Architetture Software a Microservizi

Corso di Laurea Magistrale in Informatica

BPMN

Davide Rossi

Dipartimento di Informatica – Scienze e Ingegneria

Università di Bologna

BPMN

Business Process Model and Notation is an **OMG** specification.

BPMN has a MOF-based metamodel and an executable semantics. An XMI-based standard representation is also provided.

BPMN target

BPMN is targeted at a high level for **business users** and at a lower level for **process implementers**. The business users should be able to easily read and understand a BPMN business process diagram. The process implementer should be able to adorn a business process diagram with further detail in order to represent the process in a physical implementation.

BPMN is targeted at users, vendors and service providers that need to communicate business processes in a standard manner.

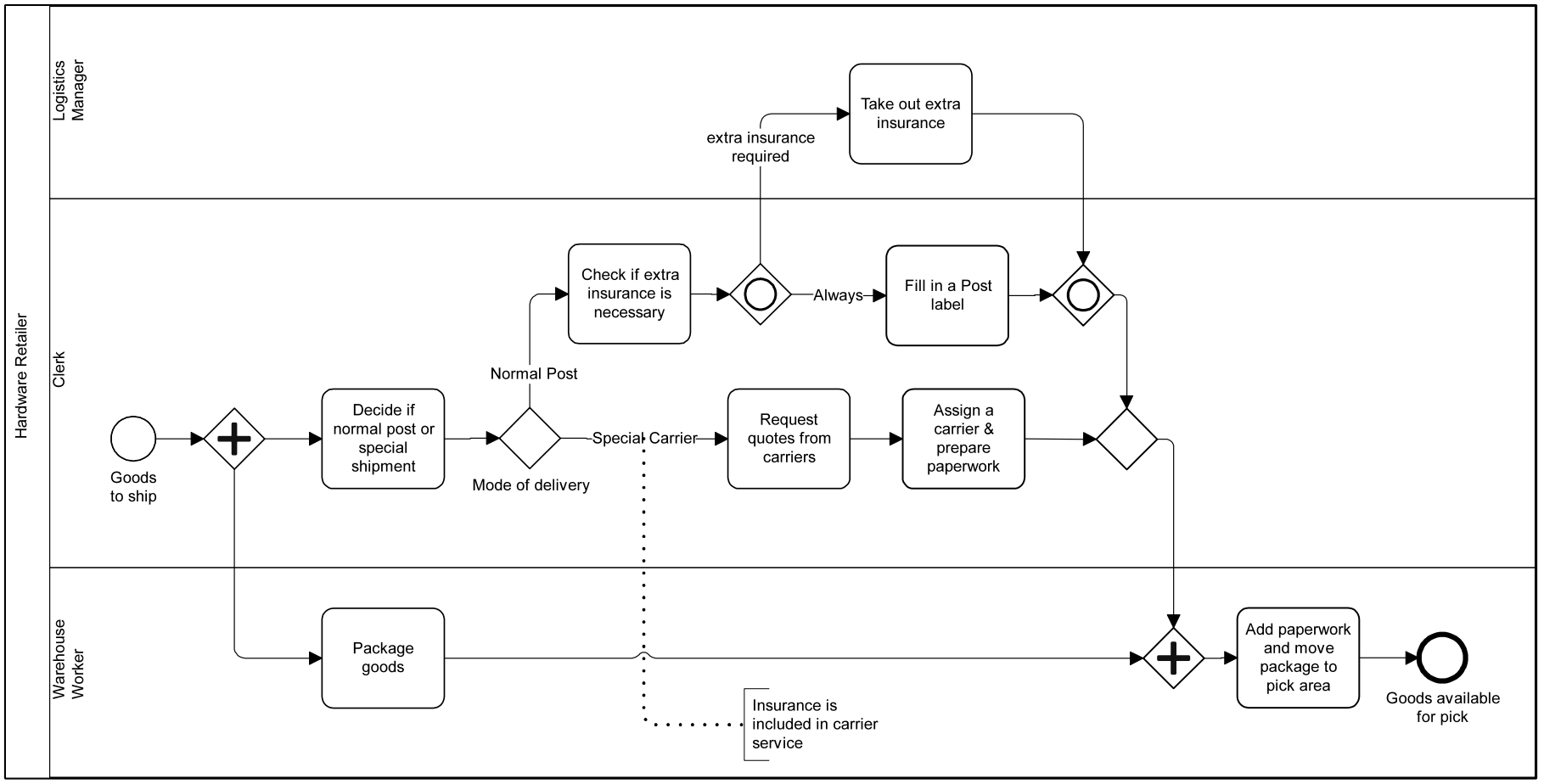
Diagram types

- **Process/Collaboration**
  Process describes a sequence or flow of Activities in an organization with the objective of carrying out work. Collaboration also includes pools and message flows to describe the interactions between two or more organizations.
- **Conversation**
  is a simplified version of Collaboration.
- **Choreography**
  is a type of business contract between two or more organizations.

BPMN process diagram

Depicts the end to end flow of a business process. The diagram represents a network of flow objects, which are a set of activities, and connecting objects that sequence them.

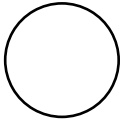
Shipment process example



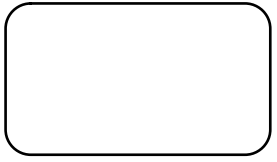
Flow objects

Flow objects are the main objects expressing the semantics of a process model

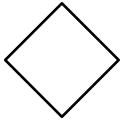
Events:



Activities:



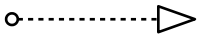
Gateways:



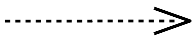
Connecting objects

Connecting objects are used to depict how flow objects interacts

A **sequence flow** is used to show the causal dependencies between the activities that will be performed in a Process.



A **message flow** is used to show the flow of messages between two entities that are prepared to send and receive them.

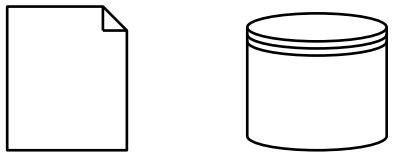


An **association** is used to associate data, information and artifacts with flow objects.

