

## BPM and MDA: Competitors, Alternatives or Complementary

### Contents:

### Background

### Competitors, Alternatives or Complementary

Workflow is a subset of BPML

EAI is a subset of BPML

A BPMS can morph to any process  
semantic

### BPM and MOF

### BPML.org Decisions From The Outset

### Conclusions



**Howard Smith**  
CTO, CSC Europe  
Co-chair, BPML.org

David Frankel has written an excellent article, entitled *BPM and MDA: The Rise of Model-Driven Enterprise Systems*.<sup>[1]</sup> It is a clear, reasoned and cogent argument and I recommend you seek it out. This article is a response to his and could have been entitled *BPM and MDA: The Rise of Process-Based Enterprise Systems*.

Frankel's article argues that BPM with MDA is more compelling than BPM alone, and that MDA with BPM is more compelling than MDA alone. He presents a strong case for some coordination between these two architectures. In doing so he avoids favoring one approach over the other. In my response I look at whether such a marriage is realistic. Are BPM and MDA competitors, alternatives or do they offer complementary capabilities? I use this opportunity to explain BPM by contrasting it with traditional software systems.

*Like Frankel's article, which presumably does not represent the formal view of the OMG, this article does not represent the formal view of BPML.org. Both articles can be considered input to BPML-OMG discussion about possible future work together.*

### Background

During the period when BPML.org was developing Business Process Modeling Language (BPML) the OMG paid little attention. Like others, the significance of this *executable process language*, with its *mobile systems* roots, was not clear to the OMG. Things changed when BPML.org announced its intention to develop a *notation* for BPML, the Business Process Modeling Notation (BPMN).

BPMN is overtly focused on *business processes*. Vendors of process modeling tools and Business Process Management Systems (BPMS) drive its development. These vendors, and many business analysts, had not found existing UML notation appropriate for business processes. This was not because of the shapes of boxes and lines, but was because UML methodology was oriented to software development, not process management. It was natural therefore for OMG to reach out to BPML.org, especially so, since new developments in the field of UML were beginning to address process notation. I can only conclude that BPMN sparked OMG interest precisely because it was a notation and because UML 2.0 was in development.

Despite some interaction between OMG and BPML, the two organizations failed to identify any substantive activities that would make sense to pursue together. This frustrated some, but from BPML.org's perspective it made perfect sense, for we were pursuing a different methodology, one based on processes, not objects. Now Frankel,

by publishing his article, has once again suggested joint OMG-BPMI work. But things may not be so straightforward, despite the clarity of Frankel's argument.

My coauthor, Peter Fingar and I also thank David for his "*landmark*" designation of our book, *Business Process Management: The Third Wave*, [2] a book that sets out to explain the BPM breakthrough and why its needed.

## Competitors, Alternatives, or Complementary

*Frankel: "BPM and MDA, while not identical, have a number of characteristics in common."*

This is only superficially the case. Although MDA and BPM are both architectures for enterprise software systems the comparison stops there. As David states in his article, MDA is designed to enable the computer-assisted generation of applications and components, i.e. software. By contrast, the purpose of BPM is to avoid writing, or generating, more software, its central thesis being to remove "process" from software as completely as possible, creating a new form of "data" that can be manipulated on a scale previously unimaginable. A similar business "digitization" occurred when relational database management systems (RDBMS) were introduced.

*Frankel: "BPMS is a software tool." There are other references to "BPM tool" throughout the article.*

I prefer to say that the BPMS is a *platform* with characteristics similar to RDBMS. Where the RDBMS supports "applications" which represent (with their data) functional processes, the BPMS represents the whole process, and supports numerous software tools, covering *process discovery, design, deployment, execution, interaction, operations, optimization and analysis*. Business people manage the lifecycle of processes that reside in the BPMS using these tools. These process lifecycle tools are the first applications of the BPMS. Many more shall follow.

This simplification of data and code to a single paradigm, *process*, is hard to imagine without an example. Everyone understands that to add a new item to the head of a linked list you need to write a piece of code to link the back and forward pointers. In fact, using *process technologies*, it is possible to model the head and the tail of the list as *participants* in a process, communicating with each other so as to grow the list. In BPM, there is no distinction between data and application. There is only process. Process, as defined by BPMI.org, is not the usual definition of workflow, a series of steps. We use the term broadly.

