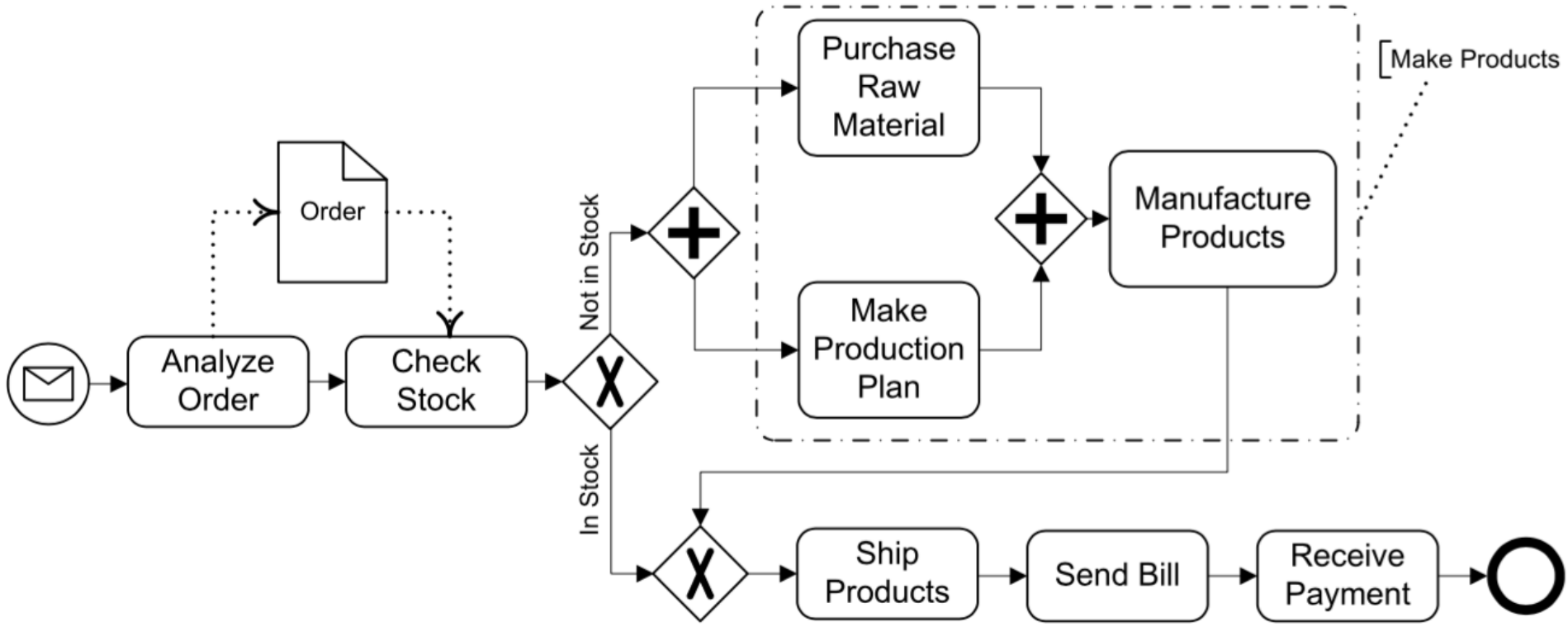
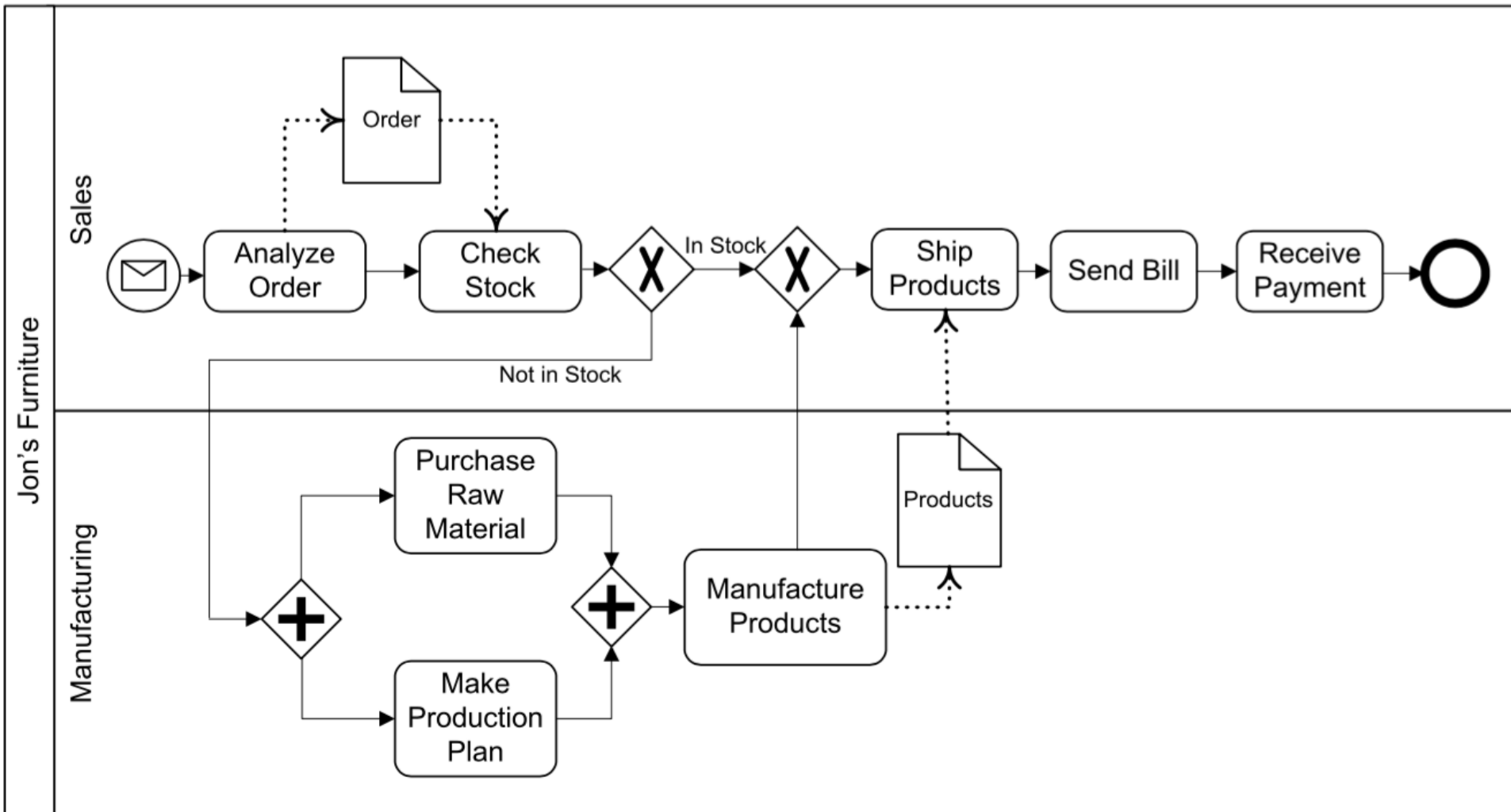


BPMN: categories of elements

**BPMN business process diagram,
representing an ordering process.**





Business process diagram with role information

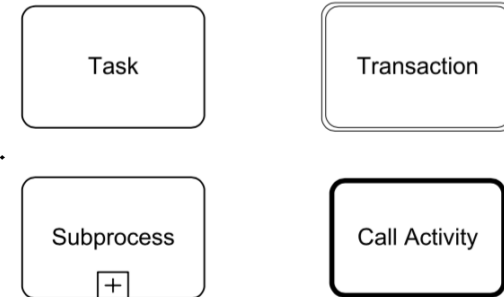
- There are two departments of the company modelled, Manufacturing and Sales.
- Receiving and analyzing the order as well as checking the stock and deciding about manufacturing the products is also decided by the sales department.
- Obviously, producing the ordered items is performed by the manufacturing department. Since hand-over between organizational entities is important, the model also contains a data object Product.
- This illustrates that also physical products can be represented by data objects in BPMN. In this case, the write edge from Manufacture Products to Products can be interpreted as the production of the physical goods.
- The read edge from Products to Ship Products refers to the use of the physical products during the shipment activity.

