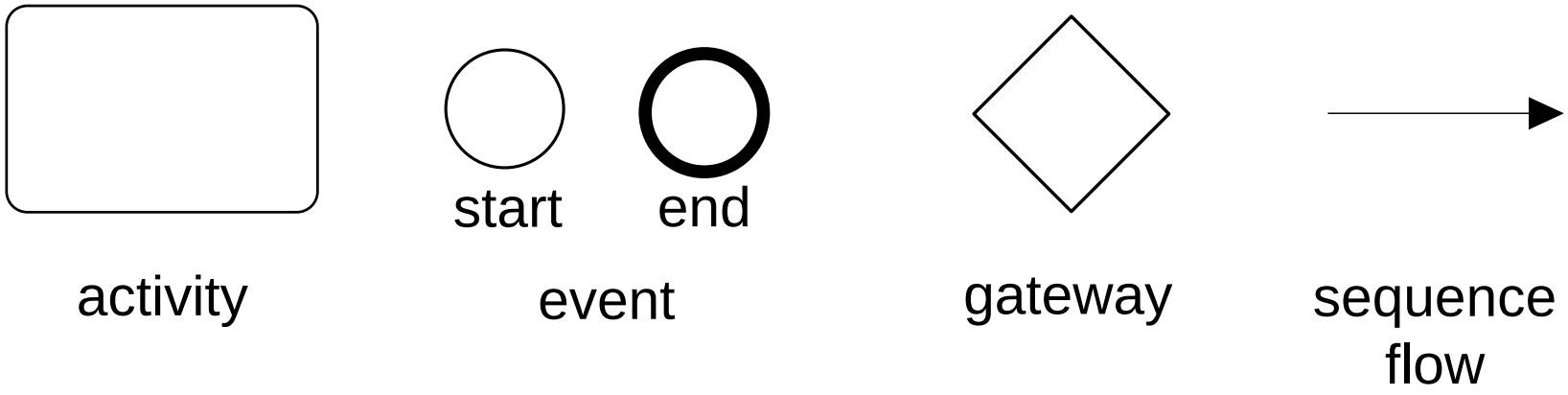
BPMN Introduction

A BPMN process model is a graph consisting of four types of **core elements**:



Let’s start modeling

Order-to-cash

An order-to-cash process is triggered by the receipt of a purchase order from a customer. Upon receipt, the purchase order has to be checked against the stock to determine if the the requested item(s) are available. Depending on stock availability the purchase order may be confirmed or rejected.

If the purchase order is confirmed, an invoice is emitted and the goods requested are shipped. The process completes by archiving the order or if the order is rejected.

Let’s start modeling – break it down

Order-to-cash

- An order-to-cash process is triggered by the receipt of a purchase order from a customer.
- Upon receipt, the purchase order has to be checked against the stock to determine if the the requested item(s) are available.
- Depending on stock availability the purchase order may be confirmed or rejected.
- If the purchase order is confirmed, an invoice is emitted and the goods requested are shipped.
- The process completes by archiving the order or if the order is rejected.

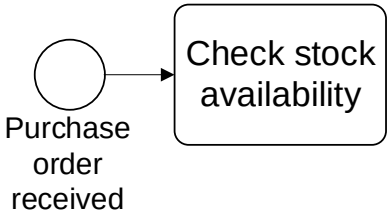
Let’s start modeling – break it down

Order-to-cash

- **An order-to-cash process is triggered by the receipt of a purchase order from a customer.**
- **Upon receipt, the purchase order has to be checked against the stock to determine if the the requested item(s) are available.**

BPMN Model

Order-to-cash



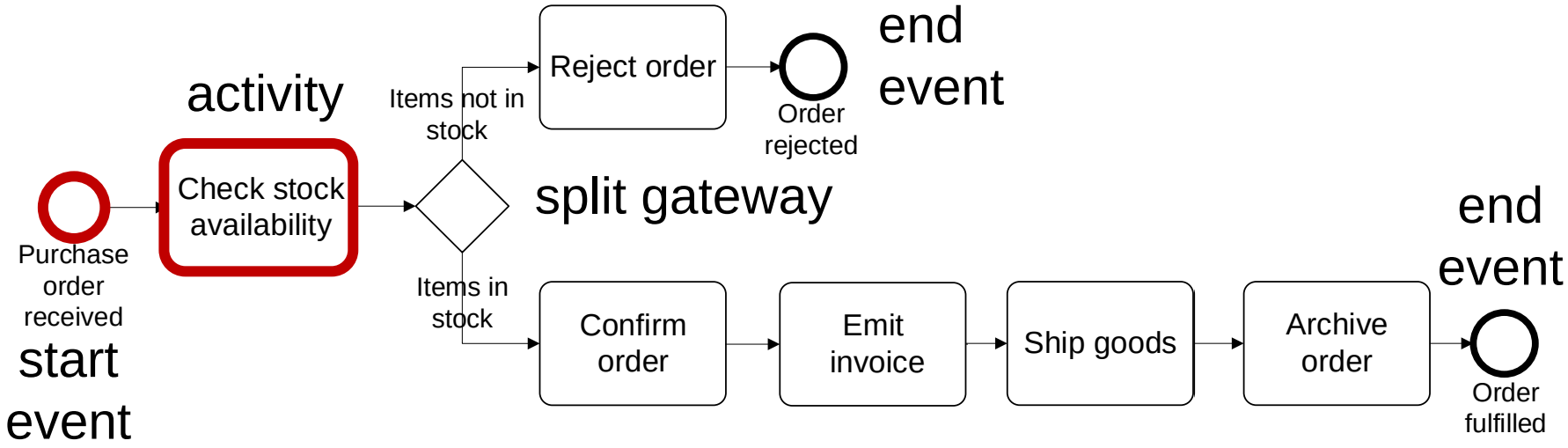
Let’s start modeling – break it down

Order-to-cash

- An order-to-cash process is triggered by the receipt of a purchase order from a customer.
- Upon receipt, the purchase order has to be checked against the stock to determine if the the requested item(s) are available.
- **Depending on stock availability the purchase order may be confirmed or rejected.**
- **If the purchase order is confirmed, an invoice is emitted and the goods requested are shipped.**
- **The process completes by archiving the order or if the order is rejected.**

BPMN Model

Order-to-cash

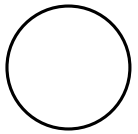


Naming conventions

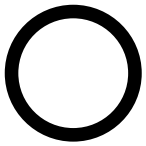
- Event: noun + past-participle verb (e.g. insurance claim lodged)
- Activity: verb + noun (e.g. assess credit risk)

A little bit more on events…

A *start event* triggers a new process instance by generating a token that traverses the sequence flow (“tokens source”)

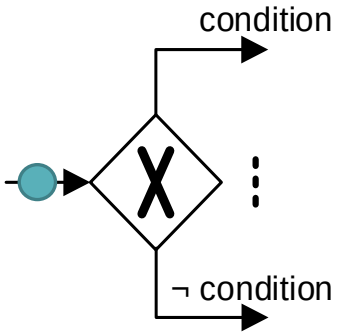


An *end event* signals that a process instance has completed with a given outcome by consuming a token (“tokens sink”)

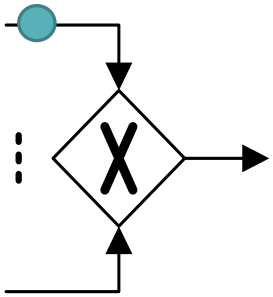


A little more on gateways: XOR Gateway

An *XOR Gateway* captures decision points (XOR-split) and points where alternative flows are merged (XOR-join)



