Business Processes Modelling

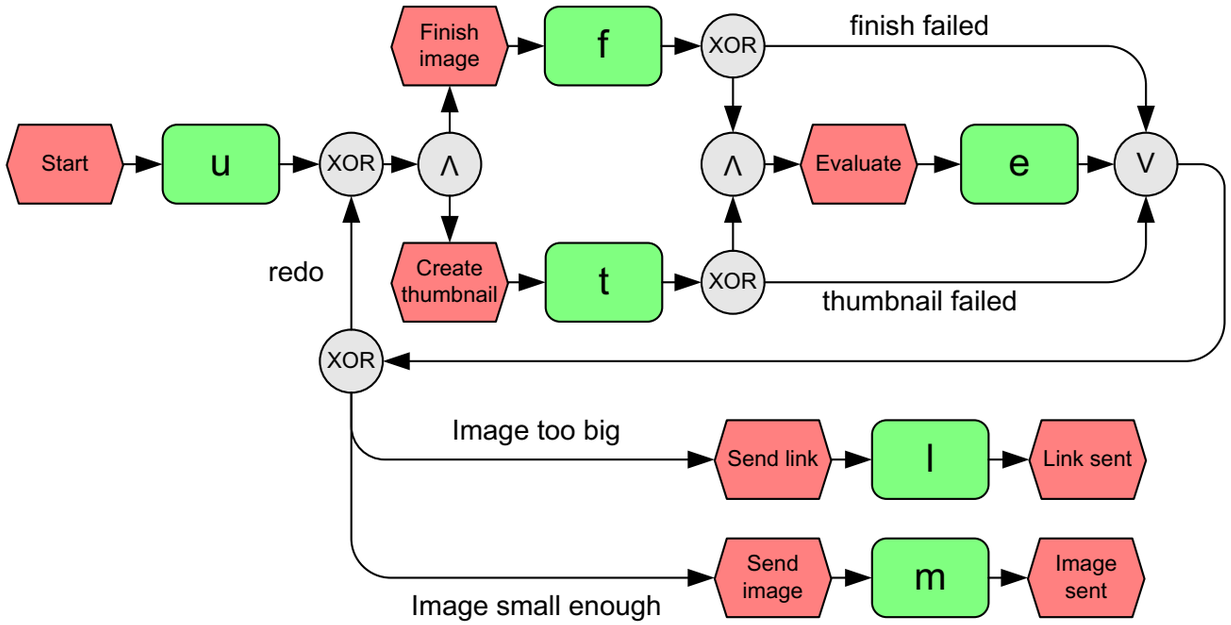
MPB (6 cfu, 295AA)

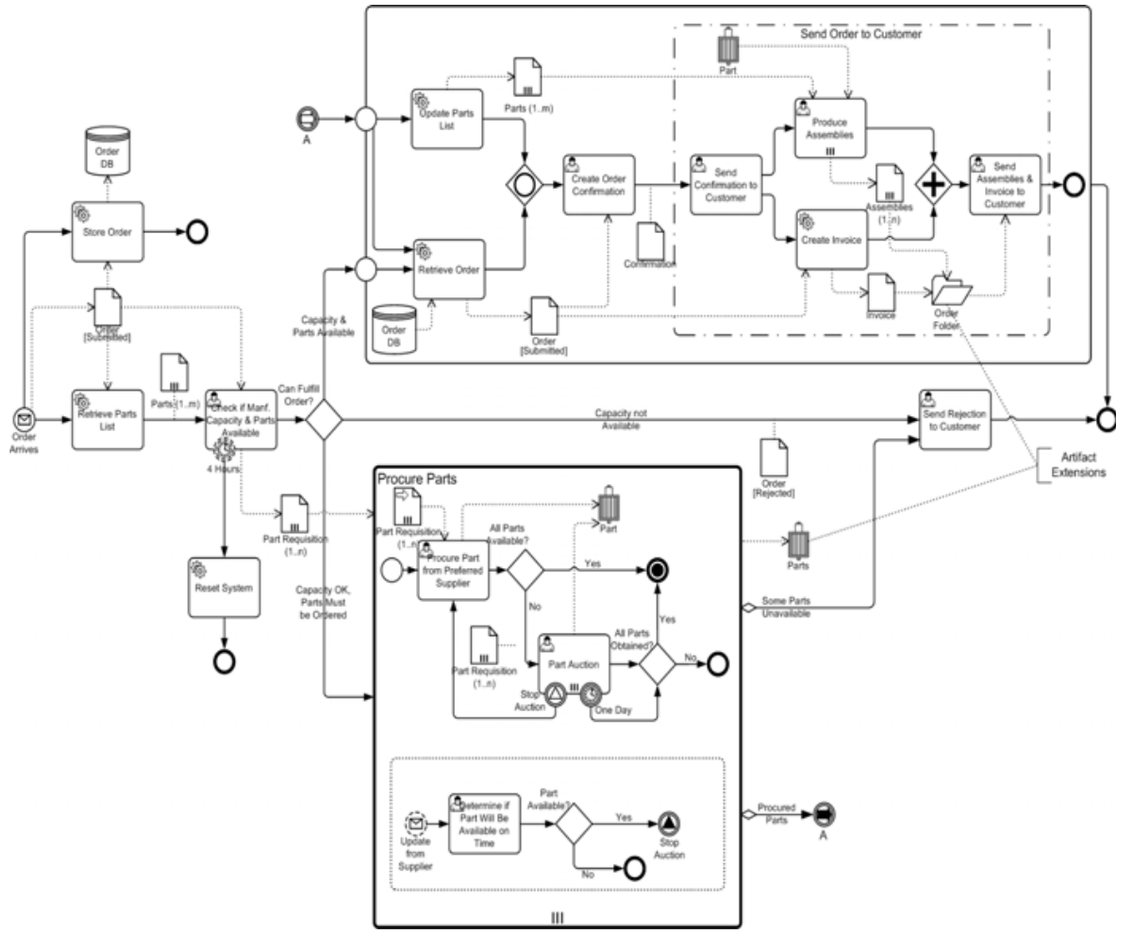
Roberto Bruni

http://www.di.unipi.it/~bruni

07b - Other BPMN features

Object





We overview high-level diagrammatic notation

Ch.4.3,4.7, 5.7 of Business Process Management: Concepts, Languages, Architectures
Ch.3, 4 of Fundamental of Business Process Management. M. Dumas et al.

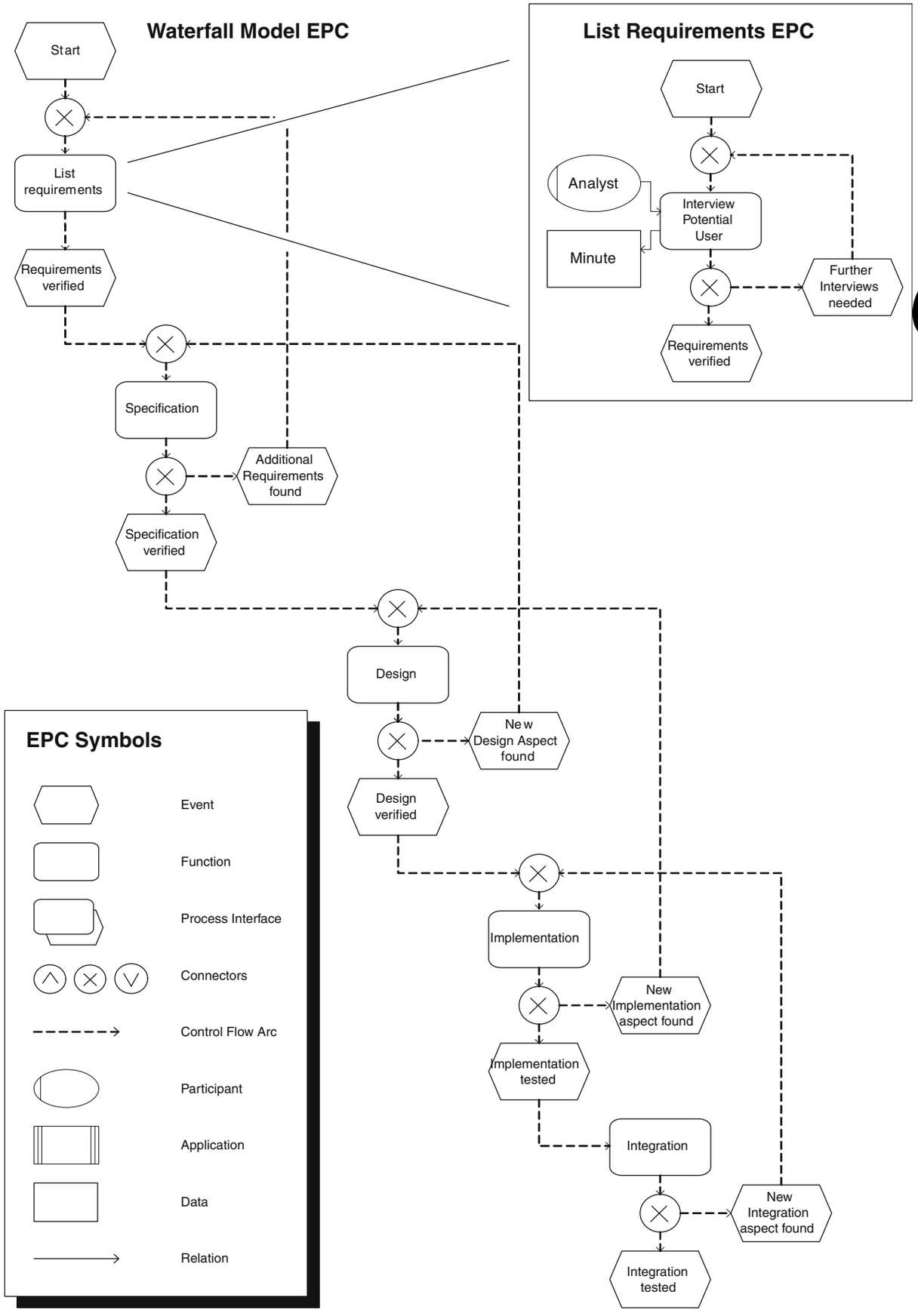


Fig. 1 Event-driven process chains representing the waterfall model for software engineering

Example: comments?

ISeB (2006) 4: 245–263
DOI 10.1007/s10257-005-0026-1

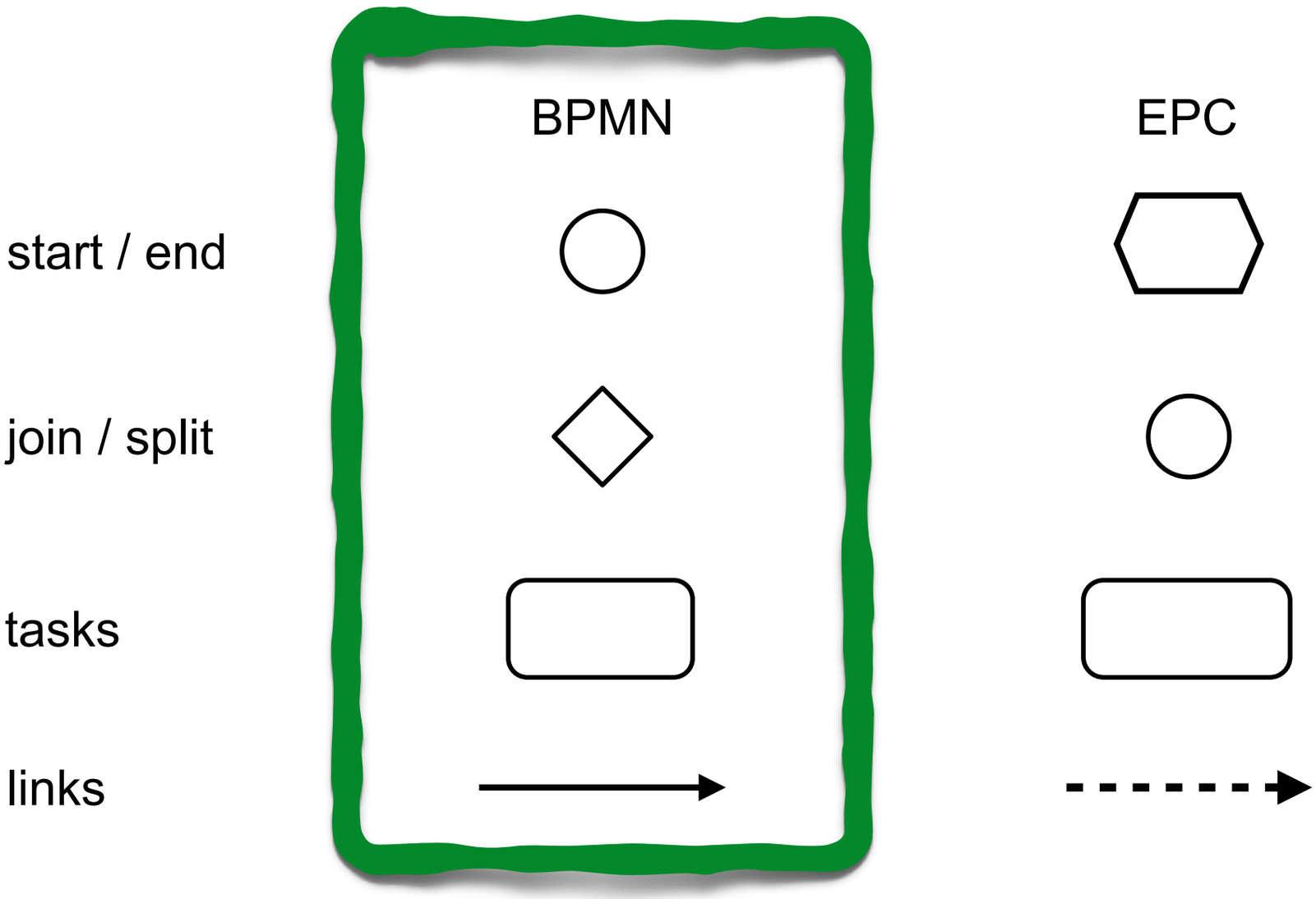
ORIGINAL PAPER

Jan Mendling · Markus Nüttgens

EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC)

Published online: 22 October 2005

A closer look at standards: EPC and BPMN



BPMN Artefacts:

(data-objects, groups, text annotations)

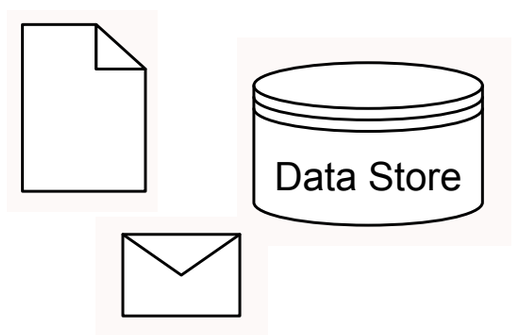
Artefacts

BPMN is designed to allow modellers and modelling tools some flexibility in extending the basic notation

Any kind of artefacts can be added to a diagram as appropriate for the specific modelling domain

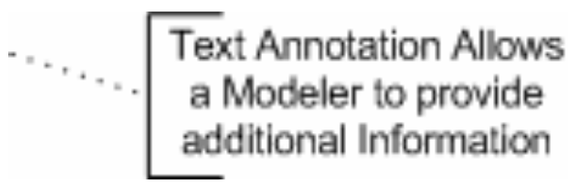
BPMN includes three pre-defined types of artefacts:

data objects **groups** **text annotation**



to be discussed later

to be discussed later



Association

An **association** is used to associate data, text, and other artefacts with flow objects

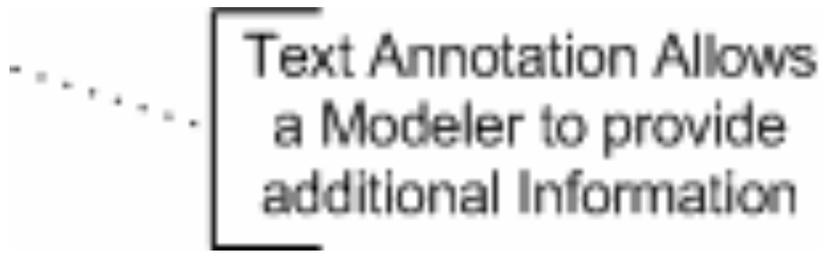
An association is represented by
a dotted line (with an optional line-arrowhead)

used especially
for text annotation

Text annotation

Any object can be associated with a **text annotation**
to provide any additional information and documentation
that can be needed

A text annotation is represented as a dotted-line call-out



BPMN key features

Markers

(events, activities, gateways)

Activity types and markers

Internal markers indicate: the activity nature (**task type**) and the way it is executed (**activity marker**)

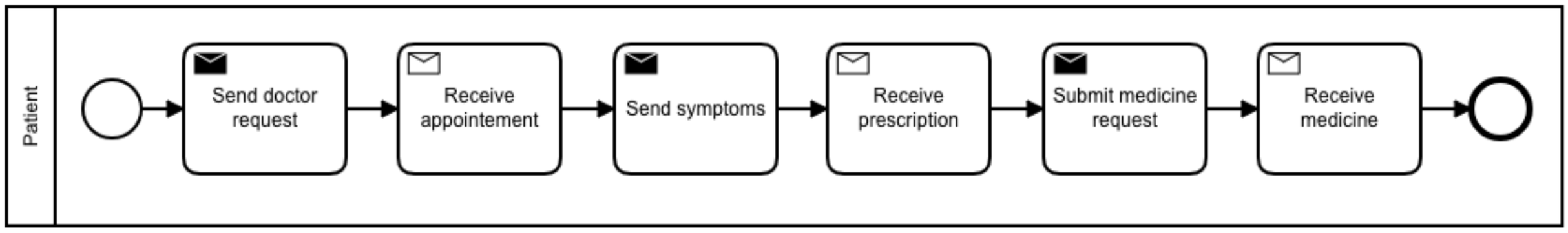
- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

some types

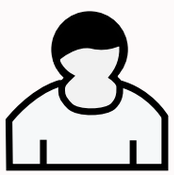
- Sub-Process Marker
- Loop Marker
- Parallel MI Marker
- Sequential MI Marker
- Ad Hoc Marker
- Compensation Marker

some markers

Sending and receiving types



User vs manual types



User Task

An activity that requires direct interaction from a person through a user interface.
It is performed by a human with the aid of a computer system or software.

Manual Task

An activity that is performed entirely by a person, without the direct assistance of a computer system.

Service vs script types

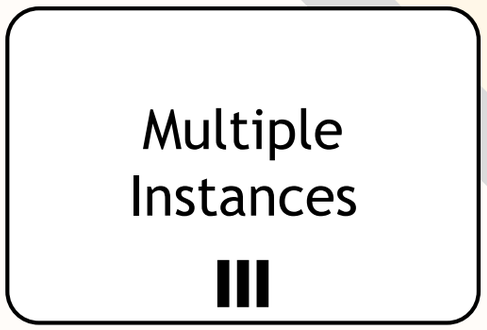
Service Task

An automated task where the process interacts with an external service or system.
This could be a web service, an API, or any system that performs the task without human intervention.

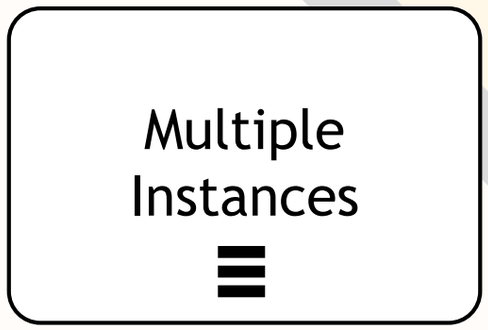
Script Task

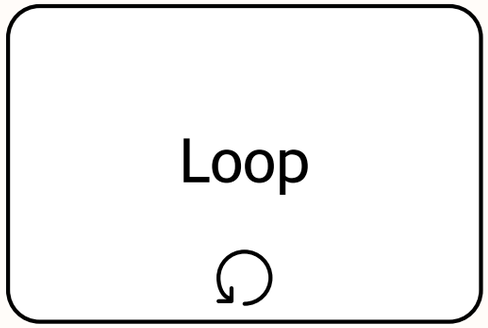
An automated task where a script (such as JavaScript, Python, etc.) is executed within the BPMN engine itself to perform a task.
The script runs internally without the need to call external services.

Some activity markers

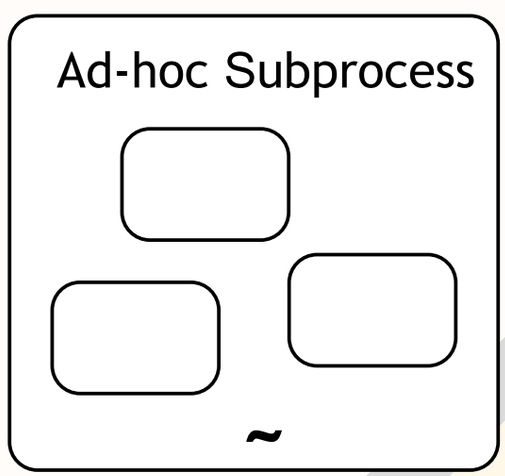


Multiple Instances of the same activity are started in parallel or sequentially, e.g. for each line item in an order.



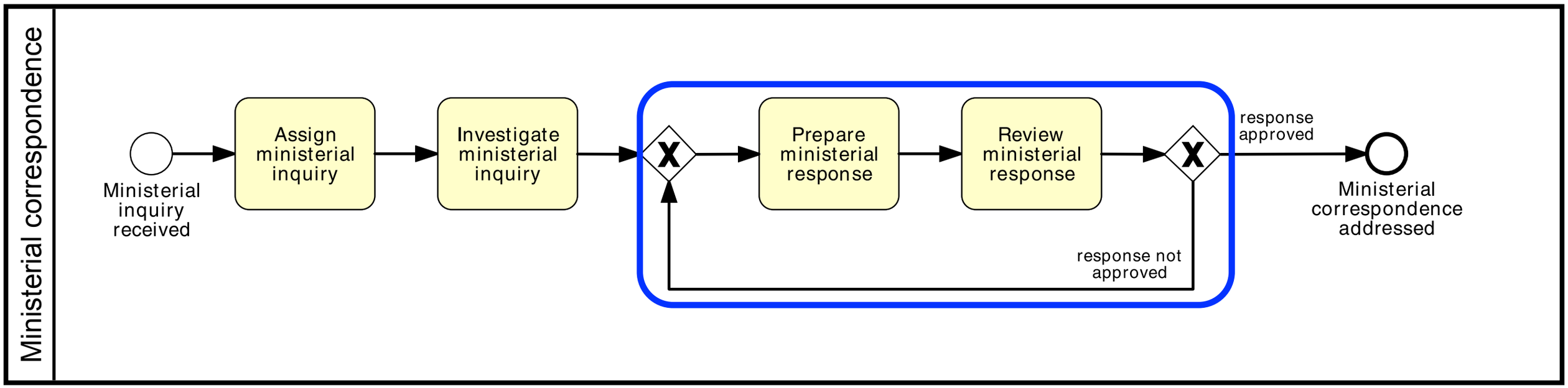


Loop Activity is iterated if a loop condition is true. The condition is either tested before or after the activity execution.

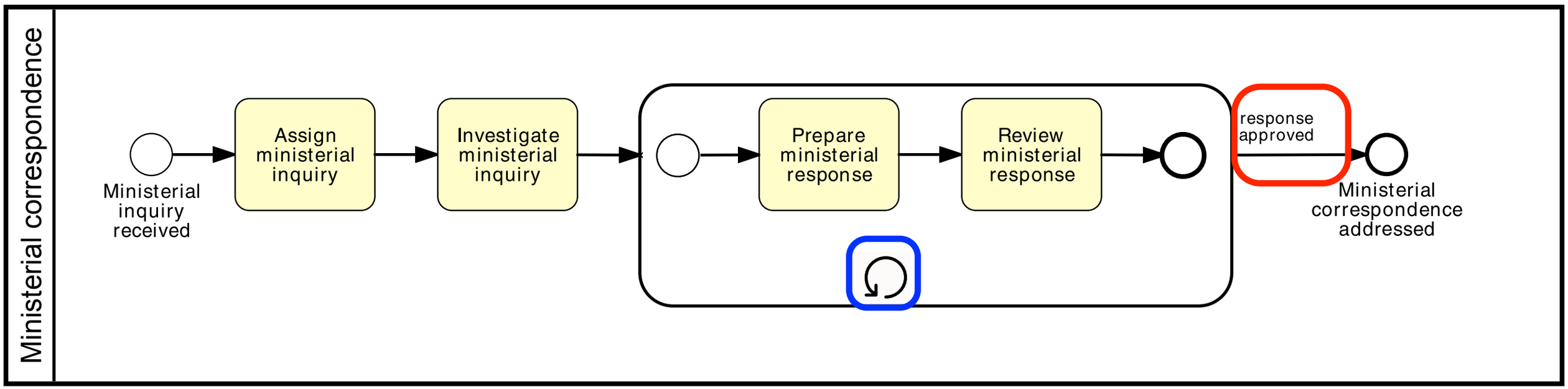


Ad-hoc Subprocesses contain tasks only. Each task can be executed arbitrarily often until a completion condition is fulfilled.

Loop marker: ministerial correspondence



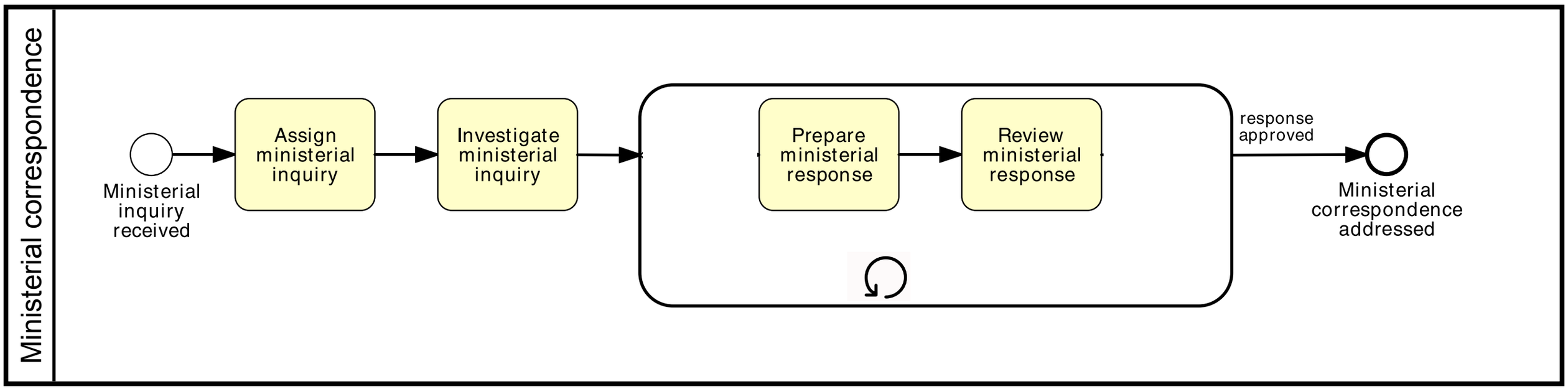
Loop marker: ministerial correspondence



the loop-symbol decoration marks the possible repetition of the sub-process activity

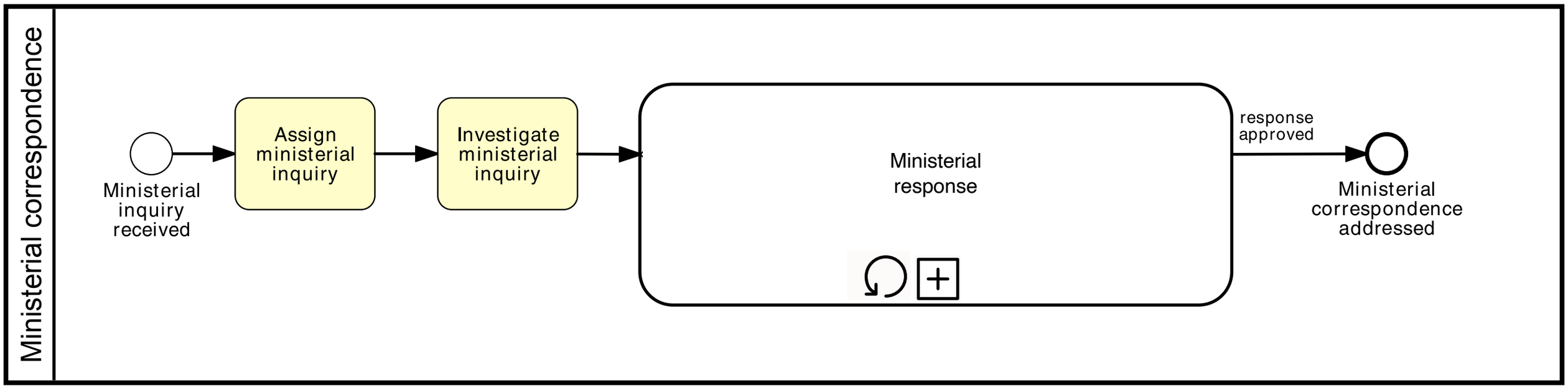
it is important to define exit conditions from loops!

