1 Executive Overview – The Benefits and Objectives of BPDM

This is an excerpt from the "Final Submission" BPDM document posted to OMG members on November 13th 2006. The full version of the specification will be made available following the December 2006 meeting of the OMG.

The "Business Process Definition Metamodel" (BPDM) provides the capability to represent business process models independently of the modeling notation. At the same time, it also provides a robust serialization mechanism for the BPMN modeling notation.

BPDM sets out to define a shared vocabulary for process modeling concepts; think of it as a sort of "universal syntax" of process. The central idea is that BPDM is capable of supporting most common types of process model and, as much as is possible, enables the robust exchange of models while preserving the intended enactment and execution semantics.

To achieve this goal, the heart of BPDM supports two fundamentally complementary views of process – "Orchestration" and "Choreography":

- Orchestration concepts in BPDM are represented through *"Process,"* which includes the traditional view where sequences of *"Activity"* are carried out, with branching and synchronization of different threads (sometimes known as managed or directed execution).[1] [2]
- Choreography is a more abstract notion of process (for most people). It describes the *"Interactions"* of collaborating entities, each of which may have their own internal orchestration processes. These Interactions are often structured into *"Interaction Protocols"* (in the sense that Interactions usually have an order) to represent the conversation between the parties. These protocols usually exist between both internal organizational roles and external stakeholders such as other departments, business units, as well as customers, suppliers and regulatory authorities.

In business process modeling, choreography and orchestration are effectively two sides of the same coin. In BPDM the "Common Behavior Model," allows them to be treated independently, yet enables the sharing of common information (about business being modeled).

Target Audience and Use of BPDM

At its core, BPDM provides interoperability across tools, so that different tools can depict a process in different ways. This also provides portability between tools. If Vendor A and Vendor B both support BPDM as their process serialization (storage format) mechanism, then, a BPMN drawing created using Vendor A's modeling tool could then be opened and executed using Vendor B's business process management system. Therefore, BPDM is a technology specification for vendors to use to define how they serialize their process depictions, allowing for industry interoperability. For most business analysts and process users, this is all they really need to know about BPDM.

[1] When BPDM metamodel classes are initially introduced in this section of the specification, they are shown using *"Capitalized Words"* in double quotes. Thereafter Initial Capital(s) refer to either an instance or collection of instances of the class. Where relevant, plurals are used. Where lower case is used, it refers to the general concept that shares a similar name (rather than a BPDM class). 'Single quotes' are used where emphasis is needed on a particular phrase (i.e. outside of the context of the specification).

[2] It is worth understanding the challenges that were faced by the authors of the BPDM specification in coming up with 'appropriate' names for the entities in BPDM. Nearly all terms in this domain have special meanings in other process modeling languages, leading to the potential for misinterpretation by the reader.

Other Common Business Benefits of BPDM

BPMN metamodel: First and foremost, BPDM provides an explicit metamodel and serialization mechanism for BPMN. Indeed, 'minimum compliance' in providing support for BPDM requires support for the subset of the elements of BPDM that provide this BPMN support.

A pre-packaged set of model elements provide BPMN support – i.e. the core elements of the Common Behavior Model covering Events, Interactions and Process Steps. Vendors looking to support BPMN can use those elements and effectively ignore the rest of BPDM. Support for the BPMN Package represents the most rudimentary compliance level for BPDM. The semantic content of a BPMN process model is managed separately from its graphical markup (enabling greater clarity).

Separation of Concerns: For most business analysts, orchestration and choreography are all they really need to know about BPDM. From the point of view of a business analyst who is trying to understand a business domain, the separation of Interactions (choreography) from the Activities (orchestration) enables a separation of concerns. This helps them to more easily model and communicate what is going on (since in many modeling techniques the interaction and activity models are conflated). Defining the pattern of Interactions between the roles describes the intentions of the parties, without necessarily committing to how they deliver against those intentions. This creates tremendous flexibility. Modeling Interactions defines the responsibilities of a role or process, effectively underpinning the definition of service level agreements—it defines a set of constraints on the private (internal) orchestration process. It also allows the business analyst to defer decisions about how an orchestration process is going to actually work, and its technical implementation detail.[3]

Compliance: Of course, when modeling an orchestration process, it is common to assign responsibility for undertaking actions to organizational entities (roles). The hand-offs that result from this sort of assignment represent the pattern of interaction—the protocols between the roles. At the level of the role, it then becomes possible to robustly describe the commitments it makes to other roles involved in the process, separating its own 'internal' implementation from those commitments. In much the same way, given an existing orchestration process, it is possible to derive the boundary conditions for that entire process—i.e. its interactions and message exchange patterns with other roles or processes. As a result, BPDM facilitates automatic compliance assessment of a process against the internal (private) process.

