CHAPTER 2

BASICS OF BPMN

In the introduction, we defined BPMN concepts as the key elements of a business process model. This chapter presents BPMN shapes with the aim of modeling a business process with the proper sequence of shapes.

Each BPMN symbol is classified by one of four shape types: rectangle, circle, line, or diamond. The shapes define classes of behaviors. Basic behaviors of the shape types include activities, gateways, events, sequences, and flows. 'Markers' within a shape define its behavior. All shapes reside in a participant's pool.

On one level, a process in BPMN is modeled by ordering these shapes; and order arises from sequences or communications. Shapes sequence with interactions or communicate with messages. With a grasp of the basic shapes and the markers, you can easily create models or read BPMN diagrams.

BASIC BPMN SUBSET— "OKAY, SO WHAT DO I REALLY NEED TO KNOW?"

The BPMN specification can overwhelm those who are new to process modeling. There is a subset of shapes that occur on most diagrams. This simplifies the diagram and makes the system more accessible to a wider audience. To learn the minimum of BPMN, concentrate on learning these shapes.

Before presenting a detailed definition of the types of events, Figure 2–1 lists the basic pallet of BPMN that every modeler should know by heart.

Many BPMN modeling tools and BPM/workflow automation systems use only the basic BPMN subset. The full BPMN pallet is quite large, and requires a lot of display "real estate". Also, if a tool is focused on ease of use, the full BPMN pallet might provide too many choices to a novice user. For many modeling tools, a right-click or a properties page for a shape will offer more details. This might include messaging options, conditions, escalations, task types, and timing options. Experiment with the modeling tool to see what else is available.

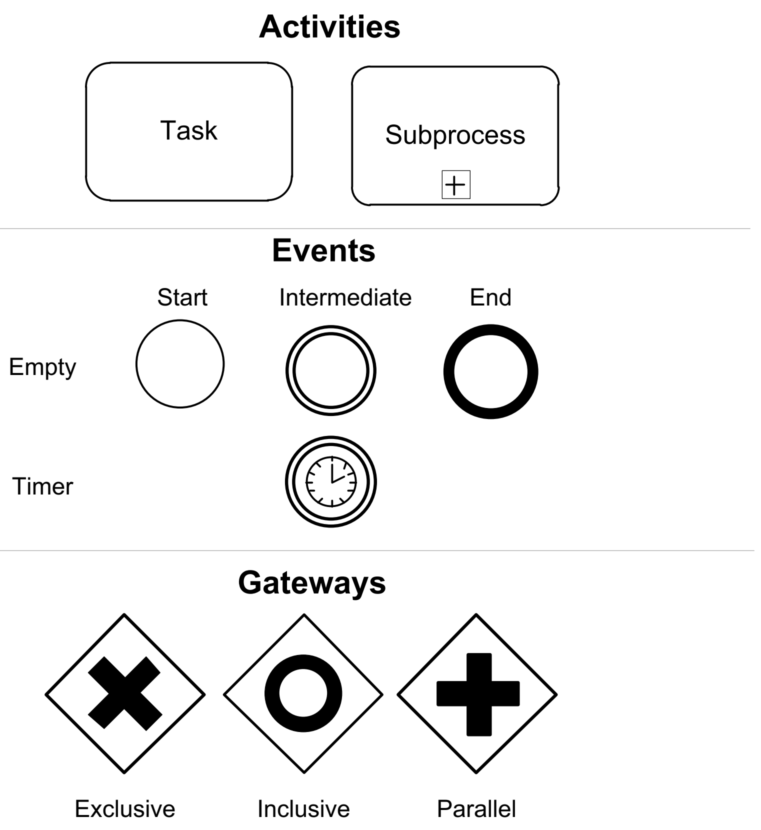


Figure 2–1 *Basic BPMN subset pallet.*

The subset in this figure of shapes is sufficient for most basic workflow processes. To detail various process behaviors, the BPMN specification builds upon these basic shapes. It is recommended that you start building your diagram with the basic subset first before jumping into more complex shapes. For example, an analyst using a message event might consider how the message works. However, an analyst, who draws an empty event might question if it is a message event at all. Chapter 3 will help guide you through this decision.

Before we delve into detailed BPMN, we will define the shape types.

Activities

An activity shape is represented by a rounded box, as exhibited here in Figure 2–2.

It defines where a process step occurs. Activity shapes include three basic types (see Figure 2–3).

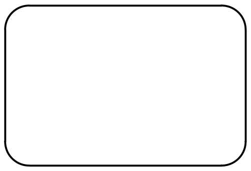
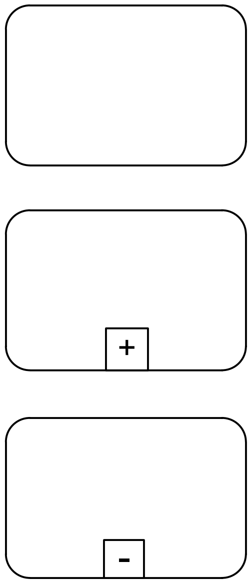


Figure 2–2 *Activity shape*



Task
A rounded rectangle showing the finest, or atomic, process step. It cannot be broken down to a finer level.

Subprocess (collapsed)
A rounded rectangle that can contain a series of other steps. The other steps are hidden from view; the plus sign indicates additional information.

Subprocess (expanded)
A rounded rectangle showing all the subprocess activities (from the collapsed subprocess).

Figure 2–3 *Three basic types of activity shapes.*

Sequence Flow Lines

Sequence lines are denoted by a solid line ending with an arrowhead (as in Figure 2–4). The arrow shows the flow or sequence of a process. This is frequently called *sequence flow* (see Figure 2–5).