Loop marker: ministerial correspondence



we can further simplify the inner process
(implicit start / end)

Loop marker: ministerial correspondence

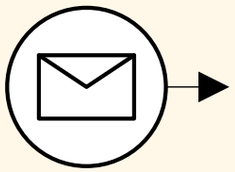


we can hide internal details

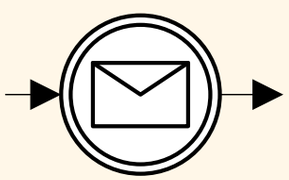
Catching and throwing

An event can catch a **trigger** or throw a **result**
Internal markers denote the trigger or result

Catching

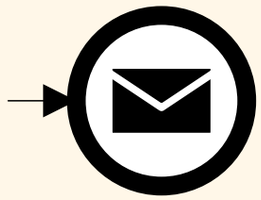


Start Event: Catching an event starts a new process instance.

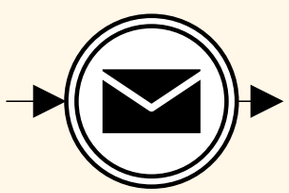


Intermediate Event (catching): The process can only continue once an event has been caught.

Throwing

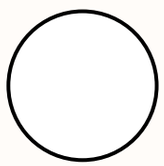
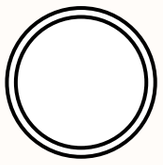
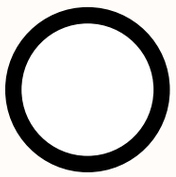


End Event: An event is thrown when the end of the process is reached.

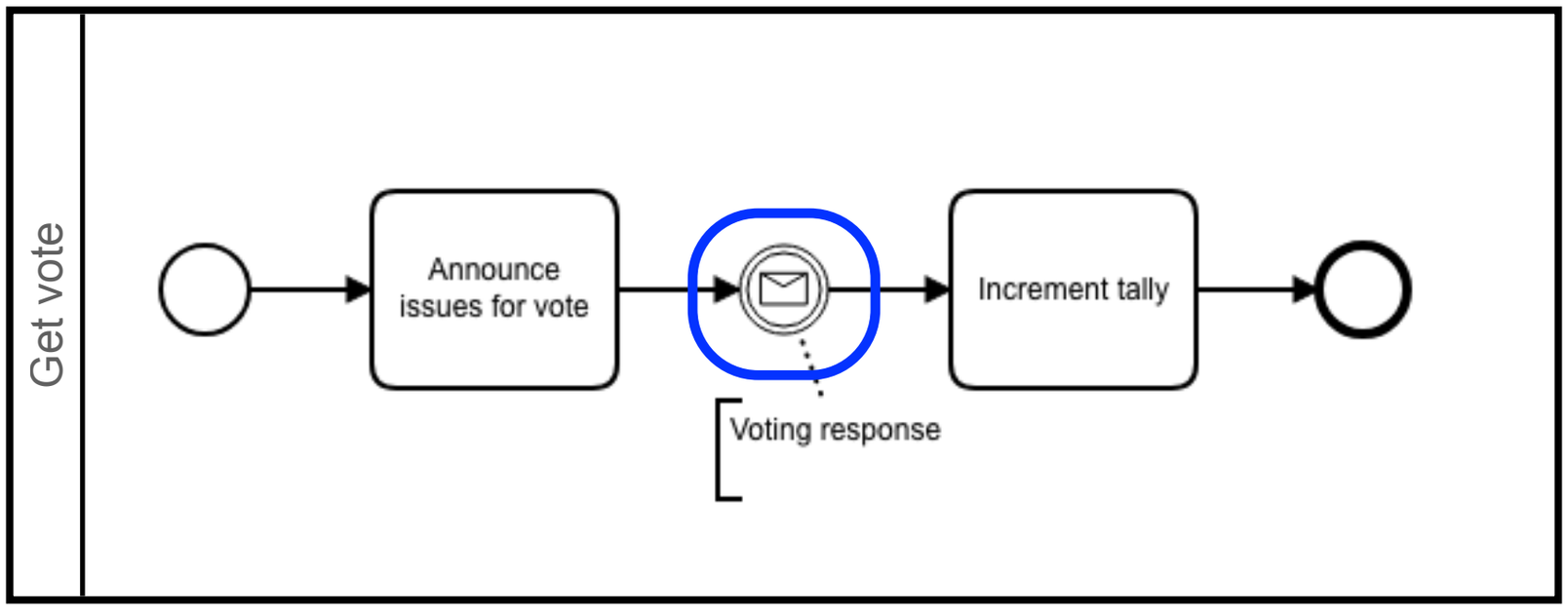


Intermediate Event (throwing): An event is thrown and the process continues.

Some internal markers

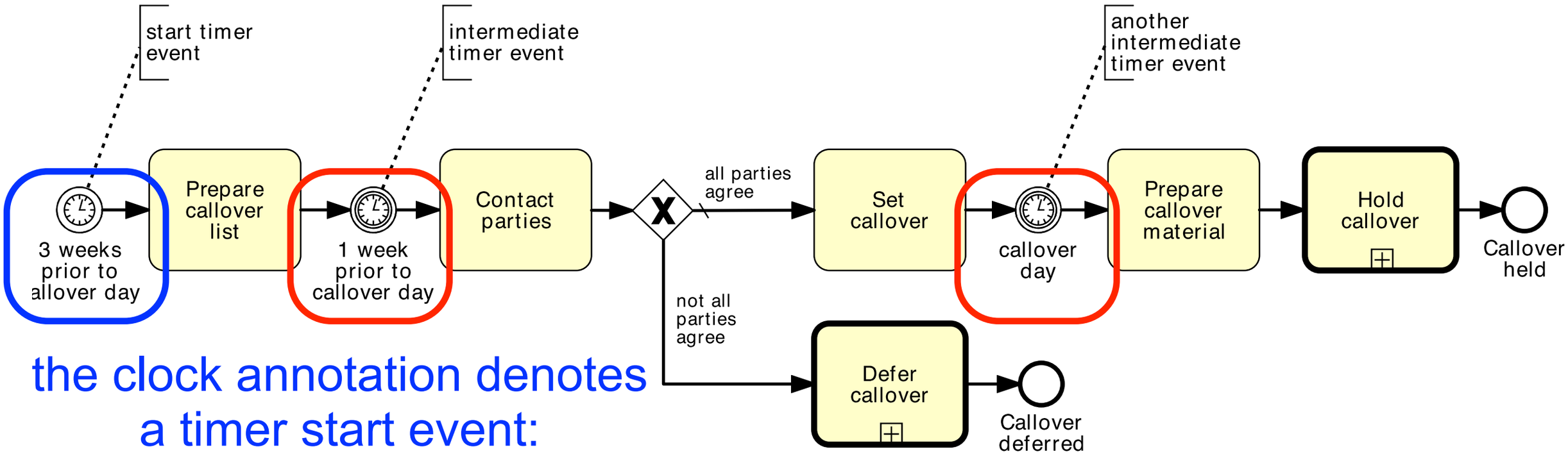
	Start	Intermediate		End	
	Catching		*Throwing*		
Plain					Untyped events, typically showing where the process starts or ends.
Message					Receiving and sending messages.
Timer					Cyclic timer events, points in time, time spans or timeouts.
Error					Catching or throwing named errors.
Terminate					Triggering the immediate termination of a process.

Process break (event waiting)



the envelope annotation denotes an intermediate message event: it signals the receipt of a message

Timer events: small claims tribunal



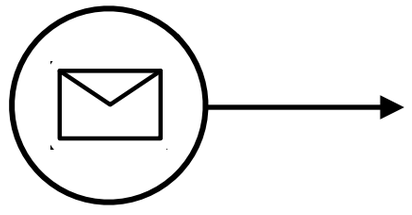
the clock annotation denotes a timer start event: an instance of the process is created when some temporal event happens

the clock annotation denotes a timer intermediate event: the process is blocked until a time-out expires

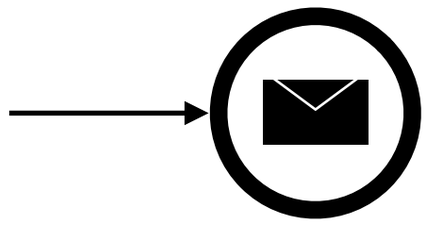
Collaboration diagrams (and message passing)

Message annotated events and activities

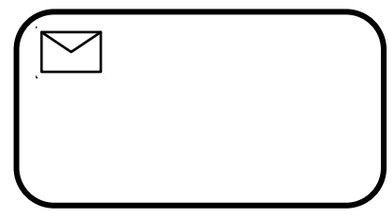
A start event can be annotated with a white-envelope:
a process instance is created
when a certain message is received



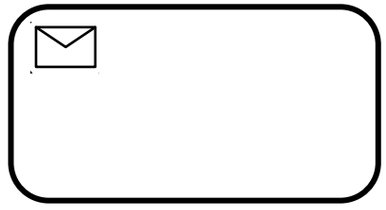
An end event can be annotated with a black-filled envelope:
when the process ends
a message is sent



Intermediate events and activities can be annotated
with both kinds of envelope
white = receipt of a message,
black = the sending of a message



Events vs Activities

Should we use  or  ?

No clear distinction is made, but typically

events are instantaneous

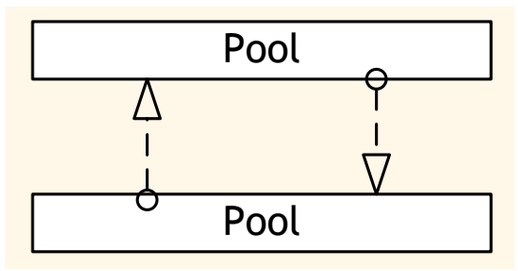
activities take time (have a duration)

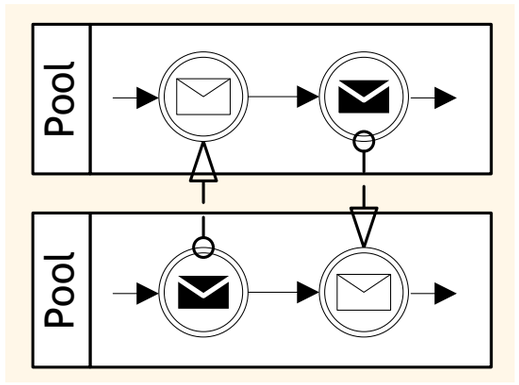
Collaboration

A **collaboration** contains two or more pools,
each representing a **participant** in the collaboration

A pool may be collapsed or exhibit the process within

Each possible communication corresponds to
a **message flow** between pools
(or between objects from different pools)



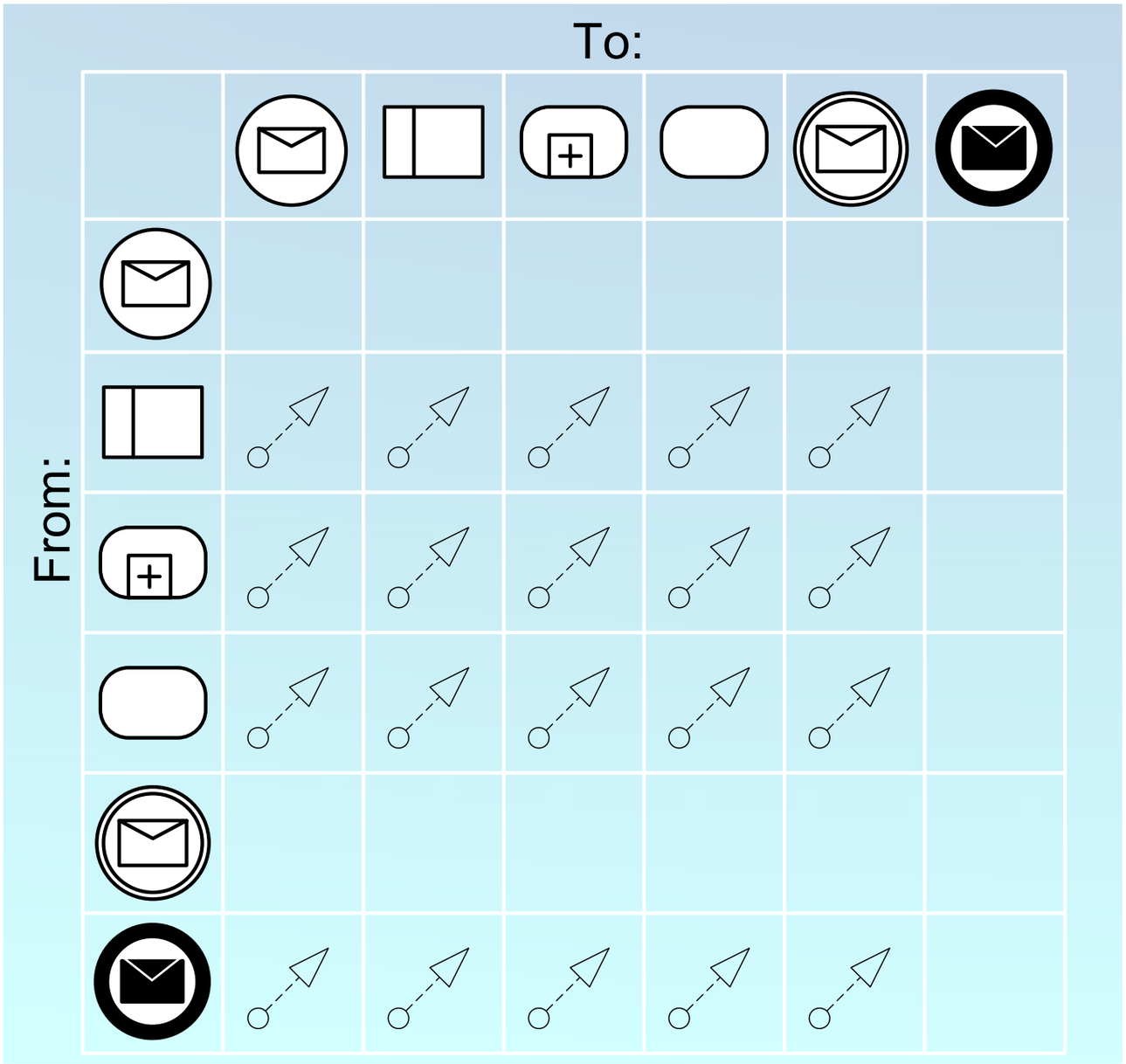


Message flow

A **message flow** represents communications (send/receive) between two separate participants (business entities or business roles)

A message flow is represented by a dashed line with a open arrowheads

Message flow requirements



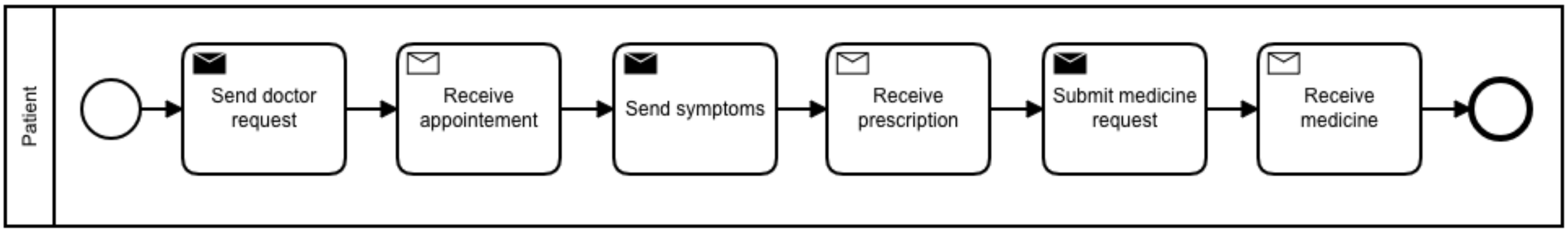
each event:
at most one message flow

each activity:
at most one message flow

each gateway:
no message flow!

