OMG releases BPMN 1.1 - What's changed?

(revised version as of April 2008)

Gero Decker[1] and Torben Schreiter[2]

[1] *Hasso Plattner Institute, Potsdam, Germany*

[2] *inubit AG, Berlin, Germany*

Abstract

The Business Process Modeling Notation (BPMN) is today's most important process modeling language. It enjoys broad acceptance among practitioners and many vendors are supporting it. The Object Management Group (OMG), the standardization body behind BPMN, released an updated version February this year.

We identified the most important changes for you and outlined them in this document. Not much has changed in BPMN 1.1 and if you were already familiar with version 1.0, we enable you to use version 1.1 right away.

1. Introduction

The Business Process Modeling Notation (BPMN) is a powerful tool for documenting your business processes. It is a standardized language for capturing what you currently do in your organization and where you want to change. And you are not the only one to use BPMN. There has been a major uptake and many vendors are supporting BPMN.

As every language, BPMN keeps on evolving over time. The Object Management Group (OMG) finalized version 1.0 in 2006. It was high time for an update and at the beginning of this year OMG released version 1.1. Now it is up to us to get you up-to-date.

In this document we will give you a brief overview of what has changed from BPMN 1.0 to BPMN 1.1. We will start with the new constructs available in BPMN 1.1. There are a number of new features you should be aware of. You should definitely have a look at signal events and at the refined notation for distinguishing catching and throwing events. We will also point you to other notational changes and changes in element attributes. On the other hand, we do not want to bother you with specifics that do not affect your modeling at all. Therefore, we keep this document short.

Before you read on and familiarize yourself with the changes in BPMN 1.1, let us put you at ease: You will not require additional training to cope with 1.1. You do not need to remodel your entire model repository. There is just a small delta you should be aware of.

2. Catching and Throwing Events

The semantical inconsistency between different types of events in BPMN has always been quite confusing for BPMN beginners. For example, the intermediate timer event is always triggered externally and blocks the flow until the particular trigger is fired (e.g. 11am on Sundays). The intermediate message event, on the other hand, can act as activity sending out a message to one of your favorite business partners. That is, it does not block the flow while waiting for a trigger, but rather throws an event and immediately continues the flow.

This difference in behavior amongst the event types is addressed in BPMN 1.1. The new specification introduces a categorization of event triggers into "catching" and "throwing" events. I.e. there are two kinds of intermediate message events now - one kind responsible for reception of messages ("catching") and one kind responsible for sending messages ("throwing"). Clearly, each kind comes with its own graphical representation (inverted colors).