BPML is based on formal theories of process calculi, notably the pi calculus.[3] The formalism is expressive enough to represent all discrete processes. Using BPML it is possible to build complete "applications," but such processes have very different properties to existing software systems, such as programs written in Java. I touch on some of these in this article, but refer readers to the book, *Business Process Management: The Third Wave*, for the full story. For more technical background than the book provides, I urge you to schedule a briefing with BPMI.org. As a start, see the BPML presentation recently presented at US Process World.[4]

*Frankel: "Some [BPM] tools actually execute business processes using technology that is evolving from workflow and EAI"*

While numerous vendors have recognized the importance of putting "process" at the heart of architecture, it is not the case that all existing technologies can necessarily evolve towards BPM. For example, a BPM Solution that uses a workflow engine to drive processes (whether system to system, human to human or system to human) is constrained by the workflow engine's ingrained workflow semantics and therefore limited to those processes that fall into the class of *workflow processes*. Combining EAI, B2B, Rules and BAM functionality into a BPM Solution, but which is driven by workflow semantics, is not the same as supporting BPMI.org's process semantics.

### ***Workflow is a subset of BPMI***

Workflow management systems embed a fixed model of one type of process, namely workflow processes. The participants in such a process are users, tasks, task lists, job schedulers and other elements of the workflow model.[5] All such elements, including all control and data flows of such elements, can be expressed in BPML. In other words, BPML can be used to model the behavior of these participants, forming an end-to-end model of workflow. Such a model is not the code of a workflow engine, but a schema for a BPMS that enables the creation of instances of workflow processes.

Each participant would be represented in a *swim lane*—formally, a computing thread. Within a swim lane BPML would be used to model the control flow as a participant. Across swim lanes BPML would be used to model the data flow between participants. Such a model could then be used as a participant in another end-to-end process that requires "workflow" functionality. Alternatively, the same process design could be used to create instances of workflow processes in response to events, such as a new workflow "case."

In BPM, what workflow calls a "case" *is* the process. Such processes are persistent; end-to-end, across all participants. A typically challenging application for such a model is the end-to-end management of health care records, not as documents processed by standard workflows and maintained in a document management system, but as processes in their own right. Such *healthcare processes* would extend throughout the health care system and would evolve in line with the unique medical history of their owner; herself modeled as a participant in the processes that form the entire medical ecosystem (doctors, health authorities, health systems, hospitals etc.).

### ***EAI is a subset of BPML***

A BPMS is very different to an EAI broker, in both architecture and in purpose. EAI grew up in response to the lack of integration between stovepipe technologies. BPM grew up to represent and manage business processes and, if required, to include existing application components within new processes.

Many EAI tools only use the process concept to integrate applications. We could call them *application-integration processes*. Not only are such processes a subset of all possible processes, EAI achieves integration in a way that is different from BPM. EAI focuses upon interfaces to applications. BPM views the same applications

in two ways, either as participants in a process, or as reusable processes. Whilst applications ingrain processes in software, a BPMS can nevertheless expose, or project, them into BPML. This allows them to be re-purposed in an end-to-end process model.

BPML can model the behavior of many other kinds of “integrative processes,” coordinating both tasks and other processes. These may have nothing to do with “applications” for which EAI was designed to integrate. BPM has a wider applicability than EAI, including:

- Automational, eliminating human labor from a process
- Informational, capturing process information for purposes of understanding
- Sequential, changing process sequence, or enabling parallelism
- Tracking, closely monitoring process status and participants
- Analytical, improving analysis of information and decision making
- Geographical, coordinating processes across distances
- Intellectual, capturing and distributing intellectual assets
- Disintermediating, eliminating intermediaries from a process
- Computational, performing distributed calculations as part of a process
- Collaborative, allowing participants to manage sets of shared work processes
- Conversational, allowing participants to exchange information among themselves
- Combinations of all above

#### ***A BPMS can morph to any process semantic***

BPMS is a complete environment for all possible process models, just as an RDBMS is a complete environment for all possible data models. A BPML model is a schema for deployment on a BPMS. A BPML model can be deployed directly on the BPMS without intervening steps. BPML is not “software.” Neither is it a design or model for translation into software. A BPML model, together with its BPMN notation, is just like a data model and its ERD, except that it’s about process instead of data.