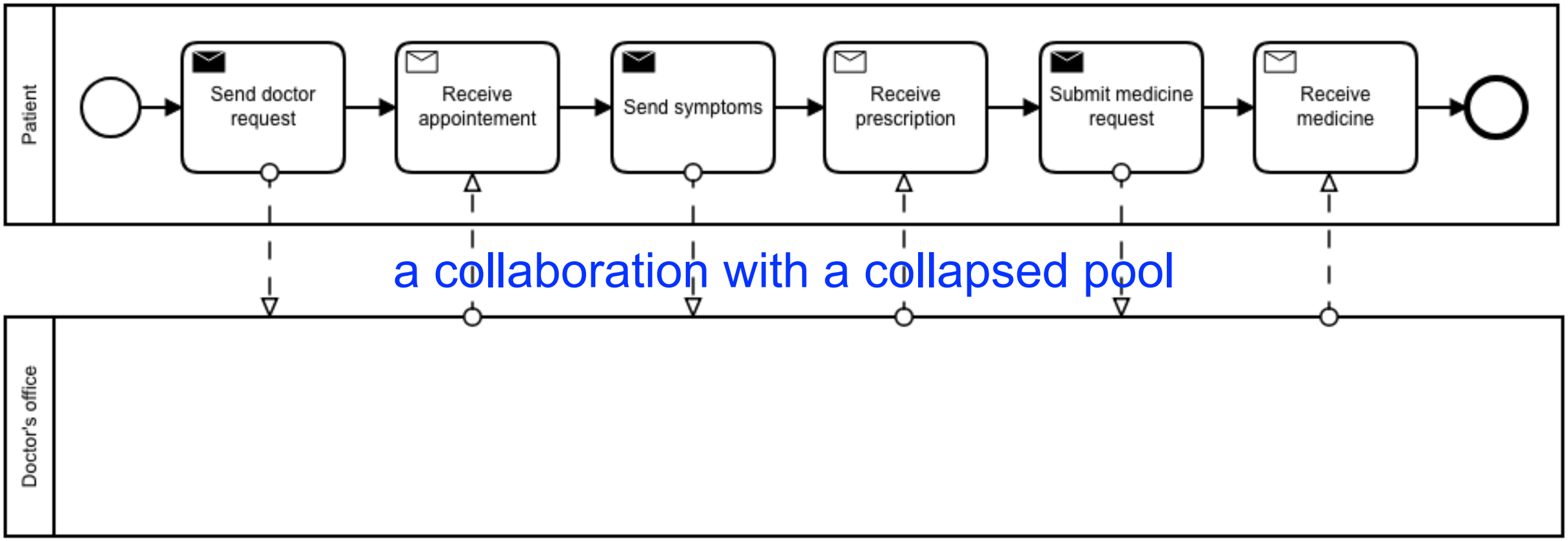
each pool:
any number of message flows

From processes to collaborations



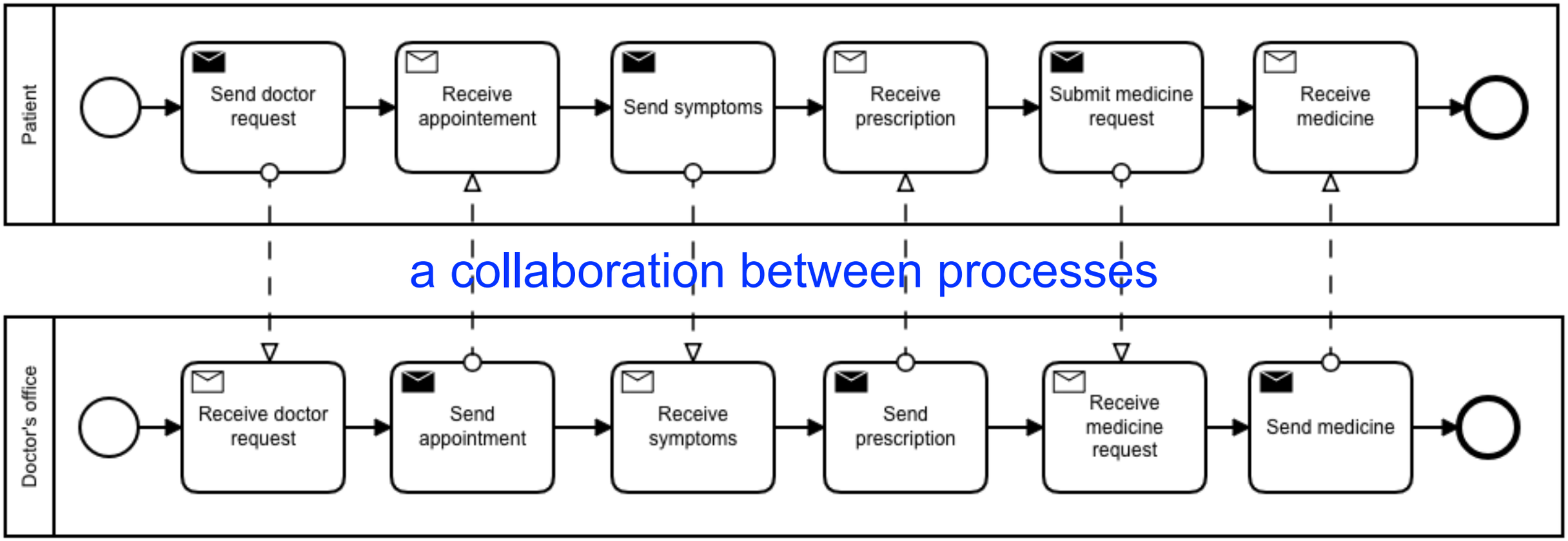
a stand-alone process

From processes to collaborations



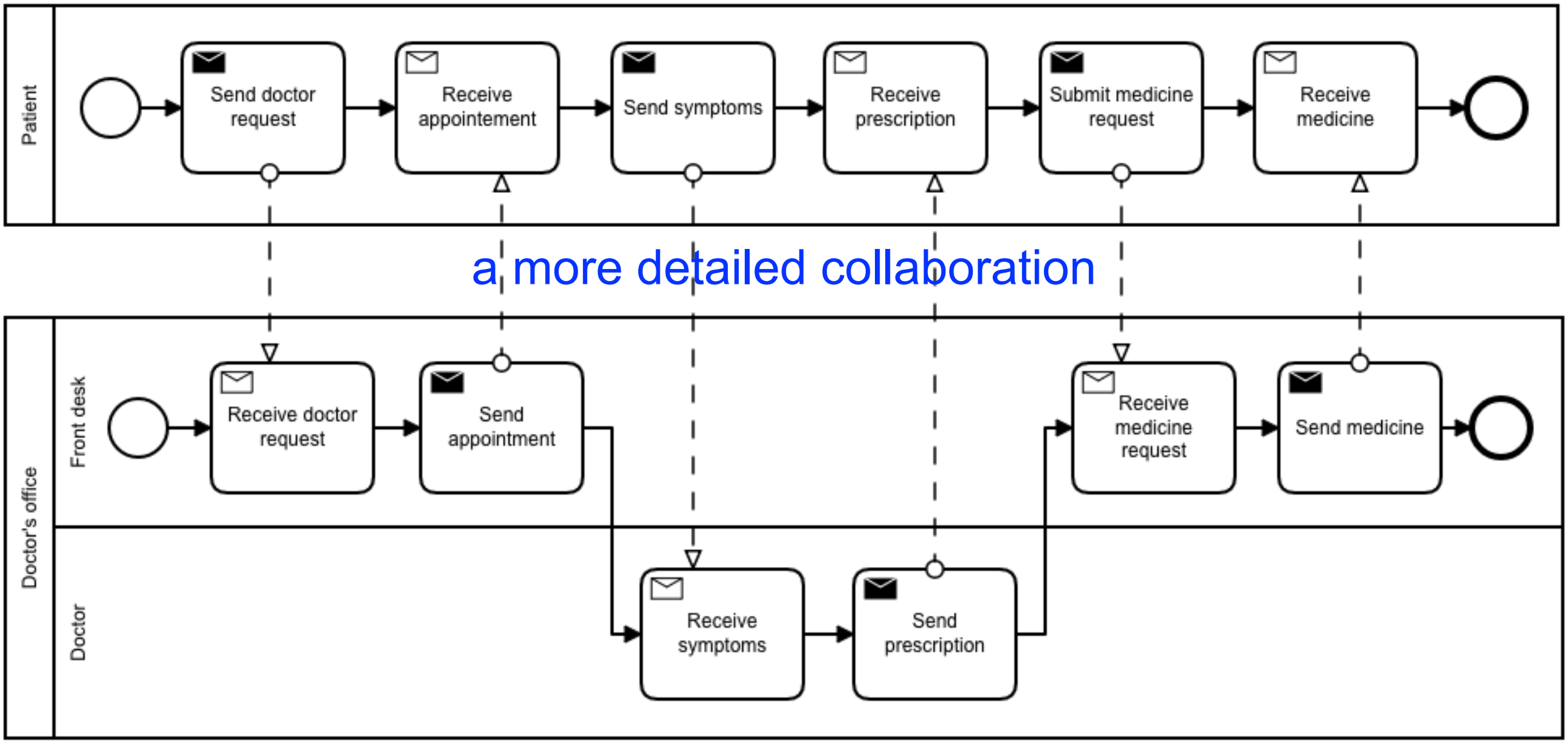
a collaboration with a collapsed pool

From processes to collaborations



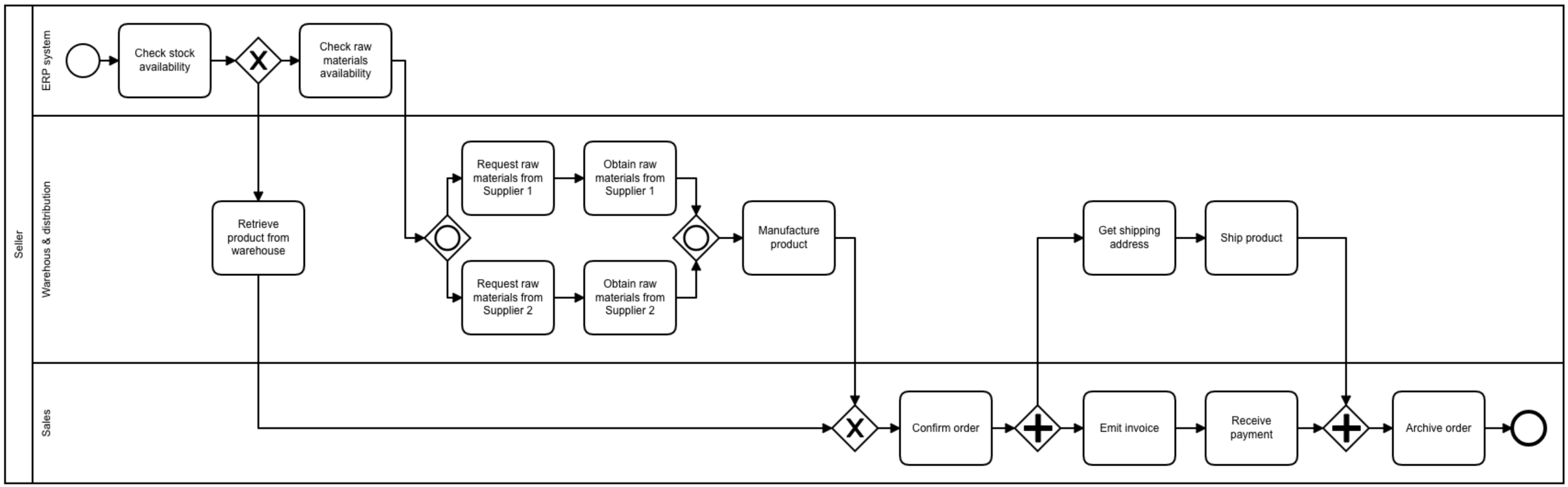
a collaboration between processes

From processes to collaborations

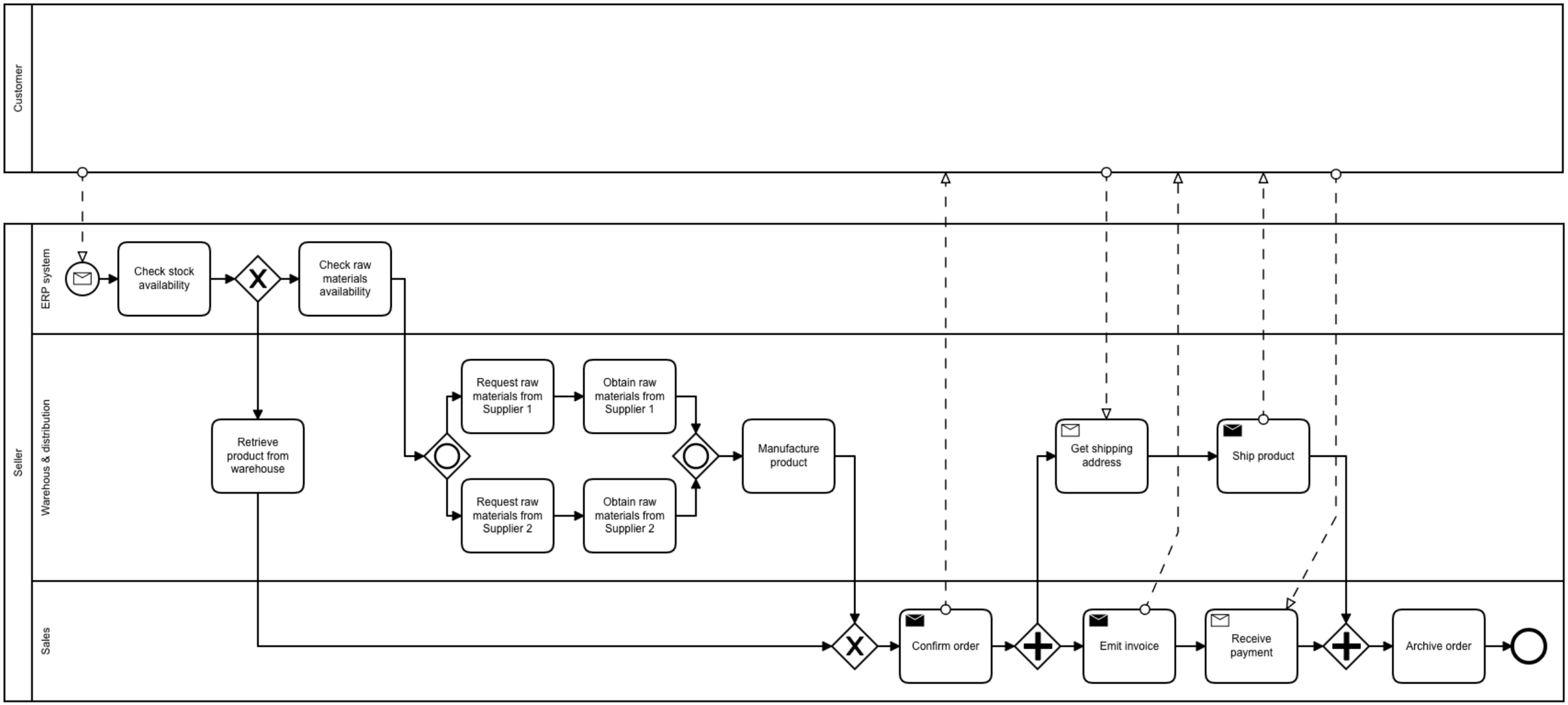


a more detailed collaboration

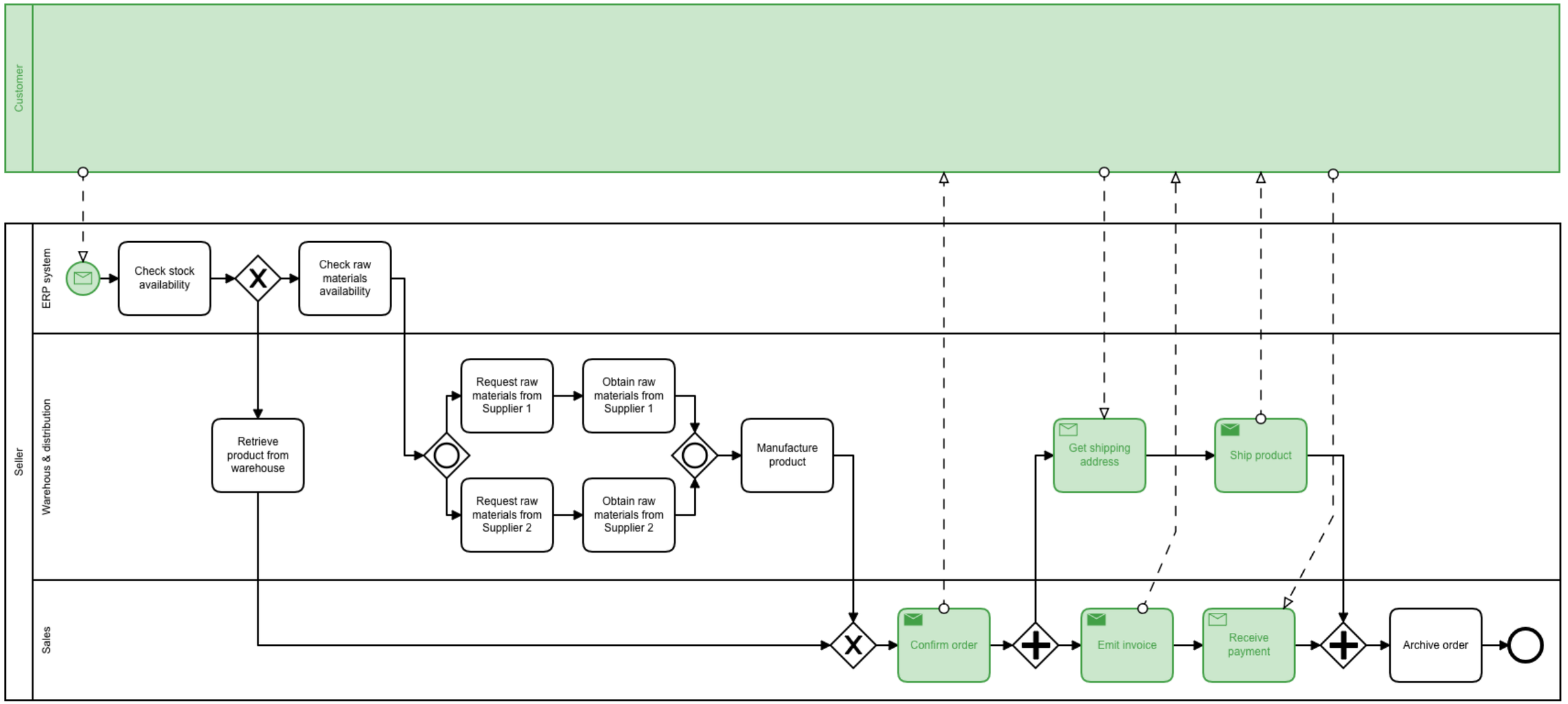
Example: Seller



Example: Seller & Customer



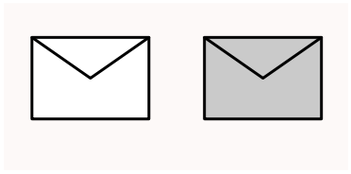
Example: Seller & Customer



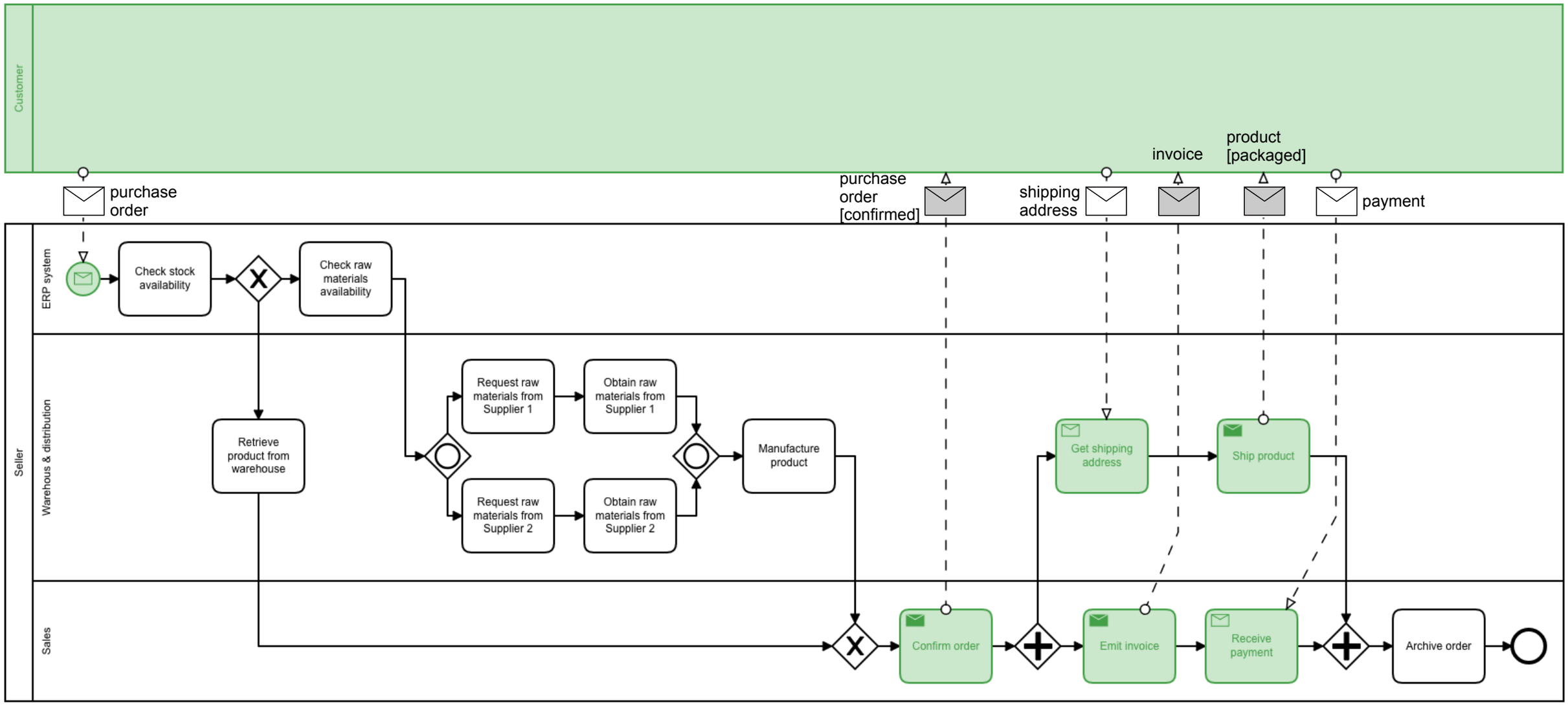
Artefacts: message data objects

A **message data object** depicts the data that are communicated between two participants

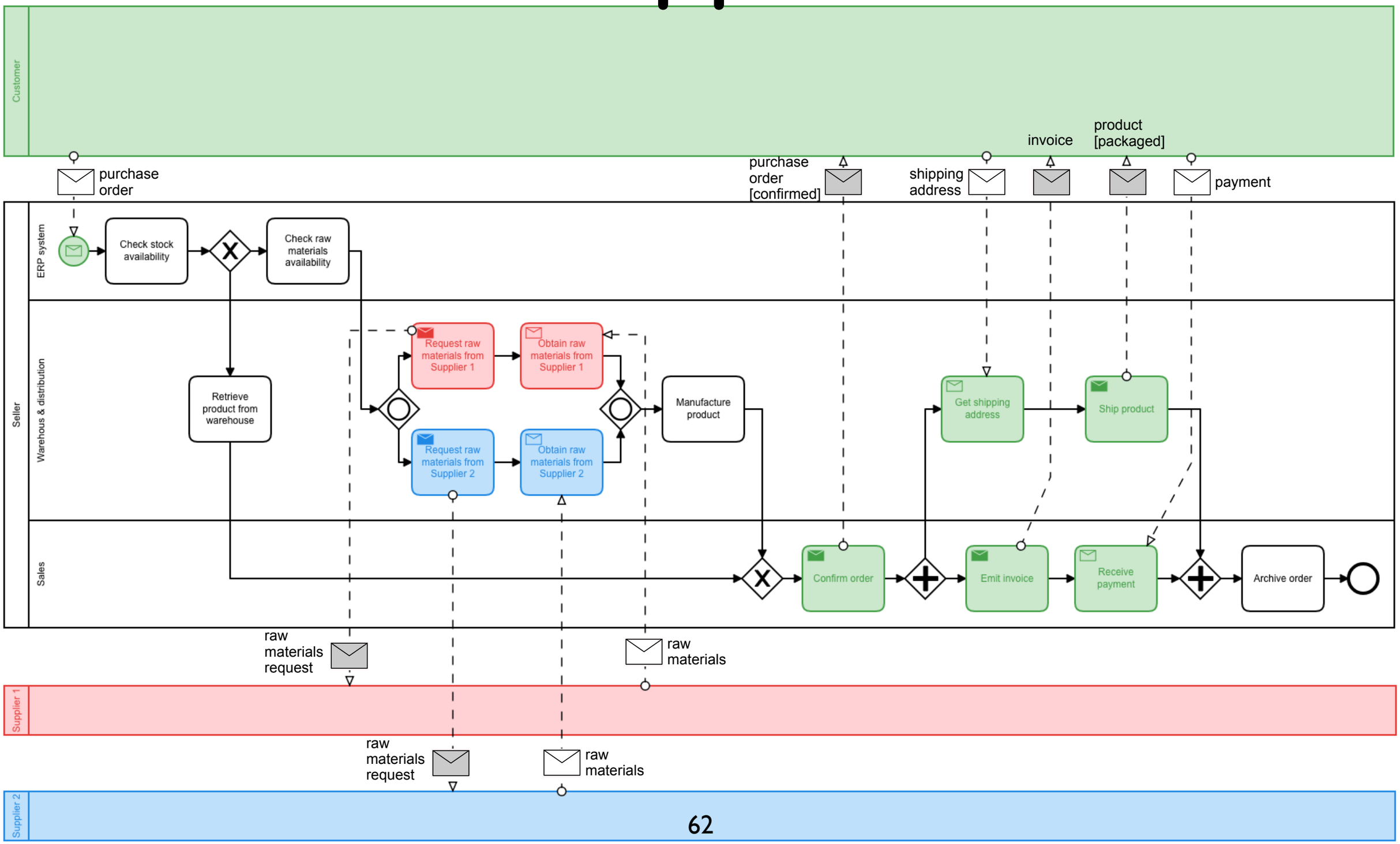
A message data object is represented as an envelope



Example: Seller & Customer

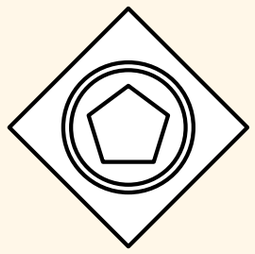


Example: Seller, Customer & Suppliers

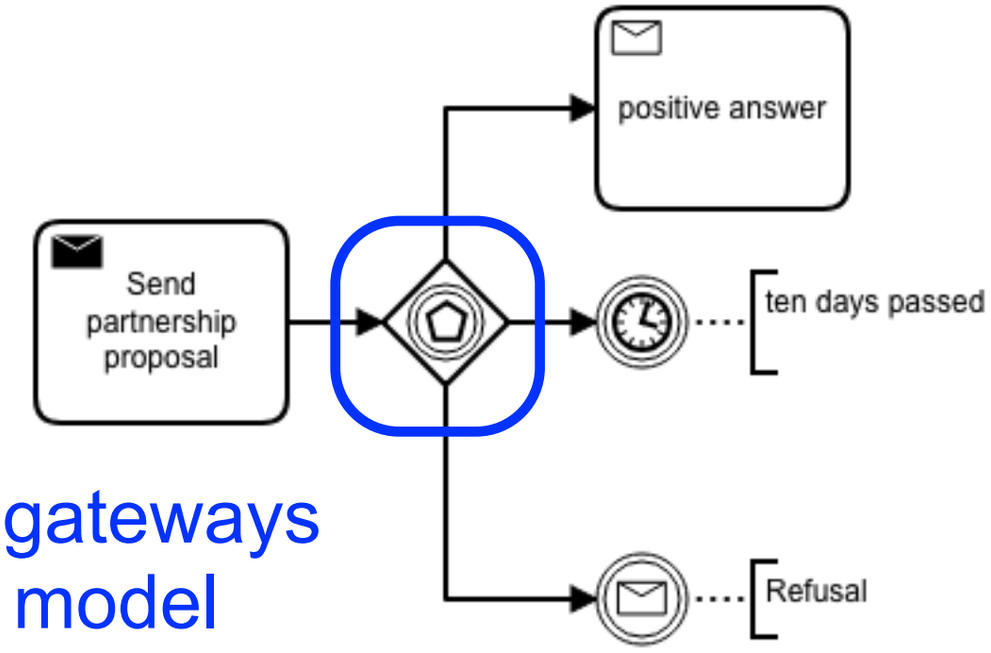


Deferred choice (event based decisions)

Event-based decisions

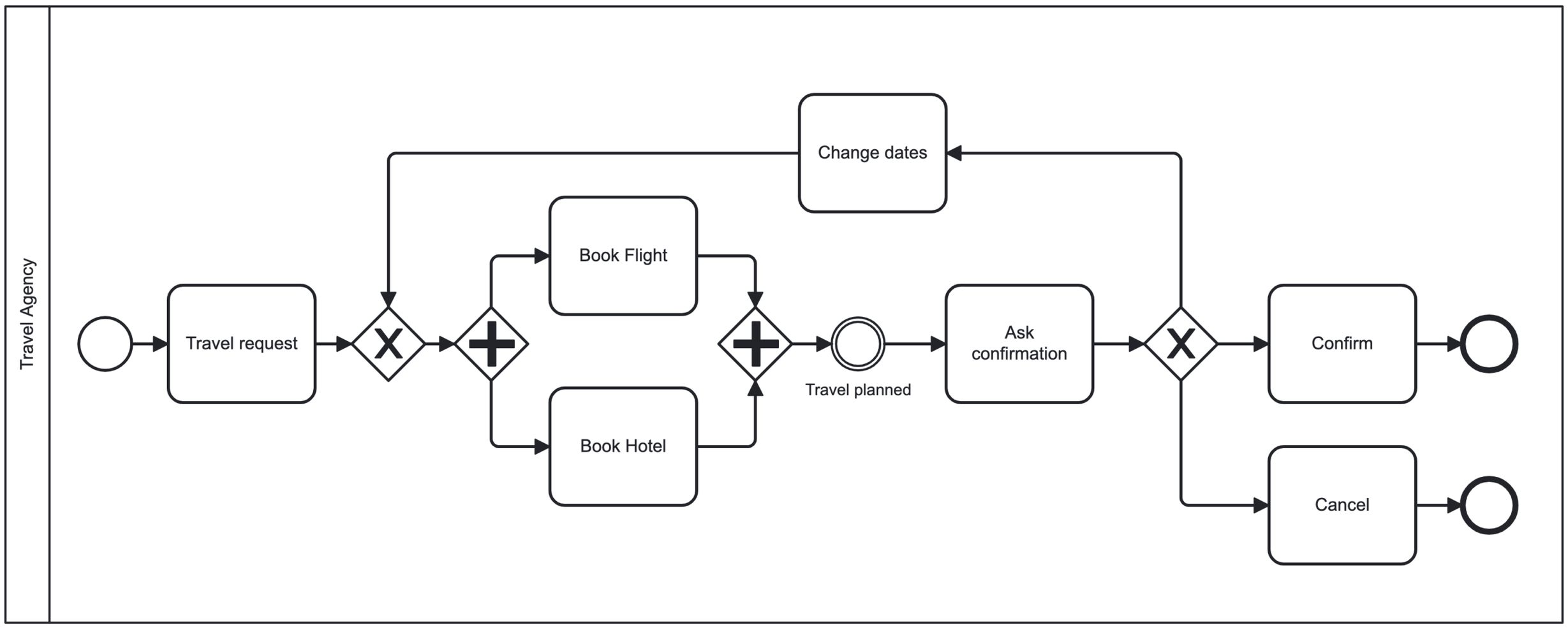


Event-based Exclusive Gateway
Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

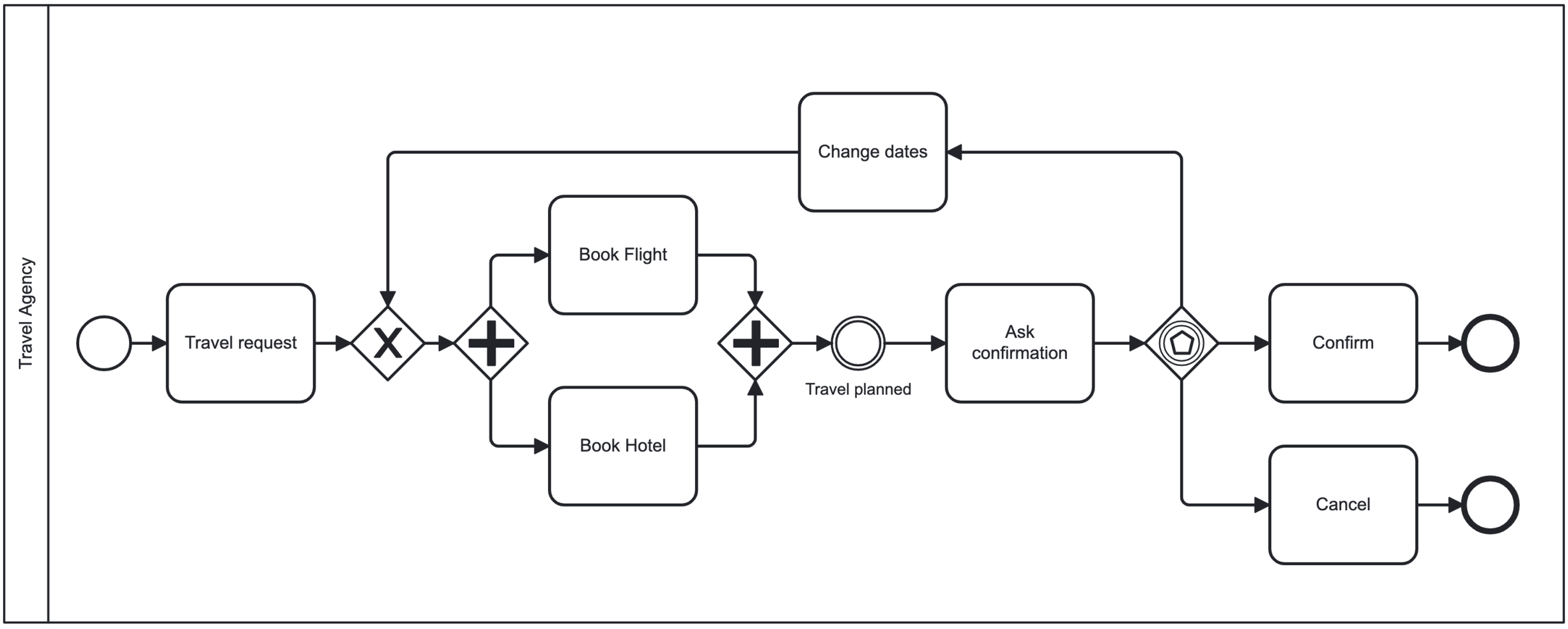


Event-based (split) gateways must be used to model decisions that depends on some external event