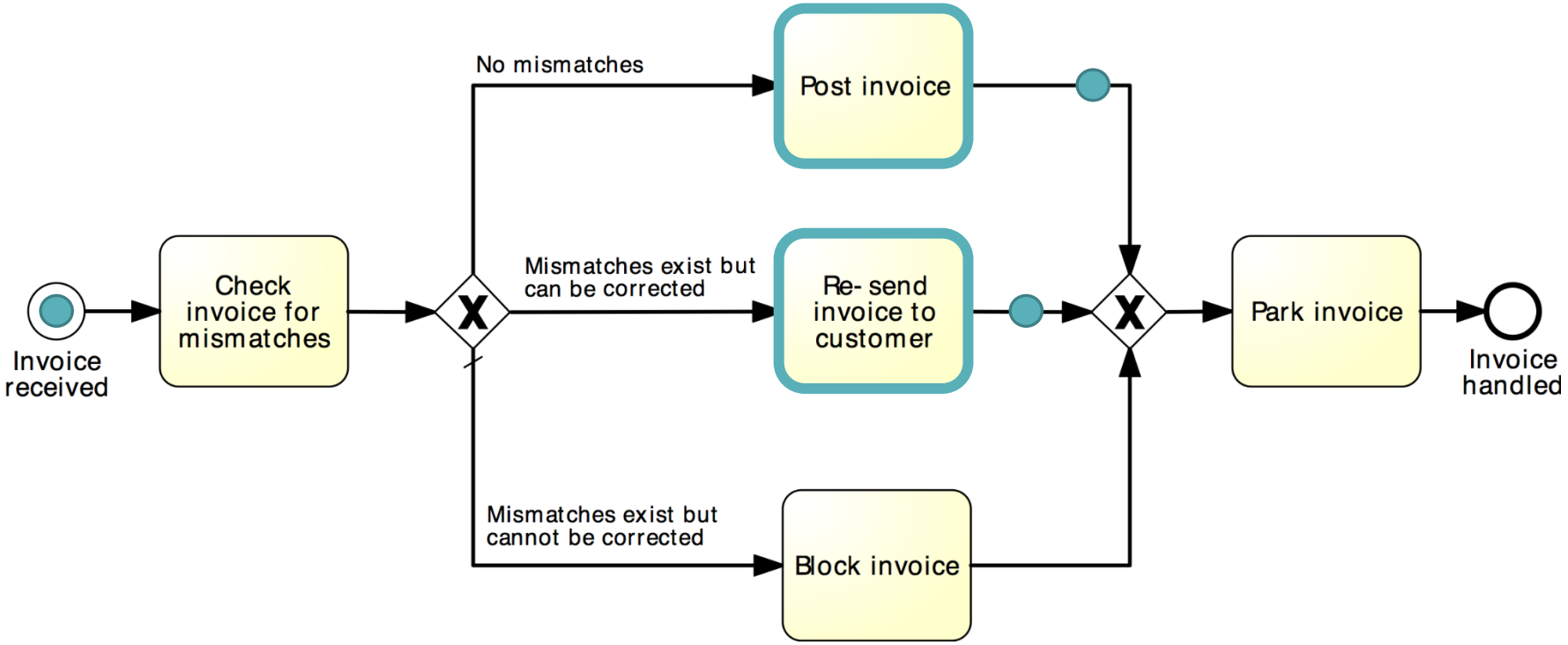
XOR-split ➔ takes **one** outgoing branch



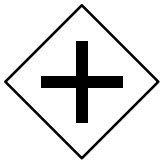
XOR-join ➔ proceeds when **one** incoming branch has completed

Example: XOR Gateway

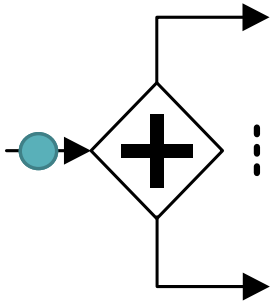
Invoice checking process



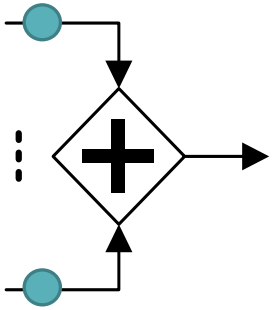
A little more on gateways: AND Gateway



An *AND Gateway* provides a mechanism to create and synchronize “parallel” flows.



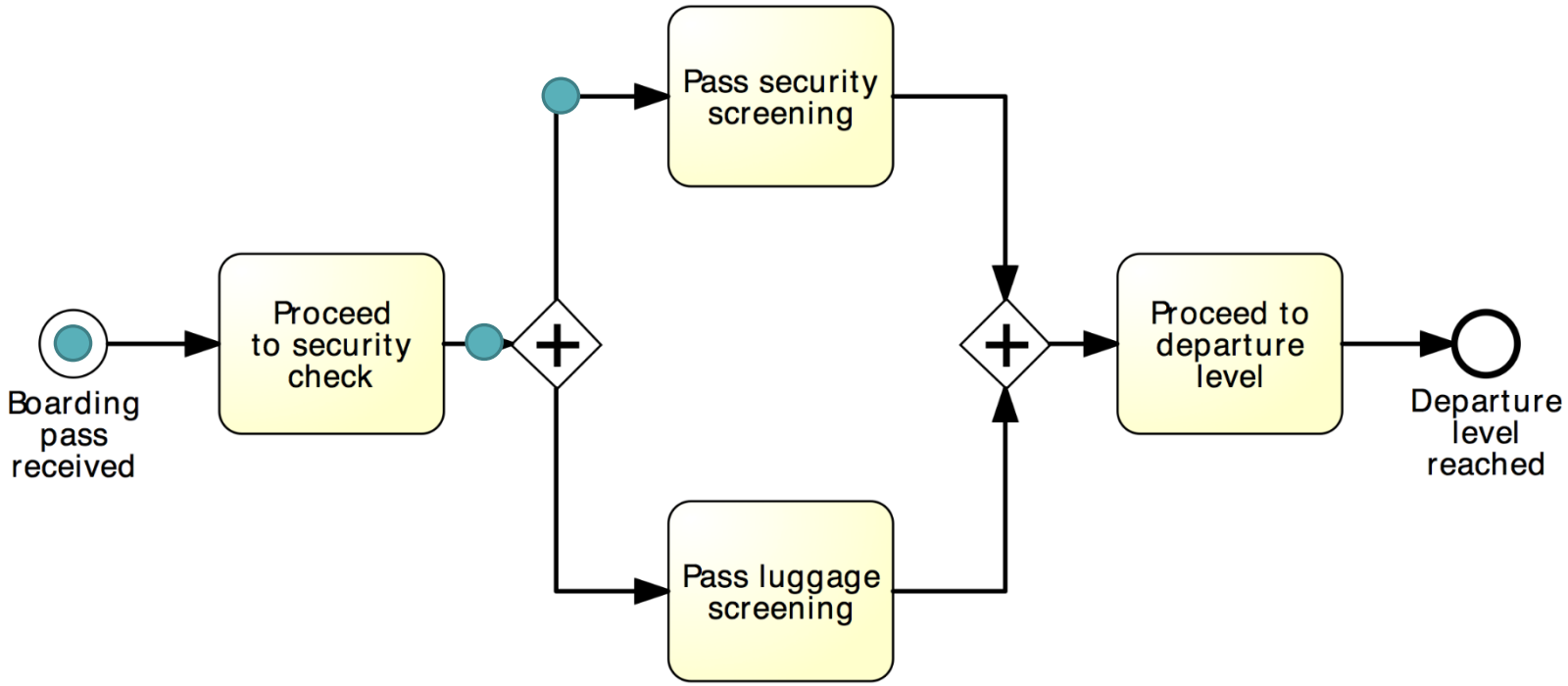
AND-split ➔ takes **all** outgoing branches



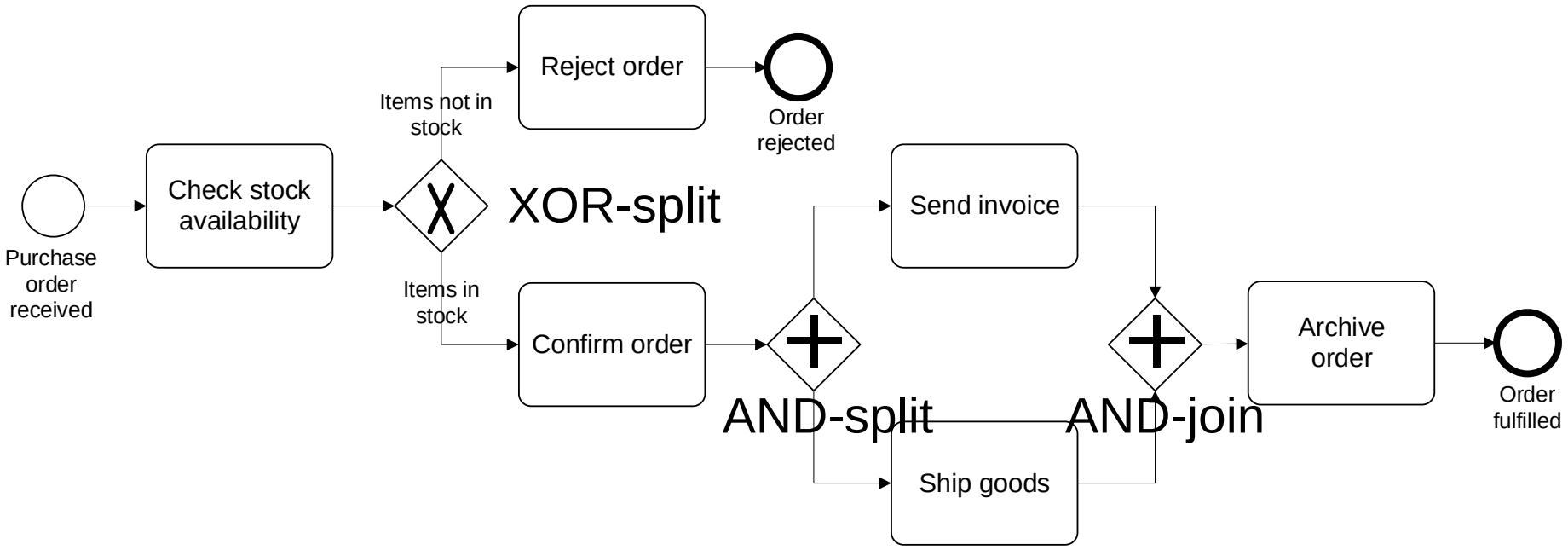
AND-join ➔ proceeds when **all** incoming branches have completed