ACTIVITIES

Type of activity

○ Activities are units of work.

- They are the major ingredients of business processes.
- The BPMN provides powerful means for expressing different types of activities.
- Figure shows the activity types that the BPMN supports.

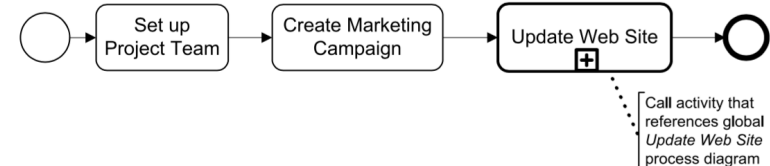


○ Activities characterize units of work.

- Activities which are not further refined are called atomic activities or tasks.
- Activities might also have an internal structure, in which case they are called subprocesses. Rather than showing the structure, the modeller can decide to hide the complexity of the subprocess, using the plus symbol. But subprocesses can also be expanded, exposing their internal structure.

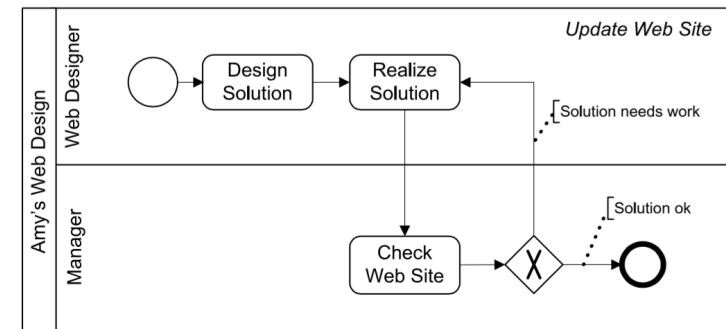
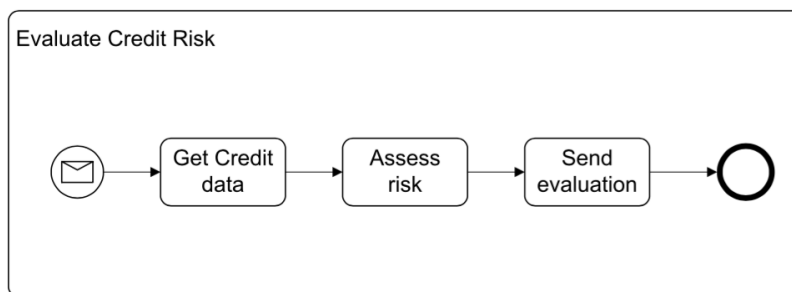
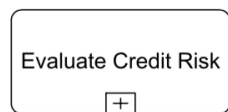
○ Call activities can be used to refer to globally defined process diagrams, or tasks, facilitating reuse of activities.

- An example of a call activity involving a globally defined process diagram is shown in Figure 4.81. In the upper part of that figure, a simple process containing a sequence of activities is shown. The first activity is an embedded subprocess with activities to set up a project team and to create a marketing campaign. After these activities have been completed, the embedded subprocess terminates. Then the Update Web Site activity is performed. This call activity references the global process diagram shown in the lower part of that figure, reusing it. This design allows to define certain processes or tasks once to be used several times. In the example, each update of the web site could be realized by a call activity, reducing maintenance effort in large process repositories.

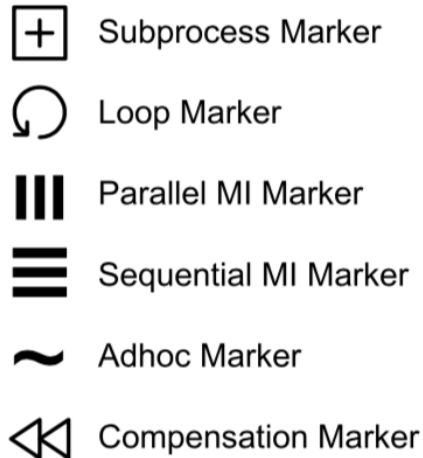


Call activity

Sub-process

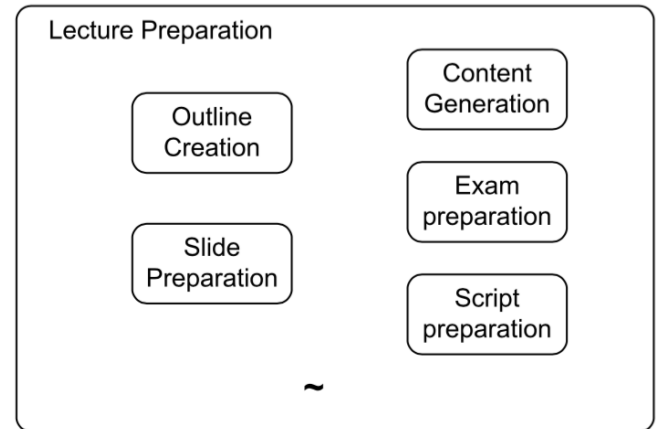


Activity Markers



Activities can be marked with symbols that refine their execution semantics; activity markers are shown in Figure 4.82.

- **Subprocess marker:** We have already used the subprocess marker. Notice that transactions will be discussed later in this section after the required events have been introduced.
- **Loop marker:** The loop marker is used to indicate that an activity is iterated during process execution. If the activity has the LoopCharacteristics attribute set, with attribute class StandardLoopCharacteristics, then the activity represents a while loop or a repeat-until loop. Whether the loop activity realizes a while loop or a repeat-until loop is guided by the testTime attribute. Setting it to Before realizes a while loop, while setting it to After realizes a repeat-until loop. The different types of loops can not be distinguished by the visual appearances of the respective loop activities in process models.
- **Multiple instance marker:** Multiple instances tasks have the LoopCharacteristics attribute set, with attribute class MultiInstanceLoopCharacteristics. The multiple instances of an activity can be executed sequentially or in parallel. The number of instances is either specified by an expression that returns an integer value or by the cardinality of a list data object, discussed below.



- **Adhoc Marker:** A subprocess that is marked with an adhoc marker consists of a set of tasks that are not related to each other by sequence flow. The execution of tasks of the adhoc subprocess is not restricted. Each activity can be executed an arbitrary number of times. This means that adhoc activities are not embedded in sequence flow; they can be invoked without a specific trigger or event.
- An adhoc subprocess is marked with a tilde symbol at the bottom of the rounded rectangle. Adhoc activities are very useful for unstructured parts of processes. Using AdHocOrdering, the modeller can define whether the activities in an adhoc subprocess can be executed in parallel or whether they are executed sequentially. An adhoc subprocess completes, if its CompletionCondition evaluates to true. An example of an adhoc subprocess that represents the preparation of a lecture is shown in Figure.

Task Types

In BPMN, tasks can be decorated with task types which makes it easier for human readers to understand the specific type the task represents. Figure lists the task types of the BPMN.

- **User tasks** represent traditional workflow tasks that involve user interaction. When the process comes to a point where a specific task is to be performed by a user, the user is informed, for instance, by the appearance of a new work item in his or her inbox.
 - When selecting the work item, an application is started that the user works with in order to perform the task. To facilitate role resolution, role and skill information are typically associated with a user task. Integration with organizational modelling is required to facilitate role resolution, because the BPMN does not support the modelling of detailed organizational aspects.
- **Manual tasks** are performed without the support of software systems. Sending a printed letter or transferring goods in a logistics environment are examples of manual activities. While the actual execution of these activities is outside the scope of an information system, the business process management system needs to be informed about the completion of a manual activity.
- The completion information typically includes a return code, so that the system is aware of a successful or unsuccessful completion of the manual task. This information can be important for the remaining parts of the business process, so that in the case of unsuccessful completion, the business process can take compensating actions for the failed manual activity.
- Business rules are logical rules to be interpreted by a rules engine. In BPMN we can model a task that triggers a business rule by marking it with a business rule marker and adding the appropriate information. When the **business rule task** task is executed, the business rule is invoked. The actual representation of business rules and their enactment using rules engines is not in the scope of the BPMN.
- A **service task** is implemented by a piece of software, either using a Web services interface or an application programming interface to a software system.
- A **script task** is a task that uses some scripting language expression in order to be performed. Script tasks are used to represent simple functionality, for which no dedicated software system is required. The particular scripting language used and the interaction platform for script expressions depend on the tool support available. When the script completes execution, the script task completes.
- There are also task types related to **sending** and **receiving** messages. Since these task types rely on events, we will introduce events first and return to send and receive tasks only when we have done so.
- A **compensation task** is invoked to compensate for activities that need to be undone. The compensation concept is strongly connected to transactions.



