Business Processes Modelling

MPB (6 cfu, 295AA)

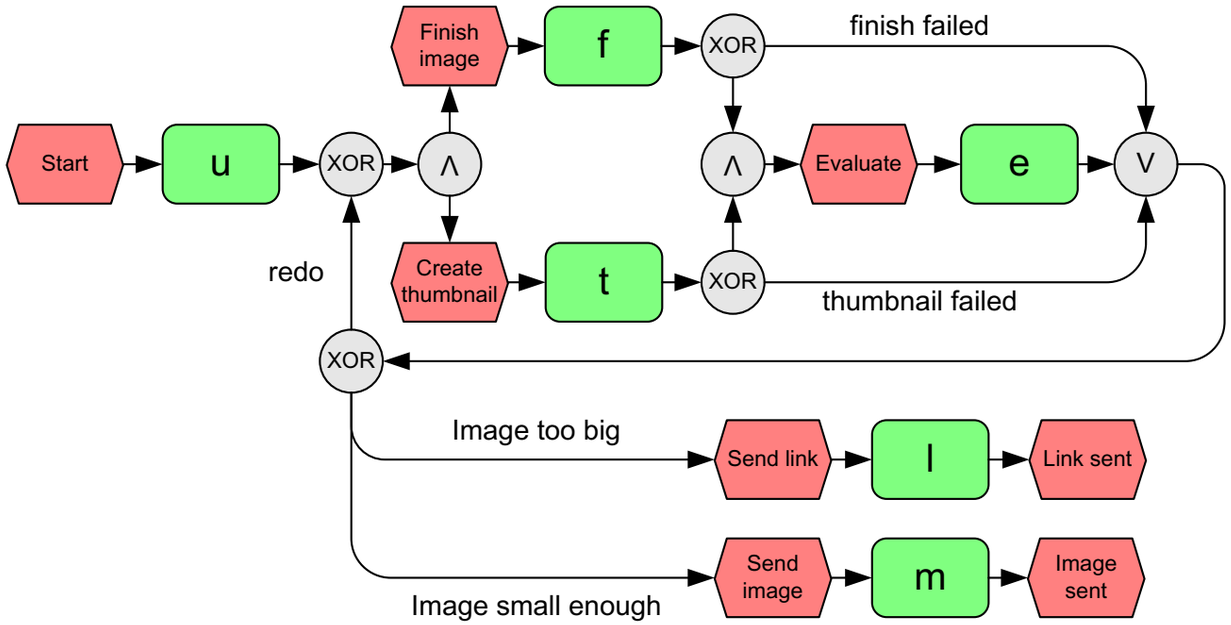


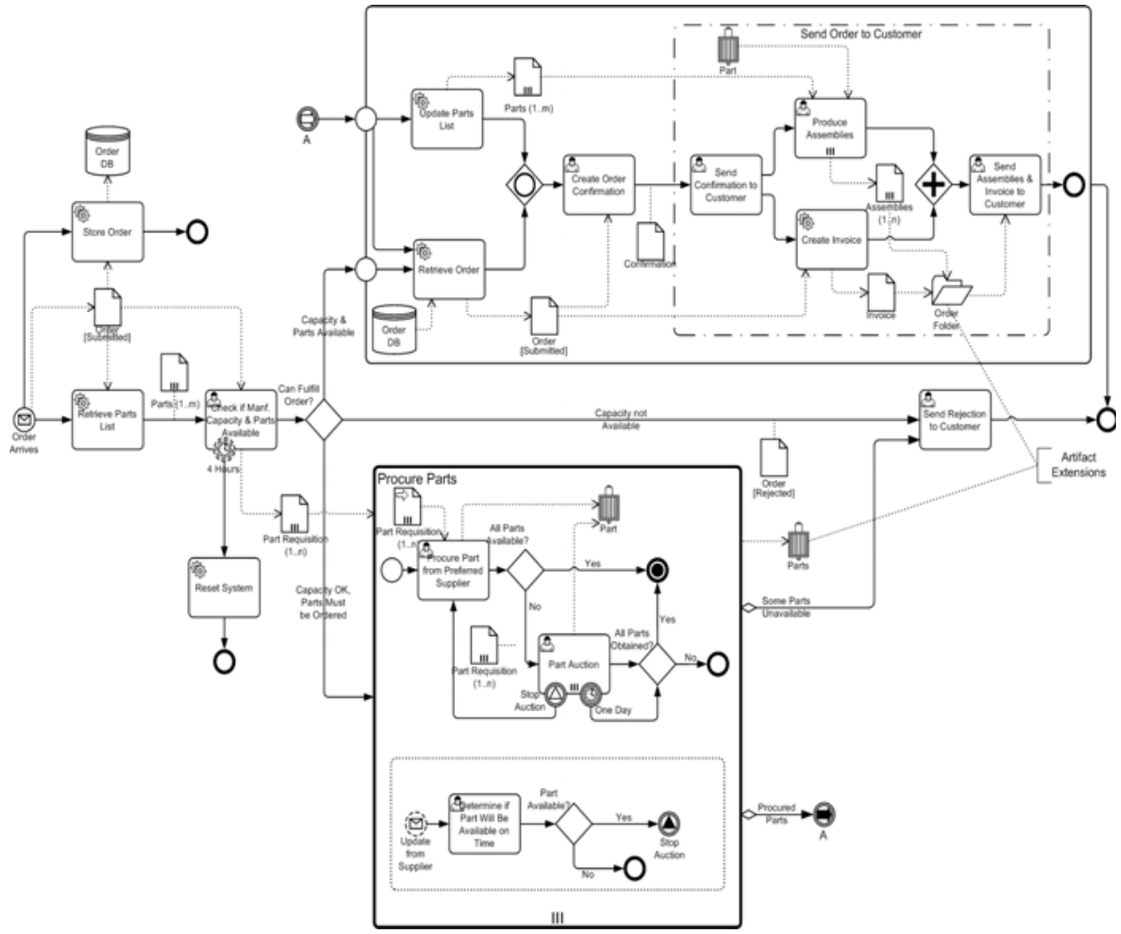
Roberto Bruni

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

07 - EPC and BPMN

Object

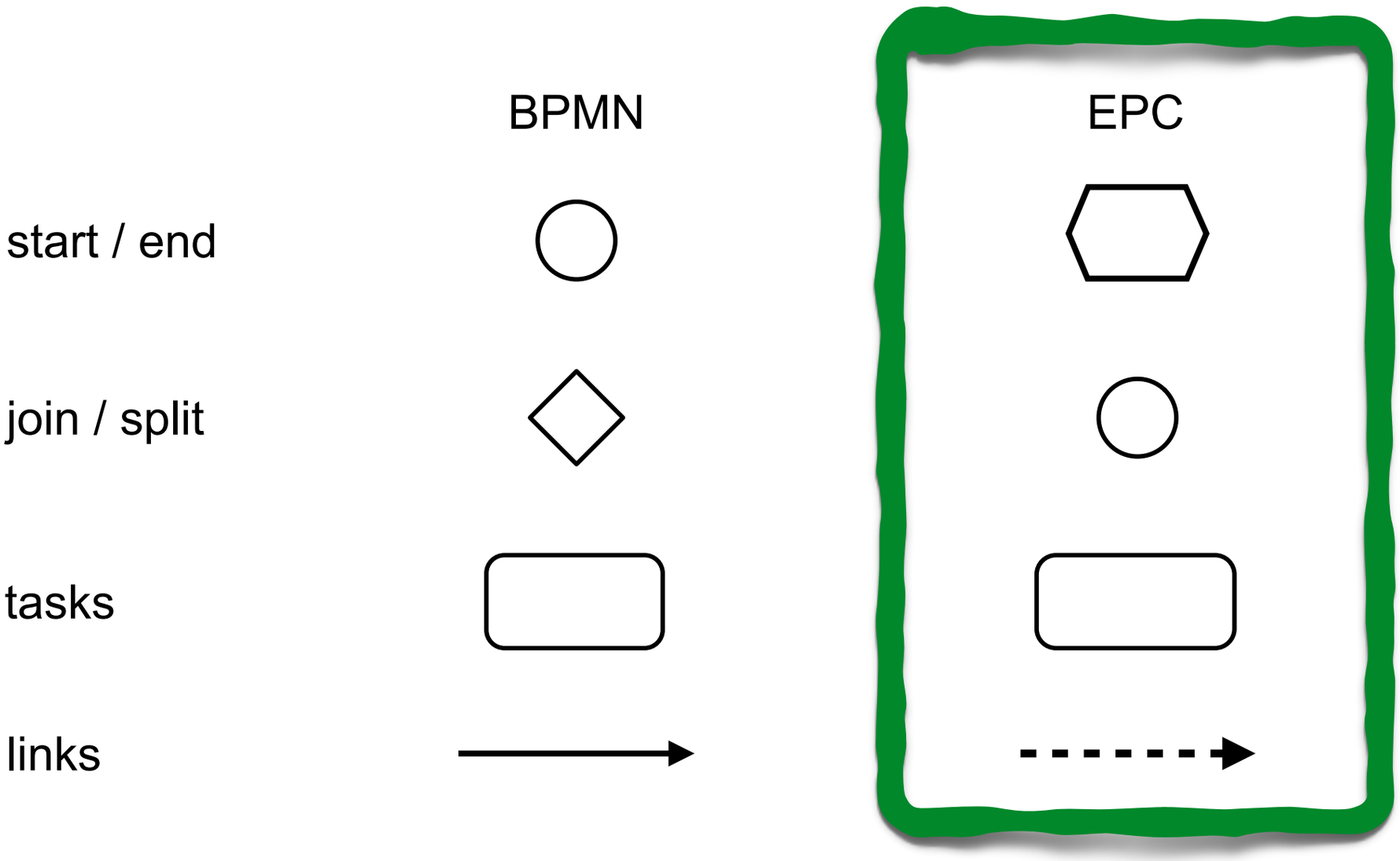




We overview high-level diagrammatic notation

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.

A closer look at standards: EPC and BPMN



EPC Diagrams: Requirements

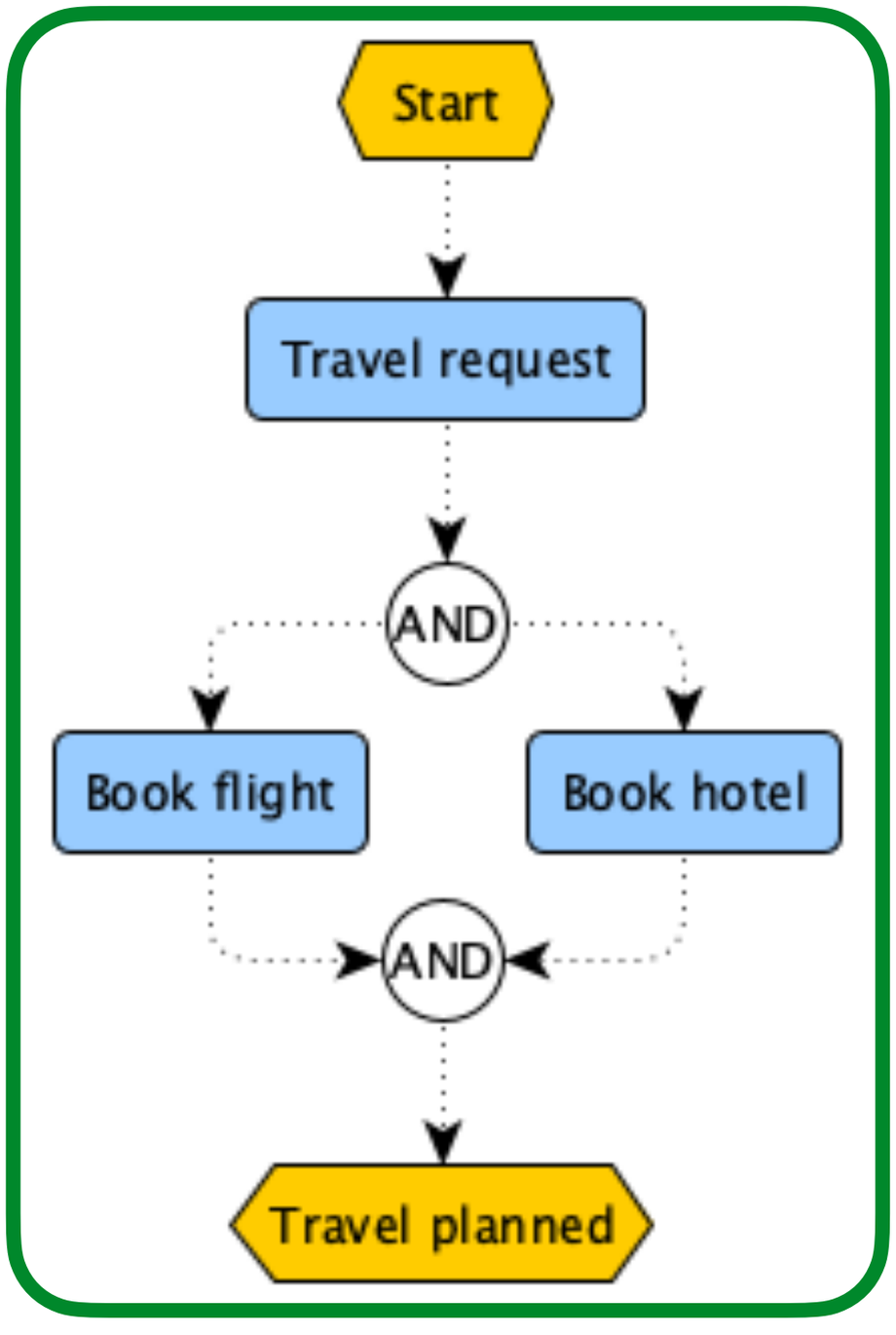
EPC diagrams: requirements

EPC elements can be combined in a fairly free manner
(possibly including cycles)

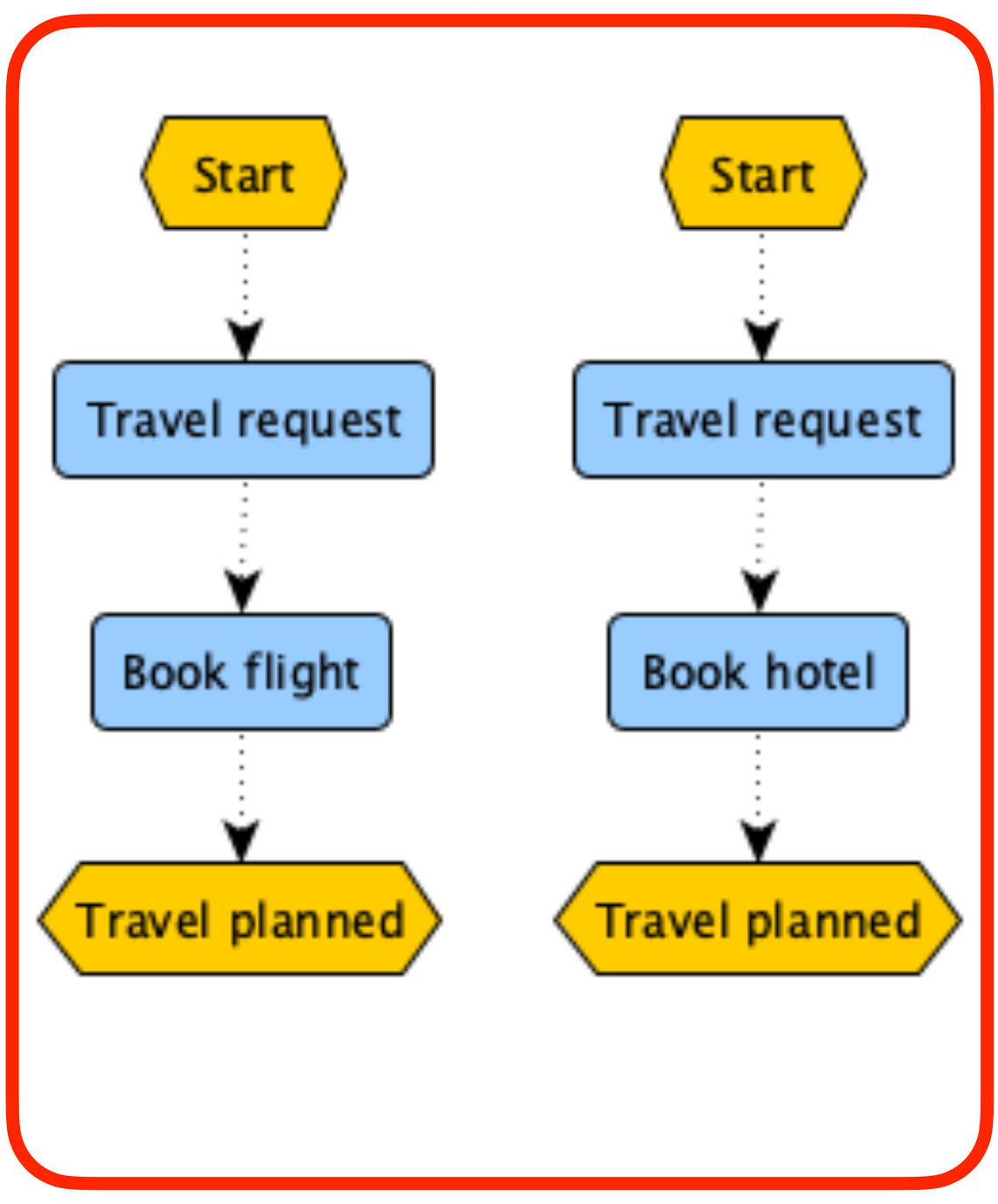
The graph must be **weakly connected** (e.g., no isolated nodes)