Sequence lines define the sequence flow or transition between the logical steps performed by a participant. In the figure below, for instance, the contract is awarded after the bids are evaluated (both by the contract office). Bid Evaluation transitions to Contract Award within the pool contract office.

A transition is conceptualized through the sequence flow line. A sequence flow from one activity or event to the next shows the next as starting—or enabled to start. In the example, "Bid Evaluation" is complete, and "Contract Award" is started.

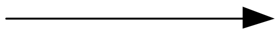


Figure 2–4 *A sequence line.*

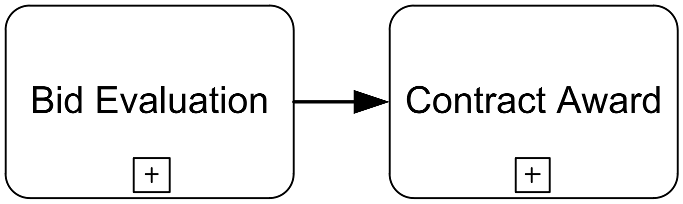


Figure 2–5 *Sequence flow.*

USING THE BASIC SHAPES: PUTTING IT ALL TOGETHER

With this understanding of the basic shapes, you can start building a diagram for a process model. Typical processes, such as supply chain management, serve as the business examples.

Suppose a modeling team is developing core business processes for a contract administration system. Figure 2–6 shows BPMN for a portion of a process's flow of activities.

Figure 2–7 shows a parallel split flow arising in a sequence flow. In parallel branches, process flow might move through any number of activities. Through modeling, analysts record the group's understanding of parallel activities in the process. For instance, they might observe organizational activities and dependencies. Branches might merge, or other branches might finish while other activities might continue to advance through the process.

Implicit Merge

In a simple merge, flow paths are rejoined. Consider a product moving through an inspection process in an inventory process. If there is no defect, the modeler notes that the following activity is "Mark Passed". Otherwise, the defect is identified and reported. In Figure 2–8, the paths that split after Inspect Item are rejoined (merged) at Shelf Item.

This is called an implicit merge because the merge is implied—the tasks merge, rejoining at "Shelf Item", with undefined conditions. The diagram fragment in Figure 2–8 does not specify how the "Mark Passed" task and "Report Defect" merge. The diagram's intentions might initially seem obvious; however, the design can be deceptive.

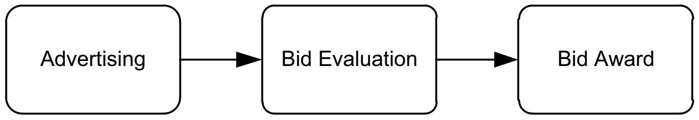


Figure 2–6 *BPMN for a portion of a process's flow of activities.*

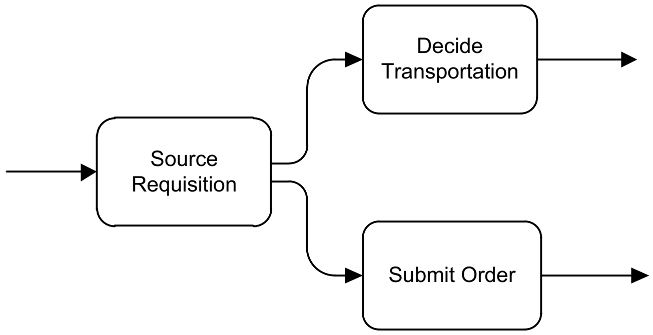


Figure 2–7 *Simple parallel split.*

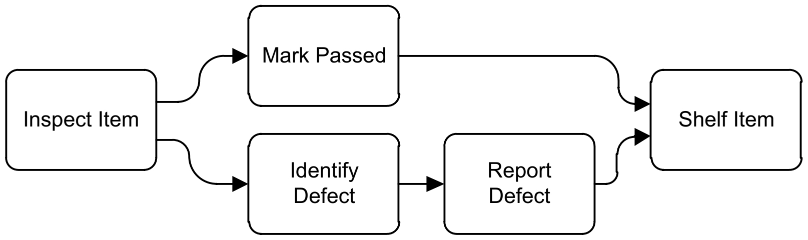


Figure 2–8 *Implicit merge in an inventory process.*

Implicit merges are ambiguous. Since it is implied, one cannot accurately see what happens at the point of the merge (Shelf Item). Perhaps the process will execute or continue uninterruptedly from "Mark Passed" to "Shelf Item". Otherwise, the process might wait until both "Mark Passed" and "Report Defect" tasks complete before continuing to "Shelf Item". For these reasons, and because there is no "flow control", the "Shelf Item" task might execute more than once.

We recommend explicitly specifying the transition from "Inspect Item" to "Identify Defect" with flow controls. This avoids the implicit process merge. Otherwise, the process diagram should describe the merge behavior at the "Shelf Item" point with more shapes. Figure 2–9 shows a choice in defining conditions for the two paths leaving the "Inspect Item" task.