Artifacts

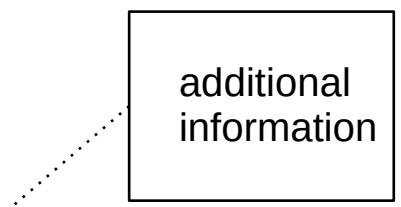
Artifacts are used to provide additional information about the process.

Data Objects:



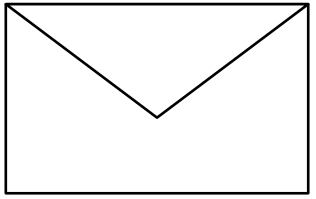
Groups:

Annotations:

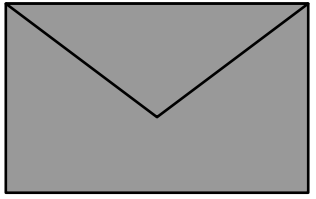


Messages

A Message represents the content of a communication between two Participants. It is is a graphical decorator.



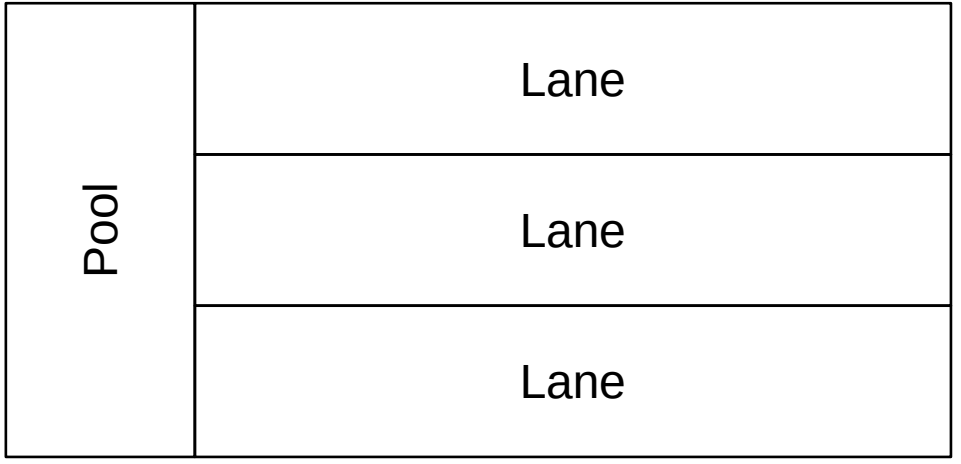
Message



Non-initiating message

Pools and lanes

In collaborations, pools and (swim)lanes are used to group the primary modeling elements.



Pools

Pools represent Participants in an choreography

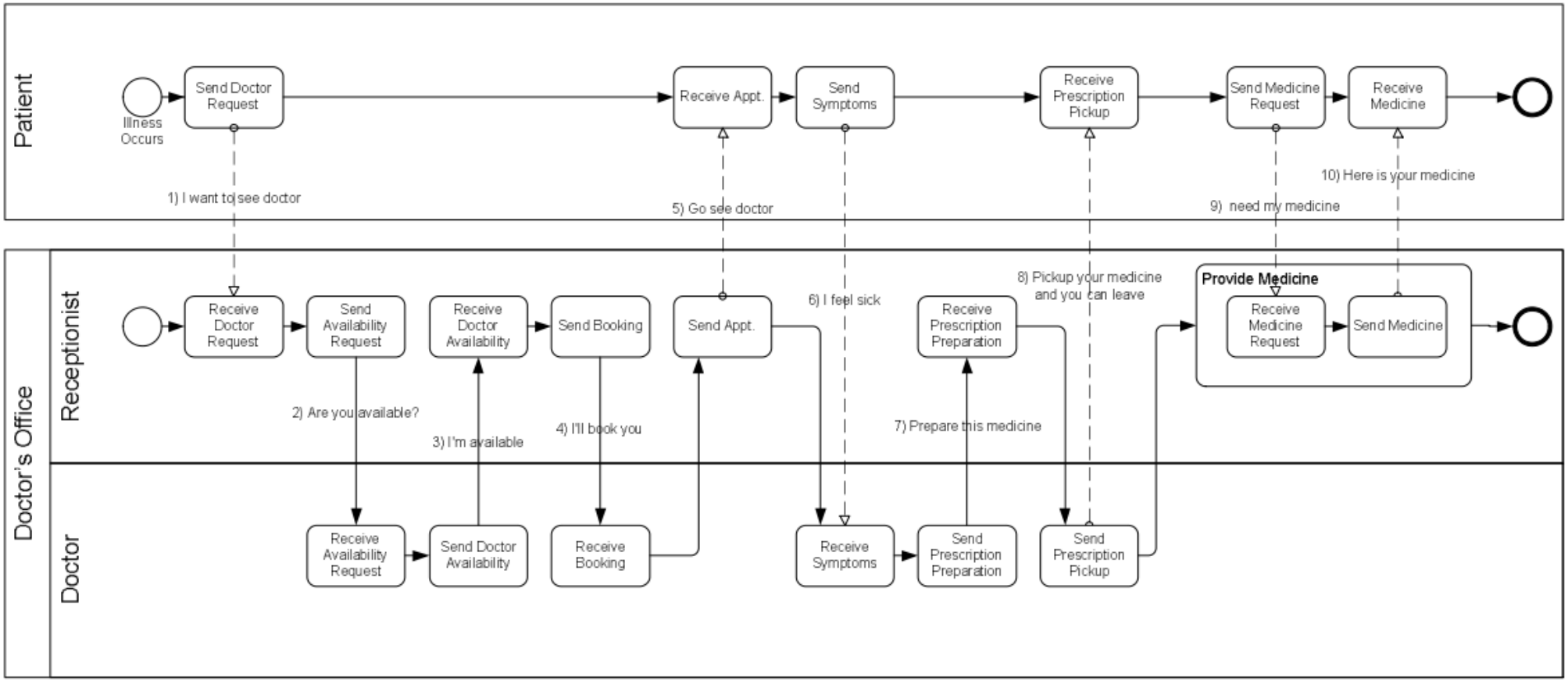
A Participant may be a business role (e.g., “buyer” o “seller”) or may a business entity (e.g., “IBM” or “OMG”).