For example, a BPML model including three participants, sender, receiver and address book, can “configure” the BPMS to act like an email server. Using this simple schema the BPMS can generate instances of email conversations that evolve over time, from short-lived conversations among a few individuals to arbitrarily complex and extended “conversations” (processes) between many individuals.

BPML can describe *all* existing software, components, services and systems so that they can participate in new processes. In BPM, even numerical computations are considered *processes*. Milner showed us how to unify computing (what happens inside a computer) with communication (what happens between computing systems). This unification can be applied at all levels, from the interactions between complete computer systems, to the interactions between different applications to the computational threads in a single application—even to the computing paths between components in the hardware, down to the level of the chip.



*Frankel: "Standards for process modeling are key to attaining BPM's goals, because organizations will increasingly compose value chains by integrating their processes"*

I view standards somewhat differently as those that are developing specific B2B process models. BPML is pursuing a standard way to represent *all* processes, not a standard process. BPML was created to bring standards-based BPMS to market as a new category of business infrastructure, whilst simultaneously leveraging existing processes fully. BPML views BPML as playing the same role as relational algebra does in helping businesses structure data for data management. Business processes can be combined using BPML precisely because of its formalism and support for end-to-end processes. Just like BPML can model workflow, EAI, healthcare records, email collaboration or linked lists, BPML can model B2B protocols. Using BPMS *businesses themselves* establish standards for B2B collaboration and create a *management platform* for the management of processes created as part of their collaboration.

*Frankel: "It is important to bear in mind that BPM is not primarily concerned with software application development"*

Actually, BPM is primarily concerned with *eliminating* the need for software application development by providing a foundation for direct representation, manipulation and execution of the abstract data type, the "business process."

While it's true today that BPML is not primarily concerned with software application development, this too may change. Initially, BPML will not be used for all software development tasks, but there are few theoretical limits as to what could be achieved in the future. The BPMS will find wider application as the performance of process technologies increases. The modeling of end-to-end processes often uncovers the need for computation and data transformation across participants. In effect, models are distributed software systems. BPMS is therefore a software application used to manage processes that may themselves resemble the complexity of "software." But it is vital to distinguish the difference between software code (based on the Lambda Calculus and to which communication has to be added as an after-thought) and process schemas (based on the pi calculus which unifies computing and communication).

The conceptual design of BPM rests on a separation of a design model (BPML) and an execution model (BPMS). This inherently implies a trade off in terms of functionality, since only those processes that can be represented in BPML can be executed. The trade off was made so as to provide very strong process management capabilities. Management is applied to the holistic BPML models that extend over the software processes that may participate. This is management by explicit intentions. Businesses consolidated data into RDBMS in order to enjoy the benefits of data management. I believe that businesses will consolidate processes into BPMS in order to enjoy the mathematically guaranteed benefits of process management.

This trade off, the separation of the design model, is similar to tradeoffs made in other areas of software engineering, such as the relational model or the simple rows and columns of a spreadsheet. I believe the trade off is appropriate, a step change in business capability, and will become the preferred platform for management of



business assets. Processes, that include their data, are the evolution of what today's workflow and document management tools refer to as "content."

Processes are a new content class. Process semantics are equally at home encoding ad-hoc collaboration (e.g. email), to semi-structured collaboration (e.g. workflow, document management, voting, commitments) to highly structured transactional "straight through" systems. Loosely coupled or tightly coupled, process semantics are equally at home representing conversational or computational processes. Looked at in this light, process technologies will be the basis for the next generation of enterprise "knowledge" management. Many future "applications" will be built on a BPMS.

*Frankel: In discussing MDA he says "each rise in the level of abstraction increased developer productivity, improved software quality, and increased software longevity"*

The purpose of BPM is not to make it easier to create more software. Software is created through a "waterfall" (or refinement, iterative) model of development, translating between levels from requirements to implementation. The purpose of MDA is to assist the translation and migration of these higher-level models to lower level code and different execution targets. The purpose of BPM is the opposite. Because BPM is a Design-Driven-Architecture (DDA), not a Model-Driven-Architecture (MDA), BPML models are directly "executable" without translation. Although some are proposing "executable UML" there are fundamental differences from BPM and fundamental differences between a running software system and a running process.

