

Business Processes Modelling

MPB (6 cfu, 295AA)

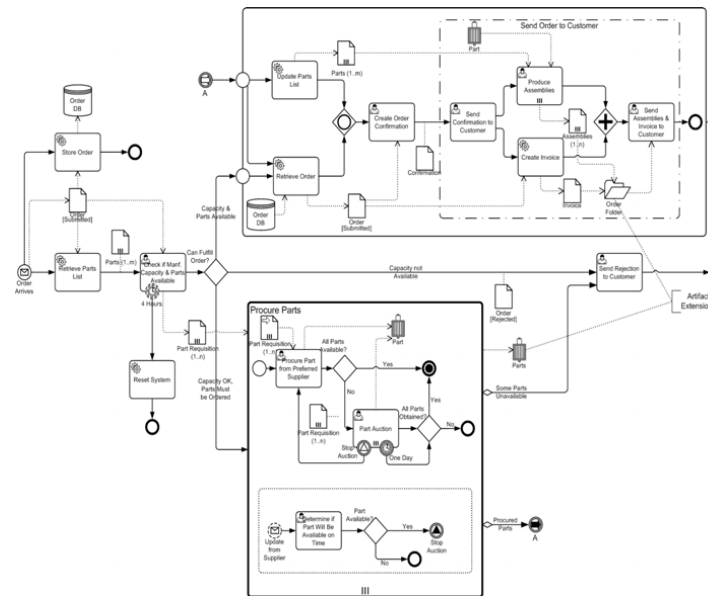
Roberto Bruni

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

07 - Business process
modelling notation



Object



We overview the BPMN notation

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

BPMN

Main goal:

to define a **graphical notation**
that is **readily understandable:**

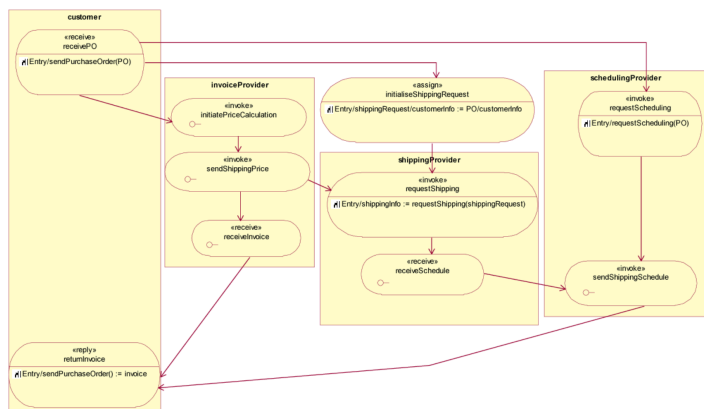
by **business analysts** (initial drafts of processes)

by **technical developers** (process implementation)

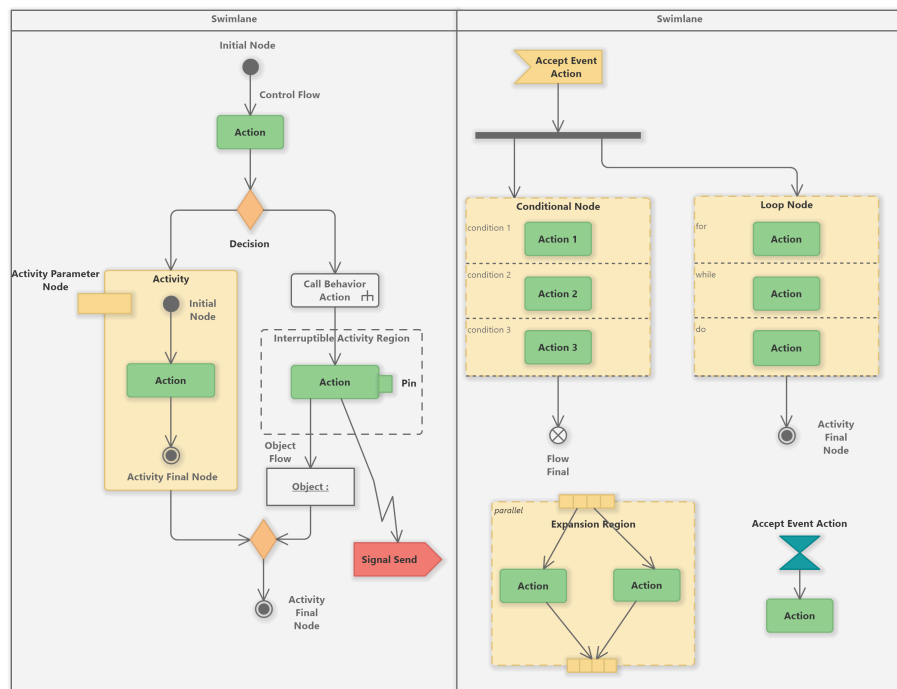
by **business people** (process management)

Before BPMN

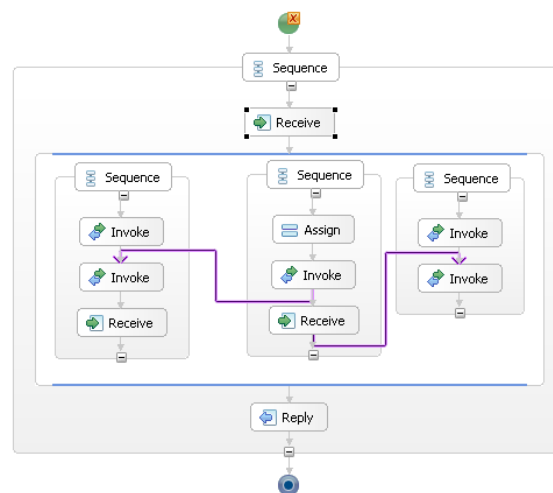
BPMD
BP Definition Metamodel



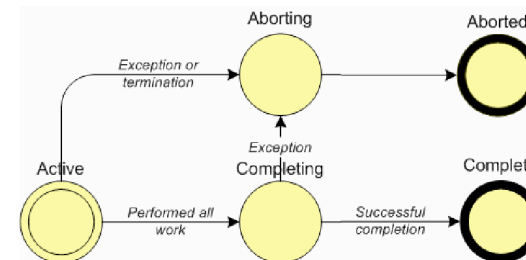
UML2 AD
Activity Diagram



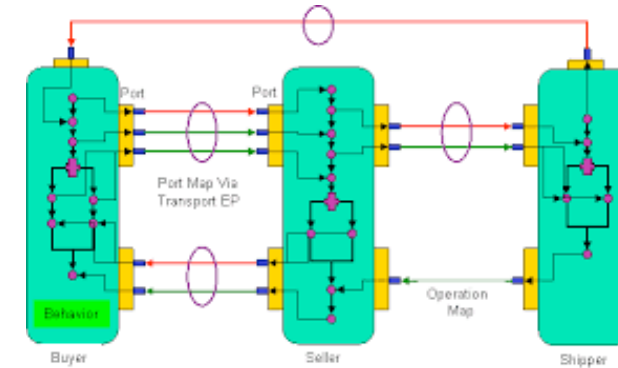
OASIS's BPEL
BP Execution Language



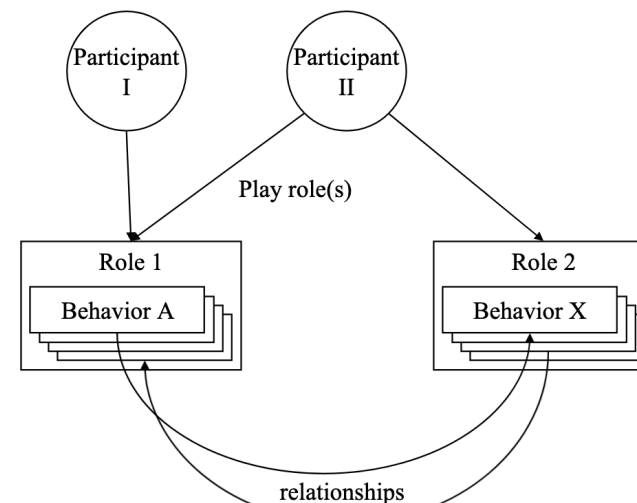
BPMI.org's BPML
BP Modelling Language



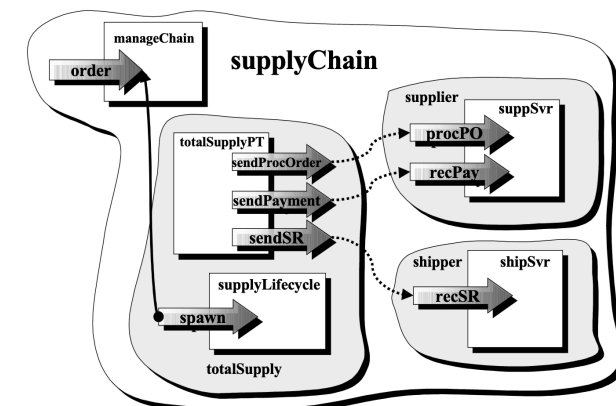
Microsoft's XLANG



W3C's WS CDL
Choreography Description Language



IBM's WSFL
WS Flow Language



A joint effort!



Standardisation

In the context of graphical models for business processes

the development of BPMN is an important step in:

reducing the fragmentation that existed
with myriad of process modelling tools and notations

exploiting experiences with many divergent proposals to
consolidate the best ideas

supporting the adoption of **inter-operable**
business process management systems

Short history

2000 - **Business Process Management Initiative** (BPMI.org)
(independent organization, studying **open specifications** for the management of **e-Business processes**)

2005 - BPMI and the Object Management Group™ (OMG™)
merge their activities on BPM forming the
Business Modeling & Integration Domain Task Force (BMI -DTF)

2006 - **BPMN 1.0 approved**

2007 - BPMN 1.1 approved

2009 - BPMN 1.2 approved

2009 - BPMN 2.0 Beta 1 proposed

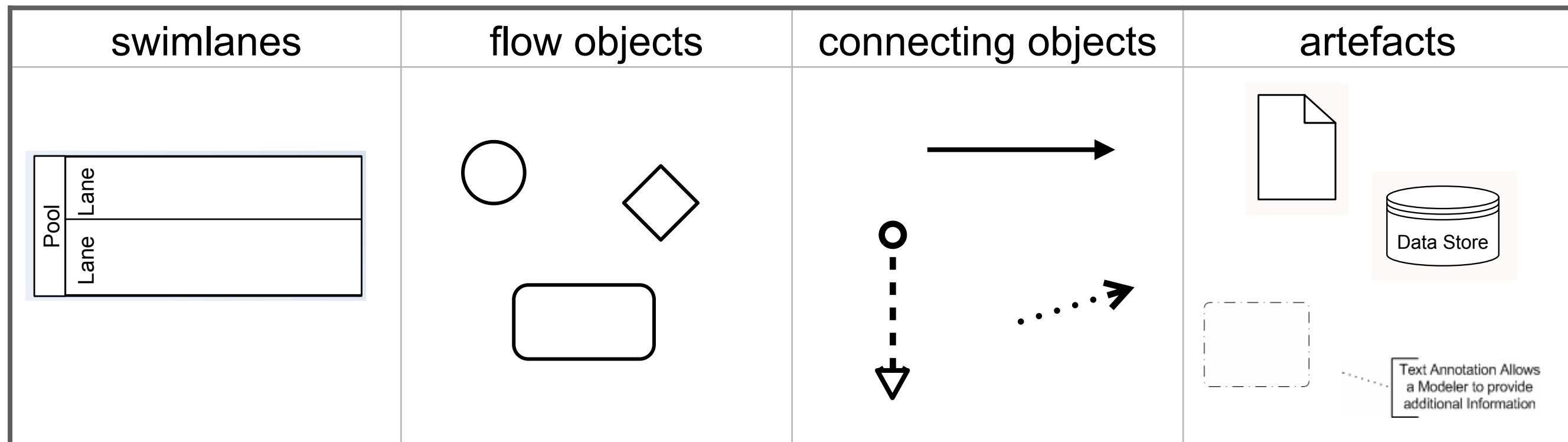
2010 - BPMN 2.0 Beta 2 proposed

2011 - **BPMN 2.0 Final delivered**

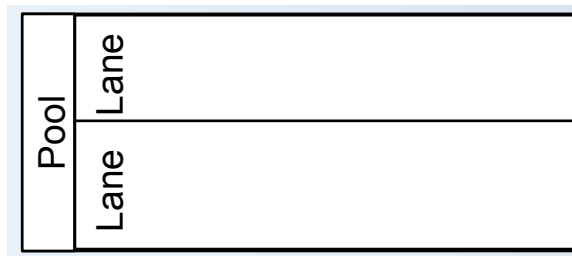
Business process diagrams

BPMN defines a standard for
Business Process Diagrams (BPD)
based on **flowcharting technique**

Four categories of elements

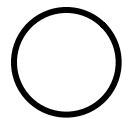


BPMN vs EPC (roughly)



swimlanes

organization unit



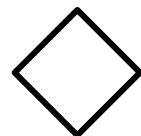
event

event



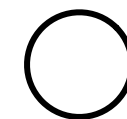
activity

function



gateway

connector



sequence flow

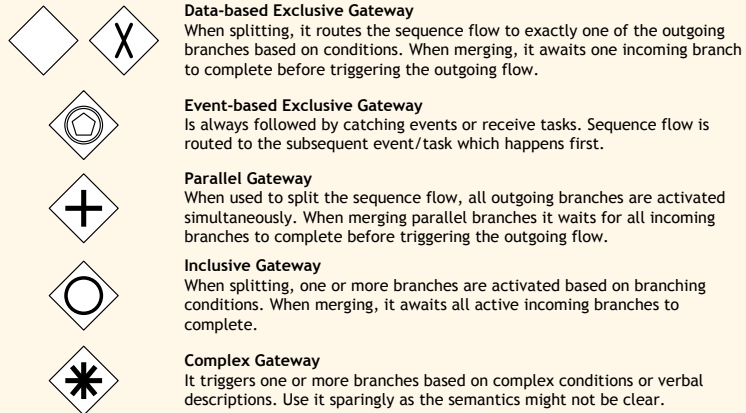
control flow



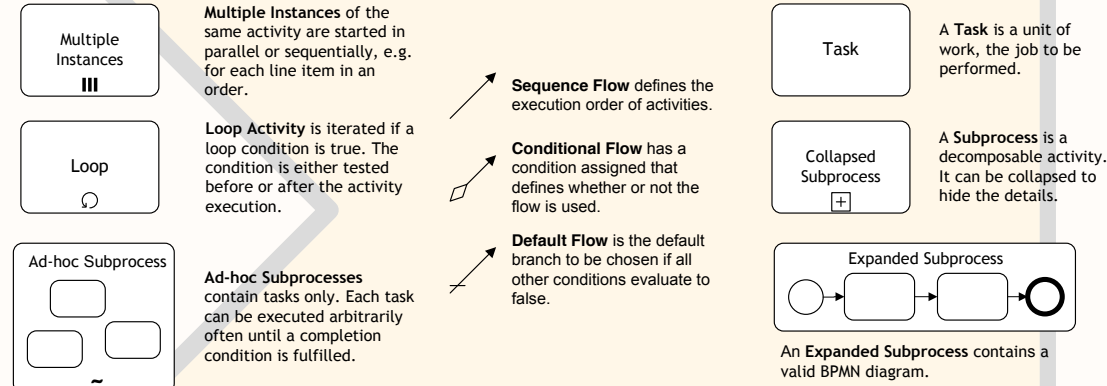
message flow

BPMN - Business Process Modeling Notation

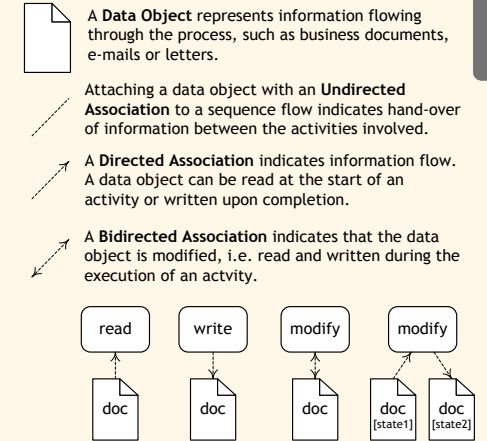
Gateways



Activities



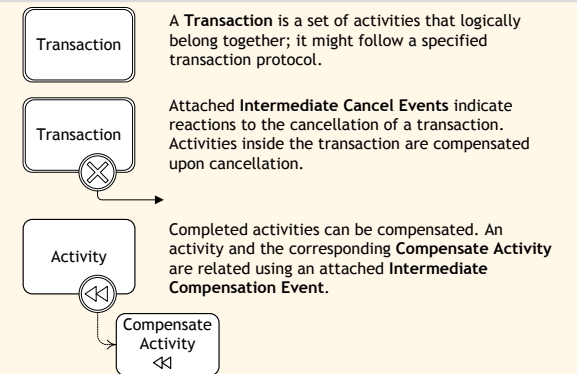
Data



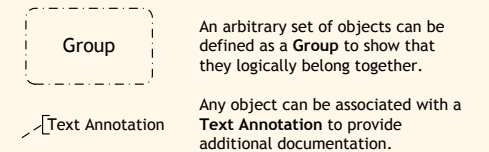
Events

	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.
Cancel				Reacting to cancelled transactions or triggering cancellation.
Compensation				Compensation handling or triggering compensation.
Conditional				Reacting to changed business conditions or integrating business rules.
Signal				Signalling across different processes. One signal thrown can be caught multiple times.
Multiple				Catching or throwing one out of a set of events.
Link				Off-page connectors. Two corresponding link events equal a sequence flow.
Terminate				Triggering the immediate termination of a process.

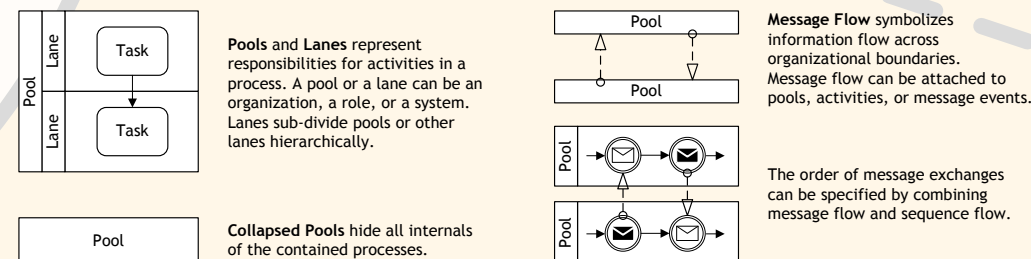
Transactions



Documentation



Swimlanes



Business Process Technology
Prof. Dr. Mathias Weske
Web: bpt.hpi.uni-potsdam.de
Oryx: oryx-project.org
Blog: bpmn.info
BPMN Version 1.2

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Authors
Gero Decker
Alexander Grosskopf
Sven Wagner-Boysen

ORX

BPMN 2.0 vs 1.0

Updated (new markers):

Events

Activities

Gateways

Artefacts

Added:

Choreographies

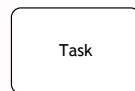
Full metamodel

XML Serialization

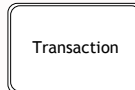
Diagram Interchange

BPMN Execution Semantics (verbal)

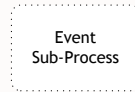
Activities



A **Task** is a unit of work, the job to be performed. When marked with a **+** symbol it indicates a **Sub-Process**, an activity that can be refined.



A **Transaction** is a set of activities that logically belong together; it might follow a specified transaction protocol.



An **Event Sub-Process** is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.



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

Activity Markers

Markers indicate execution behavior of activities:

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

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

- Sequence Flow**
defines the execution order of activities.
- Default Flow**
is the default branch to be chosen if all other conditions evaluate to false.
- Conditional Flow**
has a condition assigned that defines whether or not the flow is used.

Conversations



A **Communication** defines a set of logically related message exchanges. When marked with a **+** symbol it indicates a **Sub-Conversation**, a compound conversation element.

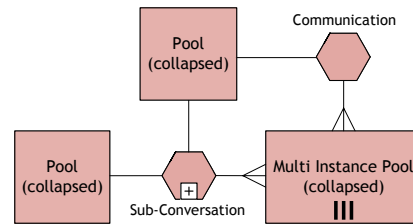


A **Conversation Link** connects Communications and Participants.

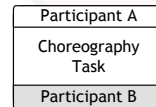


A **Forked Conversation Link** connects Communications and multiple Participants.

Conversation Diagram



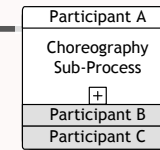
Choreographies



A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

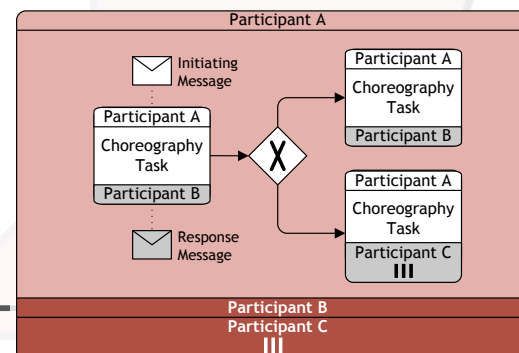
III

Multiple Participants Marker denotes a set of Participants of the same kind.

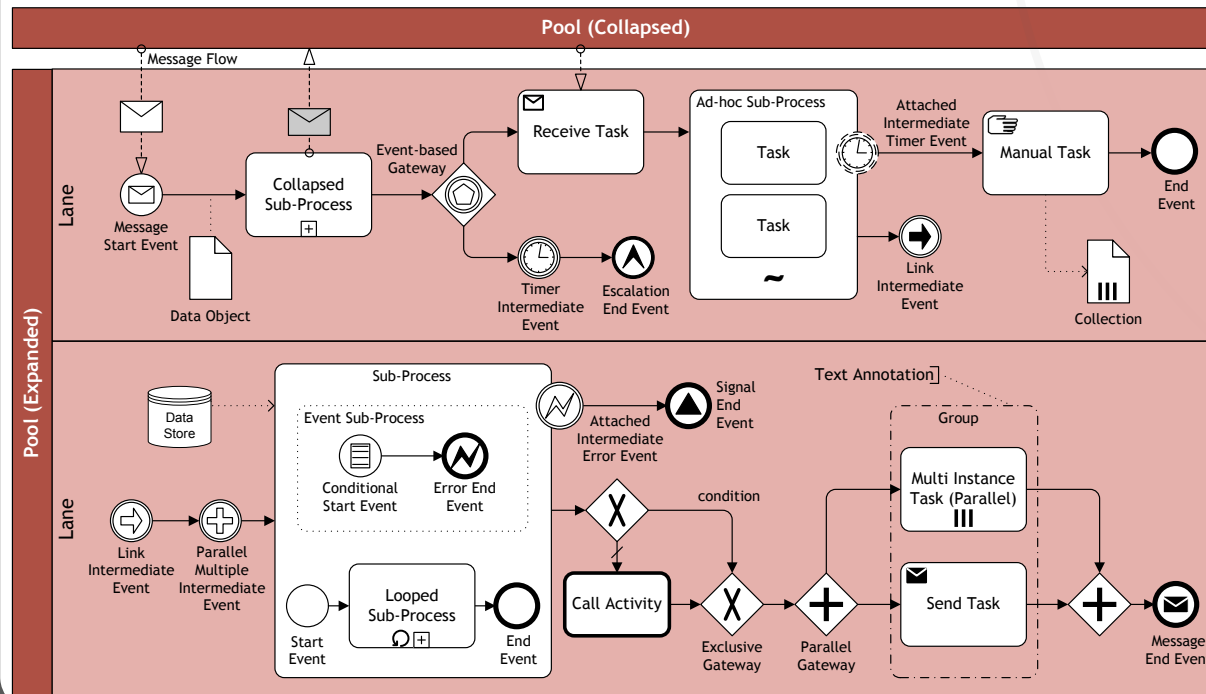


A **Choreography Sub-Process** contains a refined choreography with several Interactions.

Choreography Diagram



Collaboration Diagram



Events

	Top-Level	Start	Intermediate	End
	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Throwing
None: 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.				

Data



A **Data Input** is an external input for the entire process. It can be read by an activity.

A **Data Output** is a variable available as result of the entire process.



A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.



A **Collection Data Object** represents 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.

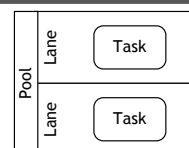


A **Message** is used to depict the contents of a communication between two Participants.

Gateways

- Exclusive Gateway**
When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.
- Event-based Gateway**
Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.
- Parallel Gateway**
When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.
- Inclusive Gateway**
When splitting, one or more branches are activated. All active incoming branches must complete before merging.
- Exclusive Event-based Gateway (instantiate)**
Each occurrence of a subsequent event starts a new process instance.
- Complex Gateway**
Complex merging and branching behavior that is not captured by other gateways.
- Parallel Event-based Gateway (instantiate)**
The occurrence of all subsequent events starts a new process instance.

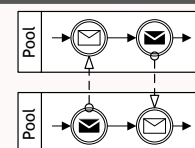
Swimlanes



Pools (Participants) and Lanes represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.



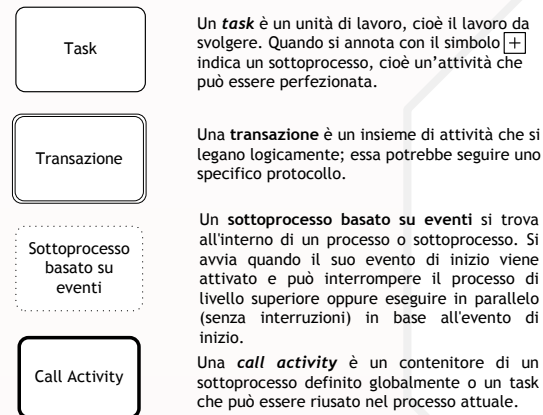
Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.



The order of message exchanges can be specified by combining message flow and sequence flow.




Attività



Simboli per attività




I seguenti simboli indicano il comportamento di esecuzione delle attività:

-  Sottoprocesso
-  Loop
-  Esecuzione in parallelo
-  Esecuzione sequenziale
-  Ad hoc
-  Compensazione

Tipologie di tasks

Le tipologie specificano la natura dell'azione da eseguire

- Task di invio
- Task di ricezione
- Utente
- Task manuale
- Regole di business
- Service
- Script

Flusso sequenziale	Flusso predefinito	Flusso condizionale
		
definisce l'ordine di esecuzione delle attività.	è il ramo predefinito da scegliere se tutte le altre condizioni vengono valutate come false.	ha una condizione assegnata che definisce se usare o meno il flusso.