The diamond symbol on a transition path shows the process path when data matches a specified transition condition. The sequence flow line with the slash marks a default path. A condition is a Boolean expression, based on process data

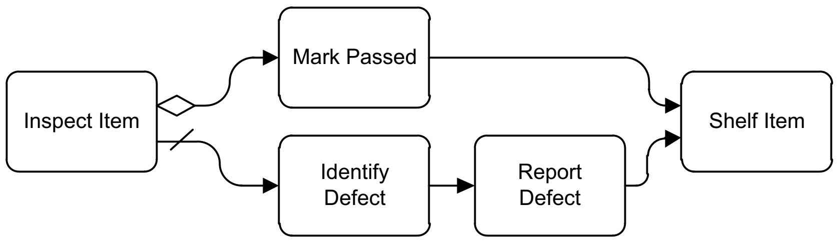


Figure 2–9   *Splitting paths with a condition.*

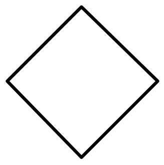
that control a sequence of activities. For instance, a condition might assess the value of a Pass Inspection flag and follow the transition to "Marked Passed" when the value is "YES." The process follows the default path when the transition condition fails to be true. Whenever conditional paths are specified, use a default path.

Figure 2–9 shows how the process takes a path to either "Mark Passed" or "Identify Defect". A default path clearly states that either path can occur, but not both. By definition, the default path has no condition. The default path indicates the sequence flow that is taken when all other paths do not meet their proscribed conditions.

Gateways

Gateways provide explicit control. A gateway splits or merges paths in a BPMN diagram at a specific point. Gateways direct sequence flows with data, or they specify various path splits. When used as a merge point, a gateway clarifies the implicit merge.

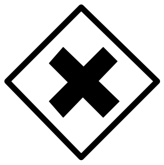
The simple gateway shape is an empty diamond, as shown here.



The simple gateway shape does not specify a behavior. In the BPMN specification, the function of the empty gateway shape is equivalent to the data-based exclusive gateway (described next). Since the data-based exclusive gateway has a clearly-marked explicit behavior, it should be used. A process diagram in Figure 2–10 shows the exclusive gateway with an X 'marker' inside the diamond. For the rest of this book, we will favor the use of more descriptive gateway shapes.

Data-based Exclusive Gateways

The data-based exclusive gateway is a diamond shape with an enclosed X, as shown here.



The data-based gateway shapes are either *exclusive* or *inclusive*. The term data-based means the data in the process selects which transition to take. So, in the data-based "exclusive" gateway, process data defines conditions for the paths leaving the gateway.

The exclusive gateway is used when the process moves along only one path, excluding all other paths. For example, examine Figure 2–10.

Returning to typical business processes, "Truck", "Air Carrier", and "Rail" tasks appear after deciding what form of transportation will be used. The gateway specifies exclusive behavior, the flow takes only one path, and all others are excluded. The sequence flow line to the "Rail task" has a marker indicating the default path. In this example the designer has chosen the "Rail Transportation" as the default, again, always use a default path whenever conditional flows exist.

As mentioned, BPMN includes a data-based inclusive gateway. So, next, it might seem logical to describe this gateway. First, however, an understanding of the concept of parallelism in processes is needed.

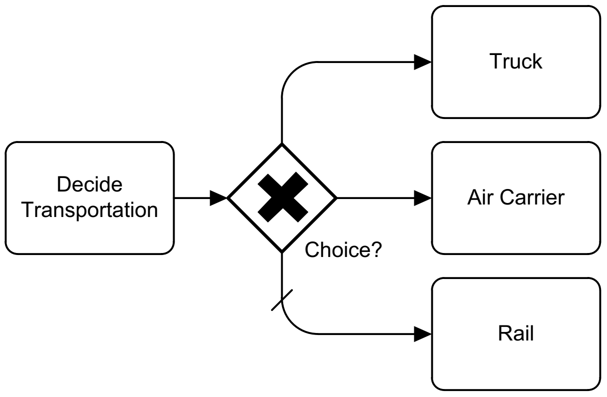
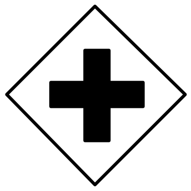


Figure 2–10 *Data-based exclusive gateway.*

Parallel Gateways

A parallel gateway is a diamond shape with an enclosed cross, as shown here.



In the parallel gateway, all paths leaving the gateway are executed (see Figure 2–11). This gateway is used in places where the process unconditionally follows multiple branches. In other words, after the parallel gateway, all the activities occur simultaneously. Parallelism is best when sequential execution is not efficient—for instance, one long-running activity might delay everything else in the process. Here, we use the parallel gateway shape to show multiple activities being performed simultaneously.

There is no default condition when a parallel gateway is used. All paths are always taken and the transition to all paths occurs at the same time.

When deciding whether or not to use a parallel gateway, there are several things to consider:

- Are all tasks always executed? If not, use an exclusive data-based gateway
- Are there any tasks in the sequence that depend on each other? When there is a dependency, use a sequence of tasks rather than parallel
- What is the impact on the subsequent tasks if all paths occur simultaneously?
- In later activities, should the process continue sequentially rather than in parallel? This determines where the parallel sequence flows need to merge before subsequent tasks can begin.

Most often, parallelism is needed for a number of time- or resource-intensive tasks that are unrelated to any other process activity. For instance, external activities such as lab testing or external reviews might take place in parallel, while internal administrative tasks continue.

Explicit Gateway Merge

An explicit gateway merge uses a gateway shape to show the joining of multiple paths. Figure 2–12 shows implicitly merging paths after a gateway shape.