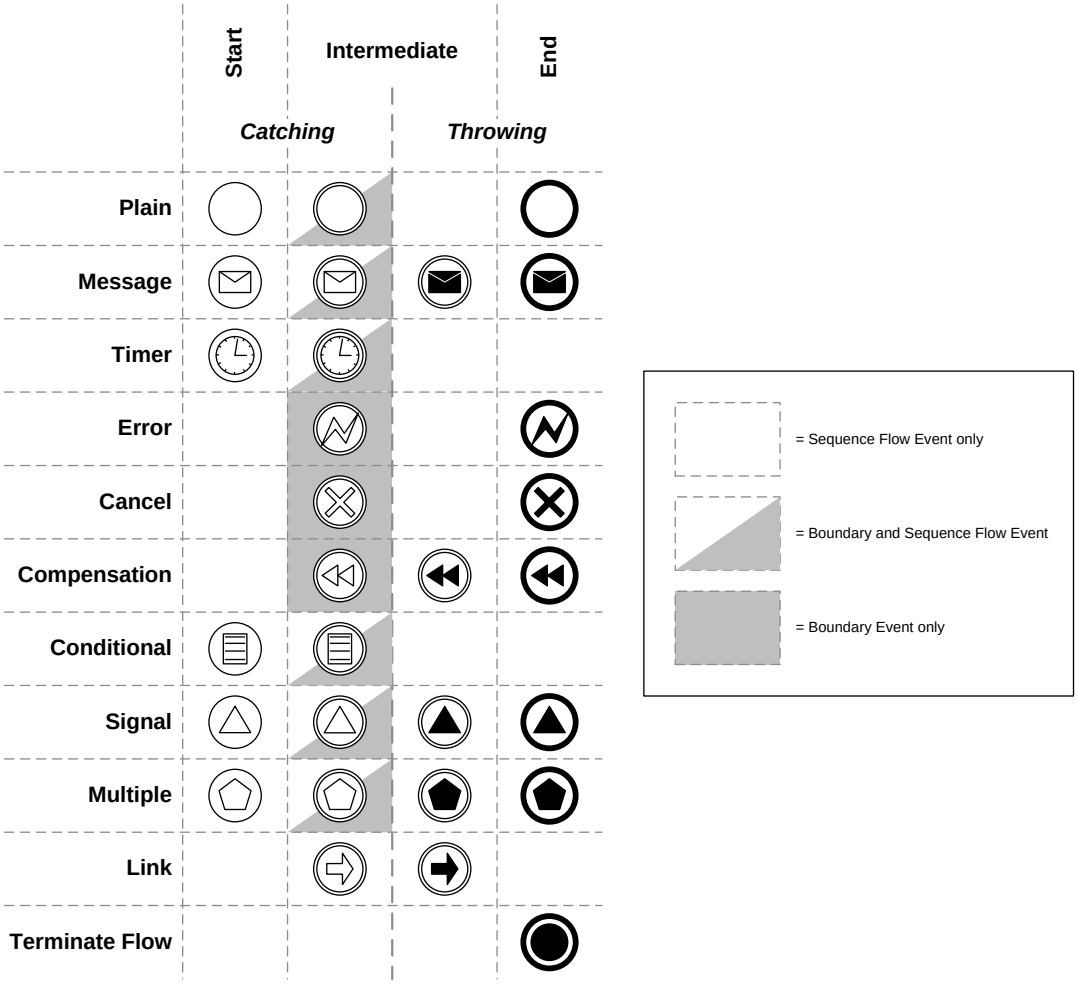

Figure 1. Distinction between throwing and catching events

Of course, start events are always of catching nature. For instance, take a BPEL process and visualize it using BPMN. The BPEL engine will wait for incoming messages, e.g. from one of your suppliers, and create new instances of the process.

Likewise, intermediate events attached to the border of activities always symbolize catching events. The occurrence of that event, e.g. a timeout, triggers the cancellation of the activity. In contrast to that, end events always represent throwing events. With those intermediate events that are not attached to activities, it becomes a little trickier: Here, either catching or throwing events applies. BPMN 1.1 really helps in this context, as catching and throwing are now clearly distinguished through their visual appearance.

You already know the large number of different event types from BPMN 1.0. This has not changed much in version 1.1. You are still confronted with a zoo of types. The new version even worsens this situation: In addition to the old types, it introduces a new type, the *signal event*. This is a powerful new mechanism that might affect you in your daily modeling. That is why we devoted an entire section to it (section 4).

The link event's description has undergone a major revision. Most likely, link events were one of the most misinterpreted constructs in the 300+ pages of BPMN 1.0's specification. 1.1 clears things up by stating that link events are solely meant to establish a point-to-point connection *within* a single process contained in a single pool. Link events can be used whenever a sequence flow does not fit into your layout. That is, two intermediate link events (throwing and catching) establish a sort of "wormhole" within the same process. Most often, link events are used as off-page connectors linking different diagrams together, jointly describing one process. Start and end link events do not exist any longer in BPMN 1.1. Too many people misinterpreted a pair of end and start link events as the call of a different process. For removing this confusion, there are only intermediate link events left.

For avoiding faulty or ambiguous models, please think of pairs of link events as simple sequence flows: They must be contained in the same pool. When used as off-page connectors, they still need to reside in pools with the same name. You are allowed to use multiple source link events, but do not use more than one target link event. The semantics for multiple source links can be thought of as multiple sequence flows merged in an XOR-gateway.

Figure 1 gives you an overview of all events present in BPMN 1.1. For your convenience we marked those intermediate events that only appear as boundary events or only in the regular sequence flow or in both ways. The overview also shows you that the old "rule events" where renamed to "conditional events". The semantics and appearance have not changed though.