Conversazioni

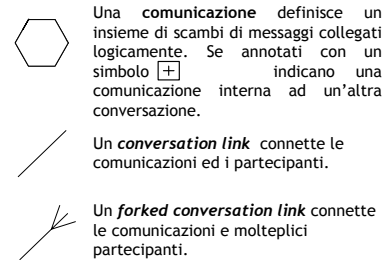
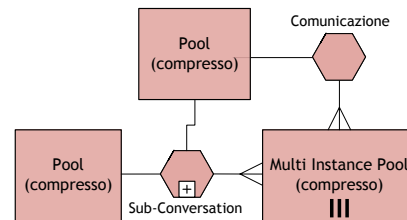


Diagramma di conversazione



Coreografie

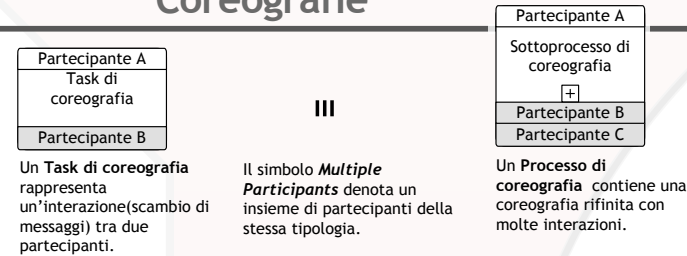
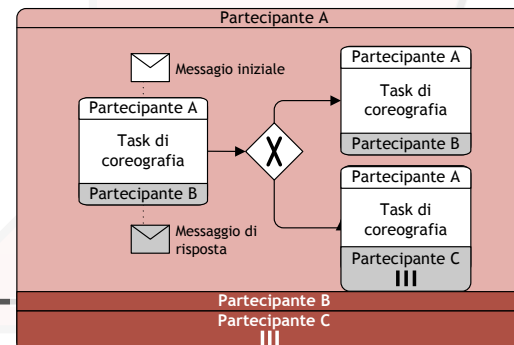
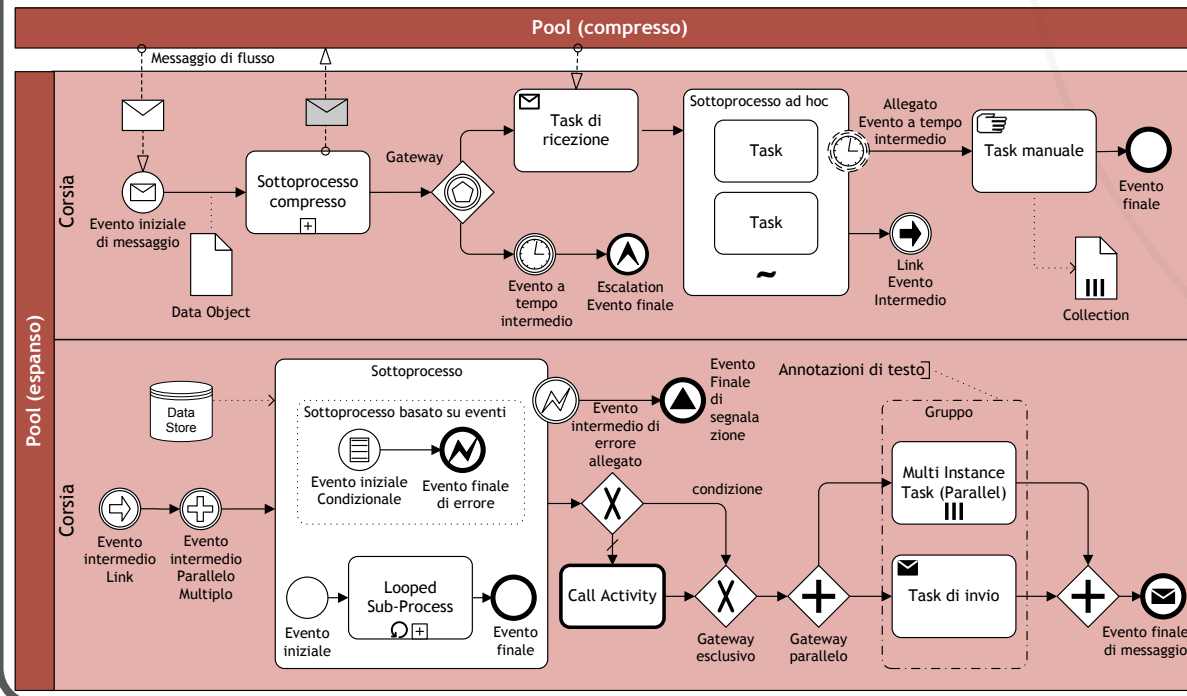


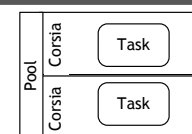
Diagramma di coreografia



Collaboration Diagram



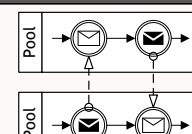
Swimlanes



Pools (Partecipanti) e Lanes(corse)
rappresentano le responsabilità per le attività in un processo. Esse possono essere un'organizzazione, un ruolo o un sistema. Le corsie suddividono le *pools* o altre *corsie* gerarchicamente.



Flusso di messaggi
rappresenta il flusso di informazioni. Un flusso di messaggi può essere unito a pools, attività, o eventi di messaggi.



L'ordine degli scambi di messaggi può essere specificato associando il flusso di messaggi e il flusso sequenziale.

Eventi

	Alto livello	Interruzione di sottoprocessi	Non-interruzione di sottoprocessi	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	Fine
Non definiti: punti di inizio, cambi di stato, o stati finali.								
Messaggio: invio e ricezione di messaggi								
Timer: eventi a tempo.								
Escalation: passa ad un livello più alto di responsabilità.								
Condizionale: reagisce a condizioni di business cambiate o integra regole di business.								
Link: Due corrispondenti <i>link events</i> sono uguali ad un flusso sequenziale.								
Errore: attiva o si occupa di un errore.								
Cancel: reagisce a delle transazioni cancellate o causa una cancellazione.								
Compensazione: gestisce o innesca la compensazione.								
Signal: comunica con più processi. Lo stesso segnale può essere intercettato più volte.								
Multiplo: intercetta uno tra vari eventi. Gestisce tutti gli eventi definiti.								
Paralelo Multiplo: intercetta tutti gli eventi.								
Terminate: causa la fine immediata di un processo.								

Data



Un **Data Input** è un input esterno usato all'interno del processo. Può essere letto da un'attività.

Un **Data Output** è una variabile disponibile come risultato di un intero processo.

Un **Data Object** rappresenta le informazioni che attraversano l'intero processo, come ad esempio documenti di business, e-mails, o lettere.

Un *Collection Data Object* rappresenta una collezione di informazioni, come ad esempio una lista di elementi ordinati.

Un **Data Store** è un luogo dove il processo può leggere oppure scrivere dati, ad esempio un database. Esso si mantiene oltre la durata dell'istanza del processo.

Un **messaggio** è usato per rappresentare i contenuti di una comunicazione tra due partecipanti.



BPMN 2.0 (2009/11)

FAQ

What is BPMN?

BPMN is a graphical notation that depicts the steps (end to end flow) in a business process.

Specifically designed to coordinate the sequence of processes and **the messages that flow** between participants in a related set of activities.

BPMN 2.0 (2009/11)

FAQ

Why is BPMN important?

The world of business processes has changed dramatically over the past few years. Processes can be coordinated from behind, within and over organizations boundaries. A business process now spans multiple participants and coordination can be complex.

Until BPMN, there has not been a standard modelling technique developed that addresses these issues. BPMN provides users with a **royalty free notation**.

This will benefit users in a similar manner in which UML standardised the world of software engineering. There will be training courses, books and a body of knowledge that users can access in order to better implement a business process.

BPMN 2.0 (2009/11)

FAQ

Will there be a major rewrite?

Not for 2 or 3 years...

(good work! 13+ years and still no revision is planned)

Strengths of BPMN

Simplicity: A small set of basic symbols

Extensibility: many decorations available
(new ones can be added in the future)

Graphical design: **intuitive**

Generality: **orchestration + collaboration + choreography**

Tool availability: **.bpmn exchange format**

Weaknesses of BPMN

over 100 graphical elements

verbose description (500 pages)

difficult to learn comprehensively:

different readings of the same diagram are possible

different BPMN vendors implement the execution of BPMN diagrams in different ways (and for different subsets)

BPMN basics

BPMN key features

More on BPMN

BPMN semantics

Swimlanes (pools, lanes)

Swimlanes

A **swimlane** is a mechanism to **organise activities into separate visual categories** to illustrate different capabilities or **responsibilities**

Present in many process modelling methodologies

BPMN supports two main swimlane objects:

pool

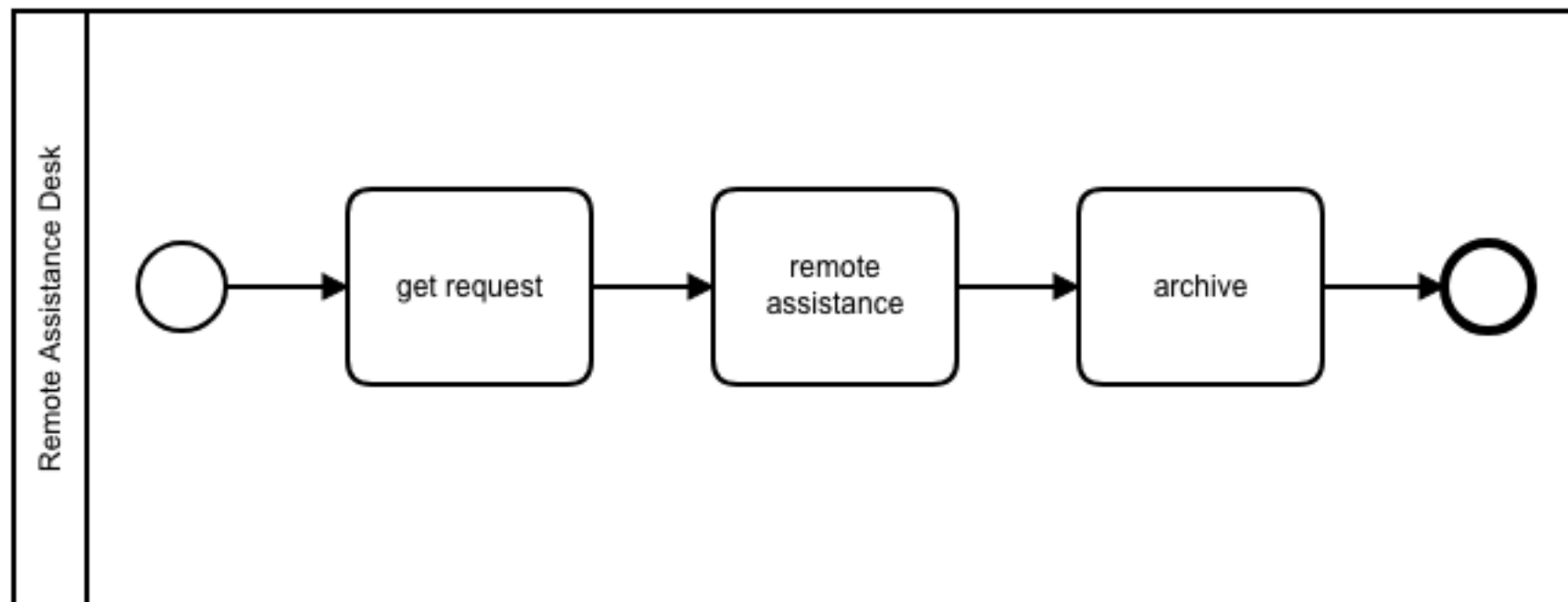


lanes



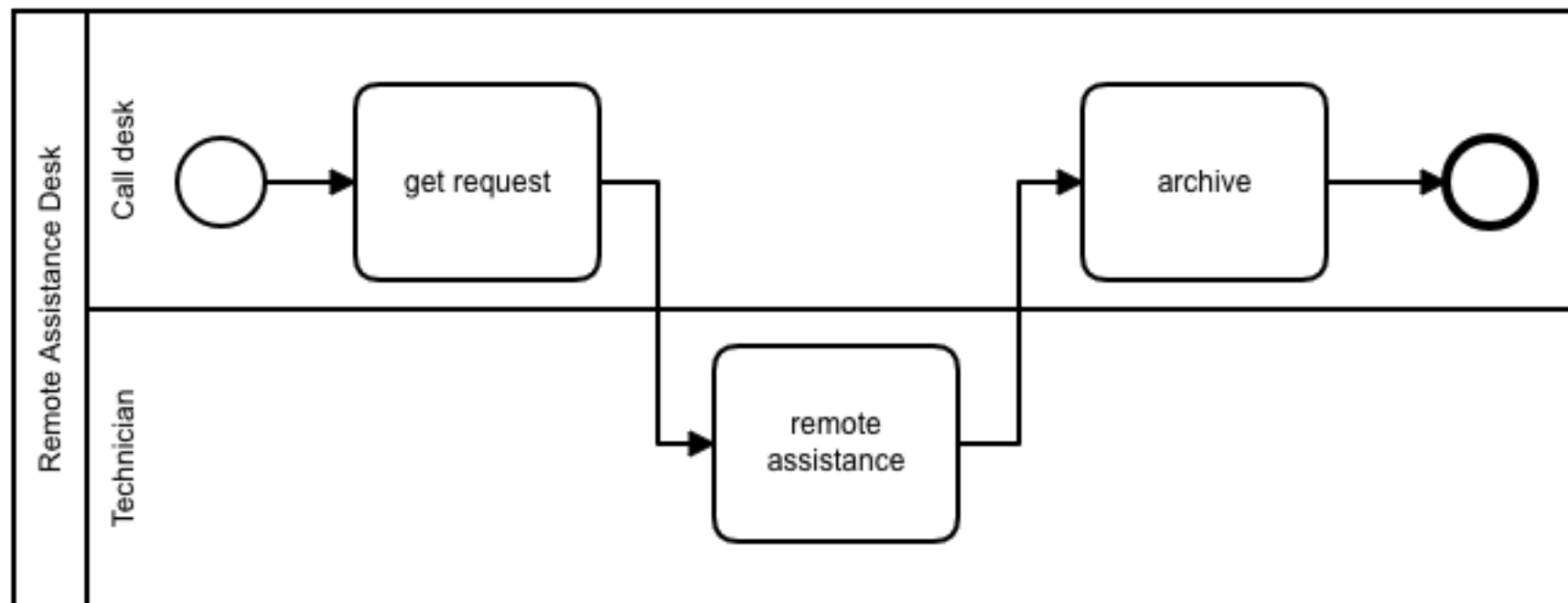
Pools

A **pool** represents a participant (or role) in a process (represented as a rectangle with a name)



Lanes

A **lane** is a hierarchical sub-partition within a pool that is used to organise and categorise activities (inner rectangle that extends to the entire length of the pool)



Collapsed pools

Internal process is not exposed
(like a black-box)



Requirements

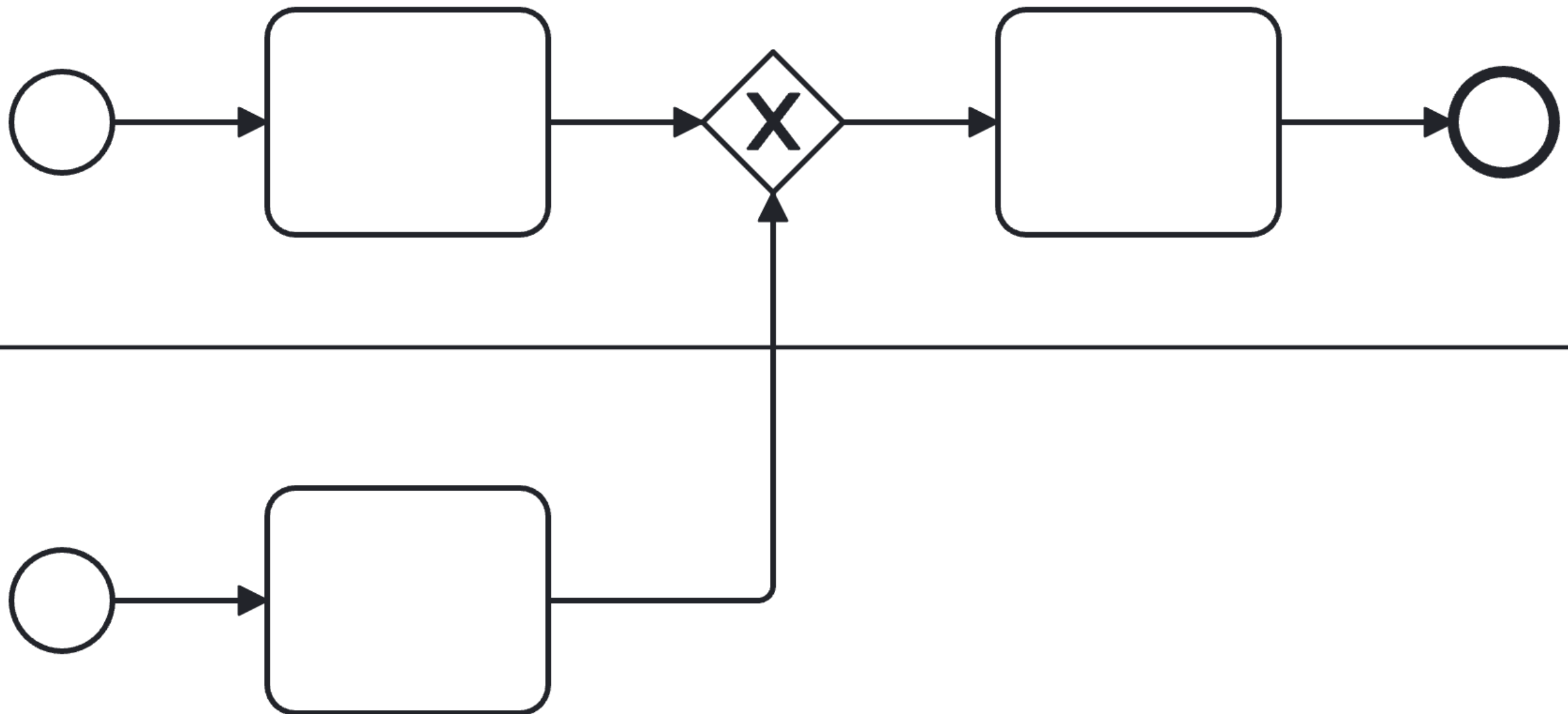


A Pool **MUST** contain 0 or 1 business process.

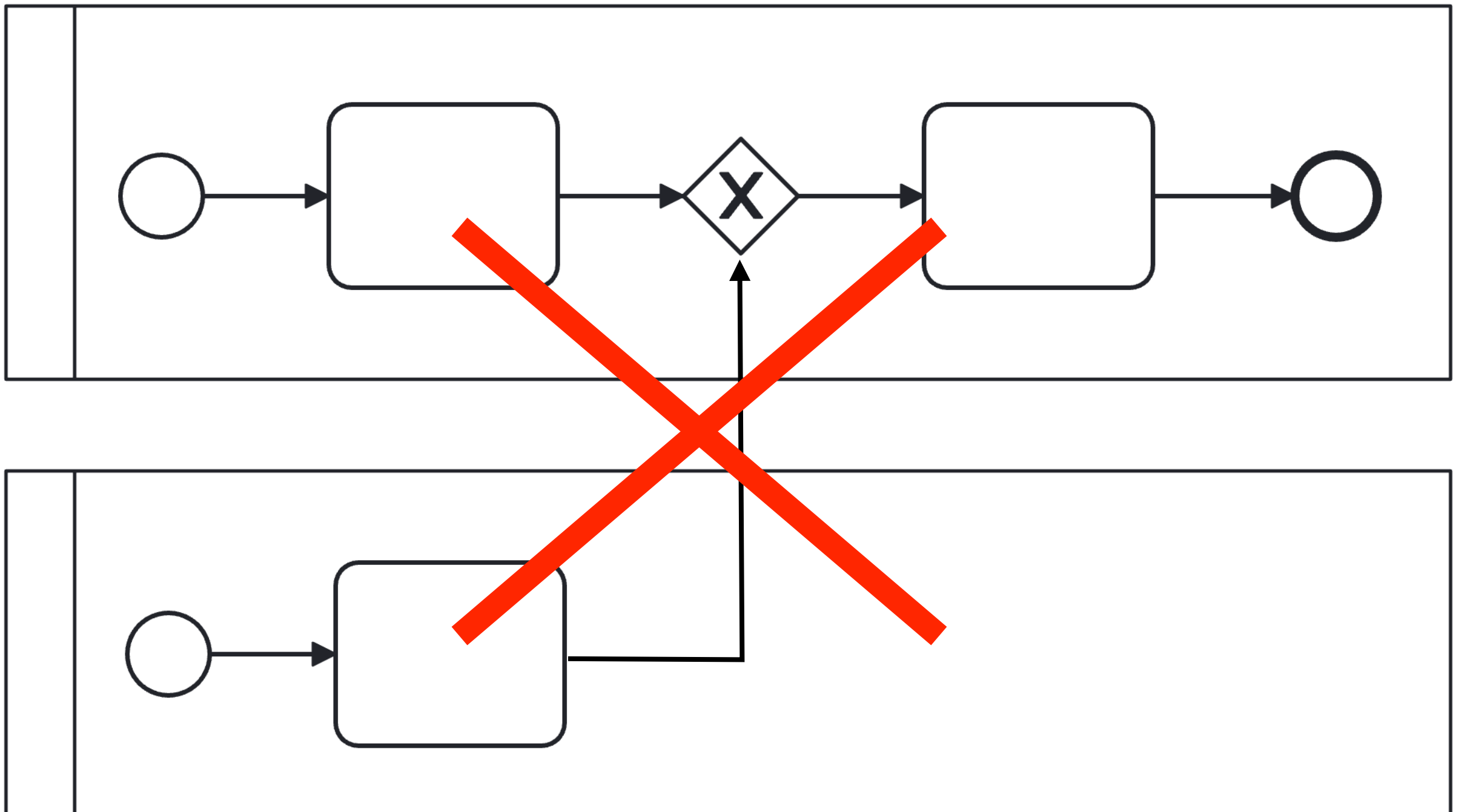
A Pool can contain 0 or more lanes.

Two pools can only be connected with message flows.

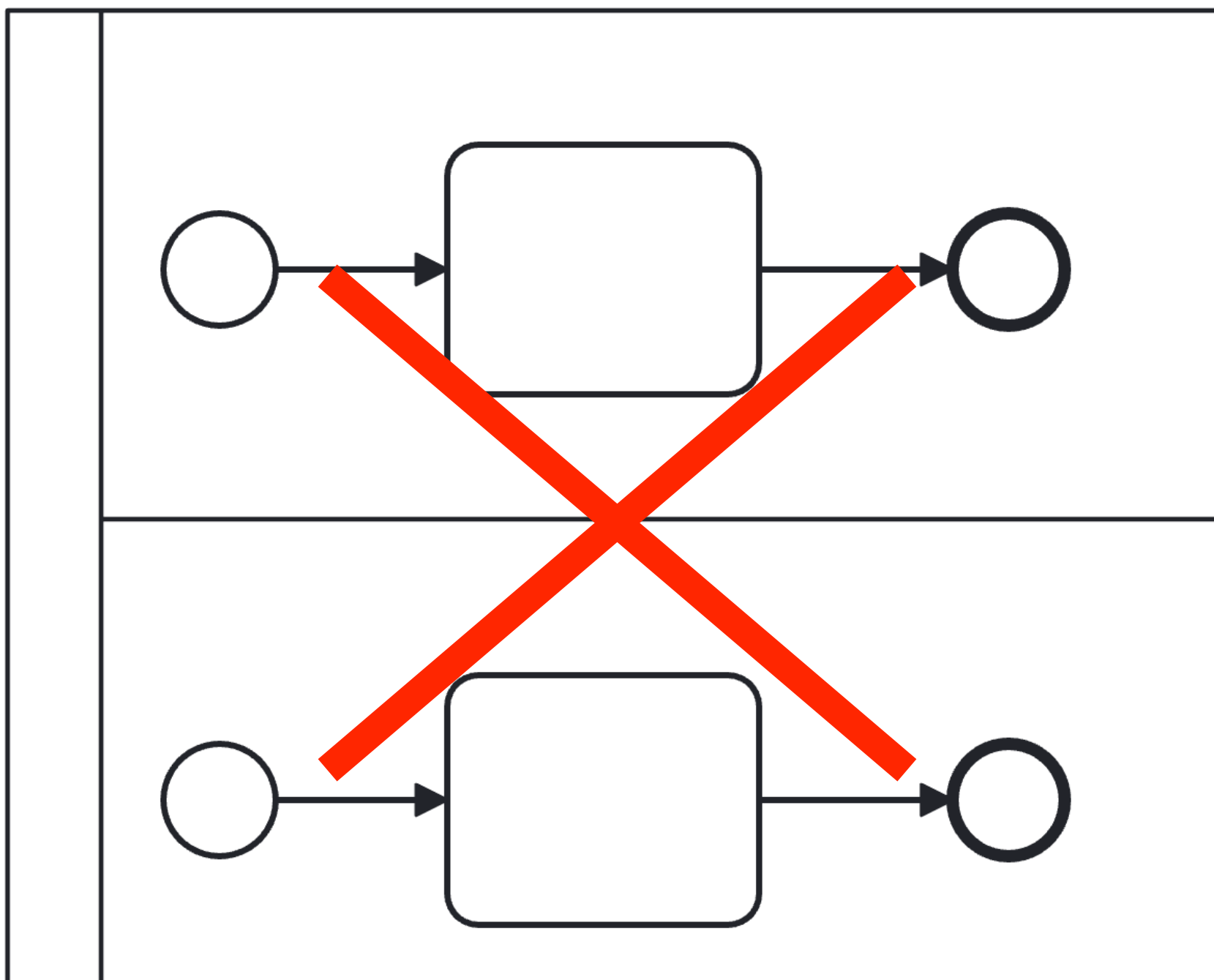
Example



Example



Example



Naming conventions

Process models:

a noun possibly preceded by an adjective

the label is often obtained by “nominalizing” the verb that describe the main action in the process
(e.g., **claim handling**, **order fulfillment**)

Avoid long labels

Articles are often omitted

Flow Objects

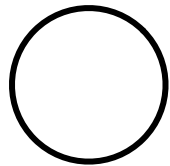
(events, activities, gateways)

Flow objects

Rationale:

fix a small set of core elements
so that modellers must learn a small number of shapes:

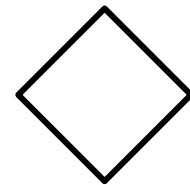
events



activities



gateways



Flow objects

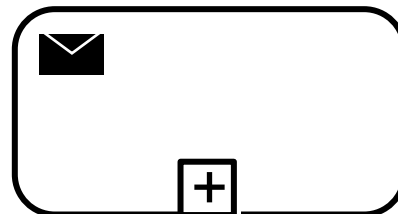
Rationale:

fix a small set of core elements
so that modellers must learn a small number of shapes:

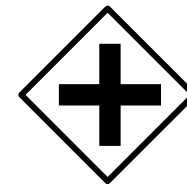
events



activities



gateways



use different border styles and internal markers
to add many more information
(this way the notation is **extensible**)

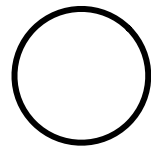
Flow objects: Events

Events

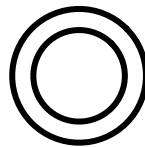
An **event** is something that “happens” during the course of a business process

An event is represented as a circle
different borders define the **type** of the event

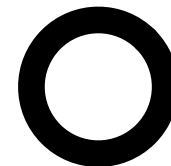
start



intermediate



end



Naming conventions

Events:

the label should begin with a noun and end with a verb in past participle form to indicate something that just happened
(e.g., **Invoice emitted**)

the noun can be preceded by an adjective
(e.g., **Urgent order sent**)