Send Task



Receive Task



User Task



Manual Task



Business Rule Task



Service Task












































Script Task

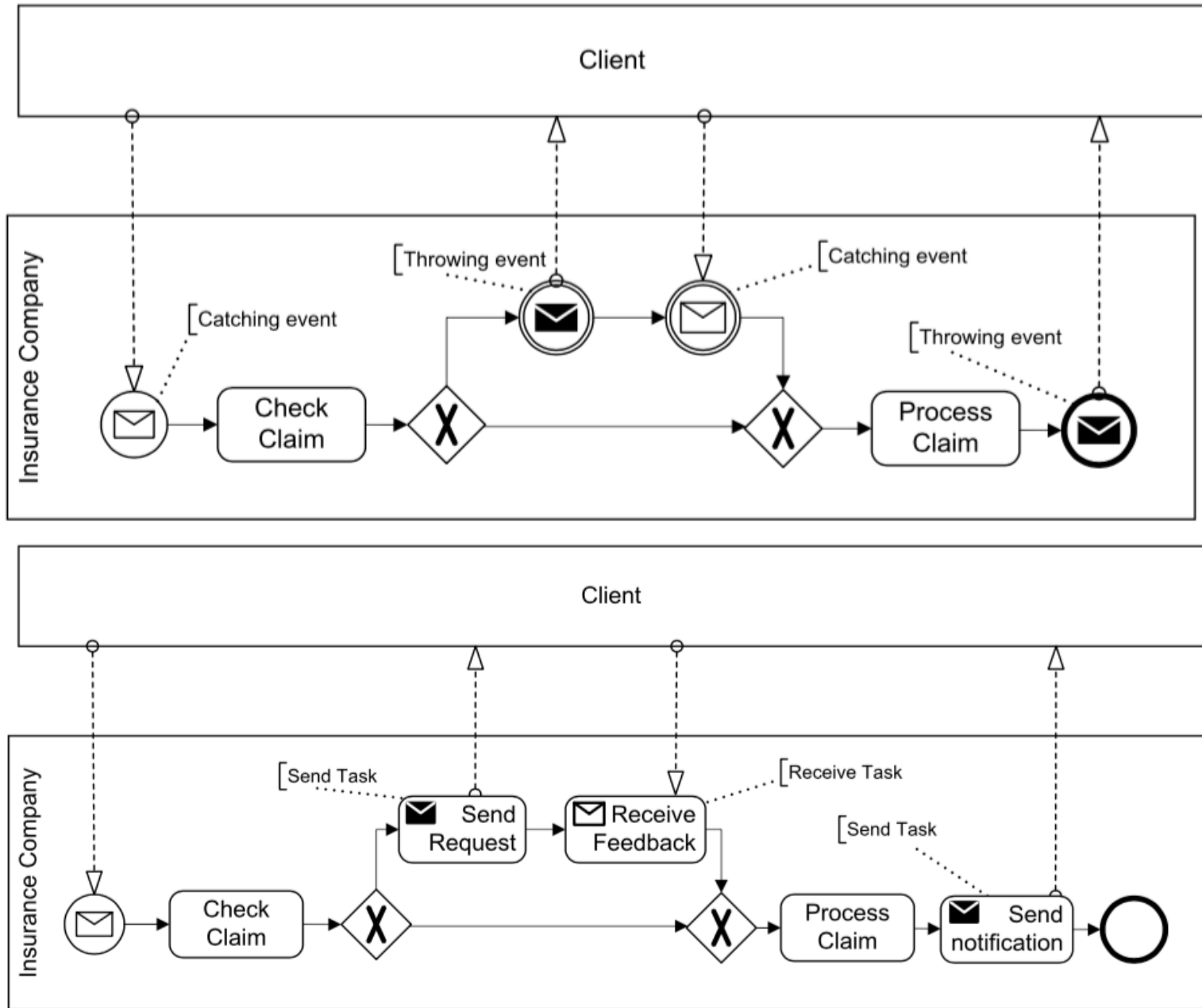
Events

- Events play a central role in business process management, since they are the glue between situations in the real world and processes that will react to these events or trigger them. Events in a business process can be partitioned into three types, based on their position in the business process:
 - **start events** are used to trigger processes, intermediate events can delay processes or they can be triggered during process executions.
 - **End events** signal the termination of processes. There are obvious connection rules associated with these events. Start events have no incoming edges, end events have no outgoing edges, and
 - **intermediate events** have both an incoming and an outgoing edge.
- This book covers the most common event types, shown in Figure. The rows contain the event types, the columns the **position** (start, intermediate, end) and the **nature** of the event, discussed shortly. There are also intermediate events that are attached to boundaries of activities rather than having an incoming sequence flow.
 - The simplest type of event is the **blanco event** that has no marker. (The standard calls this event none event. Since blanco events are in fact events, we stick to the former terminology and use the term blanco event.) This event type is used whenever the cause of the event is either not known or is irrelevant for the current modelling purpose. Blanco events can be used as start events or as end events.
- Events play two major roles, and each event in a process model plays exactly one of those. These roles are referred to by **catching** and **throwing**.
 - An event is of catching nature, if the process listens and waits for the event to happen. Whenever the respective event happens, the process catches it and reacts accordingly. All start events are catching events.
 - An event is of throwing nature, if it is actively triggered by the process during process execution. Sending a message to a business partner is an example of a throwing event. All end events are throwing events, because the end event is actively triggered by the process.
- Intermediate events can be either catching or throwing. A good example is the intermediate message event, which comes in two flavors.
 - As throwing event, the intermediate message event sends a message to a business partner.
 - As catching event, the process waits for a message to come in, that is, it waits for the event to happen.