It is my view that the IT industry has been too obsessed by finding new ways to rapidly define more and more software, itself creating a "management" problem. Rather, BPM aims to redress the balance. The objective of BPM is to find formalisms that place "business processes" into the realm of understanding of every day business people, as spreadsheets did for numerical processing, or databases did for data collection and analysis.

In MDA, and software engineering generally, layers of abstraction are used to enable translation of software from one form to another. The cost of this to business is well known. Indeed, Frankel talks about these "model compilers." In BPM, while it is possible to structure and layer a process model so that it can be understood, there is no step required between process design and deployment. A BPMS adjusts to a process model immediately, just as a RDBMS adjusts to a data model.

And processes are not set in stone as software, but are able to adapt, grow, shrink, combine, split, merge and morph—an infinity of *mobile behaviors* as existing participants communicate and new participants are found through communication. An analogy is the spread of a rumor at a cocktail party. Processes are simply not like software, period, but they can play the same role as software in support of business processes.

*Frankel: "The increased overall complexity [of the software infrastructure] strains development organizations and is spurring the next rise in the abstraction level."*

While it is true that the IT software industry is keen to simplify the complexity it has created, BPM takes a different perspective. Instead of adding more software layers

and software abstractions to an already complex environment, I say, eradicate the complexity and try a new model, the BPMS. While one can expect the software industry to continue to innovate to create higher-level abstractions, e.g. MDA, I expect BPM to offer a different path that offers its own advantages. The cat is out of the bag. BPML, building on the shoulders of giants like pi calculus, has found a way to represent software as malleable schema. It is this that creates the opportunity for the BPMS, a design-driven, not model-driven approach.

In BPM there are no differences of representation at any level. The very same type of model that a CEO can understand uses the same semantics at lower levels to express detailed computation or communication. The same semantic building blocks, or BPML elements, are available at any level of the model.

BPM, for the class of problem for which it was designed, takes the layering and translation *out of* the solution, by formally separating the process from the process virtual machine, the BPMS. In effect, BPM moves software development complexities *outside* of the process design effort. This gives, to *every* BPM application, inherent qualities of process management and lifecycle control that have to be otherwise *built in manually* in other software development paradigms.

Using a BPMS, “programmers” and “business analysts” never have to worry about preserving the past state, evolution track, present state or future potential of *any* process, because this is the responsibility of the BPMS. Nothing is pre-committed or locked into a design. All processes, and every instance of a process, are free to evolve and adapt as they are used throughout their lifecycles, just as data evolves within a RDBMS today. This is a very different style of reuse from that proposed for object-based systems, where the method is encapsulation of functionality for potential future reuse. By contrast, simply using a BPMS *inherently* creates reusable assets in the same way that the use of a RDBMS over time naturally creates reusable data.

*Frankel: “MDA ... pushes the software production process in the direction of emulating automated industrial production processes.”*

This may be true, but it is also true of BPM, and perhaps more so. CAD/CAM is another example of a design-driven architecture. The product design models correspond to the BPML processes in BPM. The CAD/CAM tools correspond to the BPMS.

CAD/CAM was successful in support of Computer Integrated Manufacturing, yielding huge productivity gains for manufacturers, precisely because a design-model was developed for Product Design Data which was separated out from Product Design Tools, allowing the designer to work with an appropriate level of abstraction for the task and allowing models of products to be placed directly into manufacture creating Computer-Integrated Manufacturing (CIM). BPM aims to do the same for business processes. Analogous to CAD/CAM, BPM is sharply focused on providing businesses the tools they need for process management, not just a better way of producing more software.[6] BPM will eventually spawn Process Manufacturing Environments.

*Frankel: "BPM uses models to automate the management of business processes ... MDA uses models to describe and automate the development and integration of software ..."*

The models discussed are very different. MDA is a model of software. BPM is a model of process. MDA's model of software is based on relationships and objects. BPM's model of process is based on pi calculus, a unification of computing and communication. Frankel says that, "the key distinction alone has to do with the role played by the process model." Not true. The models are different in character and purpose.

*Frankel: "Both BPM and MDA will require time to penetrate industry. A company can adopt neither overnight."*

BPM can be applied immediately following installation of a BPMS just as a spreadsheet, a database or a CAD/CAM tool.

Although it took 10 years for relational database management systems to become pervasive, those companies that had severe data management problems successfully deployed and used early RDBMS products. While using early RDBMS products may not have been as easy or straightforward as today's sophisticated products, the value of data management was evident even in the first attempts. The same is true of BPMS today.