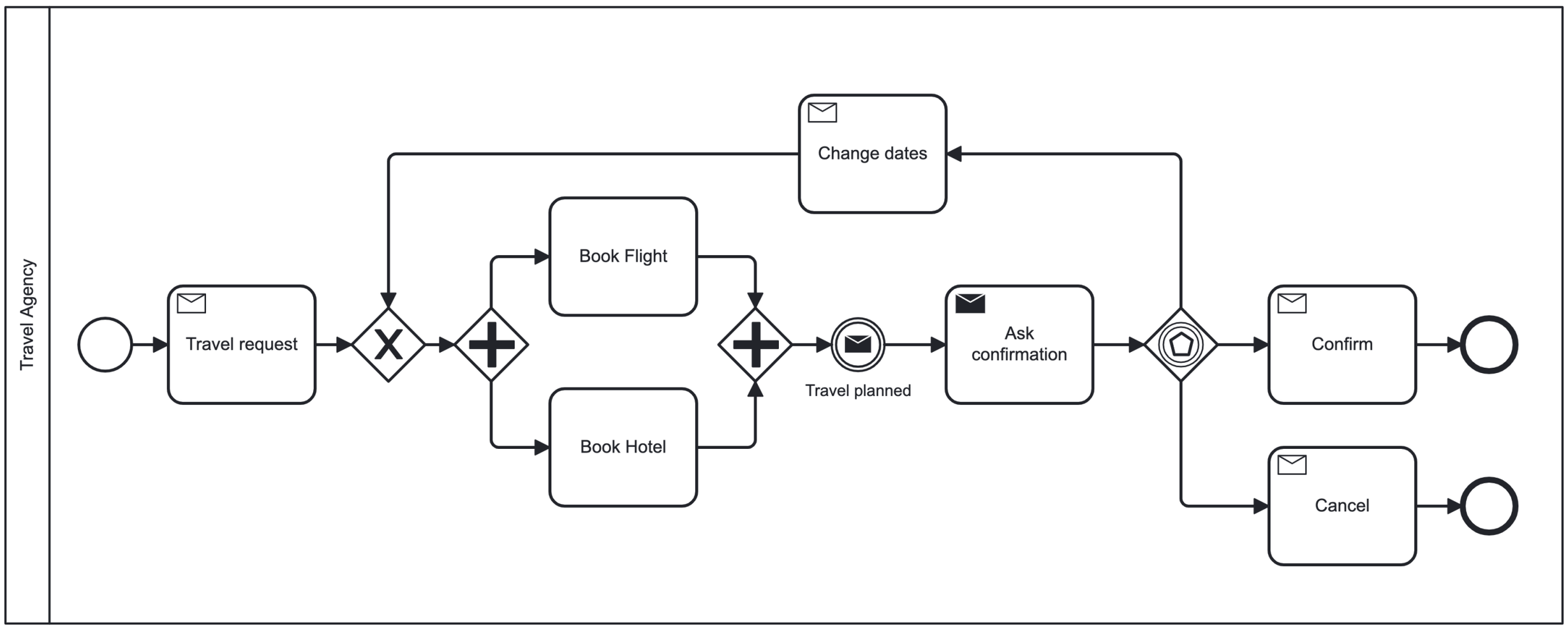
Travel agency example



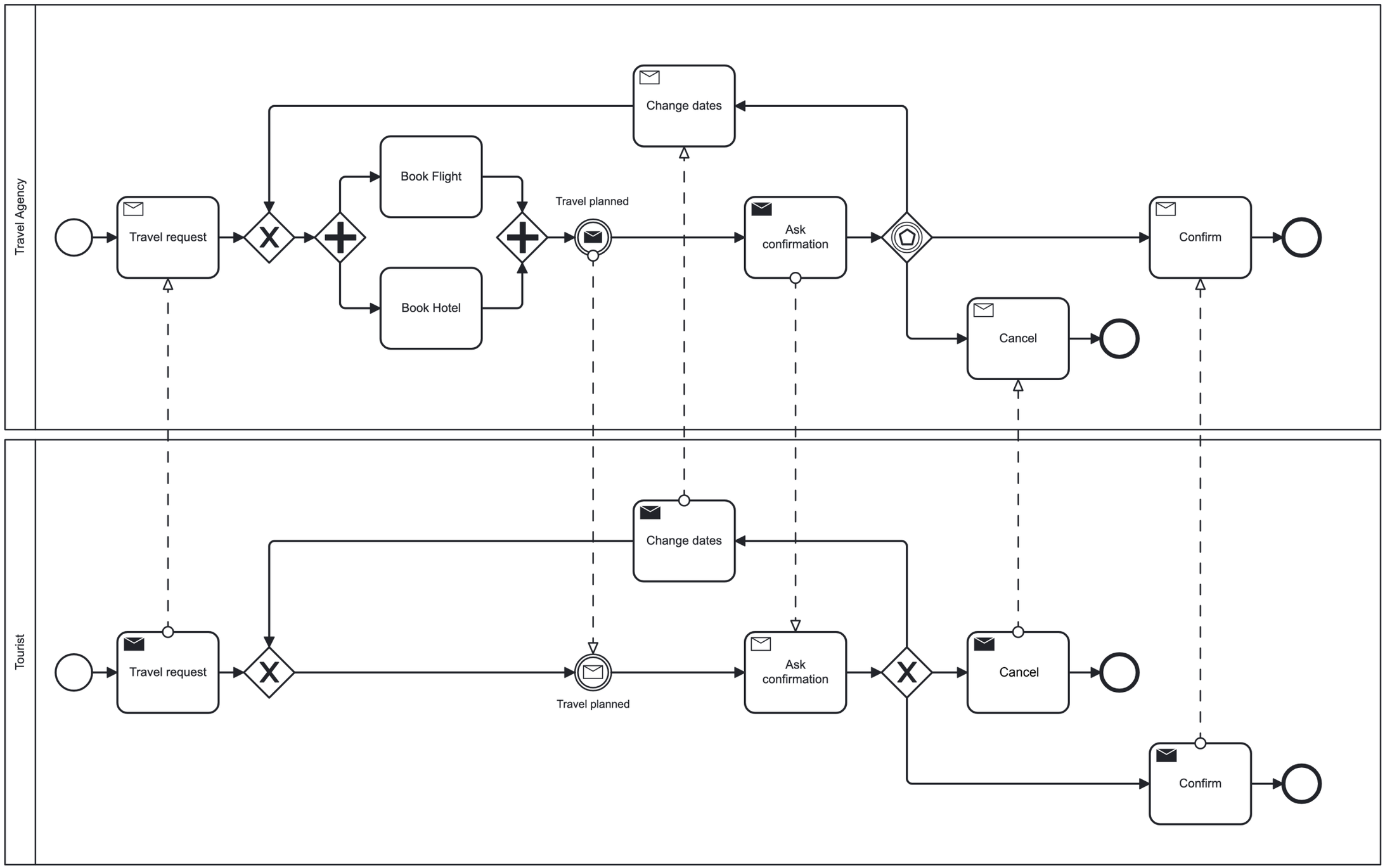
Travel agency example



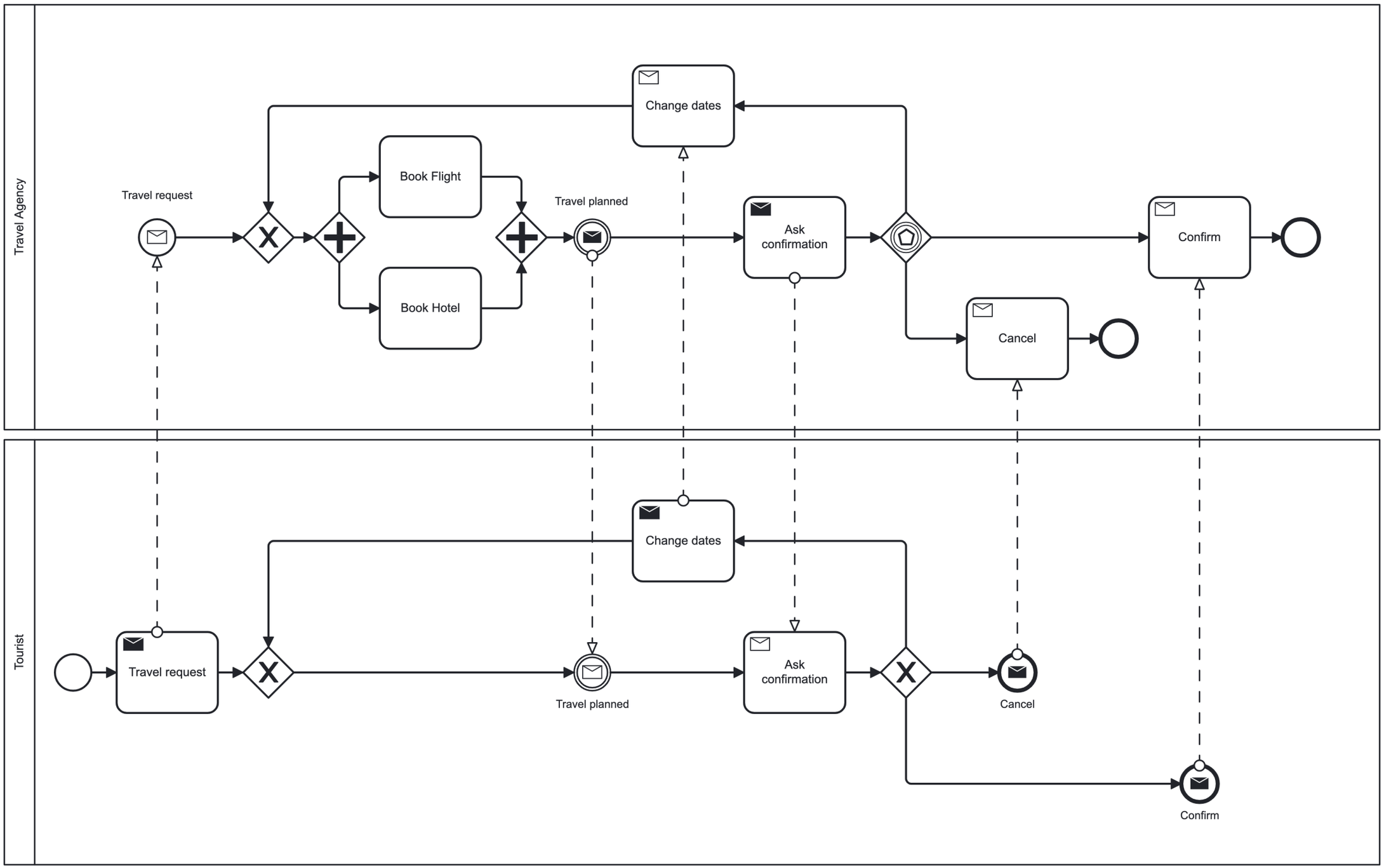
Travel agency example



Travel agency example

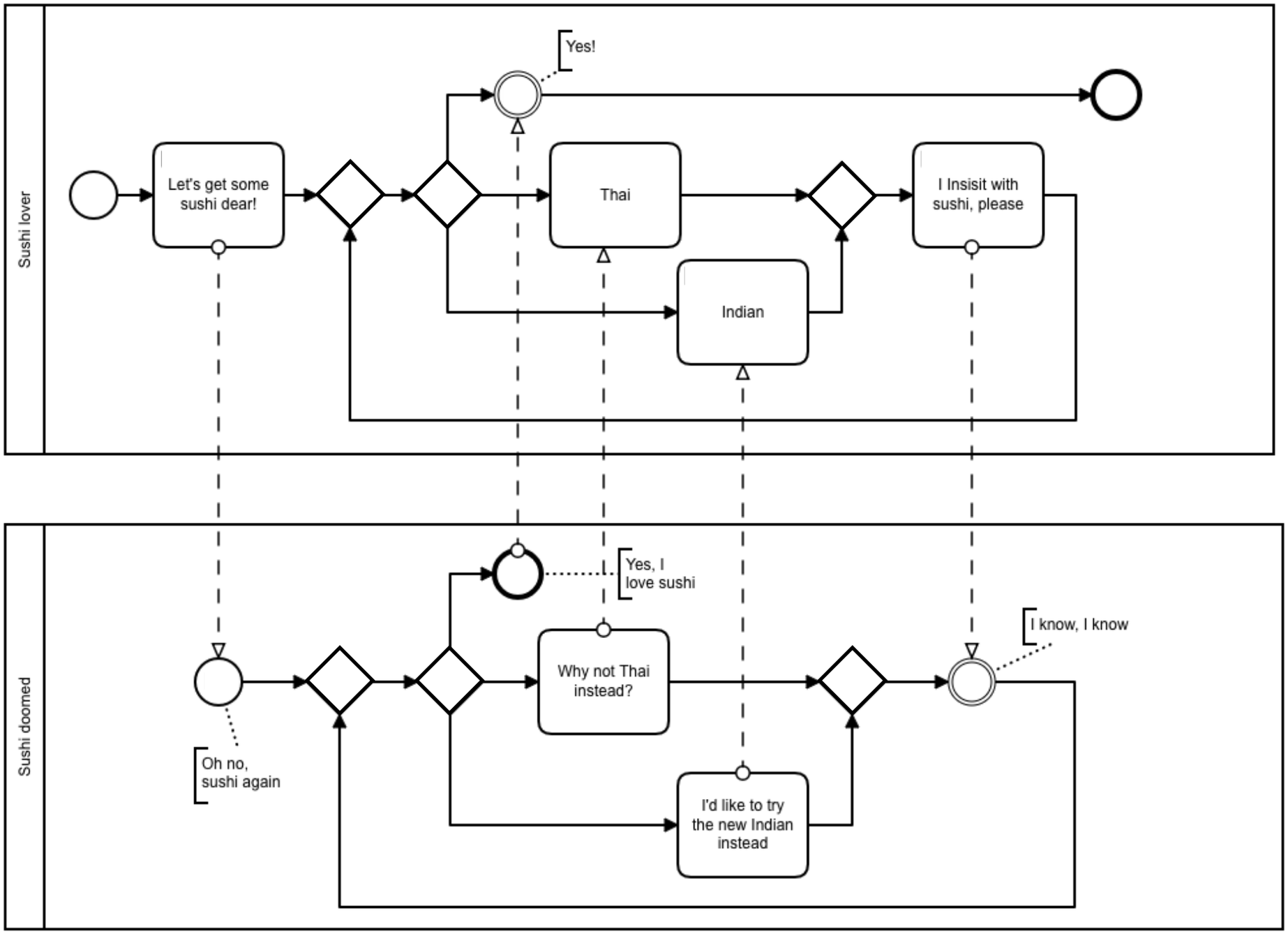


Travel agency example



A negotiation without choice

Which type annotations?



Some remarks

Lanes are often used to separate activities associated with a specific company function or role

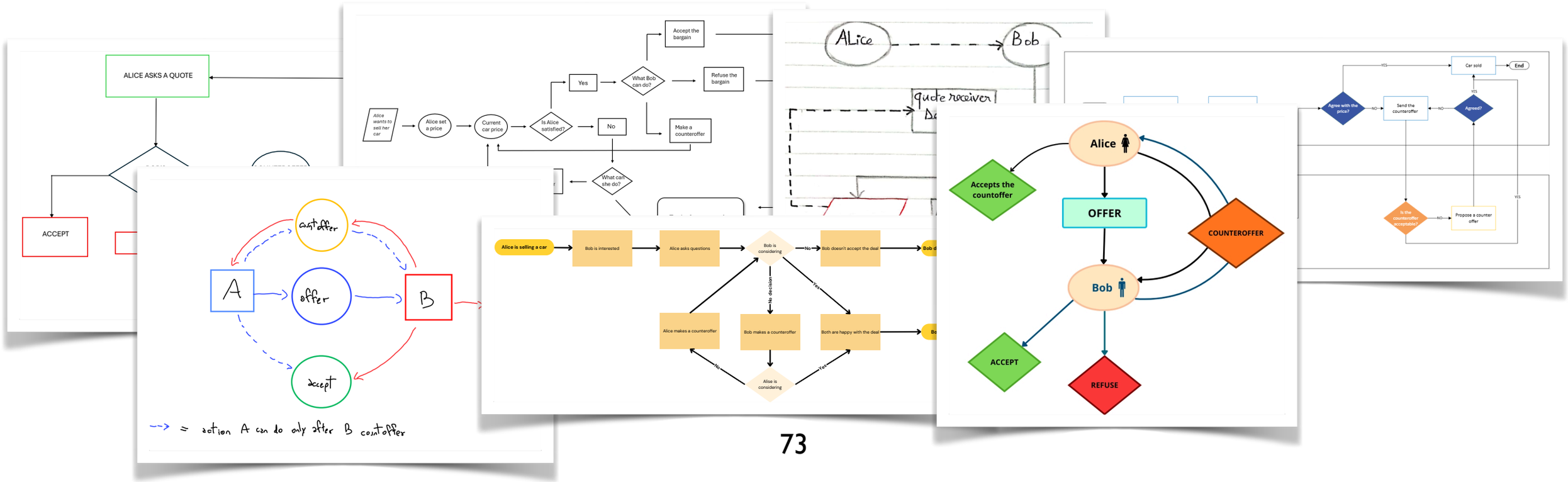
Sequence flow cannot cross the boundaries of a pool (it can cross lanes in the pool)

Message flow cannot connect flow objects in the same pool

Exercise: key features

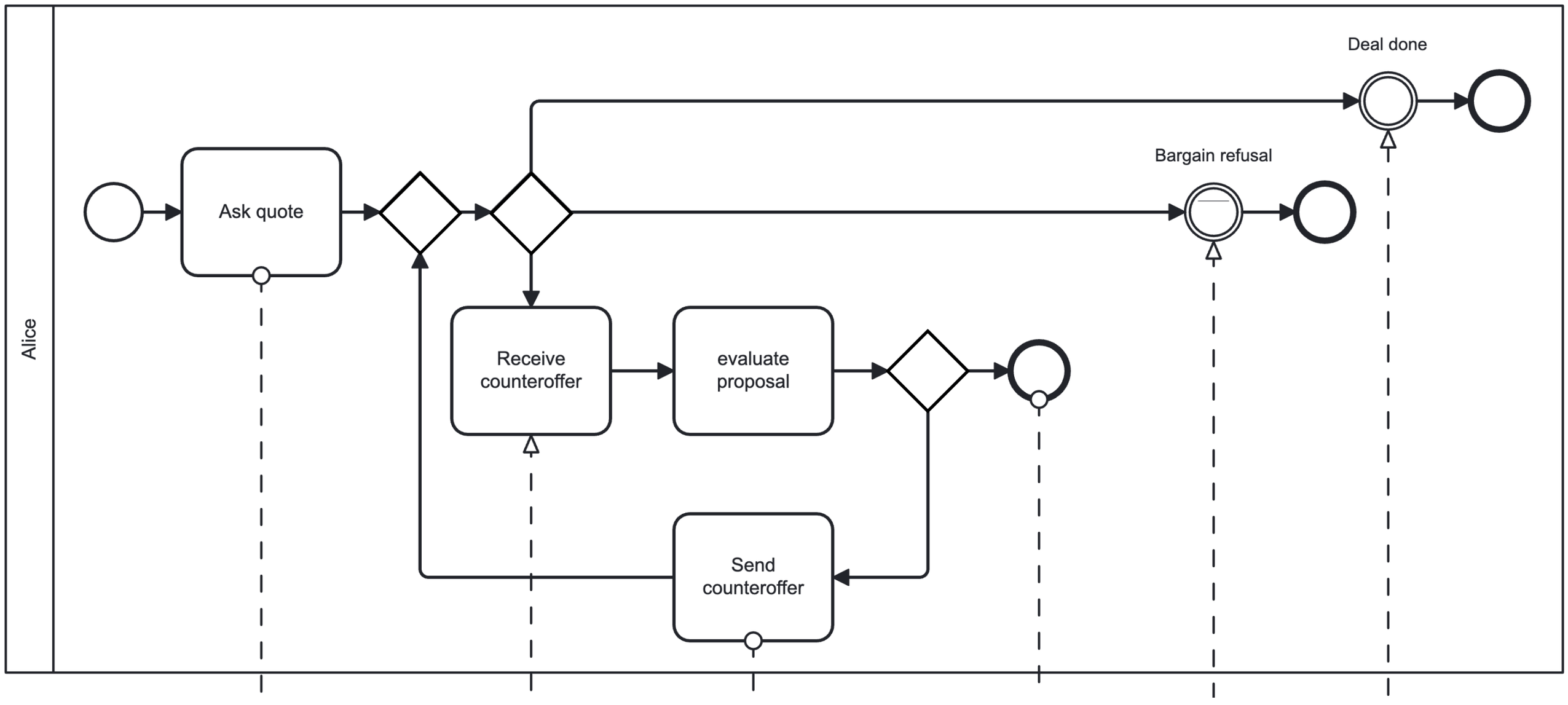
Draw the BPMN collaboration diagram
for the Alice-Bob car-selling scenario

Alice wants to sell her car, Bob is interested in buying it.
Alice asks some quote.
Bob can accept the bargain, refuse it or make a counteroffer.
Alice can accept or make a counteroffer and so on,
Until either the bargain is accepted or refused.



Exercise: key features

Draw the BPMN collaboration diagram
for the Alice-Bob car-selling scenario

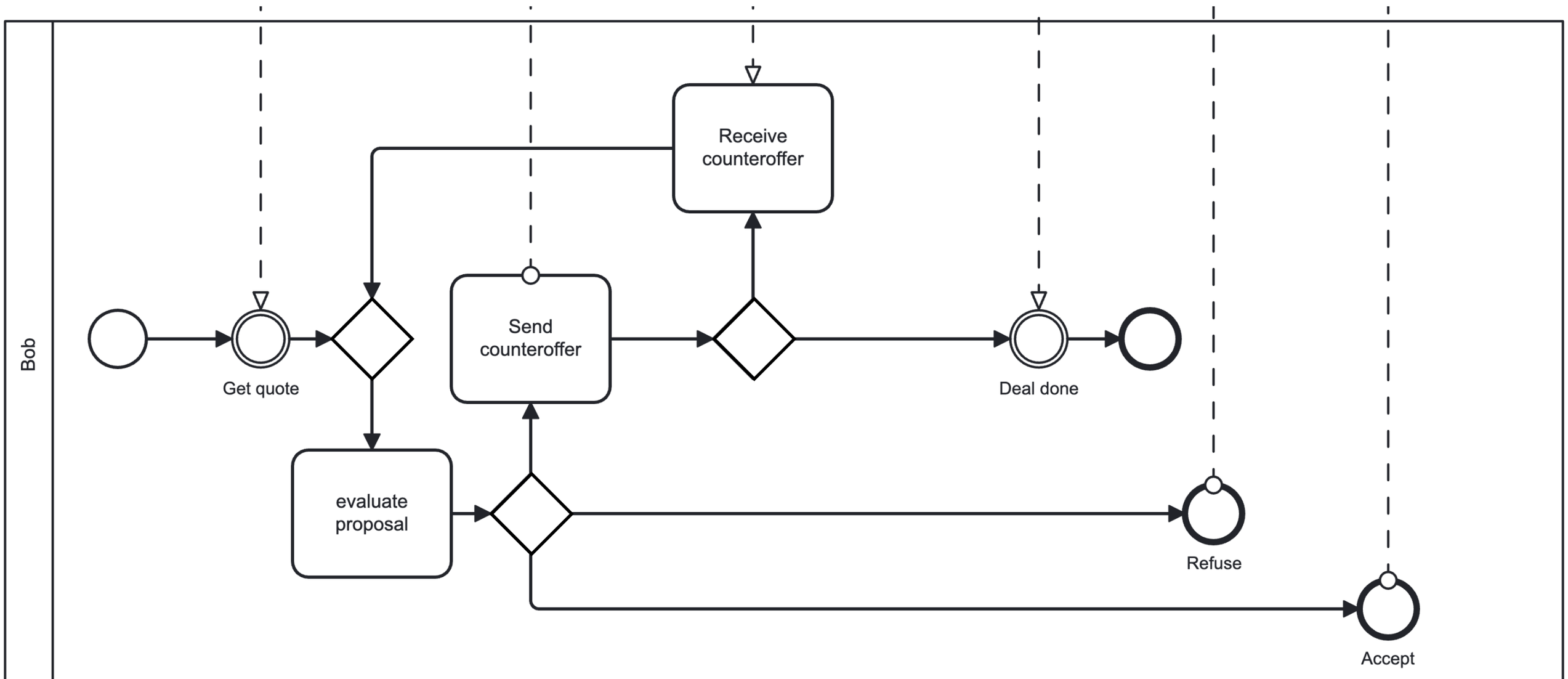


Which type annotations?

Exercise: key features

Draw the BPMN collaboration diagram
for the Alice-Bob car-selling scenario

Which type annotations?

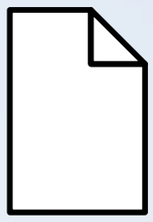


More artefacts (data-objects, groups)

Data object

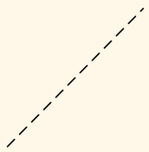
A **data object** represents information flowing through the process, such as documents, emails and letters

A data object is often represented by the usual file icon

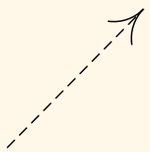


Data objects provide information about what activities are required to be triggered and/or what they produce. They are considered as Artefacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process. The state of the data object should also be set.

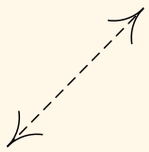
Association, again



Attaching a data object with an **Undirected Association** to a sequence flow indicates hand-over of information between the activities involved.



A **Directed Association** indicates information flow. A data object can be read at the start of an activity or written upon completion.



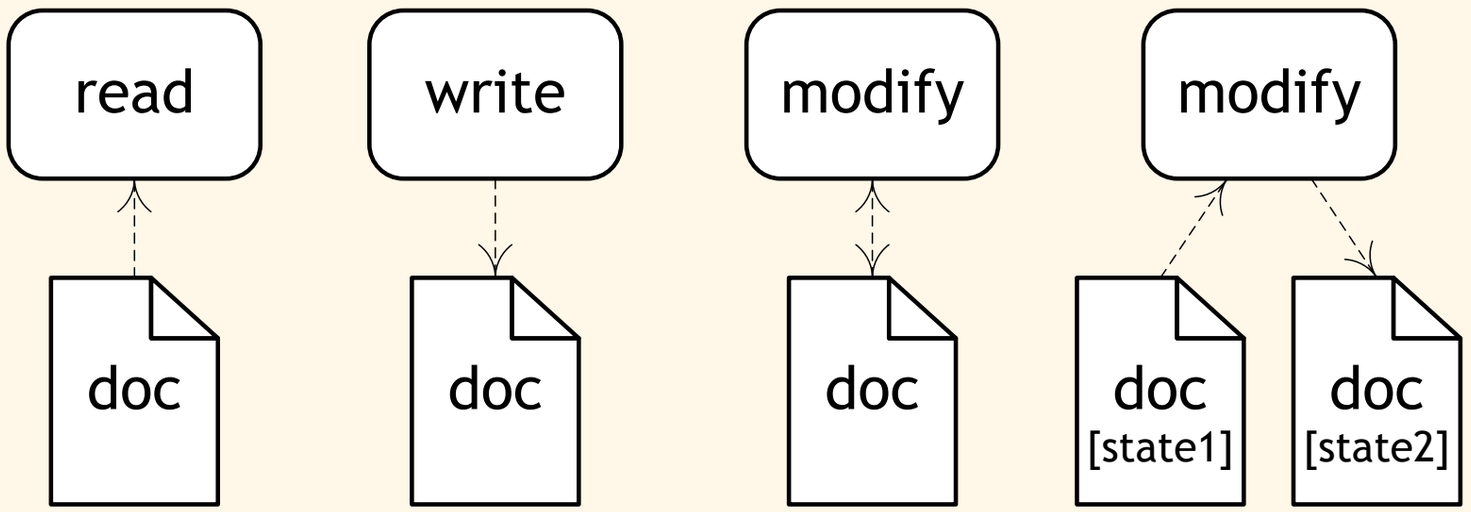
A **Bidirected Association** indicates that the data object is modified, i.e. read and written during the execution of an actvity.

Association, again

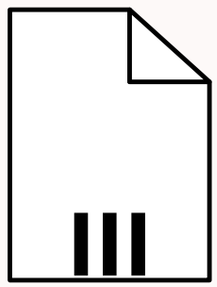
Attaching a data object with an **Undirected Association** to a sequence flow indicates hand-over of information between the activities involved.

A **Directed Association** indicates information flow. A data object can be read at the start of an activity or written upon completion.

A **Bidirected Association** indicates that the data object is modified, i.e. read and written during the execution of an actvity.



More data objects



A **Collection Data Object r**epresents a collection of information, e.g., a list of order items.

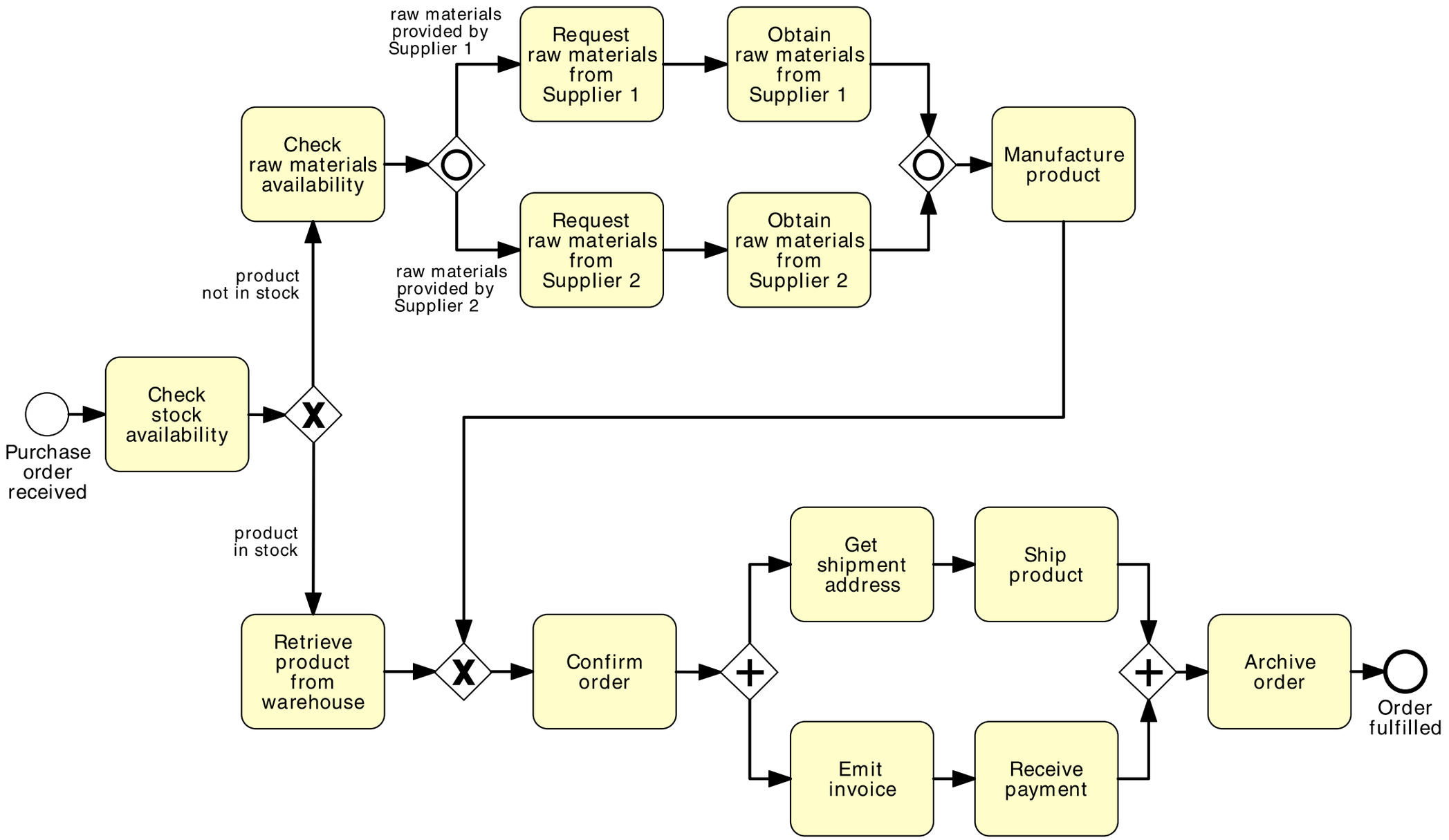
A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

Group

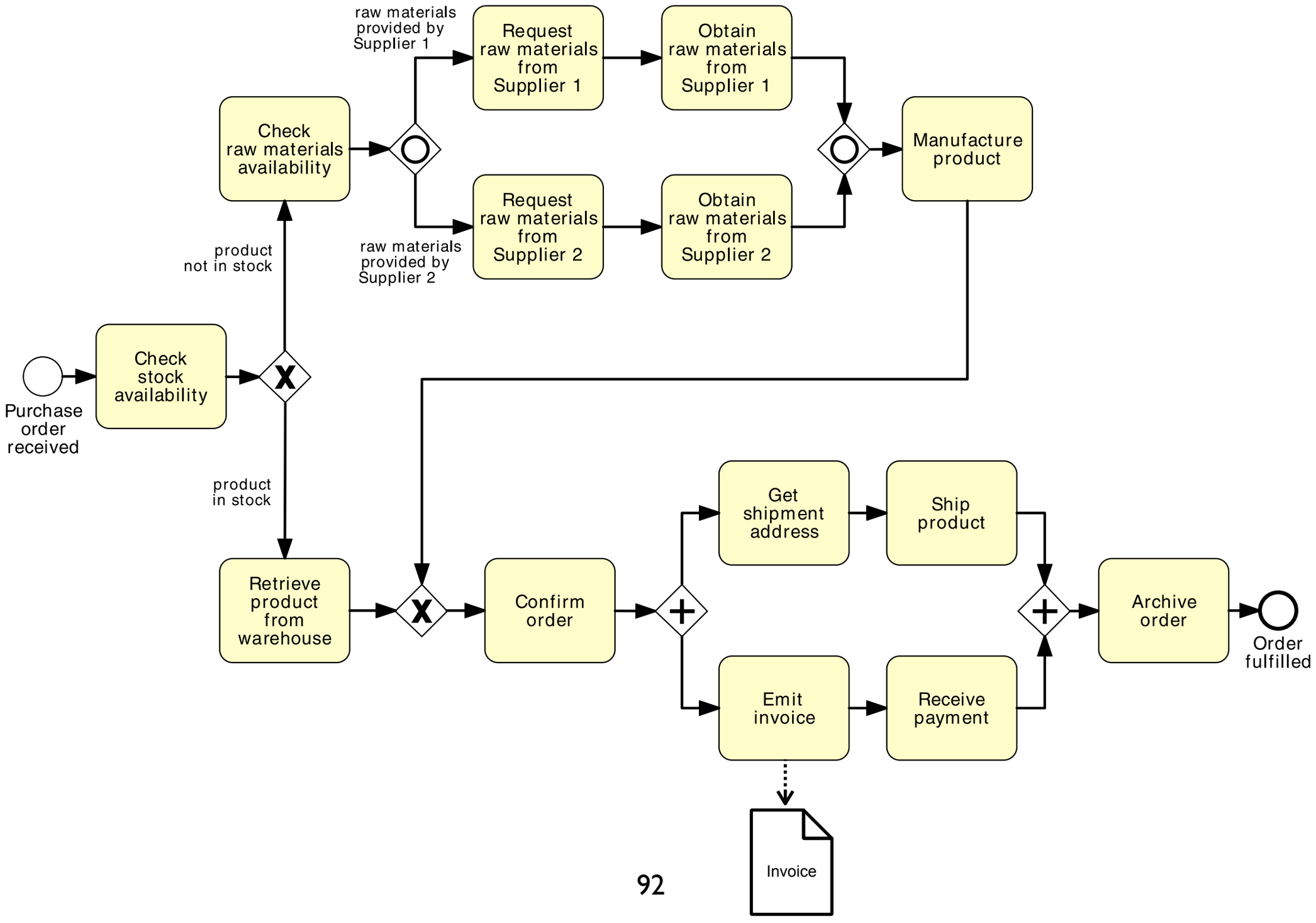
An arbitrary set of objects can form a **group**
(if they logically belong together)
it has non behavioural effect (only documentation)