3. Signal Events

The only completely new construct in BPMN 1.1 is the signal event. It can be used as start, intermediate or end event and is denoted as a triangle in the well-known event shape. Its semantics is described quite nicely in the specification through a metaphor:

> *"A signal in BPMN is to be seen as a signal flare broadcasting the occurrence of an event. Anyone seeing this flare is now aware of the signal and can react on it."*

Signal events give you an easy way for modeling certain control flow situations that were not possible in BPMN 1.0. Imagine you have two sub-processes that need to synchronize. BPMN is very strict with sub-processes: They have exactly one entry point and one exit point (or multiple exit points in the case of attached intermediate events). However, while the individual sub-processes have not been completed or canceled, no synchronization was possible in BPMN 1.0. Some modelers resorted to message flow within the same pool -- a workaround that was and still is not allowed. Signal events are a nice solution for this case.

But signal events can do more. They have a broadcasting nature, i.e. an event that is thrown once can be caught multiple times. Now, some readers might say to themselves: "Great, I do not need message events any longer -- they did not work for broadcasting scenarios, anyway. I will only use signal events from now on." Be careful! Please, stick to message events as much as possible and only use signal events sparingly. This will keep up the precision of your models. By the way, on an execution level, signal events assume a certain signaling infrastructure. Signals need to be published and subscribed for. Many process engines do not support this. And even worse: If you model throwing and catching signals across pool boundaries you assume that the different organizations share such a signaling infrastructure. This is quite unrealistic.

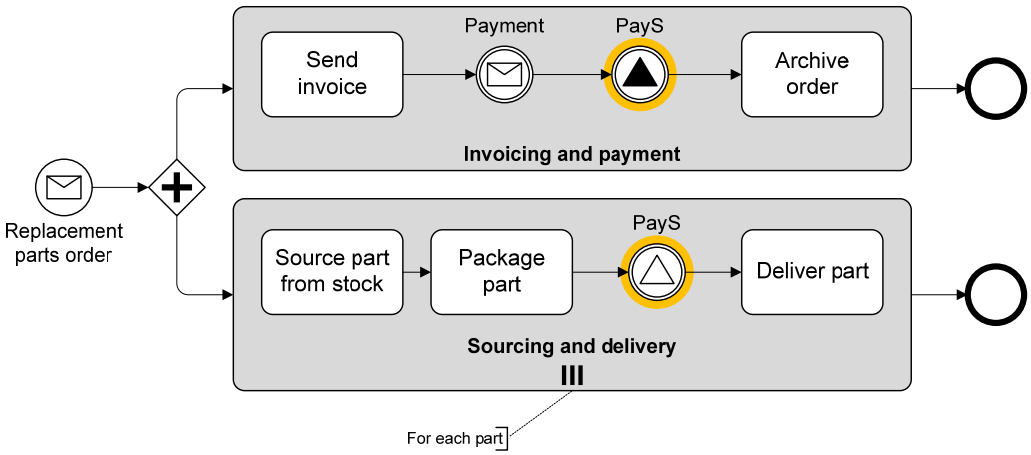

Figure 2. Signal events

On a business level, signal events might be an option for you as alternative to non-typed events. With signal events you can identify corresponding throwing and catching of events in your models. This again gives your models a little bit more precision.

Figure 2 shows you how signal events can be used. In this example you see that one invoicing and payment sub-process instance runs in parallel to a number of sourcing and delivery sub-process instances (indicated by the multiple instances marker). While the different parts can already be prepared for delivery, the actual delivery must only happen once the customer has paid for the parts. The "PayM" signal is thrown once payment has occurred. This event is caught in each of the sourcing and delivery sub-process instances. Here, we took advantage of both the broadcasting nature of signal events as well as the possibility to break the strict boundaries of sub-processes.

Figure 2 shows you that the marker for multiple instances activities changed. Too many people mistook the old symbol for a "Pause" symbol. The new symbol has improved slightly: Instead of two parallel lines, there are three of them now.