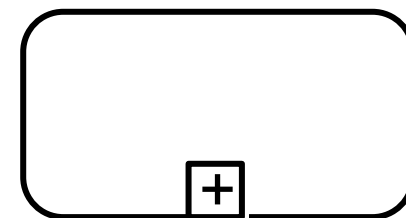
Avoid long labels
Articles are often omitted

Flow objects: *Activities*

Activities

An **activity** is some “unit of work” (job) to be done during the course of a business process

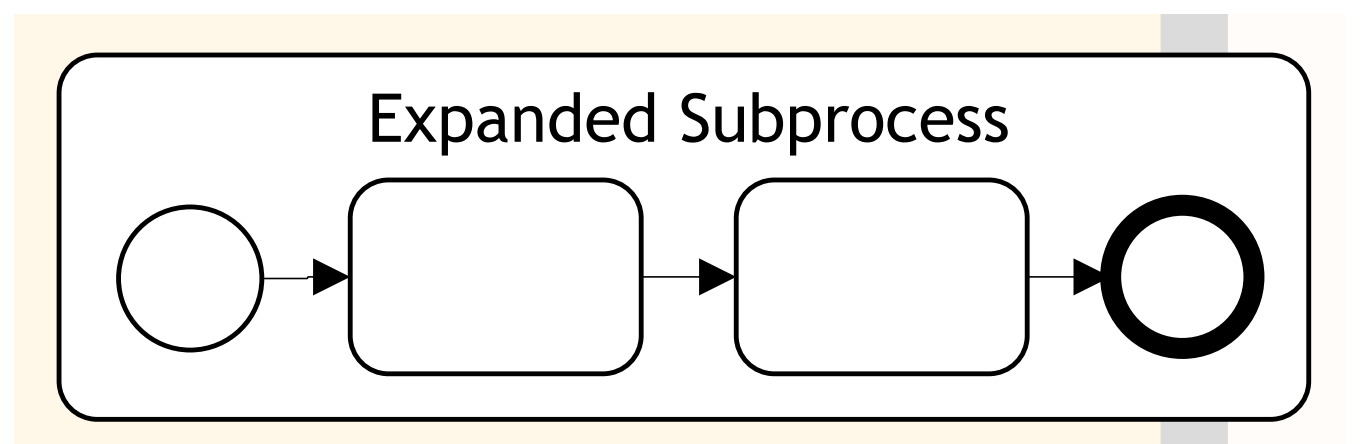
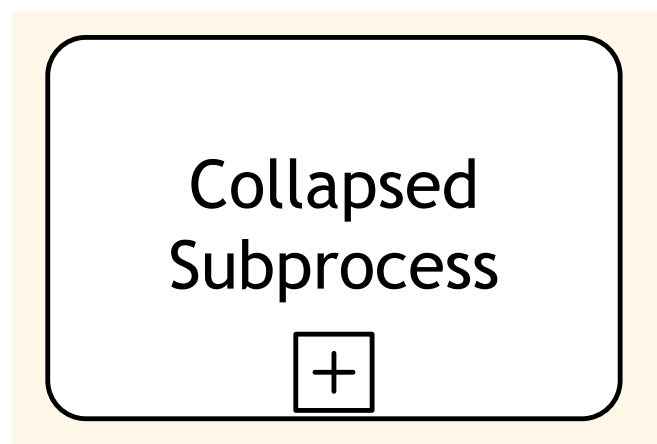
An activity is represented as a rounded box
BPMN has two main types of activities
atomic (**task**) or compound (**sub-process**)



Sub-processes

Large process models are hard to parse:
we improve readability
by hiding certain parts within sub-processes

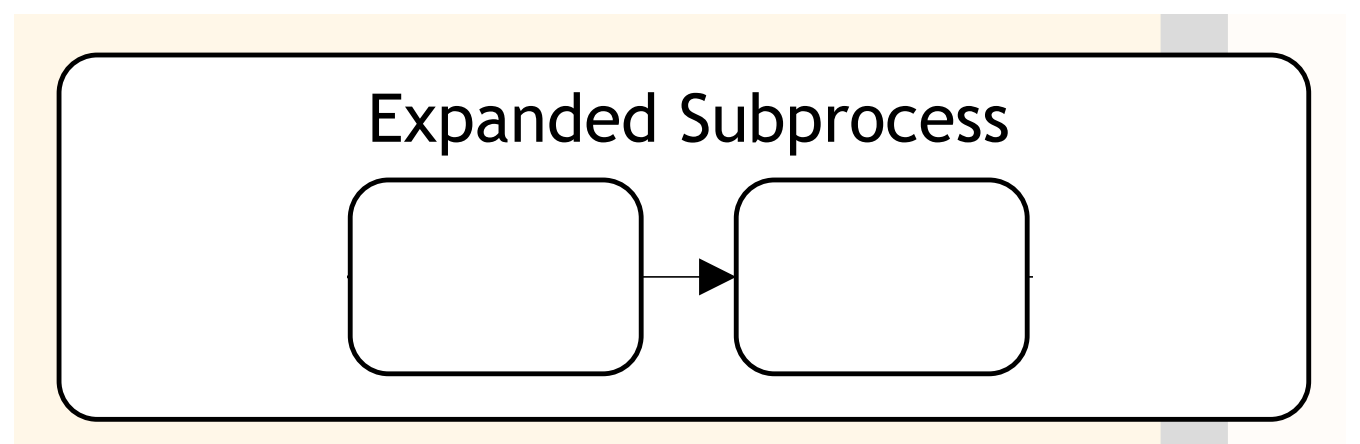
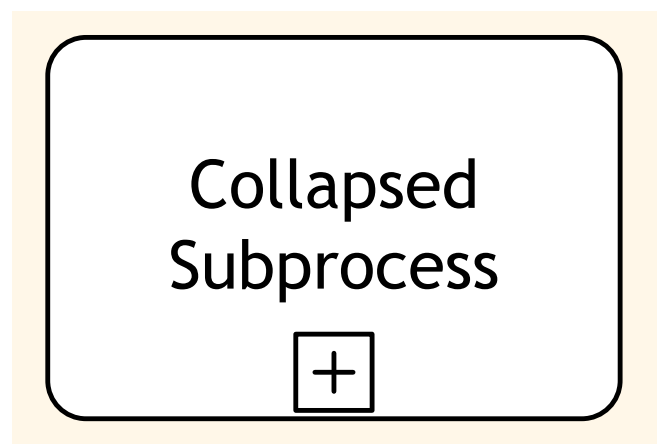
A **sub-process** is a self-contained, composite activity that can be broken into smaller units of work



Sub-processes

Large process models are hard to parse:
we improve readability
by hiding certain parts within sub-processes

A **sub-process** is a self-contained, composite activity that can be broken into smaller units of work



implicit start / end

Naming conventions

Activities:

verb in the imperative form followed by a noun
(e.g., **Approve order**)

the noun can be preceded by an adjective
(e.g., **Issue driver license**)

the verb may be followed by a complement
(e.g., **Renew driver license via offline agencies**)

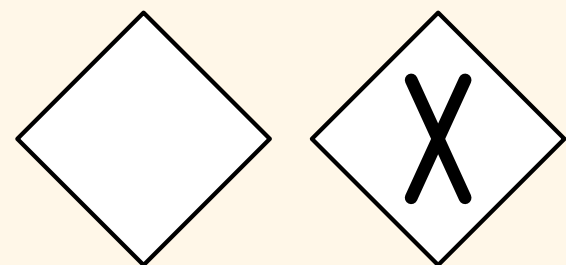
Avoid long labels
Articles are often omitted

Flow objects: Gateways

Gateways

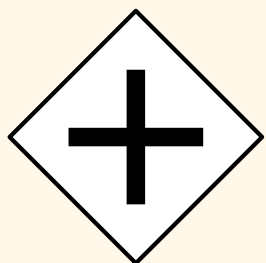
A **gateway** is used to split/join the sequence flow (conditional, fork, wait)

A gateway is represented as a diamond shape
internal markers indicate the nature of behaviour control



Data-based Exclusive Gateway

When splitting, it routes the sequence flow to exactly one of the outgoing branches based on conditions. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.



Parallel Gateway

When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Connecting objects

(sequence flow, message flow, association)

Connecting objects

The Flow objects are connected together in a diagram to create the basic skeletal structure of a business process

Three connecting objects can be used:

Sequence flow



connected objects must
reside in the same pool
(but they can be in
different lanes)

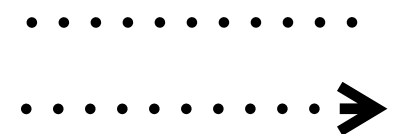
Message flow



connected objects must
reside in different pools

to be discussed later

Association



connects flow objects
with artefacts

to be discussed later

Sequence flow

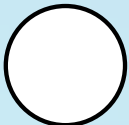
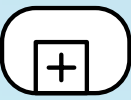


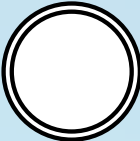

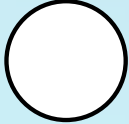























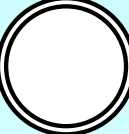





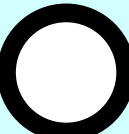
A **sequence flow** is used to show the order in which activities are to be performed

the term “control flow” is generally avoided in BPMN

A sequence flow is represented by a solid line with a solid arrowhead



Requirements

		To:					
From:							
							
							
							
							
							
							

each event:
at most one incoming and
at most one outgoing
sequence flow

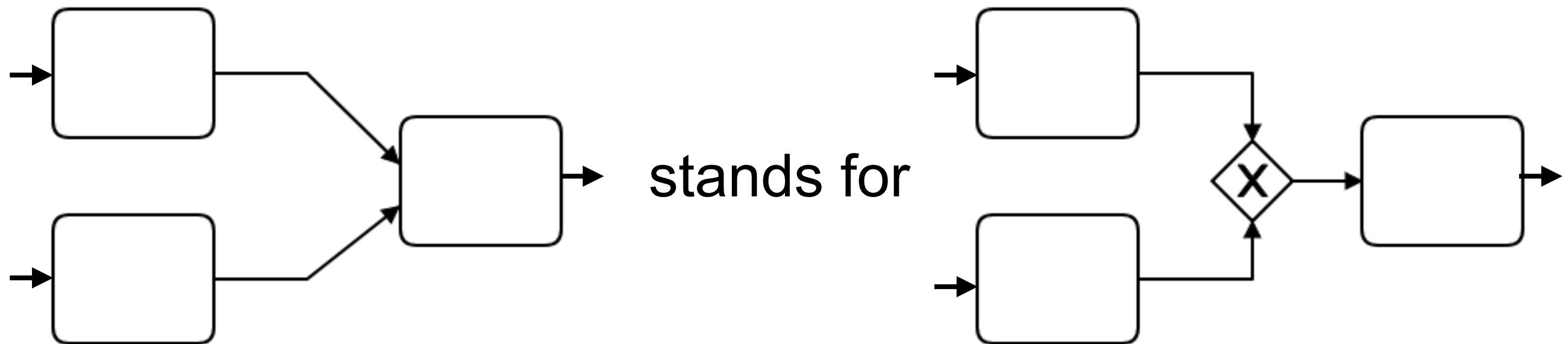
each activity:
exactly one incoming and
exactly one outgoing
sequence flow

each gateway:
one-to-many,
many-to-one,
many-to-many

Multiple flows and implicit gateways

In principle each activity should have exactly:
one incoming arc, one outgoing arc

Be careful if this is not the case!

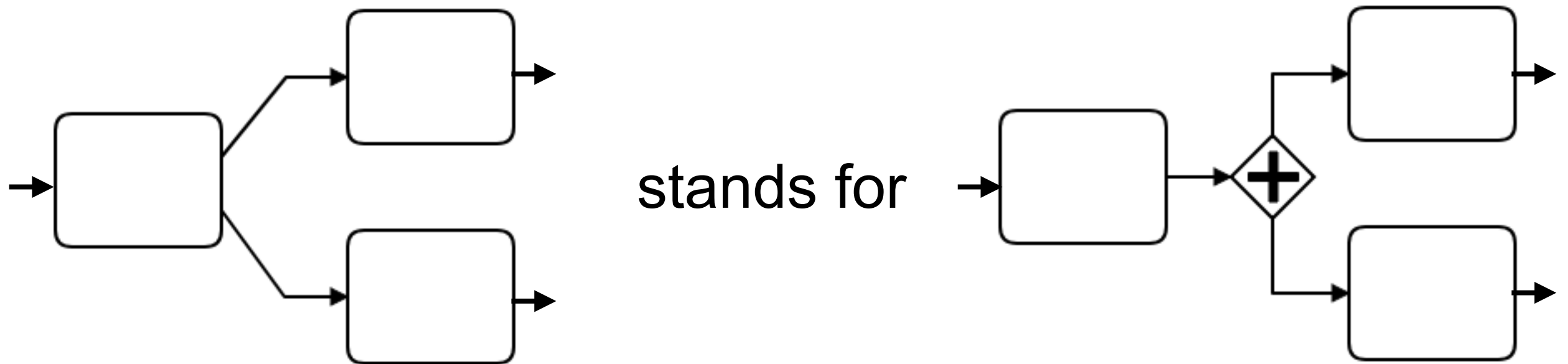


Multiple incoming flows are mutually exclusive

Multiple flows and implicit gateways

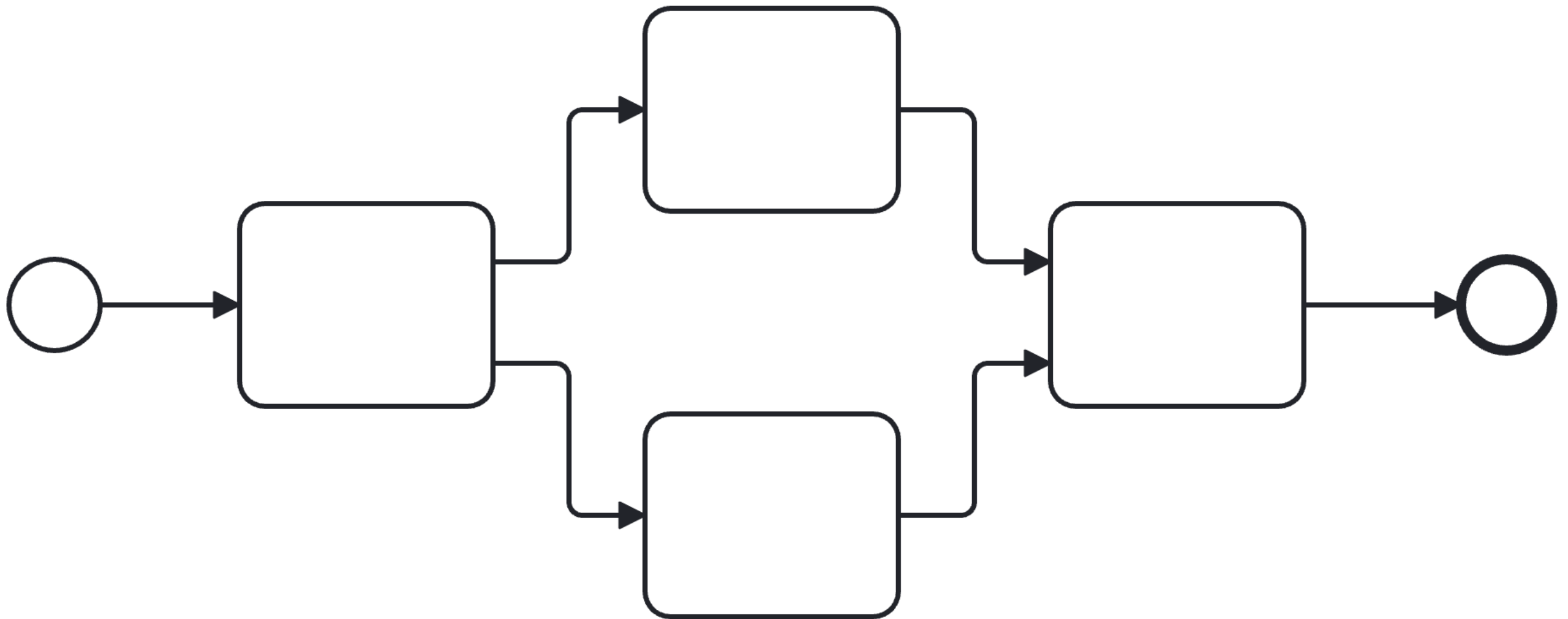
In principle each activity should have exactly:
one incoming arc, one outgoing arc

Be careful if this is not the case!



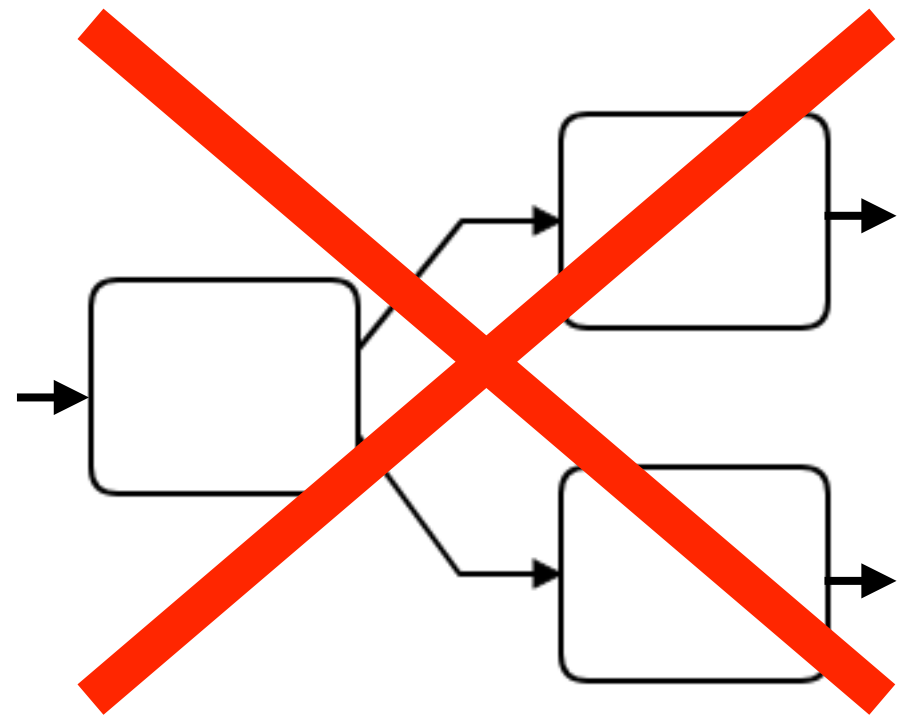
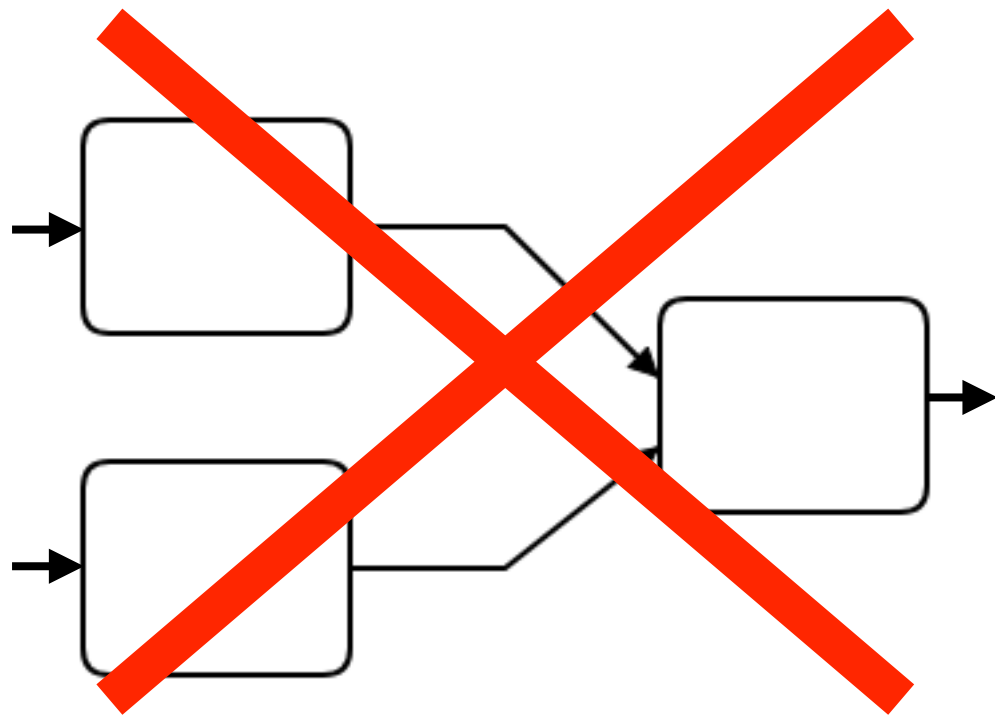
Multiple outgoing flows are activated in parallel
(unless conditions are attached to them)

Hidden issue!



In your final projects

Please avoid



Artefacts:

(data-objects, groups, text annotations)

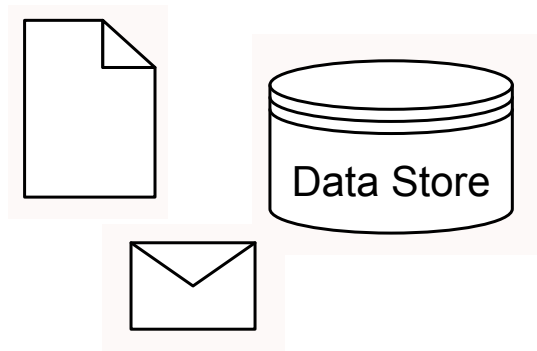
Artefacts

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:

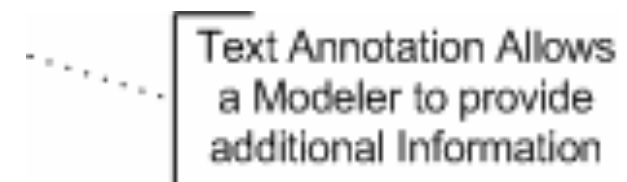
data objects	groups	text annotation
---------------------	---------------	------------------------



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)

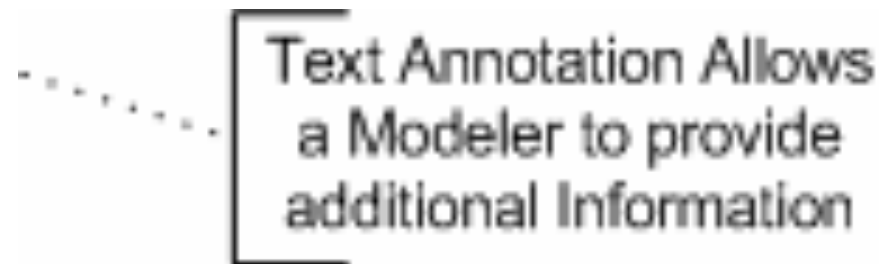
.....➤

..... used especially
for text annotation

Text annotation

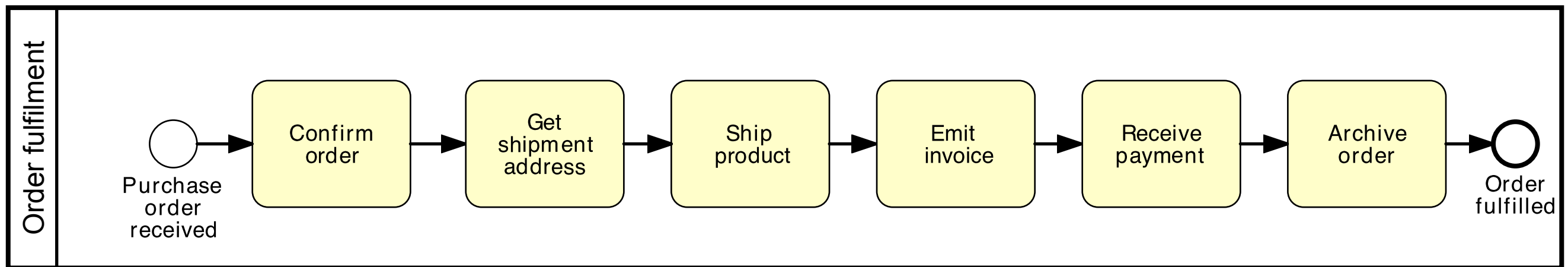
Any object can be associated with a **text annotation** to provide any additional information and documentation that can be needed

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

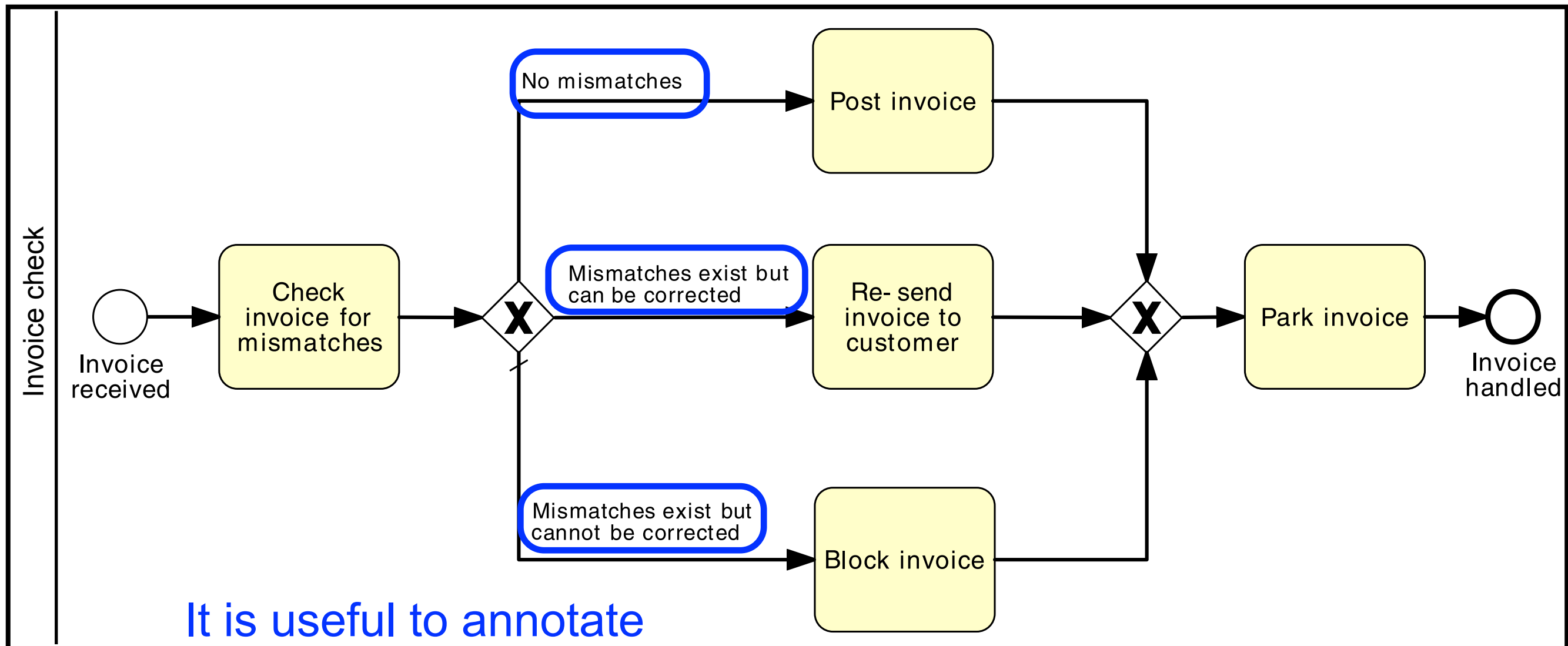


Typical patterns

Sequence: order fulfilment

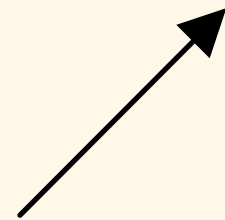


Exclusive decisions: invoice checking process

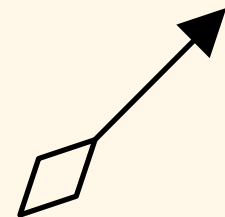


It is useful to annotate
branches with the conditions
under which they are taken

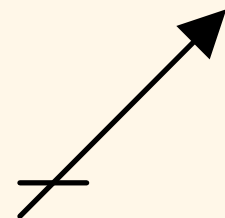
Annotated sequence flow



Sequence Flow defines the execution order of activities.