A group is represented by
rounded corner rectangles with dashed lines

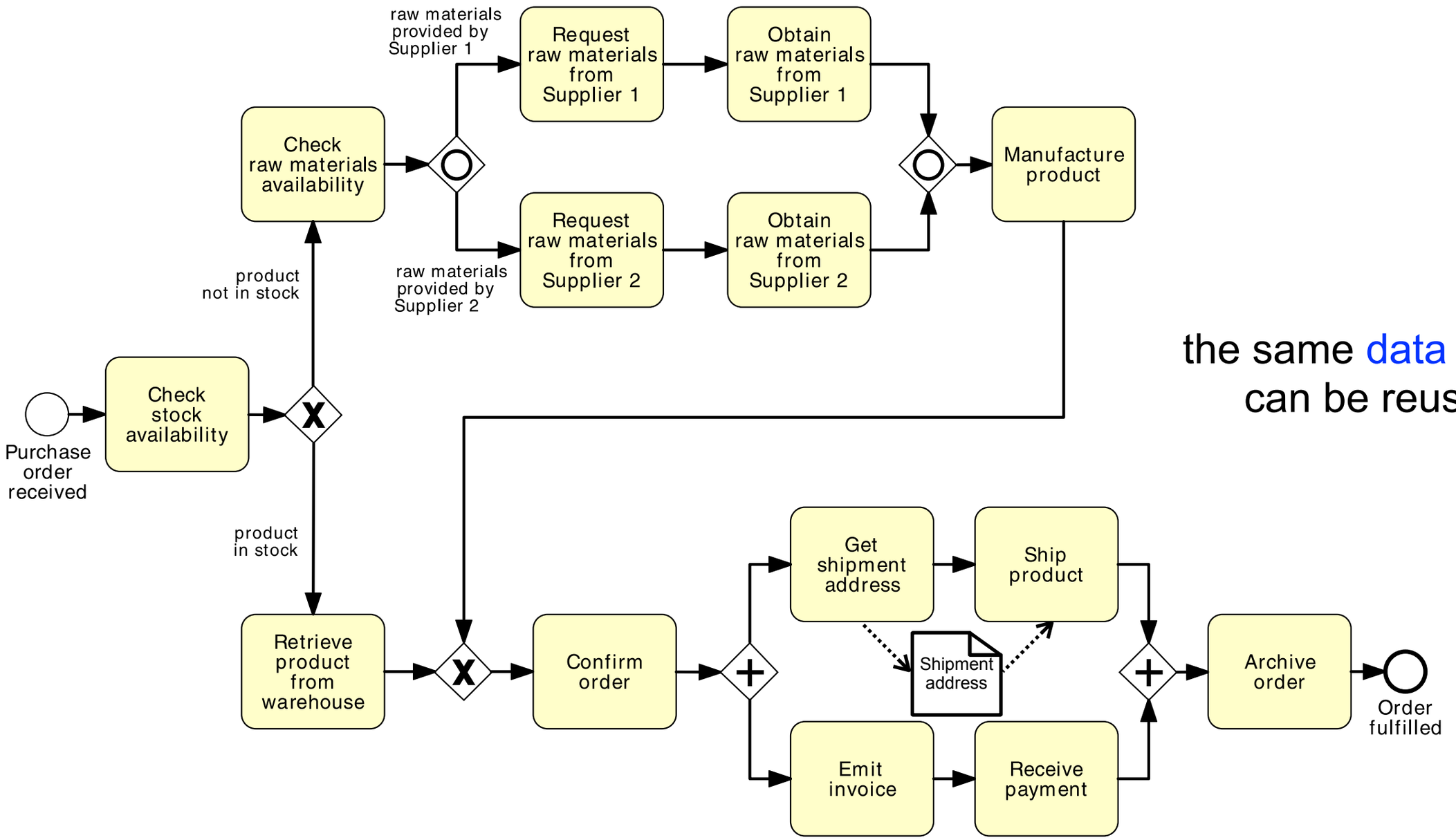
Example: data objects



Example: data objects

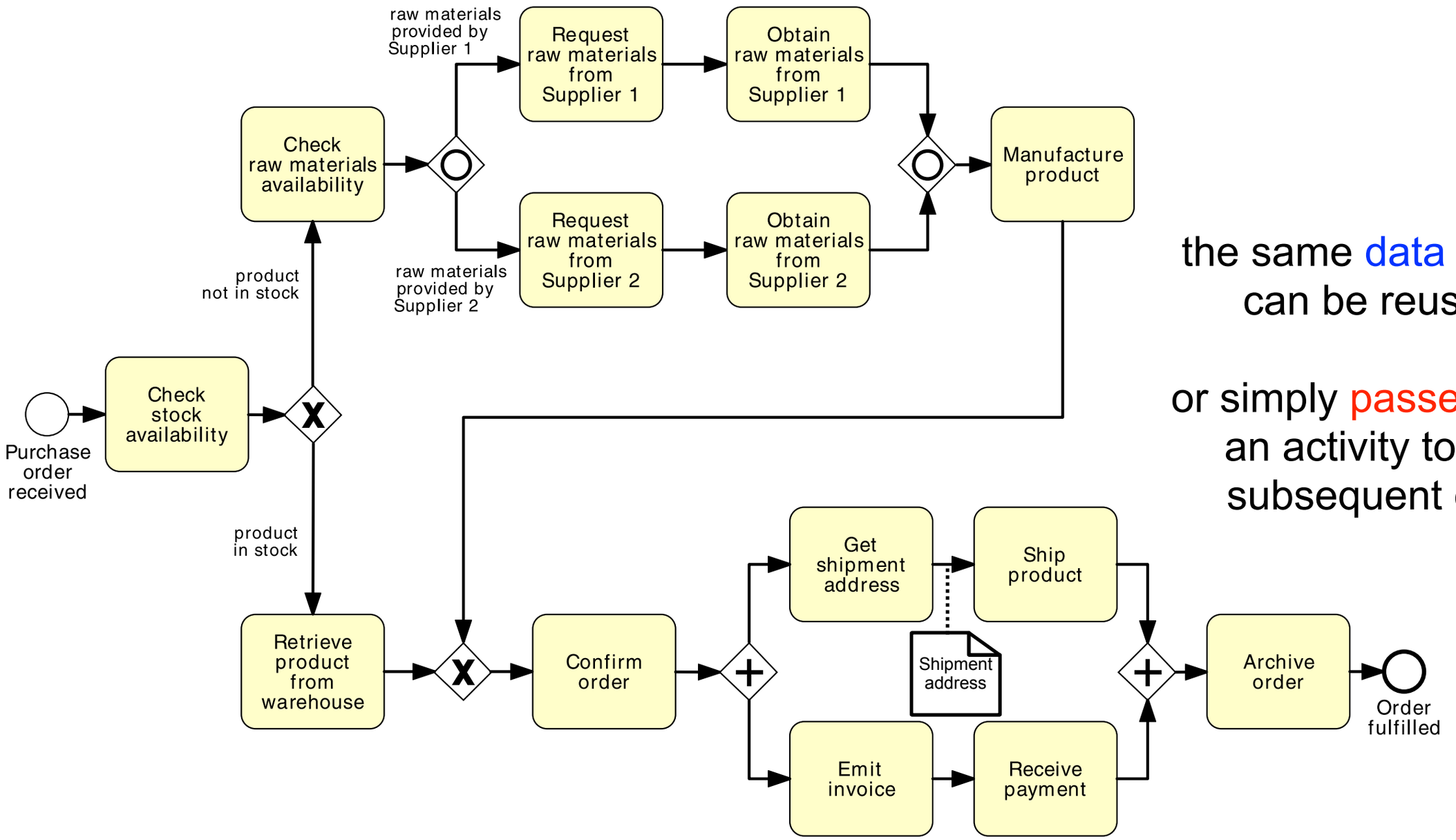


Example: data objects



the same data object can be reused

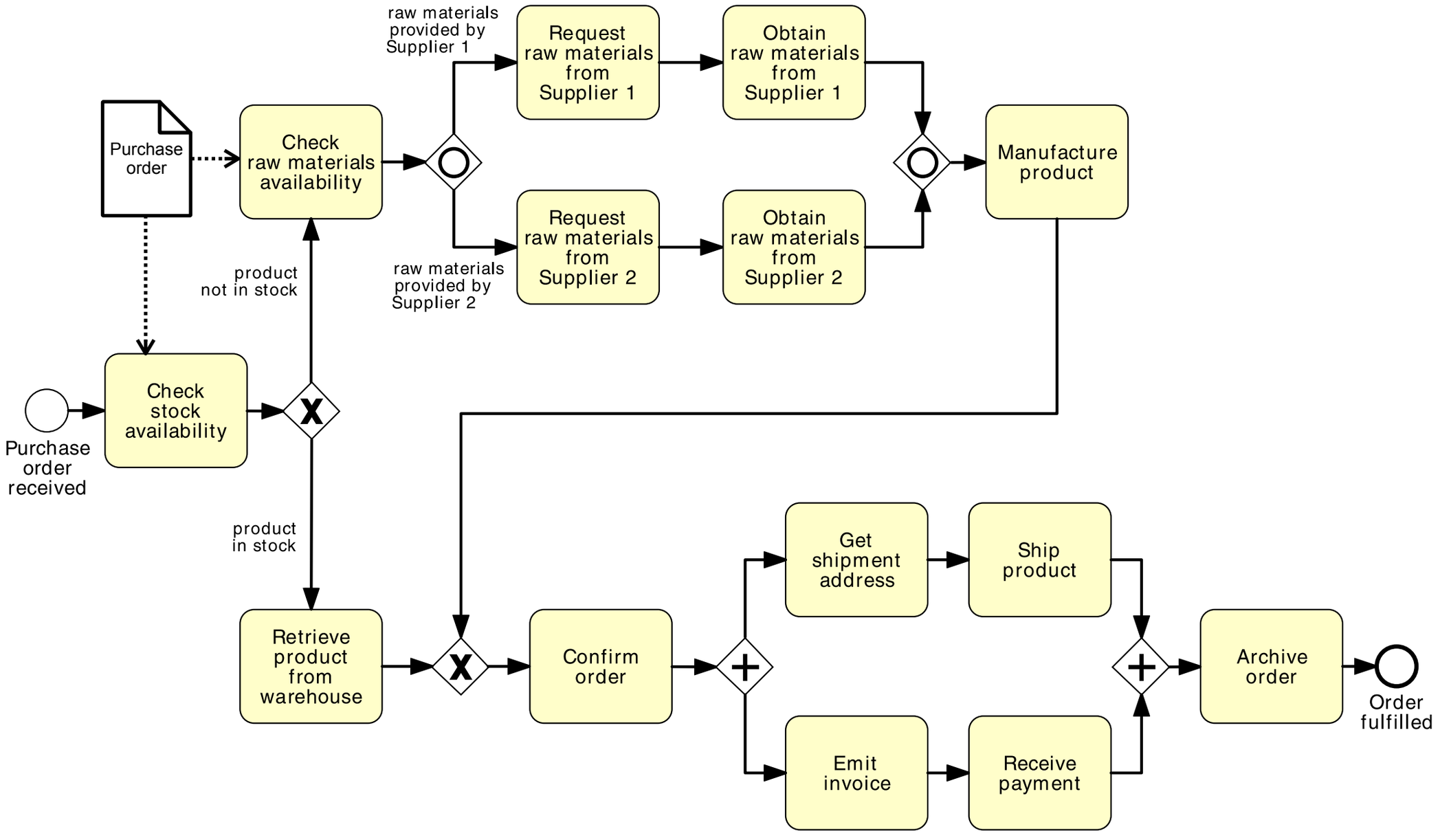
Example: data objects



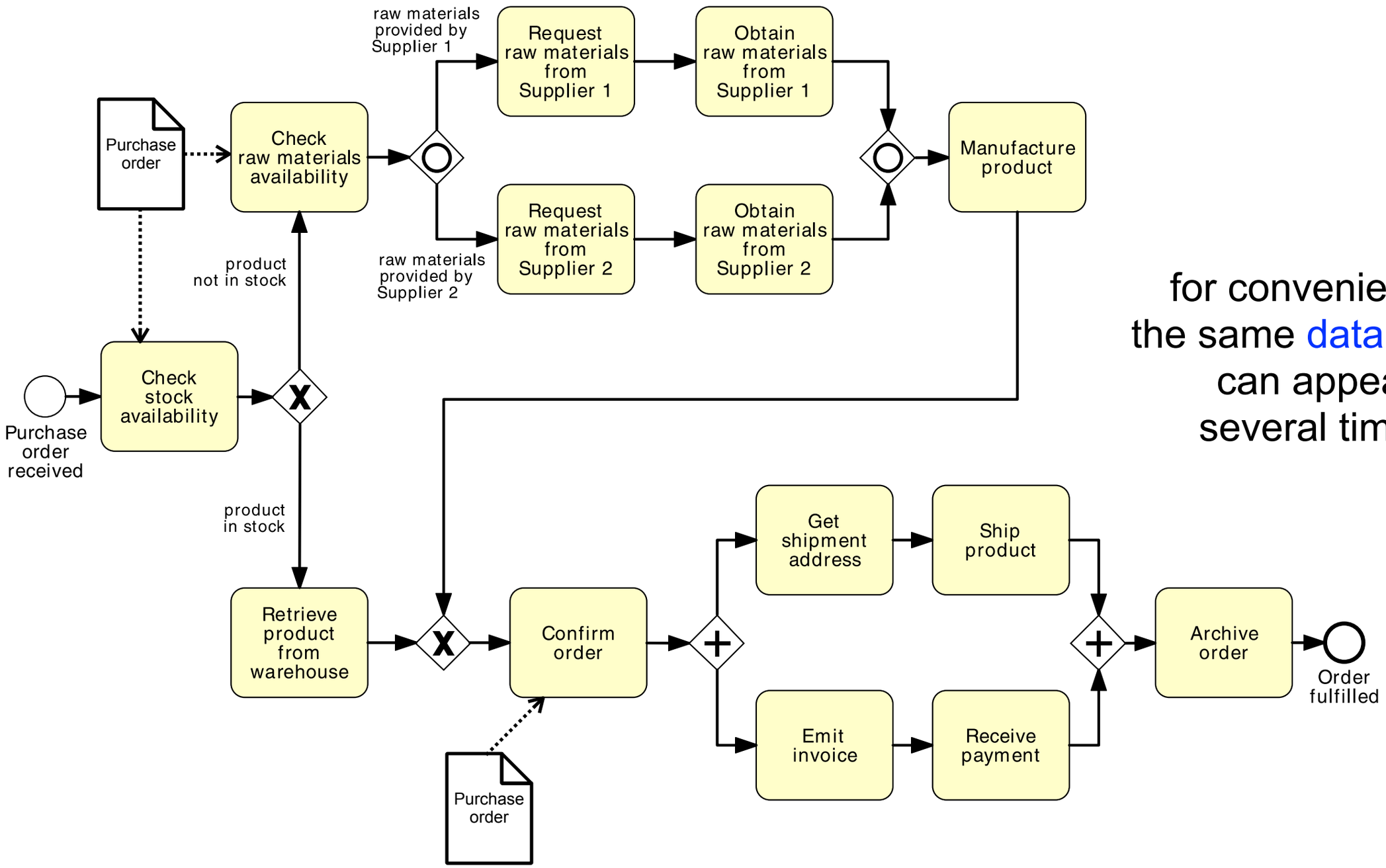
the same data object can be reused

or simply passed from an activity to the subsequent one

Example: data objects

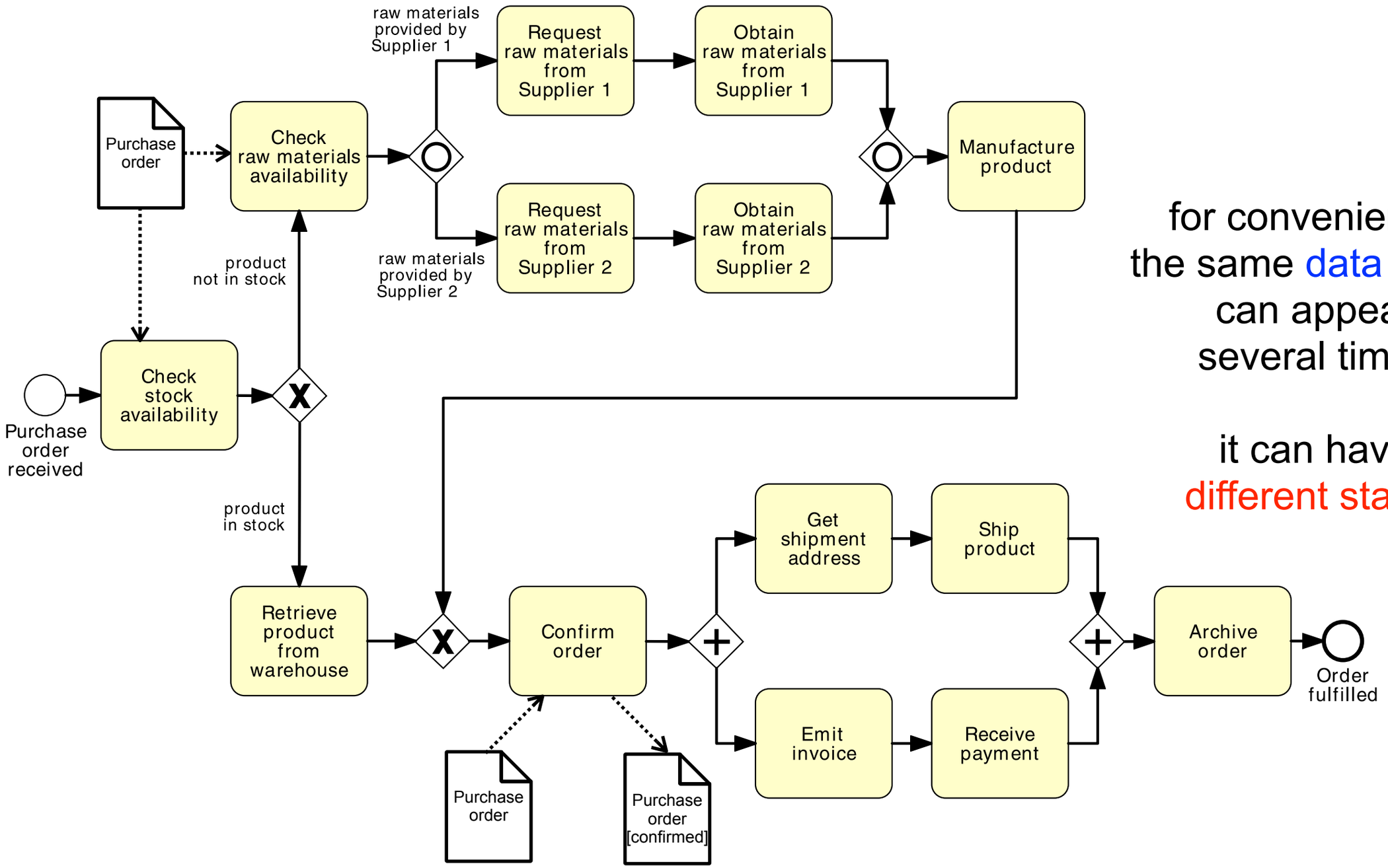


Example: data objects



for convenience, the same data object can appear several times

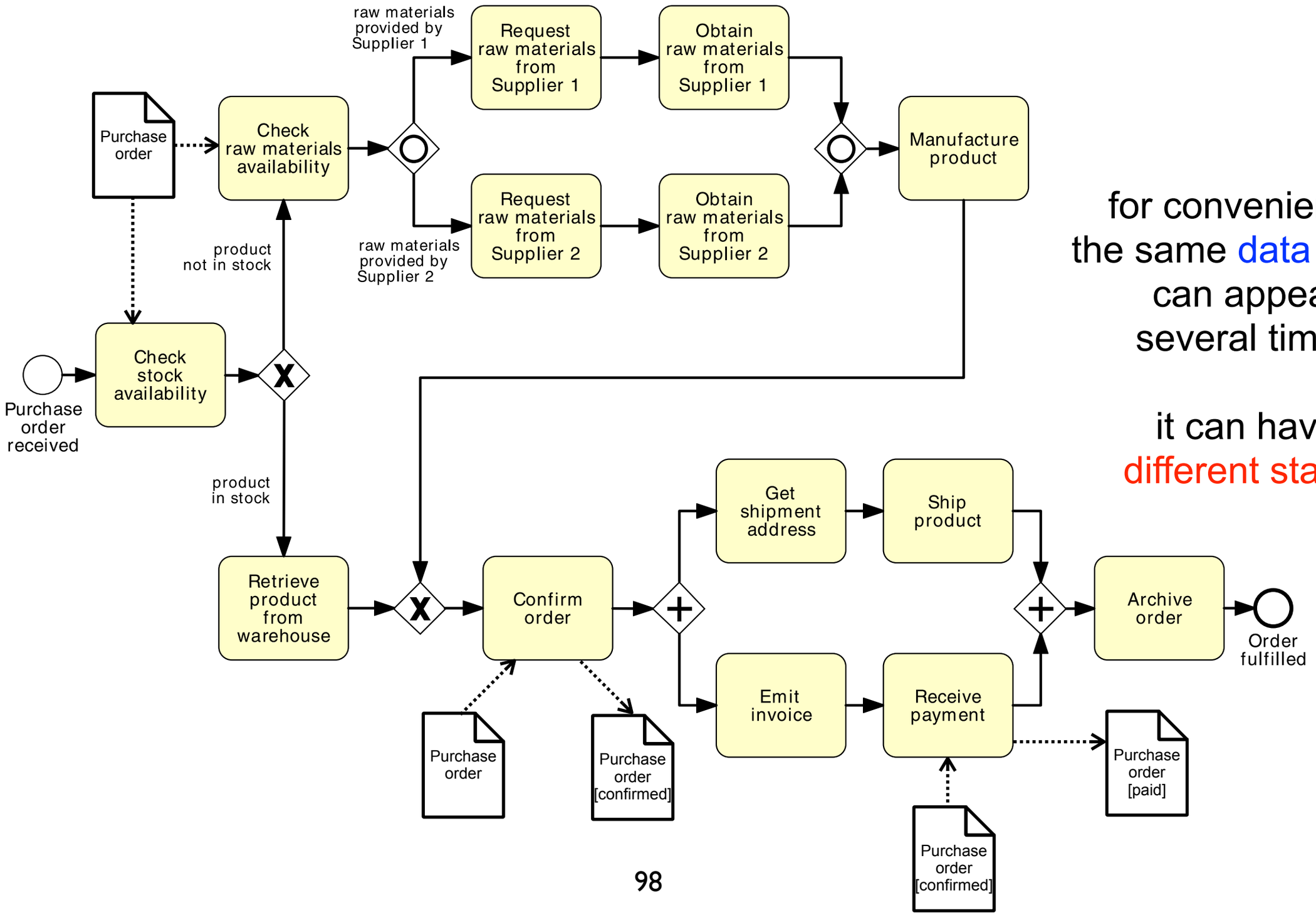
Example: data objects



for convenience,
the same data object
can appear
several times

it can have