Exactly, what happens at the merge point—the “Release Funds” task? The parallel gateway following the “Terminate Contract” task says that notification of the contractor, contractor administrator, and accounting occurs in parallel. Each

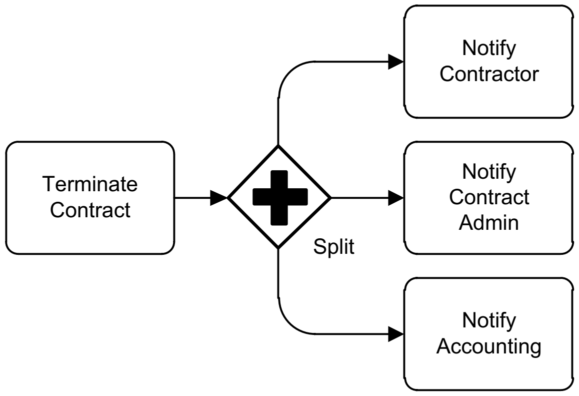


Figure 2–11 *Parallel gateway—all paths will be taken simultaneously.*

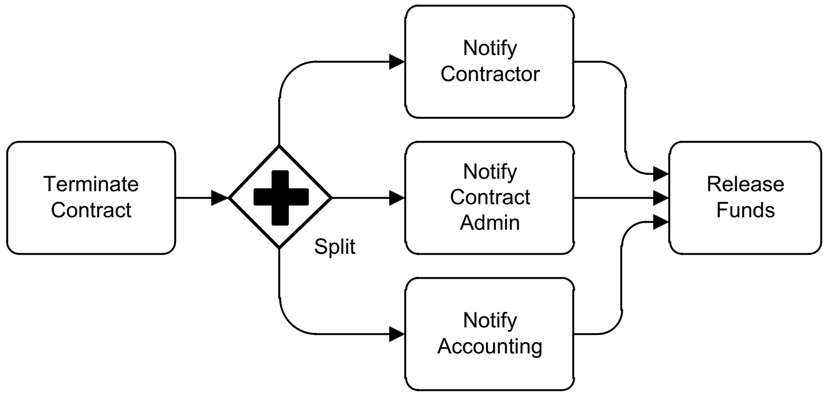


Figure 2–12 *An implicit merge with a parallel split.*

begins at the same time. The notification tasks require different times to complete. One may conclude before the others. So, the diagram does not seem precisely defined at the merge point. Potentially, the Release Funds task could happen three times. Below is an example of why.

After the "Terminate Contract" task, the three notification tasks begin. Suppose "Notify Accounting" completes before "Notify Contract Administration" and transitions to the "Release Funds" task, continuing through the following steps. Meanwhile, the activity at the "Notify Contract Admin" task completes a few hours later. Here, another copy of the "Release Funds" task begins. It continues processing the flow. Finally, the "Notify Contractor" completes, and so forth. Consequently, the "Release Funds" activity has occurred three times. If this is the intent of the process, then the model in Figure 2–13 is correct.

Implicit merges are often obscure and misleading. Figure 2–14 depicts this possible scenario. It also shows what could happen in each path.

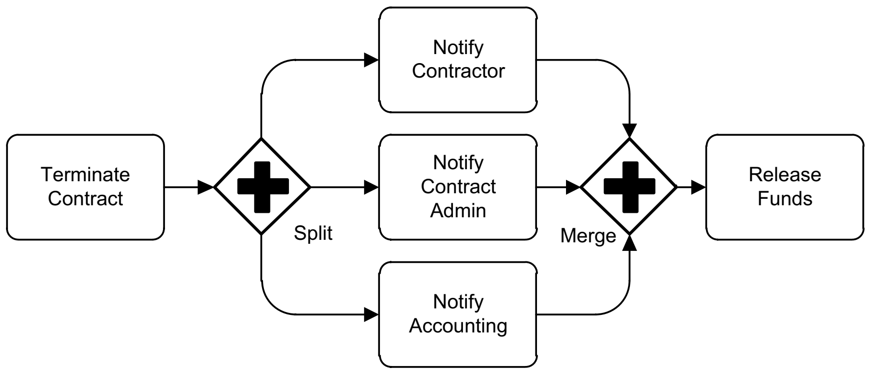


Figure 2–13 *Proper merging of parallel paths.*

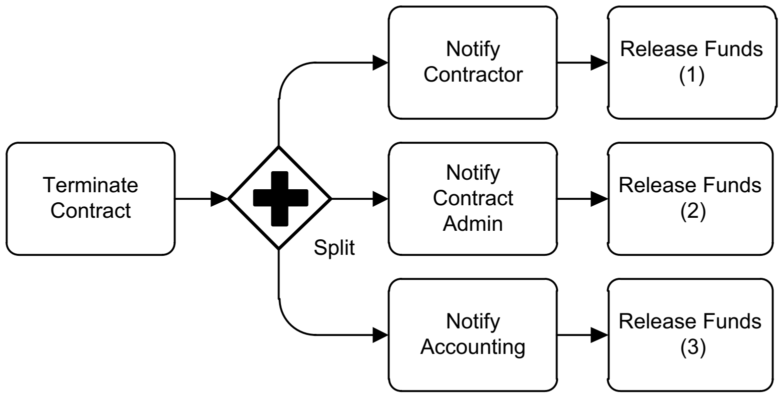


Figure 2–14 *Illustrating the ambiguity of the implicit merge.*

If the "Release Funds" task should not occur three times, then the process might be clearer with a parallel merge. The parallel merge shape is identical to the parallel split. Placement is the only difference between the parallel split and merge.

At the diagram's merge point, before the "Release Funds" task, all paths must complete before continuing. The "Release Funds" task is dependent on the completion of the three notifications; the process will coordinate all paths.

In contrast to a parallel merge shape, the exclusive gateway merges paths, but with different behavior. Figure 2–15 depicts a fragment of a supply chain process.