Example: AND Gateway

Airport security check



Revised order-to-cash process model



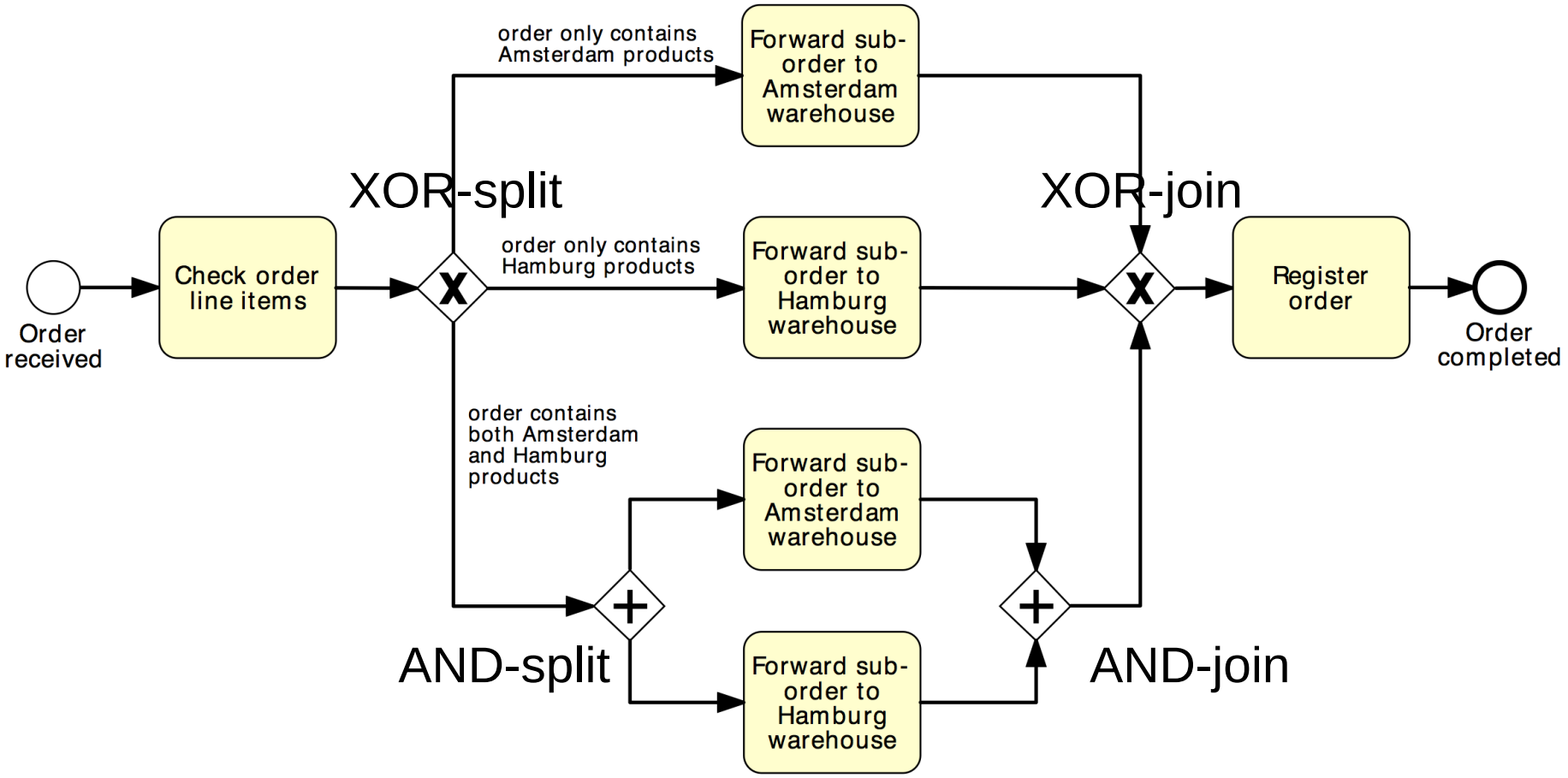
Between XOR and AND

Order distribution process

A company has two warehouses that store different products: Amsterdam and Hamburg. When an order is received, it is distributed across these warehouses: if some of the relevant products are maintained in Amsterdam, a sub-order is sent there; likewise, if some relevant products are maintained in Hamburg, a sub-order is sent there. Afterwards, the order is registered and the process completes.

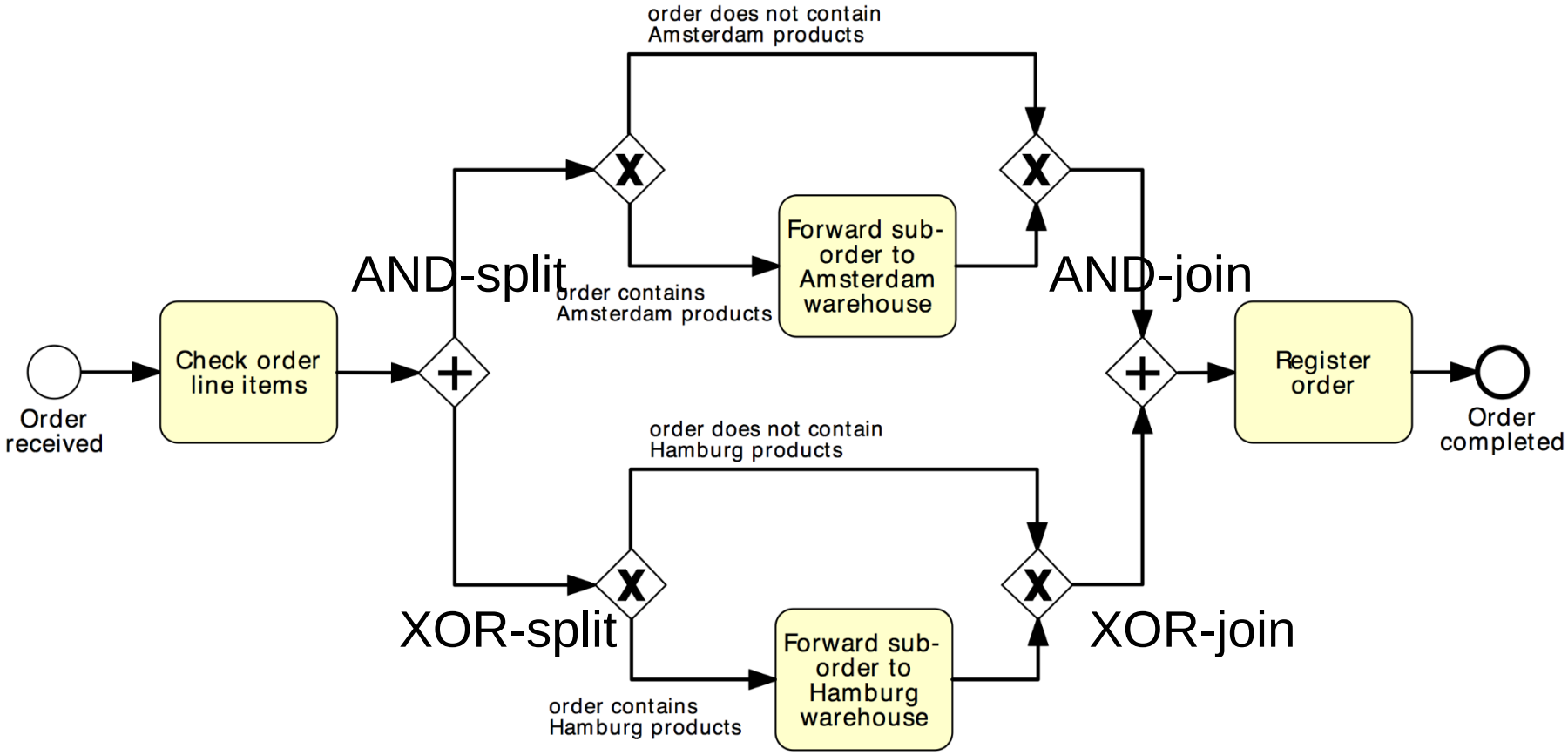
Solution 1

Order distribution process

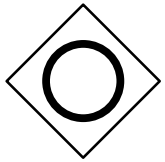


Solution 2

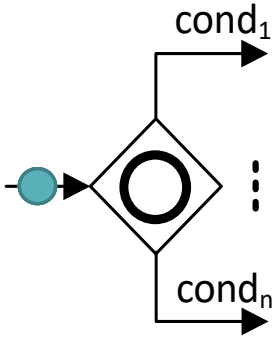
Order distribution process



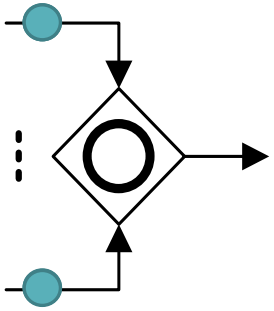
OR Gateway



An *OR Gateway* provides a mechanism to create and synchronize n out of m parallel flows.