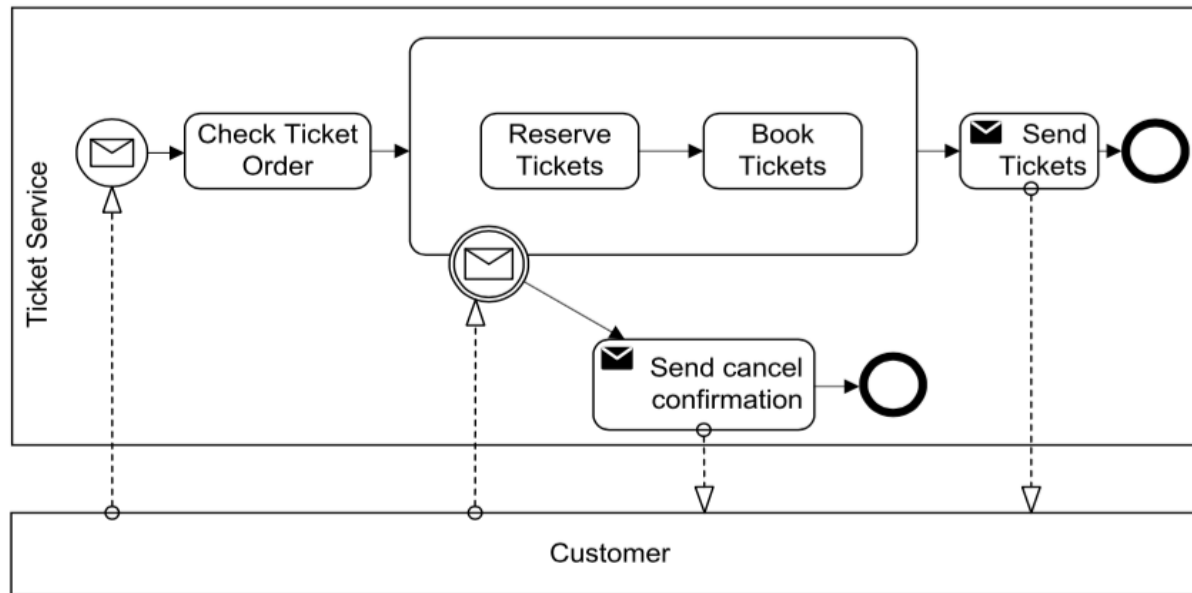
Position/Nature (type) of Events

	Start Events		Intermediate Events		End Events
	Catching	Catching	Boundary Interrupting, Catching	Boundary Non-Interrupting, Catching	Throwing
None or blanco: Untyped events, indicate start point, state changes or final states.					
Message: Receiving and sending messages.					
Timer: Cyclic timer events, points in time, time spans or timeouts.					
Escalation: Escalating to an higher level of responsibility.					
Conditional: Reacting to changed business conditions or integrating business rules.					
Link: Off-page connectors. Two corresponding link events equal a sequence flow.					
Error: Catching or throwing named errors.					
Cancel: Reacting to cancelled transactions or triggering cancellation.					
Compensation: Handling or triggering compensation.					
Signal: Signalling across different processes. A signal thrown can be caught multiple times.					
Multiple: Catching one out of a set of events. Throwing all events defined.					
Parallel Multiple: Catching all out of a set of parallel events.					
Terminate: Triggering the immediate termination of a process.					

Catching/Throwing Events vs. Send/Receive Activities



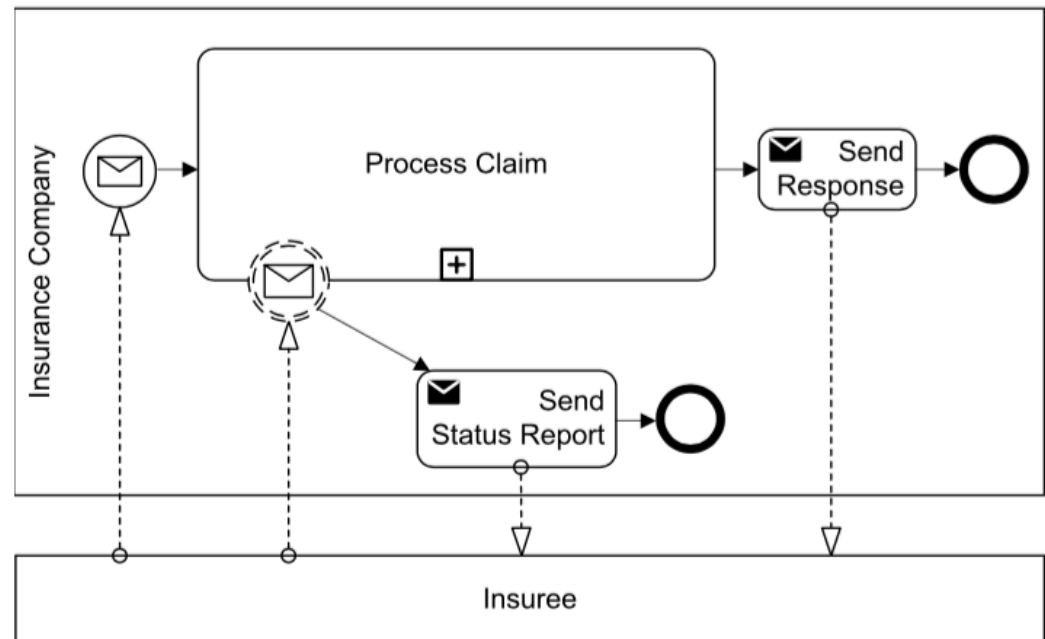
Interrupting/Non-interrupting Boundary Events



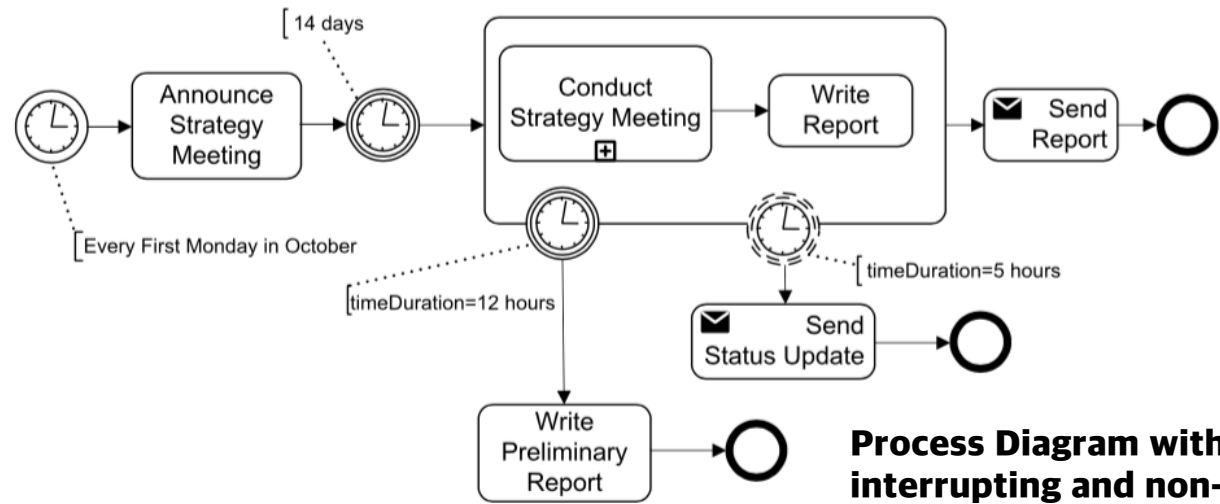
- The process starts by a customer sending a ticket order to a ticket service. After receiving the message and checking the order, a subprocess is entered. In the subprocess, the tickets are reserved and finally booked. Notice that processes and subprocesses do not require start and end events.
- While it is good practice to use start and end events on the process level, they might be dropped for simple subprocesses. The boundary event represents the option of the customer to cancel the order.
- If the cancellation message is received while the subprocess is still active, the subprocess is cancelled, and the confirmation of the cancellation is sent. If no cancellation message is received while the subprocess runs, the subprocess completes and the tickets are sent.
- In this example we have discussed a **boundary event that interrupts the subprocess** it is attached to.

An example of a non-interrupting boundary event is shown in Figure.

- In this process, an insuree sends a claim report to an insurance company. The complex processing of the claim is hidden in the subprocess Process Claim. While this subprocess is active, the insuree can ask, even multiple times, for the current status of the claim handling.
- The respective incoming message sent is caught by the boundary event, and a status report is sent. Since the processing of the claim should not be interrupted by this request, **the boundary event is non-interrupting, indicated by its dashed outline.**
- Timer events are used frequently in process diagrams. They are quite versatile, since they can represent time intervals, points in time, and timers, similar to count down watches.