*Frankel: "BPM models describe business processes"*

In comparing MDA and BPM, Frankel refers to business processes and software production processes. In fact, software production is a business process and could be modeled using BPM and managed using a BPMS. One computer services firm is already doing this. The potential advantages to the computer services firms that master this are huge, as occurred in CAD/CAM/CIM.

While a model of the software development process is very different from a model of software, we should not lose sight of the fact that BPML *is* a programming language, and therefore a form of "software." It is just a very different form of software, as different from Java as Java is from COBOL, or more so. BPML is the first industrial strength concurrent language that unifies computing and communication—formally. For example, two BPML processes can be "integrated" not through the need to employ an EAI solution, but because BPML models inherently "communicate" and can be combined to form end-to-end process. Thus, BPML is a form of software engineering in its own right, with its own characteristics. There is nothing to stop proponents of layered compilers applying these techniques to the production of BPML as they do to Java. However, with BPM, the BPMS takes on some of the responsibilities Frankel puts forward for MDA, as RDBMS does for data management.

BPML is not concerned only with creating architectures independent of technology, but of bringing a new BPMS capability to the business. Layering a design-driven architecture (BPM) is not the same as layering a model-driven architecture (MDA), just as layering a data model is not the same as layering a Java program. These differences in technology architecture lie at the heart of why it may not be so simple

to apply MDA to BPM. In fact, it could be easier to create an MDA IDE using a BPMS than visa-versa.

*Frankel: "MDA's move up the abstraction ladder carries the development paradigm inexorably away from the computing paradigm and toward business processes."*

This statement could perpetuate a common view that BPM is some form of upper level abstraction or layer above existing technologies and applications and middleware. While this is convenient for those vendors who wish to add BPM as a thin layer above existing software, it is far from a complete approach to BPM.

While a BPMS may be mounted along side existing applications, so that ingrained processes in applications can be projected and reused, the BPMS exposes a new level playing field, the naked vista of process management. In BPM, models may also be layered, from top to bottom, hiding detail at one level and offering it at another. BPM imposes no standard set of layers or abstractions. BPM practitioners will develop these over time, as occurred in software engineering.

Where today some vendors position BPM above applications, for example orchestrating services, the BPMS should in fact be placed beneath, or in support, of the architecture.<sup>[7]</sup>

*Frankel: "A key issue is how to treat the remaining gap—which some have called the abstraction gap—between the business process specification language and the software specification language."*

This comment could perpetuate the view that BPM is some how a layer "above" other mechanisms. Indeed, some "BPM" vendors translate process models into software for execution, completely missing the advantages of the BPM's pi calculus. These are not BPM solutions at all, but are in fact software code generators. Nor is a BPMS only a binding between a high level abstraction and an execution environment, for example, between a process modeling tool and an EAI broker, or between a process modeling tool and a workflow engine.

While acknowledging that BPM is different, Frankel claims that, "IT does have to be involved at times to hook sub-processes to applications and components." While this might be true for some BPM solutions, taking into account their level of maturity, the situation is likely to change rapidly. Zero code BPM is pretty much here today. But this does not mean that "IT" is out of the loop, far from it, for BPML and its pi calculus foundation creates, for the first time, a common language amenable to all specialists in the business. Don't be surprised if business people are not as "intelligent" as software engineers and, once presented with a familiar language (to them) they are not just as adept at changing, and managing, lower level models if they feel their business objectives depend upon it. Similarly, don't be surprised if "IT" skills are needed to develop sophisticated business process schema models. BPM creates the common language between business and IT that can then amplify each of our skills, across disciplines. That is the power of a design-driven approach, putting tools in the hands of those that best know how to use them for business benefit.

*Frankel: "How we derive a software specification from a BPM specification falls into the category of black art" and comments such as "BPM tools could generate a skeletal software model ..."*

I don't want to turn BPM models into software any more than software engineers want to turn software models into cheese. BPM models are *directly* deployable, at all levels, and the BPMS is assumed to exist for this purpose. The possible OMG view that BPM models themselves must be mapped to multiple target environments is not a necessary objective of the BPML. BPML and BPMS, go hand in hand. A BPM model is complete, and not a skeletal model. The resolution between the BPM model and its potential reuse of existing software is a function of the BPMS.