A Pool may be a “black box” or may contain a Process. **Interaction between Pools is handled through message flows**. Sequence flows cannot cross the boundary of a Pool (i.e., a process is fully contained within a Pool).

Swimlanes

Swimlanes can be used to organize activities. A particular swimlane presents the tasks relevant to a specific business unit for instance.

BPMN collaboration



Activities

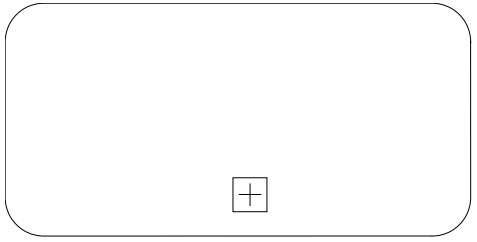
An activity is a unit of work to be performed. It might be a **task**, a **process** or a **sub-process**.

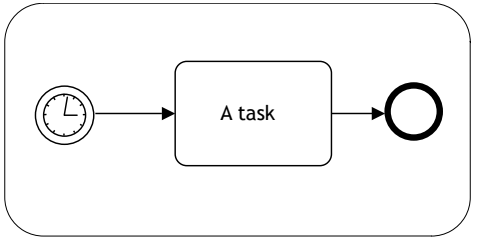
BPMN defines two main kinds of activities:

- A Task is an atomic activity that is included within a Process

Activities

- A Subprocess is a Process that is included within another Process

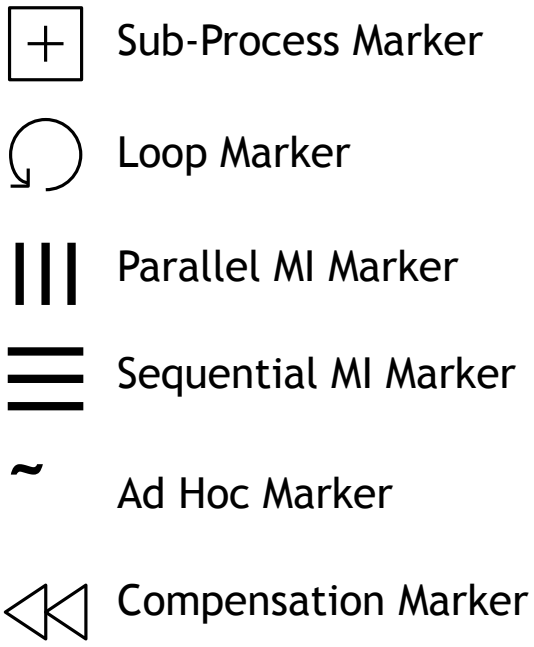




- An event subprocess is triggered by a subprocess event

Activity Markers

Markers are defined to specify additional semantics such as loops.



Task types

Types specify the nature of the action to be performed.


Send Task


Receive Task


User Task


Manual Task


Business Rule Task


Service Task


Script Task

Gateways

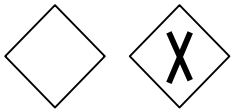
Gateways are used to control how sequence flows interact as they converge or diverge within a process.

Decisions, such as forks, merges and joins in the process flow are modeled with a Gateways.

The behavior of each type of Gateways will determine how many of the Gates will be available for the continuation of flow.

Gateways

Exclusive Gateway



Parallel Gateway

Event-based Gateway

Inclusive Gateway

Complex Gateway

Exclusive Event-based Gateway (instantiate)

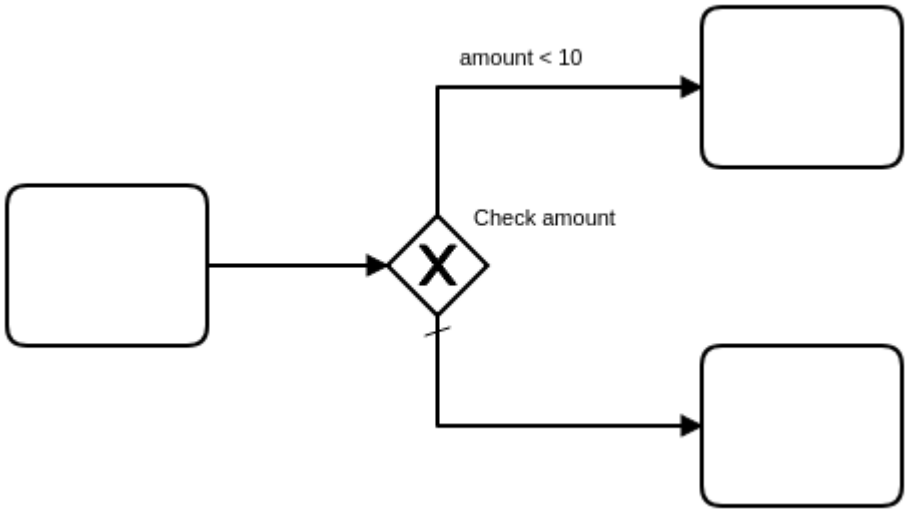
Parallel Event-based Gateway (instantiate)

Exclusive data-based

An exclusive data-based gateway can be used:

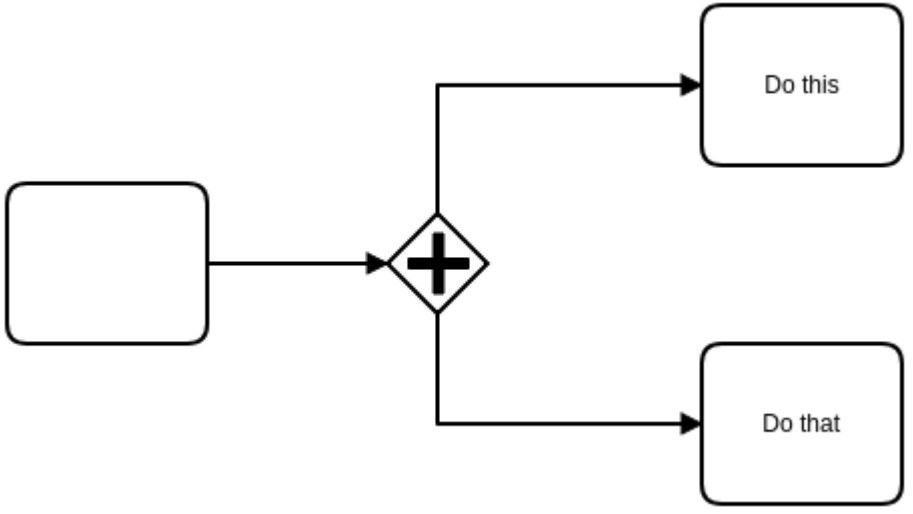
- As a decision point where several outgoing sequence flows are possible, yet they are all constrained by a condition and only one of them will be used. Such a condition will be evaluated based on the process data.
- As a merge, the exclusive gateway routes the upcoming flows into a single outgoing path.