Weak connectivity

Weakly connected



Non weakly connected



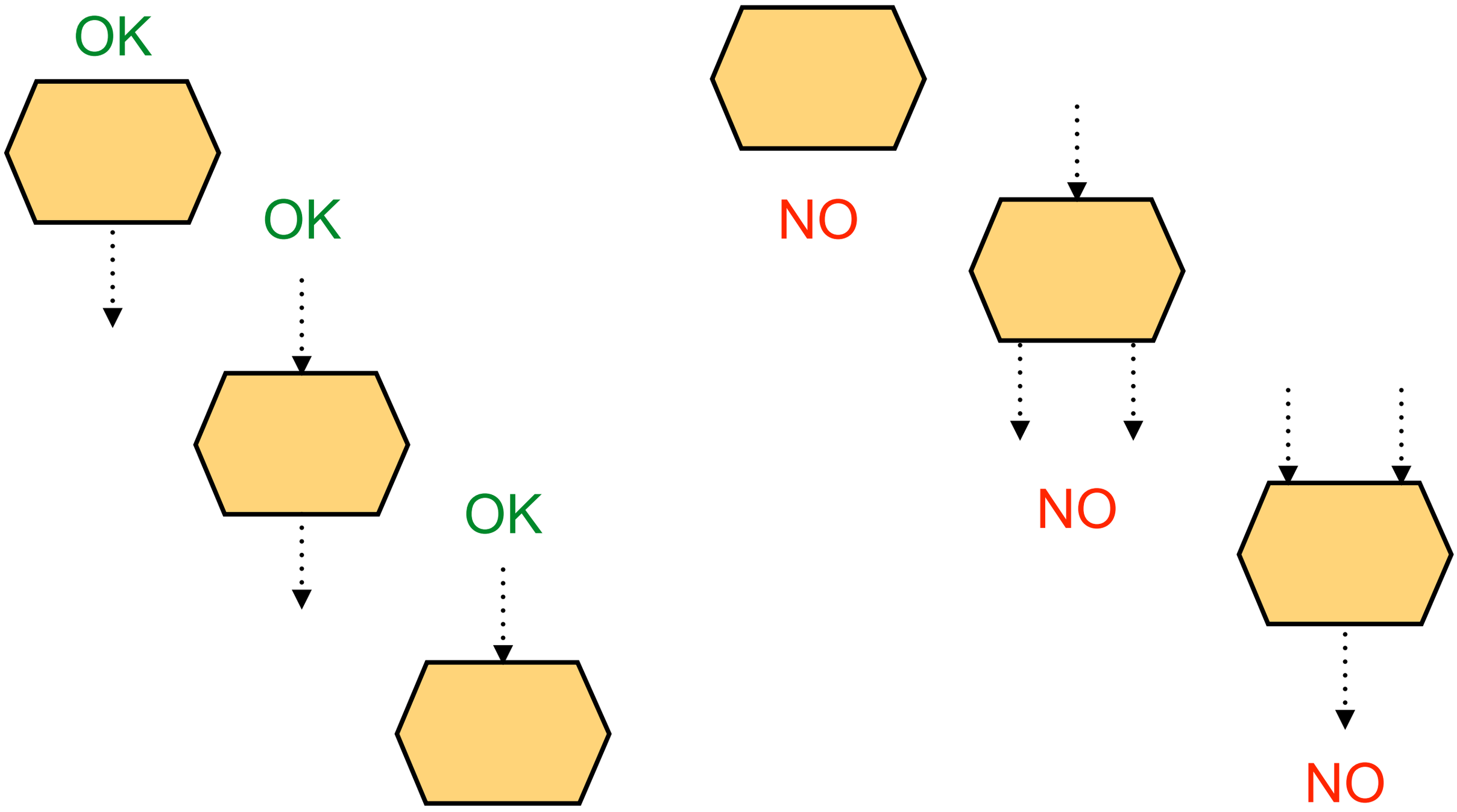
EPC diagrams: requirements

EPC elements can be combined in a fairly free manner
(possibly including cycles)

The graph must be **weakly connected** (e.g., no isolated nodes)

Events have at most one incoming and one outgoing arc
Events have at least one incident arc
There must be at least one start event and one end event

Event connectivity



EPC diagrams: requirements

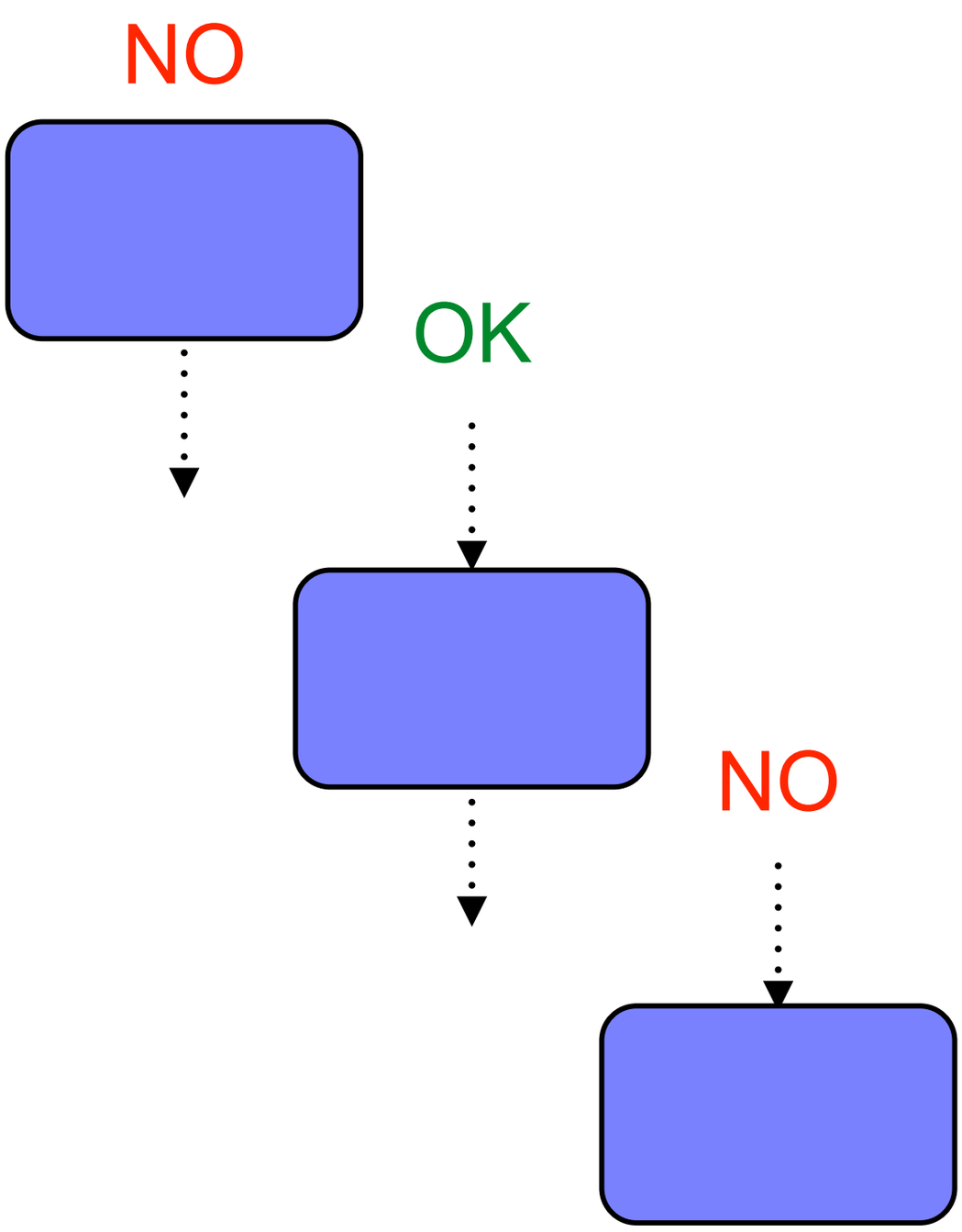
EPC elements can be combined in a fairly free manner
(possibly including cycles)

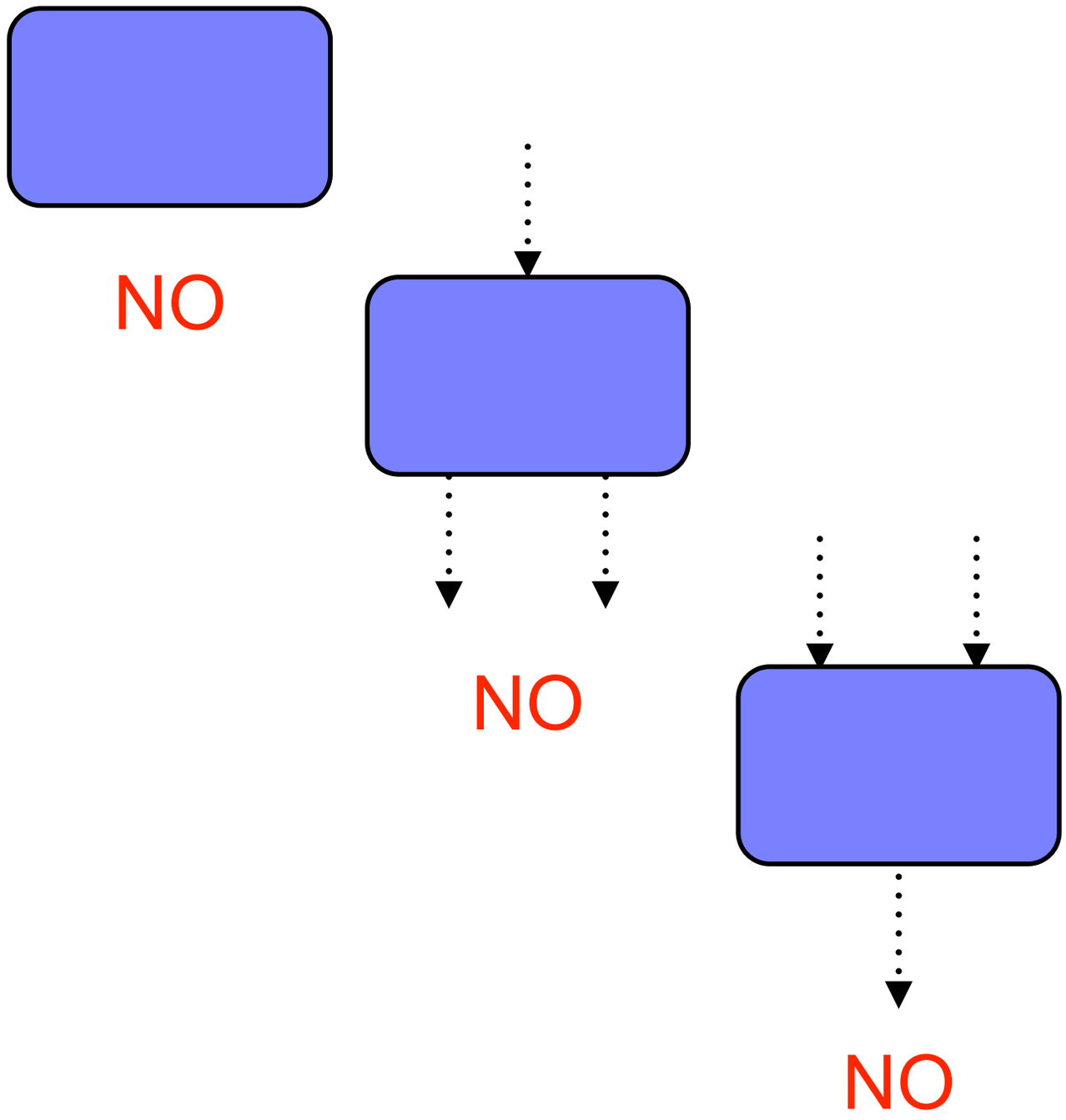
The graph must be **weakly connected** (e.g., no isolated nodes)

Events have at most one incoming and one outgoing arc
Events have at least one incident arc
There must be at least one start event and one end event

Functions have exactly one incoming and one outgoing arc

Function connectivity





EPC diagrams: requirements

EPC elements can be combined in a fairly free manner
(possibly including cycles)

The graph must be **weakly connected** (e.g., no isolated nodes)

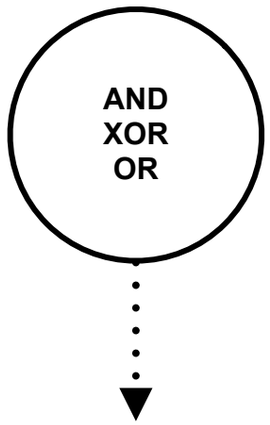
Events have at most one incoming and one outgoing arc
Events have at least one incident arc
There must be at least one start event and one end event

Functions have exactly one incoming and one outgoing arc

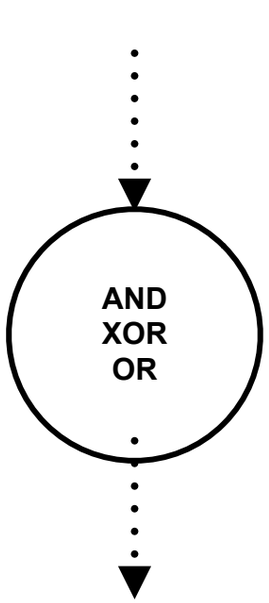
Connectors have either one incoming arc and multiple outgoing arcs
or viceversa (multiple incoming arcs and one outgoing arc)