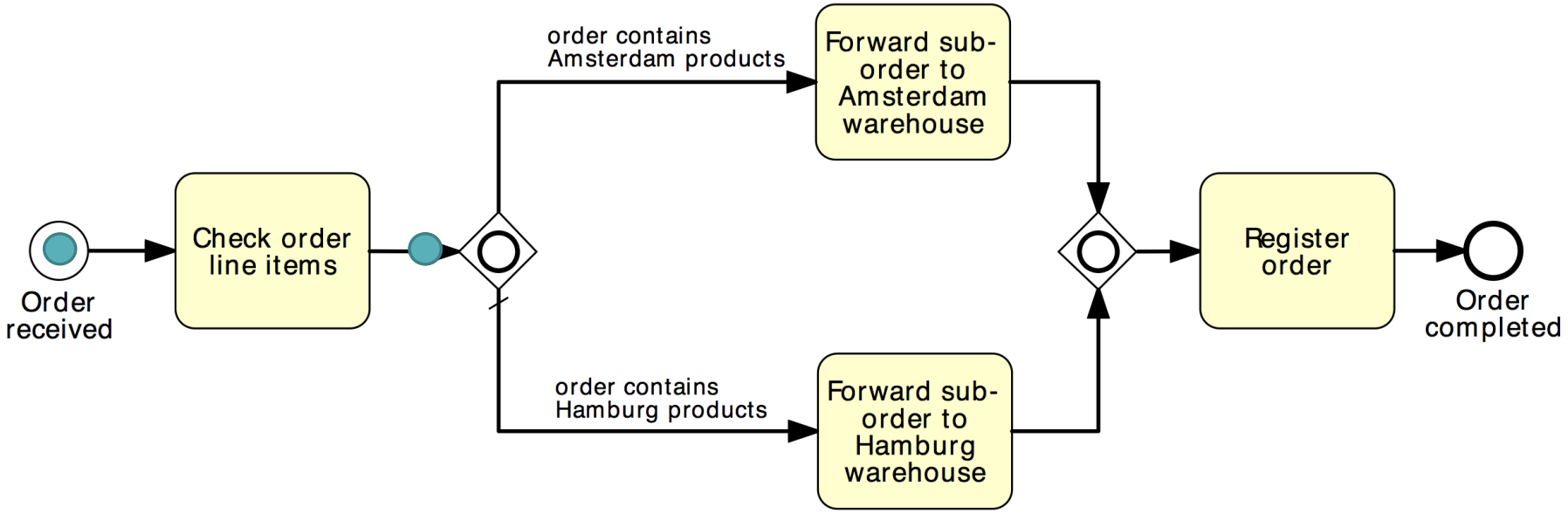
OR-split ➔ takes one or more branches depending on conditions



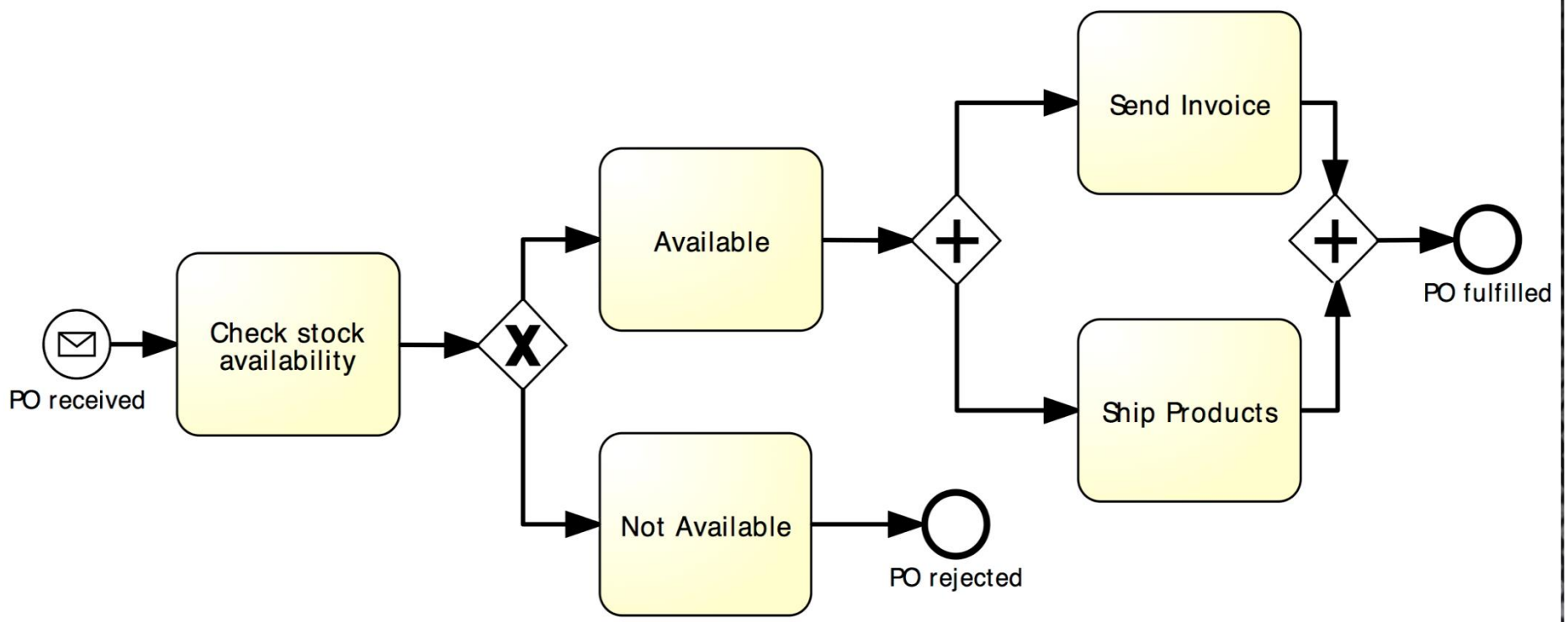
OR-join ➔ proceeds when all **active** incoming branches have completed

Solution using OR Gateway

Order distribution process



Beware: Beginner's Mistake…



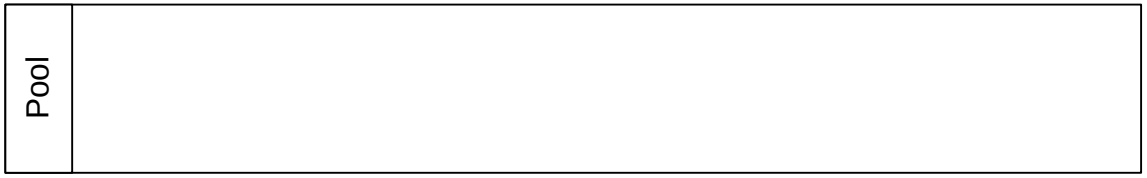
Guidelines: Naming Conventions

1. Give a name to every event and task
2. For tasks: verb followed by business object name and possibly complement
   - Issue Driver Licence, Renew Licence via Agency
3. For message events: object + past participle
   - Invoice received, Claim settled
4. Avoid generic verbs such as Handle, Record…
5. Label each XOR-split with a condition
   - Policy is invalid, Claim is inadmissible