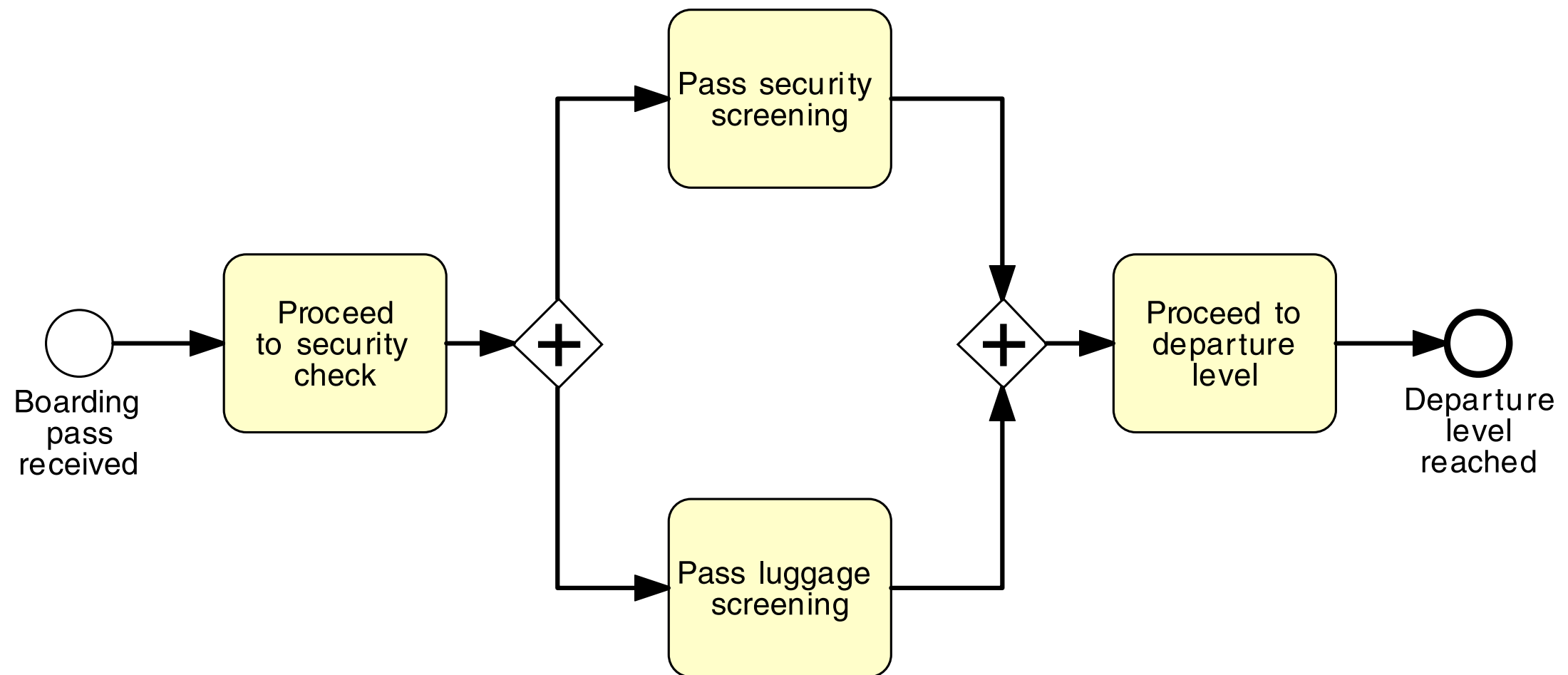
Conditional Flow has a condition assigned that defines whether or not the flow is used.



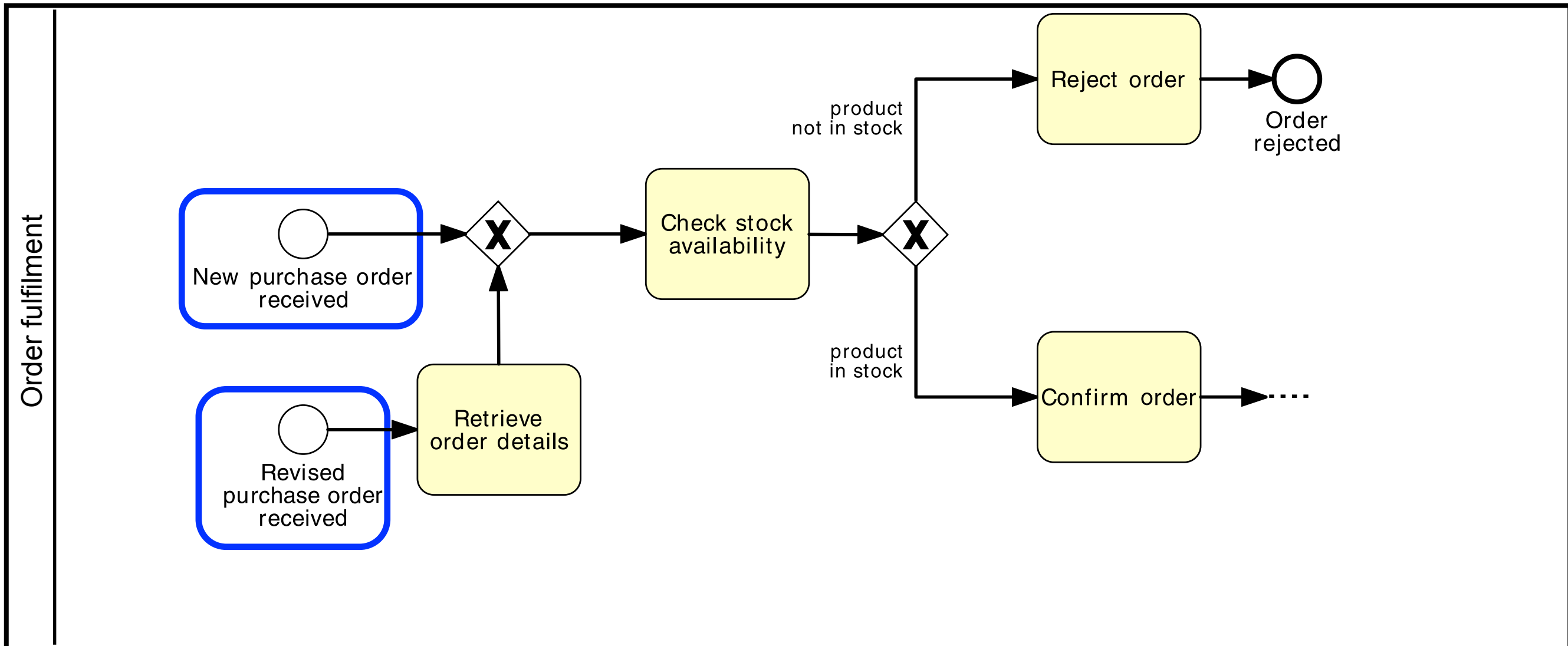
Default Flow is the default branch to be chosen if all other conditions evaluate to false.

read as
“otherwise”

Parallel activities: airport security check

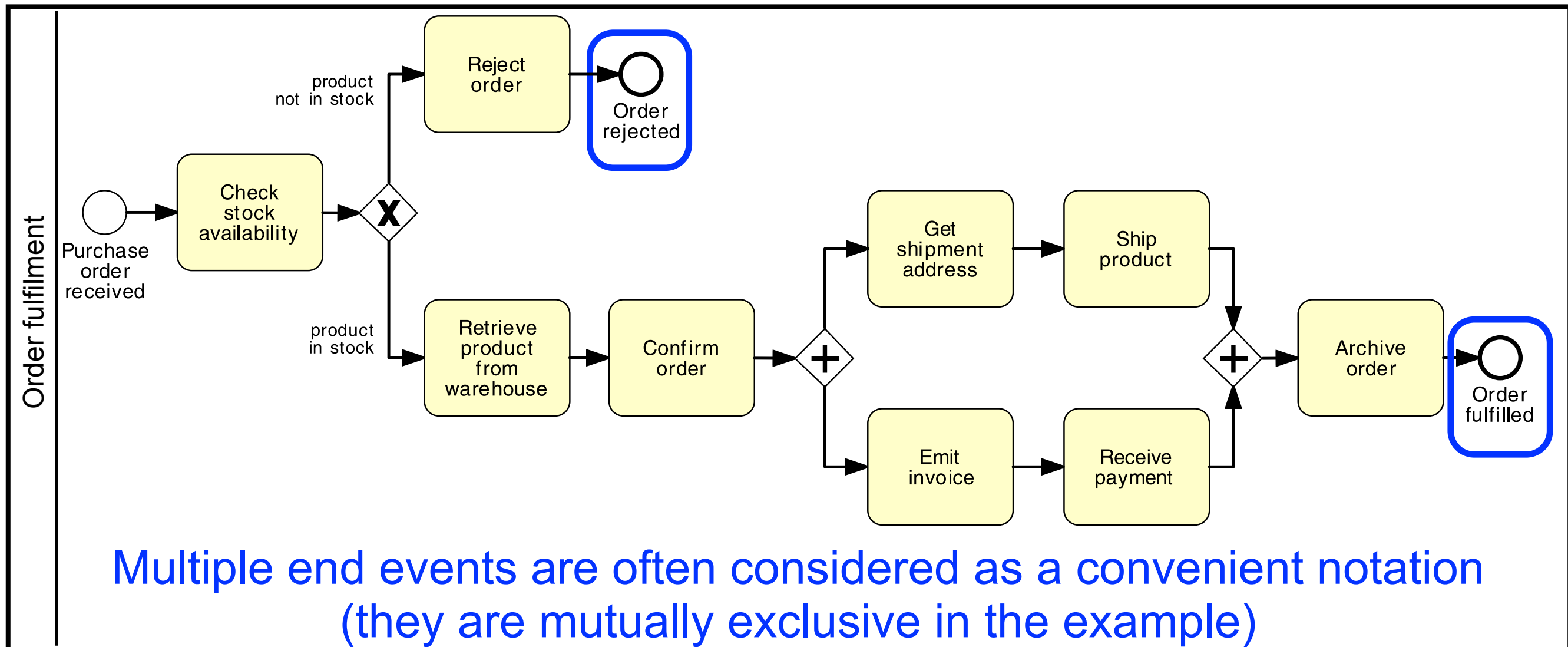


Multiple start events: order fulfilment



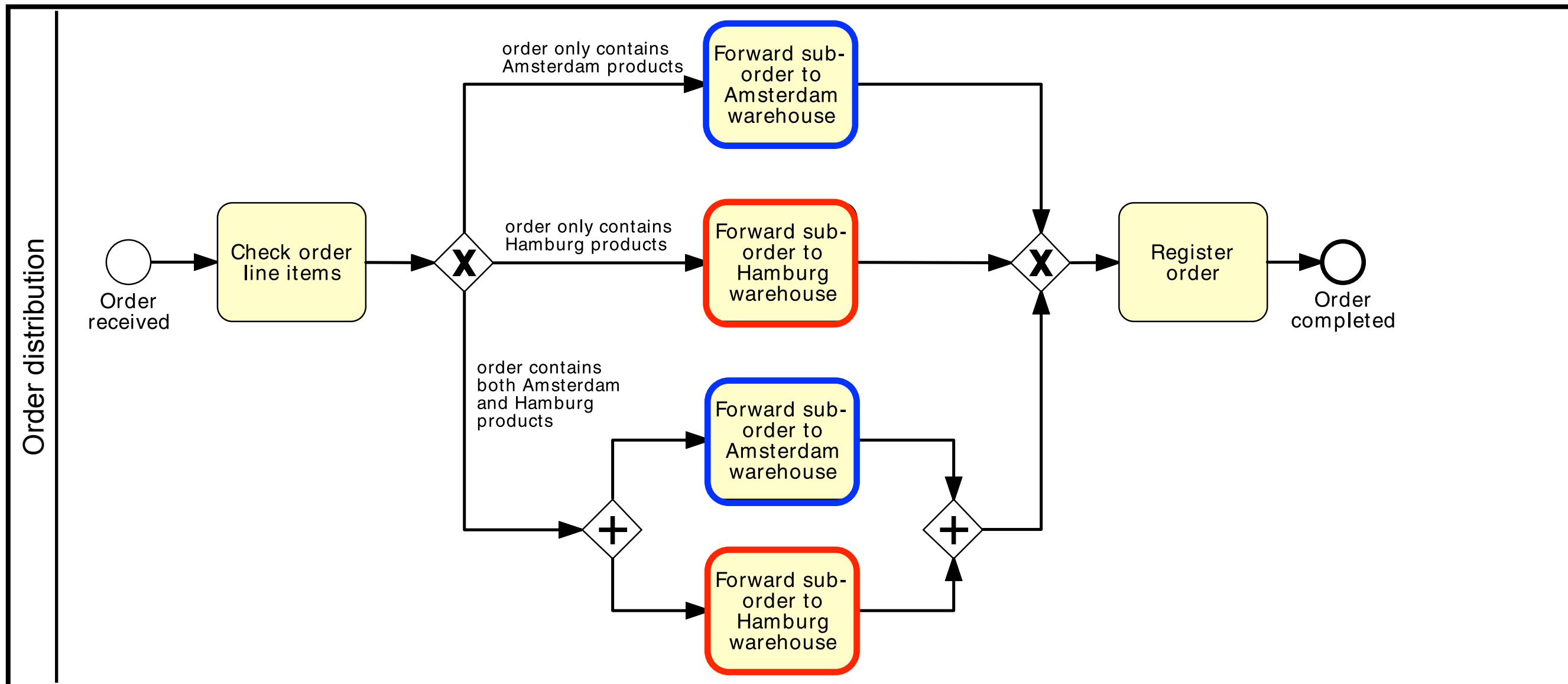
Multiple start events are often considered as a convenient notation (they capture mutually exclusive triggers to start a process instance)

Multiple end events: order fulfilment



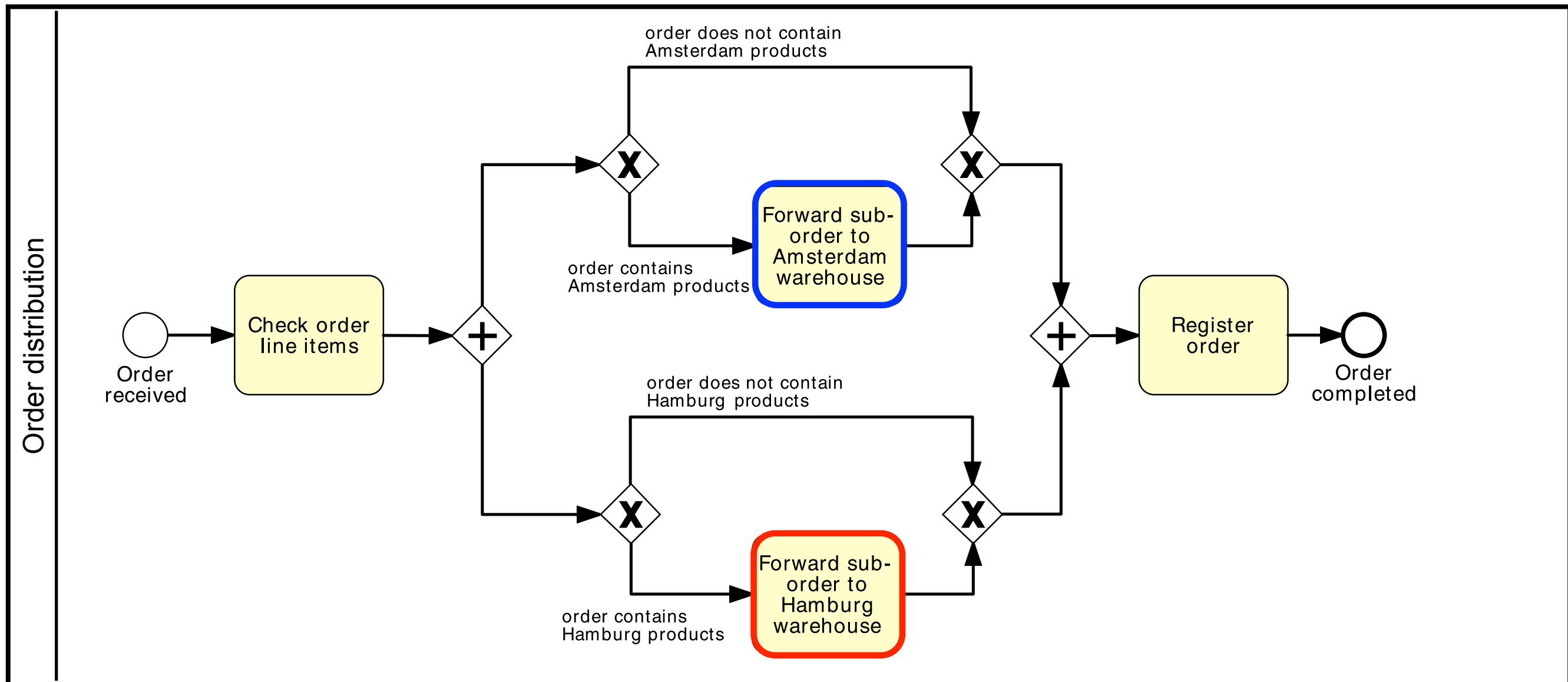
BPMN adopts **implicit termination** semantics:
a case ends only when each ``token'' reaches the end

Inclusive decisions: order distribution



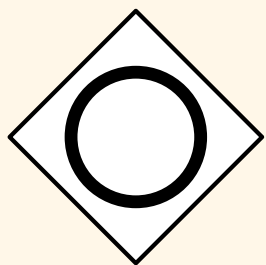
Only XOR / AND gateways, but the diagram is convoluted!
What if we had three or more warehouses? (does not scale)

Inclusive decisions: order distribution



Only XOR / AND gateways, the diagram can ``scale'',
but is it correct? (also the case no-warehouse is now possible)

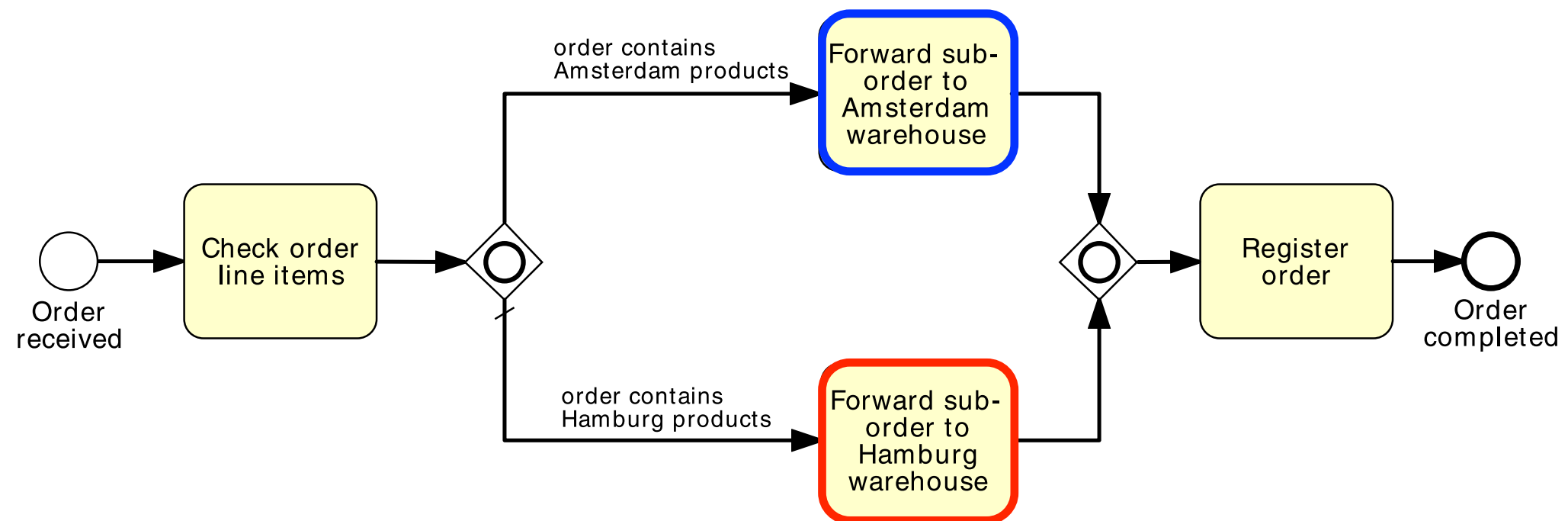
Inclusive decisions (one, many)



Inclusive Gateway

When splitting, one or more branches are activated based on branching conditions. When merging, it awaits all active incoming branches to complete.

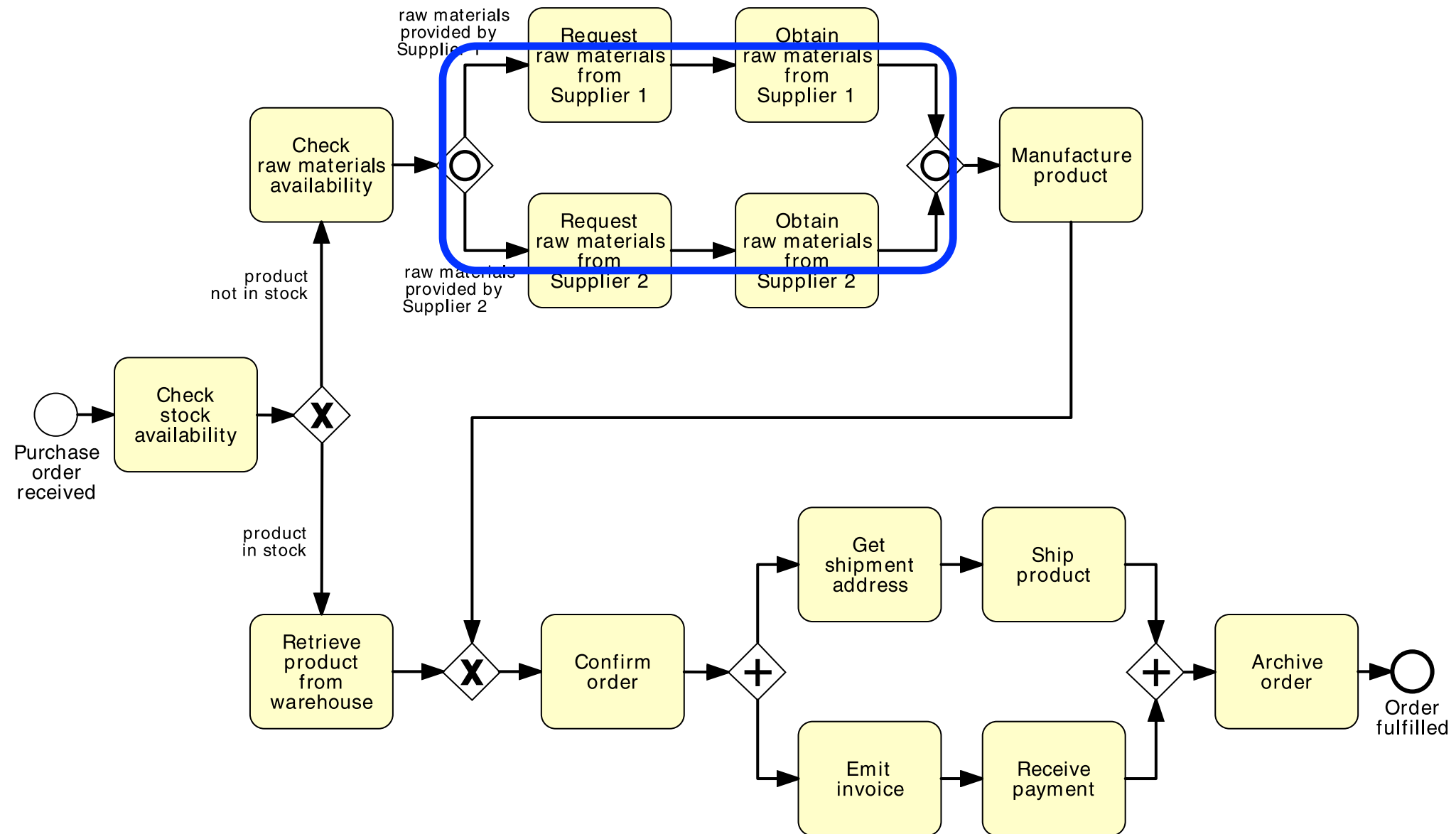
Inclusive decisions: order distribution



using OR gateways, the diagram can ``scale'',
but all the **issues with unmatched OR-joins in EPC** are still valid!