*Frankel: "At a minimum, tools could maintain trace relationships from the software models and generated code modules back to the business process ..."*

Once again, I stress that there is no generated code, not one line! It's all about taking a BPM model and directly "executing" (or what we call *proceeding*) it on a BPMS. Some may view this as being no different to the immediate execution of a Java program on a Java virtual machine. While it is true that BPML is immediately executed on a process virtual machine the execution of a process is utterly different from the execution of software. Java programs don't unify computing and communication and therefore don't enjoy mobile behavior, so critical to the representation of business processes.

Thus, there are no such complex relationships to manage. Nevertheless, if one were developing a process that did reuse existing *software* code, and that code was subject to a separate development lifecycle to the end-to-end process model, then facilities to manage the configuration of model elements would certainly be helpful. Therefore, I'd urge vendors of BPMS to study the MDA concepts.

*Frankel: In Table 1, he seems to equate BPM with IBM's Holosofx, Microsoft's Jupiter and Oracle workflow.*

The "M" in BPM stands for management, not modeling. IBM's process modeling and mapping tool, acquired from Holosofx, is not a BPMS. Perhaps not in Frankel's case, but it is a common misunderstanding to confuse process-modeling tools and process management systems. Process modeling tools are applications of a BPMS, supporting parts of the process lifecycle. For example, one tool may be used to create a process, another to visualize it, and a third to analyze it.

IBM needs to bring together a complex stack of technologies to create a BPMS and we urge them to do so. The danger for IBM is if they define BPM as BPM-driven EAI or EAI-extended workflow. Both miss the BPMS opportunity for semantic unification. Microsoft's Jupiter may be closer to what BPML.org defines as a BPMS, although details are sketchy. BPML.org co-chair, Intalio, has implemented a sophisticated product that many regard as the reference architecture for a BPMS in the industry today. Oracle workflow is a workflow product, with quite different semantics to BPM. Most of the gorilla vendors will eventually implement a BPMS, as they did a RDBMS.

*Frankel: Misconception #1: UML forces you to use object-oriented modeling.*

The benefits of object-orientation are well known in supporting the composition of software components. Frankel here appears to be distancing himself from objects, the “O” in OMG, for reasons I don’t understand. While Frankel states “UML allows a modeler to stray far from OO when required,” I wonder what formalism the modeler would then be using, if the model can be adapted in any direction. It is true that when using UML designers have to corrupt the model to better represent processes. I say UML is just fine for software engineering, but is a distraction for process engineering. BPM, not UML, provides the required formalism, the process calculi.

Object practitioners respond to BPM by asking how the advantages of object-oriented systems can be realized in the world of processes? What we face here are two entirely different worldviews, one built on objects as “first-class citizen” and one built on processes as “first-class citizen.” The two do not sit together comfortably, and from a computer science perspective, we are only now learning how to formally represent “objects,” in the world of processes. In fact, the mathematics of process systems appears to have advanced further than formal systems applied to OO, even though process technologies are relatively new and object technologies are well established.

To look into this matter deeply is beyond the scope of this article. Take one claim for objects, “reuse.” While reuse is achieved using quite different methods in the world of processes (sub-process, idiom, pattern, participant, service-as-process, process-as-service), some developers report it to be more “natural.” While having little in common with the concepts of generalization and specialization of class methods, process reuse achieves similar goals.

Perverting an object-based formalism (UML) to superficially look “more like” processes (UML 2) will lead to confusion. I urge more to be published on the BPML process model and to compare and contrast it with the object model. The development community is in the earliest days of understanding the characteristics of process-based systems. We urge not the unification of process and object, but to understand their unique advantages and limitations, through practical use.

## BPM and MOF

The remainder of Frankel’s article is an exposition of the advantages of the MOF (the Meta Object Facility), a sister specification to UML and a component of MDA, as it relates to OMG’s vision of platform independent computing and abstraction. At this point Frankel’s article becomes highly technical, and indeed, MDA and the MOF are complex. They are all about software metadata, and the use of such metadata to define, annotate, map and manage models in a variety of languages for the purposes of creating a platform independent, and integrated, development environment. Worthy aims. The objective, as always it appears with the OMG, is to find a way to unify different technologies. OMG achieved great success when they unified competing OO modeling languages to one, UML. Their fervor with unification continues to this day. I suspect that some at the OMG believes the object-process universe truly can be unified, as physicists believe in a Grand Unified Theory of Everything. However, as in all searches for the holy grail, different folks take different paths.