Several Types of Timer Events

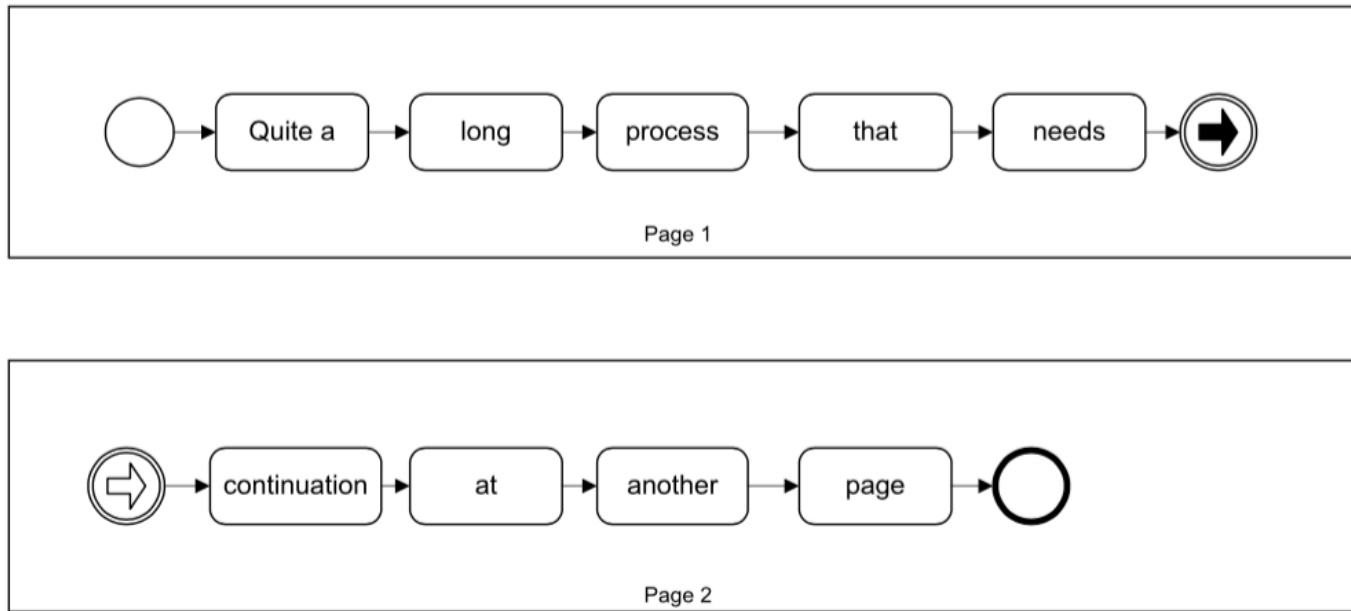


Process Diagram with interrupting and non-interrupting boundary timer events

Figure shows a process diagram involving several types of timer events.

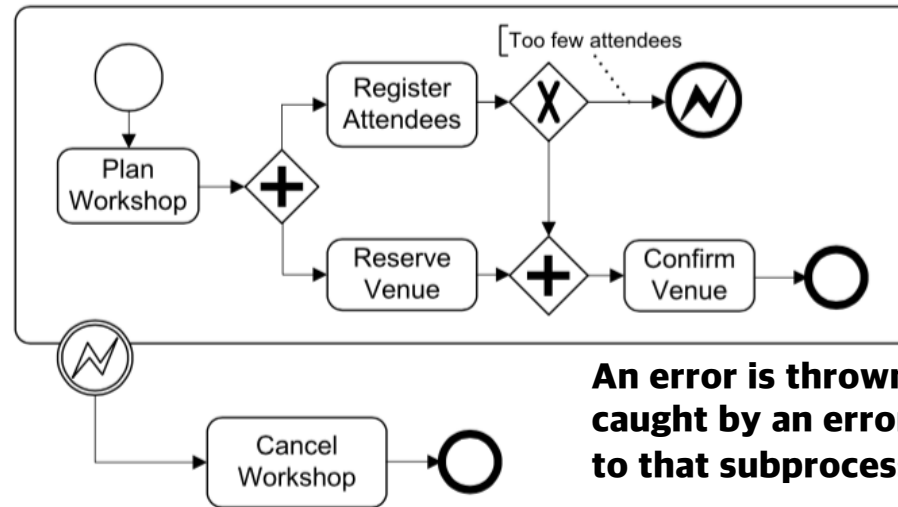
- The process starts with a start timer event. By the annotation we learn that the process is instantiated every first Monday in October. When this event is caught, a strategy meeting is announced. Then the process pauses for 14 days, represented by the intermediate timer event.
- The execution semantics of intermediate timer events is as follows. When the previous activity is completed, the timer is started. This is like starting a count down watch with the value set to 14 days in this case. In Figure, we used an annotation to show the time period.
- In BPMN, timer events have attributes that are used to represent timer values in a structured fashion. In particular,
 - the attribute `timeDuration` holds an expression that defines the time duration the timer waits for.
 - Attributes `timeDate` and `timeCycle` are used to specify points in time and recurring timers, respectively.
- After the duration has elapsed, the subprocess and thereby the strategy meeting can be started. When this happens, two additional timers are started for the boundary events.
 - Timer 1 with duration 12 hours for the interrupting timer event and
 - Timer 2 with duration 5 hours for the non-interrupting timer event.
- Assuming the subprocess is still active after 5 hours, Timer 2 is triggered, and a status update is sent. This event does not interrupt the subprocess. After being triggered, this timer is immediately reset, so that it can trigger the sending of the next status update after another 5 hours, if the meeting is still ongoing at that time.
- This example shows that non-interrupting boundary events can occur multiple times while the subprocess is active. If the subprocess is not completed after 12 hours, Timer 1 elapses, and the subprocess is interrupted. A preliminary report is written, and the process terminates.

Link Events



- **Link events** are quite specific, since they—unlike all other events—do not represent something that happens in the real world.
- Rather, they are a means to layout large process diagrams that span multiple pages or screens. A part of the process ends with a link event of throwing nature, while the next part of the process starts with a link event of catching nature.
- Consequently, link events are intermediate events, even though they have no outgoing (throwing link event) or no incoming edge (catching link event).
- Regarding the execution semantics, two matching link events are equivalent to a sequence flow. It is important to stress that link events do not connect multiple processes, but just parts of one process. Link events are illustrated in Figure.

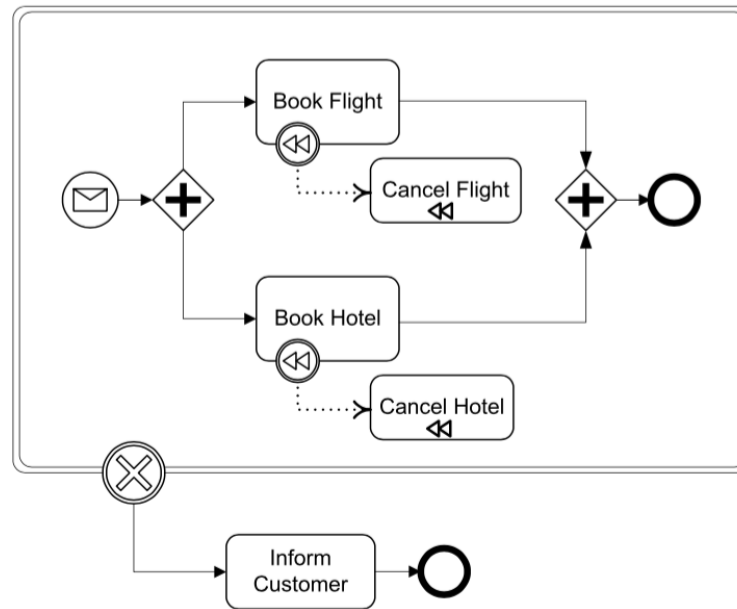
Error Events



An error is thrown in a subprocess; it is caught by an error boundary event attached to that subprocess

- **Error events** also come in two flavours.
- A throwing error event indicates the occurrence of an error in a certain scope, for instance, in a subprocess. Catching error events are always on the boundaries of subprocesses. They catch the error and interrupt the subprocess, in case certain parts of the subprocess are still active. After catching an error, typically error handling activities are performed.
- An example involving error events is shown in Figure. A subprocess for planning a workshop consists of planning the workshop followed by concurrent activities involving the registration of attendees and reserving a venue. If too few attendees register for the workshop, an error is thrown, and the workshop has to be cancelled. The cancellation is facilitated by a boundary error event that catches the occurrence of the error within the subprocess.
- In this example, error handling is done by the Cancel Workshop activity.
- Notice that once the error event is thrown, any running activity in the subprocess will be interrupted. In the example, Reserve Venue might be still running at that point in time. If this is the case, it is interrupted immediately, since no venue needs to be reserved when the workshop is cancelled.

Compensation Events



Business process diagram with transaction and compensation elements, adapted from Object Management Group (2011)

Compensation events are strongly connected to transactions. Transactions are specific subprocesses, whose activities have transaction semantics, specified by a transaction protocol. Probably the most widely used transaction protocol is the ACID model, which states that transactions have the following ACID properties:

- **Atomicity:** Either all activities in a transaction are executed successfully or none is.
- **Consistency:** The correct execution of a transaction brings the system from one consistent state in another consistent state.
- **Isolation:** The activities of a transaction are executed in isolation from other transactions, that is, transactions do not interfere with each other.
- **Durability:** Effects of transactions survive any system failure that might occur at a later point in time.