Exclusive data-based



Parallel

Parallel gateways provide a mechanism to fork and synchronize (join) flows.

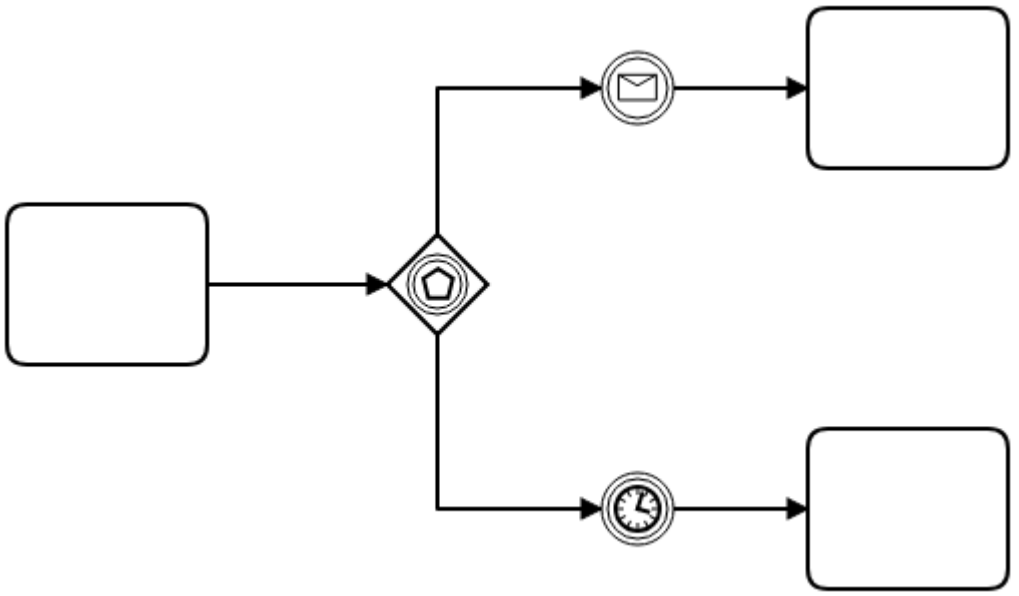


Exclusive Event-Based

An exclusive event-based gateway is structurally similar to an exclusive data-based gateway (but only used as decision point).

The event based gateway will start a race between the different events the process might receive, the first one to be received wins the race and that determines which outgoing sequence flow should be used.

Exclusive Event-Based



Inclusive

An inclusive gateway can be used

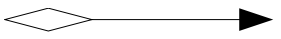
- As a decision point where several outgoing sequence flows are possible, they are all constrained by conditions, each outgoing sequence flow with a condition evaluated as being true will be followed. It might effectively spawn several execution points.
- As a merge, the inclusive gateway will synchronize all the execution points produced upstream but at most one for each incoming sequence flow.

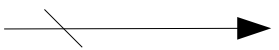
Complex

The complex gateway was created to address complex cases which would require the combination of several other gateways. To avoid this, the behavior of the complex gateway can be scripted using an expression language. As a result the complex gateway can be used to handle every situations. Its use prevents the process from being *executable*.

Conditional flows

Conditional flow and default flow are special versions of sequence flows used to represent conditional flow control.



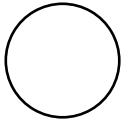
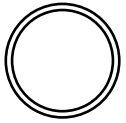
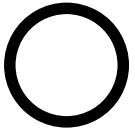


Events

An event is something that happens during the course of a business process and affects its execution flow.

An event has a cause and an impact.

BPMN defines three kinds of events:

- Start events 
- Intermediate events 
- End events 

Start events

The start event indicates where a process starts.

The start event starts the flow of the process (no sequence flow can connect to a start event).

A start event is optional: if a start event is not used, activities with no incoming sequence flow are considered as connected to an implicit start event.

Start events might be used to:

- Indicate how a message reception will trigger a process instance (e.g. Order Reception)
- Show when a process instance shall be triggered (e.g. End of Month)

Intermediate events

An intermediate event indicates where something happens during the execution of a process.

An intermediate event affects the flow of the process.

Intermediate events might be used to:

- Indicate where a message might be received
- Show where delays are expected
- Disrupt the normal flow through exception handling
- ...

End events

An end event ends the flow of the process.