BPM's path took it into the world of pi calculus, where previously different things (computing and communication) were found to be the same thing—processes. This was not so much a unification, more a scientific finding or discovery, and not made by BPMI.org, but smartly applied by BPMI.org. OMG's path is based on mappings between different representation and execution systems.

It appears to me that OMG and BPMI are pursuing different paths and therefore any suggestion for application of MDA to BPM, or visa-versa, must be examined very carefully. Frankel makes explicit recommendations. Because they are controversial Frankel goes to great lengths to be politically correct at this point in his article, stating:

*“One could interpret the adoption of an XMI-compliant schema for BPM to be an expansion of BPM into MDA's realm or an expansion of MDA into BPM's realm. However, this is not an important issue. The real issue is facilitating BPM-MDA coordination.”*

Coordination is certainly better than unification, for I don't believe the two architectures can be unified; they are just different.

*Frankel: “UML 2.0 activity modeling notation and the latest draft of BPMI.org's BPMN are conspicuously similar”*

True, because in some cases there will be no need to be different. But there are other cases where software notation and process notation may need to be very different. In fact, despite the desire of some BPMI members to unify notations, this may not be possible. Most process modeling tool vendors today support both process and software modeling in their tools. One notation would certainly be convenient. But experience with BPMS and therefore of “process” is limited, for BPM is in its infancy. As we learn more by integrating modeling tools with a BPMS, we will quickly realize that what works for software models may not work for process models. Take just two examples:

- A process model can be viewed in several ways, for example, as communicating participants or as a time based Gantt chart. It could also be viewed as a set of relationships, established among participants, through the dynamic exchange of information.
- Can software, UML-type, diagrams show the dynamic behavior of mobile processes, or the rich interactions between process participants over time?

I suspect that a myriad of notations will be used to describe processes, as befitting the nature of the process being modeled. Do not let superficial similarities in the shapes of boxes and lines hide fundamental semantic differences. There is more work to do to publish BPMI.org's semantic model, stripped of its XML baggage.

Frankel makes explicit recommendations to BPMI.org. He points out that the MOF-XML mapping is named XML Metadata Interchange (XMI) and the MOF-Java mapping is named Java Metadata Interface (JMI) and that MOF-CORBA has also been mapped. He urges BPMI.org and those developing BPEL to apply a similar technique, to

create “MOF-BPML,” to create XMI-compliant schemas that might presumably be named “BPML Metadata Interchange (BMI),” or some such. He correctly points out that in doing so none of the BPML semantics would be lost. He says that the effect would be only a different XML-wrapping which would, in his view, be “a slight adjustment of the BPM XML schemas to comply with MOF,” enabling organizations to create the integrated management of BPM and other MDA models. The problem is this: BPM is not only BPML.

## BPML.org Decisions From The Outset

At the outset of BPML.org, back to the very first meetings in 2000, even to the founding conversations at the end of 1999, BPML.org took the decision to separate two aspects of architecture, BPML and the BPMS (containing the process virtual machine), thereby creating the design-driven architecture (DDA) I spoke of before. BPML and BPQL assume the existence of a BPMS just as relational algebra and SQL assumes the existence of RDBMS.

When the relational data model was created, no one suggested unifying it with other methods of data management at the time. Nor did one suggest meta-tagging its structure so as to be more in line with existing methods of data description. We should not rush into conclusions about how MDA and BPM fit together. Other aspects should be borne in mind when deciding upon these matters:

- It is not an intention for every software company on the planet to implement a BPMS. Indeed, I suspect only a very few would have the required resources. Rather, a few vendors (beginning with start ups) will bring a BPMS to market, to be followed by gorillas emulating the innovation. Afterwards, many others will build new applications upon it and many existing technologies, both tools and applications, will be integrated to it.
- To date, BPML has not focused on interoperability of BPMS systems or the standardization of a reference BPMS architecture. BPML has not evolved existing technologies or mapped one system to another, or one language to another. BPML was created to bring *new* technology to market. In doing so, different vendors will take different approaches, primarily *inside* the BPMS, not in the *skin*, BPML and BPQL.
- BPML was wary of mistakes in the past. Workflow technologies were the innovation that heralded BPM, but lack of rigor in platform definition and incomplete workflow semantics led to numerous (hundreds) vendors developing many different workflow engines each adding semantic features in order to compete. The focus of standards development in the field of workflow therefore *had* to be upon interoperability of workflow technologies rather than the creation of a standards-based workflow environment. For these reasons workflow remains to this day an add-on to application architecture, and not a foundation technology. Yet because it too is a design-driven architecture, its significance has increased recently, driven by the new found interest in BPM among customers and their frustration with the complexity of application software architecture.