Organizational Elements in BPMN – Pools & Lanes

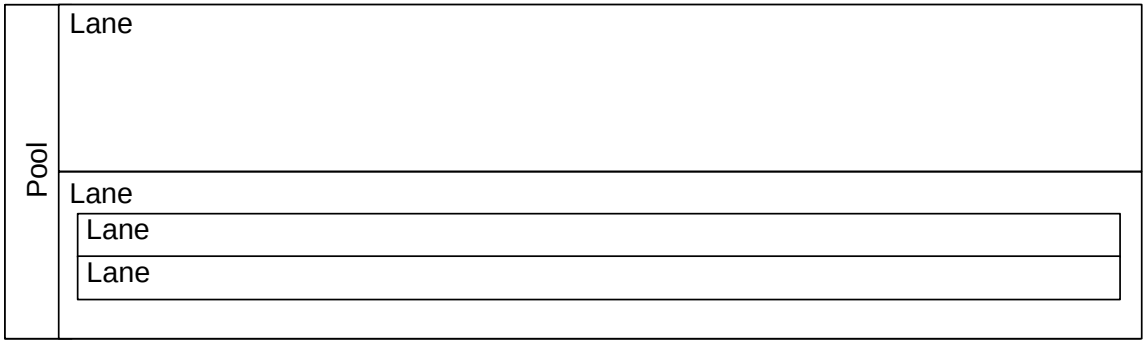
Pool

Captures a resource class. Generally used to model a business party (e.g. a whole company)

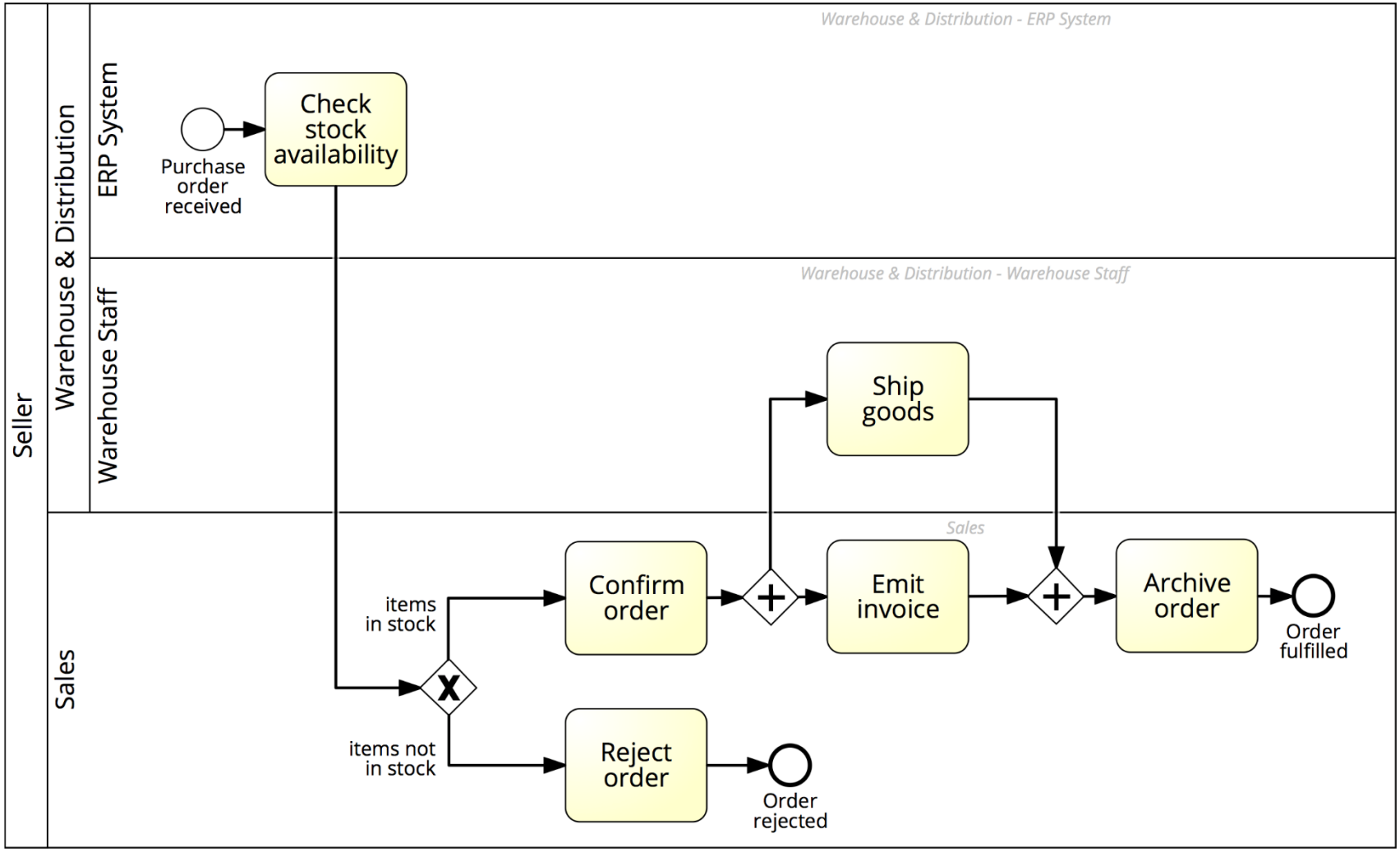


Lane

A *resource sub-class* within a pool. Generally used to model departments (e.g. shipping, finance), internal roles (e.g. Manager, Associate), software systems (e.g. ERP, CRM)

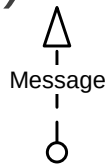


Order-to-cash process with lanes



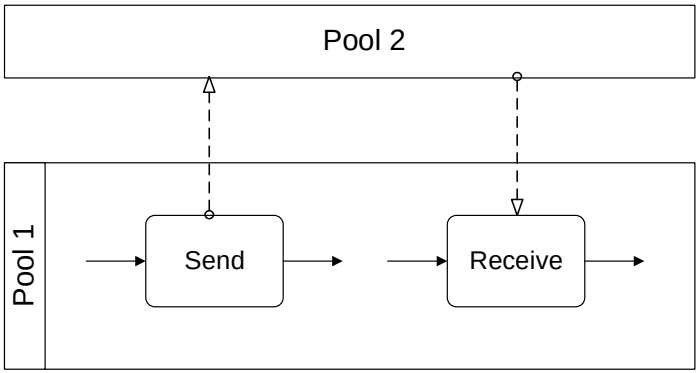
Message Flow

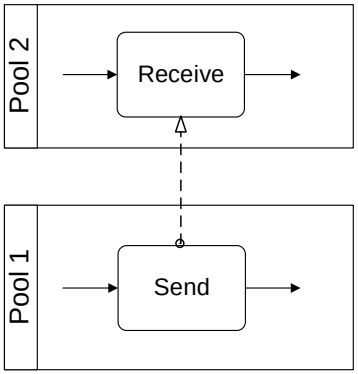
A *Message Flow* represents a flow of information between two process parties (Pools)



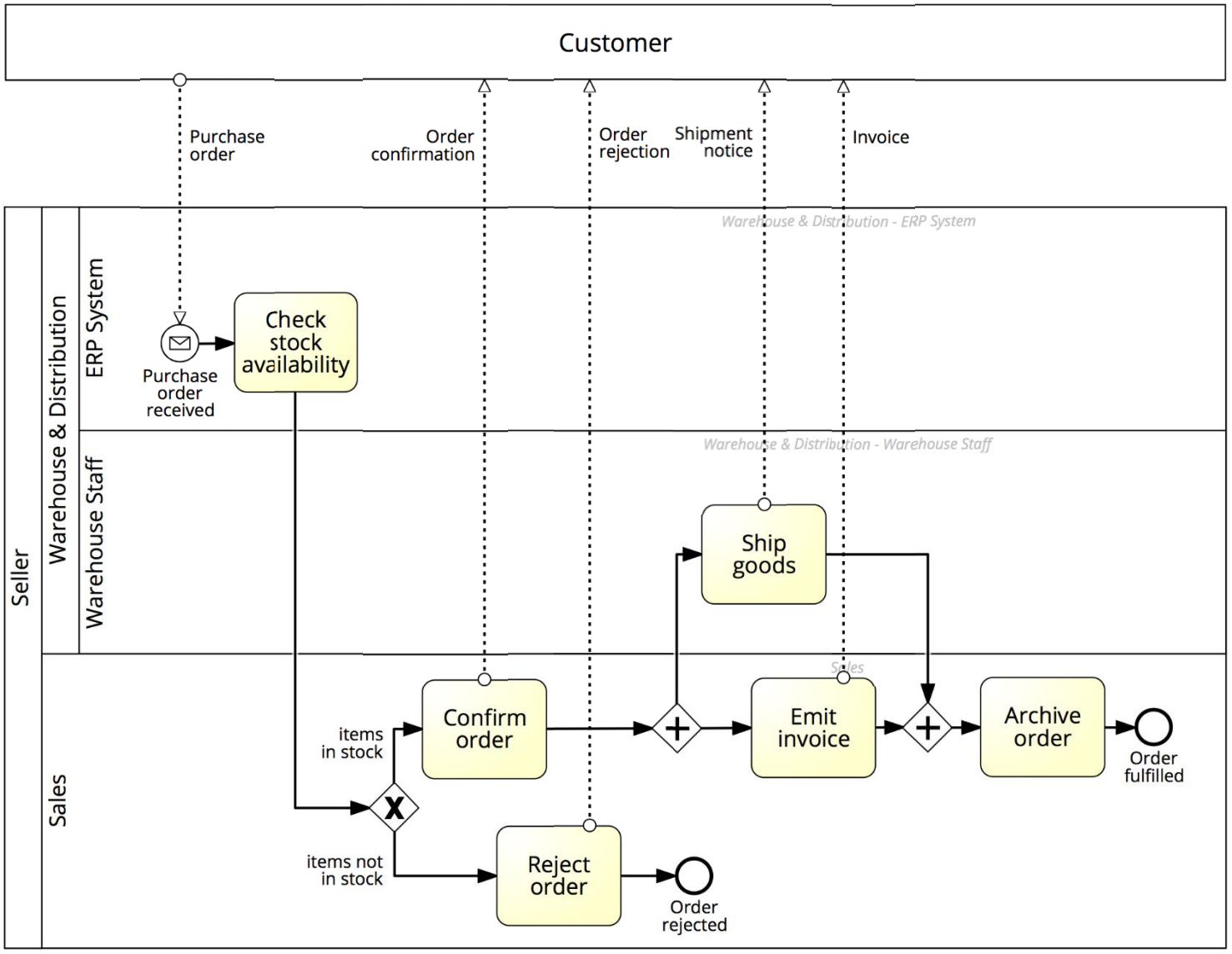
A Message Flow can connect:

- directly to the boundary of a Pool ➔ captures an *informative* message to/from that party
- to a specific activity or event within that Pool ➔ captures a message that triggers a specific activity/event within that party



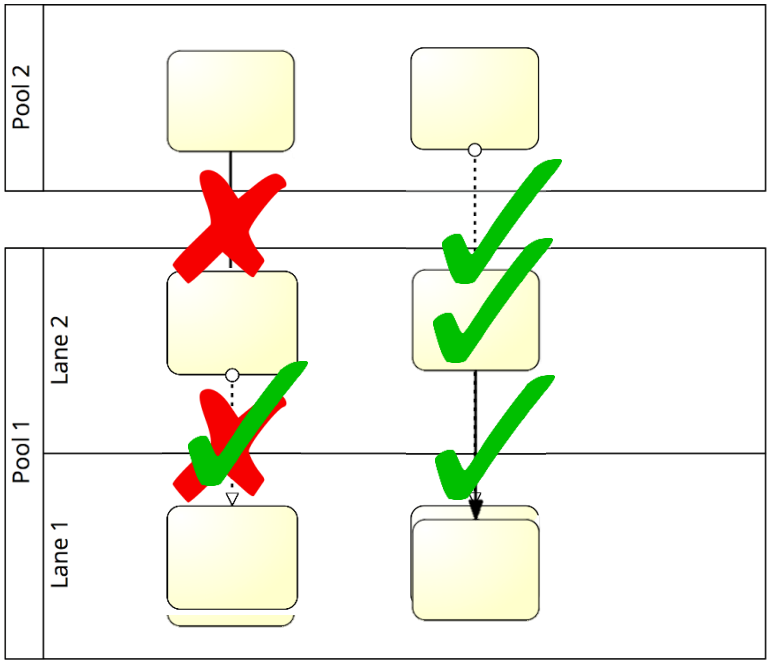


Order-to-cash process with a black-box customer pool

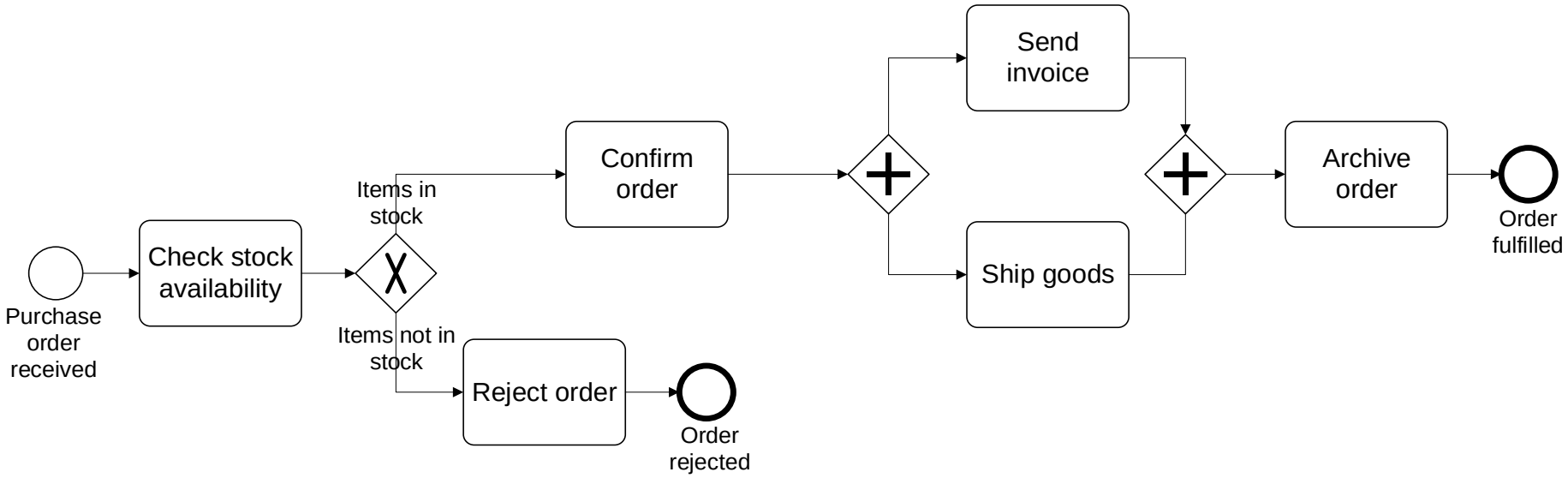


Pools, Lanes and Flows: syntactic rules

1. A Sequence Flow **cannot** cross the boundaries of a Pool (message flows can)
2. Both Sequence Flow and Message Flow **can cross** the boundaries of Lanes
3. A Message Flow **cannot connect** two flow elements within the same pool