Split/Join connectivity

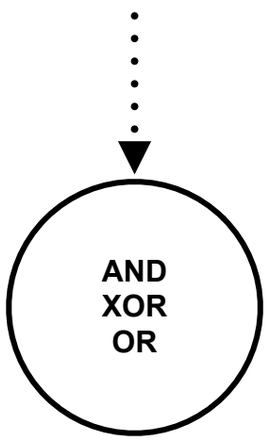
NO

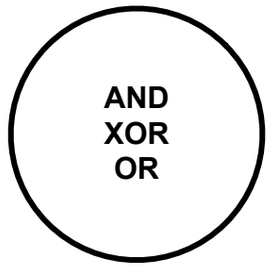


NO

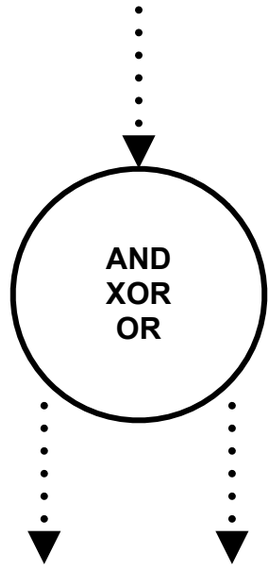


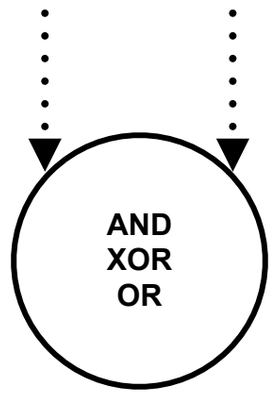
NO





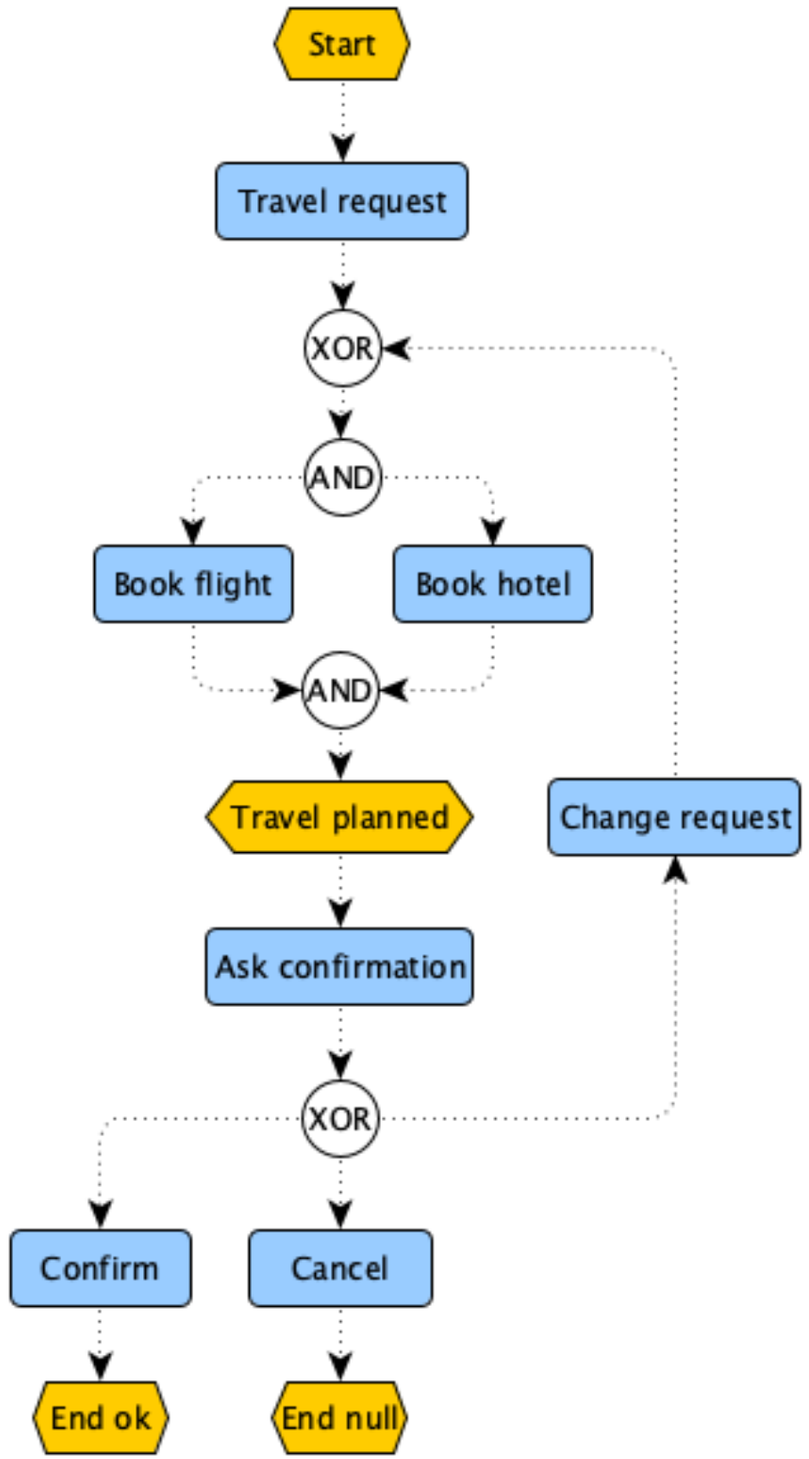
OK



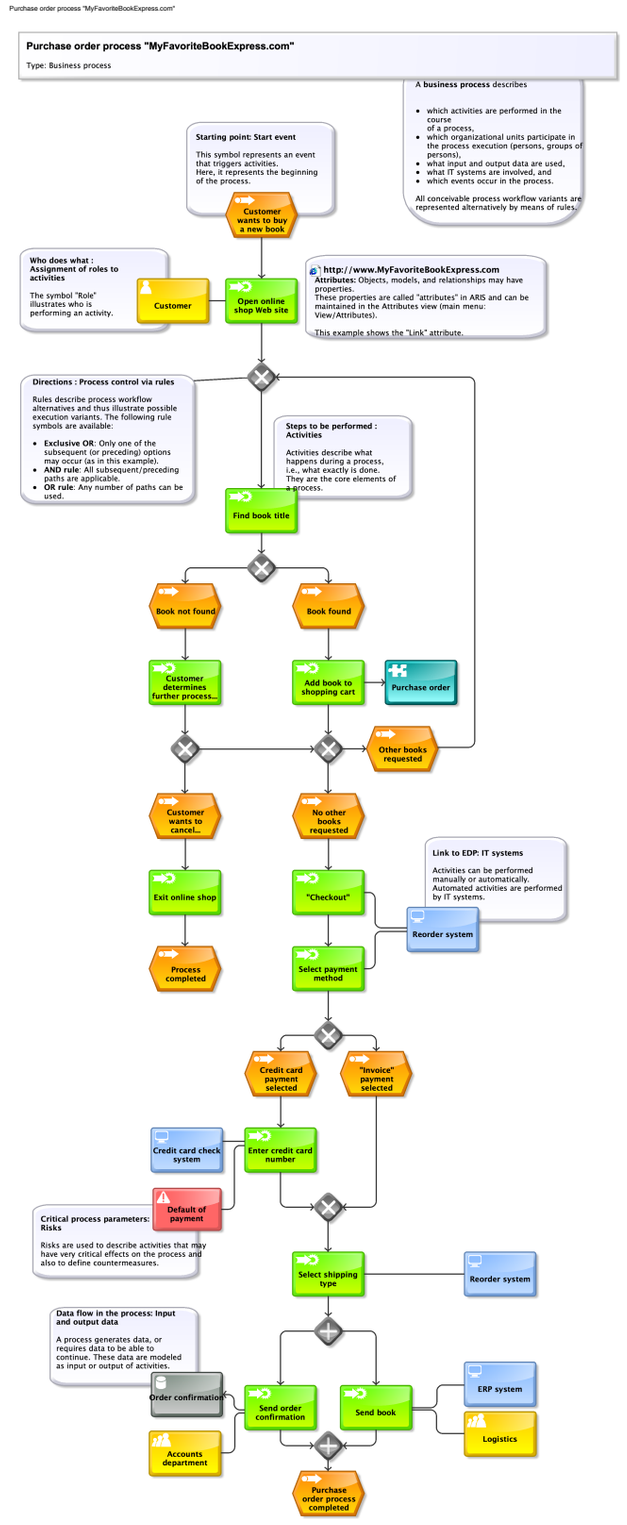


OK

EPC: Example (yEd)



EPC: Example (ARIS Express)



EPC Diagrams: Guidelines

EPC Diagrams: guidelines

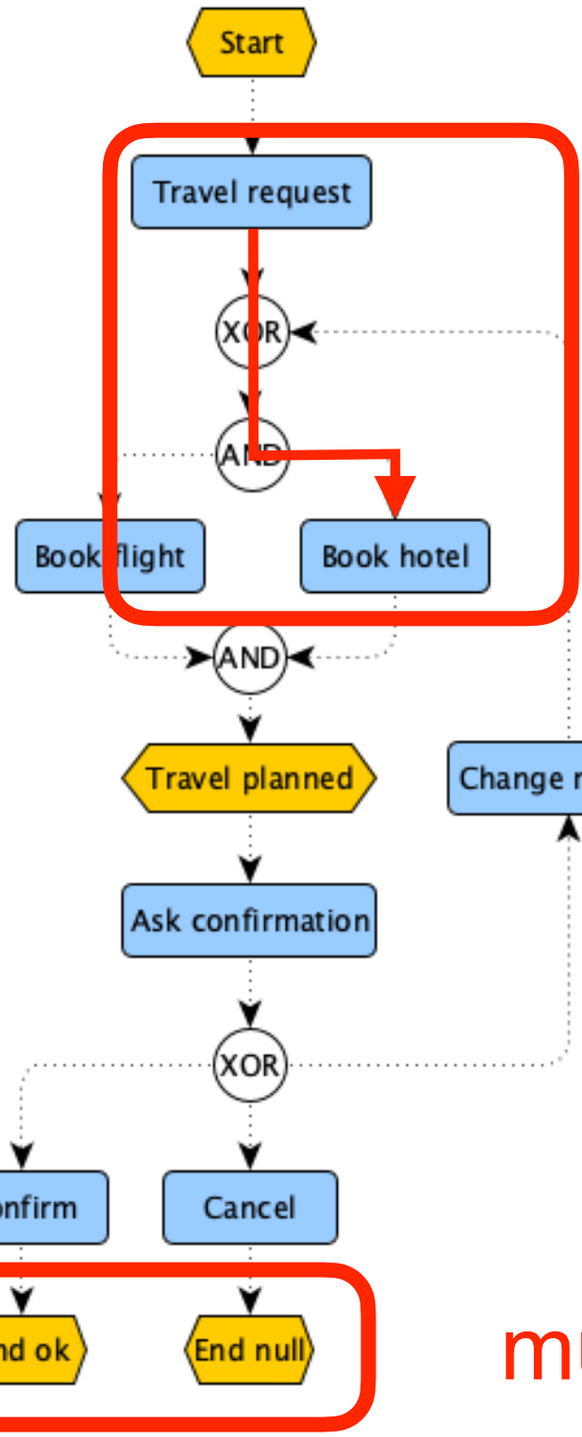
Other constraints are sometimes imposed

Unique start / end event

No direct flow between two functions

No event is followed by a decision node
(i.e. (X)OR-split)

EPC guidelines: Example



direct flow between functions

multiple end events

Problem with guidelines

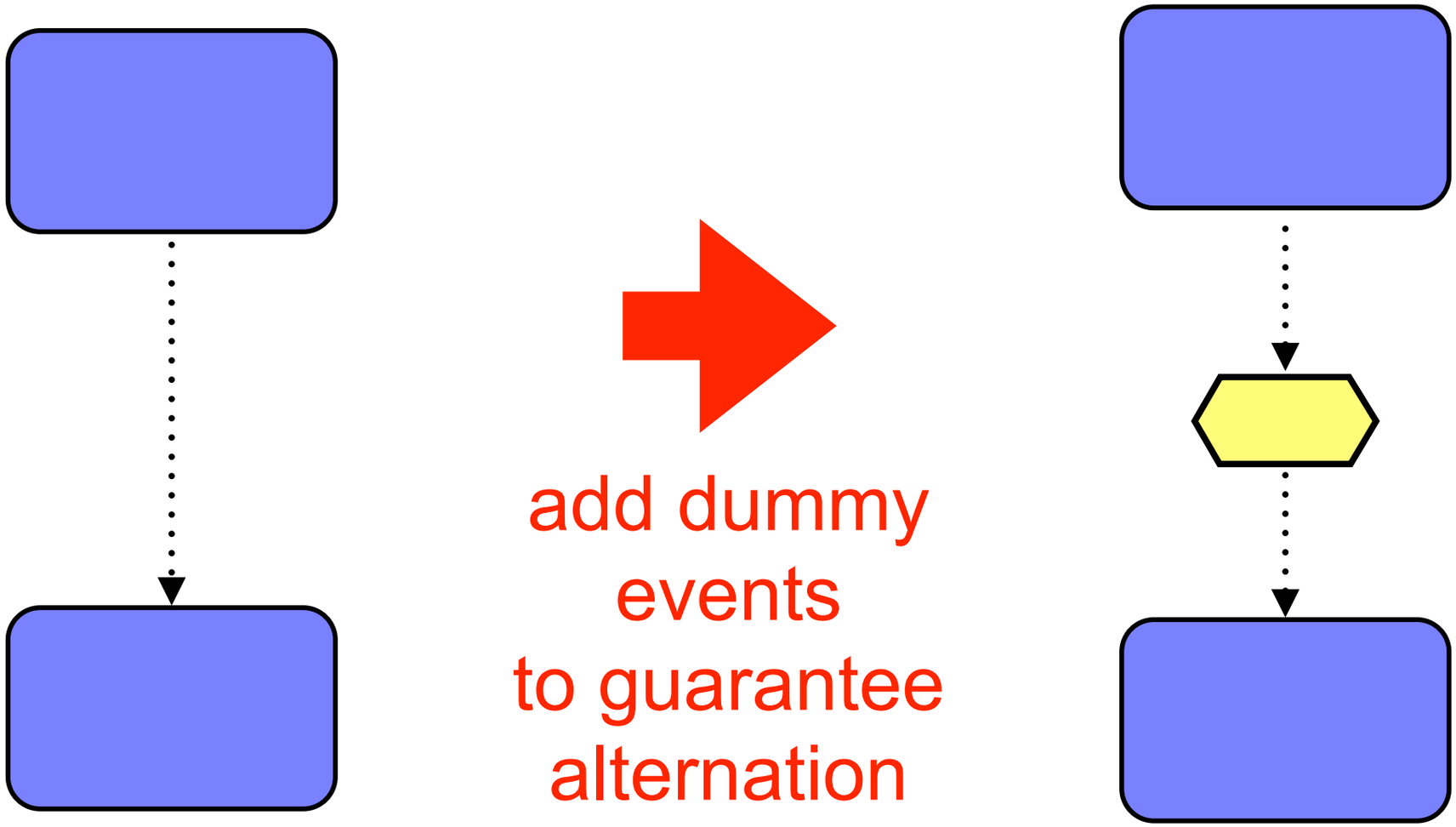
From empirical studies:
guidelines are too restrictive and people ignore them
(following rigid guidelines can lead to unnecessarily complicated diagrams, more difficult to read and understand)

Solution:
It is safe to drop most constraints
(e.g., implicit dummy nodes might always be added later)

EPC: repairing alternation

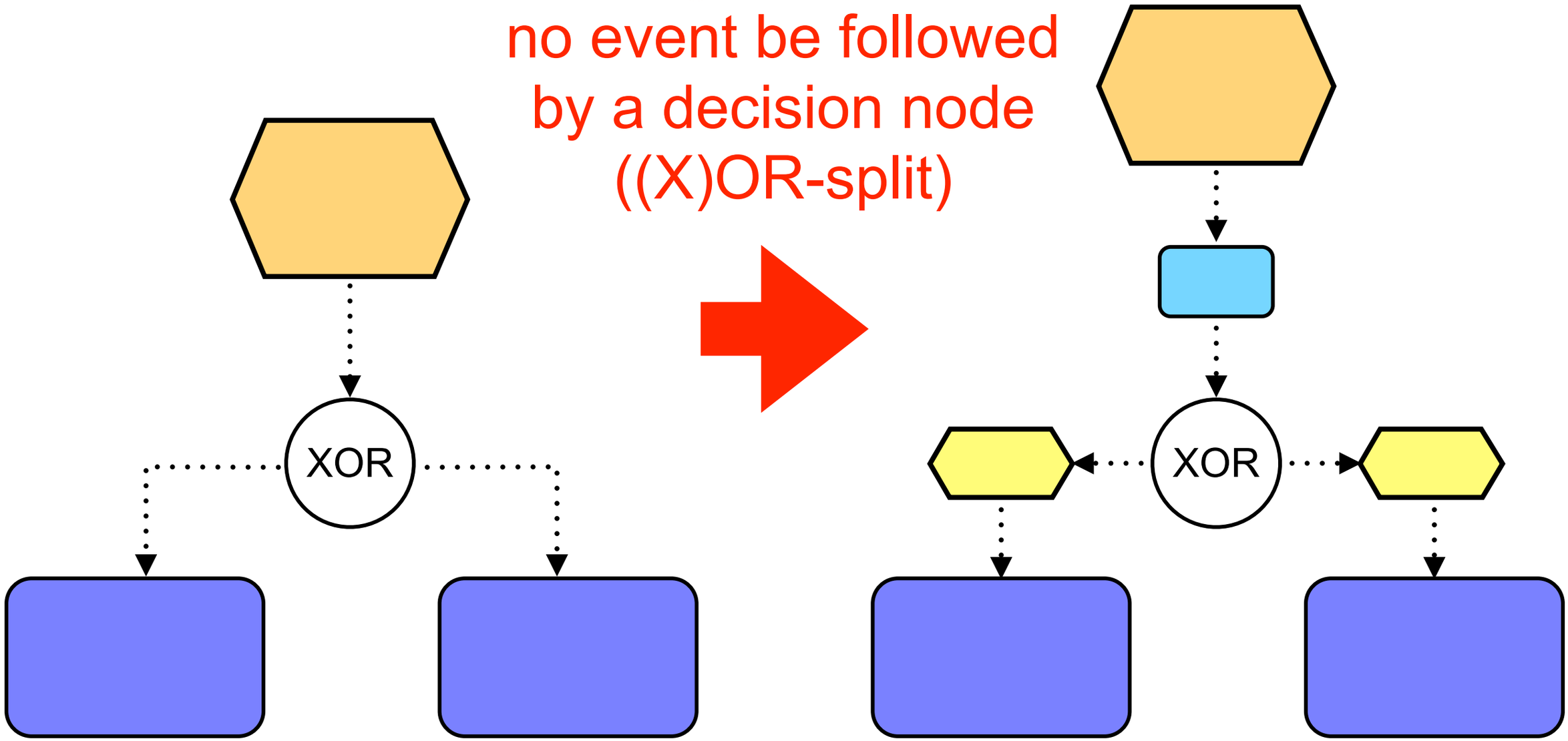


EPC: repairing alternation



EPC: repairing decisions

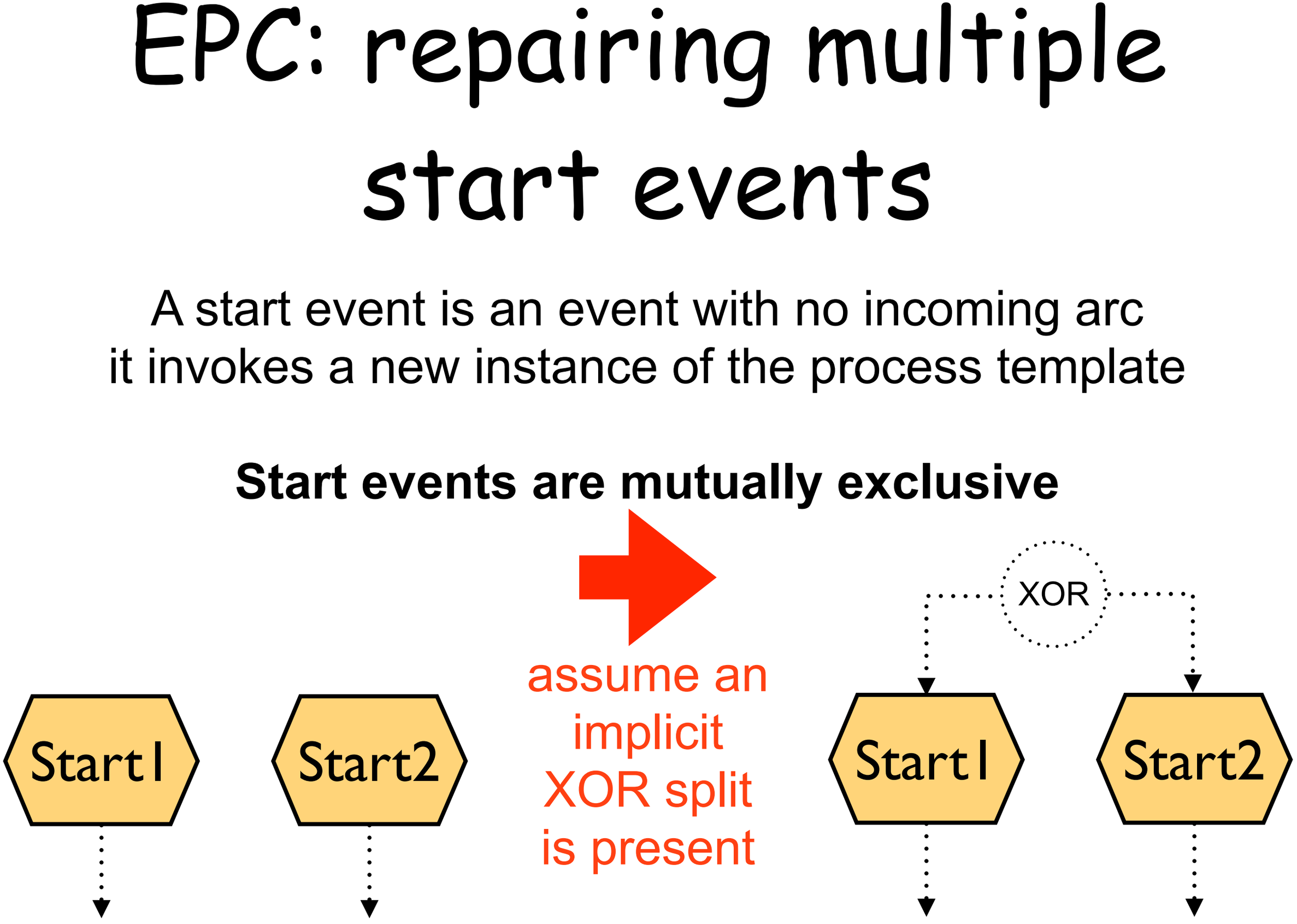
add dummy nodes
to guarantee
no event be followed
by a decision node
((X)OR-split)