## Conclusion

BPM and MDA are different. While MDA could be used to increase the portability of the BPMS itself or the reuse of process model these could be a decisions taken by individual vendors. While vendors can easily agree on BPML and the coming BPQL, they may choose to pursue very different approaches to a BPMS implementation. This competition is critical at this stage of market development.

MDA builds on years of experience with object-based systems. BPM is new, and like all new things readers have few anchor points with which to understand the innovation. Understanding will come only through experience. Readers familiar with existing software engineering methods will relate to Frankel's argument for MDA. Software engineering has, in fact, become too complex, and end users struggle with this across the world. If MDA can help resolve this, all to the good! However, BPM is partly *a response to that complexity*. Let's not add complexity to it.

Like spreadsheets and databases took numerical processing and data management off the critical path of the software technicians and placed those capabilities firmly in the hands of business people as tools, so too will BPM for business process management. As the BPM technologies mature, managers will find that many things for which they were previously dependent on software engineers will fall within their own grasp. MDA is for software engineers, making their job easier. BPM is for business people, making their "process work" easier. Business people are generally not interested in making more software, even if MDA is there to help them, but they are interested in creating, and managing, business processes. For this they don't need a sophisticated new approach to software development, they need a design-driven architecture based on processes and a BPMS able to immediately and easily act as the lens through which the existing world of computing, and communication, are seen as one, in the form of processes.

Of course, the world of software development and process management are intertwined, for all software supports parts of business processes, and all processes are likely to include both "computing" and "communications," the elements of software. But rather than unifying these *two software aspects* to one, why not let each find its way in the world? MDA builds on the complex universe of software development practice and this extends and enriches what has become a preoccupation of the IT industry. BPM has less "heritage" (or legacy depending upon your viewpoint) to build upon, but its potential is large for the future.

Not that Frankel suggests this, but it is too early to standardize *both* the design-driven (BPML+BPMS), and model-driven (BPMS) parts of the BPM architecture. Now is not the time to "unify" the BPM space. To do so will likely stifle innovation and limit the BPM marketplace. Let's celebrate diversity, rather than homogeneity.

Let's have some fun: While the OMG may be tempted to rename itself the PMG, or the UMG, I personally struggle to see what unification for unification's sake will bring to business. Nevertheless, there remains the need for the developers of MDA, and the developers of BPM, to continue to talk, and for this we very much welcome Frankel's article. Together with this response they make excellent starting points for discussion.

Lastly, I am sure that David would want me to clarify that where I frame a response to him, the reader should make no assumption that he is claiming the contrary. However, even he must admit that awareness of the true nature of BPM is limited and that some points need to be made forcibly.

## Footnotes

---

[1] Frankel, David, S., "BPM and MDA: The Rise of Model-Driven Enterprise Systems," White paper published at [www.BPTrends.com](http://www.BPTrends.com), June 2003.

[2] Howard Smith and Peter Finger. *Business Process Management: The Third Wave*. (M K, 2003).

[3] For more information on the Pi-Calculus check: [www.fairdene.com/picalculus](http://www.fairdene.com/picalculus) or see Robin Milner, *Communicating and Mobile Systems* (Cambridge Uni Press, 2001).

[4] Business Process Modeling Language, [www.bpm3.com/presentations](http://www.bpm3.com/presentations).

[5] See [www.wfmc.org](http://www.wfmc.org)

[6] Pages 121, 122, 153 and 235 in *Business Process Management: The Third Wave*

[7] Pages 254 and 255 in *Business Process Management: The Third Wave*

## Howard Smith

Howard Smith is CTO of Computer Sciences Corporation (CSC) Europe, co-chair BPMI.org and co-author with Peter Finger of *Business Process Management: The Third Wave* ([www.bpm3.com](http://www.bpm3.com)).