Assuming the ACID transaction model, all transactions need to obey the atomicity property: Either all activities of the transaction need to be successfully completed, or none at all. In database systems, this all-or-nothing property of transactions is typically implemented by locking protocols or multiversion concurrency control schemes.

- In business processes, the situation is a bit more complex, since we cannot lock large parts of business processes for an extended period of time or create multiple versions of the same data object. In business process management, the typical assumption is that each activity is executed in an atomic fashion. This means, however, that one activity of a transaction can have completed already, when another activity decides to fail. In this case, the first activity needs to be undone, using compensation.
- An example is used to illustrate this concept.
- The cancellation boundary event catches the unsuccessful completion of the transaction. It can be used to execute activities after the transaction has unsuccessfully completed, like informing the customer in the example.
- Signal events communicate certain situations to a wide audience. For each signal throw event, there can be several events that catch the signal. Similar to a flare that can be seen in a wide perimeter, a signal can be caught by different parts of the same process, by other processes within the same process diagram, and even by other process diagrams. Signals are similar to the event publication / event subscription mechanism in distributed computing, where a given signal event can have many subscribers throughout the process landscape.

Sequence Flow and Gateways

- In the BPMN, control flow is called sequence flow. Sequence flow is represented by solid arrows between flow objects, that is, activities, events, and gateways. BPMN supports several types of sequence flow, including **normal flow**, **conditional flow**, **default flow**, and **exception flow**.
 - The normal flow of a business process represents expected and desired behaviour of the process. It begins in the start event of a process diagram and continues via a set of flow objects until it reaches an end event.
 - Exceptional situations are represented by exception flow. With respect to process execution semantics, there is no difference between normal flow and exception flow. The only difference is that exception flow does not define the desired flow of the process, but exceptional situations. Exception flow is created by intermediate events attached to the boundary of an activity, as discussed above in the context of boundary events.
 - There are two additional types of sequence flow, namely conditional flow and default flow. Since these play important roles in the context of gateways, we introduce gateways first.
- In BPMN, each **gateway** acts as a **join node** or as a **split node**.
 - Join nodes have at least two incoming edges and exactly one outgoing edge.
 - Split nodes have exactly one incoming edge and at least two outgoing edges.
 - We can also express gateways with multiple incoming and multiple outgoing edges in BPMN. These gateways are called **mixed gateways**. Since two behaviours—split and join—are expressed by a single concept (for example, exclusive or), best practice is not to use mixed gateways but to use a sequence of two gateways with the respective split and join behaviour instead.

Gateway Types



Exclusive Gateway



Event-based Gateway



Exclusive Gateway
(alternative)



Complex Gateway



Parallel Gateway



Parallel Event-based
Gateway (instantiate)

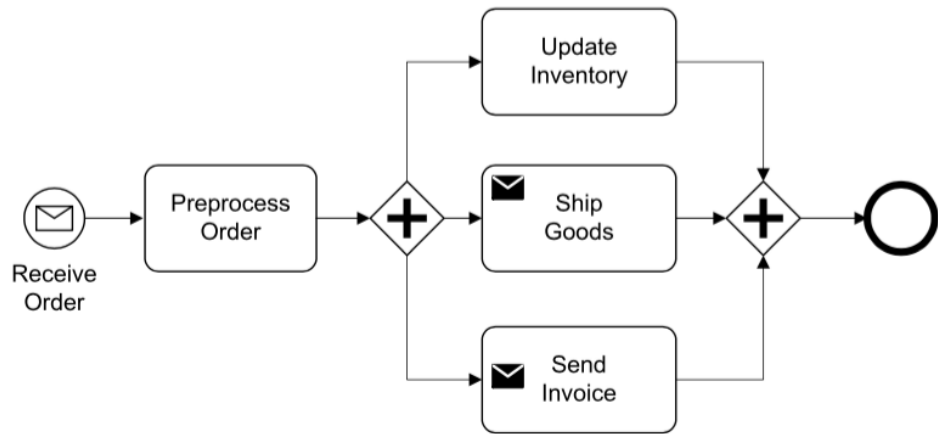


Inclusive Gateway

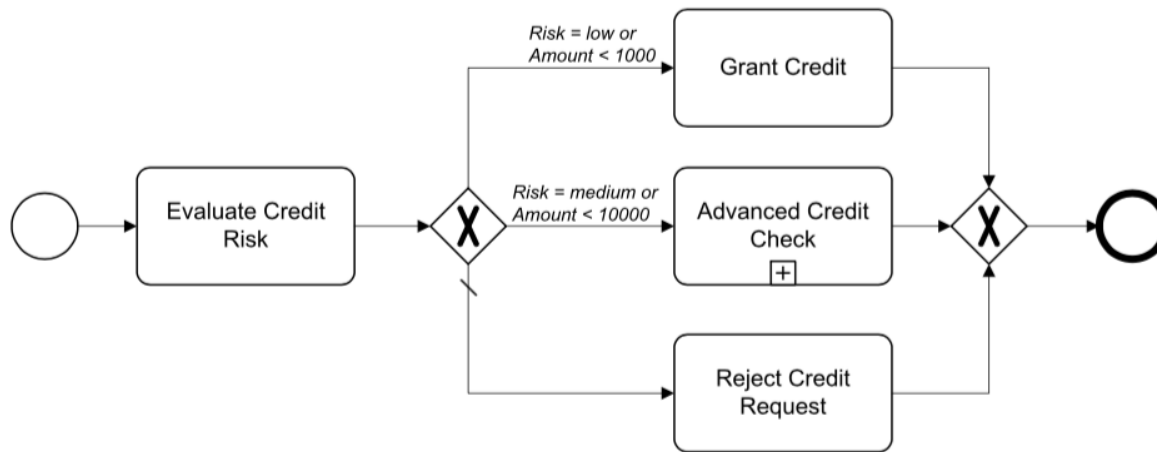


Exclusive Event-based
Gateway (instantiate)

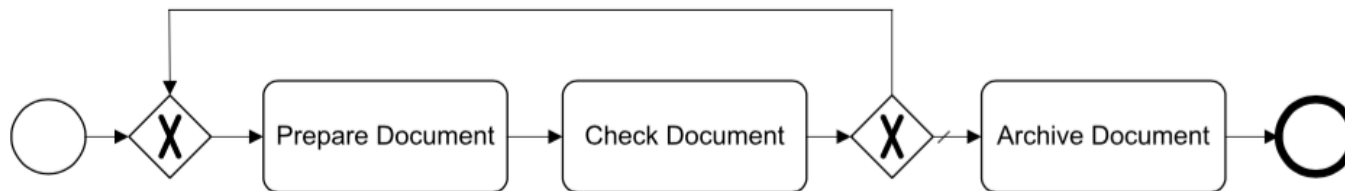
- The gateway types of the BPMN are shown in Figure, that is, the exclusive, the parallel, the inclusive, the event-based, the complex and two instantiation gateway types.
- As shown in that figure, there are two representations of the exclusive gateway, one without a marker and one with a marker.
 - This feature of an unmarked gateway in the BPMN can be considered inconsistent with the definition of an unmarked, that is, blanco event.
 - The blanco gateway actually represents a particular kind of gateway, while the blanco event does not.
 - To avoid misunderstandings, we recommend to always use gateway markers.