The data-based exclusive shape can merge activities (here, three notifying activities). The use of the gateway is optional for exclusive paths. Since one path emerges from the gateway after the "Terminate Transport" task, we connect the lines from the notification tasks directly to the "Reverse Freight" task. The exclu-

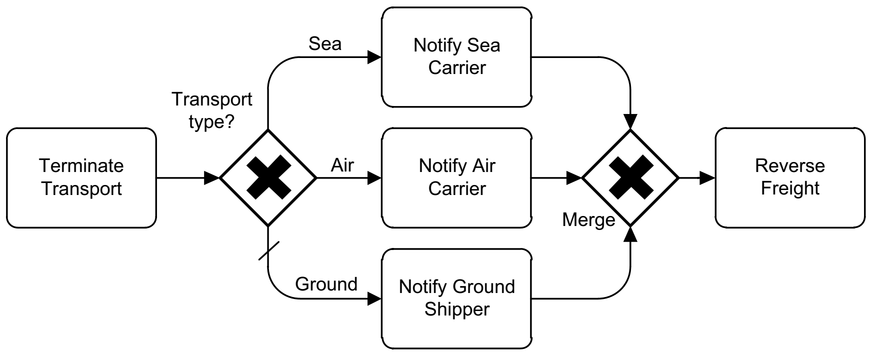


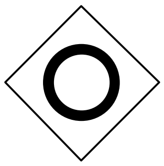
Figure 2–15 *An explicit merge for the exclusive data-based gateway, with a default condition.*

sive data path specifies the merge point, at which the final "Reverse Freight" task is run.

For diagram clarity, the explicit merge point is recommended. A good standard practice is to use an exclusive gateway shape for a merge point if the branch starts with an exclusive gateway. In Figure 2–15, the exclusive gateways for both split and merge document the end of the three paths. As a diagram becomes more detailed, dozens of steps might happen between "Terminate Transport" and "Reverse Freight". As earlier mentioned, the behavior of the implicit merge can be confusing. This might be especially true if a process diagram spans several pages. For example, consider viewing large diagrams on computer screens. You must scroll back and forth to see the entire flow. With merge shapes at the end of paths, the diagram becomes much easier to read. Since parallel and inclusive gateway shapes need a shape to merge flows, the exclusive merge gateway shape is consistent with the rest of the diagram.

Data-Based Inclusive Gateway

The data-based inclusive gateway is a diamond shape with an enclosed circle, as shown here.



Since multiple paths could be taken, this gateway is called inclusive. The gateway evaluates process data against a condition, which is where the term "data-based" comes from. The gateway includes all sequence flows that have a condition that evaluates to "True."

The inclusive gateway shape is a hybrid of the data-based exclusive and parallel gateways. There is a condition for each flow path., and one or more of the conditional paths might be taken.

Figure 2–16 presents a fragment of an order management process. Depending on the order total, various processing steps must be taken.

After the "Receive Order" task, the other paths execute whenever associated conditions are true. If the order amount is over $1000, additional validation occurs to prevent fraud. Additionally, orders over $5000 must be approved by a manager. When none of the conditional paths is taken, the process takes the default path. If the order total is not over $1000, only the standard processing activity occurs.

As with parallel gateways, care should be taken when merging inclusive paths. Consider the process flow after the Standard Processing task if we specify no default path. If no default path is specified and no condition evaluates to true, a deadlock occurs. In a deadlock, the process cannot go past the gateway. It never continues or completes. To avoid a deadlock, if the process needs inclusive gateways, then verify that each has a default path.

As with other gateways, the BPMN specification allows implicit merging for data-based gateways. Again, your design should be unambiguous at the point where sequence flows merge. Also, the use of explicit merging for parallel and implicit gateway shapes is a good best practice. Sometimes a process might perform an optional, added activity, or it might bypass activities under other conditions. An activity is not compulsory on every branch coming from a gateway.

In Figure 2–17, the Receive Order process takes the extra step ("Get Manager Approval") when the order amount is more than $5000. By default, the Get Manager Approval activity is bypassed. With this notation, it is simple to create a default path to bypass the extra step.

