets, and Words: Managing Detail and Perspective in Models," including the idea that there are three kinds of models that can be developed whether the subject matter is business processes, data, or something else – scope, concept, and specification models. Here's an extract of what I wrote then.

Scope models

Scope models provide the big picture view and are used primarily to show (not surprisingly) what is in and out of scope for an undertaking. As a student said … they "let me see the whole story all at once." Visually, that model was just a set of labeled boxes within a larger box indicating the scope of the undertaking. They are so simple that in many organizations they don't even bother with the boxes; they just list the items, divided into "in scope" and "out of scope." Sponsors, architects, portfolio managers, project managers, and others will use these to plan and initiate work.

Concept models

Concept models are visually richer, but not so much that they would alienate the average person who knew something about the subject being depicted. They aim to support a shared understanding of the main concepts within some area, and their interrelationships. "Shared" is an important word there, because a primary objective of concept models is to get a diverse group of people on the same page, which is why this is a very high-value space. They might help analysts, business subject matter experts, and designers agree on the lay of the land before getting into the detail needed to develop a functioning product.

Specification models

Finally, specification models provide all the detail needed to construct some functional product. They will serve a much more technical audience, whether that is the people building a house, or the people building applications and databases. A critical point, often missed in discussions about modeling, is that the purchaser or user of the product being built may not be able to read the specification level model, which is intended for the designers and builders, not the owners.

Different models, different uses, different symbol sets

It is often possible to build scope models with just boxes, concept models with boxes and lines, and specification level models with boxes, lines, and other visual "widgets" that impart specific rules or details. Nothing is absolute – a scope model might add a few lines, a concept model add a few widgets, and a detail model will almost surely add not just more widgets, but different styles of boxes and lines, and supplementary pieces such as detailed instructions or drawings of very fine details.

Returning to BPMN, I'd say that it is useful at the specification level, and not useful at the conceptual or scope level, where relatively simple boxes and lines are perfectly adequate.

How do you model complex processes without all the widgets?

Again, this is a topic I brought up in the June 2009 column, so I'll just cut and paste what I wrote then:

One complaint I've heard often is "we need all those operators to show all the variations of what really happens in a business process." This baffled me until I realized that we were taking very different approaches to modeling. Rather than trying to put all those variations into a single diagram, I always break them out across multiple diagrams. For each business process, such as Hire Worker, I'll identify the main cases or variations, such as Hire Contract Worker and Hire Permanent Worker. There are usually more than just two, and, at a minimum, I'd draw a process model for each case. More often than not, however, there will be different scenarios for each case, such as hiring a contract worker under a government-sponsored retraining program, hiring an independent contract worker, and hiring a contract worker from an agency. Each of these would warrant its own diagram. This always sounds like a lot of additional work, drawing all these

different versions of the same process, but in reality it goes very smoothly. Clients can focus on one set of circumstances at a time, and one model can serve as the basis for another. Most important, you avoid building a model that is festooned with decisions – "if this is a Contract Worker then…" and "if this involves a Retraining Program then..."

The urge to employ detail-oriented modeling techniques in a conceptual (business-oriented) setting always reminds me of something the late Michael Hammer ("the father of reengineering") said years ago. He had been asked about the advisability of using CASE (computer aided software engineering) tools on a BPR (business process reengineering) project. He replied that it could be done, but "it will be like eating rice with a steak knife – it's going to be messy, and someone's going to get hurt."

One Model to bring them all and in the darkness bind them

When you bring all these different cases and scenarios together into a single model for implementation purposes, then, of course, you'll need to use more of the BPMN symbols. But to expect your business partners to be able to follow this diagram is ridiculous; no more reasonable than expecting them to review your C++ code. It seems that some people using BPMN recognize this leading to one of the more insidious side-effects of the complexity. Depicting an entire end-to-end business process in BPMN, with all the process's variants, would in many cases be mind-boggling in its complexity, so what we see, instead, are multiple BPMN diagrams, each depicting a single piece of the overall process. Missing is the comprehensible view of the *entire* process that is, in practice, vital to encouraging process change.

My next column will look at some of the specifics of how to create useful models without descending into "widget oblivion."

But it's the standard, right?