You might have noticed that signal events are already the third event type in BPMN for linking different parts of processes. For avoiding confusion, the following list will give you a rule of thumb when to use link, signal or message events:

- **Link events** help you to get a nice layout for your diagrams. Always use them within the same pool. You can spread them over different diagrams (off-page connectors). Link events do not allow you to model tricky control flow situations. Anything that cannot be done using sequence flows cannot be done using link events, either.
- **Signal events** help you to realize such tricky control flow scenarios, typically within the same pool and spread over different processes/sub-processes. However, try to use them as little as possible. And remember: they are not a substitute for message events!
- **Message events** symbolize the exchange of business documents or electronic messages between different pools. Corresponding pairs of send and receive events appear in the same diagram, connected by message flow. Even notifications between different partners should be modeled as message flow. Remember that message flow does not have a broadcasting semantics, nor does it assume any publish/subscribe infrastructure.

4. Other Notational Changes

Just like the rigorous distinction between throwing and catching events through different notational elements, there are further modifications that will help you understand and create BPMN 1.1 diagrams.

The event-based exclusive gateway is an important construct if you use message flow between pools. Here, it is often the case that one party makes a decision, e.g. a financial institution decides whether to grant a loan or not. The other party depends on this decision and waits for an acceptance or a rejection message. This scenario forces you to use an event-based gateway: Depending on which event occurs first, the corresponding branch is taken. This example is illustrated in Figure 3.

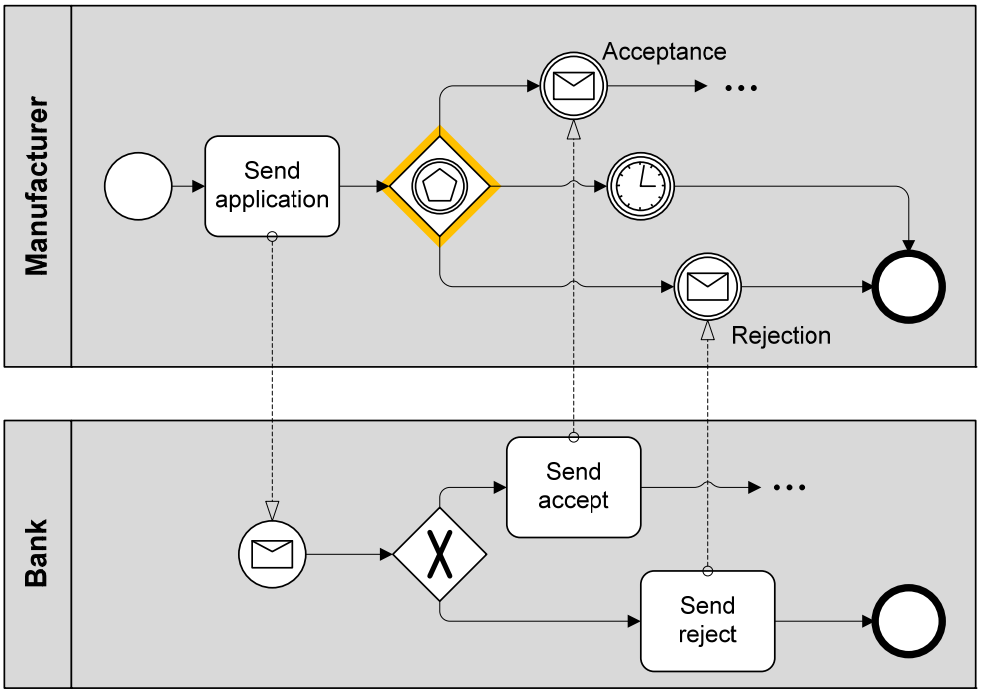

Figure 3. Event-based Gateway

The event-based gateway in BPMN 1.1 looks slightly different to what it looked like in 1.0. Instead of the hexagonal star it now has a pentagon in its center. The same shape is also used for the multiple events (start, intermediate, end). Now you ask yourself whether you did something wrong in BPMN 1.0 because you never used these multiple events constructs? Well, then you are not alone.