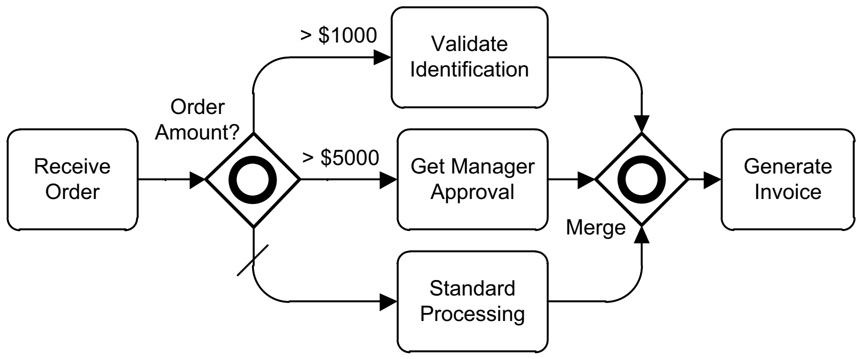


Figure 2–16 *An explicit merge with a data-based inclusive gateway.*

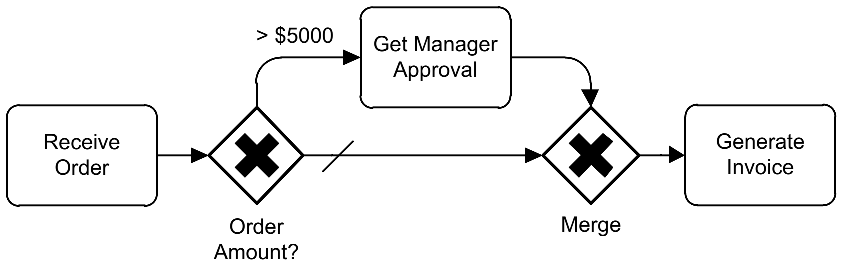


Figure 2–17 *A data-based exclusive gateway controlling an optional step.*

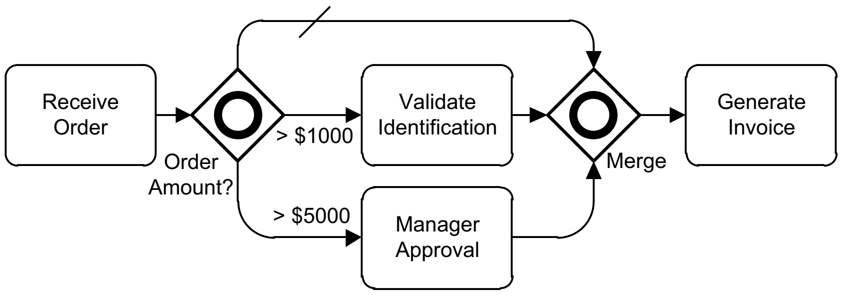


Figure 2–18 *Inclusive gateway with a default bypass path.*

Inclusive gateways might also use transitions without activities. In Figure 2–18, the process's default path bypasses all the conditional paths. Without one or more conditions in the gateway evaluating to "True", the process takes the default path—straight to the merge point—with no additional activities.

Figure 2–18 shows an order management process fragment. The Standard Processing task in Figure 2–17 was replaced with a default path void of activity. As an inclusive gateway, the default path is not taken if any conditional path is true. One or more other tasks (e.g. "Validate Identification", "Manager Approval") can occur in parallel.

Gateway Labels

The transitions entering the gateway in Figures 2–15 through 2–18 are labeled. Document gateway conditions with labels in a question form. For instance, a clear label might be "Selected color?". The sequence flow transitions leaving the gateway should be labeled to answer the question the gateway asks. The answers "red," "blue," and "green" could be possible answers to the question. In Figure 2–18, the answers to the question 'Order Amount?' are covered by the conditions '>$1000' and '>$5000'.

Inclusive/Exclusive Gateway Best Practices

There are a few best practices that should be considered:

- Limit use of the inclusive gateway to situations needing parallel execution. In other situations, try to use multiple exclusive gateways
- Avoid using the inclusive gateway where the conditions are not related, such as document gateways. A gateway should ask only one question—for example, "Selected color?", "Low inventory," and "Order amount more than $5000" can be assigned to separate exclusive gateways.

Ad-hoc Subprocess

Ad-hoc activities must be completed, yet the order in which they are performed is unknown. The ad-hoc subprocess is used for these situations.

Figure 2–19 depicts a vendor evaluation process from a Supplier Relations Management (SRM) process. The four activities must be completed, yet there is no apparent order to the evaluation steps.

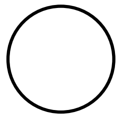
Activities inside any subprocess start with the first shape in a sequence flow. When sequence flow is not defined, all the subprocess activities start simultaneously. All activities must stop before the subprocess is complete. Therefore, subprocesses also include an "implicit merge."

An ad-hoc subprocess shows activities that will probably be performed in a sequence, but whose order is not defined. A tilde marker is shown on the subprocess shape when it is ad-hoc. The ad-hoc behavior is different from an implicit split in a subprocess. For example, a grocery list could be like an ad-hoc subprocess. All items on the list are required, but items are added to the cart as they are found in the store—in no particular order and one at a time. In a parallel pattern, all items would be simultaneously added to the cart. Without the ad-hoc marker the subprocess depicts a parallel pattern.

The ad-hoc subprocess simplifies some complex patterns. During the development of a diagram, when the execution order is yet unknown, use the ad-hoc subprocess. A parallel split or a defined sequence flow with gateways describes most processes. In the shopping cart example, however, the ad-hoc subprocess shows the desired pattern with the minimum of shapes.

Events

All events are circular, as in the empty start shape.



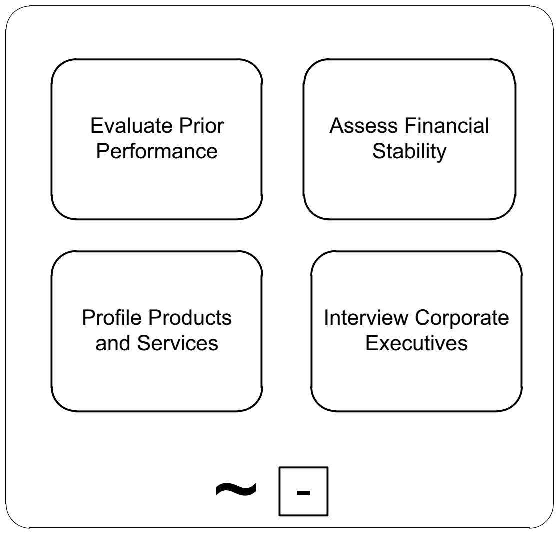

Evaluate Prior Performance
Assess Financial Stability
Profile Products and Services
Interview Corporate Executives

Figure 2–19 *Example of an ad-hoc process.*

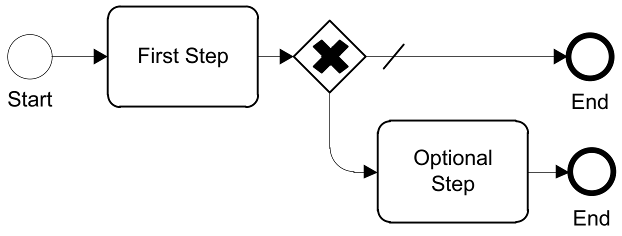


Figure 2–20 *Ending example with a gateway.*

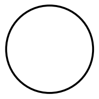
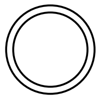
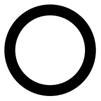
The event shape defines a point of interest within the progress of the process. It can be at the start of a process, the end of a process, or within a process flow (see Figure 2–20).