different states

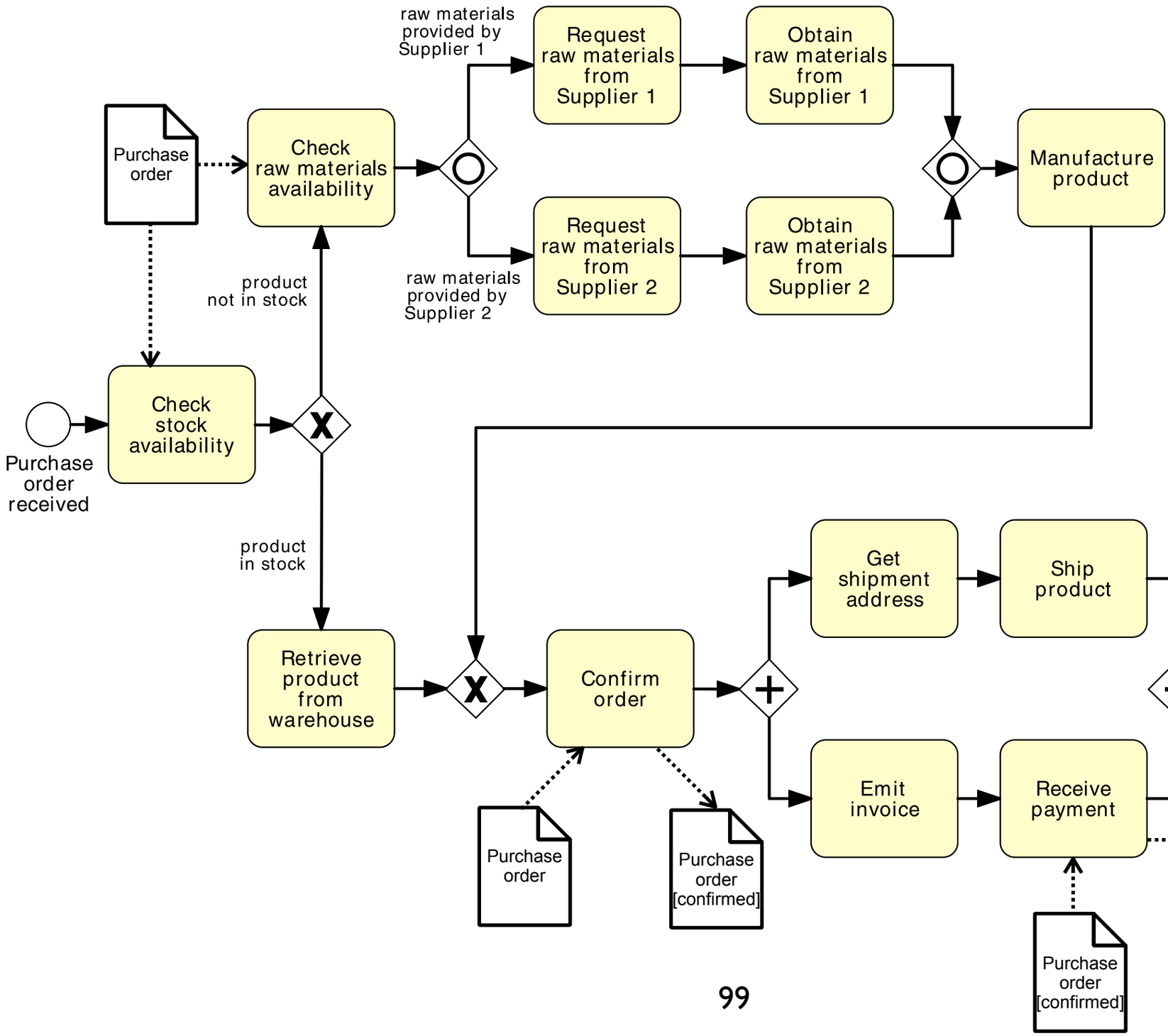
Example: data objects



for convenience,
the same data object
can appear
several times

it can have
different states

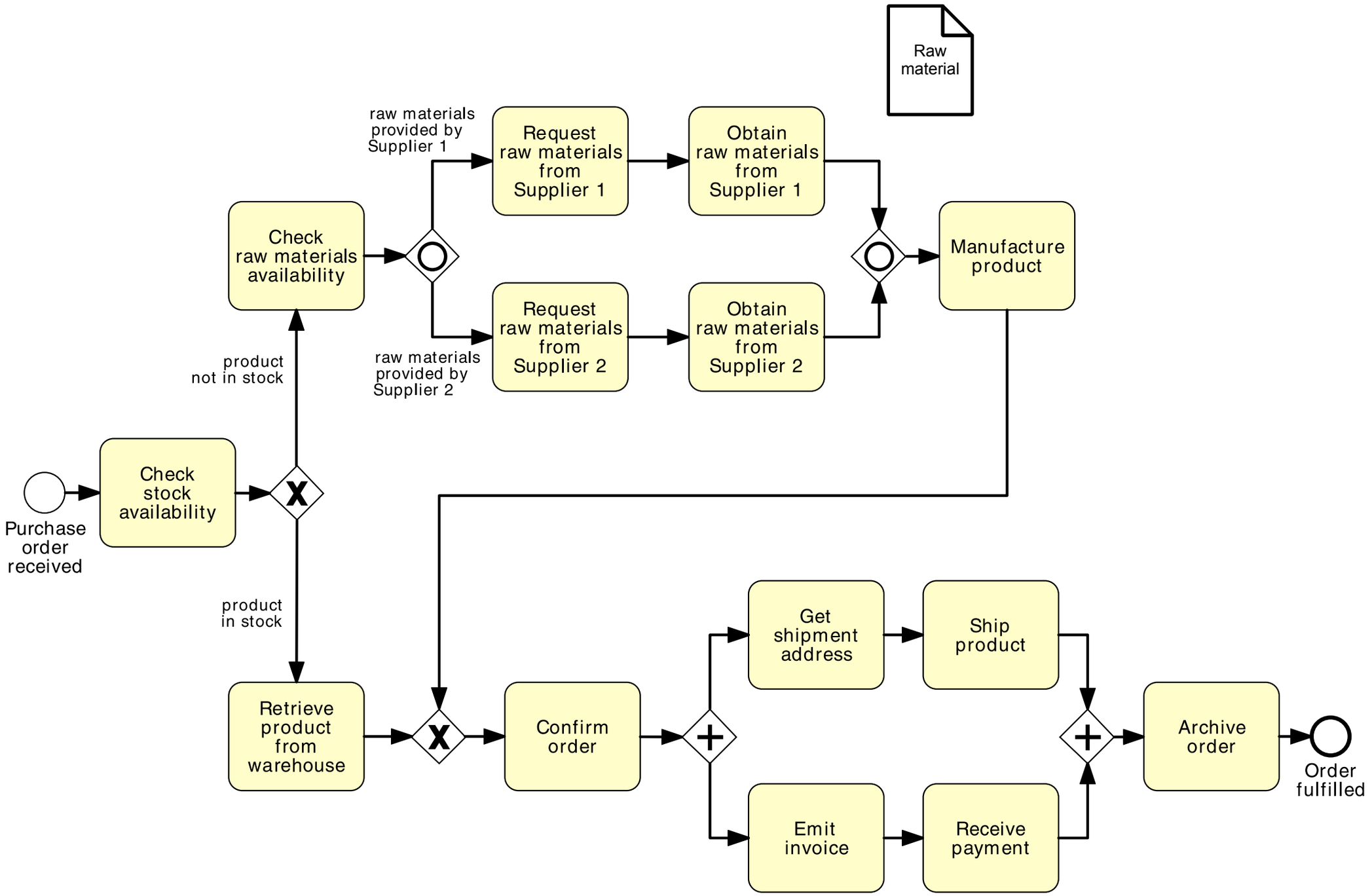
Example: data objects



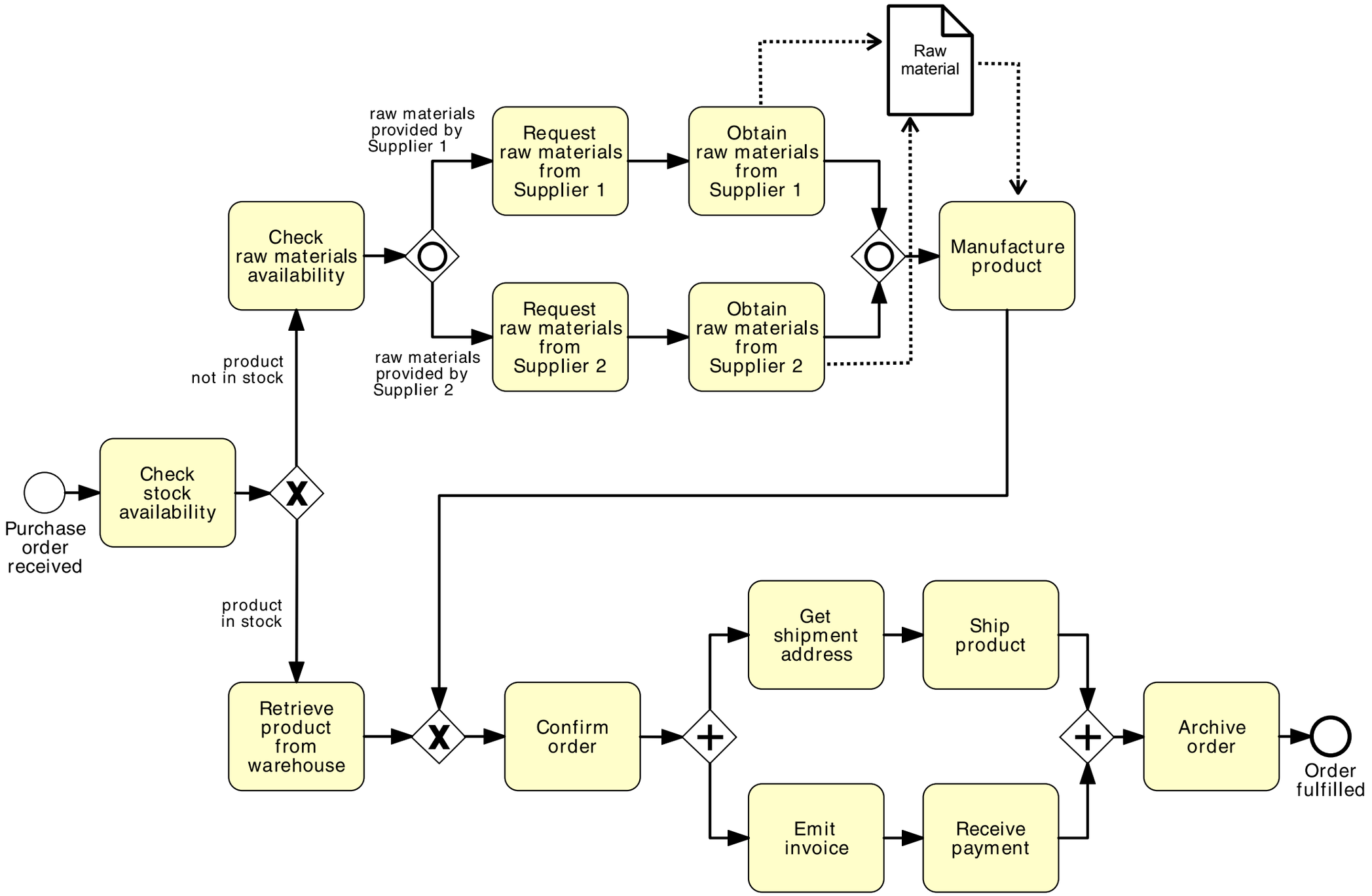
for convenience,
the same data object
can appear
several times

it can have
different states

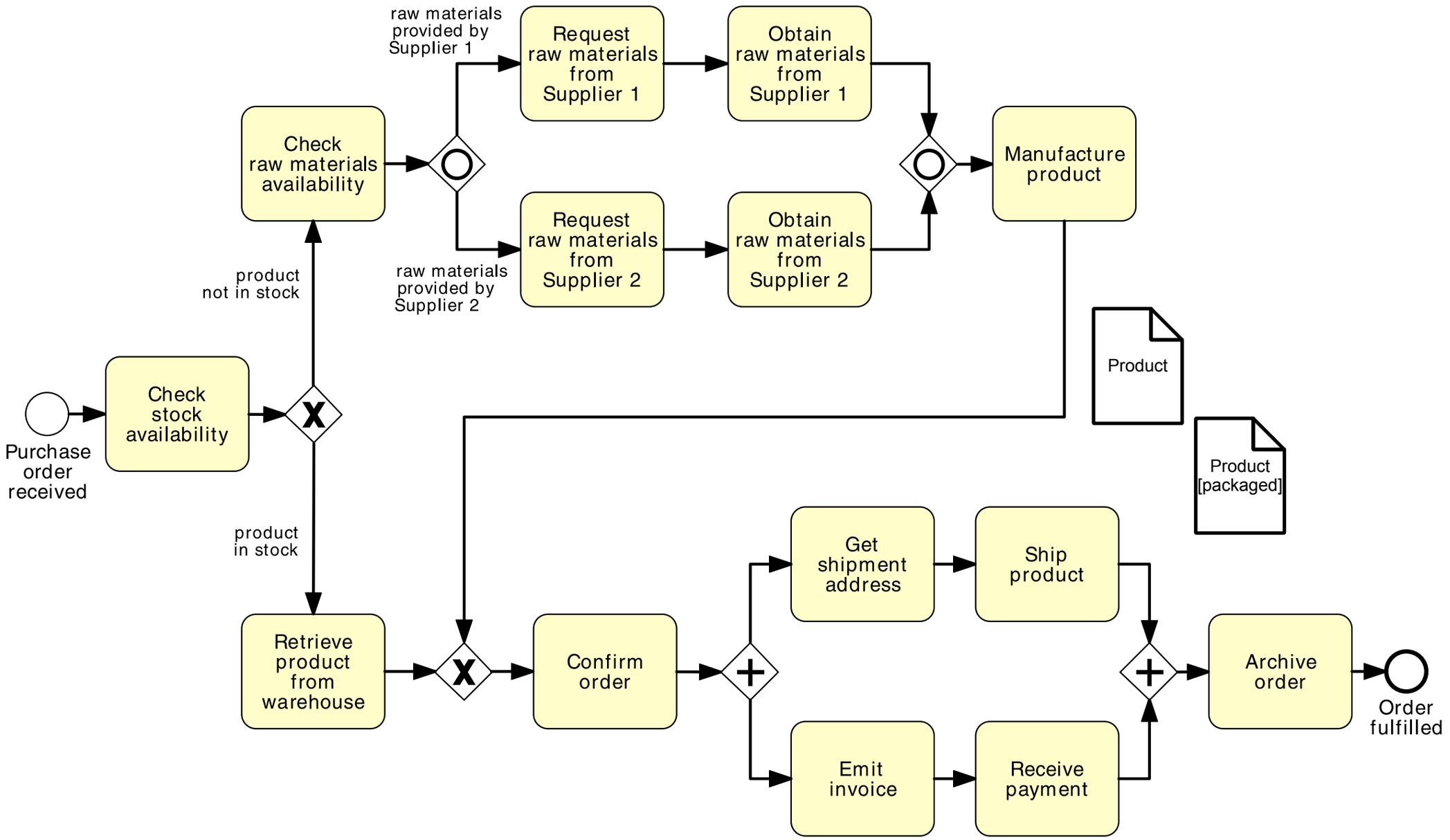
Question time: connections?



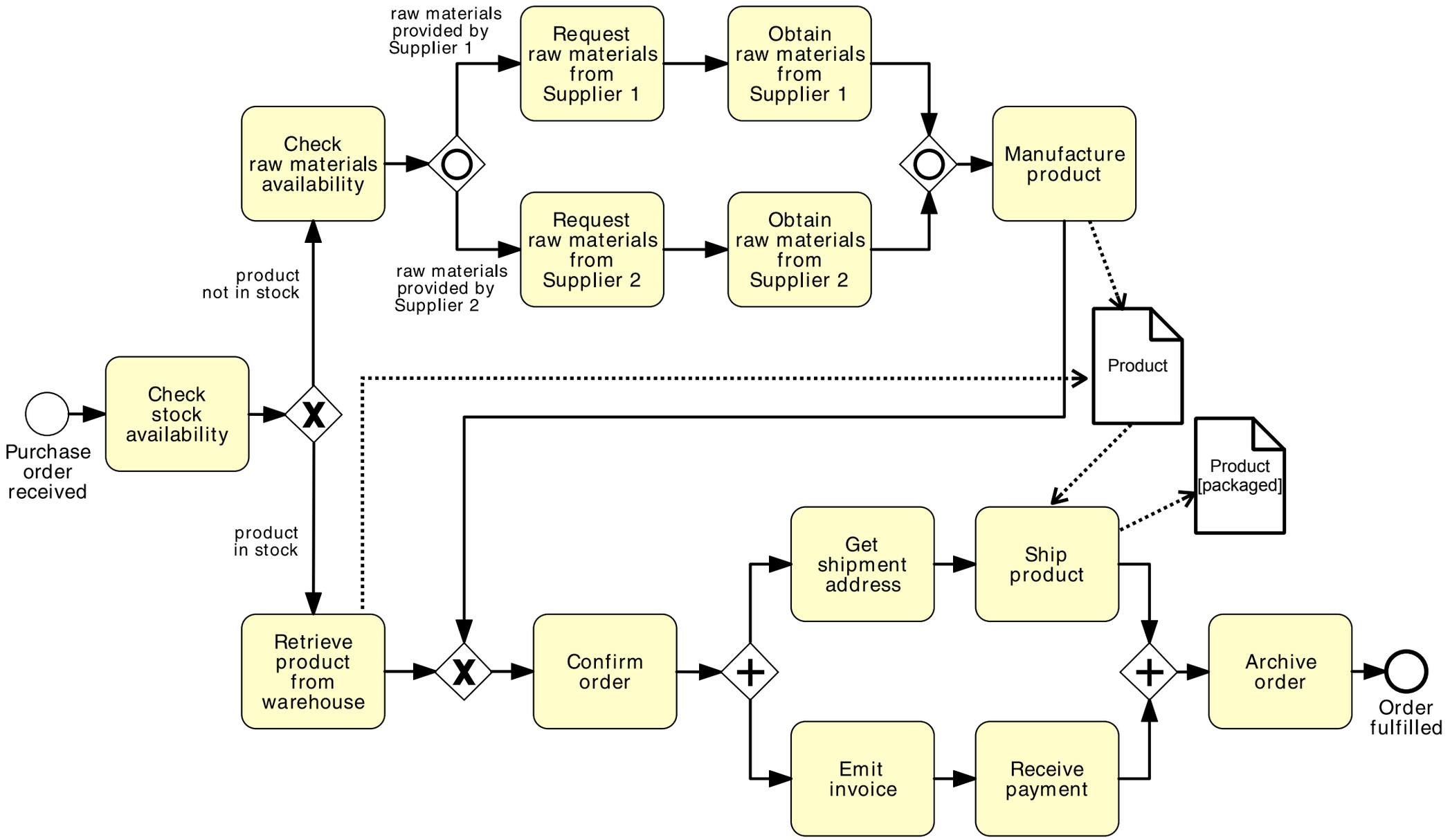
Question time: connections?



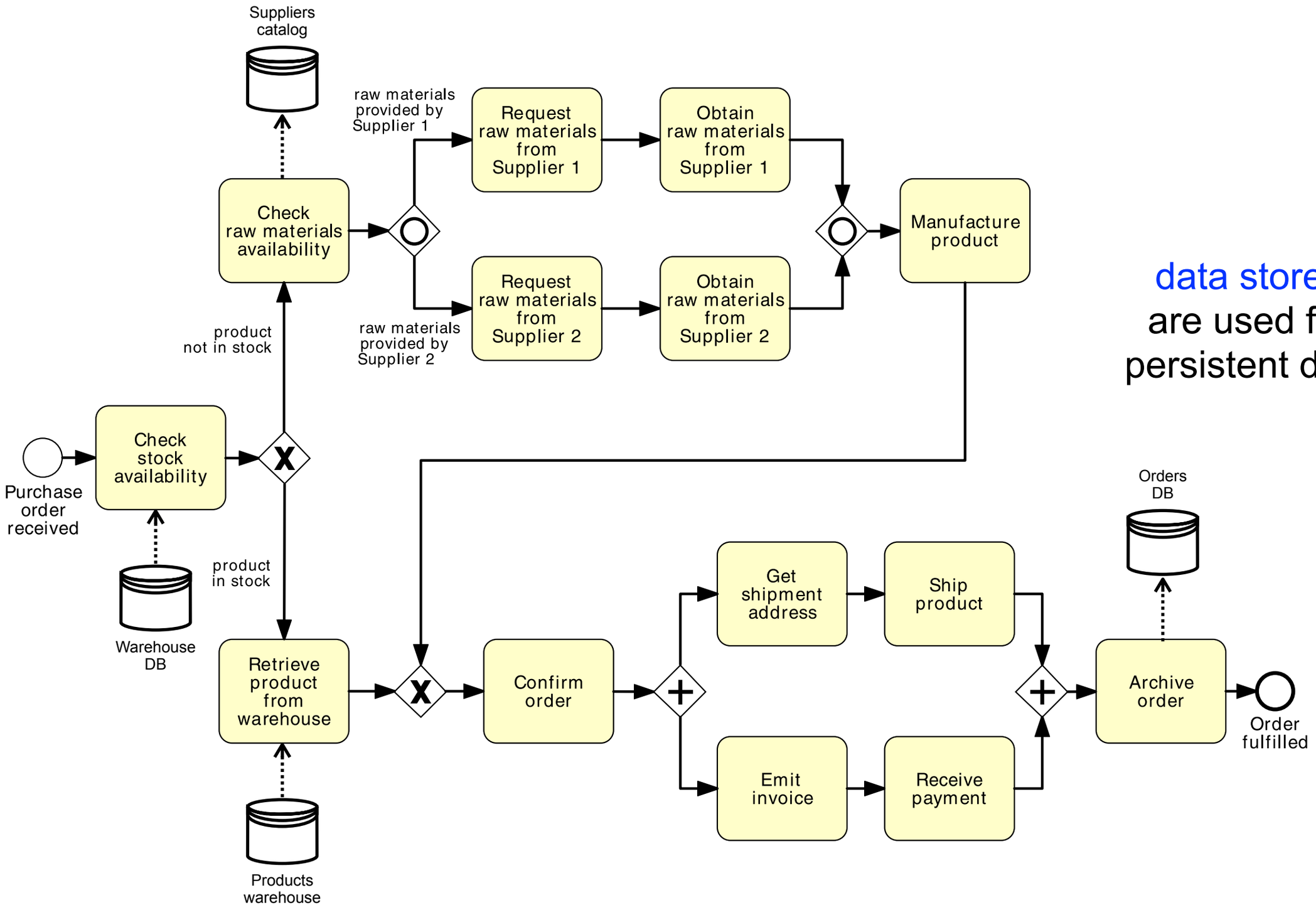
Question time: connections?



Question time: connections?

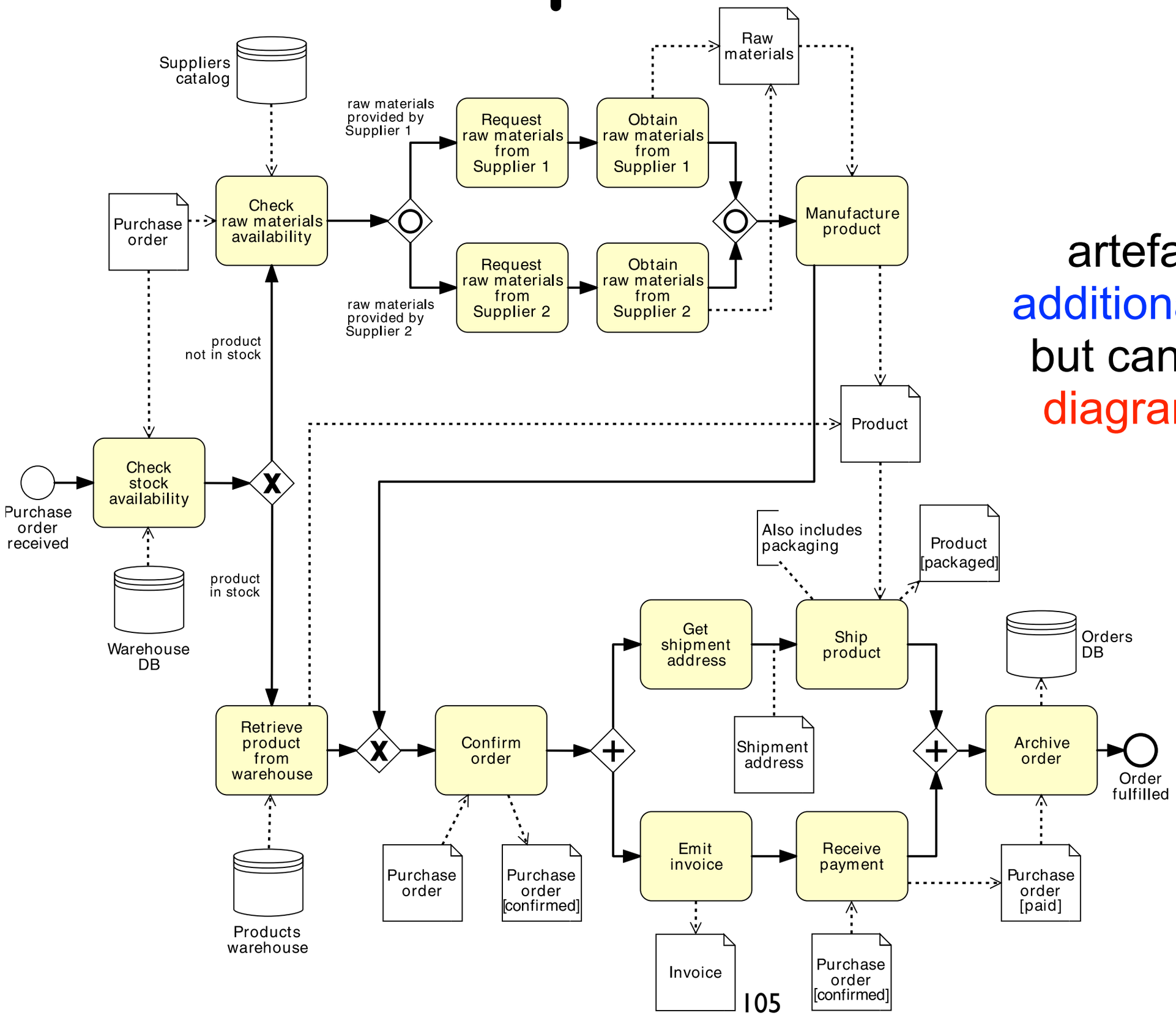


Example: data stores



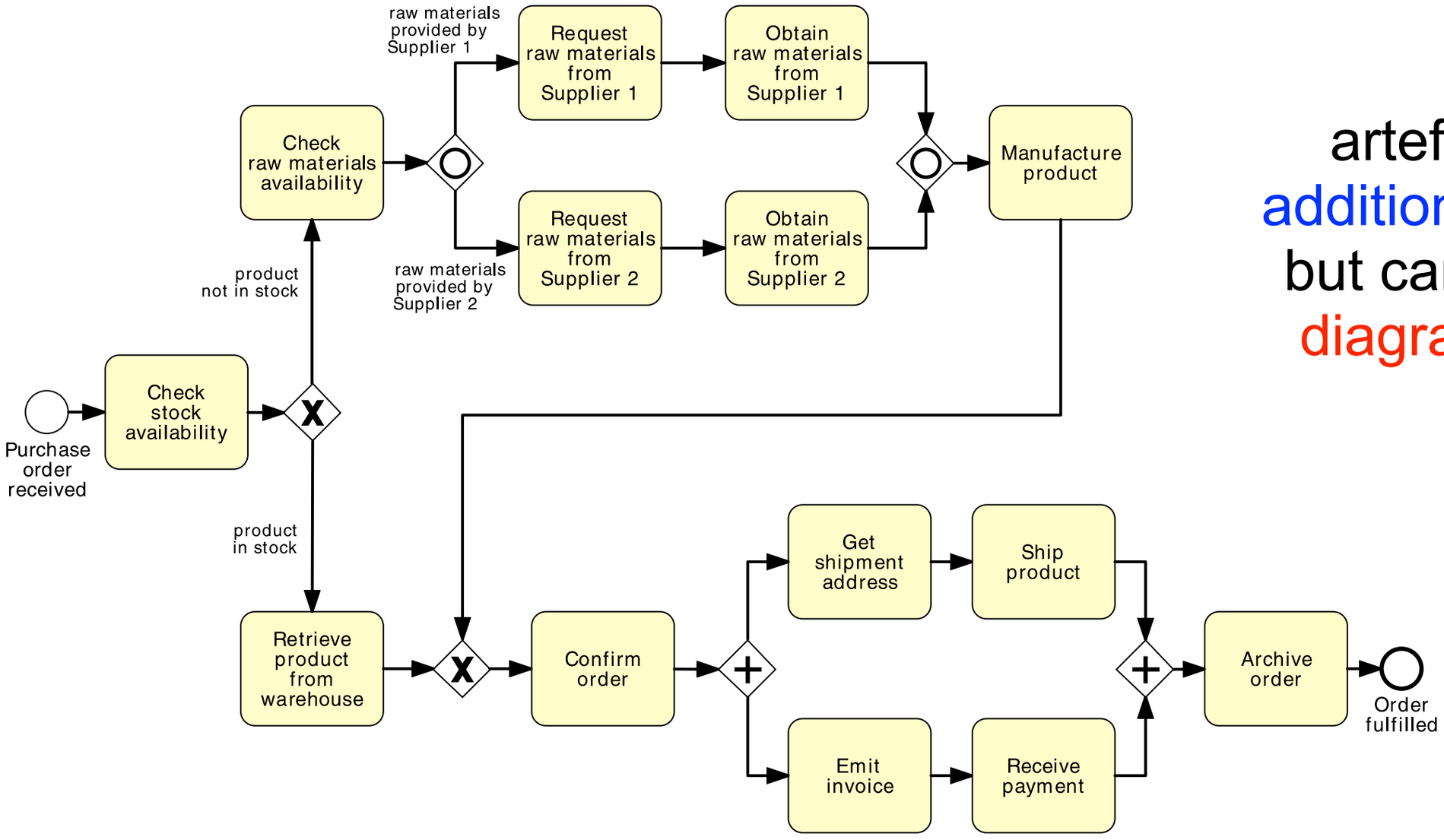
data stores
are used for
persistent data