EPC: repairing multiple start events

A start event is an event with no incoming arc
it invokes a new instance of the process template

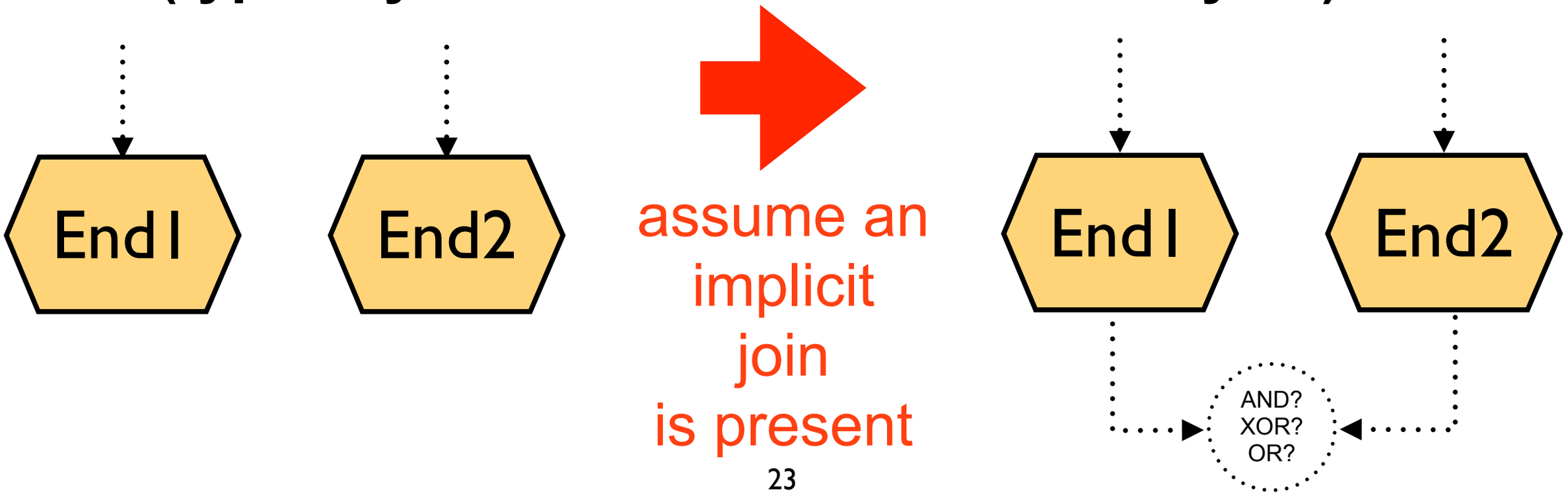
Start events are mutually exclusive



EPC: repairing multiple end events

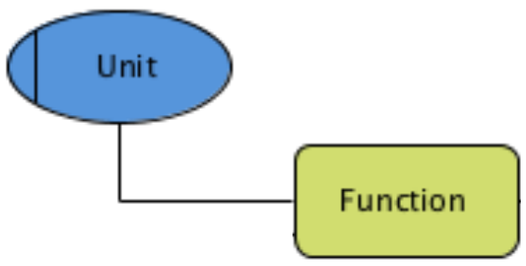
An end event is an event with no outgoing arc
it indicates completion of some activities
What if multiple end events occur? No unanimity!

they are followed by an implicit join connector (typically a XOR… but not necessarily so)

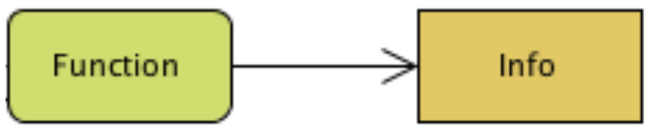


Other ingredients: function annotations

Organization unit:
determines the person or organization responsible for a specific function
(ellipses with a vertical line)

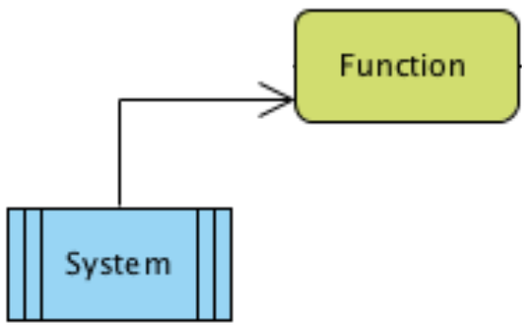


Other ingredients: function annotations



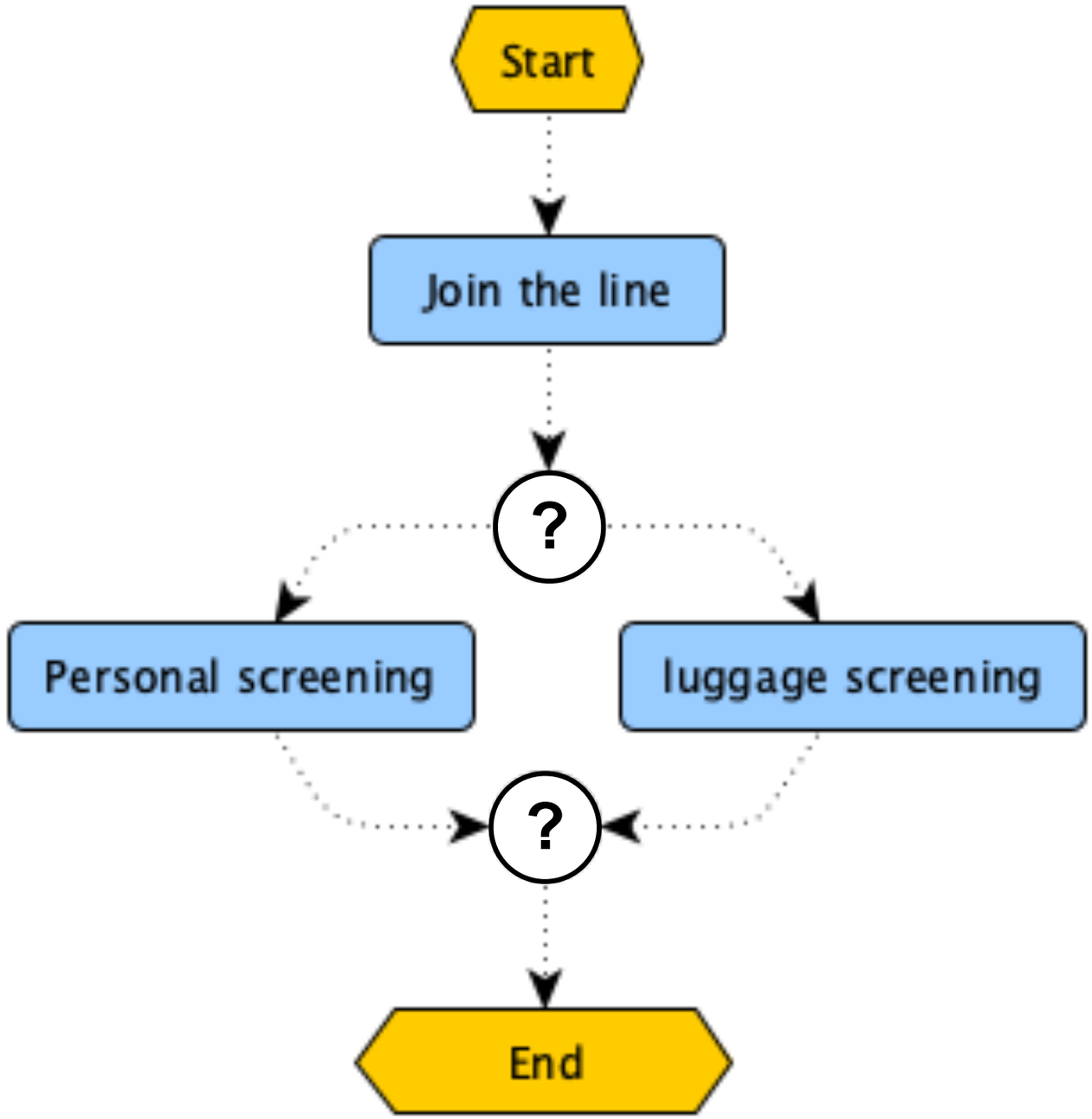
Information, material, resource object:
represents objects in the real world
e.g. input data or output data for a function
(rectangles linked to function boxes)
angles with vertical lines on its sides)

Other ingredients: function annotations

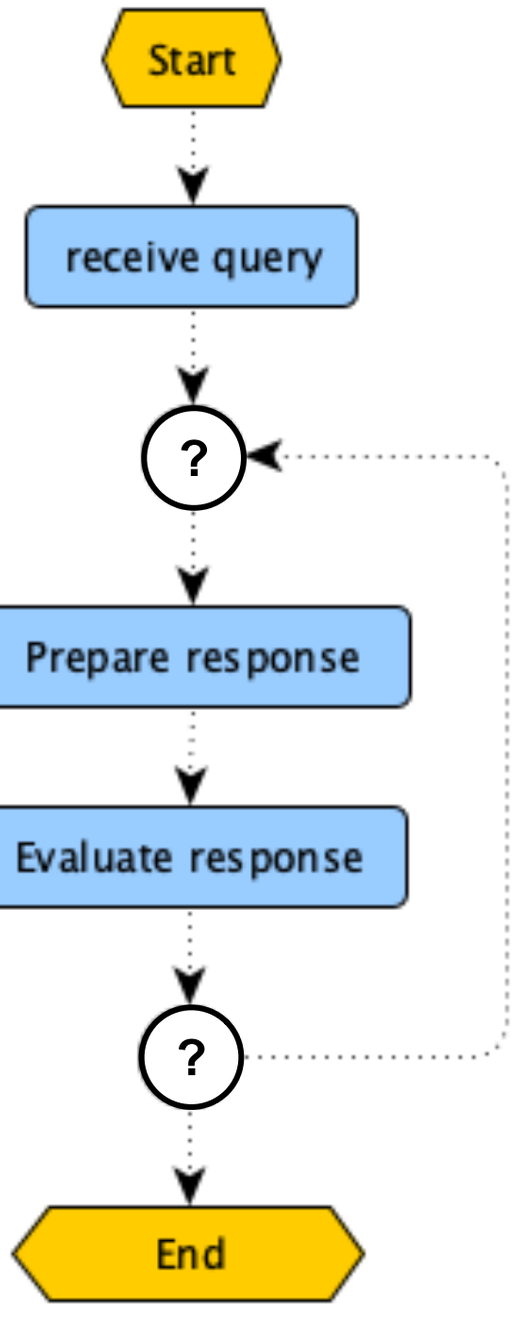


Supporting system: technical support (rectangles with vertical lines on its sides)

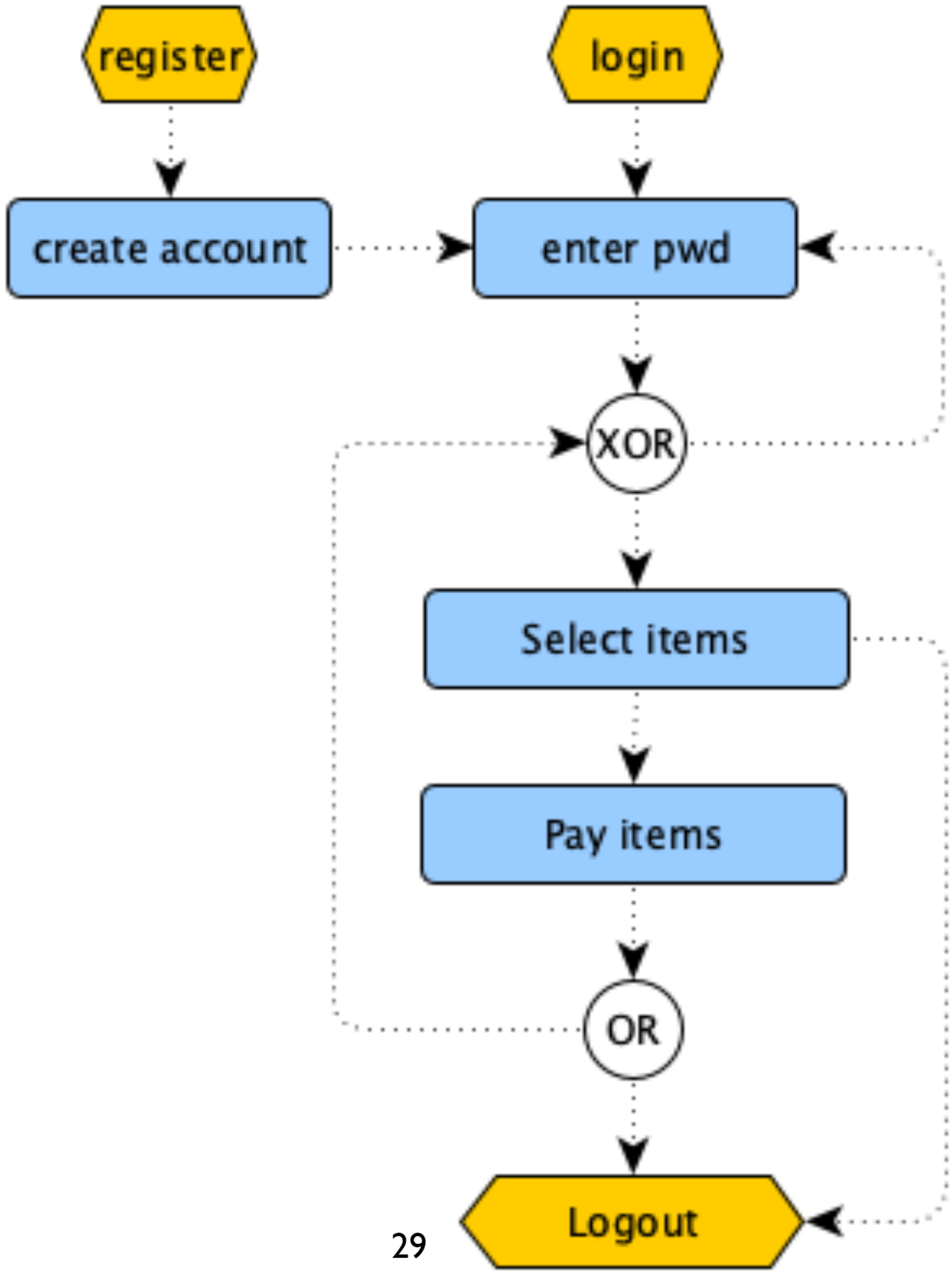
Question time: which connectors?



Question time: which connectors?



Question time: what's wrong?



EPC Sample Diagrams

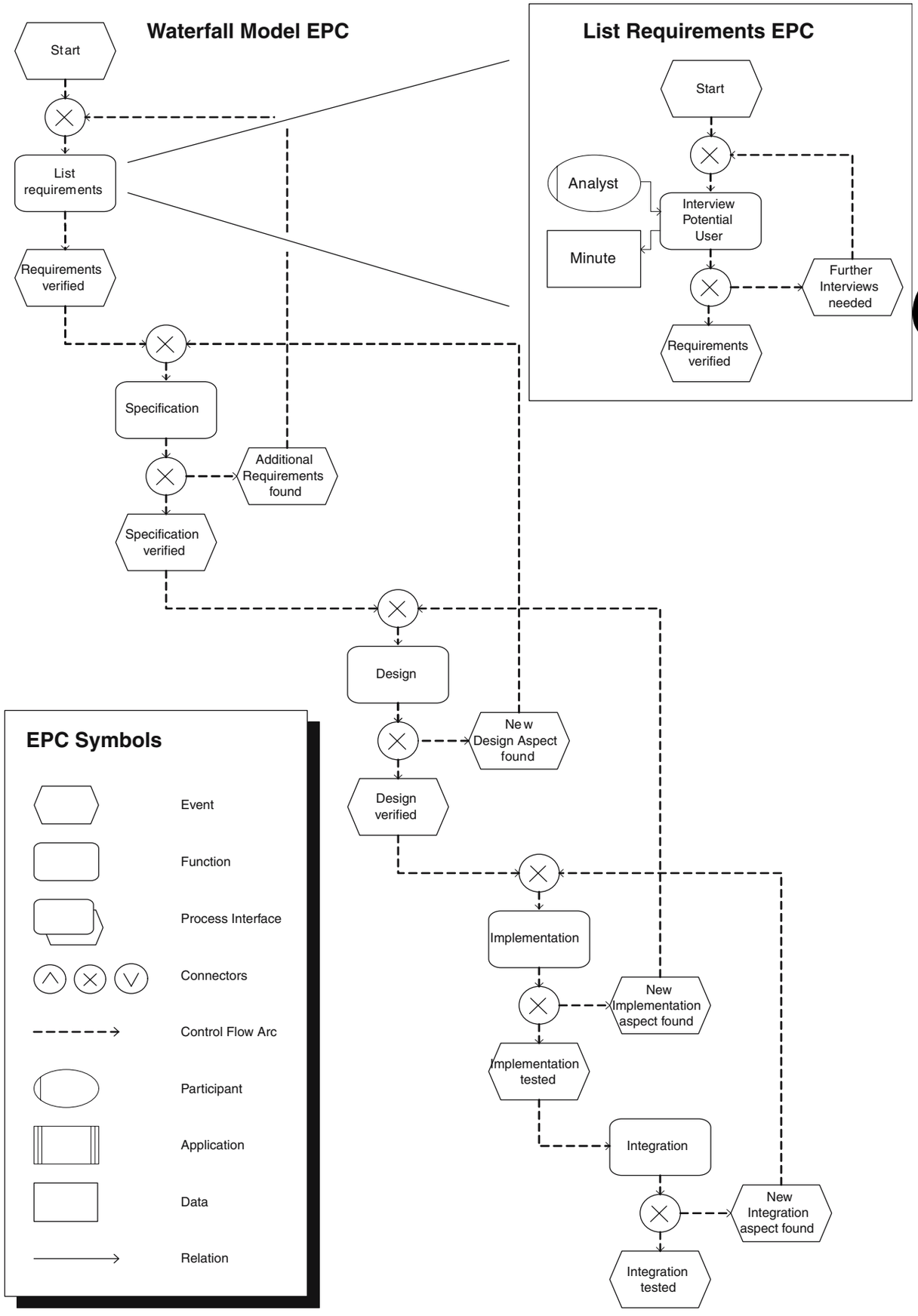


Fig. 1 Event-driven process chains representing the waterfall model for software engineering

Example: comments?