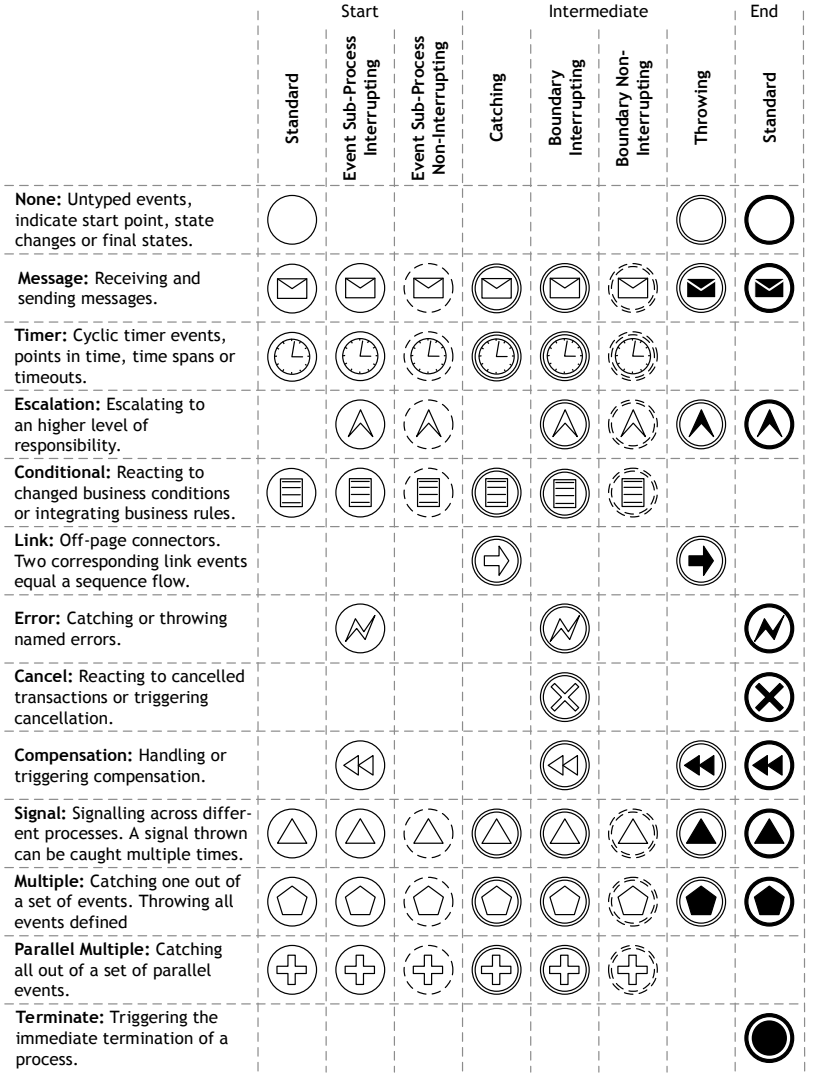
An end event will not have any outgoing Sequence Flow.

End events are optional.

End events might be used to:

- End a process flow and send a message
- End a process flow and raise an error
- End a process flow and request for a compensation
- ...

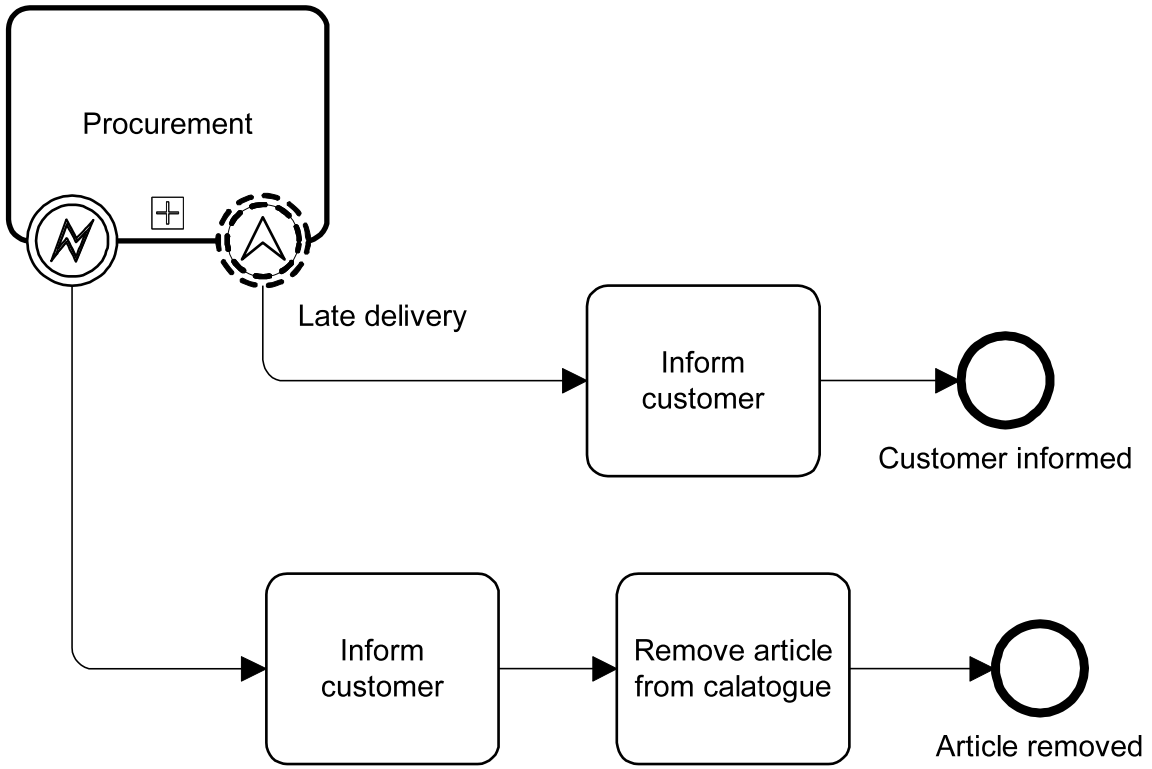
BPMN 2.0 events



Boundary events

- Are attached to activities and used to represent exception or compensation handling
- Can be interrupting or non-interrupting
- Interrupting boundary events work similarly to exceptions in most programming language
- Non-interrupting boundary events generate additional sequence flows

Boundary example



Semantics: termination

A process terminates when all tokens representing concurrent flows reach an end event.

A **terminate end event** can be used to terminate a process as soon as one token reaches it.

Data in BPMN

- BPMN assumes an hierachical map data structure (process instance/subprocess instance/sub-sub...)
- Conditions (in gateways or in sequence flows) can use predicates operating on this structure

Handling exceptions

Proceed as with use cases: first focus on the regular (“happy”) path then start handling exceptions.