Multiple events were and probably still remain some of the less used constructs in BPMN. You probably simply use untyped (plain) events whenever you cannot exactly name an individual message, timer or conditional event to trigger something in your process or to be thrown in your process. As its name already indicates, multiple events can be used if multiple events are to be thrown or caught at a time.

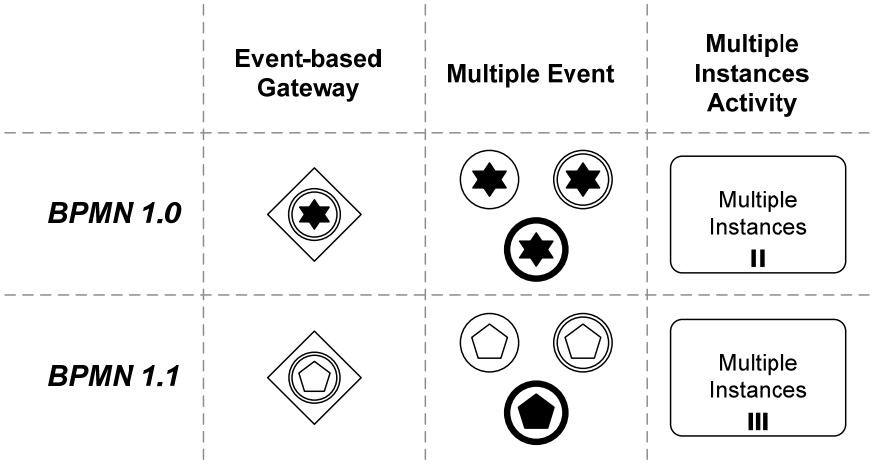

Figure 4. Notational changes

Pools and lanes are already known from the initial version of BPMN. Pools represent organizations, e.g. your own company, a supplier or a shipping partner. Lanes are organizational sub-structures of pools, representing departments or roles, for instance.

In fact, partitioning lanes into sub-lanes was possible in BPMN 1.0, already. Using sub-lanes you can subdivide a department into work groups or a role into sub-roles. Of course, the exact meaning of a lane must still be defined by you, the modeler.

What was missing in BPMN 1.0 was the clearly stated notation of nested lanes. The new specification retrofits this notation and comes with a minor notational change concerning lanes in general. The only thing that has changed for you is that there is an additional line separating your lane's description from its content. In BPMN 1.0 only Pools had this separate compartment for the title whereas lanes had their title in the lane itself.

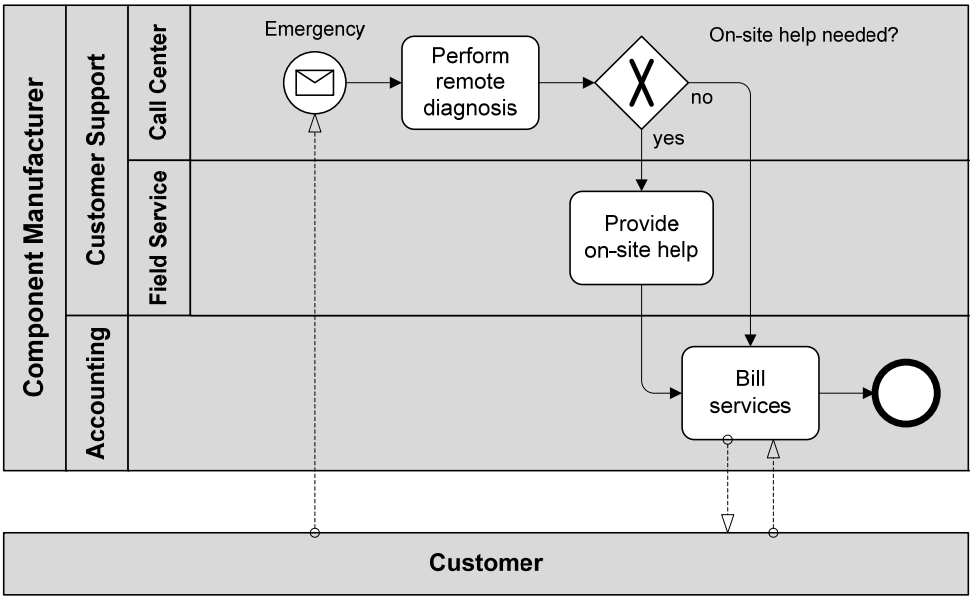

Figure 5. Nested Lanes with new layout

In Figure 5 you see how the pool "Component Manufacturer" is refined using the lanes "Accounting" and "Customer Support", both representing departments within the same company. "Customer Support" is further refined using the lanes "Call Center" and "Field Service".