Example: artefacts



artefacts provide additional information, but can compromise diagram readability

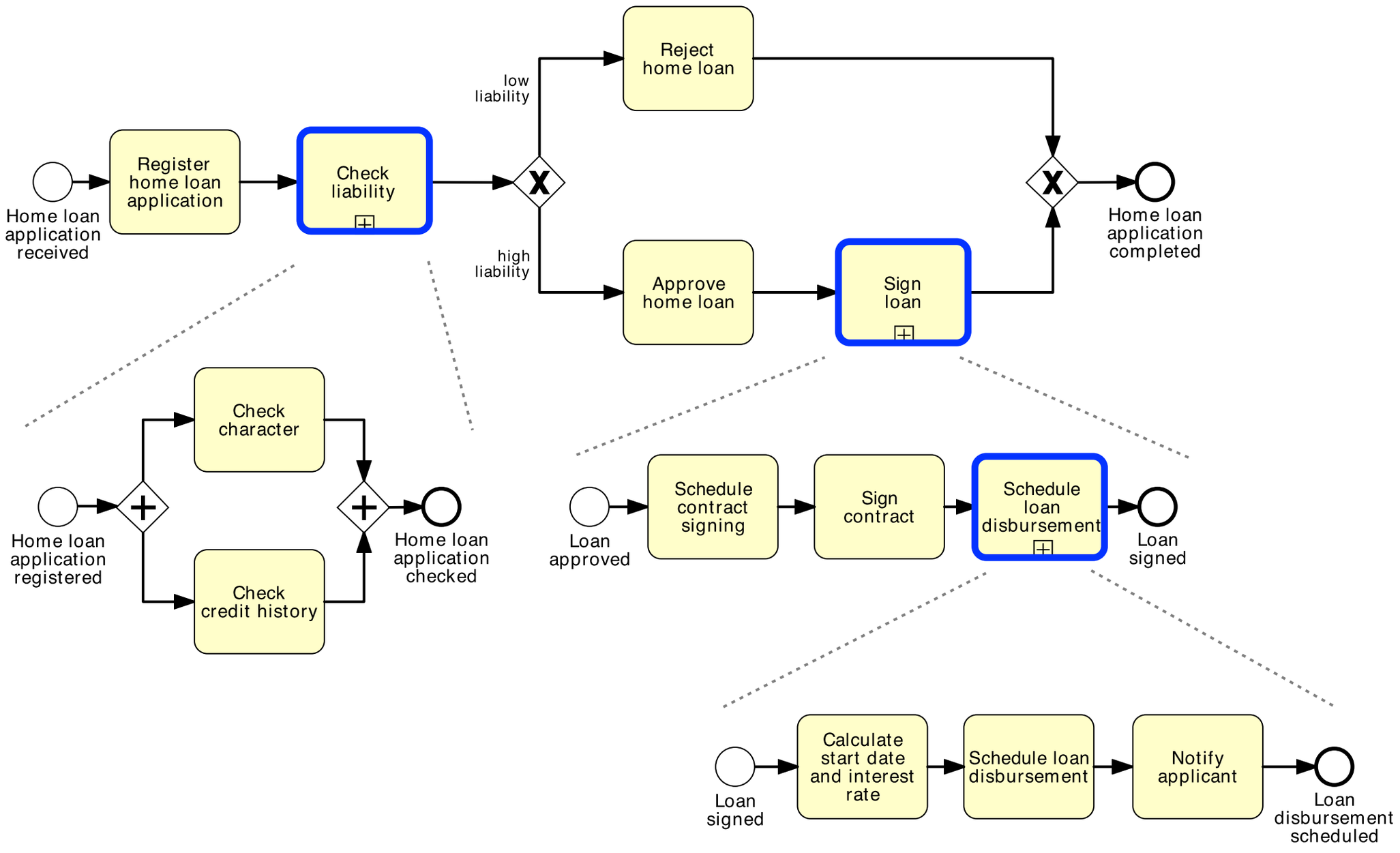
Example: artefacts



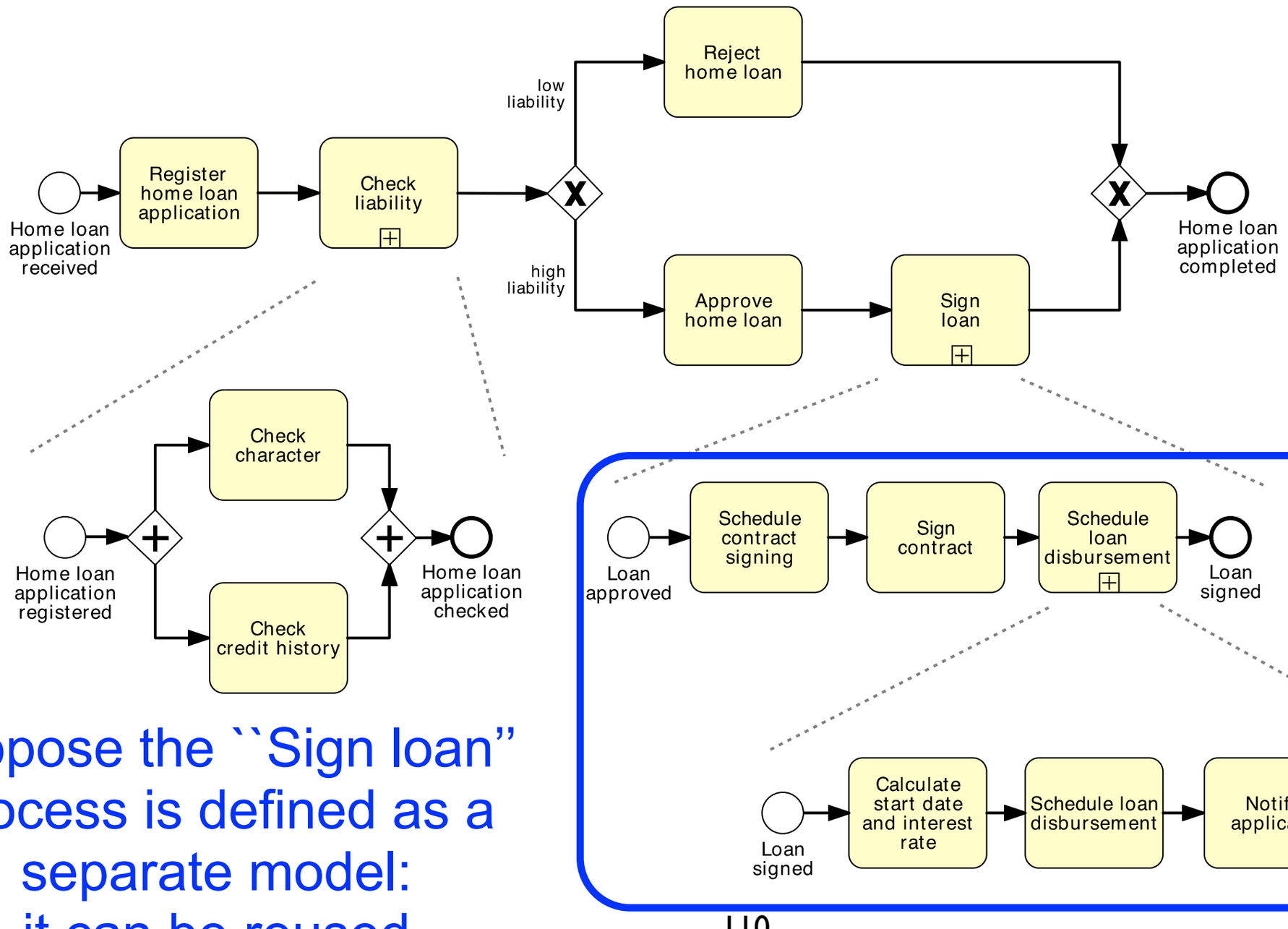
artefacts provide additional information, but can compromise diagram readability

Call activities

Nesting sub-processes: home loans



Global sub-processes: home / student loans

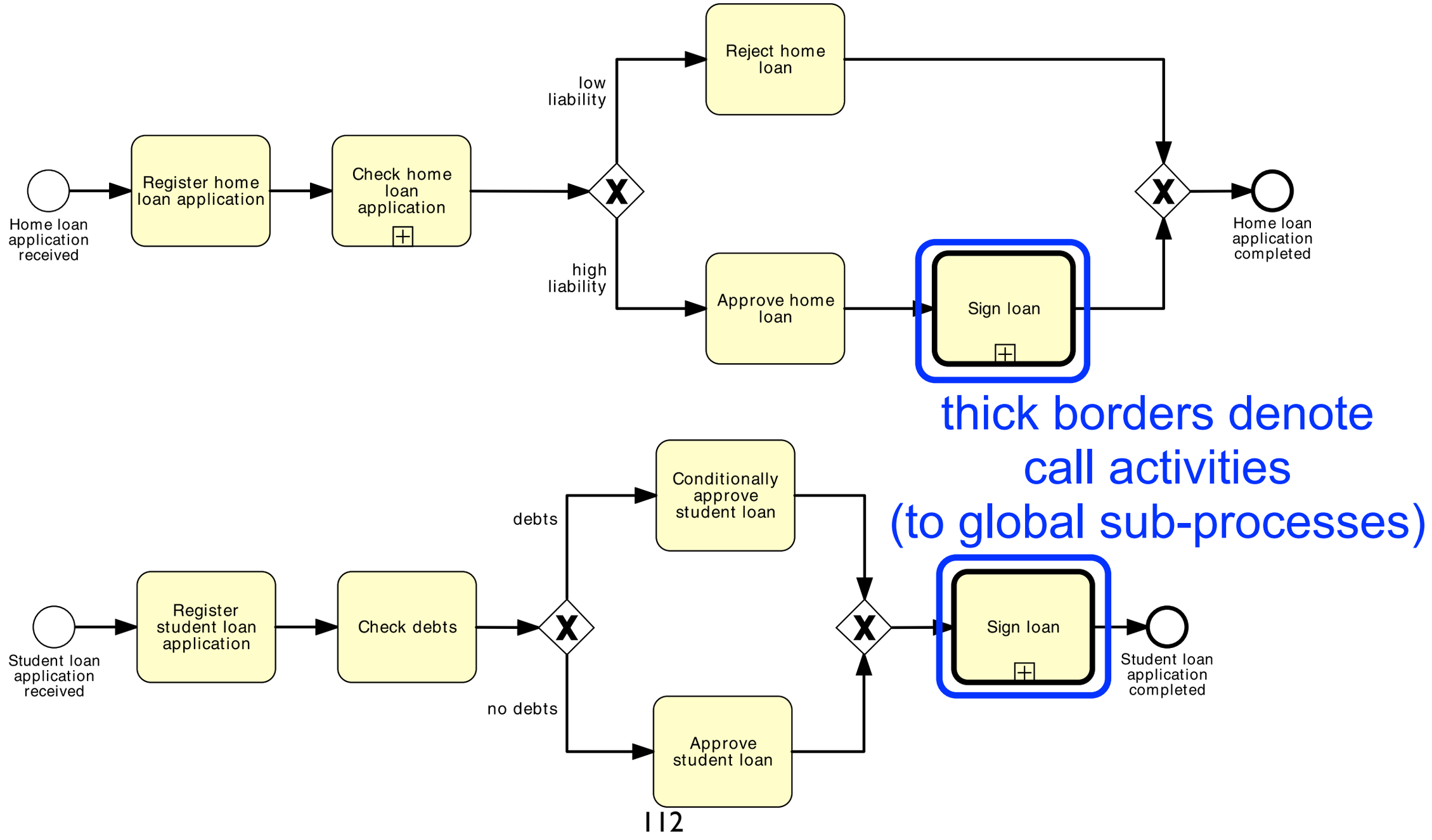


suppose the ``Sign loan'' process is defined as a separate model: it can be reused

Call activities

A **Call Activity** is a wrapper for a globally defined Sub-Process or Task that is reused in the current process.

Call activities: home / student loans



Global processes: advantages

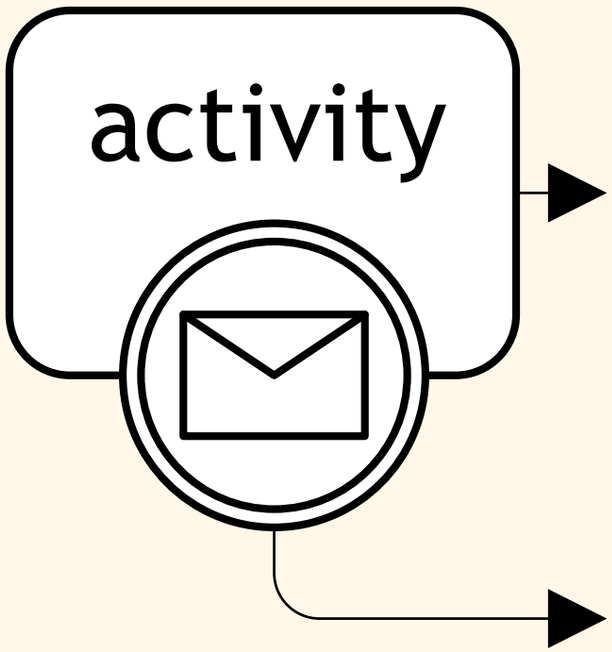
Readability: processes tend to be smaller

Reusability: define once, use many time

Sharing: any change made to a global process
is automatically propagated to all models that invoke it

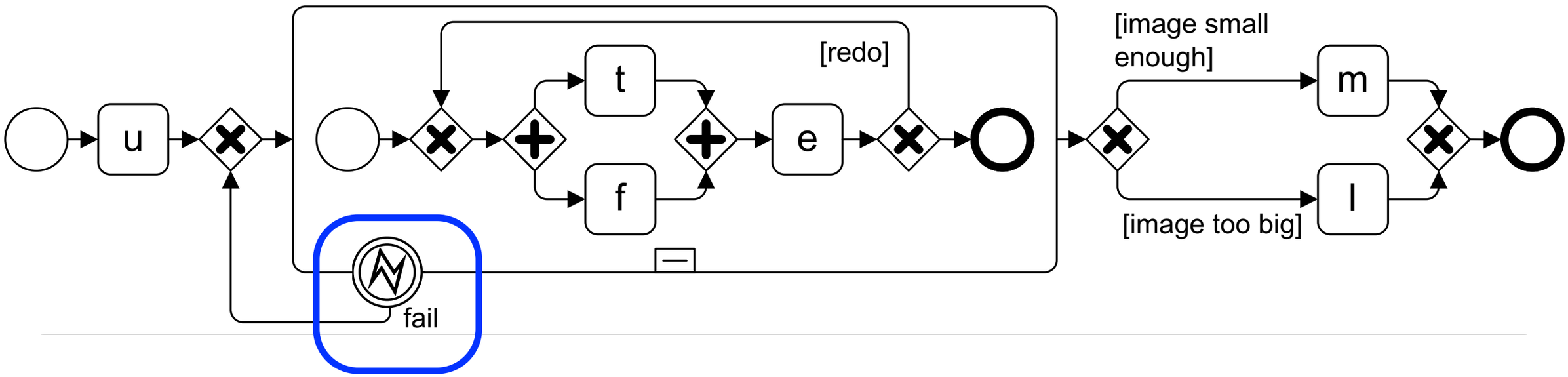
Throwing and catching

Attached events



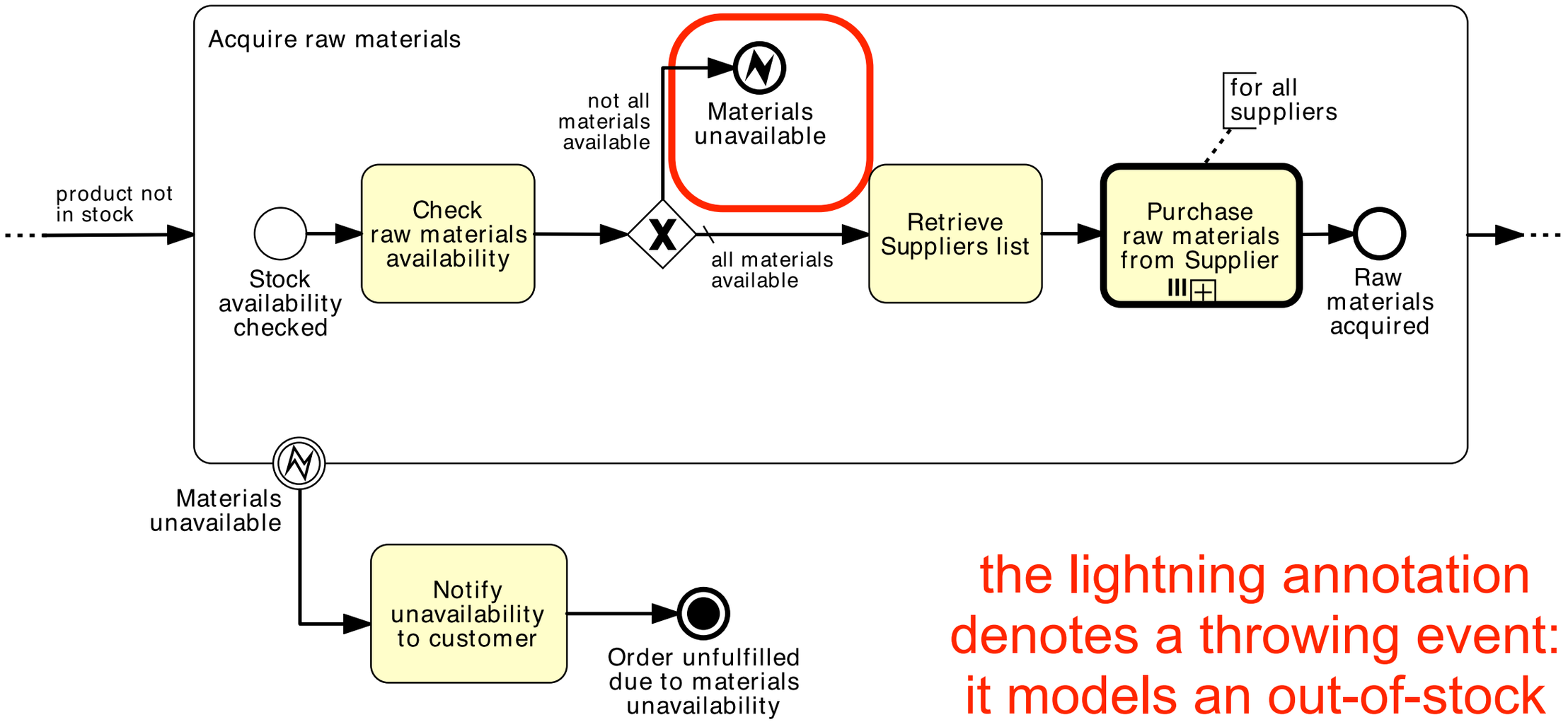
Attached Intermediate Event: The activity is aborted once an event is caught.

Recovery from faults: image manipulation



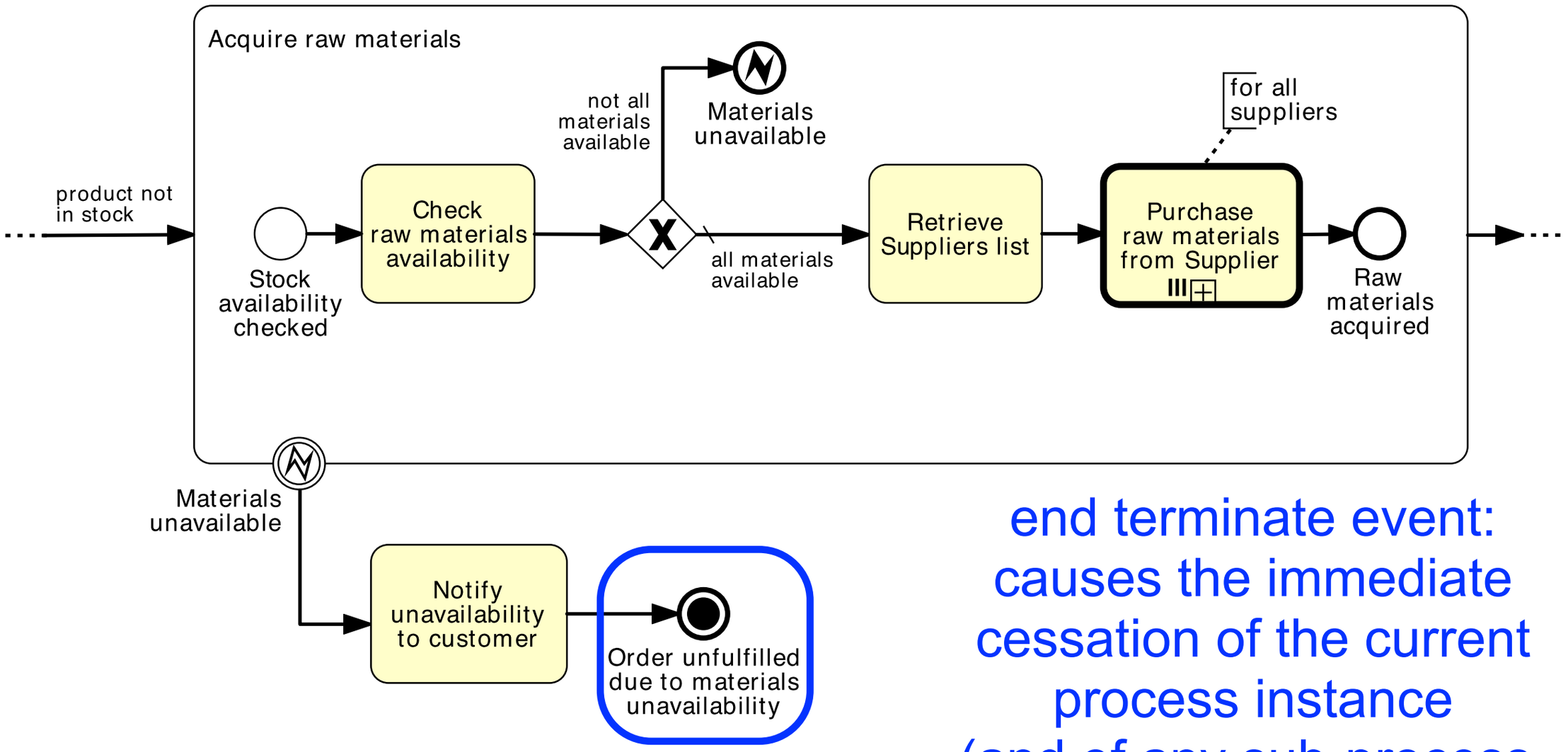
the lightning annotation
denotes an error-catching event

Throwing and catching: order fulfillment



the lightning annotation denotes a throwing event: it models an out-of-stock exception

Throwing and catching: order fulfillment

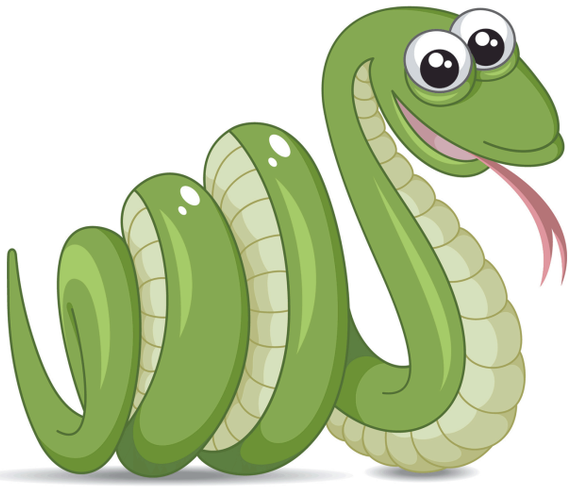


end terminate event:
causes the immediate cessation of the current process instance
(and of any sub-process, but not of the parent process if any)

Choreographies

Guess the animal

like a snake

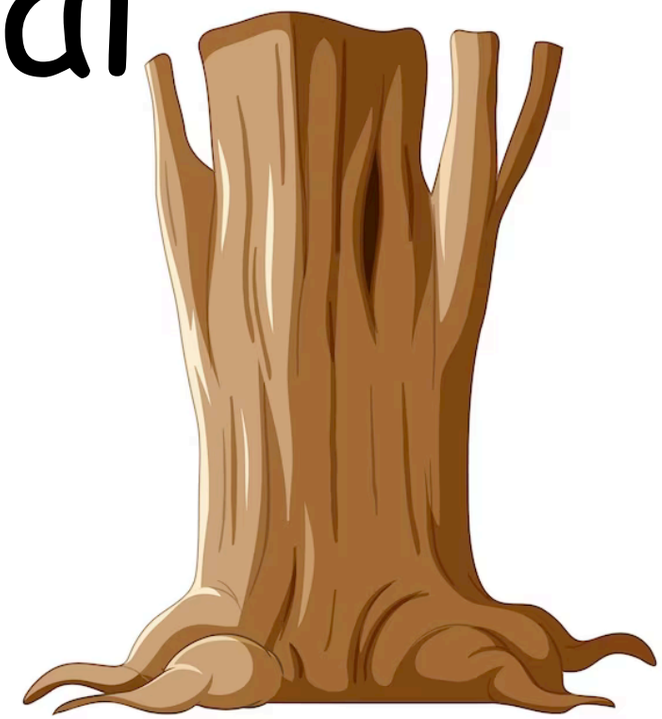


but not a snake

like a fan

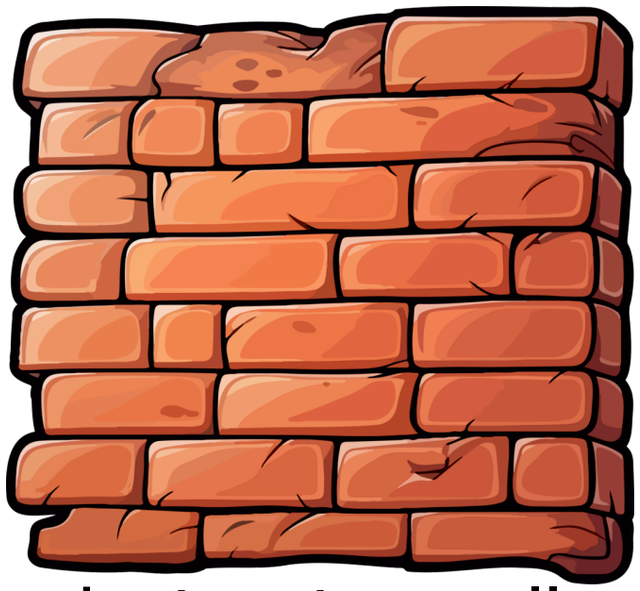
but not a fan

like a trunk



but not a trunk

like a wall



but not a wall

like a rope

but not a rope

like a spear

but not a spear

Choreography

A **choreography** defines the sequence of interaction between participants

A choreography does not exists in a pool and it is not executable

It describes how the participants are supposed to behave

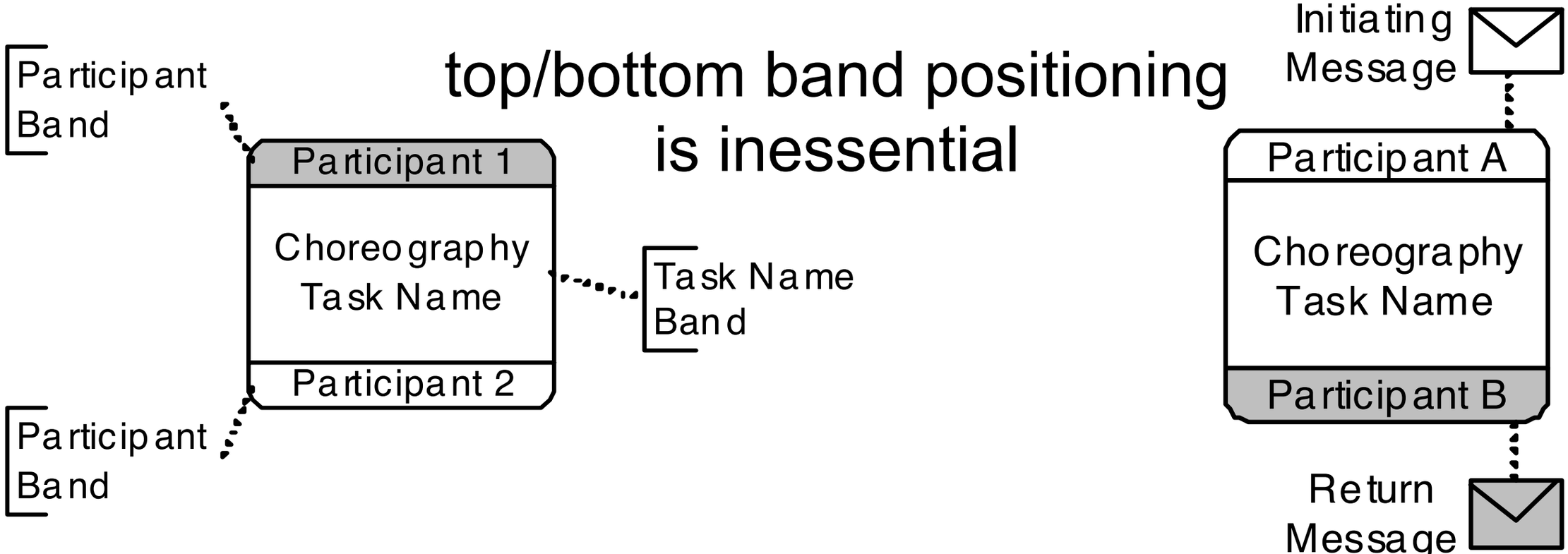
a choreography can also use message data objects

Choreography task

A **choreography task** is an activity in a choreography that consists of a set (one or more) communications