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

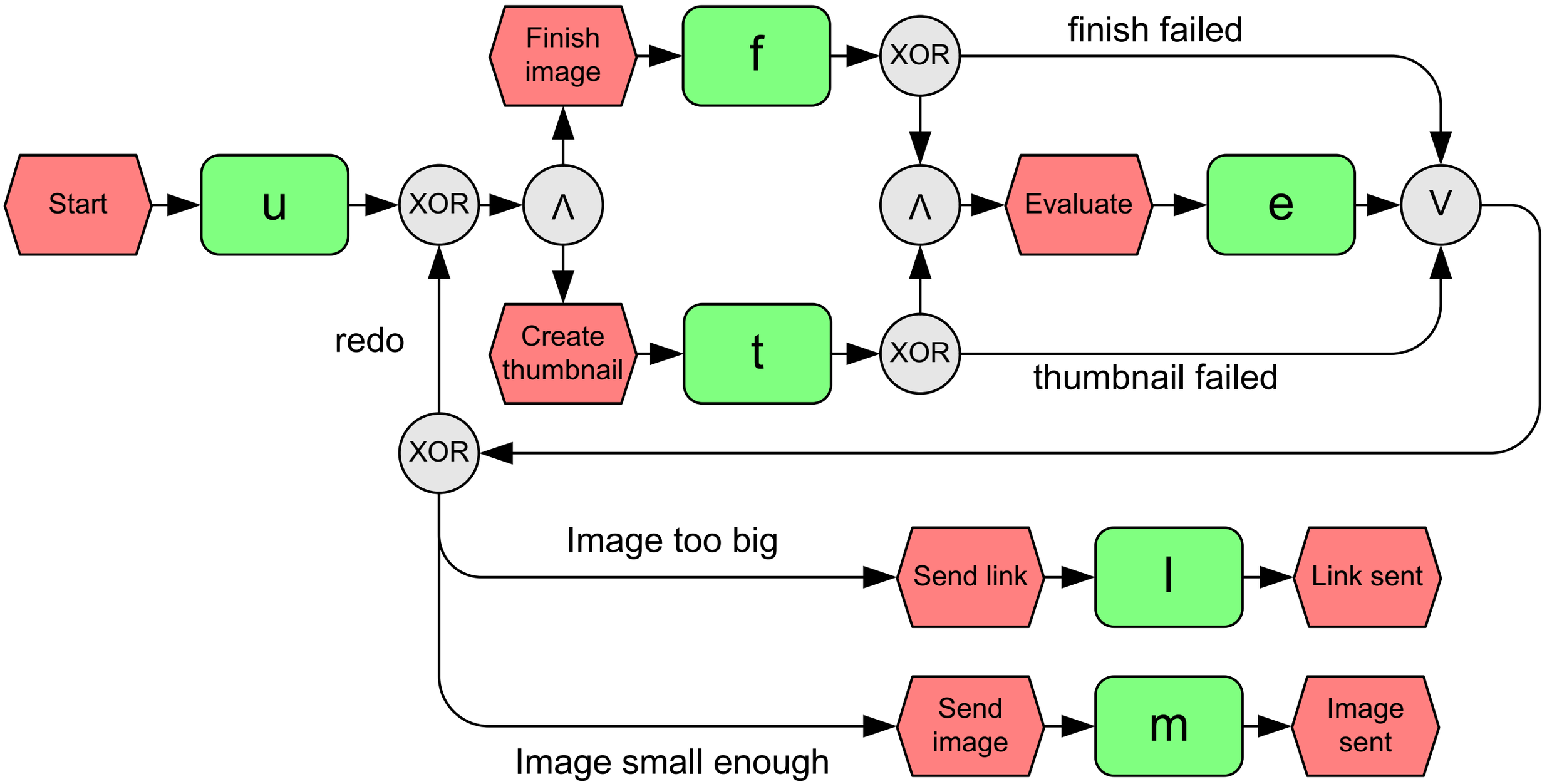
ORIGINAL PAPER

Jan Mendling · Markus Nüttgens

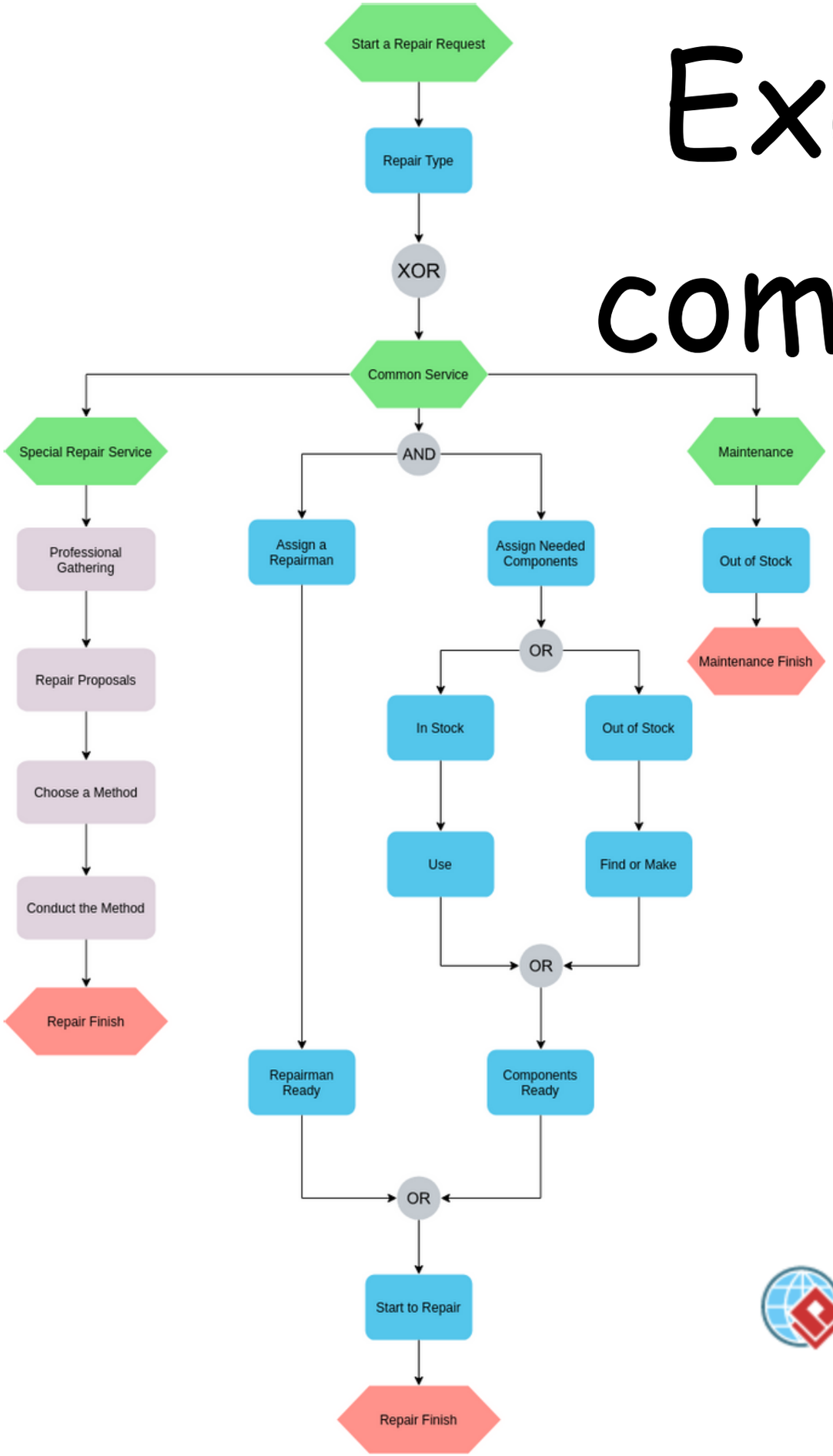
EPC markup language (EPML): an XML-based interchange format for event-driven process chains (EPC)

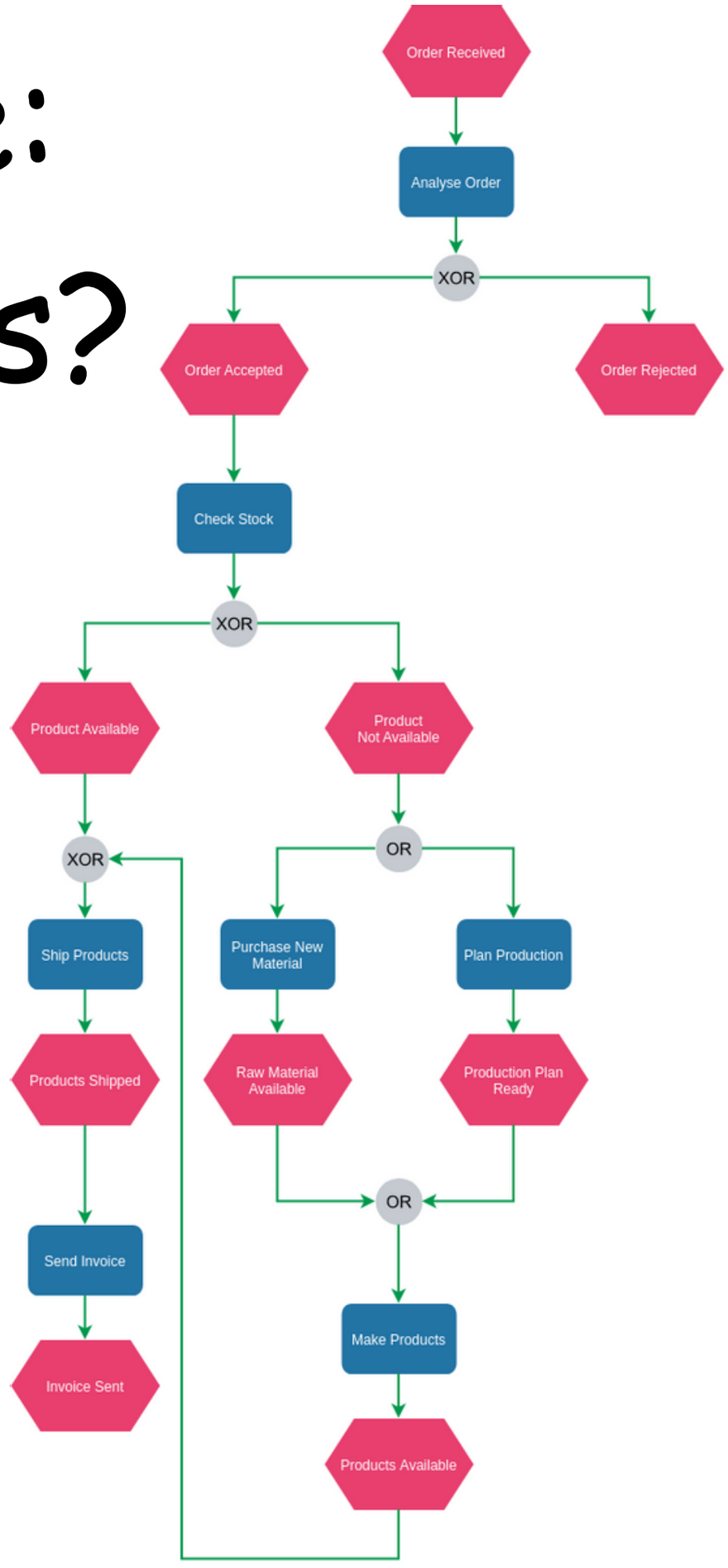
Published online: 22 October 2005
© Springer-Verlag 2005

Example: any comment?



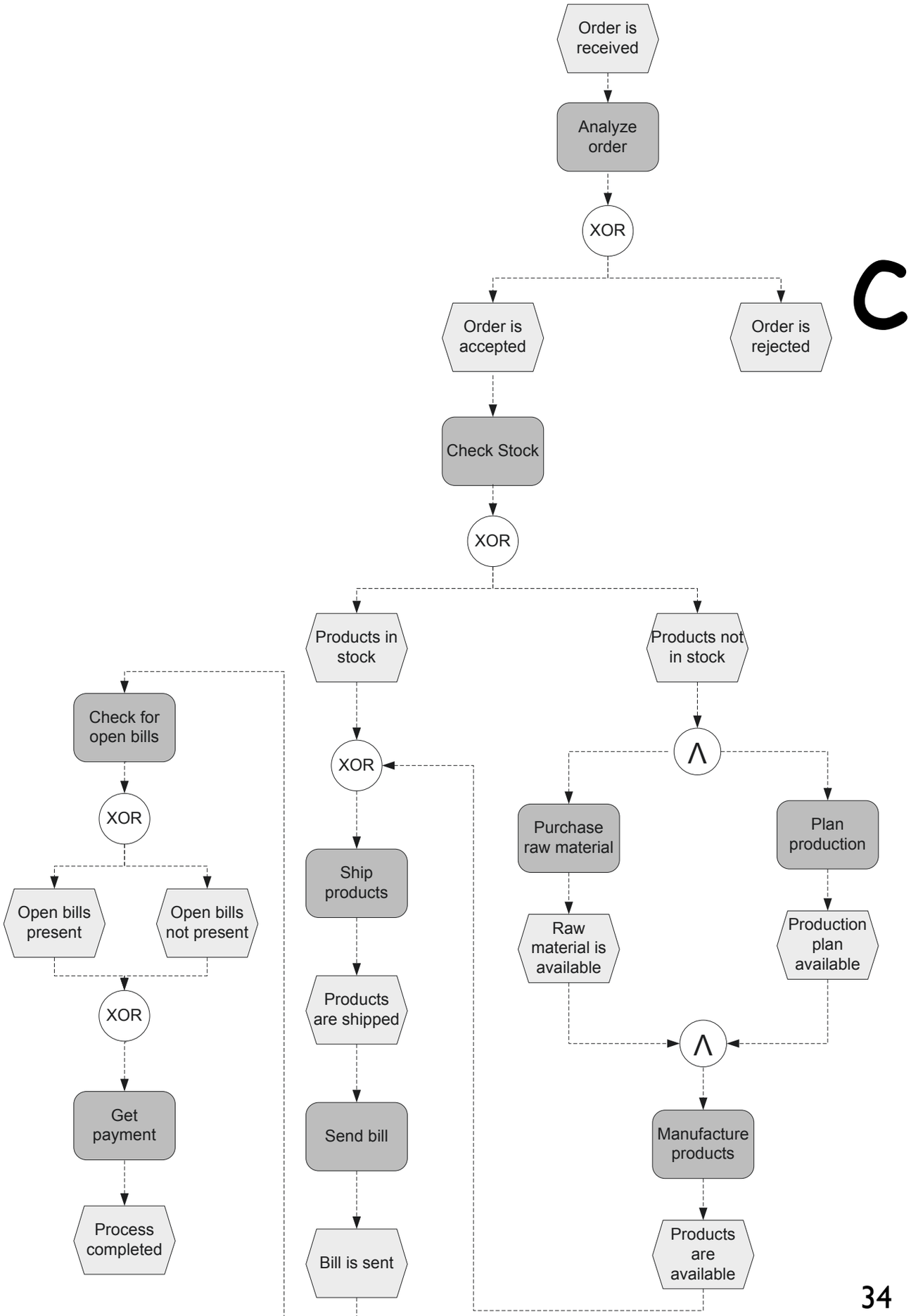
Example: comments?