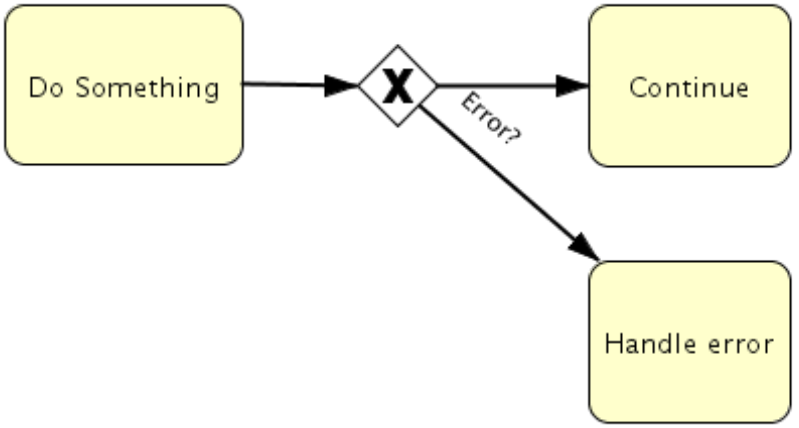
Exceptions belong to different categories, each handled with different policies.

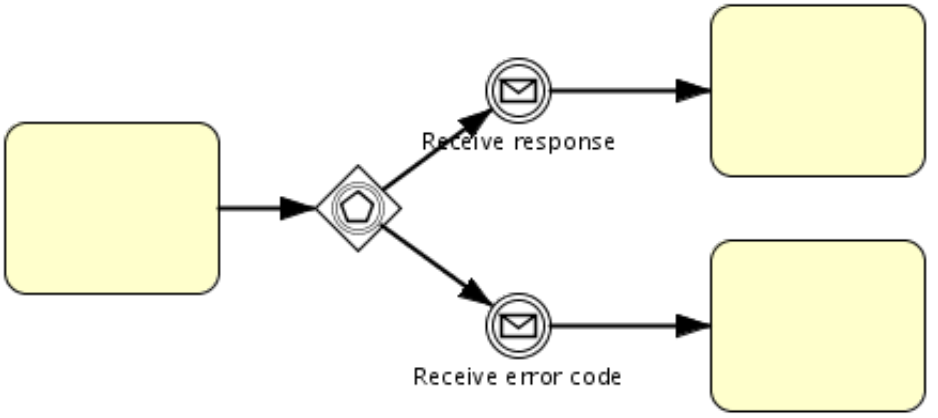
Exceptions categories

- **Business exception** (in-band error notifications from humans or software components)
- **Partner exception** (out-of-band notifications from humans or software components)
- **Timeout**
- **System error**

Business exceptions

Use an exclusive (single message, different codes) or an event-based (different messages) gateway in the process.

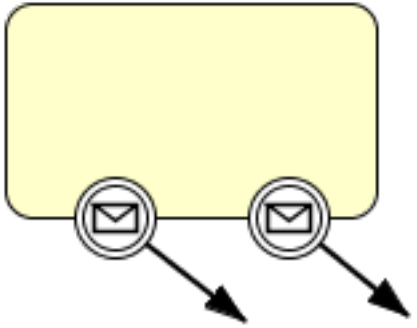


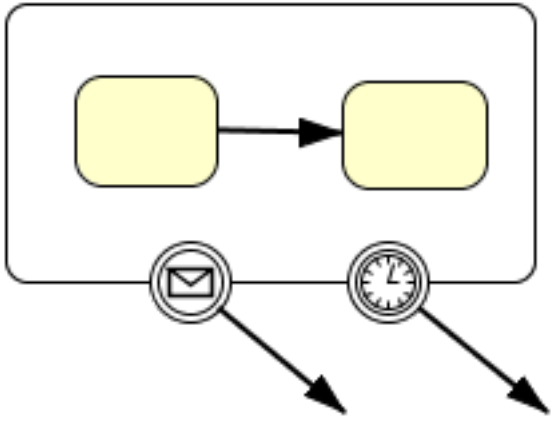


Partner exceptions

Use boundary events.

Multiple events can be attached to the same activity/subprocess.





Timeout exceptions

Handle like external exceptions (think about the timer service as an external component).

If you need to set up a timeout for a sequence of activities wrap them in a subprocess.

System errors

Put them in the model only if you want to express specific handling strategies.

In the case, use boundary events.

Transactions

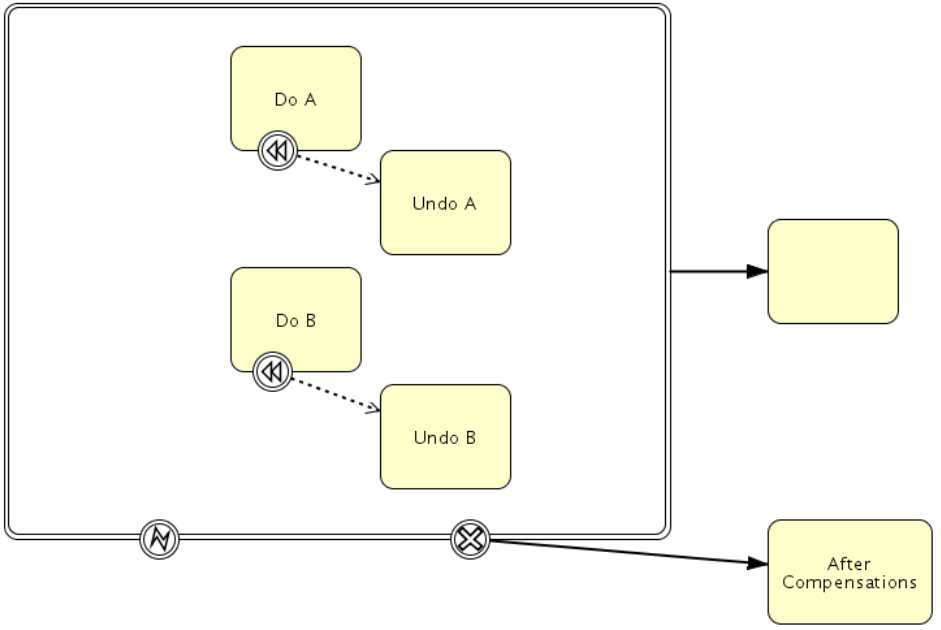
BPMN assumes long-running transactions (not the ACID model).