The event shape is further divided into three major categories, as shown in Tables 2–1.

The BPMN specification allows a process to start with either a task or a gateway. A start event, however, is often a better choice for starting a process diagram. Start events explicitly show how and where the process starts.

Let's begin our discussion of events with the empty start and events shapes. Consider the diagram in Figure 2–20 without the start and end events. With or without the start and end events, the process in the diagram says the same thing. The events define the start and end points for the reader.

Table 2–1 The Event Shape

Start event This is used at the start of a process. Start event shapes are drawn as a single thin line circle.	
Intermediate event This is used between the start and the end of a process. Intermediate event shapes are drawn as a double thin line circle.	
End event This is used to show where a process flow may end. The shape is drawn with a thick solid line.	

There can only be one start event. The end event, however, may be used more than once in a pool. Figure 2–20 shows the usage of more than one end event.

Adding the gateway after the "First Step" task, the optional path is the "Optional Steps" task. Under certain conditions, the process just ends. A process end is an event, not a task. Therefore, the empty end event shape is more suitable for this scenario. Adding the end event simplifies the diagram. Without the end event after the "Optional Step" task, there would be an implicit end. Explicit shapes refine a process design's clarity.

The intermediate empty shape, the double circle, specifies an intriguing point in the diagram or shows a place where the state or status changes. Since the shape is an empty event and behavior is not specified, the description for this shape is uncertain. Consider a business process that has entered a new status. For example, it might transition from a "pending" status to "approved."

The intermediate empty event (as shown in Figure 2–21) documents a point, such as a Key Performance Indicator (KPI), in the diagram. KPI's quantify objec-

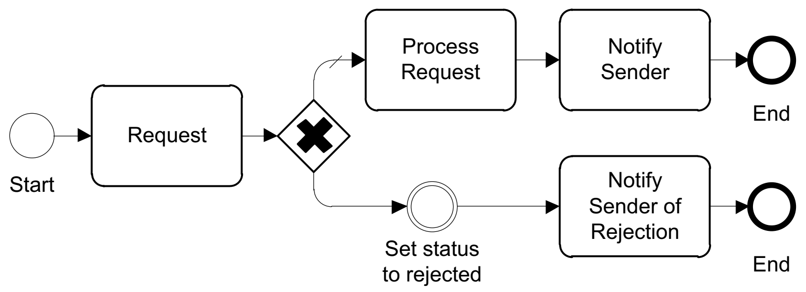


Figure 2–21 *Examples of an intermediate event.*

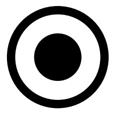
tives that measure the strategic performance of an organization. When a status changes to "rejected" in the process example, a KPI might be the number of rejected requests. In addition to recording status counts, the process might record more process data at this point. To track this data, a business builds business intelligence (BI) and business activity monitoring (BAM) into the processes.

The rules for the proper use of the intermediate empty event include the following:

1. A transition line must leave and enter an intermediate shape[1]; otherwise, use a start or end event.
2. The intermediate empty event does not show any delay in the process.
3. It does not have any conditions associated with it.
4. It does not imply a point of synchronization.

Terminate Event

The terminate event causes all activities in a process to be immediately ended, and its shape in a BPMN diagram is depicted as shown here.



You can use the terminate event shape to cancel all activities in a process.

In Figure 2–22 the activities "Search for Candidate" and "Negotiate Contract" are enabled to occur in parallel. If contract negotiation fails, it is useless to look for project staff. If the gateway "continue" condition is "No", then the process will terminate. This includes all activity in the Search for Candidates subprocess. Since the Search for Candidates process is ongoing and does not merge with the parallel flow, a terminate event is an excellent option to stop both flows.

A terminate event is not used for process flow that stops normally; rather, it is used for abnormal circumstances. Always use an end event unless there is a specific intention to terminate, such as in the example above.

Participant Pool

The pool shape (shown in Figure 2–23) contains the elements of a process flow performed by the process participant. The pool shape is sometimes called a swimlane, but the terms lane and swimlane can be ambiguous. Be careful not to confuse the term swimlane with a BPMN lane. The term swimlane comes from the UML specification. BPMN does not have a shape called swimlane. Although the

1. One exception to the rule is when an intermediate shape is used within an expanded subprocess. This will be discussed later.