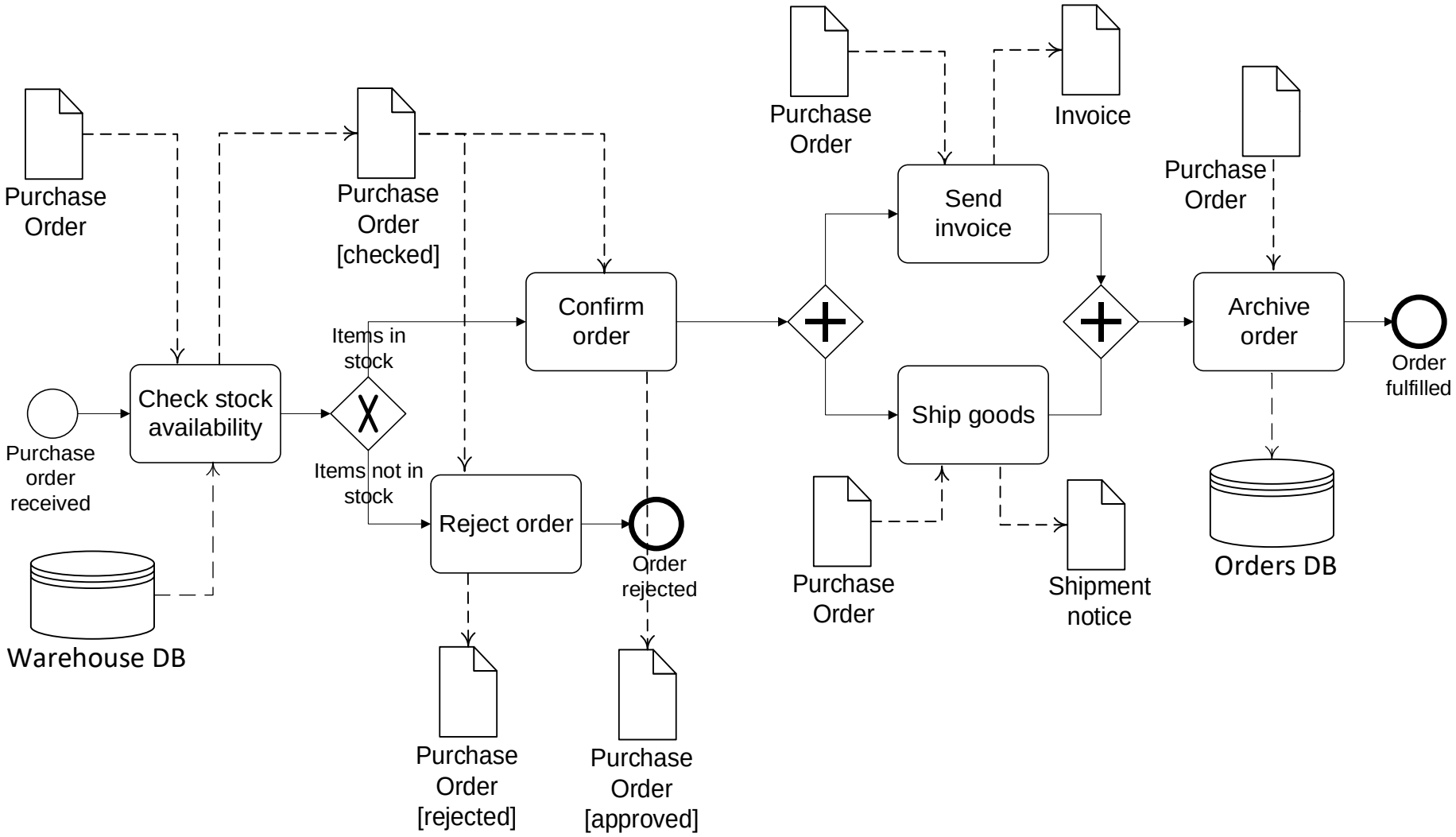


Order-to-cash process, again

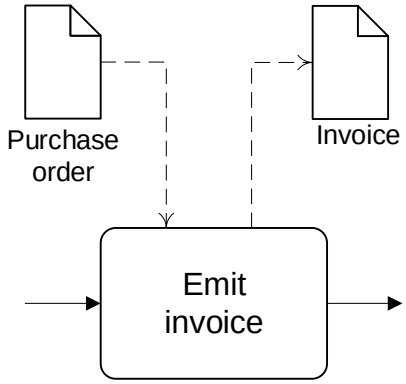


The purchase order document serves as an input to the stock availability check. Based on the outcome of this check, the status of the document is updated, either to “approved” or “rejected”. If the order is approved, an invoice and a shipment notice are produced.

Model with information artifacts

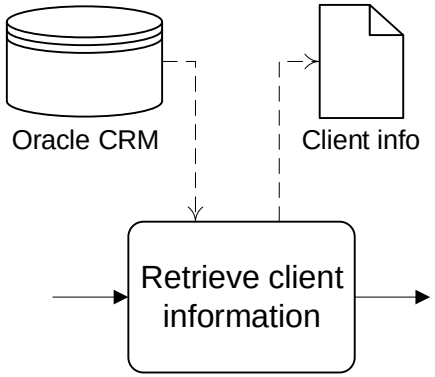


BPMN Information Artifacts



A *Data Object* captures an artifact required (input) or produced (output) by an activity.

- Can be physical or electronic



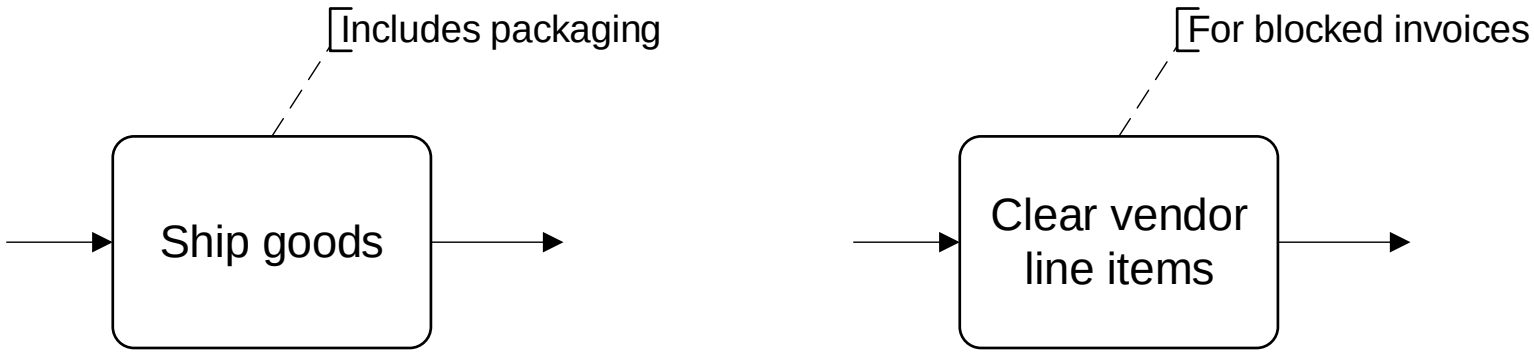
A *Data Store* is a place containing data objects that must be persisted beyond the duration of a process instance.

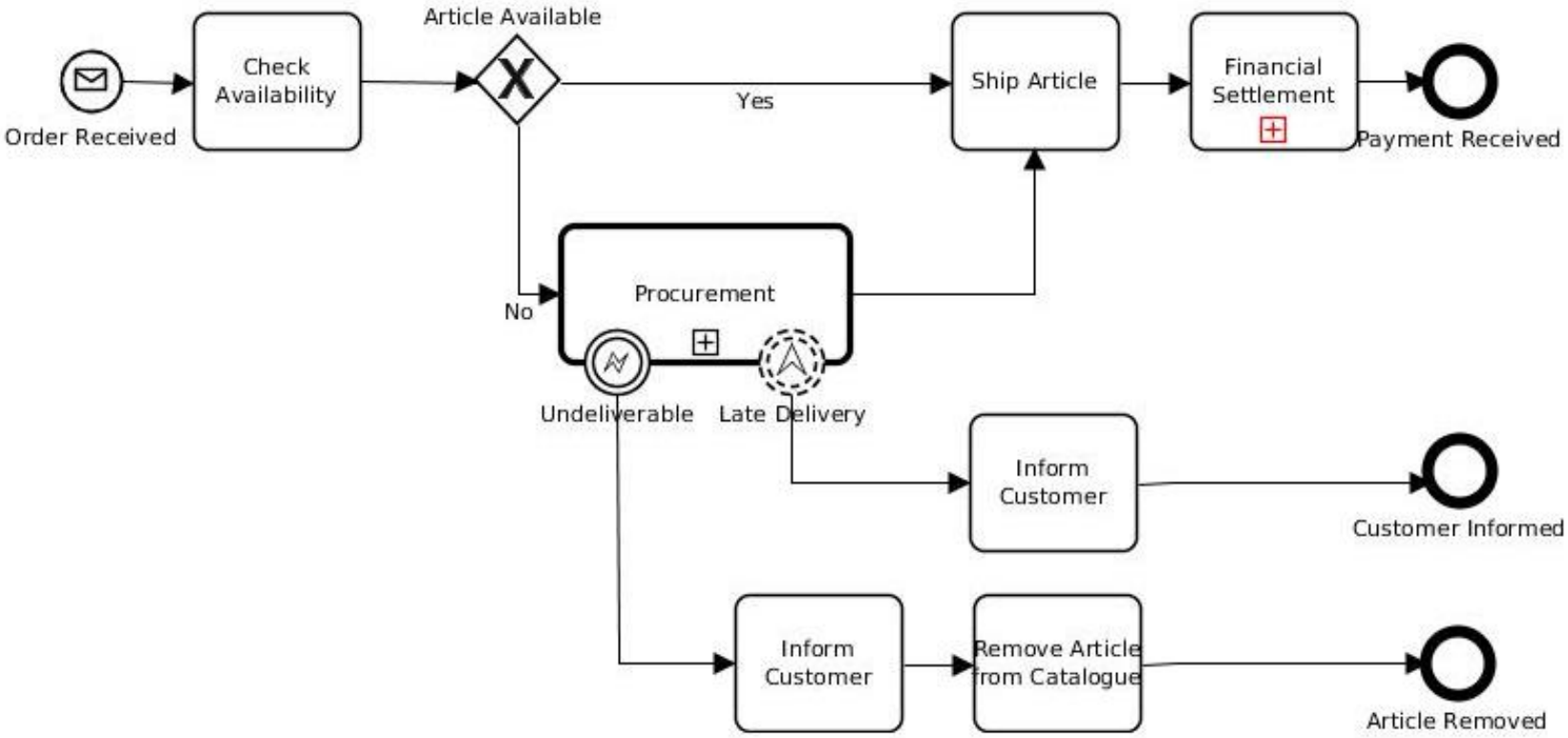
It is used by an activity to store (as output) or retrieve (as input) data objects.