Activities contained in a transaction have to be wrapped in a transactional subprocess (depicted with a double border).

When transactions are aborted by cancel events, compensations are called for all terminated activities.

Compensations

Compensating action can be associated to actions within a transaction by using the compensation event (which is not really an event).

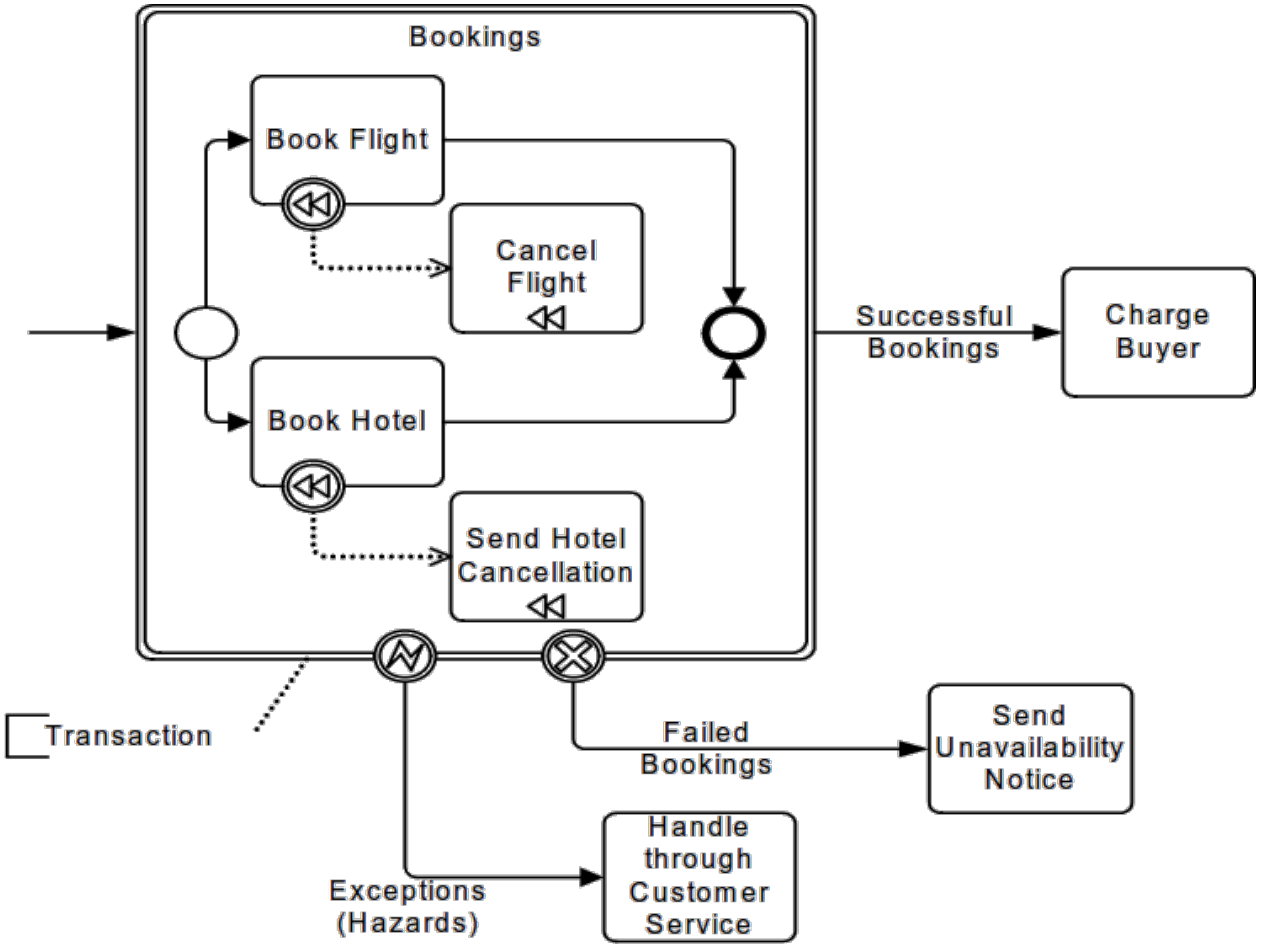


Transactions

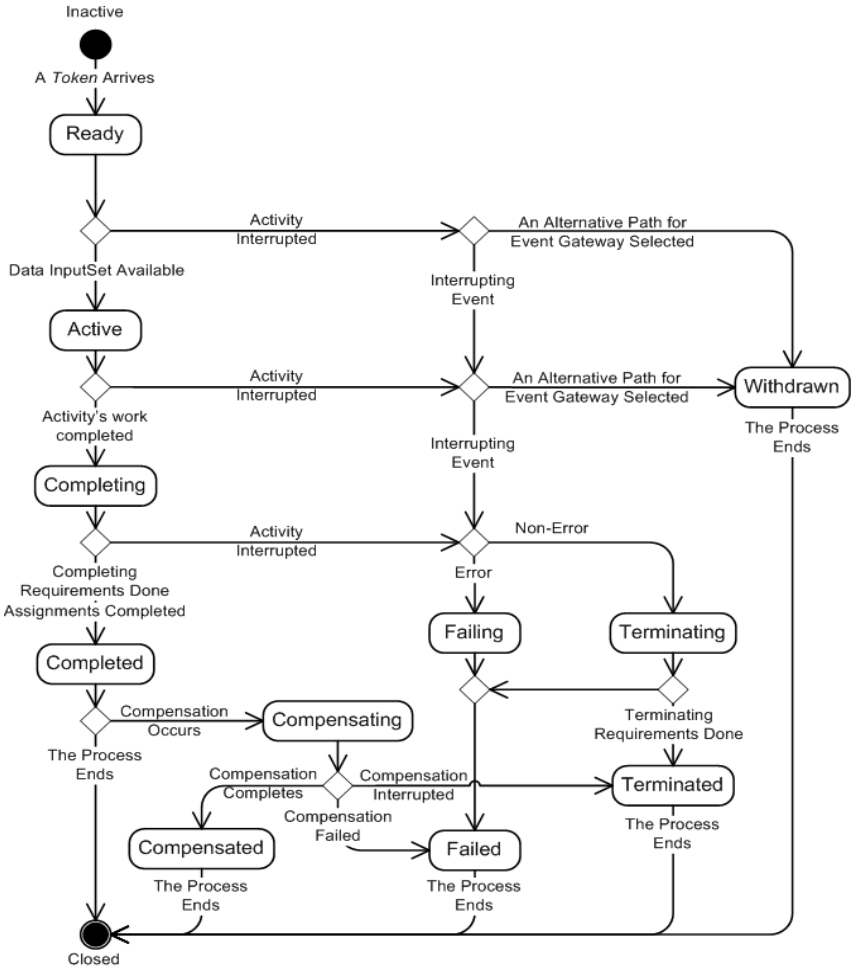
When a transaction aborts because of an attached event that is not a cancel event no compensation is performed.