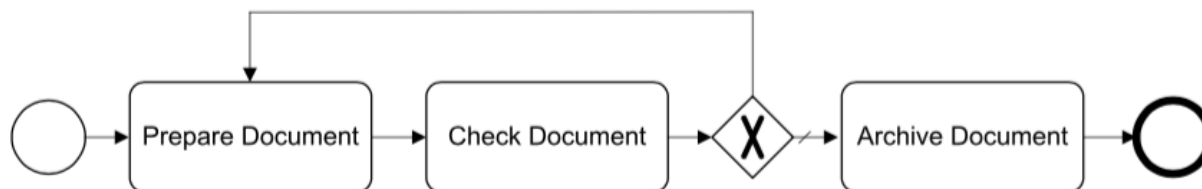
Business process diagram with the parallel gateway



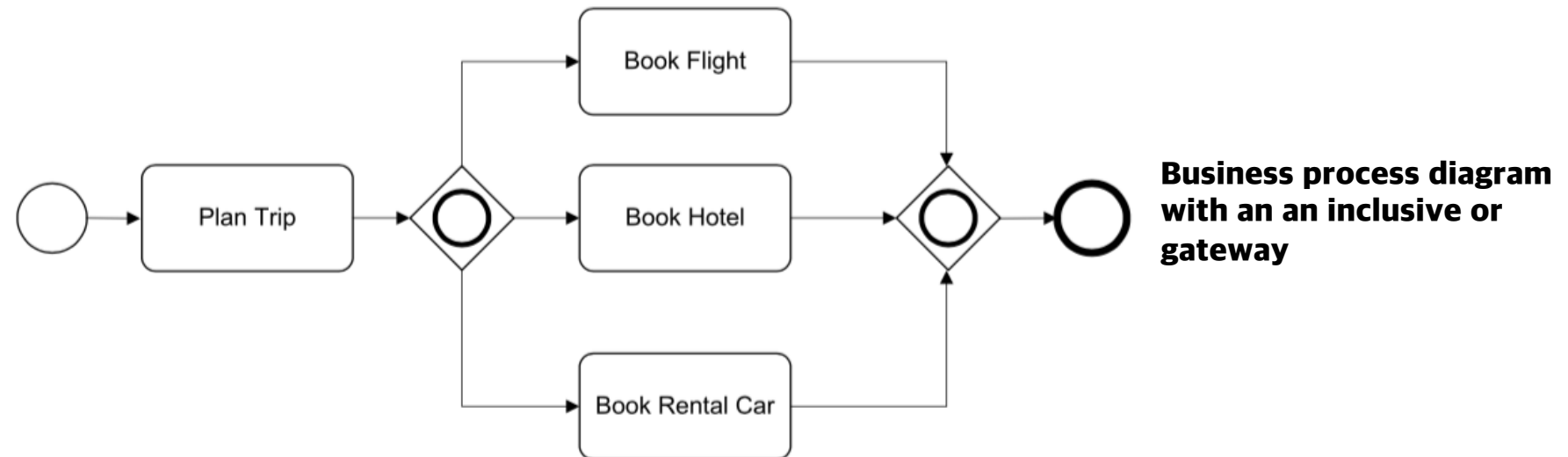
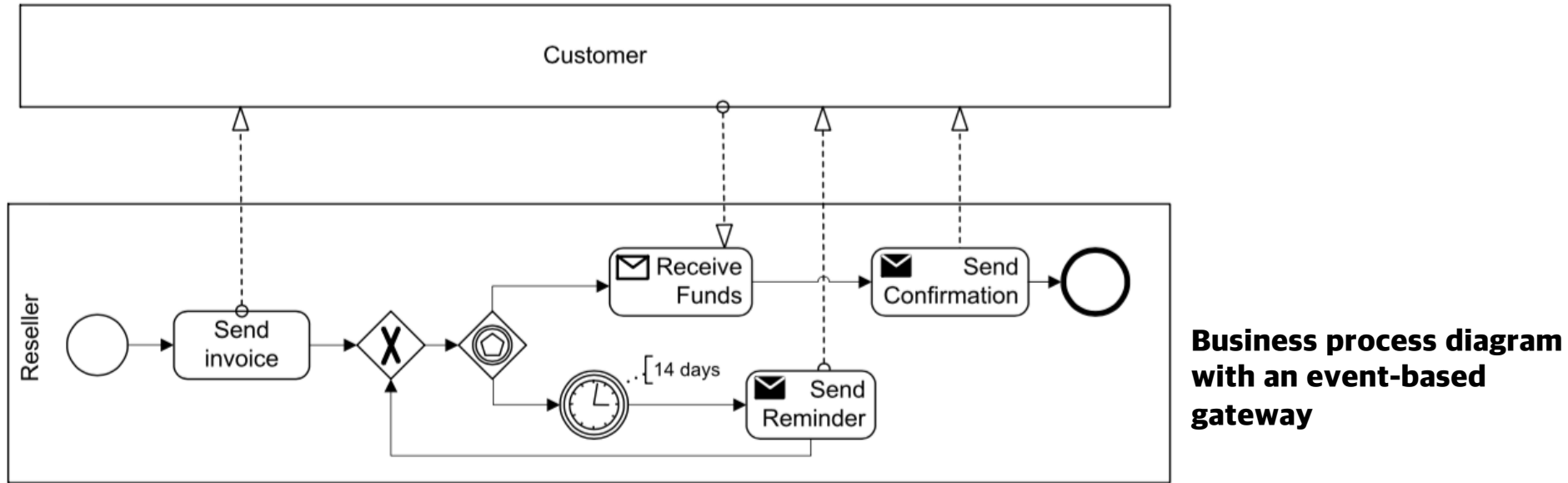
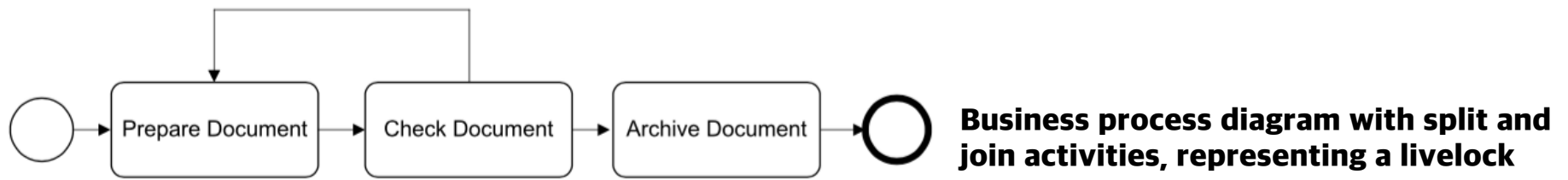
Business process diagram with the Exclusive gateway with conditions and default flow



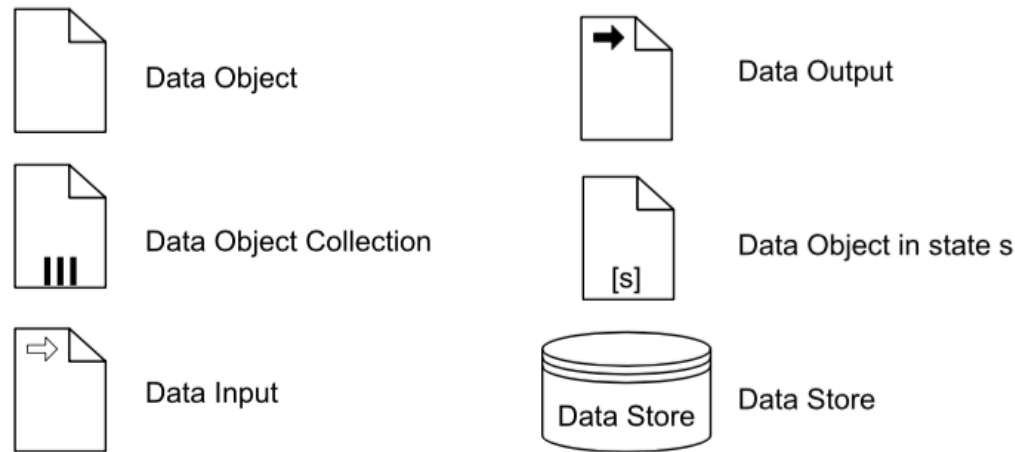
Business process diagram with the Exclusive gateways realizing a loop



