THE
BPMN
GRAPHIC
HANDBOOK
Chapter 10
How to Model with BPMN
page 139

Appendix I
Diagram Summary
page 147

Appendix II
Events Summary
page 151
Chapter ONE

Introduction
What is a process and process modeling?

Let’s define these terms in simple words.

A process is the work someone (or something) do to accomplish an objective.

Process modeling means documenting the current state of a process to describe for example, how it starts, when it’s complete, and how a process get from A to B.
Chapter ONE

THE IMPORTANCE OF PROCESS MODELING

How do you organize to get the work done? Which are the steps? Who should do them?

All organizations need to know the answers of these questions in order to **UNDERSTAND** how their business actually works and **IMPROVE** how things are done.

However, until processes are documented, it’s impossible to **UNDERSTAND** them, **IMPROVE** them, and **CONTROL** them.

**PROCESS MODELING SHOULD BE THE FIRST STEP TO UNDERSTAND AND BEST ORGANIZE THE WAY A BUSINESS WORKS.**
Models and communication

So models are a **COMMUNICATION** tool since they help an organization to understand its processes.

Process modeling might be an easy task in a small organization, but think about a large organization:

- An organization with thousands of employees.
- An organization spread across many locations around the world.
- An organization that has relationships and/or dependencies with many suppliers, partners and customers.

It can easily become a mess. Interpretations of the model are hardly the same because people no longer have the same context or the same cultural references, leaving the interpretation of the model to the reader.

Obviously, this is a problem. The elements used to describe business processes must **COMMUNICATE** the **MEANING** intended by the modeler.

And that’s where BPMN comes in.
Chapter ONE

BUSINESS

PROCESS

MODELING

NOTATION
What is BPMN?

The Business Process Modeling Notation provides a STANDARD way of representing business processes.

This is done by specifying the sequence of activities that make up the process and its relating information in basically two levels of detail:

- **Simple**
  - Diagrams with activities, some decisions and important events. If needed, more information can be added, like more specific types of activities and events, roles, etc.

- **Executable**
  - Diagrams that contain sufficient detail and information that enables the execution of the model by software (Business Process Management Suites).

In some environments, having very detailed models with precise instructions of how the work should happen, allows to use them to drive the work itself by importing them in tools that can execute those instructions.

However, the focus of this book is to show how the Business Process Modeling Notation is used for representing business processes for "high-level" analysis purposes.
Chapter ONE

**August 2001**  •  BPMI Notation Working Group formed

**May 2004**  •  BPMN 1.0 is published

**June 2005**  •  BPMI becomes part of OMG

**February 2006**  •  BPMN 1.0 is formalized as an OMG specification

**January 2008**  •  BPMN 1.1 released

**January 2009**  •  BPMN 1.2 released

**August 2009**  •  BPMN 2.0 Beta 1 released

**May 2010**  •  BPMN 2.0 Beta 2 released

**January 2011**  •  BPMN 2.0 released
History of BPMN

In August 2001, 35 organizations formed the BPMI (Business Process Management Institute) Notation Working Group and BPMI.org. Their objective was to create a single notation for process modeling, since at that time, there were many notations and tools for that purpose. They published the first version of BPMN in 2004.

With the growing popularity of BPMN, BPMI.org was acquired by OMG (Object Management Group), home of the popular UML (Unified Modeling Language).

In 2006, the process of formally adopt BMPN as an OMG standard finished. Two years later, a minor update was issued (BPMN 1.1), and a year later, another version with some corrections (BPMN 1.2).

The notation from BPMN 1.2 to BPMN 2.0 didn’t change a lot. The two main motivations for BPMN 2.0 were to provide an official interchange format based on XML (eXtensible Markup Language) for process models and to make those models executable in a process engine.

After two beta versions, BPMN 2.0 was completed in 2010 and released to the public in January 2011.
More about modeling

One common mistake is assuming that there’s always only ONE correct model.

When you make a model, you’re always choosing what to include and what to exclude. You may want to show different levels of detail or focus on certain areas, but generally, there’s often a tendency to capture more detail than necessary.

For example, consider how to model the scheduling of an appointment for a medical checkup. The first step could be request for the patient information, then check for an available date, and finally, register the appointment. But what if we like to schedule more than one appointment, or if there’s no availability, and at the end, should we print or email a confirmation, or both?