A transaction can use a cancel end event to cancel itself activating the compensation mechanism.

Transaction example



States of an activity



Tips

- Designers should strive for “evident from the diagram” semantics. Try to express as much as possible using the visual notation.
- The “happy path” is not enough unless you just want to provide a high-level documentation of how a business process behaves.
- Don't expect solicited messages to arrive (use event gateways with timeouts)

Tips

- Name activities after the verb-noun prototype.
- Activities are expected to actually do something that cannot be expressed with other control-flow notations. A notable exception is a receive message task.
- Gateways cannot perform **any kind of action** or “computation”, they just “take note” of what is in the process data to take their “choices”. Decisions about these choices must be taken by upstream activities and committed to process data.

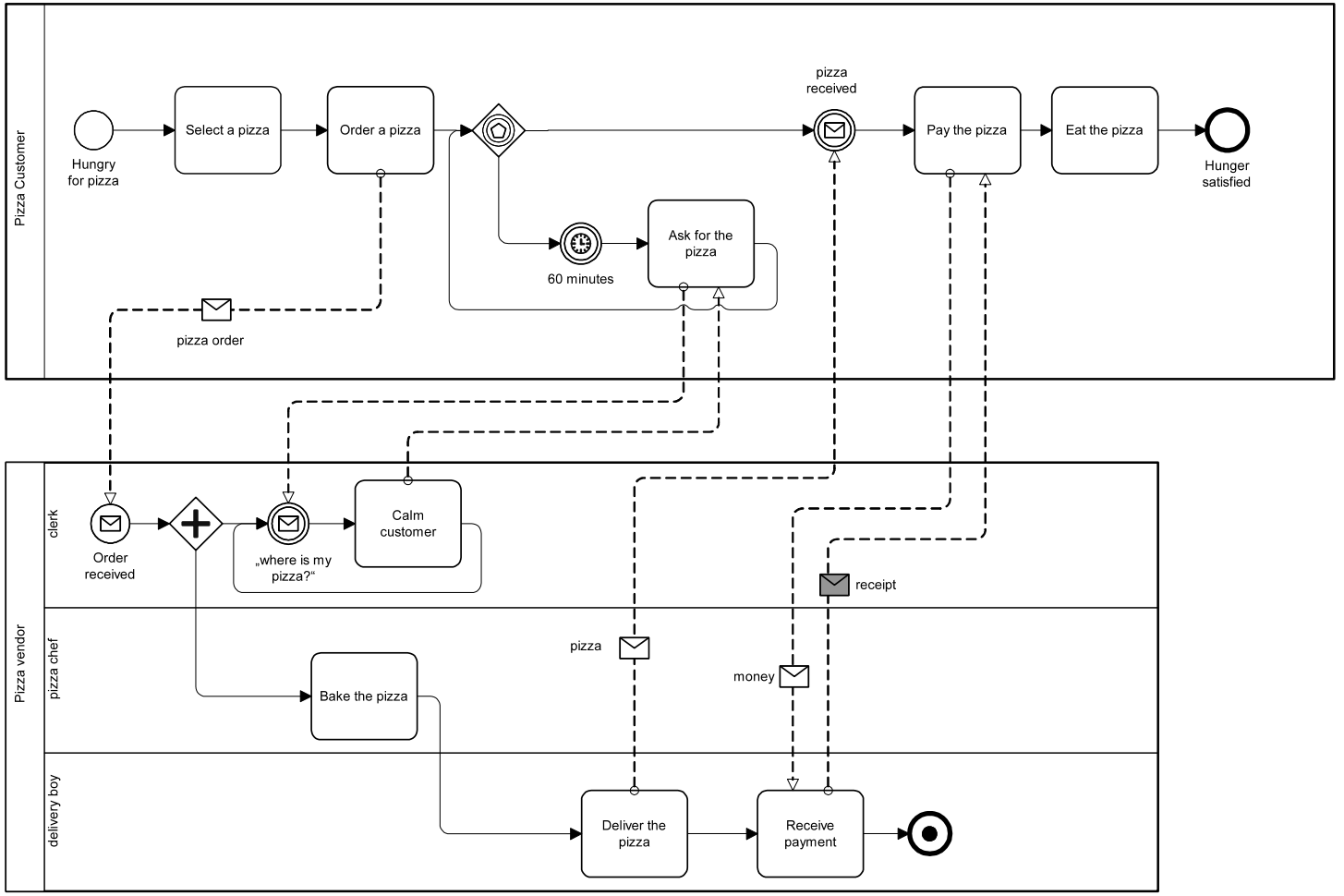
Pizza time: the hungry man

- When you’re hungry for pizza you pick up the menu and select your favorite Italian delicatessen
- You then phone call Pizza Pazza and make your order
- Now you wait for the pizza to appear at your door, if you wait for that more than 1 hour you make a phone call asking: “where’s my pizza?”
- When the pizza is finally delivered you pay for it; you can now eat your pizza!

Pizza time: the business

- Each time a new phone order is placed you start working on it
- The clerk receives the order and pass it to the chef
- The chef bakes the pizza and handle it to the delivery boy
- The delivery boy delivers the pizza and collects the payment
- If, at any time, the customer calls asking for their pizza just reply: “it just got out of the oven, it’s coming!”

Pizza collaboration example



Trivia time: can you read this?