Business Encapsulation: At the business level, support for both orchestration and choreography is becoming increasingly important to modern organizations. Business boundaries change ever more quickly as firms reorganize, merge and split their operations. The results are changes in both the internal boundaries (between departments and business units), and those elements that are owned or outsourced to both collaborators (in the supply chain) and the customer. Yet there is still a common process operating across the value chain.

BPDM facilitates this evolving, permeable boundary by identifying and managing the common elements, enabling the modeler to more easily communicate what is going on and transition between these views (without having to continually refactor the entire model). Once the commitments of each role are defined (its boundary conditions), it becomes possible to change the realization of the process without changing the commitments.

[3] Most other models and file formats for choreography and orchestration were defined as separate languages, sometimes at different standards bodies, and consequently lack the cohesion necessary to smoothly transition from one to the other, as described in the next section.

Thus, at the business level, BPDM enables a better 'service orientation,' where processes are composed of discrete, reusable business services as needed. This enables better alignment with Service Oriented Architecture and encapsulated services in the IT world. Moreover, multiple (internal) entities can then collaborate on a common goal, without having to commit to, or know about, each other's internal processes (a big issue for large enterprises).

Granular Contract Definition: Such models could be created from scratch (as protocol definitions), or created through the composition of existing protocol fragments (from a library). The resulting model can then provide the terms of reference for each participants' own internal orchestration model. In turn, these orchestration models could be teased apart to explore the business protocols that exist between the internal participants (internal roles). It would then be possible for one participant to redesign its internal process, realizing the roles independently of each other, and evolving how an individual role operates, independently from the other resources deployed by the organization to support its commitments. This sort of functionality is particularly useful to the Business Process Outsourcing (BPO) community as it allows an organization to outsource parts of a process without triggering a complete re-factoring of the work.

The robust models that result from this sort of analysis might automatically generate BPEL execution models to support the agreed choreography. In much the same way, BPDM therefore facilitates multiple BPMS engines interacting (via messages). The engine itself could then automatically detect situations where constraints are broken (and raise appropriate alerts to business owners).

The "Common Behavior Model" and its Abstractions

It is important to realize that these fundamentally different perspectives (views) of a process are based on distinct, yet related sets of semantic information. In order to support the sorts of benefits described in the previous section, BPDM must handle each perspective separately yet share information in a robust fashion. The discussion thus far has focused on two distinct perspectives of business processes – Orchestration and Choreography. The Activity oriented view is what the business process 'does'; the Interaction (protocol) view is the definition of the 'commitments' made by the parties.

The "Common Behavior Model" provides the structure to integrate these distinct notions of process, through which different views can share information. The primary purpose of the Common Behavior Model is to bring together the common elements of choreography and orchestration. Without such a fusion, users and vendors would have more than one way of expressing the same thing, leading to confusion and creating inefficiencies when managing models at execution time.

One aspect of the Common Behavior Model is an event-oriented view of processes called *"Happening"*, covering all those sorts of elements that involve time—either at a single point or over an extended duration. The second aspect is *"Processing Behavior"*, which leverages Happenings to represent sequences of generic steps (applicable to both orchestration and choreography). The third aspect "Simple Interaction" addresses the definition and reuse of protocols. It is applicable to both choreography, and to the interaction of an orchestrated process with its environment. These three aspects of the Common Behavior Model support specification of the dependencies between Activities (in orchestration), between Interactions of participants (in a choreography), and between the Interaction of an orchestrated Process with its environment (a combination of both orchestration and choreography).

On top of the Common Behavior Model are two levels of abstraction – the "Course Model" and the "Composition Model". Think of them as layers of functionality that ensure consistency and enabling validation mechanisms that guarantees transformations occur accurately. The key point is that these two abstractions are necessary when either traversing from one modeling notation to another, when supporting the translation to run time execution, or when extending the environment into other domains. BPDM also leverages the Infrastructure Library of UML.