BPMN is flexible, meaning it can handle many situations and requirements. So instead of worrying about developing the correct model, you should be worrying about using the notation in a correct way.

WE ALWAYS NEED TO THINK ABOUT WHO WILL USE THE MODEL AND FOR WHAT PURPOSE.
THERE IS HARDLY EVER ONLY ONE CORRECT MODEL
Good BPMN models

To create a good BPMN model, it must be:

**Valid**
The model should follow the rules of the BPMN specification.

**Accurate**
The model should reflect the actual process without any erroneous or biased view.

**Clear**
The model should be unambiguous and easy to understand. It should not depend on any other documentation.

**Complete**
The model should be as simple as possible, but it should indicate how the process starts, its significant events and its end states.
A good BPMN model **DOES NOT** show:

- **Why**
  A process are performed.

- **When**
  A process is performed.

- **How**
  The steps of a process are performed.
Chapter ONE

Basic elements and diagrams

BPMN uses a set of graphical elements to describe a process. Together, and refined with attributes and properties, these elements create diagrams for organizations of all sizes and domains.

However, there’s only four key elements in BPMN:

- **Activity**: The steps to perform a process.
- **Gateway**: Used to control the flow in a process.
- **Event**: Something that happens in a process.
- **Flow**: Used to show the order of the activities of the process.
On a higher level, BPMN allows you to model four different aspects of business process in four diagrams:

**Business Process**
Focuses on the activities and events of a process.

**Collaboration**
Focuses on the sequence of activities, events and messages between participants of a process.

**Conversation**
Focuses on the message exchange between participants of a process.

**Choreography**
Focuses on the interactions between participants.
TOKEns

The concept of a token is used to represent the behavior of a model. It’s simple, you just imagine a token moving along the diagram, executing the process along its way from start to end.

Let’s consider a simple registration process:

In this example, the start event (represented by the circle at the beginning of the diagram), generates the token:

The token then flows to the activity Register User Information, causing the activity start:
When a token arrives at an activity, the activity starts. This beginning and execution is called an **INSTANCE** of the activity. A new instance is created when a token arrives at an activity.

When the activity finishes its work, it emits a token that travels through the flow to the *Send Confirmation* activity:

Once the confirmation is sent, the token reaches the end event (represented by the circle at the end of the diagram) and the process completes:
Chapter ONE

Each of the elements in a BPMN diagram handles the flow of a token in an unique, and in some cases, complex way.

With this concept we complete the introduction to BPMN. In the following chapters, we’ll focus on the elements and diagrams of the notation.
Chapter TWO

Activities
WHAT IS AN ACTIVITY?

An activity represents an unit of work performed. A step inside a process.

It has a defined start and end and generally requires some kind of input to produce an output.

There are two types of activities:

- **Task**
- **Subprocess**

The "+" sign indicates that the subprocess contains more detail.
A task has no internal parts, it represents a single action.

On the other hand, a subprocess has parts that are modeled in a child-level process, a process with its own activity flow and start and end states.

NAME = NOUN + VERB

SUBPROCESSES HAVE OTHER PROCESSES INSIDE
AN ACTIVITY SHOULD BE LABELED AS NOUN-VERB
Task types let you represent in a PRECISE way how a task should be done.
NONE Task

This is the generic representation of a task and the one you will use more than 90% of the time. All other types are made for advanced uses of BPMN.
This task represents sending a message to an external participant. Once sent, the task is completed. A message can only be sent between different roles (more on that on later chapters).
**RECEIVE Task**

This task waits for the arrival of a message from an external participant. Once received, the task is completed.

Have you notice that the only (graphical) difference with the *Send Task* is the color of the envelop? Keep that in mind, it will be useful later.
Manual Task

This task represents work that is not automated and is performed outside the control of the BPM engine.
This task represents work that is performed by a human user with the help of the BPM engine or another software application.
This task represents work that is performed by an external system where there is no human intervention, like a web service.
This task represents work that is performed by the BPM engine as an automated function written in a script language like Javascript.
This task represents work executed at run-time in a business rule engine, generally, a complex decision.
Subprocesses can be represented in two ways:

**Collapsed**

As a **collapsed** subprocess, an activity with a “+” sign indicating that the child-level elements are detailed in a separated diagram.