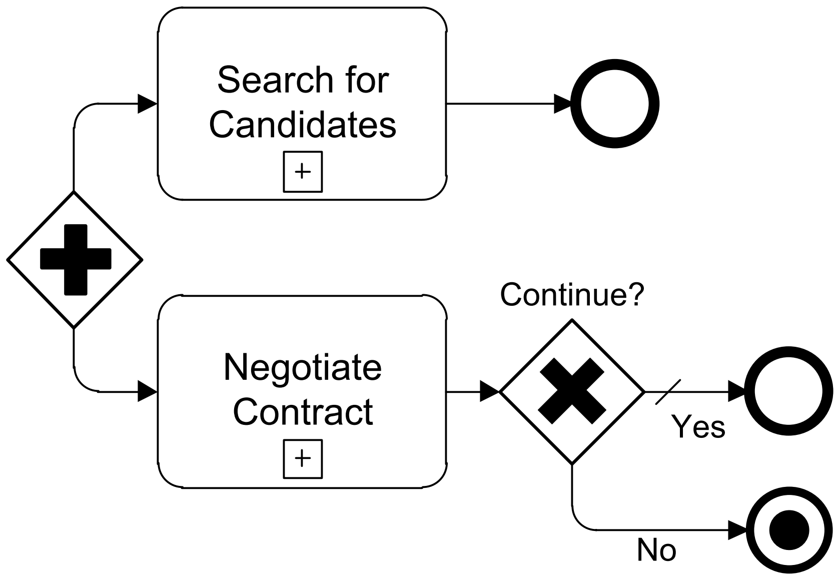


Figure 2–22 *Usage of the terminate event.*

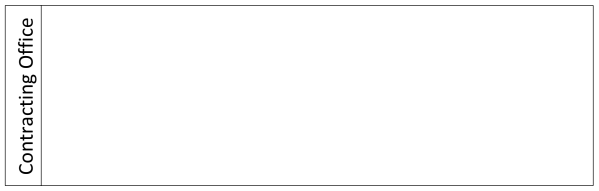


Figure 2–23 *Pool.*

usage of pools and lanes in BPMN are similar to the usage of a swimlane in UML, there are significant differences.

The pool shape represents a participant. As stated in the introduction, a participant is anything involved in the process—a person, a system, or even another process. The pool creates a context for the diagram, referring to a participant. Participants can be specified, such as a manager in HR, or they can be general, such as the entire company.

BPMN allows for a further breakout of pools into logical groupings called lanes. In some cases, this could suggest a role for the participant, such as legal, recruiting, or contract negotiator. But actually, the pool itself is one participant. This is different from the UML swimlane concept where a lane defines a person's role. In BPMN, the role is defined on the pool, and the lane is used to group activity. For example, Figure 2–24 shows the contracting office (a role), with activity groupings in lanes (Contract Negotiation and Staffing Search).

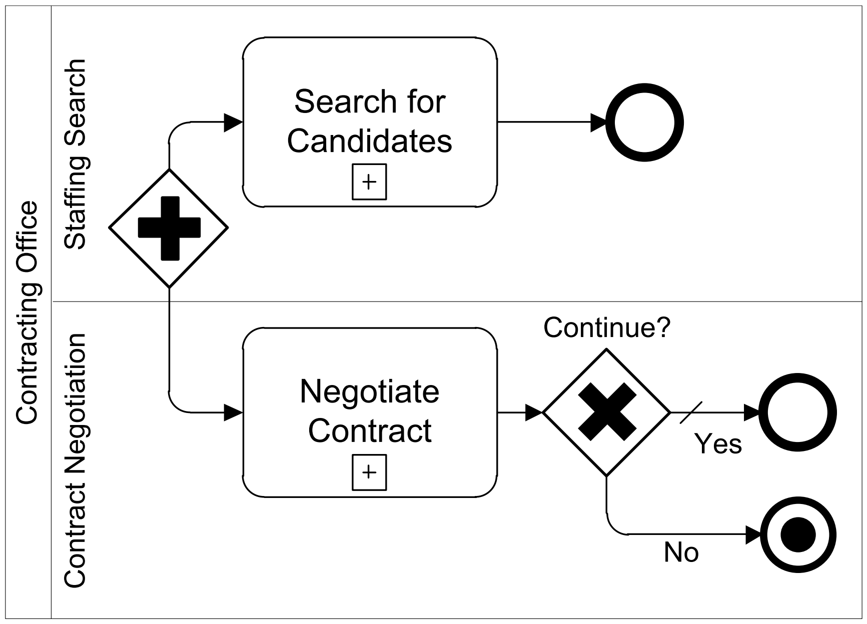


Figure 2–24 *Addition of pool and lanes to the project ramp-up process.*

Usage of lanes is entirely optional. The addition of lanes often adds clarity to the diagram.

To clarify the application of lanes, consider that all activity within a pool is being performed by one participant. The parallel shape indicates that this participant is potentially doing many activities simultaneously. If we want to show that more than one participant is involved, another pool is used. Messaging (covered in the next chapter) is used to coordinate activity between participants.

Another common style of BPMN is to show several participants, such as HR, Accounting, Sales, and Legal, all in one pool, divided by lanes. This is a dated method, the legacy of the flowchart process model. However, there is still justification to use this style today in BPMN. A pool represents a single participant—and a participant can be people, systems, or processes. When the pool represents a process, the participants in that process could logically be grouped into lanes. This is a common approach used in many popular BPM/Workflow automation systems.

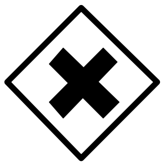
Summary

The first part of the chapter presented a basic pallet of shapes that every BPMN designer should know by heart. All but the most complex process scenarios can be modeled with these shapes.

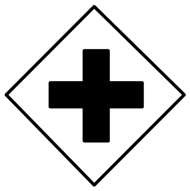
Next, we covered the basics of activities, gateways, and events in BPMN.

In BPMN, a gateway directs the flow of a process. There are three basic gateways: the data-based exclusive gateway, the parallel gateway, and the data-based inclusive gateway.

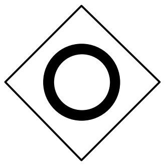
The data-based exclusive (remember x'clusive) gateway is denoted:



The parallel gateway is denoted:



The data-based inclusive (remember or) gateway is denoted:



Events are denoted by circles. Starting events are denoted by thin lines, intermediate events, by double lines, and end events, by thick lines. These notations hold for all events in BPMN.