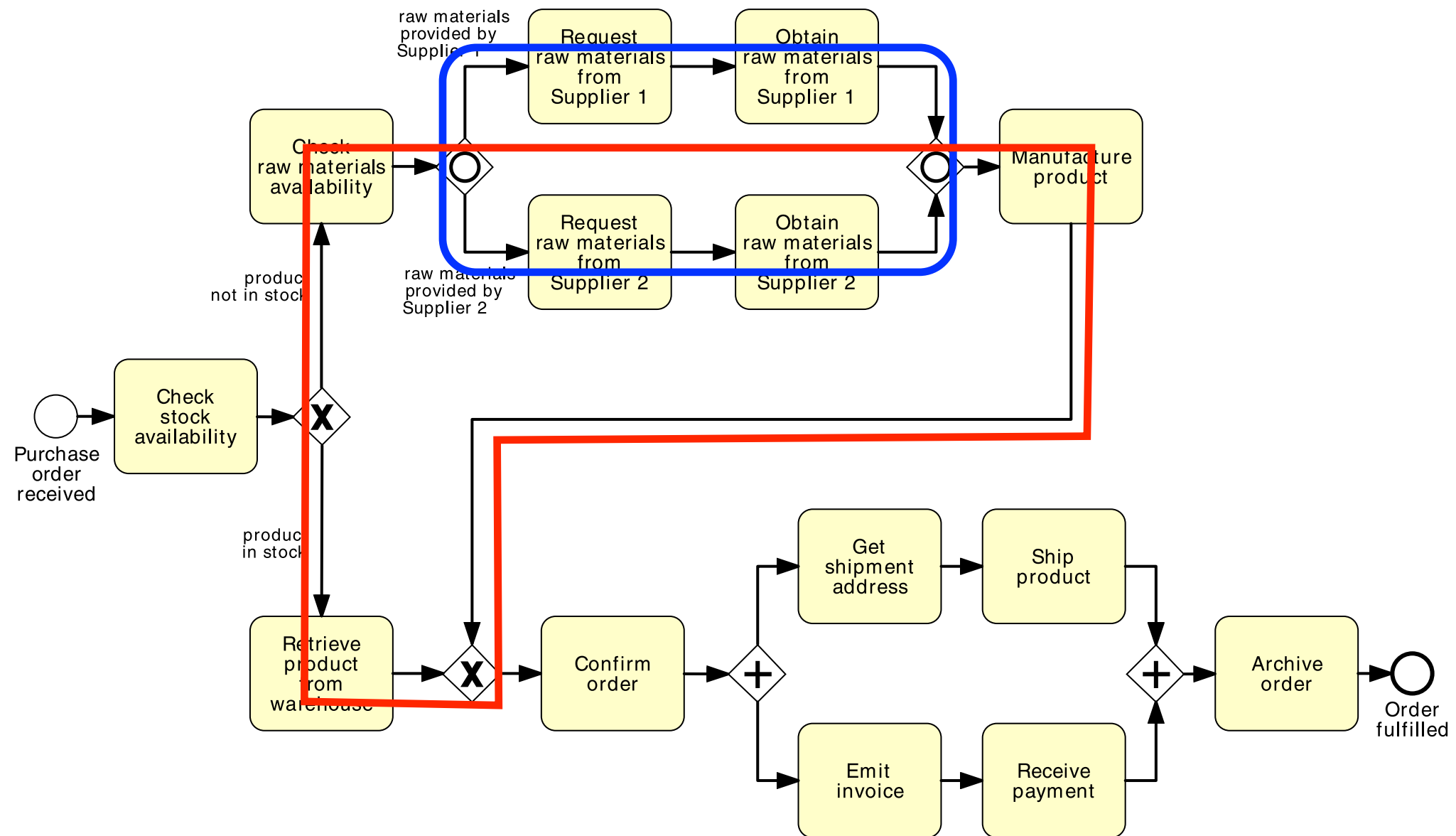
Use OR-gateways only when strictly necessary

XOR + AND + OR: order fulfilment



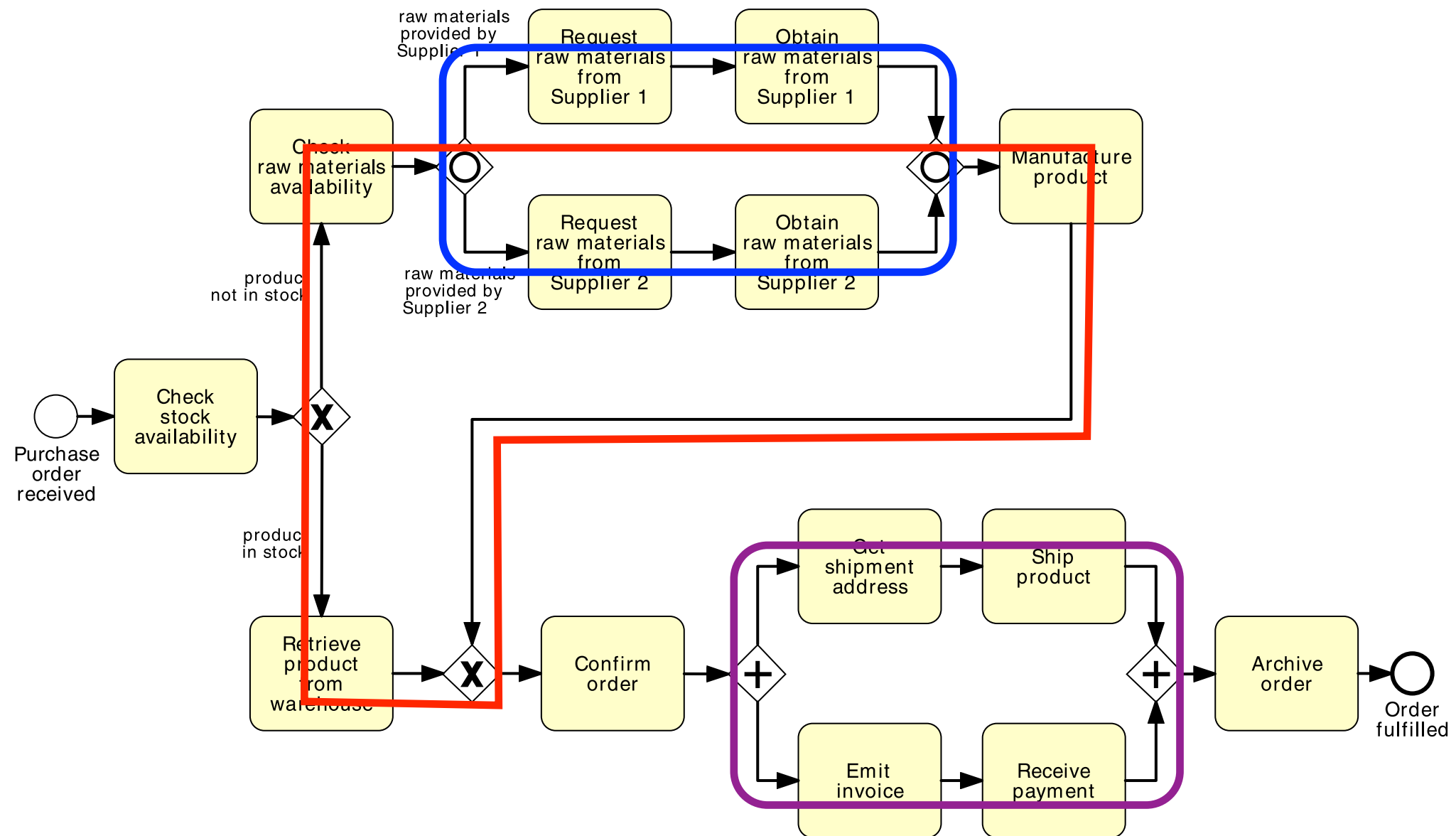
Better if gateways are balanced

XOR + AND + OR: order fulfillment



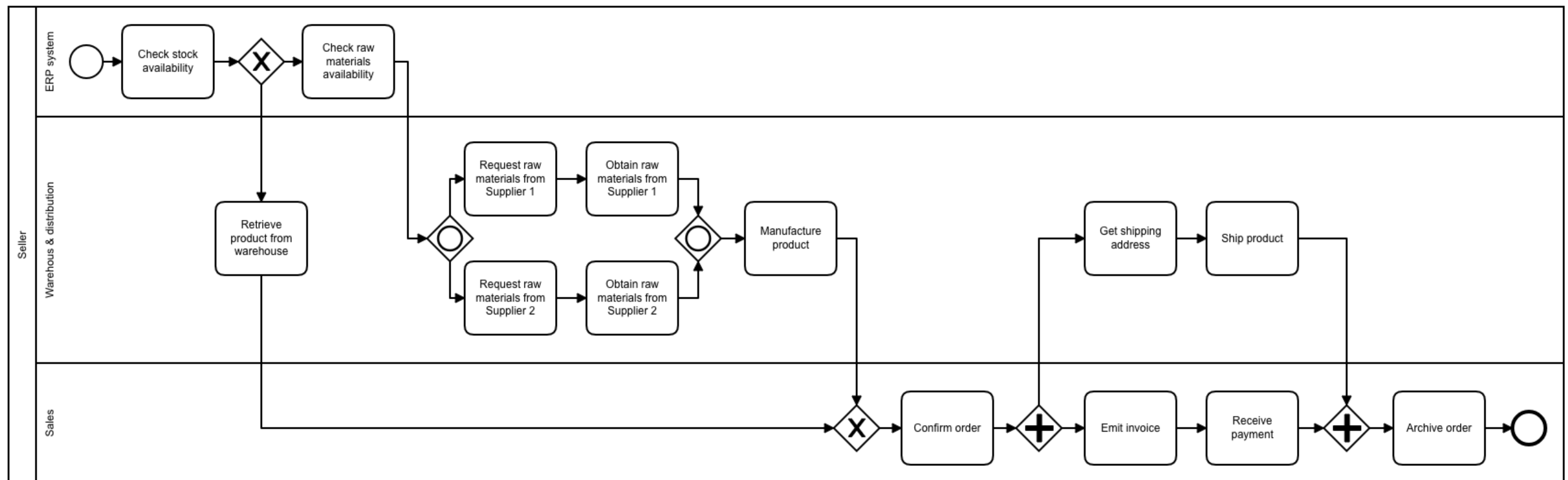
Better if gateways are balanced

XOR + AND + OR: order fulfillment

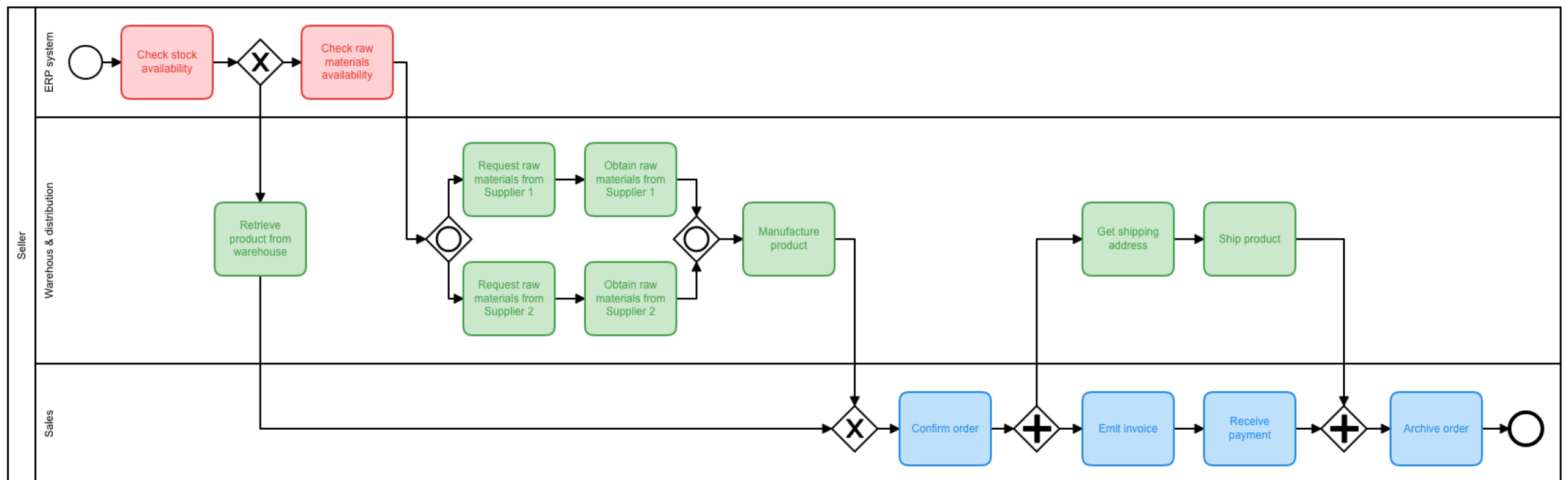


Better if gateways are balanced

Resources as lanes: order fulfillment



Resources as lanes: order fulfillment



Placing items in lanes

events: must be placed in the proper lane

activities: must be placed in the proper lane

gateways:

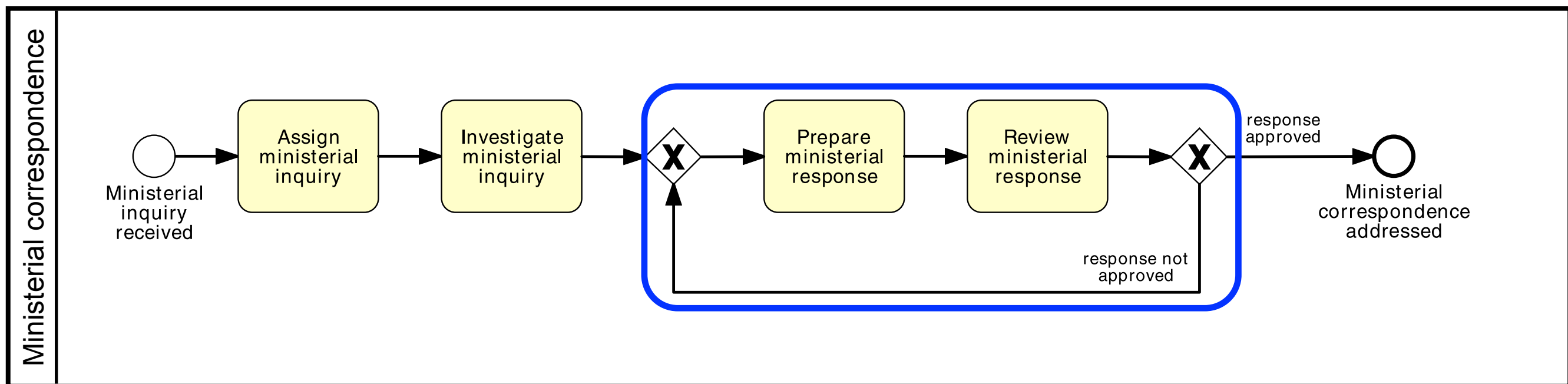
(X)OR-splits: same lane as preceding decision activity

AND-split: placement is irrelevant

(any kind of) **join:** placement is irrelevant

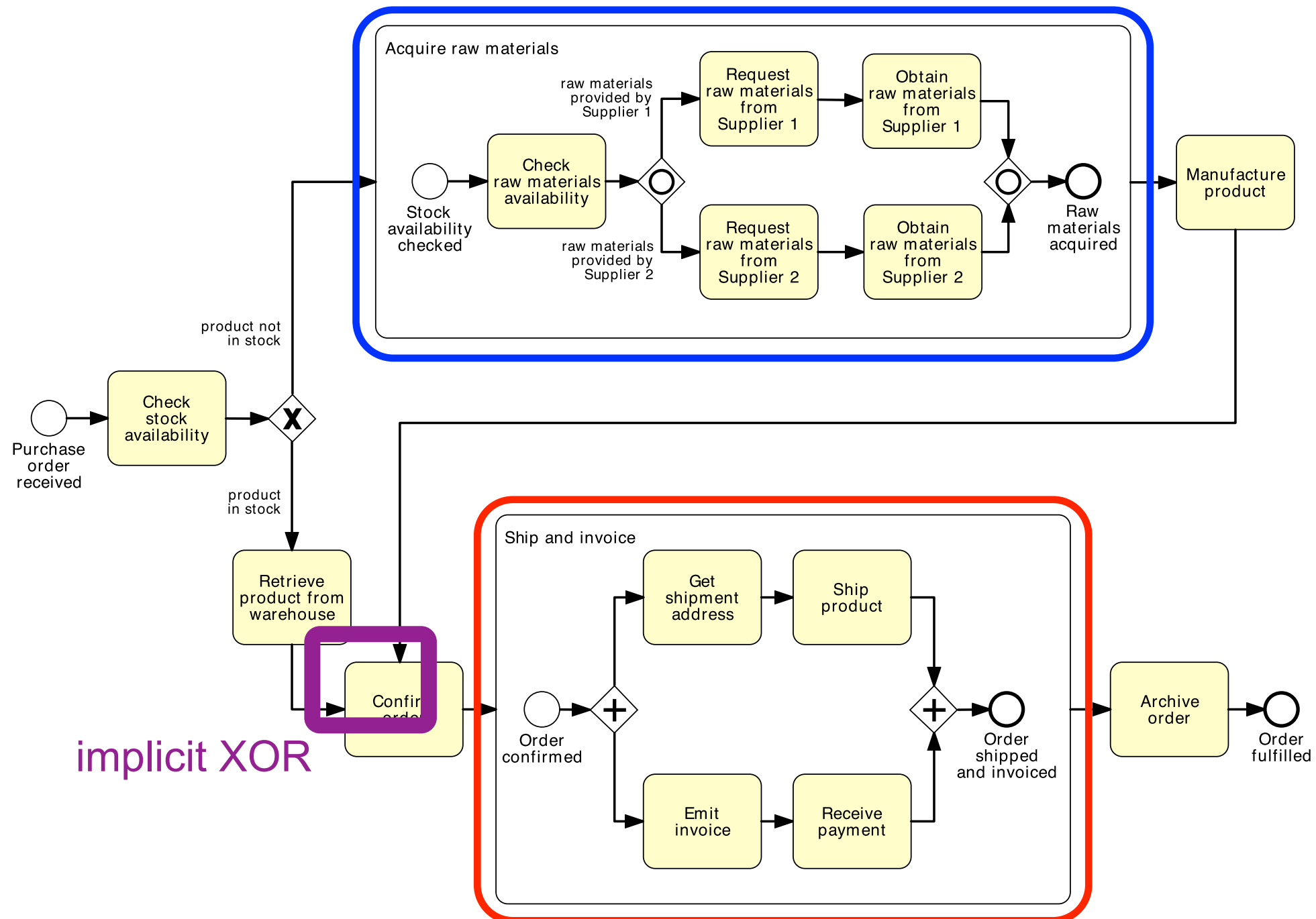
data-objects: placement is irrelevant

Rework and repetition: ministerial correspondence

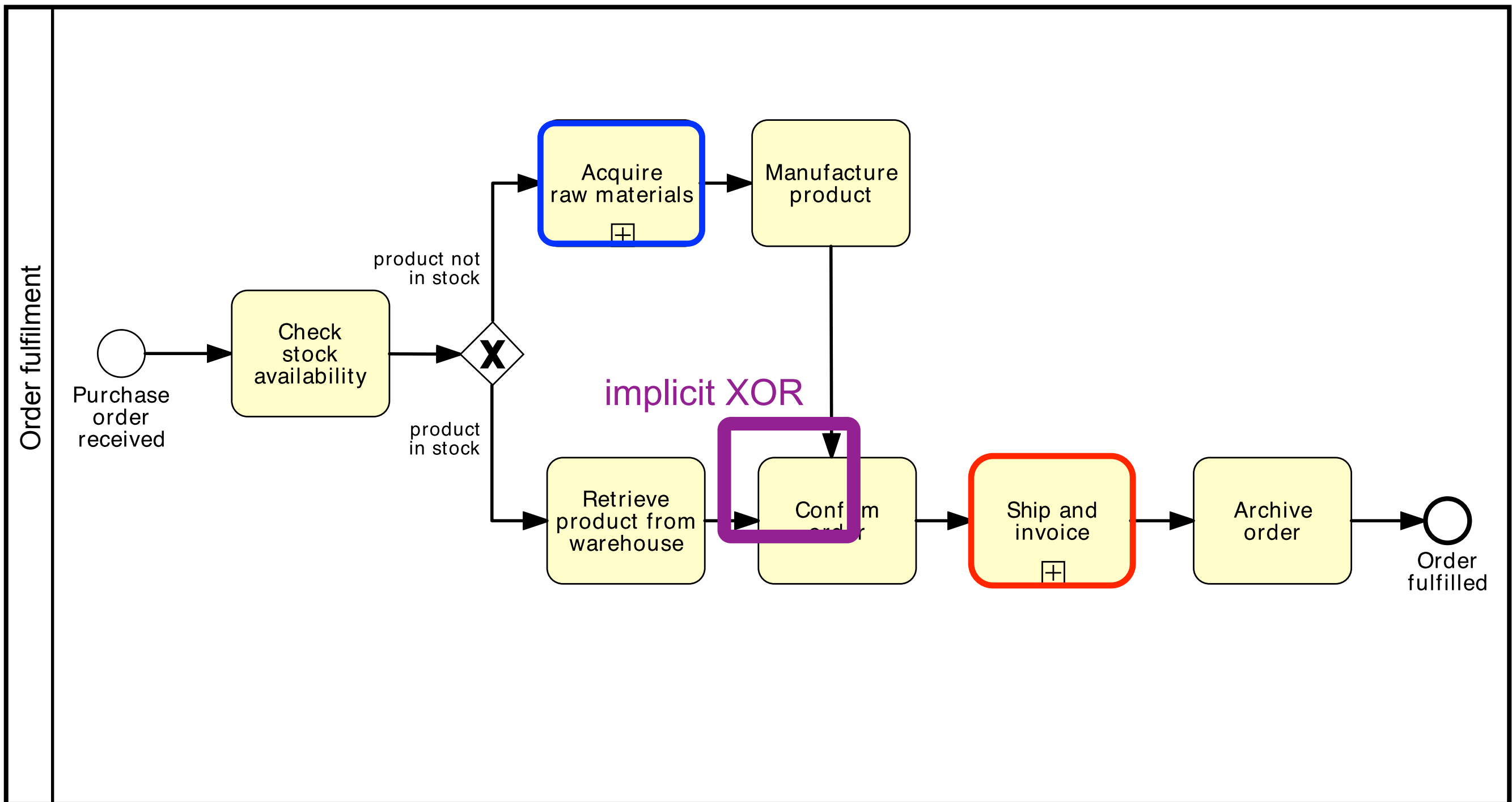


A repetition block starts with a XOR-join
and ends with a decision gateway (XOR-split)