Business process diagram with uncontrolled flow

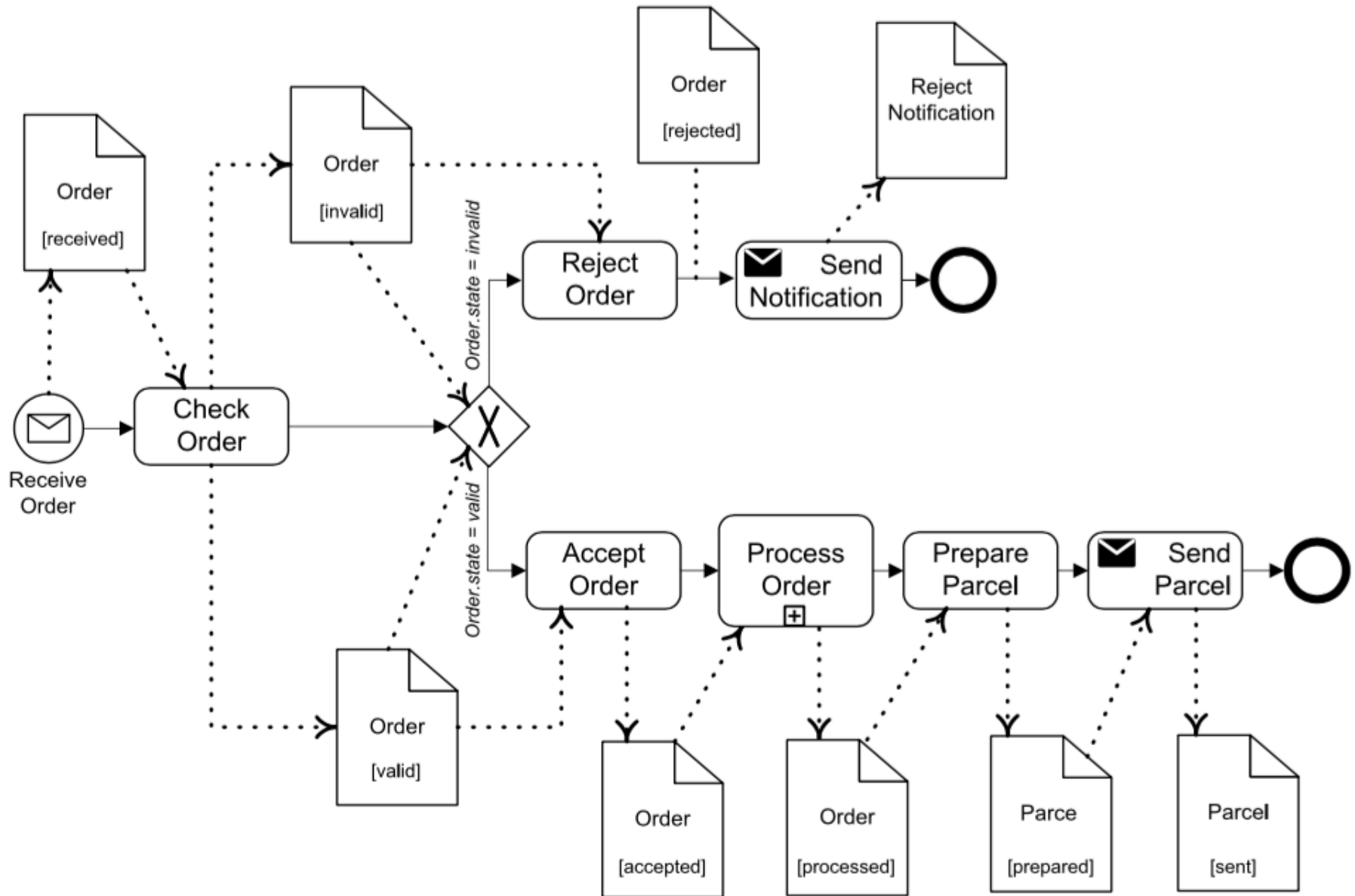


Handling Data

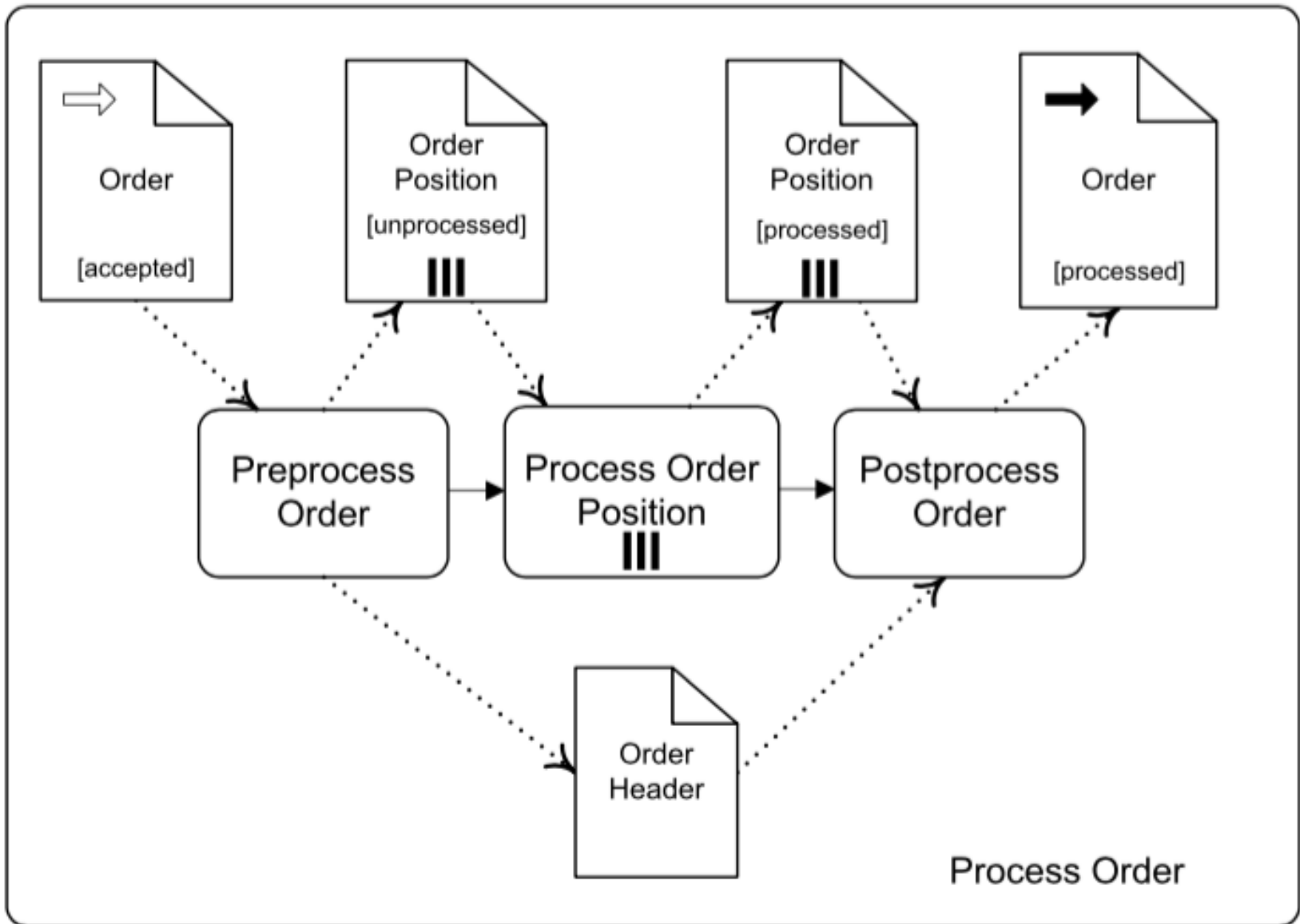


- All business processes deal with information or physical artefacts. To represent information and physical artefacts, BPMN provides data objects. While the term data object seems to indicate digitalized information, it also covers physical objects, such as documents and products.
- The notational symbols regarding data in BPMN are shown in Figure. Often, data objects represent digitalized objects, such as orders in an information system. Since BPMN concentrates on process modelling, there are no data modelling capabilities available. This would also not be appropriate, since the UML provides excellent data and object modelling capabilities, for example, class diagrams.
- The relationships between data objects and activities or, more generally, flow objects are specified by data associations. A directed edge from an activity to a data object means that the activity creates or writes the data object. Directed edges in opposite direction indicate read relationships.
- Typically processes use data that has been created before the process has started. Examples of this type of data is customer information stored in a customer relationship management system or production information stored in a database. To represent that a process uses these types of data, input data objects can be used. Analogously, if data objects are created as output to be used by other processes, these are marked with a data output marker, as shown in Figure.

Business process diagram involving data objects



Business process diagram of the Process Order subprocess from the Figure, involving data object collections



Business process diagram involving multiple input sets of an activity