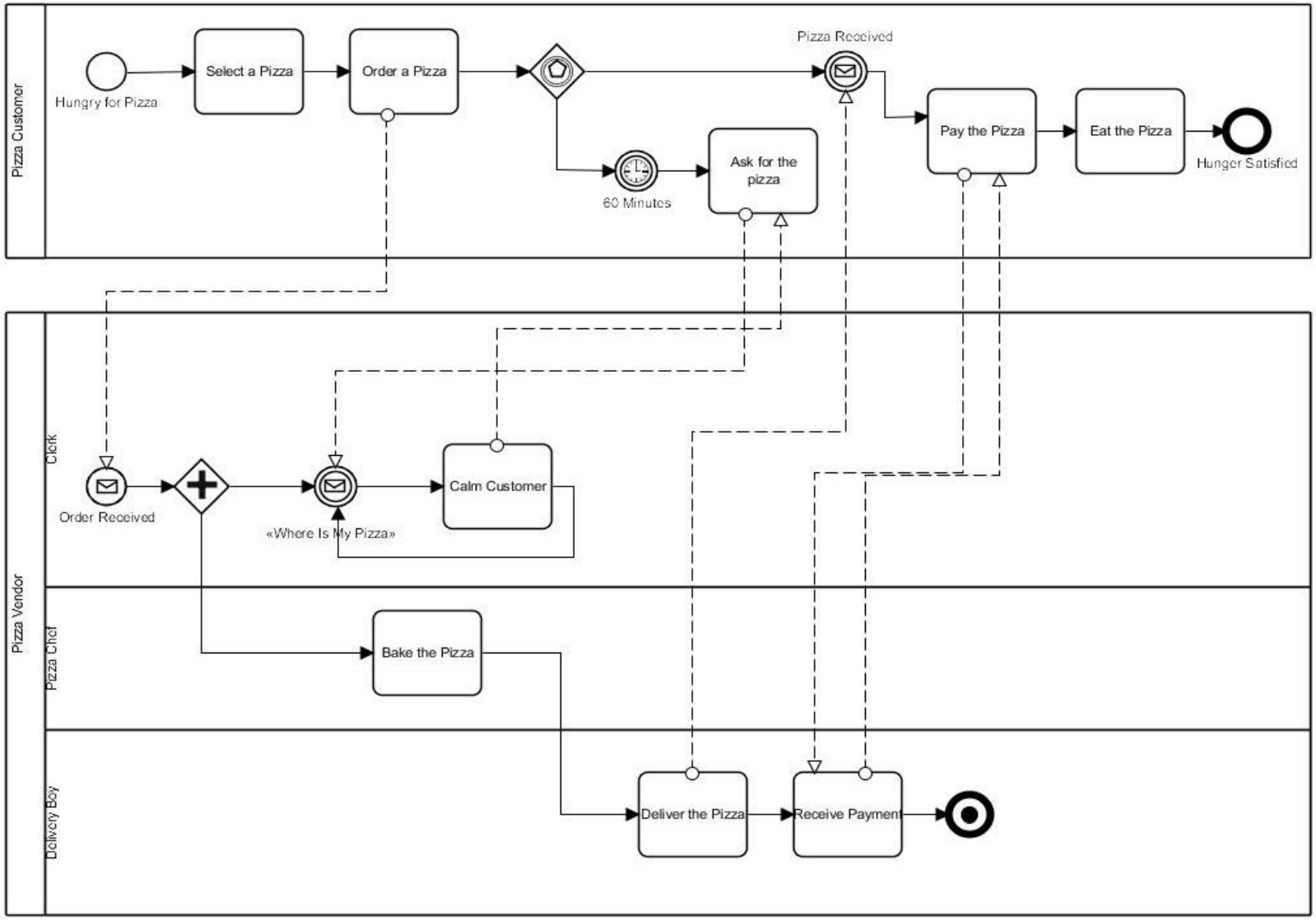
Quick Note: BPMN Text Annotations

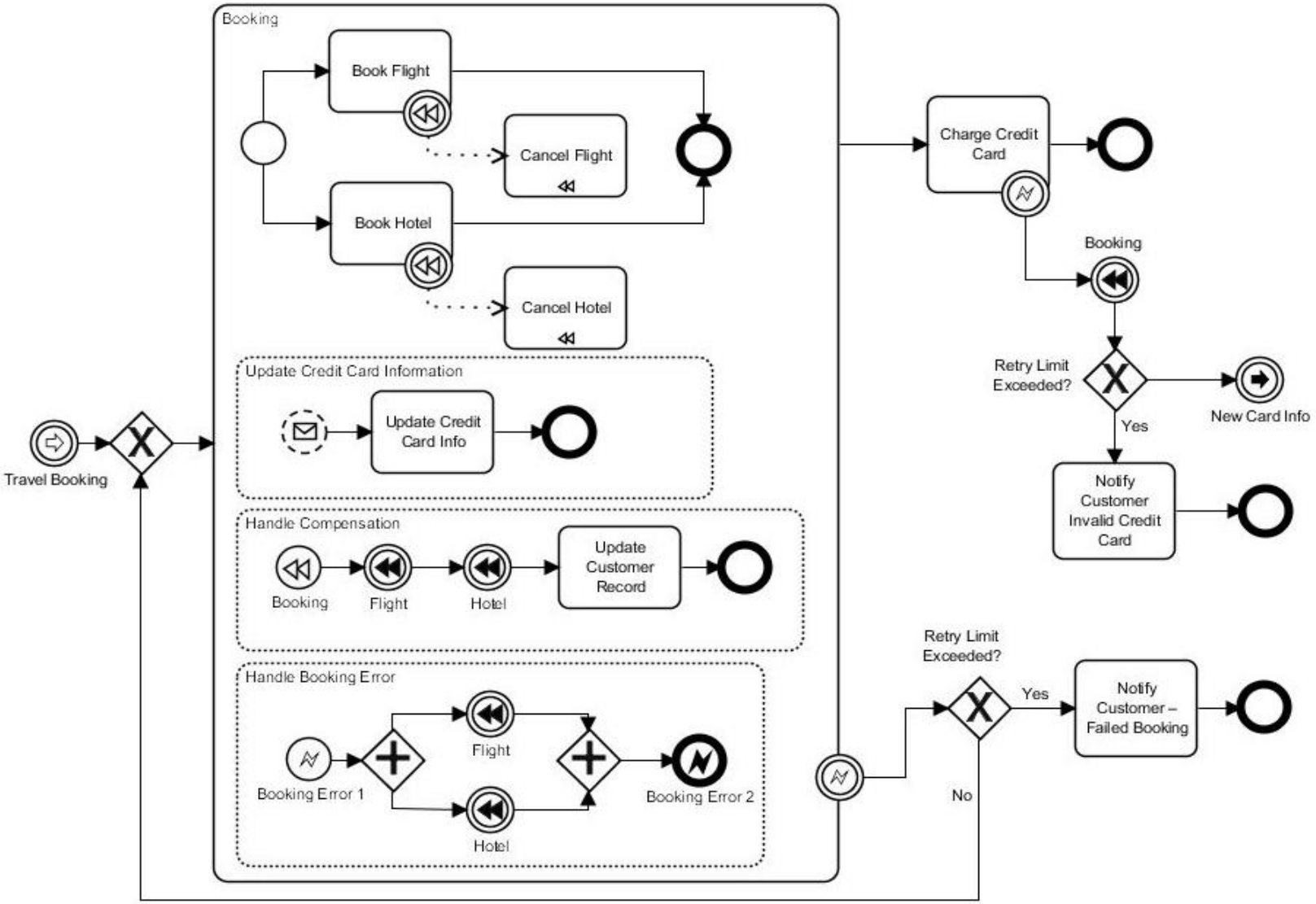
A *Text Annotation* is a mechanism to provide additional text information to the model reader

- **Doesn't affect** the flow of tokens through the process




Order Received
Check Availability
Article Available
Yes
Ship Article
Financial Settlement
Payment Received
No
Procurement
Undeliverable
Late Delivery
Inform Customer
Customer Informed
Inform Customer
Remove Article from Catalogue
Article Removed


Pizza Customer
Hungry for Pizza
Select a Pizza
Order a Pizza
Pizza Received
60 Minutes
Ask for the pizza
Pay the Pizza
Eat the Pizza
Hunger Satisfied
Pizza Vendor
Clerk
Order Received
«Where Is My Pizza»
Calm Customer
Pizza Chef
Bake the Pizza
Delivery Boy
Deliver the Pizza
Receive Payment


Booking
Book Flight
Cancel Flight
Book Hotel
Cancel Hotel
Update Credit Card Information
Update Credit Card Info
Handle Compensation
Booking
Flight
Hotel
Update Customer Record
Handle Booking Error
Booking Error 1
Flight
Hotel
Booking Error 2
Travel Booking
Charge Credit Card
Booking
Retry Limit Exceeded?
Yes
New Card Info
Notify Customer Invalid Credit Card
Retry Limit Exceeded?
Yes
No
Notify Customer – Failed Booking