Identify sub-processes

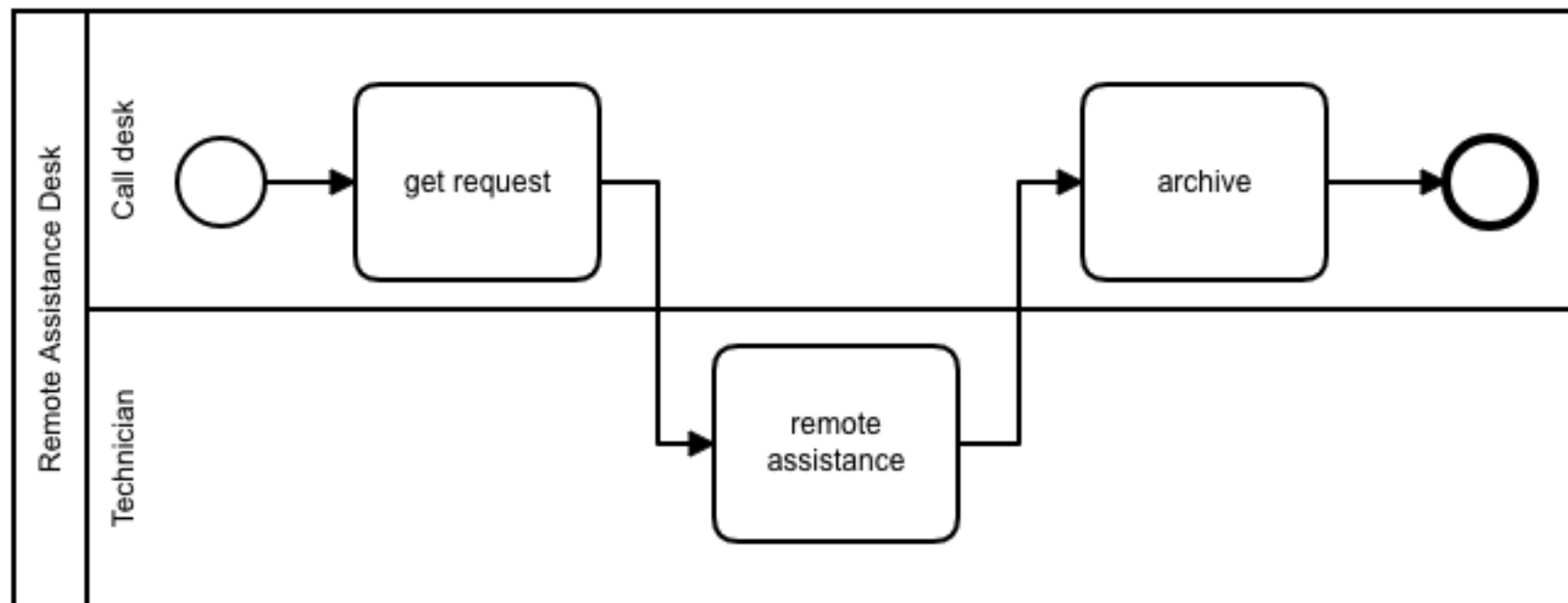


Collapsed sub-processes



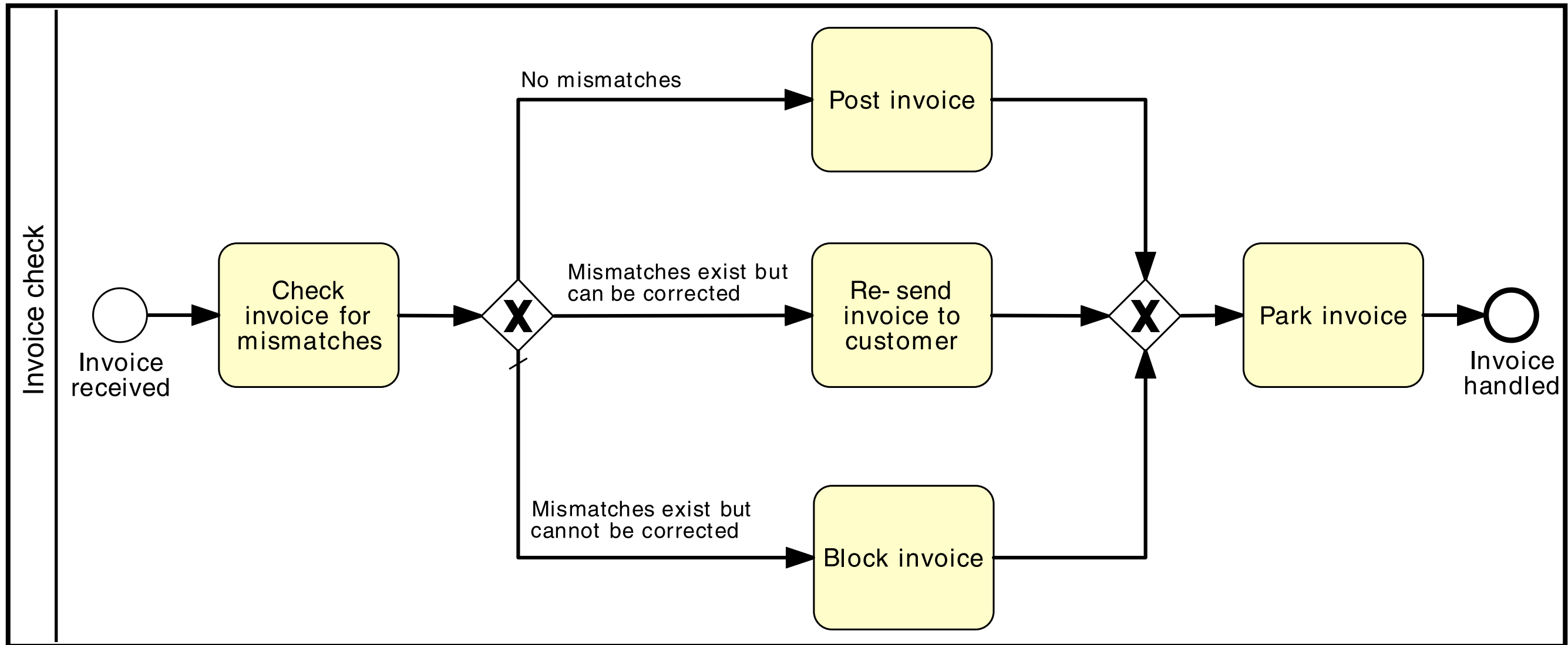
Exercise: basics

Draw the EPC diagram that corresponds to the BPMN diagram below



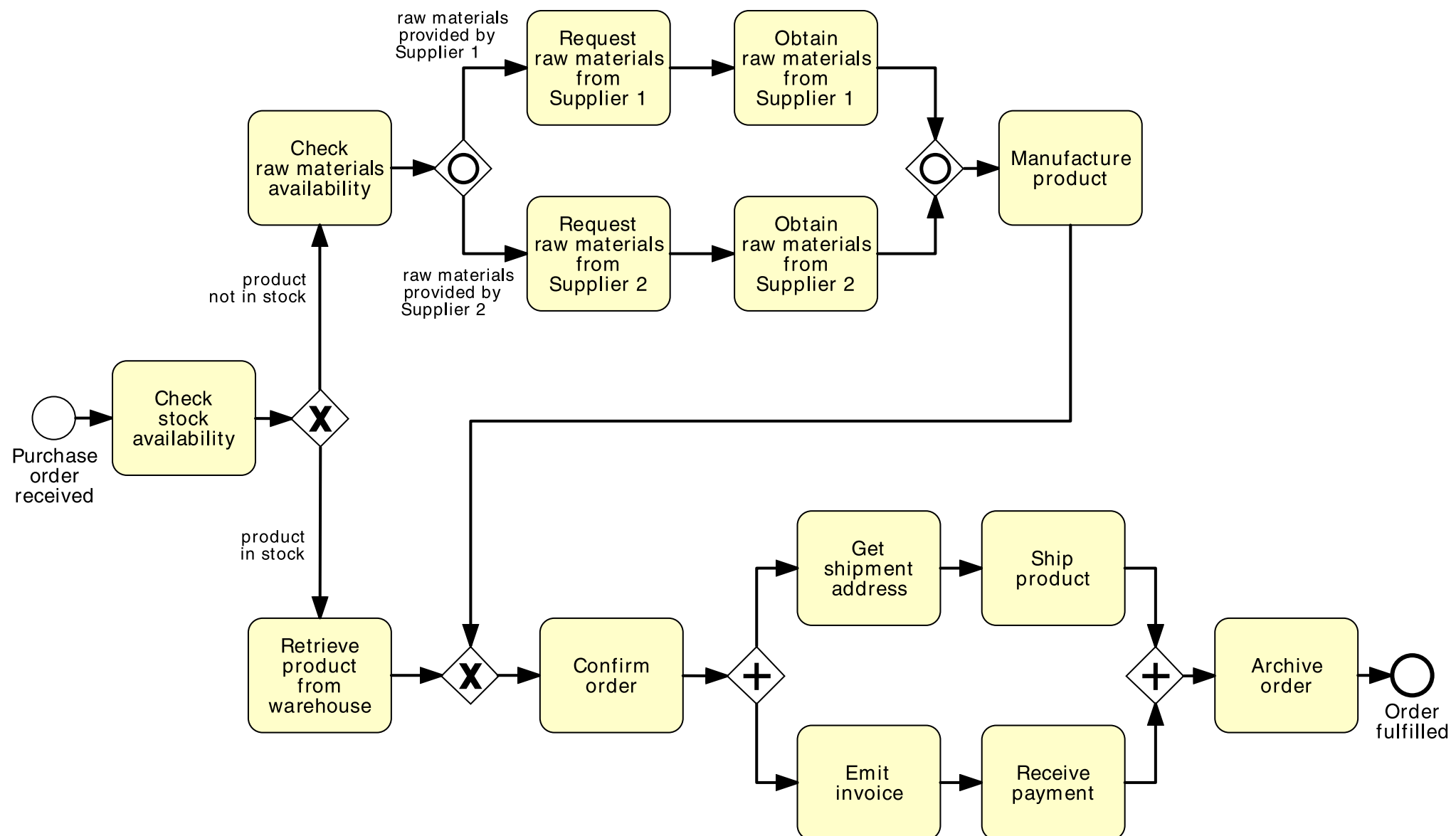
Exercise: basics

Draw the EPC diagram that corresponds to the BPMN diagram below



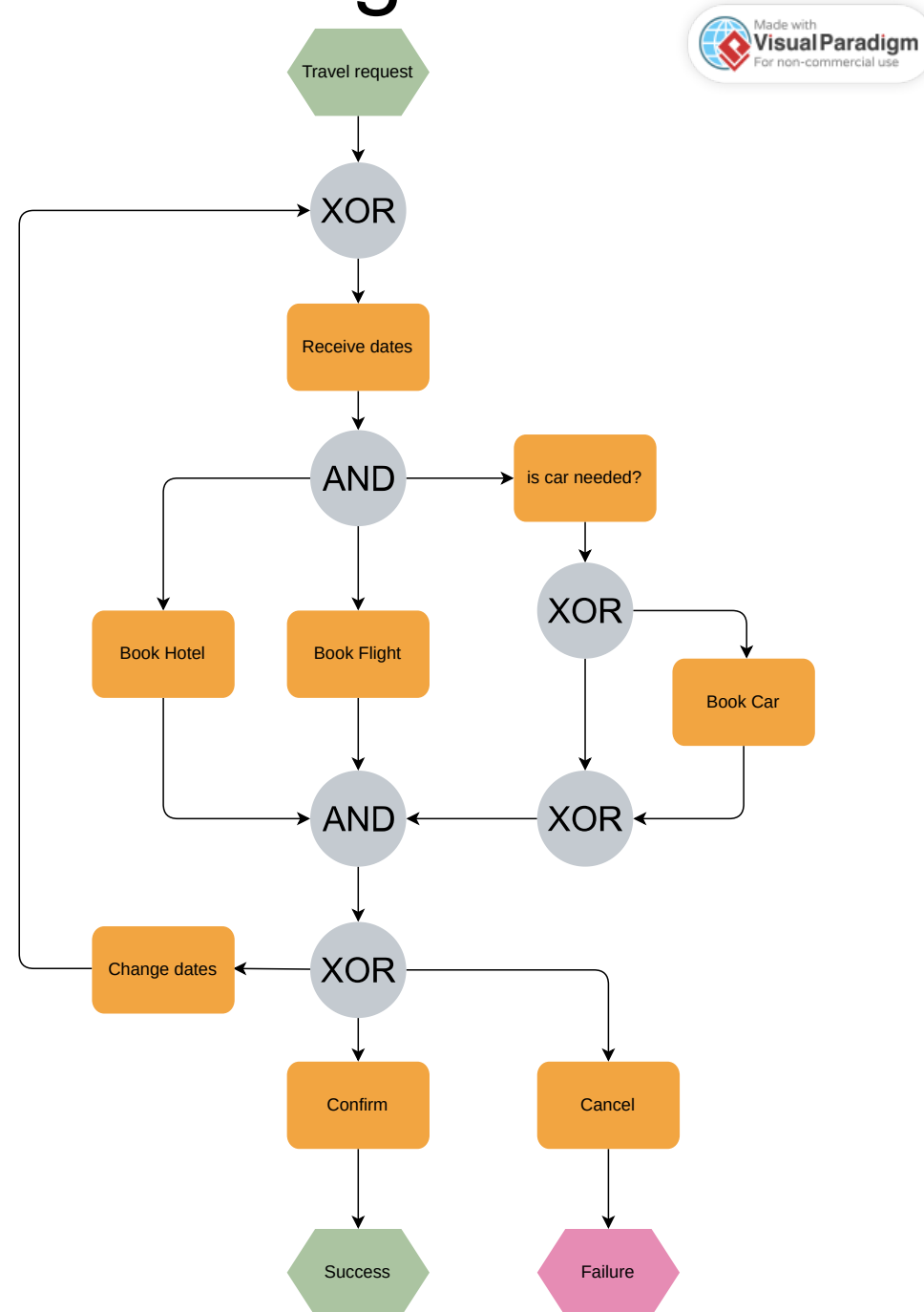
Exercise: basics

Draw the EPC diagram that corresponds to the BPMN diagram below



Exercise: basics

Draw the BPMN diagram that corresponds to the EPC diagram below



BPMN basics

BPMN key features








More on BPMN

BPMN semantics







Markers
(events, activities, gateways)

Activity types and markers

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

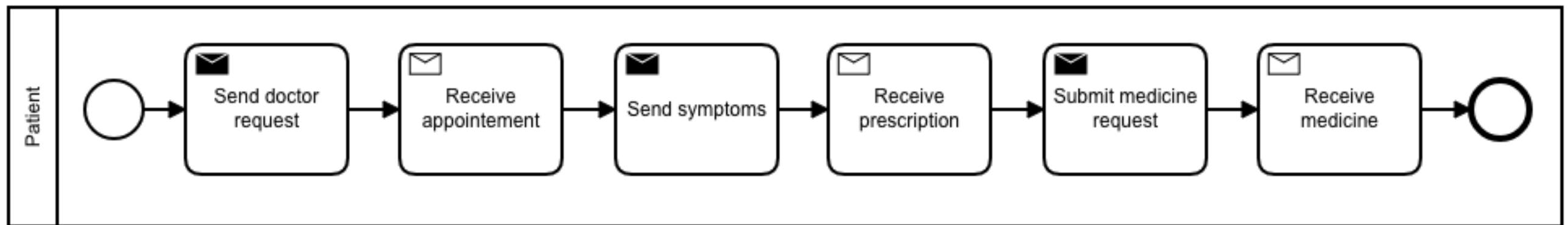
-  Send Task
-  Receive Task
-  User Task
-  Manual Task
-  Business Rule Task
-  Service Task
-  Script Task

some types

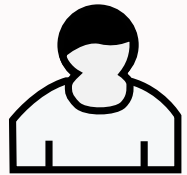
-  Sub-Process Marker
-  Loop Marker
-  Parallel MI Marker
-  Sequential MI Marker
-  Ad Hoc Marker
-  Compensation Marker

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.

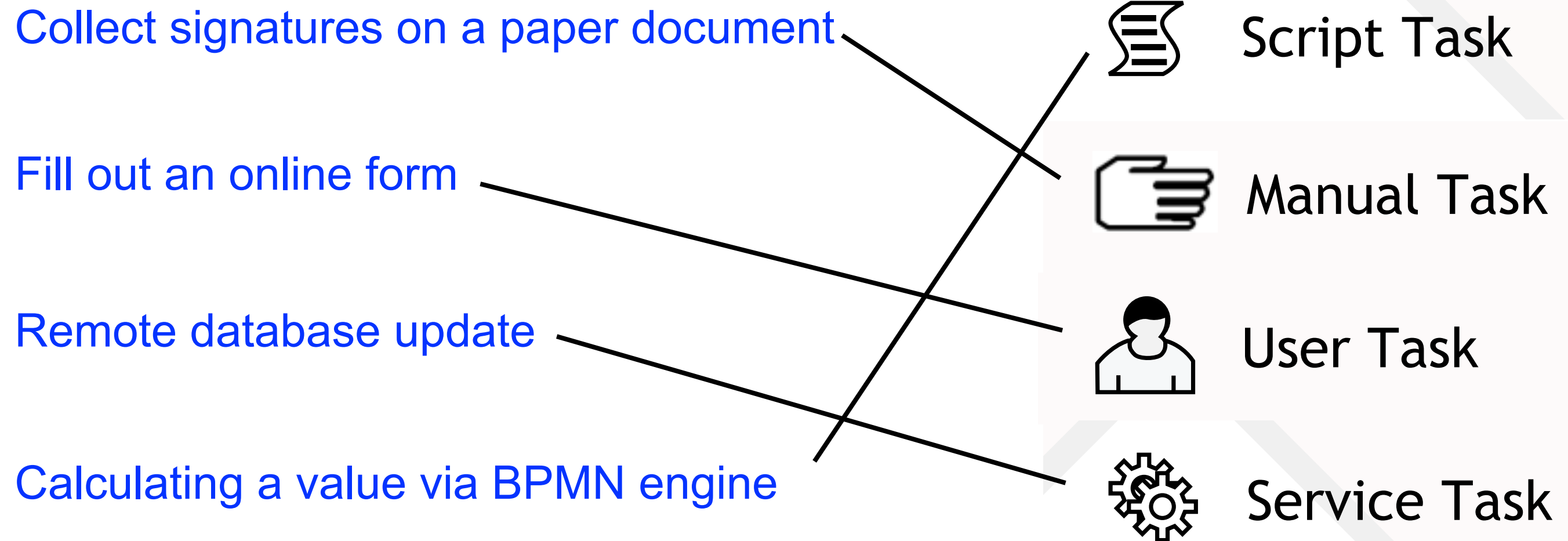


Script Task

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.

Question time: which type?



Some activity markers

Multiple
Instances



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

Multiple
Instances

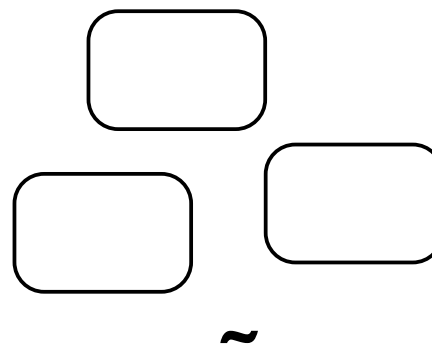


Loop



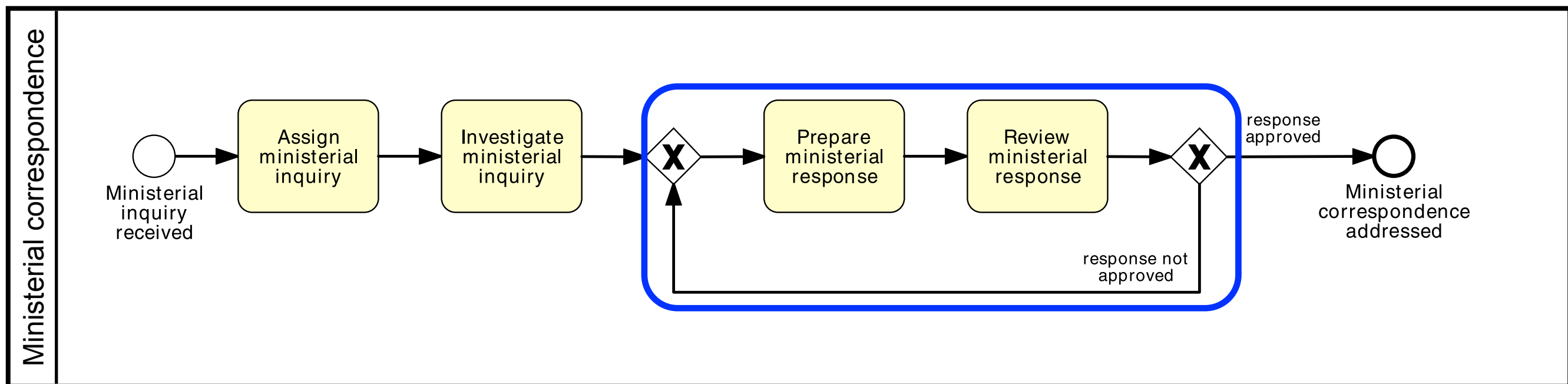
Loop Activity is iterated if a loop condition is true. The condition is either tested before or after the activity execution.

Ad-hoc Subprocess

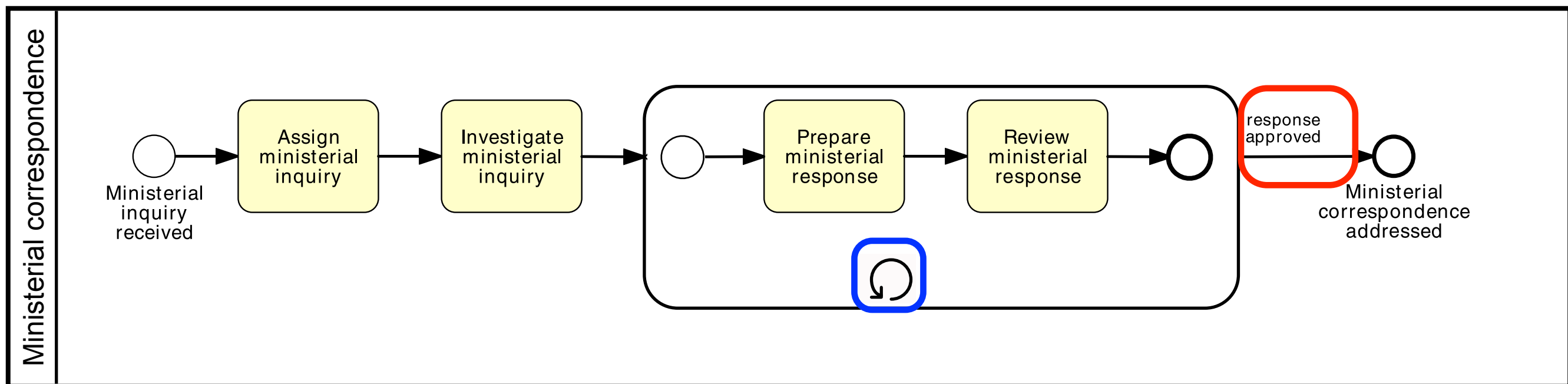


Ad-hoc Subprocesses contain tasks only. Each task can be executed arbitrarily often until a completion condition is fulfilled.

Loop marker: ministerial correspondence



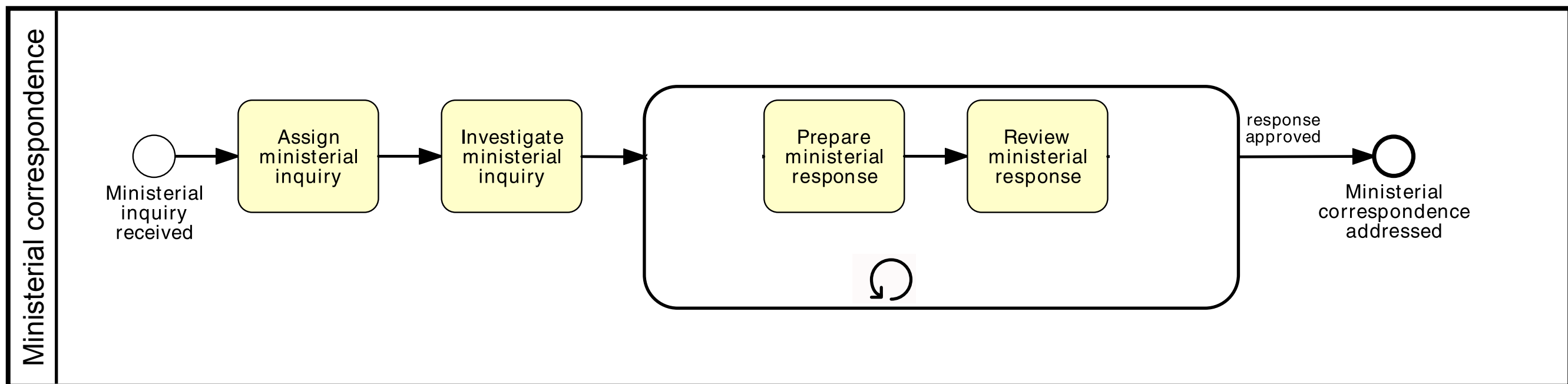
Loop marker: ministerial correspondence



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

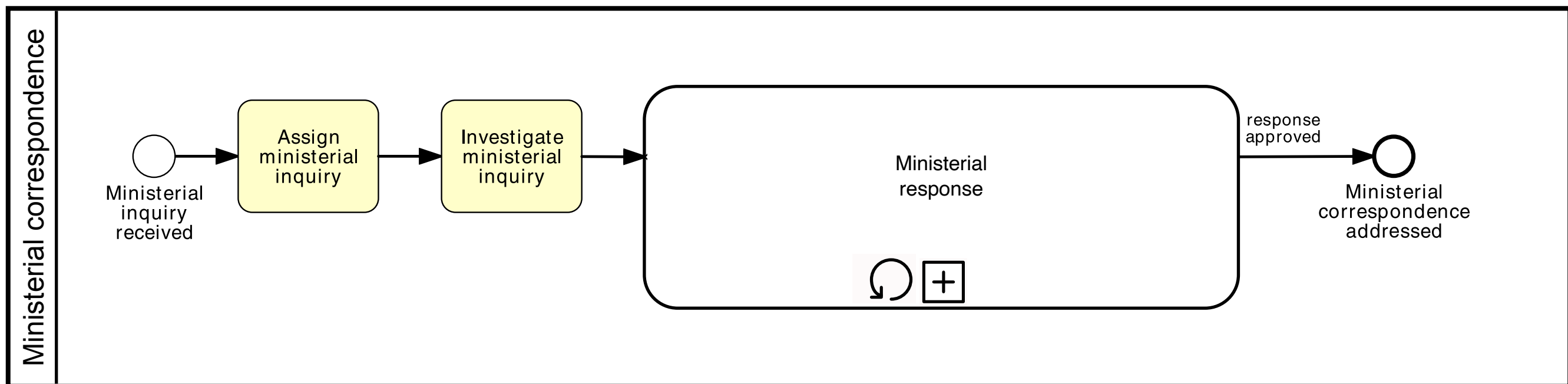
it is important to define
exit conditions from loops!

Loop marker: ministerial correspondence



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

Loop marker: ministerial correspondence



we can hide internal details

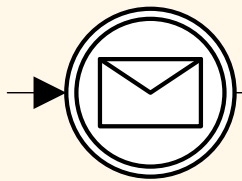
Catching and throwing

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

Catching

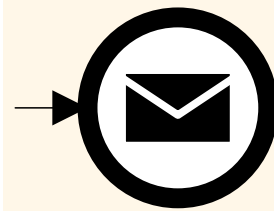


Start Event: Catching an event starts a new process instance.

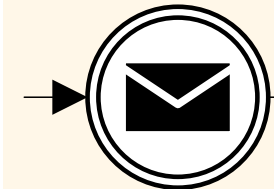


Intermediate Event (catching):
The process can only continue once an event has been caught.

Throwing

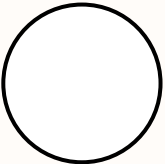
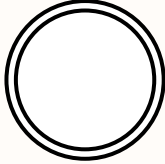
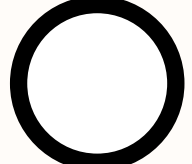



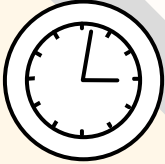



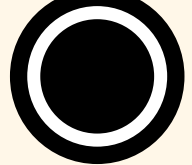


End Event: An event is thrown when the end of the process is reached.

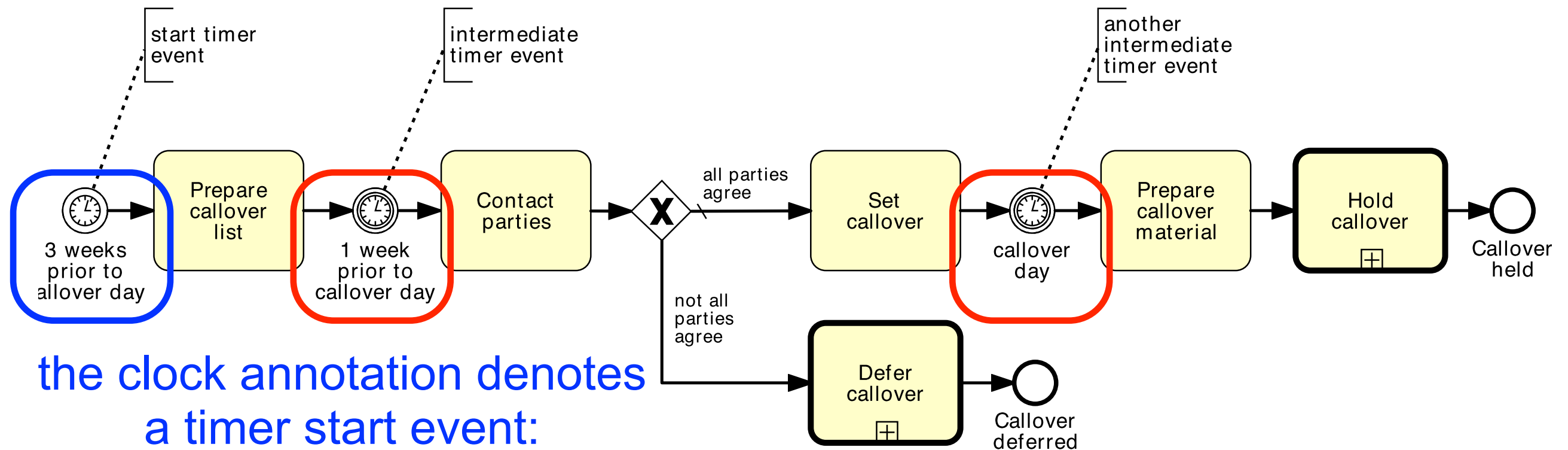


Intermediate Event (throwing):
An event is thrown and the process continues.

Some internal markers

	Start	Intermediate	End	
	<i>Catching</i>		<i>Throwing</i>	
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.