**Expanded**

As an **expanded** subprocess, where an activity element encloses the child-level elements.
Expanded subprocesses

There’s an important rule about expanded subprocesses:

**AN EXPANDED SUBPROCESS SHOULD HAVE ONE (AND ONLY ONE) START EVENT.**

Which means that diagrams like the following are wrong:

Sequence flows cannot cross the boundary of the subprocess. It must have one start event to indicate the first activity to be executed. In the other case, a subprocess with two start events it’s ambiguous. Are they parallel activities? If not, which one is executed first? However, there are exceptions and workarounds which we’ll see later.


**Subprocess types**

Here, they are shown in their collapsed form.

The embedded subprocess is the normal type of subprocess that we have already covered. It is “embedded” in the parent diagram in either its collapsed or expanded form.

For the rest of the types, the expanded form will be shown in the following pages.
The difference with event subprocesses is that they are not part of the normal flow of the process. Instead, they are triggered by one of the following events:

- Message
- Error
- Escalation
- Compensation
- Conditional
- Signal
- Timer
- Multiple

In the example above, if the parent process takes longer than 2 hours, the event subprocess *Late order* will be triggered. We’ll see more about them in the next chapter.
Transaction subprocesses have the following properties:

- **ATOMIC.** Activities inside the transaction are treated as a unit. Either all are performed or none.
- **CONSISTENCY.** The transaction leaves the process (or system) in a valid state.
- **ISOLATION.** The effects of one transaction might not be visible to other parts of the process (or system).
- **DURABILITY.** Once a transaction has finished successfully, changes are persisted permanently.

Transactions have only three possible outcomes (the last two will be reviewed in chapter four):

- Success
- Cancellation
- Exception (error)
An ad-hoc subprocess is the exception to the rule about required start and end events. In this type of subprocess, the role who performs the process:

- Choose the sequence in which activities are done (they can even be executed in parallel).
- Indicate the activities to do (not every activity is required).
- Indicate when the subprocess is finished.

Because of this, there’s no need to have start or end events. Each activity may be performed zero or more times also.
AD-HOC SUBPROCESSES PROVIDE A WAY TO MODEL AN OPTIONAL LIST OF INDEPENDENT TASKS THAT CAN BE SELECTED AND PERFORMED IN ANY ORDER
A call activity is just a reusable activity. It’s often compared with a subprocess, and the distinction has to do with the way the child-level detail is used.

If for example, a subprocess is referenced in more than one process (diagram), you can define it in its own diagram and **CALL** it from each process that uses it, instead of copying the same subprocess in every parent process.

Graphically, a call activity has a thick border while a normal activity (task or subprocess) has a thin border.

Just remember, a normal activity has only one parent and it’s not available to any other process. A call activity can be included **UNCHANGED**, in many processes (diagrams).
Loops

With BPMN you can represent that an activity is executed many times in three ways:

- **Standard Loop**

- **Parallel Multi-Instance**

- **Sequential Multi-Instance**
There are two variations:

- **WHILE loop**: First, the loop condition is checked and if it evaluates to true, the activity is performed. Otherwise, the activity is not executed and the process continues (this means there’s a chance that the activity will never be performed). Every time the activity finish executing, the condition is evaluated again until it becomes false.

- **UNTIL loop**: First, the activity is performed and then, the loop condition is checked. If it evaluates to true the activity is performed again, if not, the process continues (this means that the activity is performed at least once until the condition is false).

In both cases:

- The looping symbol is the same (the variations are just a property of the activity).
- You can’t start an iteration before the previous iteration is finished.
- The number of iterations is unknown (it’s determined by the condition).
**Multi-Instance Loops**

A multi-instance loop is used when a process acts on some kind of collection. In this case the activity is performed once for each item in the collection by an instance of the activity.

This means that the number of iterations will be equal to the number of items in the collection. It is recommended to indicate this with a text label like *For each X*.

If you want the instances to perform in parallel, use three vertical parallel bars.

If you want the instances to perform sequentially, use three horizontal bars.

Just remember, a sequential multi-instance is not the same that a standard loop, being the main difference that the number of iteration are unknown in the latter.
Chapter TWO