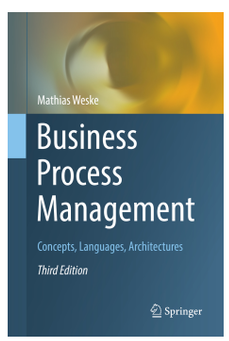




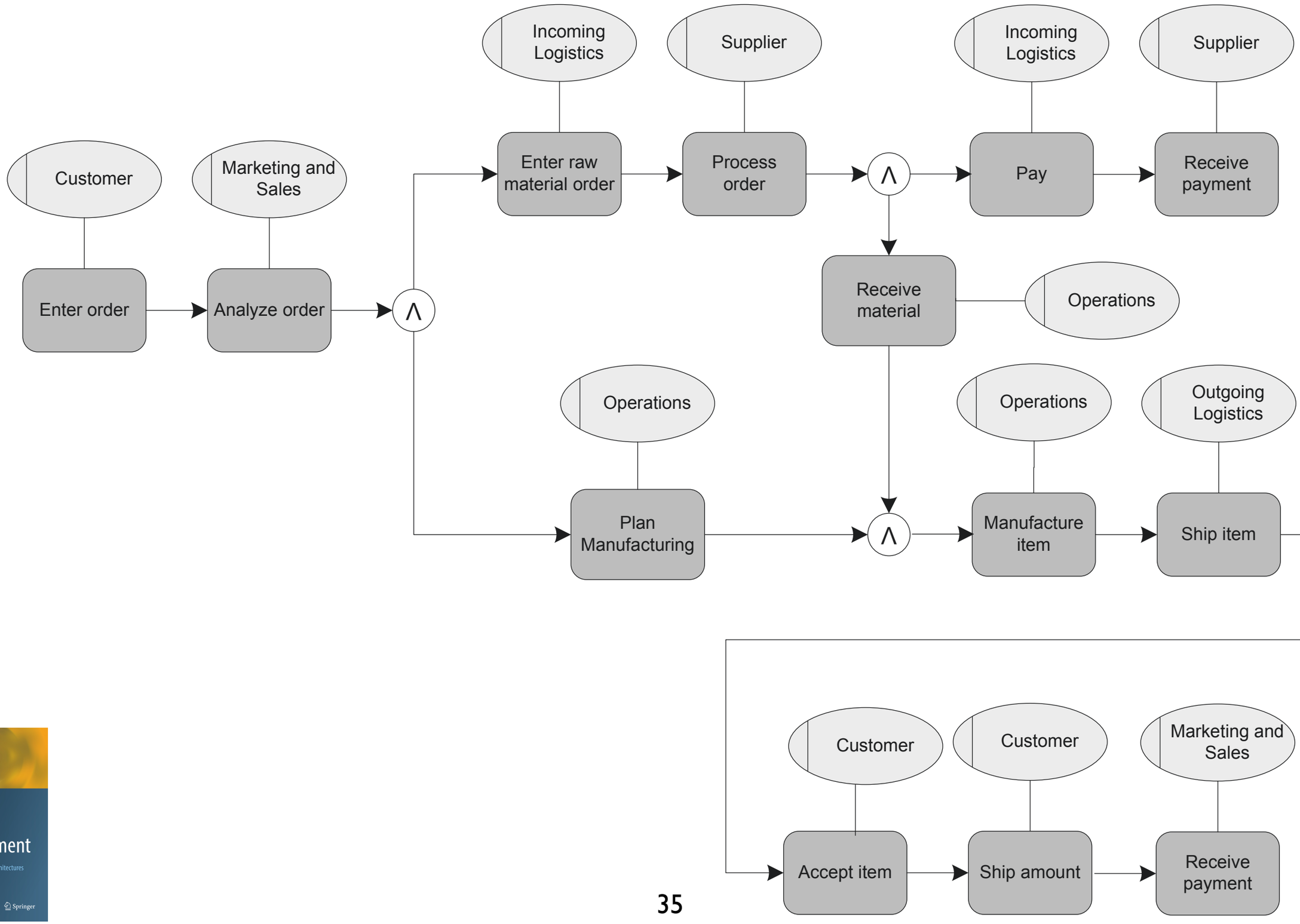
VisualParadigm Online

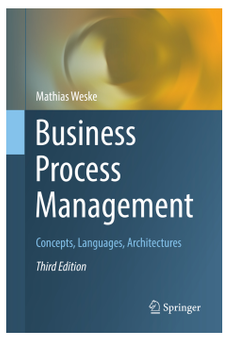
Example: comments?





Example: any comment?





EPC Semantics

EPC intuitive semantics

A process starts when some initial event(s) occurs

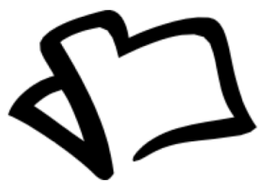
The activities are executed according to the constraints in the diagram

When the process is finished,
only final events have not been dealt with

If this is always the case, then the EPC is “correct”

Folder-passing semantics

The current state of the process is determined by placing folders over the diagram