5. Minor Changes

In addition to the changes described in the previous sections, there are a number of minor changes, which are summarized in this section. Please note that there are quite some changes concerning attributes of elements. We do not provide an extensive list of these attribute-related changes. We rather concentrate on the conceptionally meaningful differences compared to BPMN 1.0.

5.1. Activity-related changes

- For embedded sub-processes all start events are now required to be plain start events (without a trigger).
- If there are multiple plain start events contained in a sub-process, any outer sequence flow can be configured to trigger exactly one of them. Which one is to be triggered is configured via the "TargetRef" attribute of the incoming sequence flow.
- Independent sub-processes are now called "Reusable" sub-processes.
- The default type of a task has been changed from a Service task to an uncategorized task (type 'None').

5.2 Pool-/Lane-/Process-related changes

- One pool per diagram can be defined to be the main pool. This pool should then be in the center of attention.
- A process can now have one or more performers assigned. This is an interesting link to the resource perspective of a process. Indeed, this attribute enables you to capture the ownership of a process.
  However, it remains unclear why this attribute is defined to be a String rather than an organizational entity similar to a Participant.

5.3. Static Structure Diagrams for Attributes

A nice thing about the new specification document itself is the graphical visualization of element attributes in Annex B. Finally, the OMG takes the step towards providing an official model for the hierarchy of elements using UML static structures.

However, since the provided models lack any interconnecting associations, the diagrams are far from being a complete syntactical meta-model. In fact, it would be desirable to have a fully modeled meta model in the style of UML for BPMN as well. But we have to wait for BPMN 2.0 for this.

6. Conclusion and Outlook to BPMN 2.0

So what can we conclude about BPMN 1.1? Well, not much changed. BPMN 1.1 looks as pretty as BPMN 1.0 did and it still enables you to capture your business processes.

We really encourage you to use BPMN 1.1 for capturing your processes from a business point of view. As a next step, you should then consider how to support it best through information systems. Parts of the process will be executed, which in turn requires technical refinement of the process. The Business Process Execution Language (BPEL) is the de-facto language for business process implementation. The BPMN 1.1 specification still comes with a mapping from BPMN to BPEL. It creates the impression that generating BPEL out of BPMN is an easy task. But watch out! BPMN and BPEL are very different in nature and the specification does not reveal all the necessary details and limitations.

Is BPMN 1.1 perfect then? Some known issues remain. Especially academics would like to see more precision in the specification. But also vendors of process execution engines wonder e.g. how inclusive gateways should be interpreted. Or what about a well defined activity lifecycle? The data flow perspective in BPMN 1.1 also leaves room for further refinement of the specification. It is still unclear when data objects come to life in a process instance and when they cease to exist. BPMN's focus remains on control and message flow.

Major changes regarding BPMN can be expected with version 2.0. Here, a meta-model will be included in the specification along with a standardized interchange format. A focus will also be placed on a choreography view, providing an easier-to-use modeling style for describing interactions between different organizations and their dependencies. However, BPMN 2.0 is still very very far away.

You find the BPMN 1.1 specification at http://www.omg.org/spec/BPMN/1.1/PDF.

About the authors

Gero Decker is Researcher at the Business Process Technology group at the Hasso-Plattner-Institute in Potsdam, Germany. More information about Gero is available at http://bpt.hpi.uni-potsdam.de/Public/GeroDecker. He can be contacted at gero.decker@hpi.uni-potsdam.de.

Torben Schreiter works as Solution Architect for inubit, a vendor offering a comprehensive BPM Suite for modeling and executing workflow process models. inubit is based in Berlin, Germany. More information about the company and its products can be found at http://www.inubit.com/. Torben can be contacted at torben.schreiter@inubit.com.