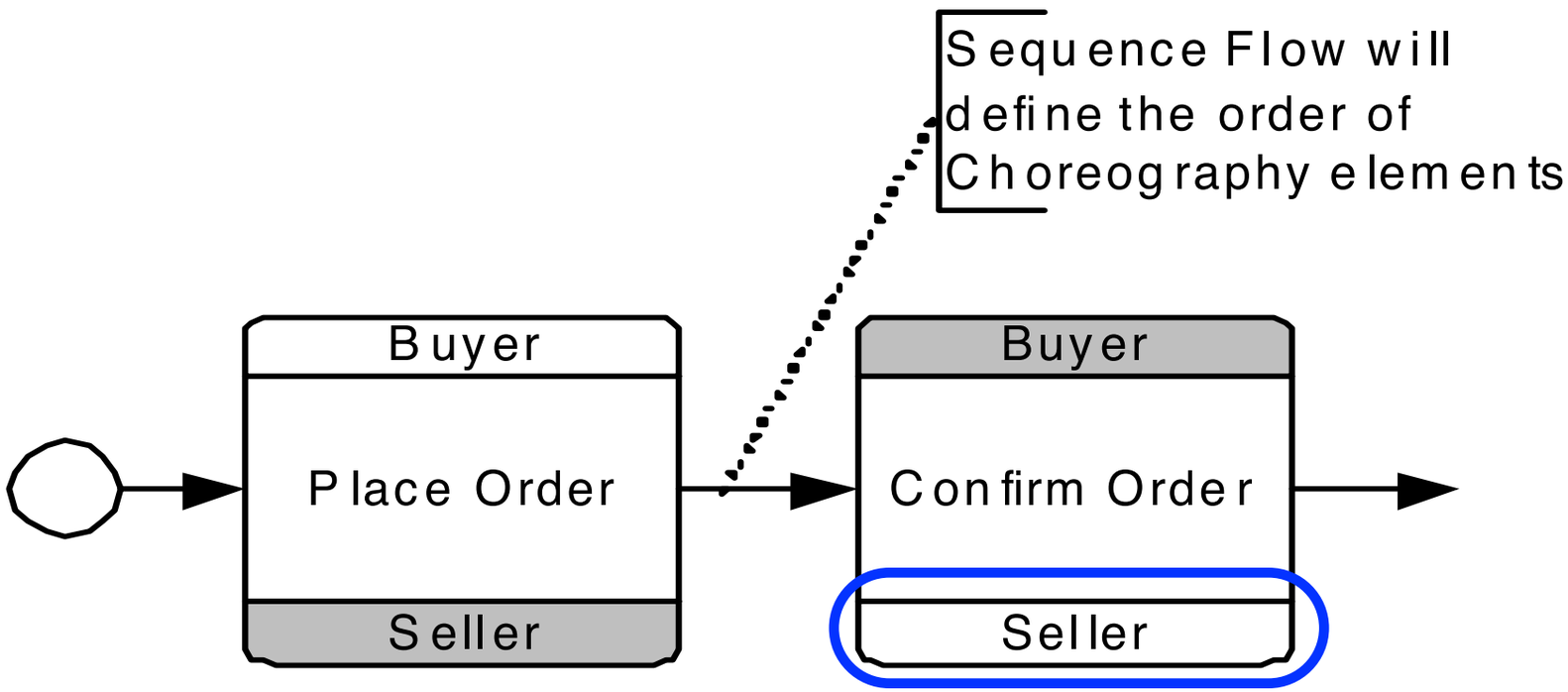
A choreography task involves two or more participants that are displayed in different bands

top/bottom band positioning is inessential



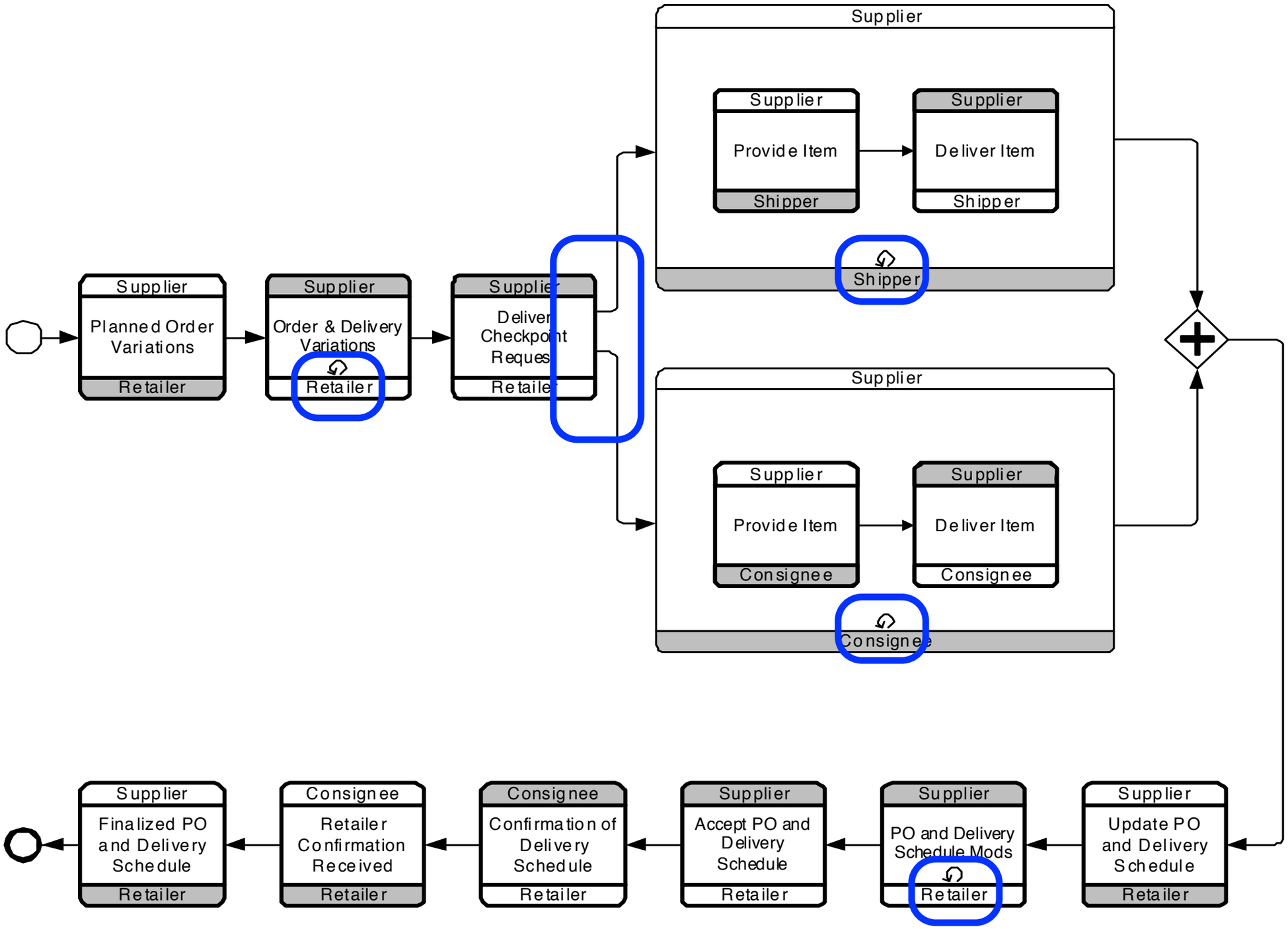
Choreography flow

Ordinary sequence flow and gateways
are used within choreographies
to show the sequence of tasks involved

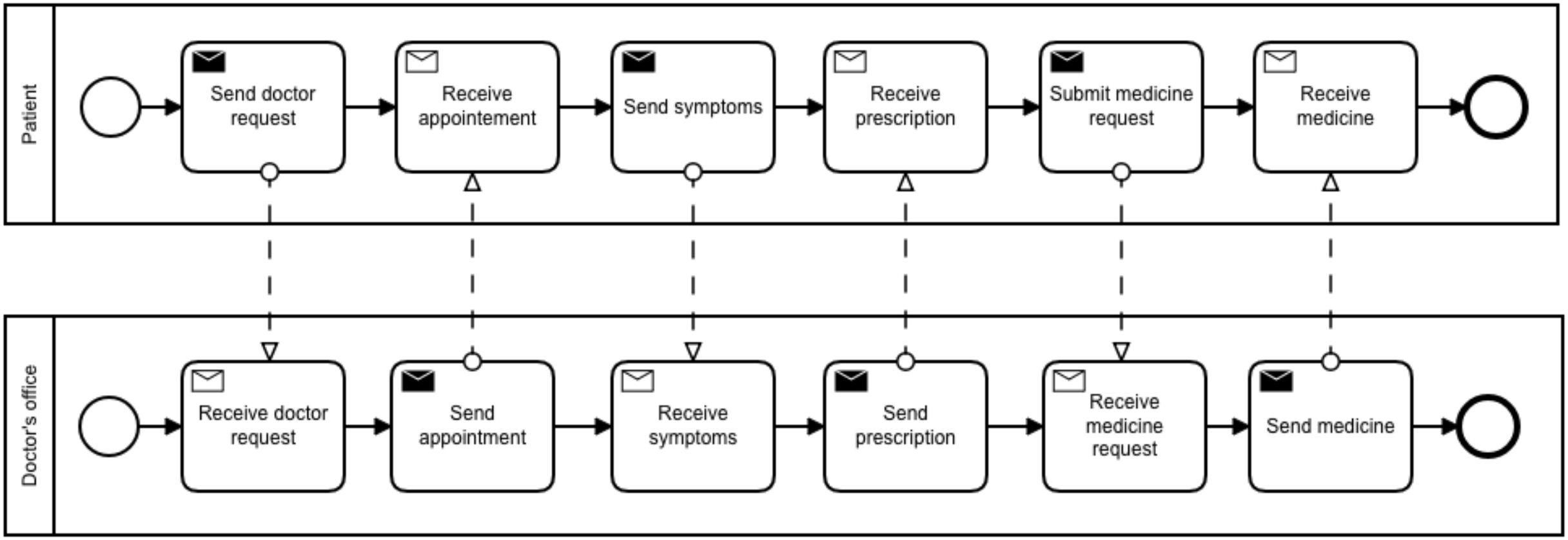


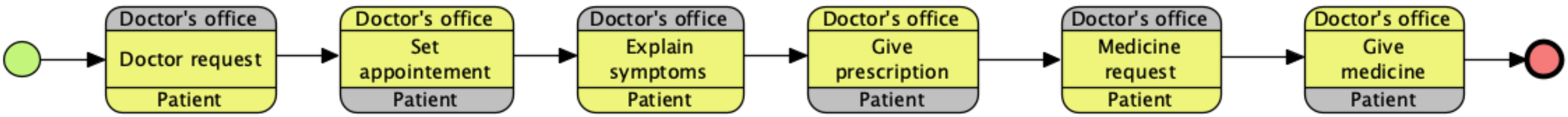
the initiator of the second interaction
must be involved in the previous one

A large choreography

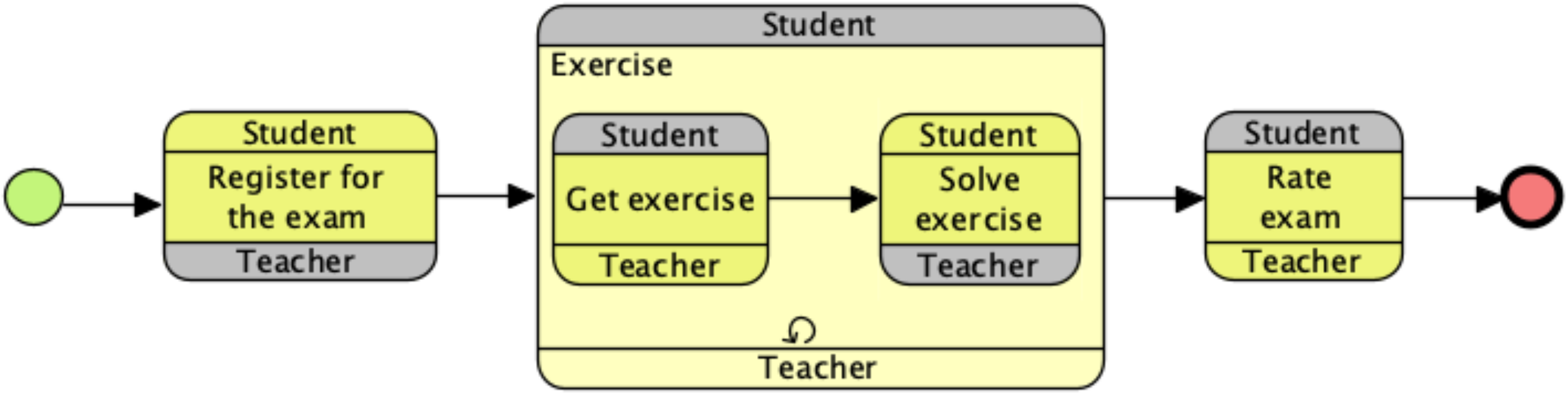


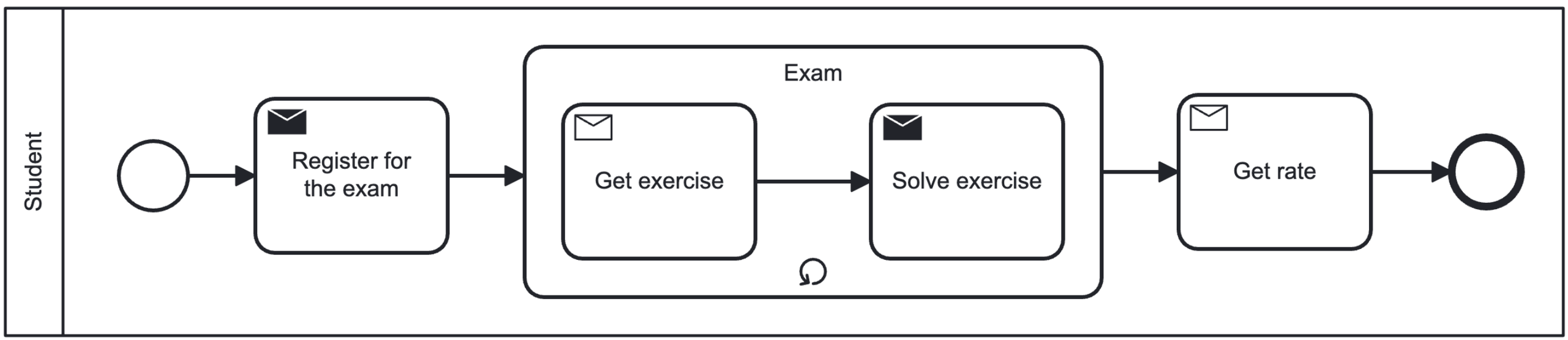
From collaborations to choreographies



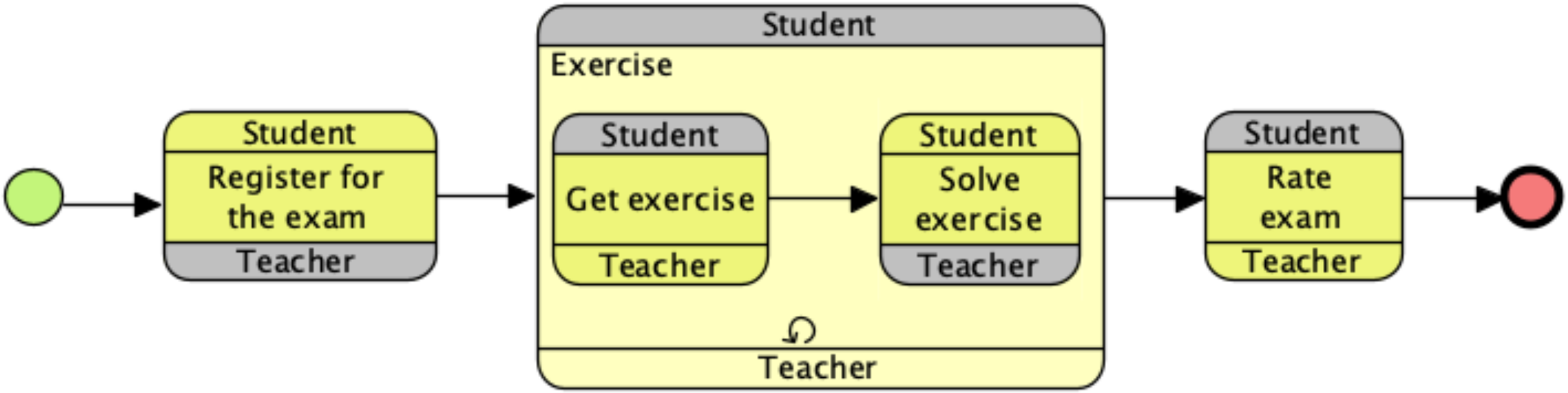


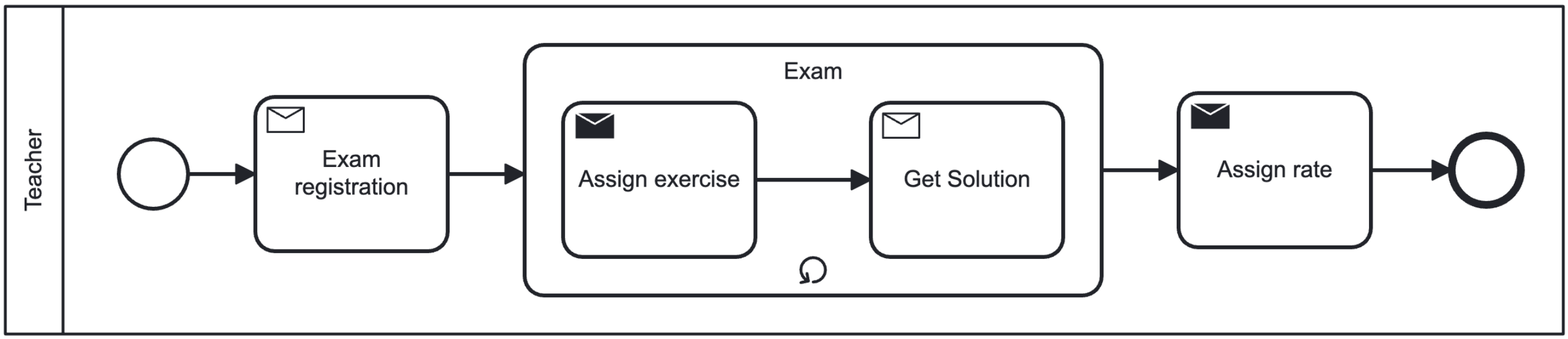
Projection (on Student)



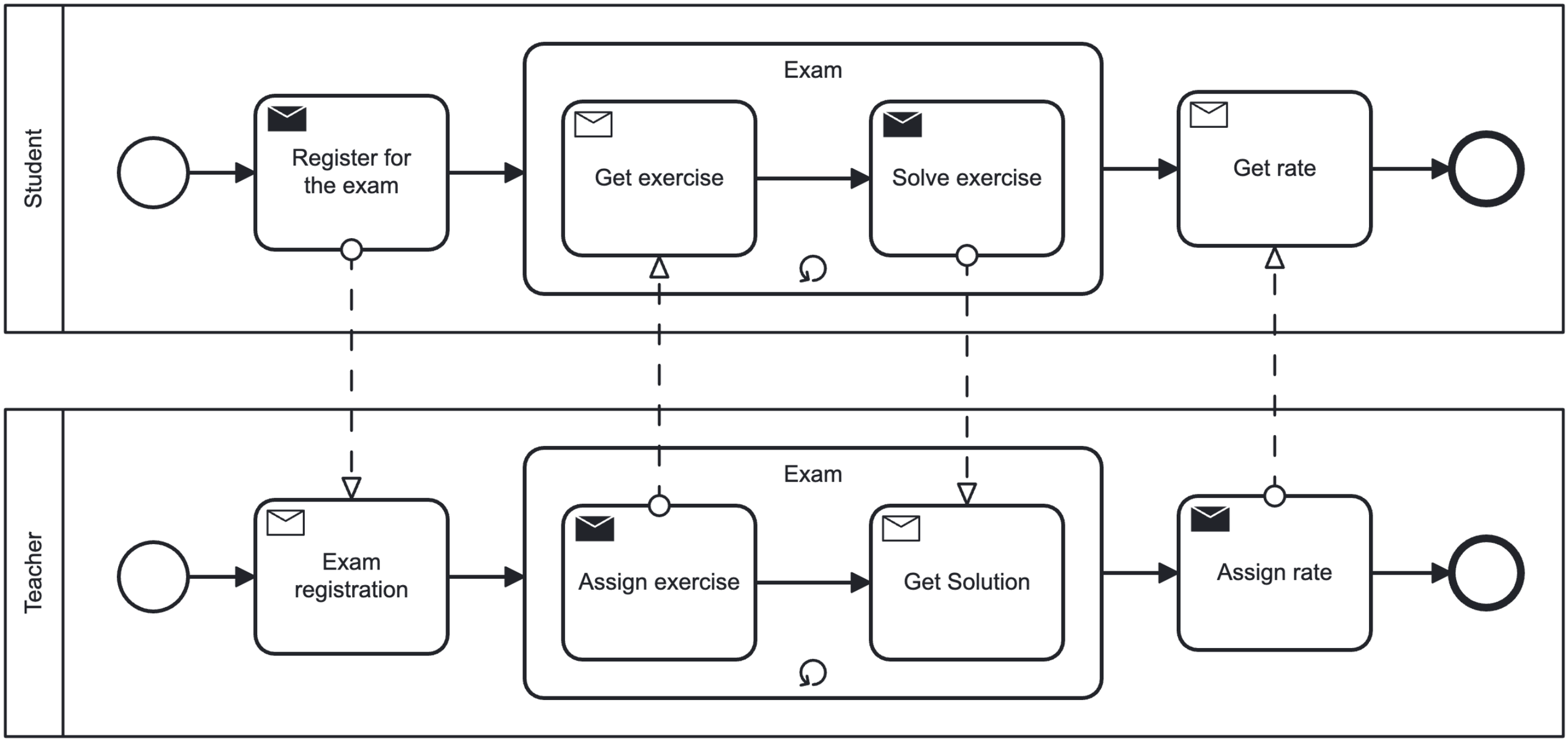


Projection (on Teacher)

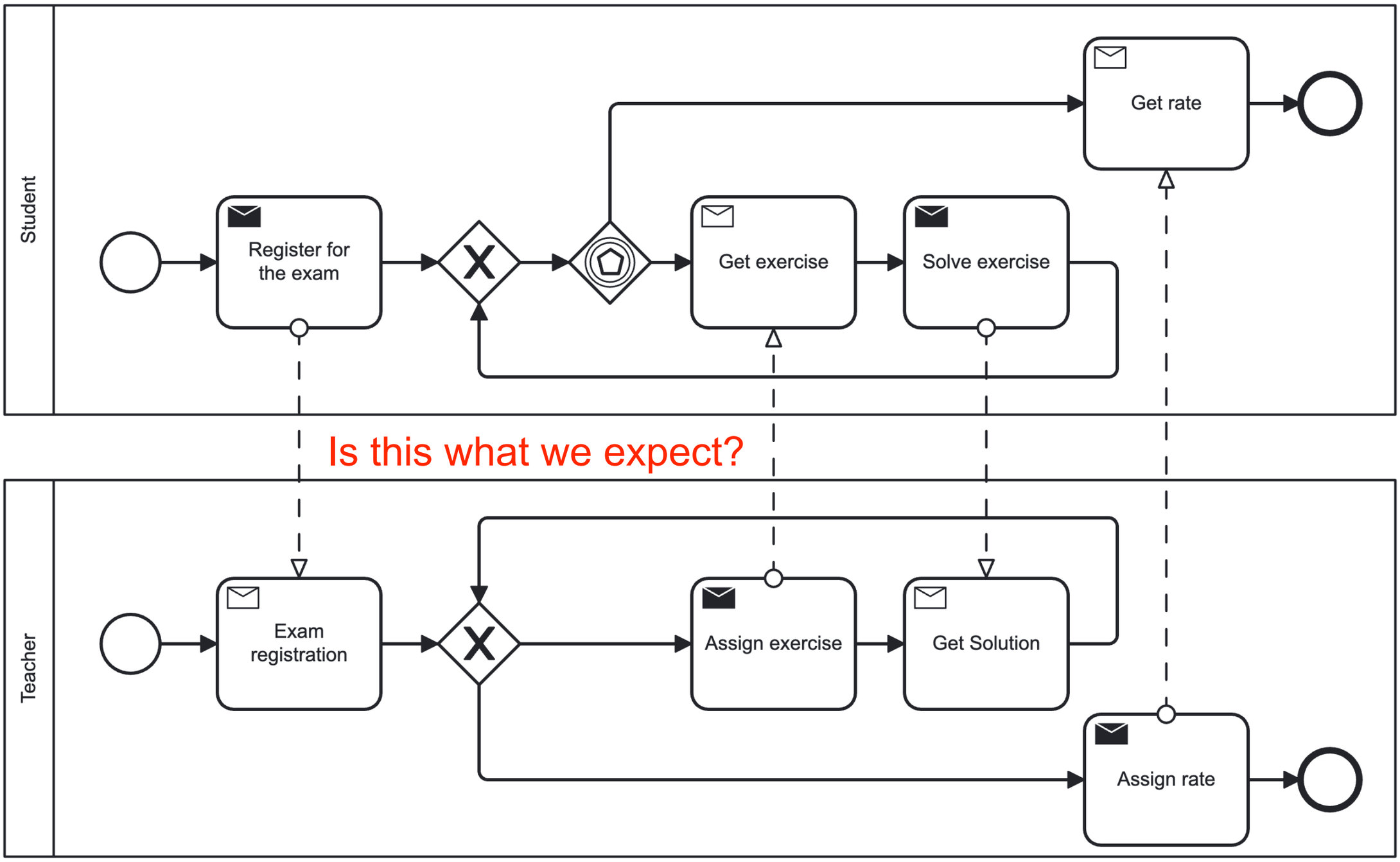




From choreographies to collaborations

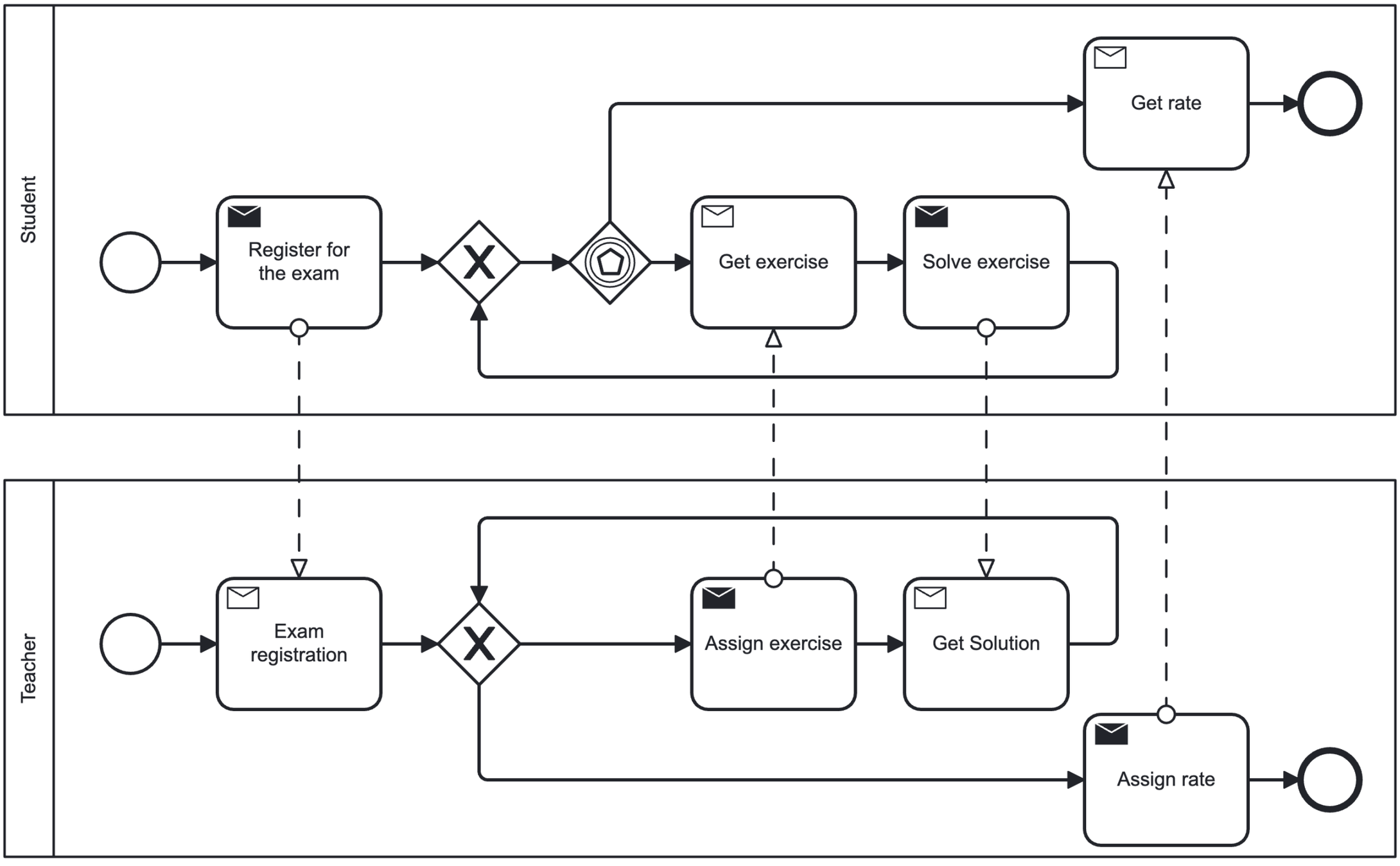


From choreographies to collaborations



Exercise

Modify the collaboration diagram to enforce the assignment of at least one exercise



Exercises

Search for EPC/BPMN diagram drawing software products.
For each product found, annotate the following features:

1) which OS is supported? (Windows, Apple, Linux,...)

2) is it free? if not, describe its pricing.

3) is .epml/.bpmn format supported?

4) if you install the product, rate your user experience / usability (on the scale 1-5 stars)

Send your findings to: `bruni@di.unipi.it`