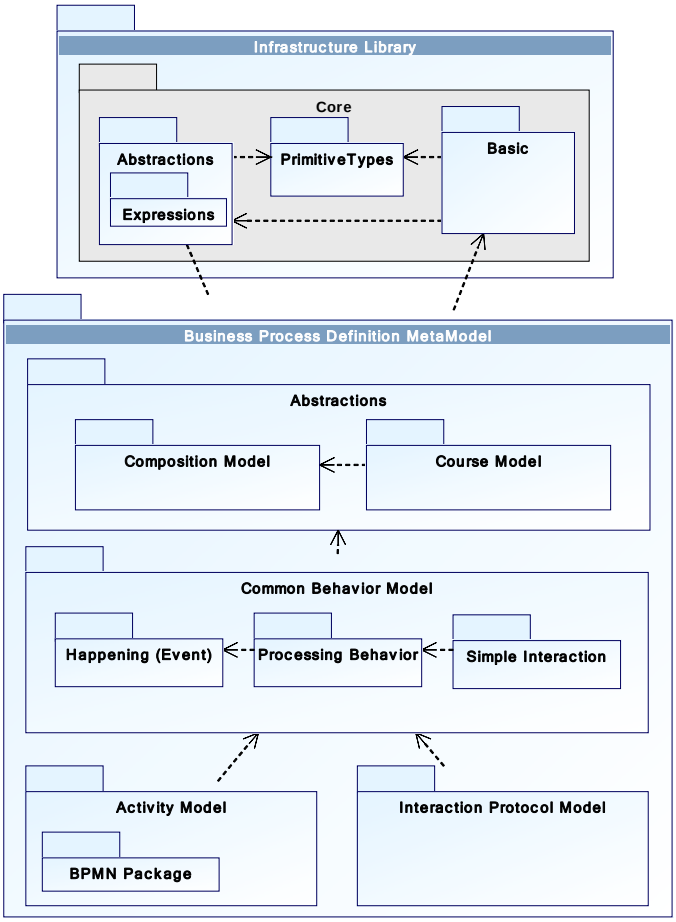

Figure 1: The Overall Structure of BPDM

A BPM Suite vendor extending the use of BPMN into execution will need the sophistication afforded by the Composition and Course Models. Similarly, a vendor seeking to extend and include alternate views of process and dynamics (such as state interaction machines) will require this sort functionality. On the other hand, those doing BPMN modeling will not need to concern themselves with these abstractions.

The Composition Model defines a framework for how the all the classes in the metamodel work together and the way they relate to runtime execution. It provides the underlying principles and mechanisms for reuse, (de)composition, interconnection, and instantiation. Effectively, the Composition Model affords a recursive mechanism to define the organization of any group of elements in a BPDM compliant model. In this way, it also facilitates extension of BPDM into other execution environments.

The Composition Model also defines composition relationships between entities (e.g., how an Activity is a part of a Process, and *"Interaction Flow"* is part of an Interaction Protocol). It also describes the recursive structure of a Composite—e.g. how a Process has *"Typed Parts"* that are Activities, some of which may in turn be *"Sub-Processes"*. It also defines relations between entities at the same level of decomposition—e.g., that one Activity comes after another under a certain Process, or that one participant interacts with another in an Interaction Protocol. These relations are also recursively decomposable—e.g. an Interaction can be composed of other Interactions.

Effectively, the Composition Model extends the capabilities of BPDM to include specific support for execution semantics. The composition defined in models is reflected accurately in the run time execution. This involves no only accurate (de)composition, but also fidelity in the relationships between the lower

level components. For example, that the order of Activities and messaging is maintained. The result is semantic interoperability—where two different process engines will reliably produce the same results at execution time when calling nested Sub-Processes and the constituent elements of a Composite Interaction.

Moreover, the Composition Model provides a uniform way for the system to walk the relationships between the entities, up and down a composition hierarchy, between elements at the same level, at both design-time and run time. Effectively, the Composition Model provides the structure that a compliant tool can use to 'understand' the structure of any modeling notation or other extension, as well as manage and monitor instances of those models when they are deployed. As a result, it is possible to define and manage virtually any sort of structure within BPDM—e.g. an organizational structure. In fact, BPDM provides mechanism that enables linking business processes to an independent organization structure, with roles in a process resolved at run time to individual people or business units.

The Course Model enables BPDM to support a wide variety of ways to model process dynamics. It provides a very general coordination mechanism of steps that is extensible to most models of process behavior. It includes functionality that is the basis for Gateways in BPMN (Splits and Joins), as well as Sequence Flows. Effectively these are constraints on a process and while they may feature strongly in a design-time process model, they are not part of the execution at run time (in the sense that they would never appear as real steps through which the process might travel). At run time, constraints define all possible paths through the process steps (represented by the Course Model and its specializations). This general model of process dynamics provides extension mechanisms for virtually any kind of state transitions.

The underlying rationale behind BPDM is that if the subtle differences and commonalities are captured appropriately, it then becomes possible to manage change effectively—even down to different scenarios, supporting what-if analyses and enabling methodology-driven rollback. BPDM also enables reliable semantic interoperability, robust execution and more effective process monitoring. As a result of this sophistication, BPDM can support a very wide range of business usage scenarios—from high level, abstract "business capability models" used in the boardroom; exchanging information with lower level, procedural modeling notations used in BPM projects, and then on into process execution environments (BPM Suites).

At first glance, BPDM may appear overly complex; full of strange terms and constructions. Yet, when one considers the multifaceted issues, the structure itself is relatively simple. The recursive nature of BPDM elegantly provides the necessary sophistication without resorting to exceptions and workarounds.