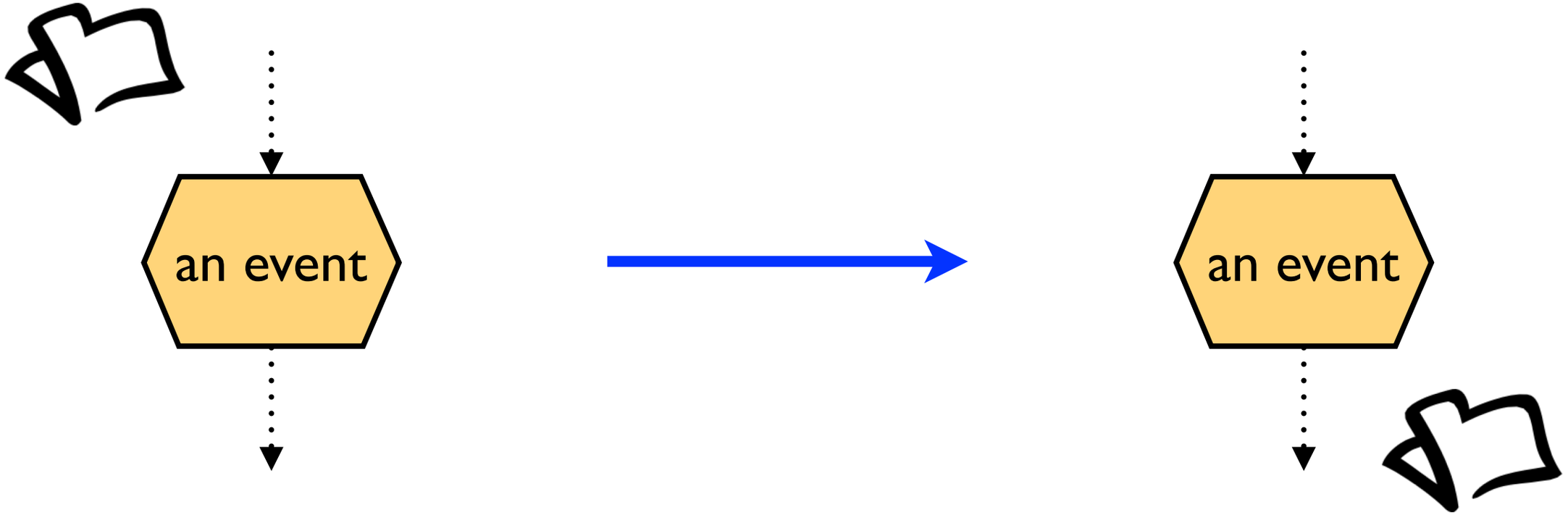


A transition relation explains how to move from one state to the next state

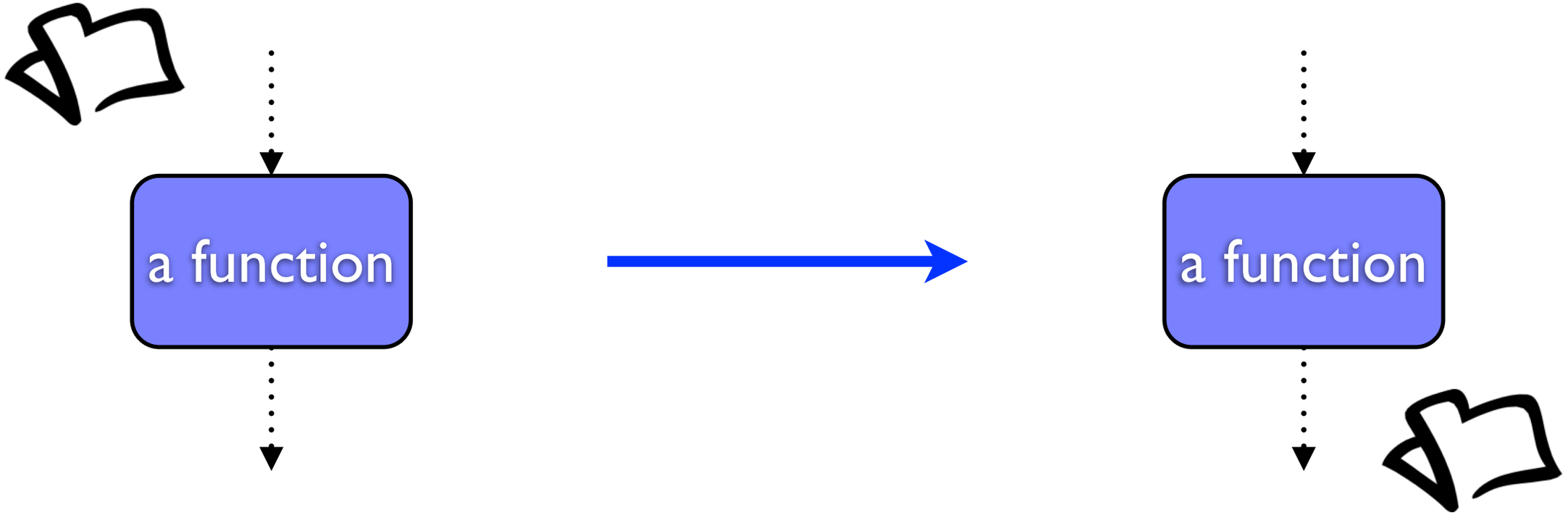


The transition relation is possibly nondeterministic

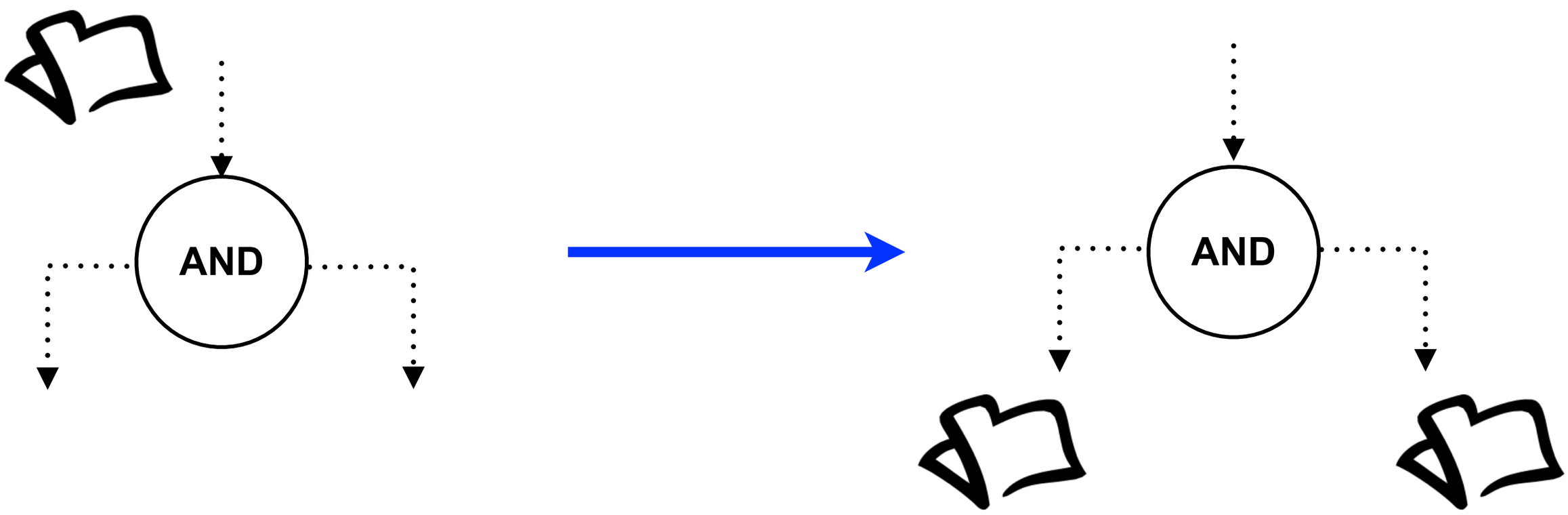
Folder-passing semantics: events



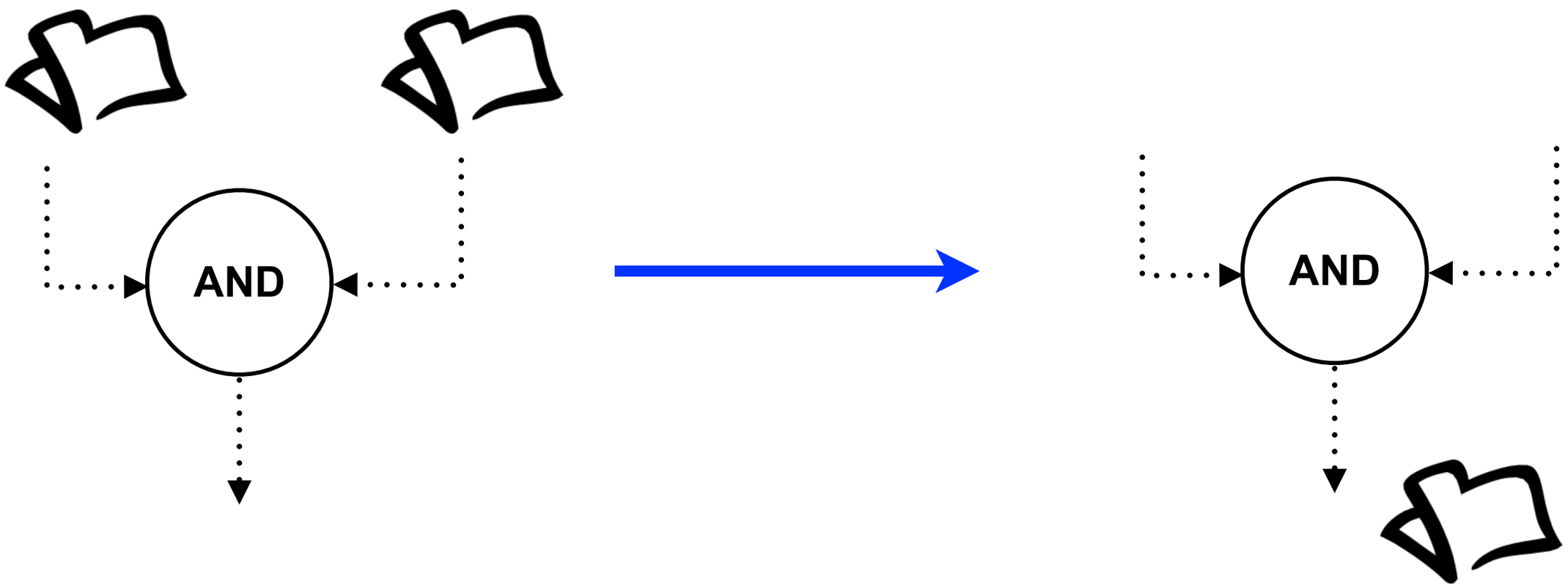
Folder-passing semantics: functions



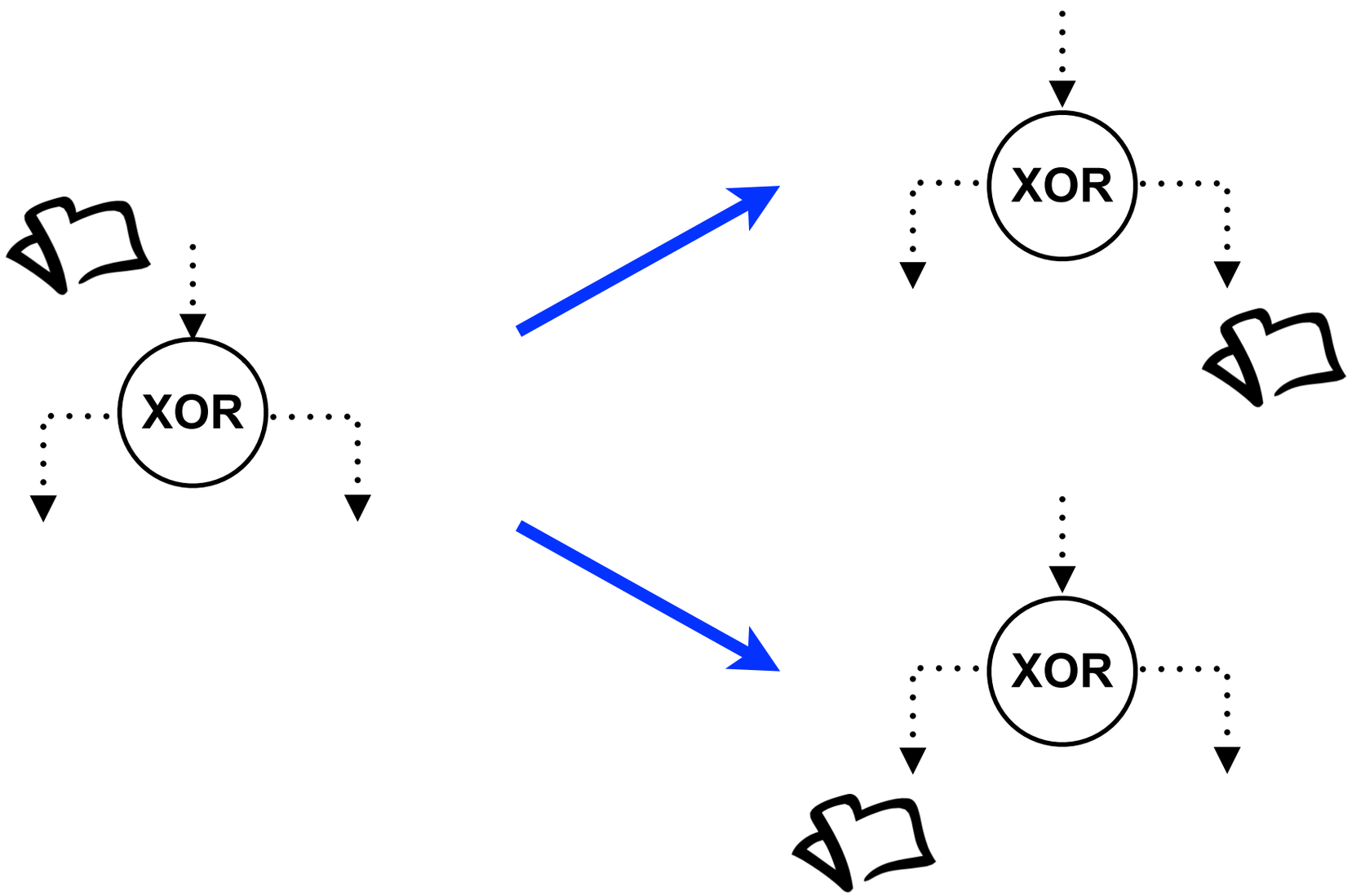
Folder-passing semantics: AND-split



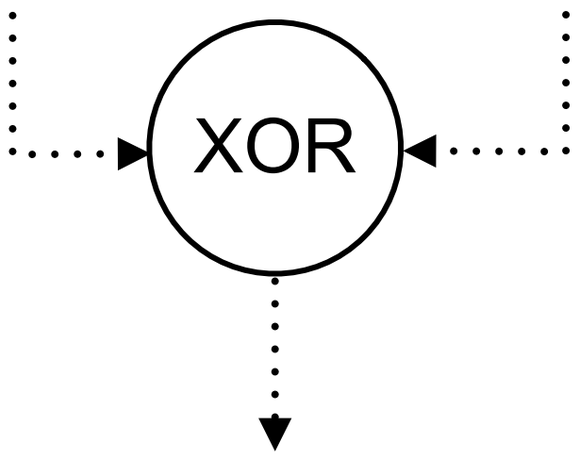
Folder-passing semantics: AND-join



Folder-passing semantics: XOR-split



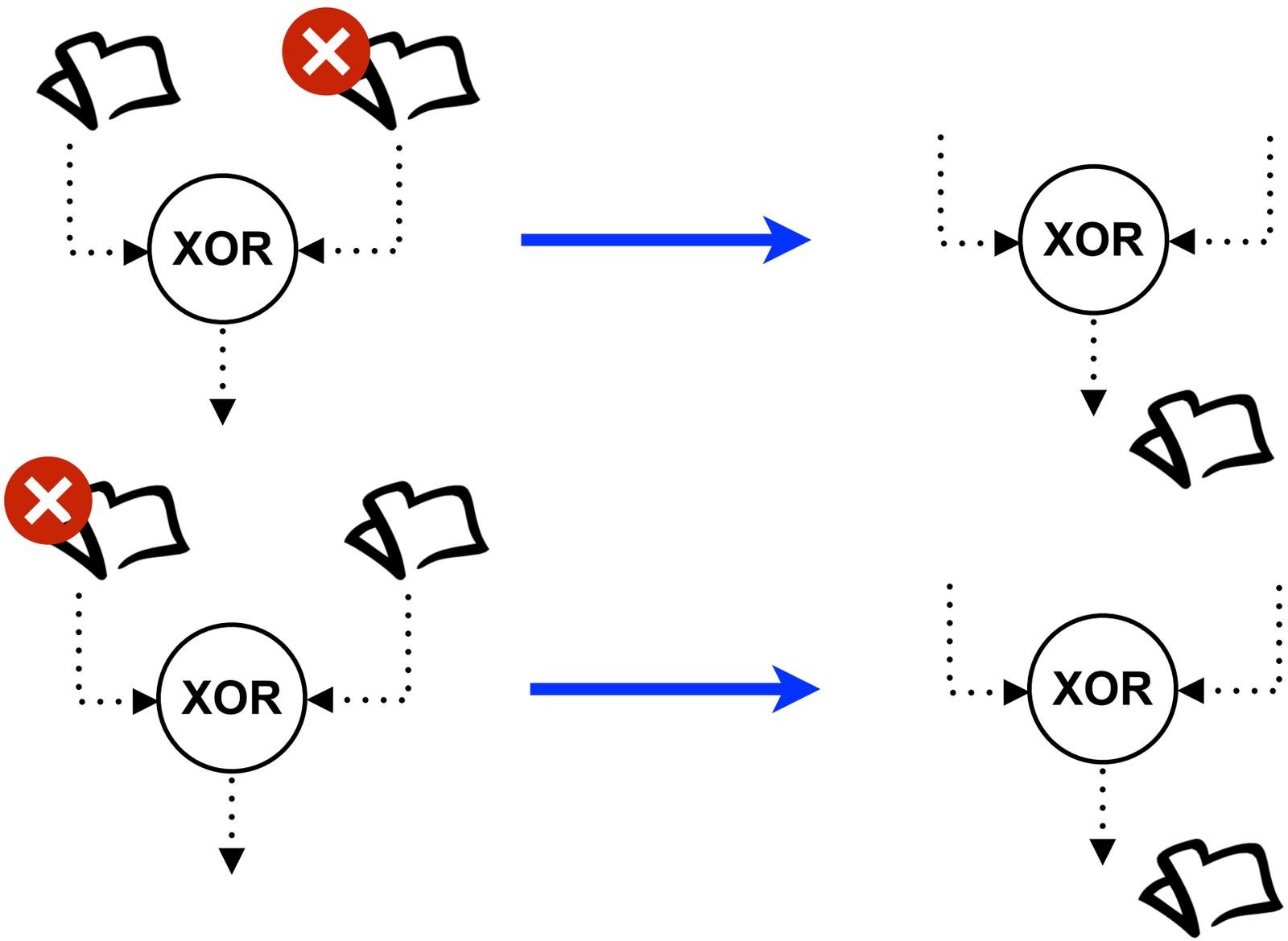
XOR join: intended meaning



if both inputs arrive,
it should block the flow

if one input arrives,
it cannot proceed unless
it is informed that
the other input will never arrive

Folder-passing semantics: XOR-join



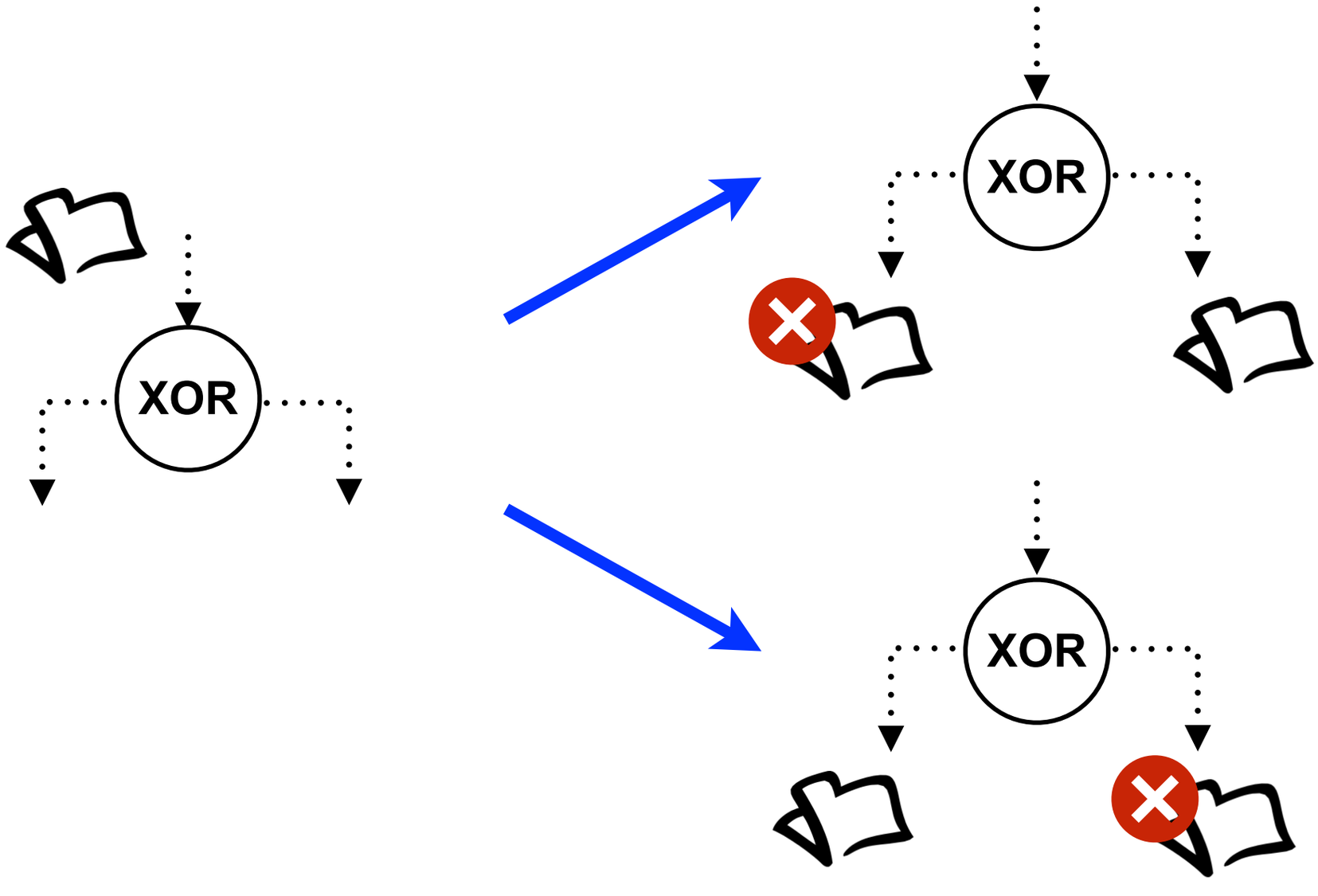
Folder-passing semantics?

How can we infer the absence of folders?

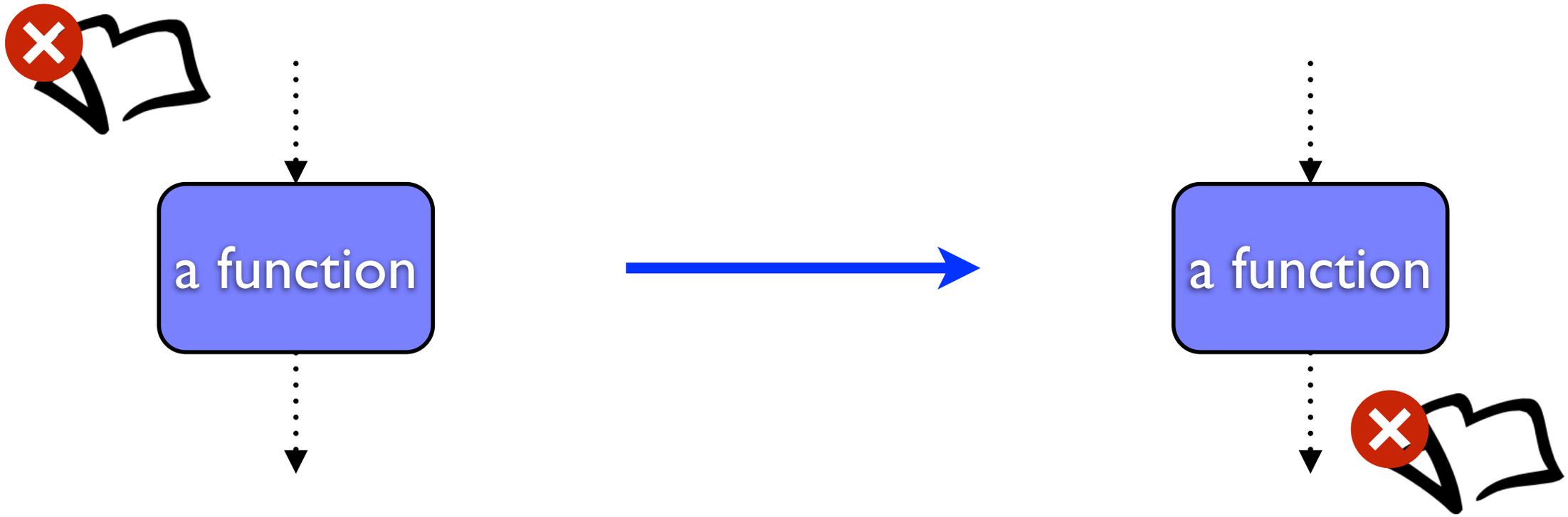


When and how should such information be propagated?

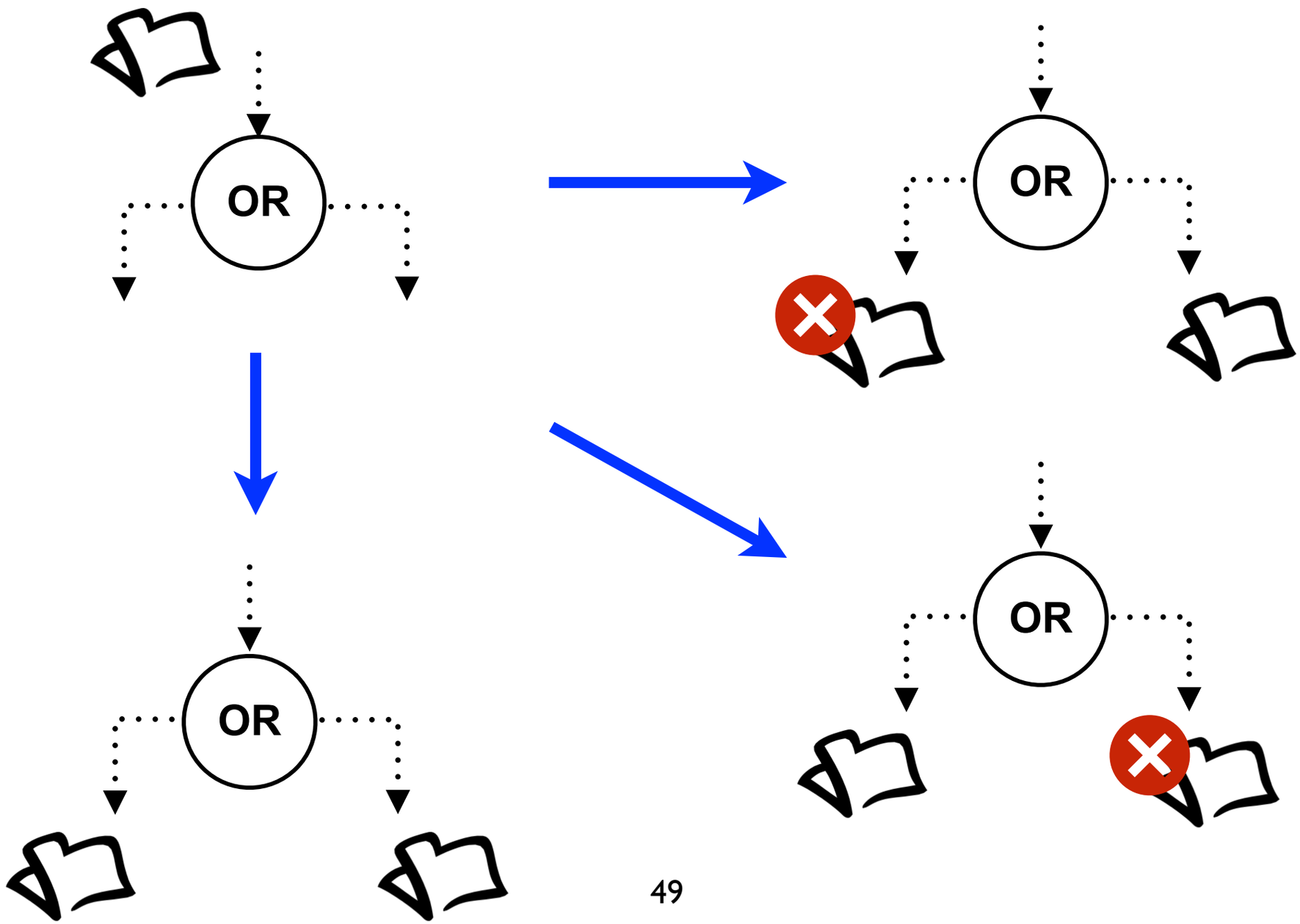
Absence of folders: creation



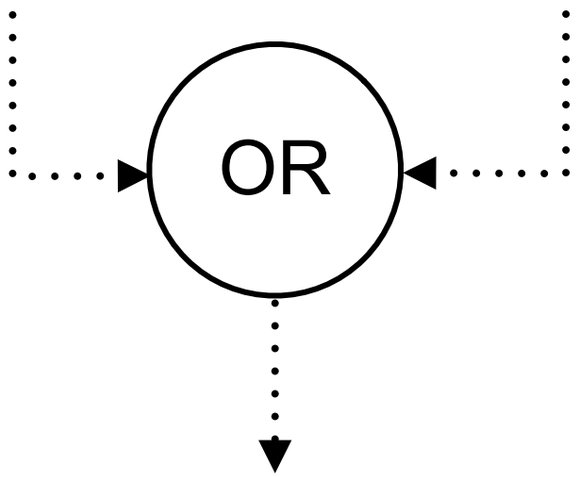
Absence of folders: propagation (example)