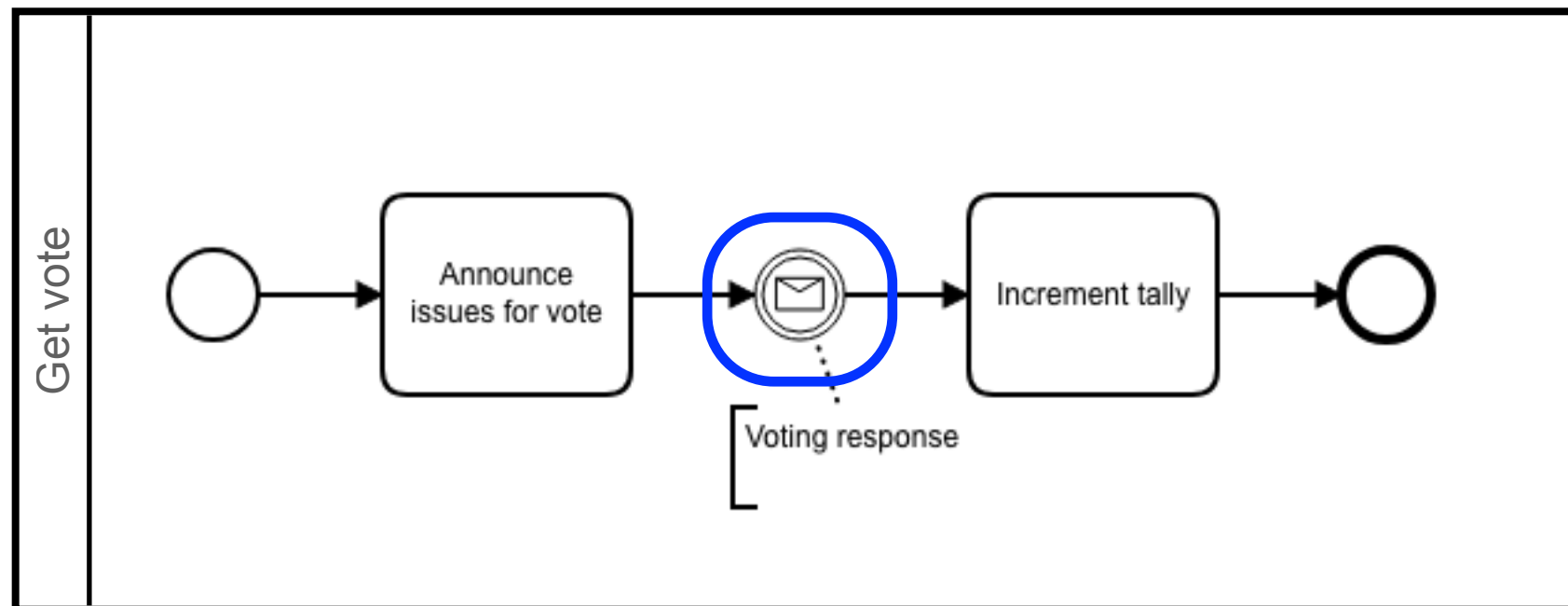
Timer events: small claims tribunal



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

Process break (event waiting)

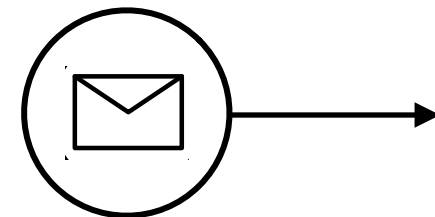


the envelope annotation denotes an intermediate message event:
it signals the receipt of a message

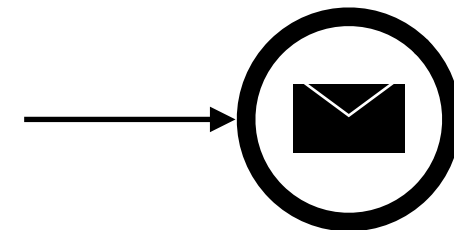
Collaboration diagrams (and message passing)

Message annotated events and activities

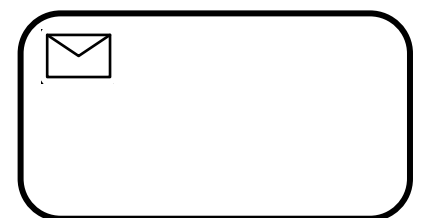
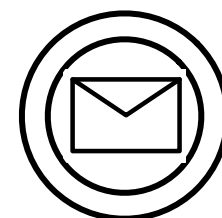
A start event can be annotated with a white-envelope:
a process instance is created
when a certain message is received



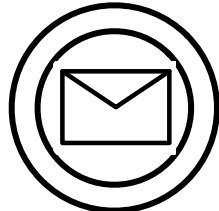
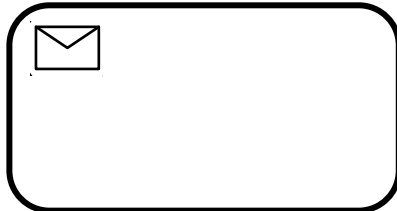
An end event can be annotated with a black-filled envelope:
when the process ends
a message is sent



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

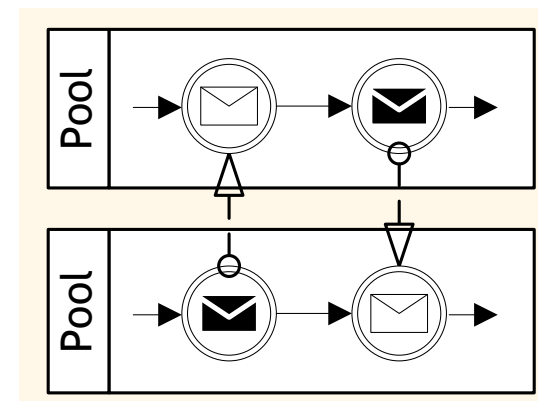
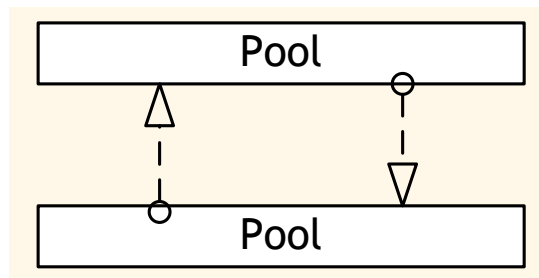
activities take time (have a duration)

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

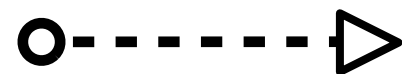
Each possible communication corresponds to a **message flow** between pools (or between objects from different pools)



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

To:							
From:							

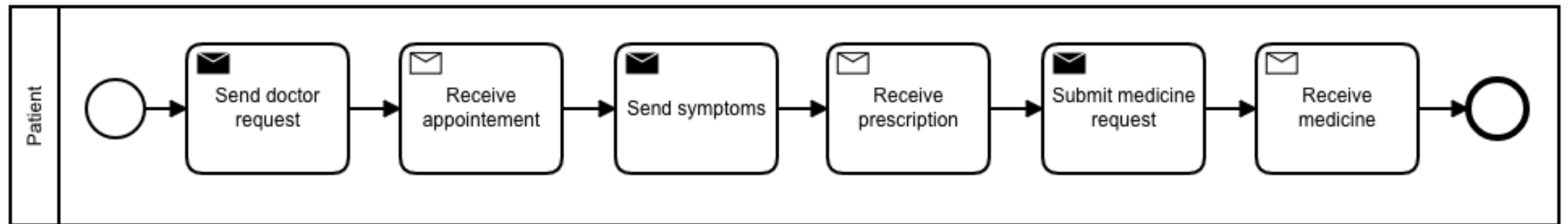
each event:
at most one message flow

each activity:
at most one message flow

each gateway:
no message flow!

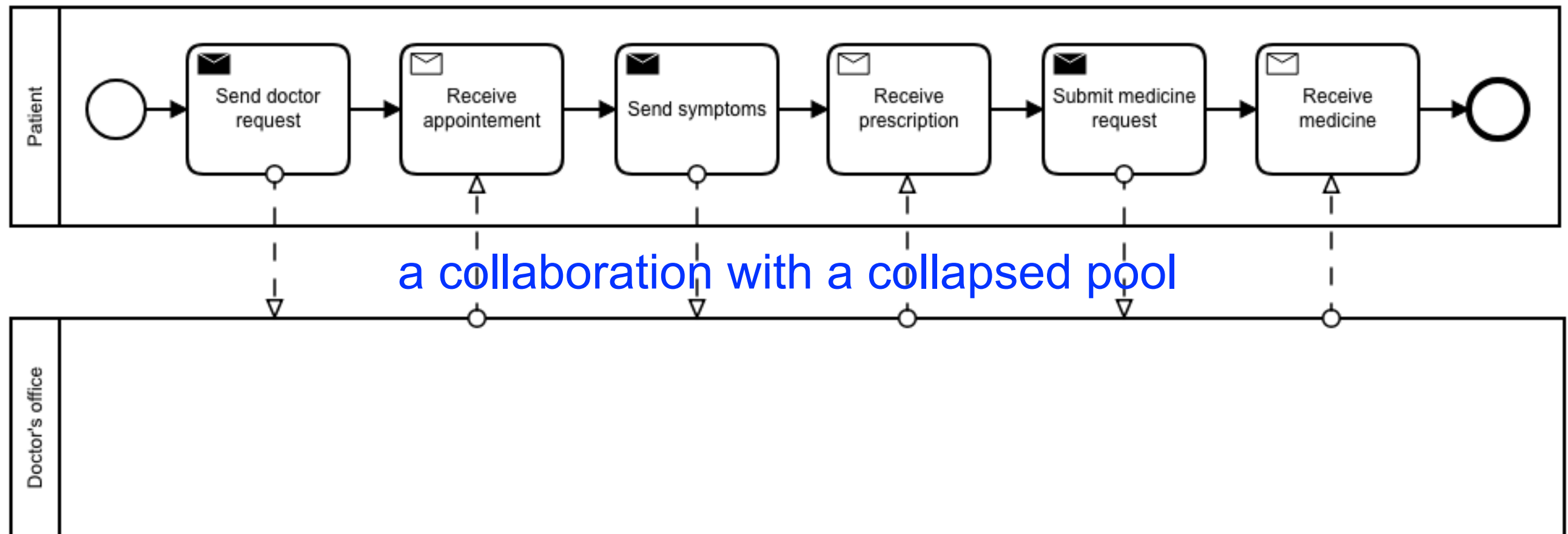
each pool:
any number of message flows

From processes to collaborations

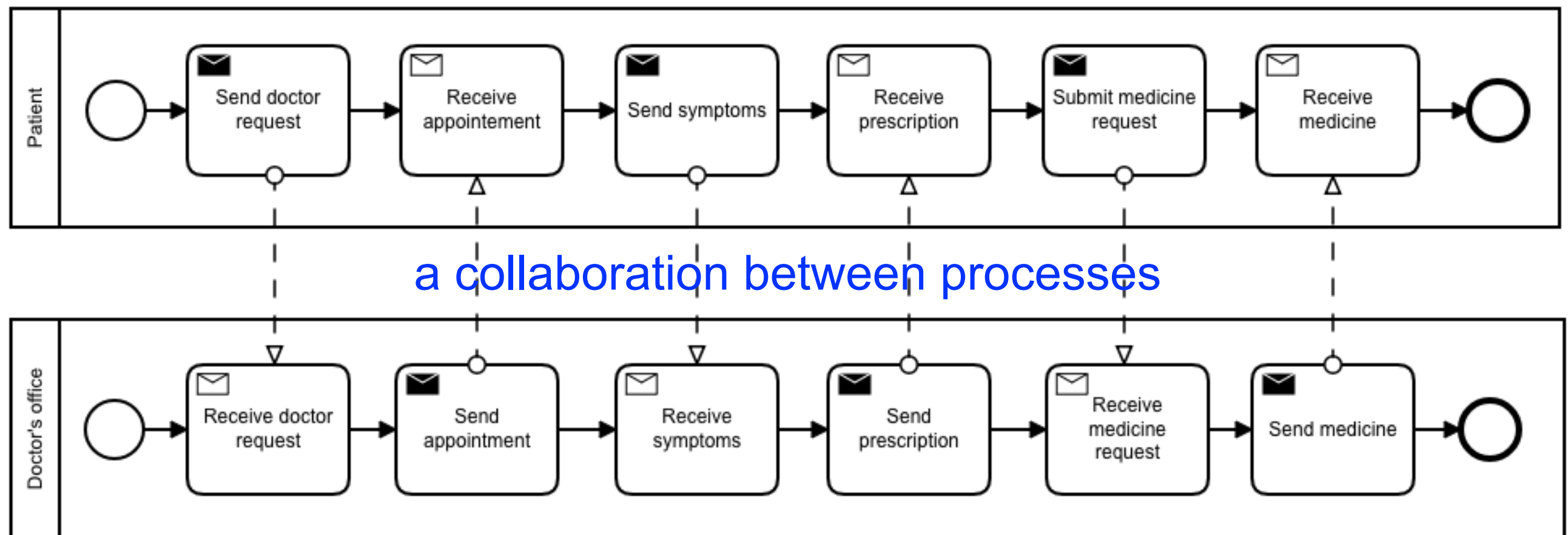


a stand-alone process

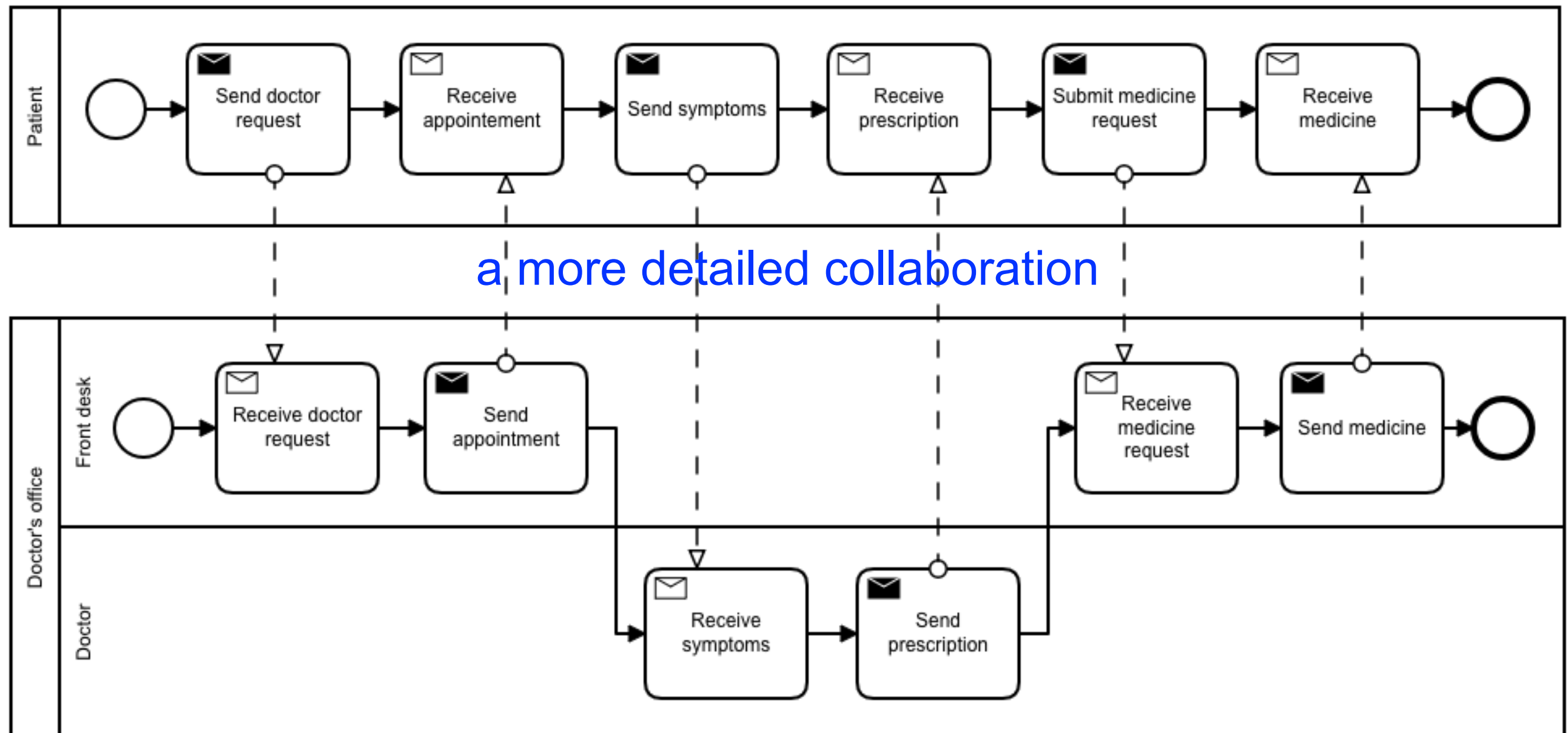
From processes to collaborations



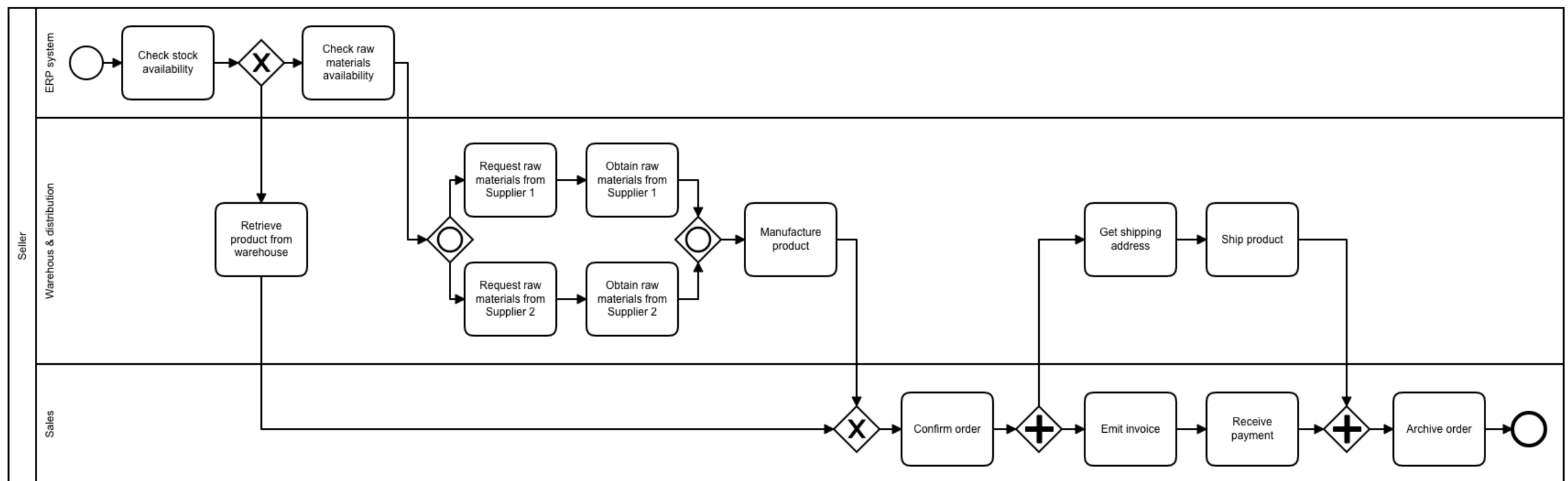
From processes to collaborations



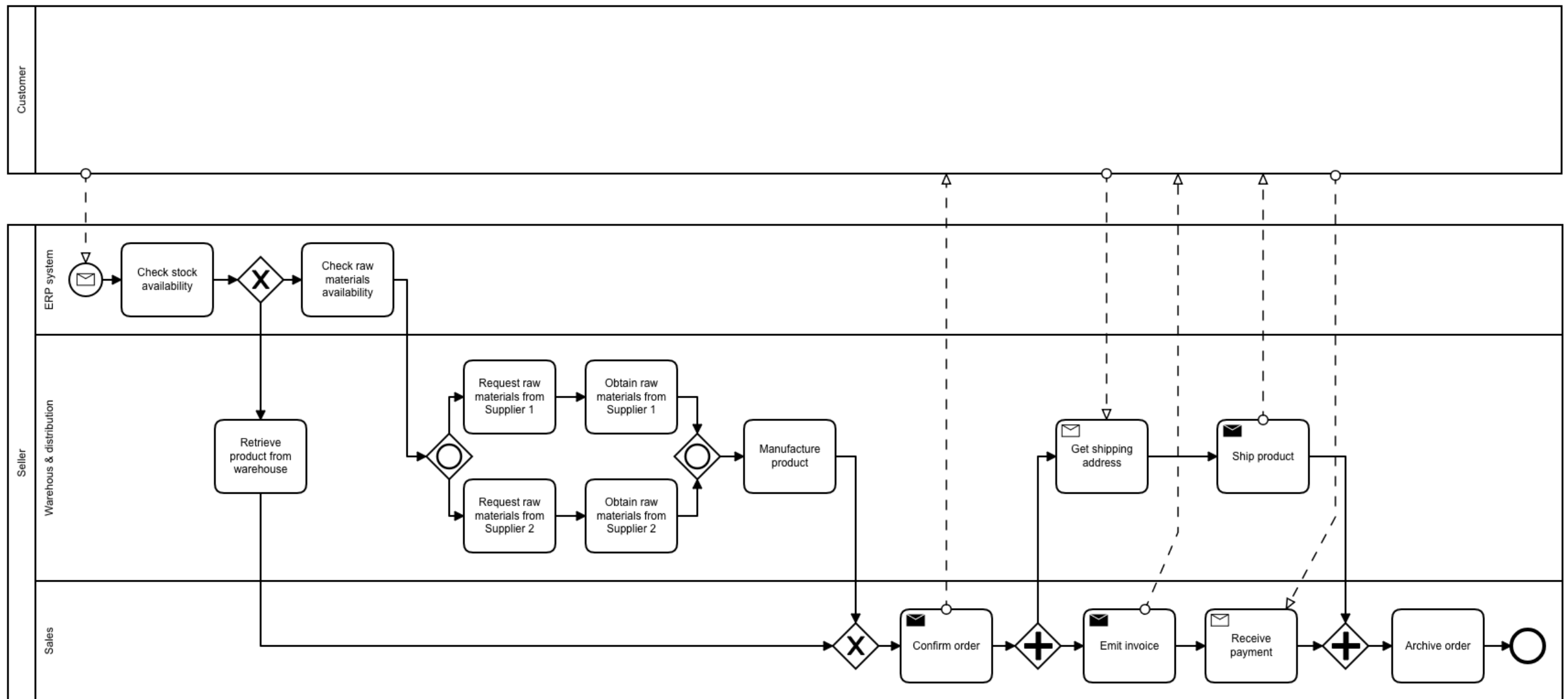
From processes to collaborations



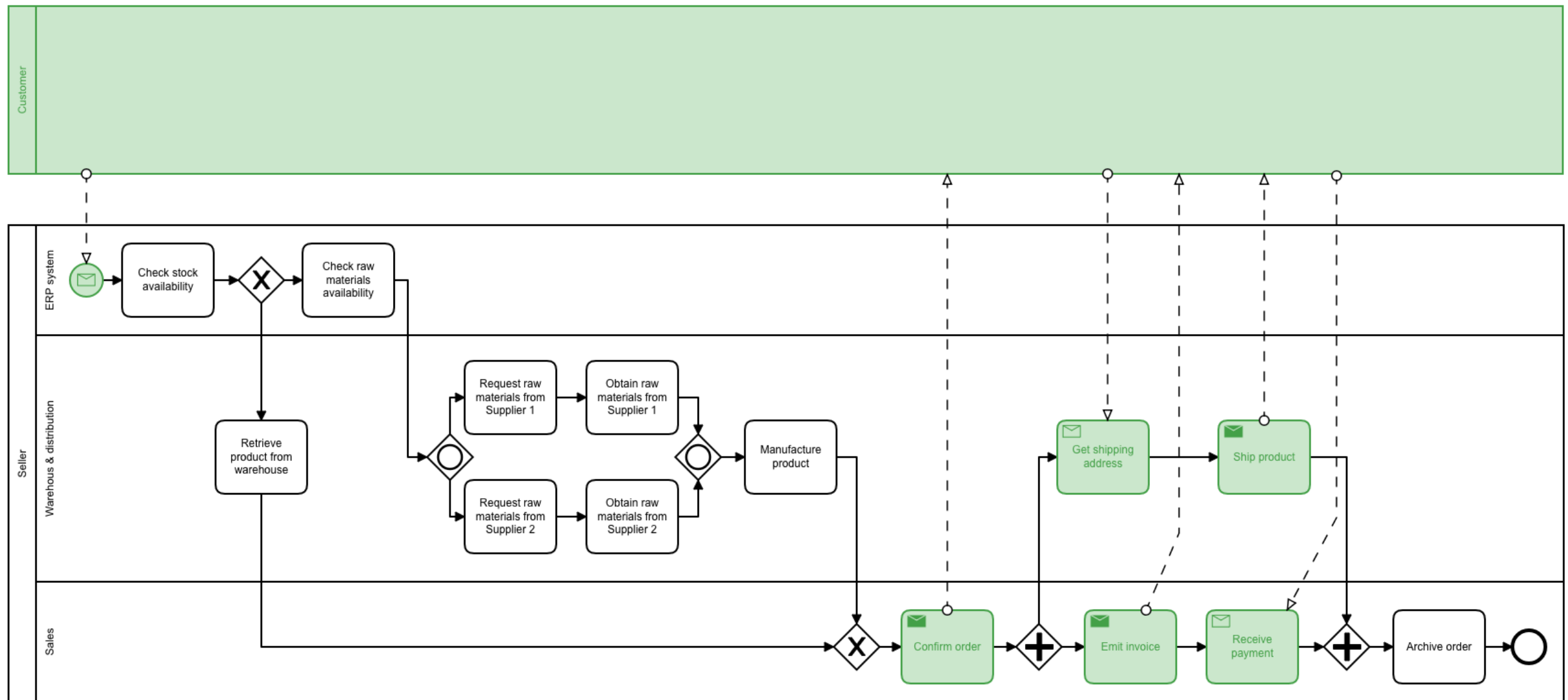
Example: Seller



Example: Seller & Customer



Example: Seller & Customer

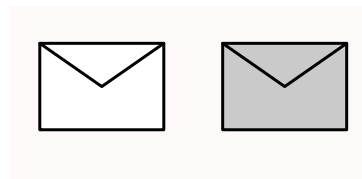


Artefacts:

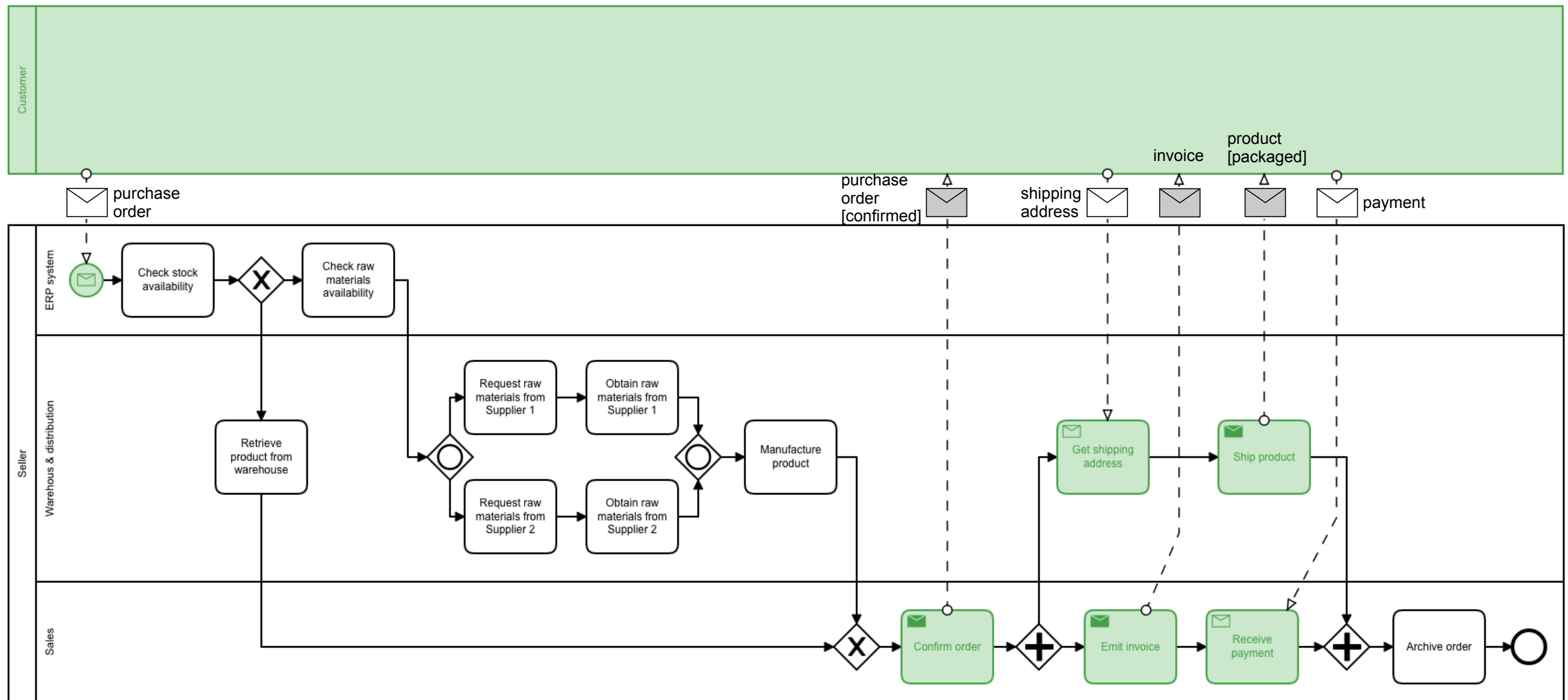
message data objects

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

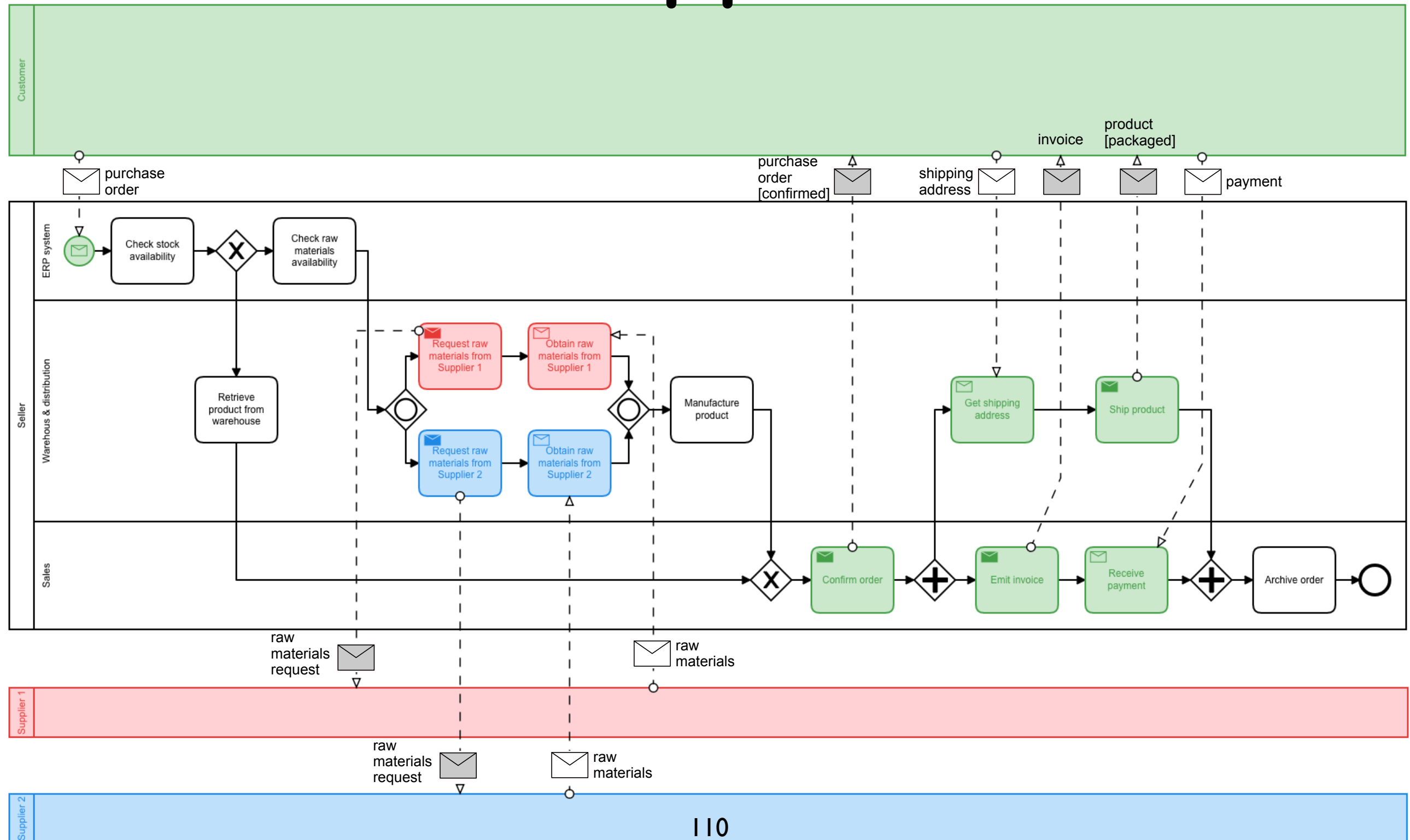
A message data object is represented as an envelope



Example: Seller & Customer

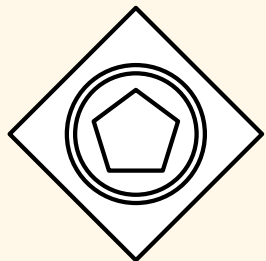


Example: Seller, Customer & Suppliers



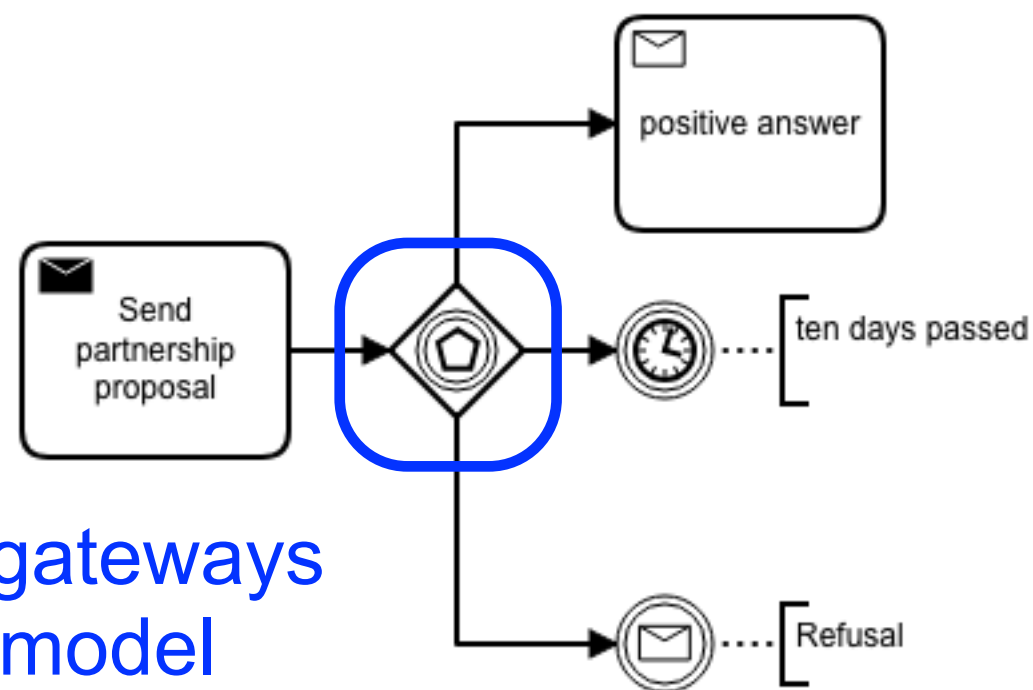
Deferred choice
(event based decisions)

Event-based decisions



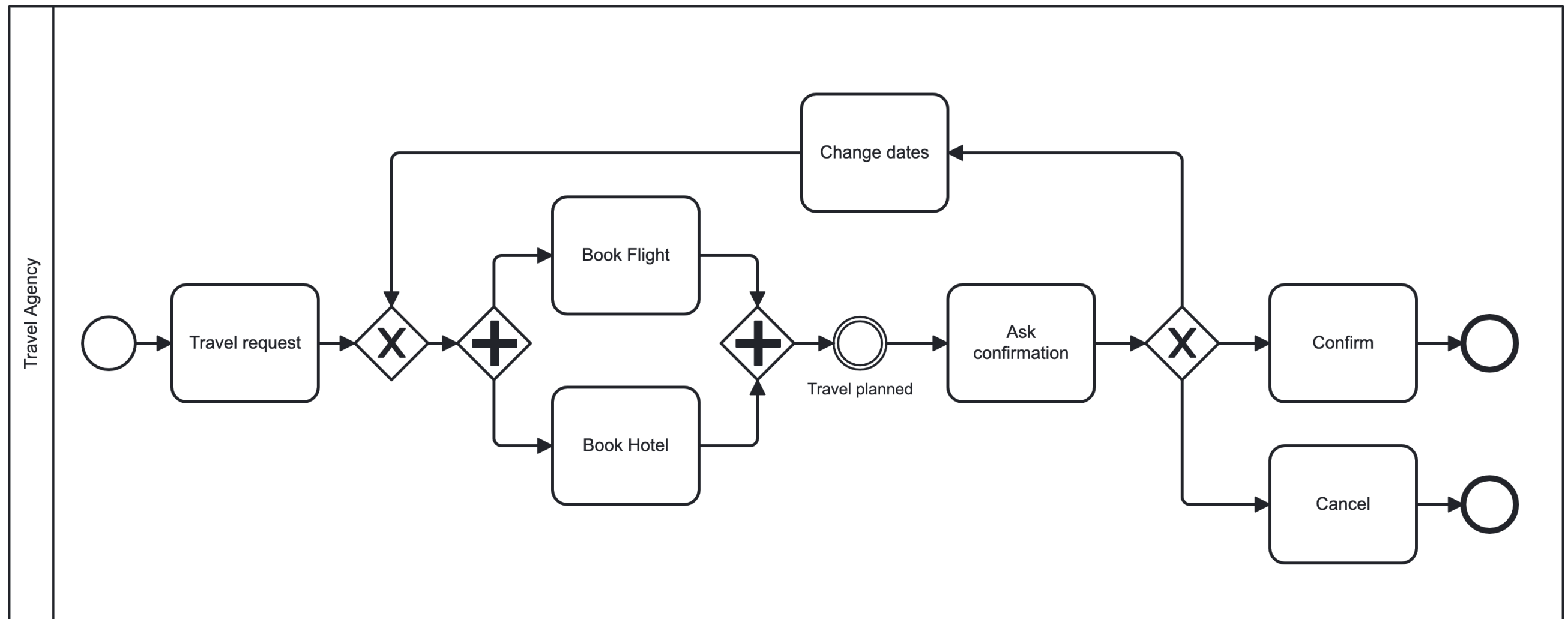
Event-based Exclusive Gateway

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

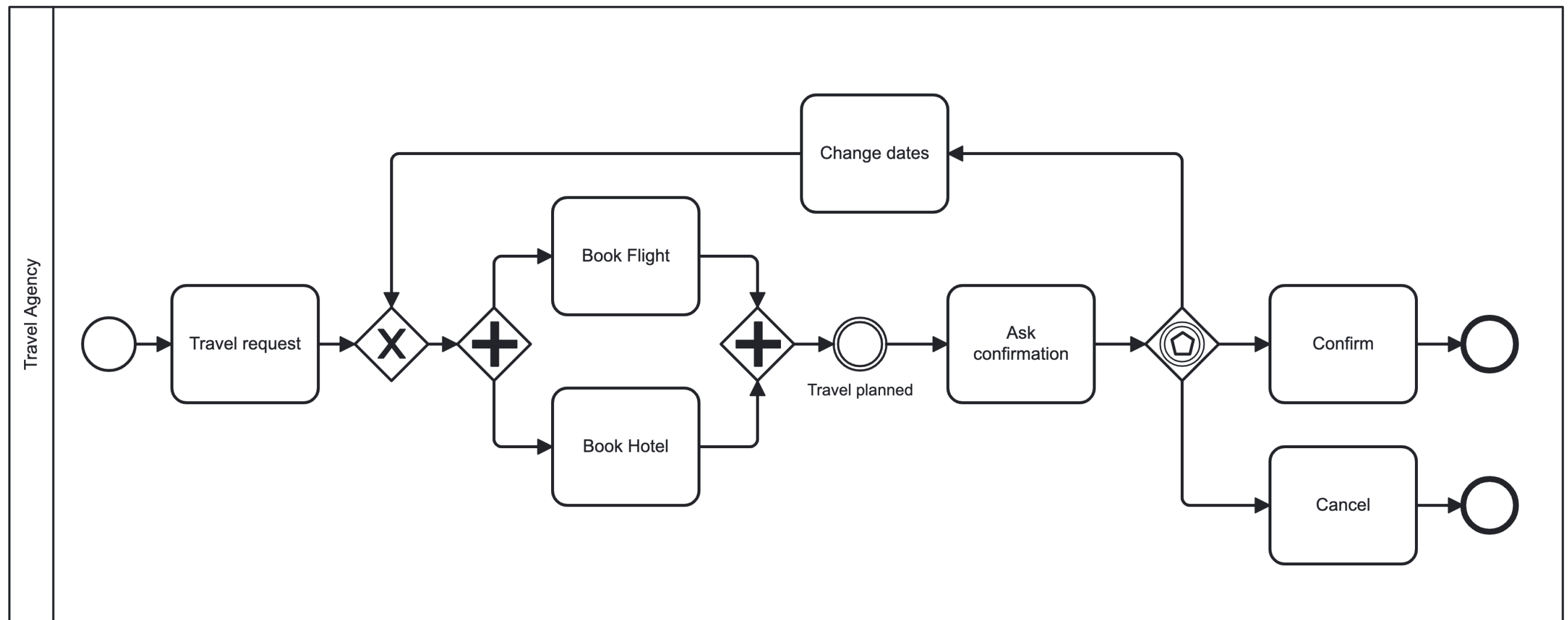


Event-based (split) gateways must be used to model decisions that depends on some external event

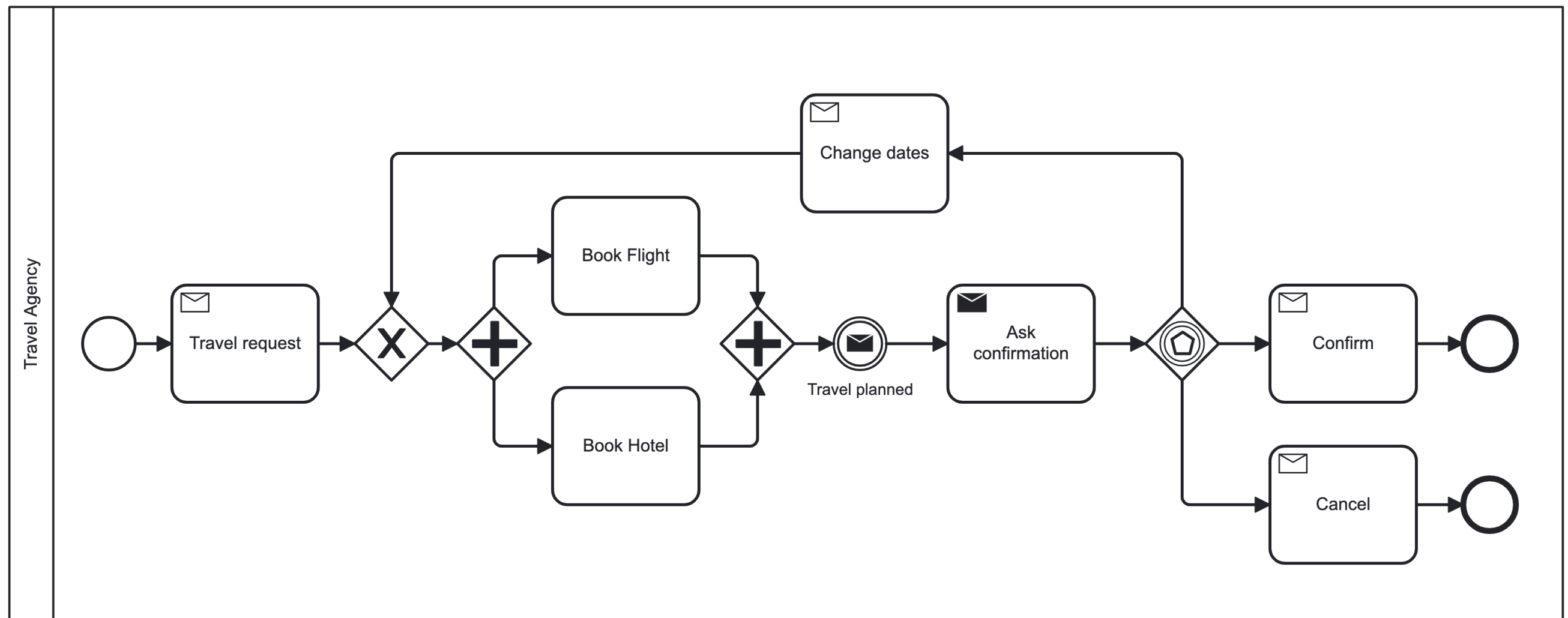
Travel agency example



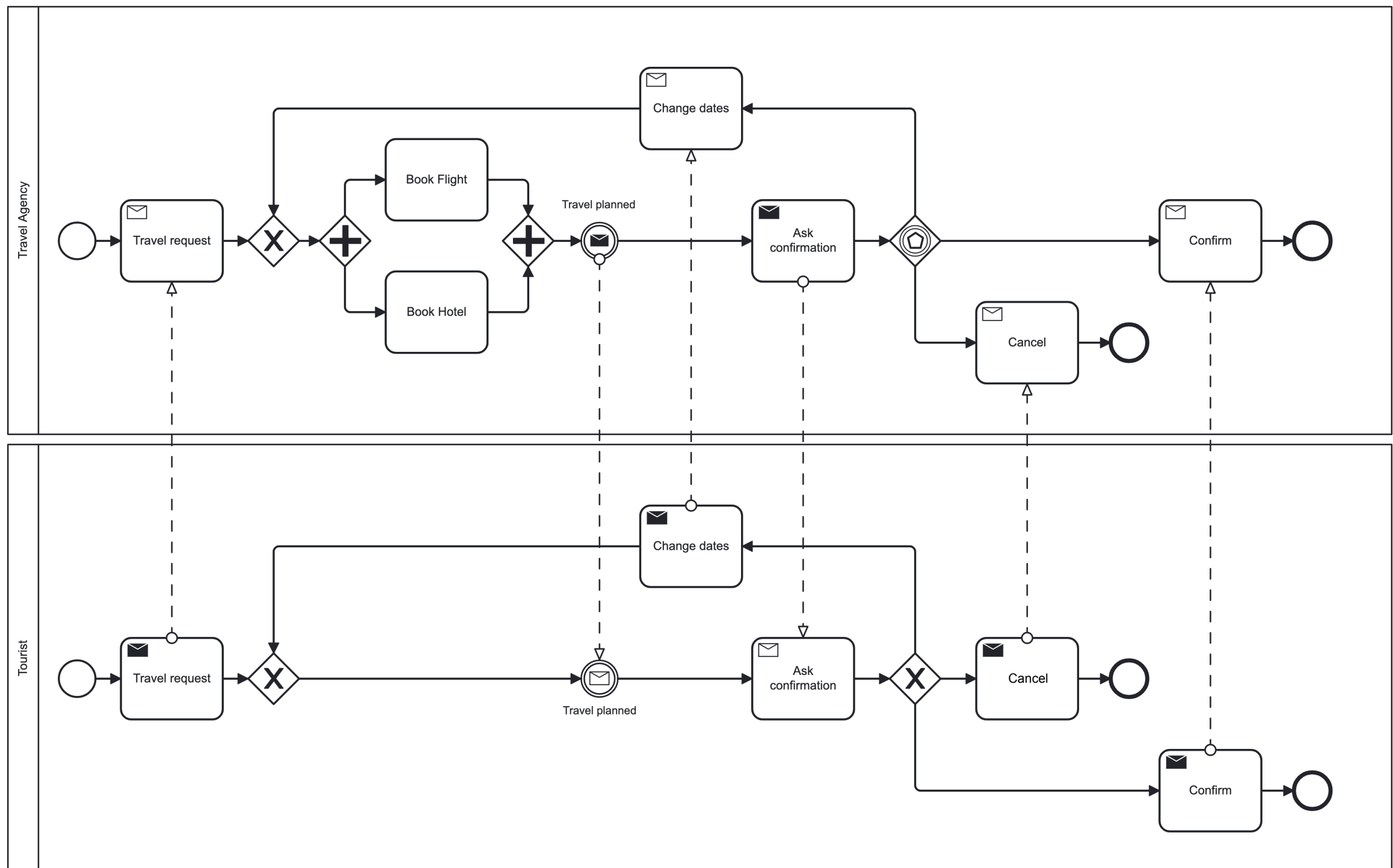
Travel agency example



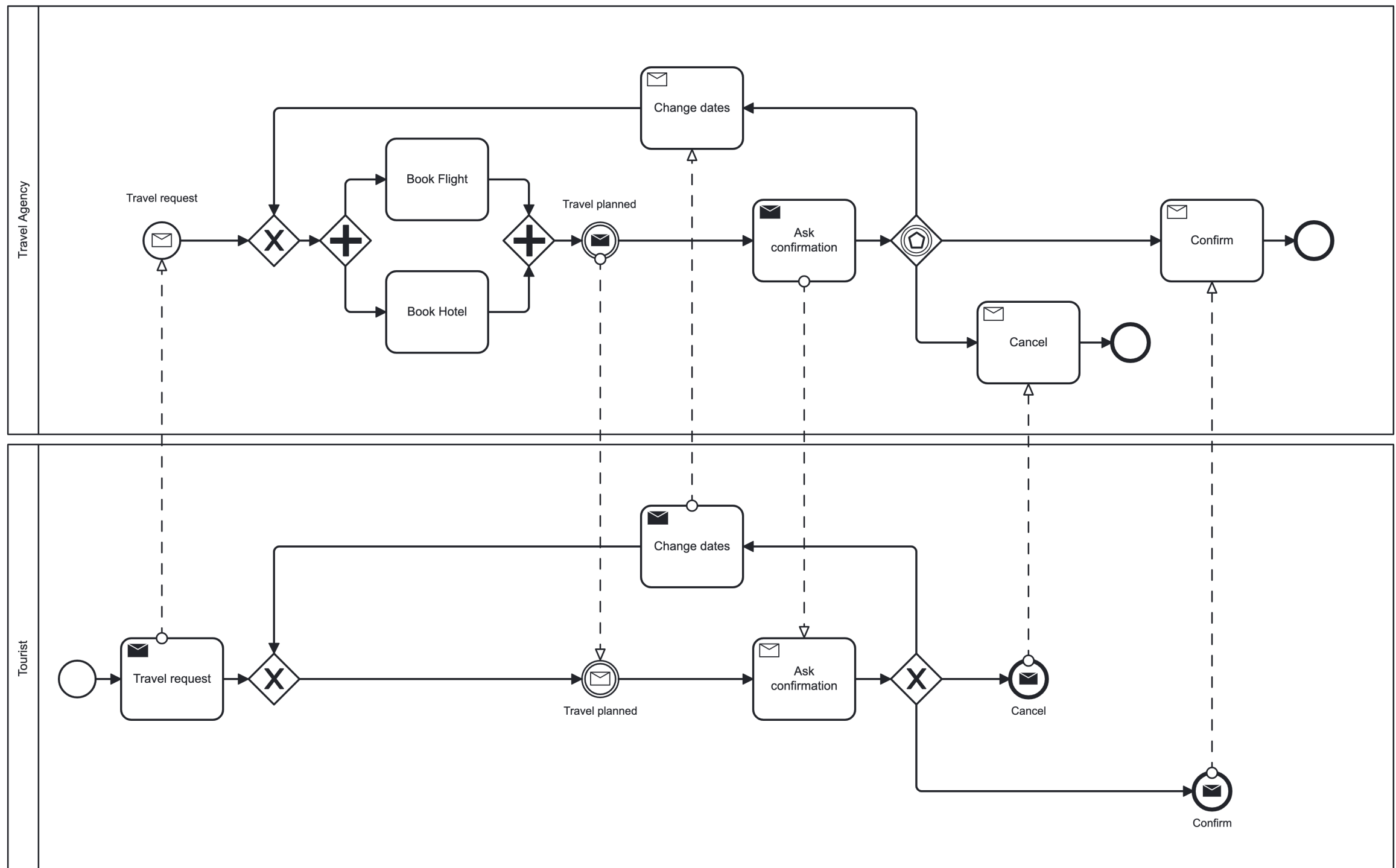
Travel agency example



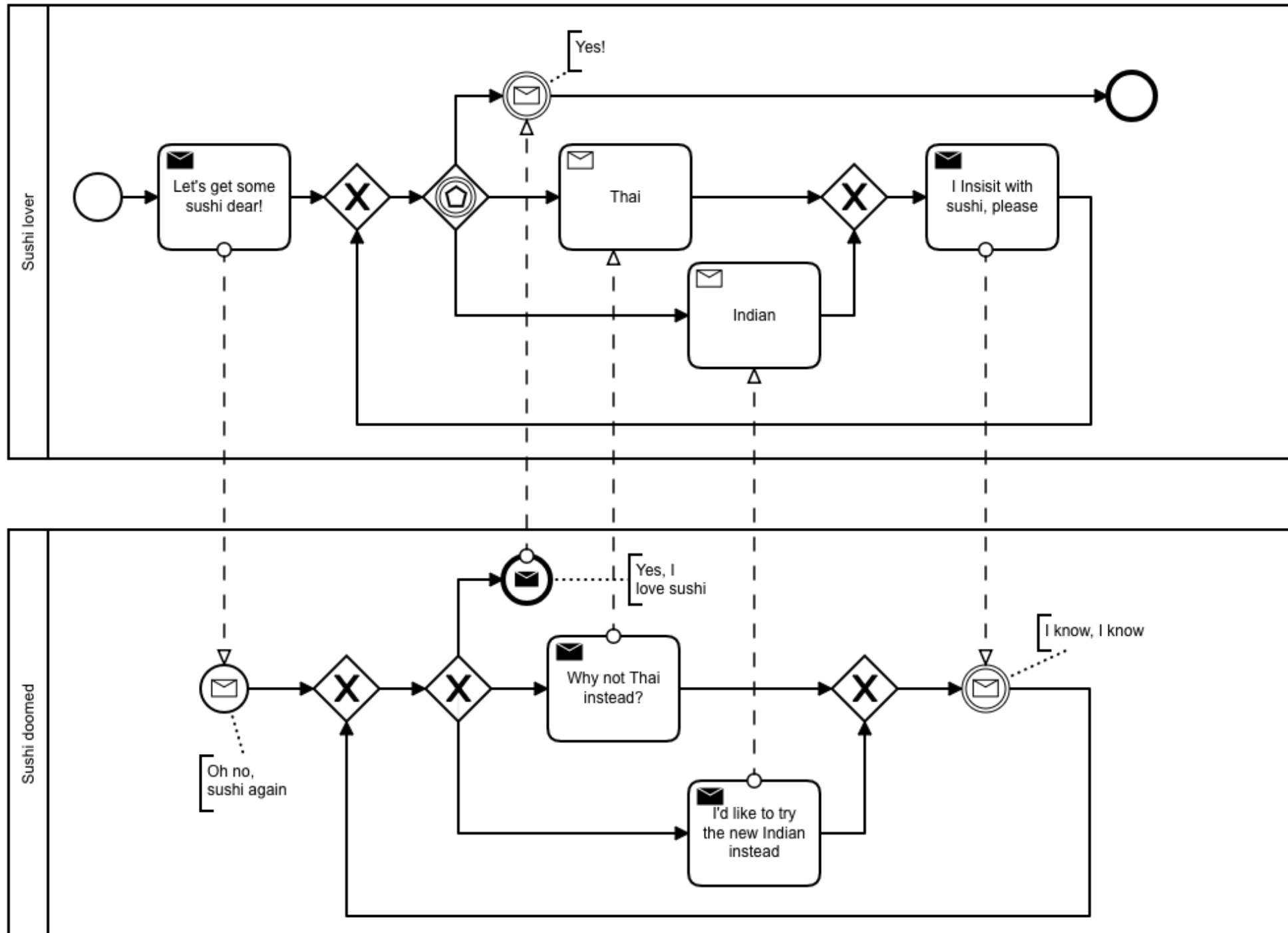
Travel agency example



Travel agency example



A negotiation without choice



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