In some circles you'll hear repeatedly that BPMN is "the standard" or "the only standard." Maybe so, but that doesn't mean it's particularly widespread. I've found that BPMN penetration is far greater in the minds of industry analysts and pundits, tool vendors, standards bodies, and instructors than it is on the front lines where people are actually doing significant, hands-on process change work. We've heard of other social phenomena that are discussed out of all proportion to the number of people engaging in them, and BPMN seems no different.

Who really uses BPMN?

Every year, hundreds of people around the globe attend my workshops, including business analysts or systems analysts, business process professionals, business people interested in or responsible for process change, architects of various stripes, project managers, and so on. At some point, I usually ask, "Who is using BPMN?" and the first reaction is lot of blank faces. I'll explain, and then ask again. In a typical class of 16 to 20 people, I've never had more than 2 people say they were using BPMN, and a typical number is zero or one. That means I've never had a case where more than 10% were using it, which doesn't sound like a standard that's taking the world by storm, at least outside the technical disciplines

As an example, last week I completed one of the most enjoyable "Working with Business Processes" workshops that I've run in quite some time. I had an excellent group of 18 people, most of whom had significant experience in modeling, analyzing, and redesigning business processes. While discussing the modeling of business processes, I asked how many were using BPMN, and the answer was none. Only one was even aware of it, and that was because he'd attended a course a few years before, at another company, at which they were taught that you could employ BPMN to model processes with business people, and then seamlessly transition into your implementation environment. The company tried, but, of course, it was a pipe dream.

Do BAs use BPMN?

As Paul Harmon noted in a recent column, most process change is driven not by "business process professionals" but by business analysts, and BPMN has achieved very little visibility with this group. Even among those who are involved enough in the profession to join the IIBA (International Institute of Business Analysis – www.theiiba.org) – I'm told the rate is somewhere

between 20 and 30%. I'm a little suspicious of even this low figure, and suspect that survey respondents, as they often do, are answering according to what they think they *should* be doing as opposed to what they *are* doing.

What about business process professionals?

In the same vein, I've never seen a case where process modeling and analysis leading to significant process change involved the use of BPMN. I've checked with other well-known and widely respected professionals and hear the same thing. BPMN may be used at the implementation level, along with other programming languages that are part of the solution, but it has no bearing whatsoever on helping business people understand their process, assess it, rethink it, and design a new process. That's why, for me and every other professional I know who works with businesses in process change, BPMN is a non-issue. Except, of course, for the chorus telling us, our clients, and our analysts that we should be using it. I hope this will start to settle down.

Would anything make you supportive of BPMN?

In general, I'd be a lot more positive about BPMN if it was formally divided into two specifications. The existing BPMN spec would be renamed something like the "Workflow Programming Language" and a business oriented subset would replace the current BPMN. Of course, such a radical renaming wouldn't fly, so I'd be happy with formally defined "BPMN – biz" and "BPMN – tech." BPMN-biz wouldn't be much more than boxes and lines, but at least that way the uninitiated wouldn't be tempted to use symbols they don't really need.

Specifically, even though I don't see myself using it, I'd be happy if some of the more egregious graphical errors in BPMN were sorted out. Principal among these – the repurposing of our old friend, the decision diamond, into the multi-purpose "gateway." This decision (ha ha) seems to defy all logic, and certainly defies some precepts of good graphic design. In flowcharting, and the process modeling dialects that it spawned, the diamond has for decades represented a mutually exclusive decision point. Why would such a universally understood symbol be given so many different meaning, depending on the symbol inside it? When I explain this to my classes the response is inevitably astonishment. Why not use a more neutral and suitable symbol such as a circle with different symbols ("x", "+", etc.) inside it? Wait, that *is* what other notations did! Maybe, as they did with some of the basic start/stop event symbols, the BPMN designers were too clever by half in trying to be different than other notations.

Over and out

The point I've tried to make in this column is that BPMN is good for some things – technically-oriented workflow specification – and not so good for other things – business-oriented process modeling. Next time, assuming you're back, we'll look at techniques for business-oriented modeling using a minimum of extraneous symbols.

From the Trenches
Alec Sharp

BPTrends Linkedin Discussion Group
We recently created a BPTrends Discussion Group on Linkedin to allow our members, readers and friends to freely exchange ideas on a wide variety of BPM related topics. We encourage you to initiate a new discussion on this publication or on other BPM related topics of interest to you, or to contribute to existing discussions. Go to Linkedin and join the **BPTrends Discussion Group.**