Folder-passing semantics: OR-split



OR join: intended meaning

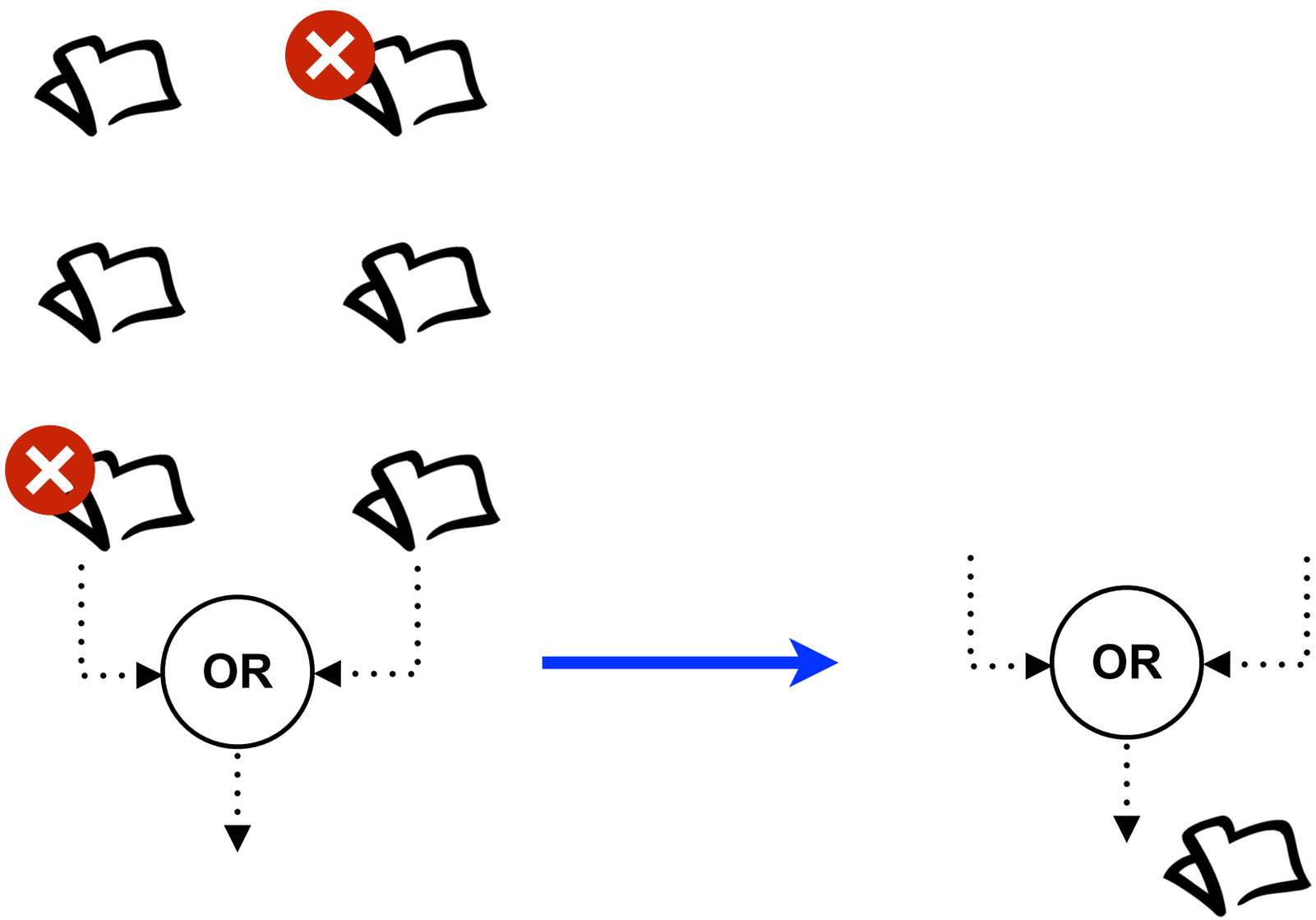


if only one input arrives,
it should release the flow

if both inputs arrive,
it should release only one output

if one input arrives,
it must wait until the other arrives or
it is guaranteed that the other will never arrive

Folder-passing semantics: OR-join?



Decorated EPC

To remove ambiguous behaviour for join connectors, designers can further annotate EPC diagrams

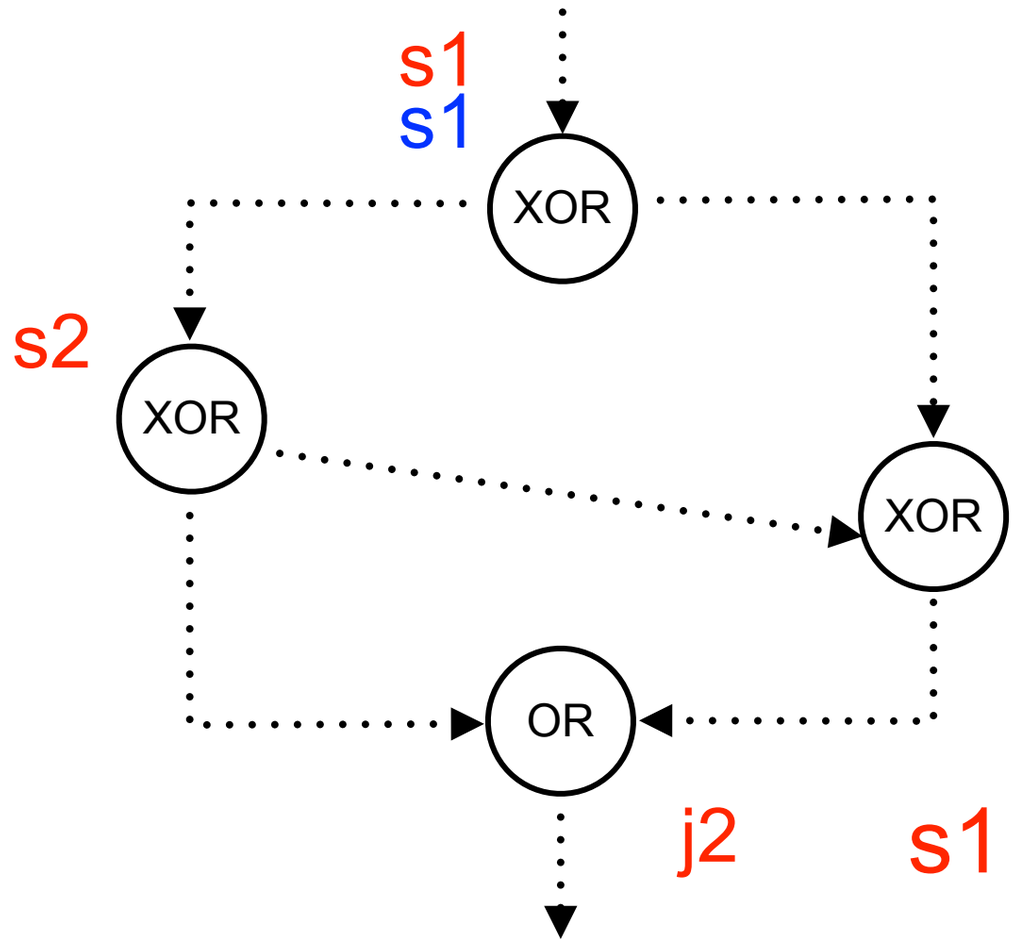
In particular we require to know:

corresponding split
which node separated the flows we are joining
(in the case of XOR/OR join)

applicable policy
how to trigger outgoing flow
(avoid OR join ambiguous behaviour)

Candidate split

A **candidate split** for a join node is any split node whose outputs are connected to the inputs of the join

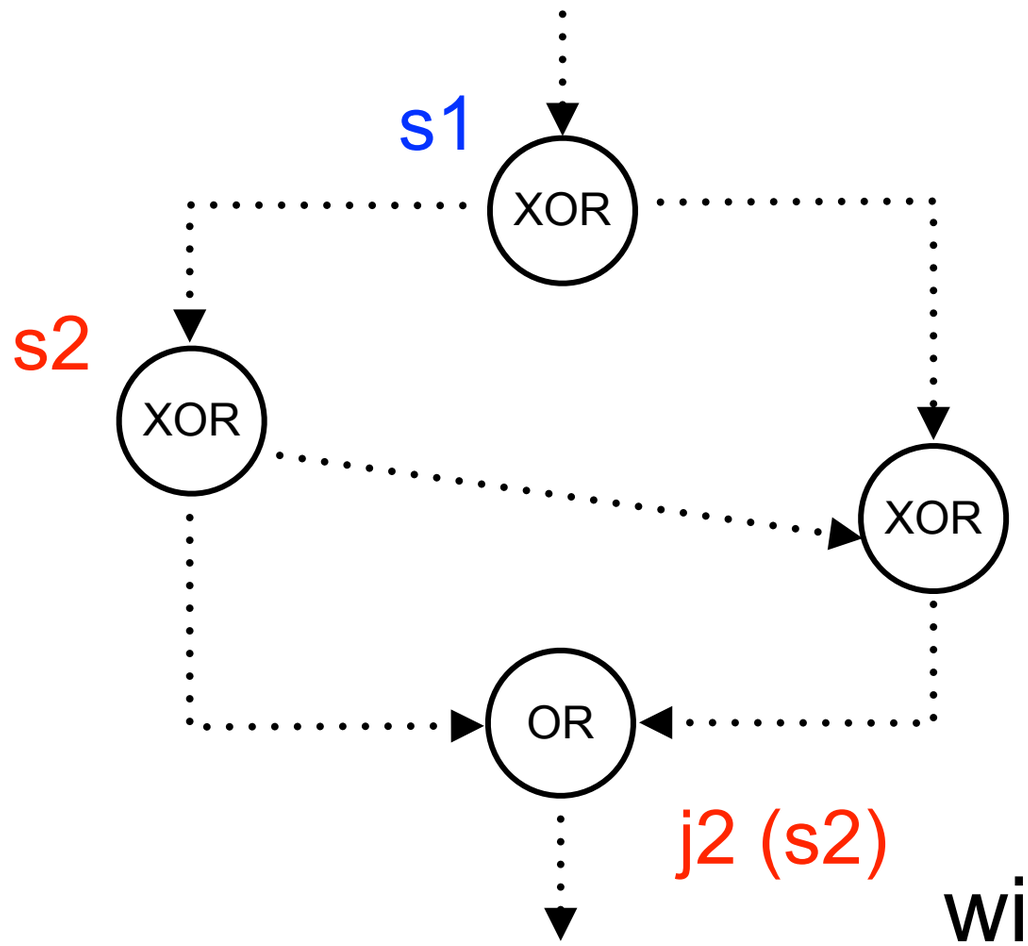


j1 s1 is a candidate split for j1

j2 s1 and s2 are candidate splits for j2

Corresponding split

A **corresponding split** for a join node
is a chosen candidate split



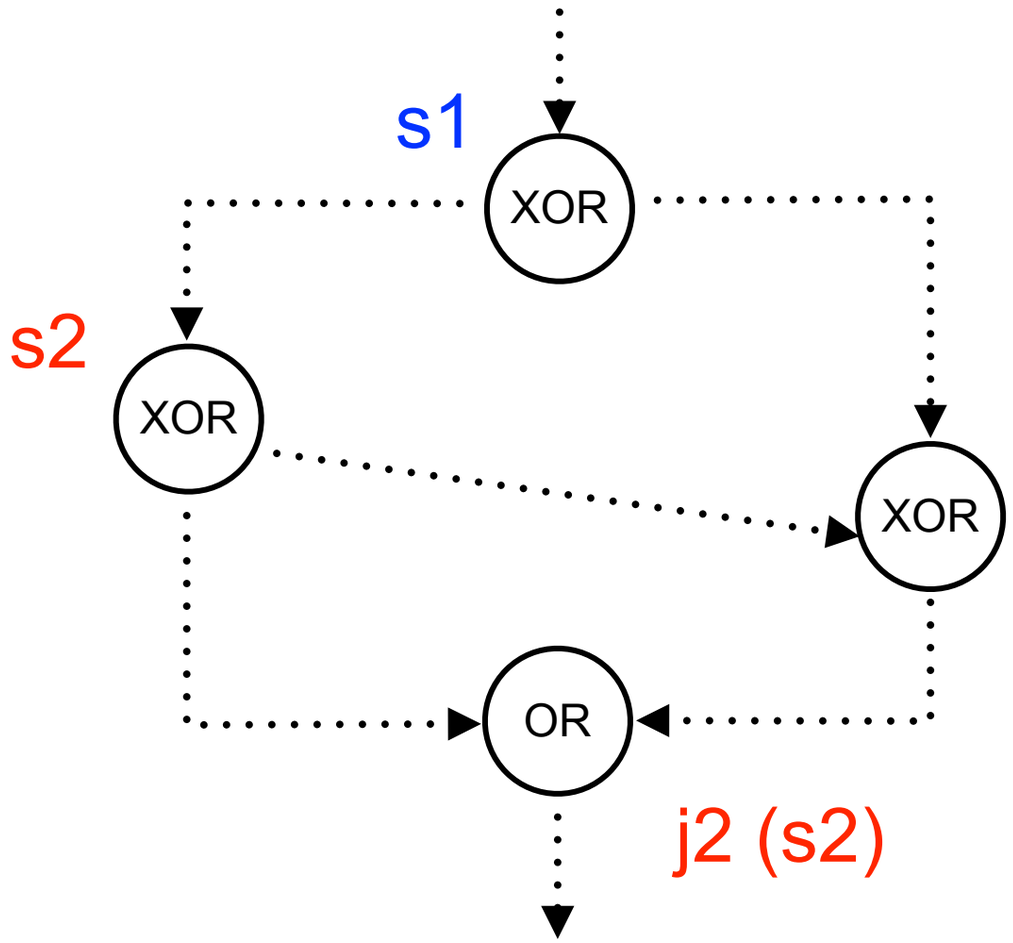
we choose s1 as a
corresponding split for j1

we choose s2 as a
corresponding split for j2

(we tag each join
with its corresponding split)

Matching split

A corresponding split for a join node is called **matching** if it has the same type as the join node



s1 is a matching split for j1

s2 is not a matching split for j2

OR join: policies

If an OR join has a **matching split**, its semantics is **wait-for-all**: wait for the completion of all *activated* paths

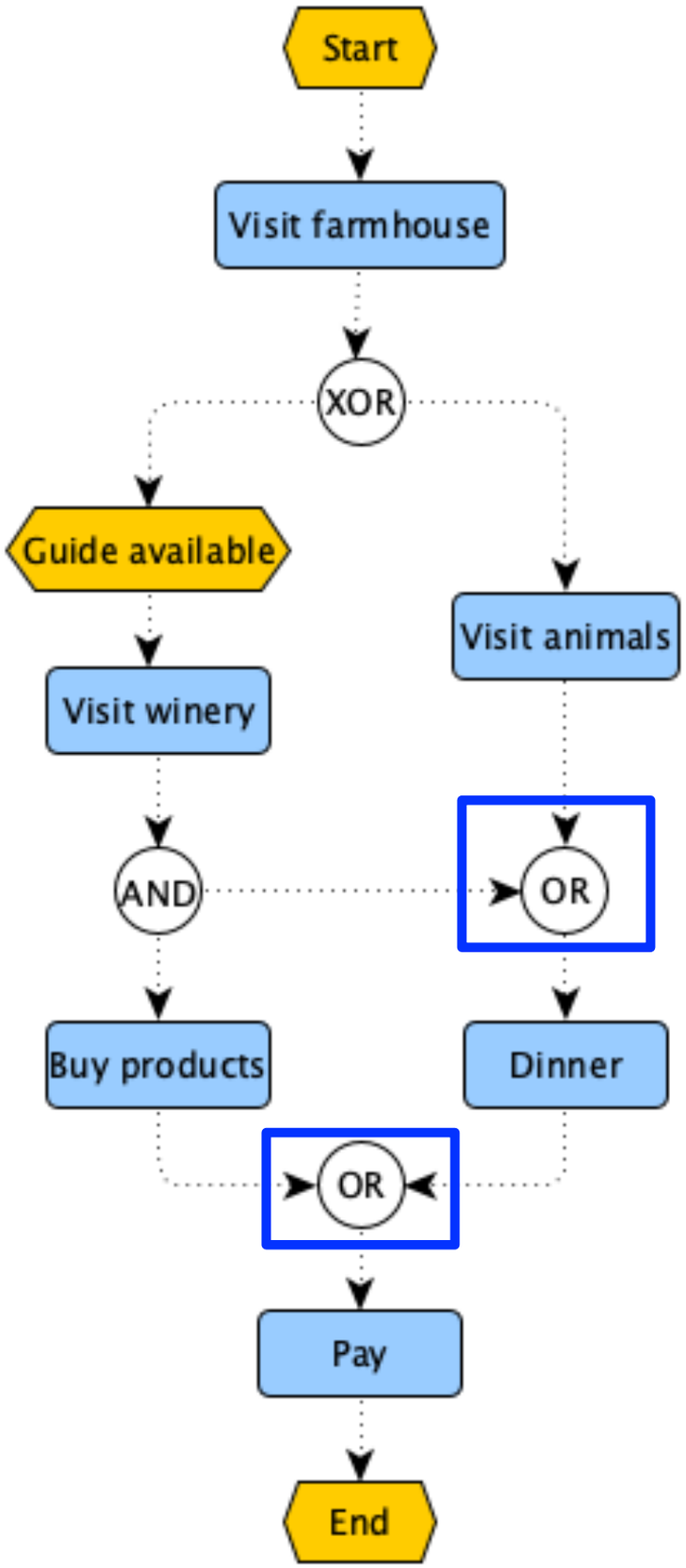
Otherwise, also other policies can be chosen:

every-time: trigger the outgoing path on each input

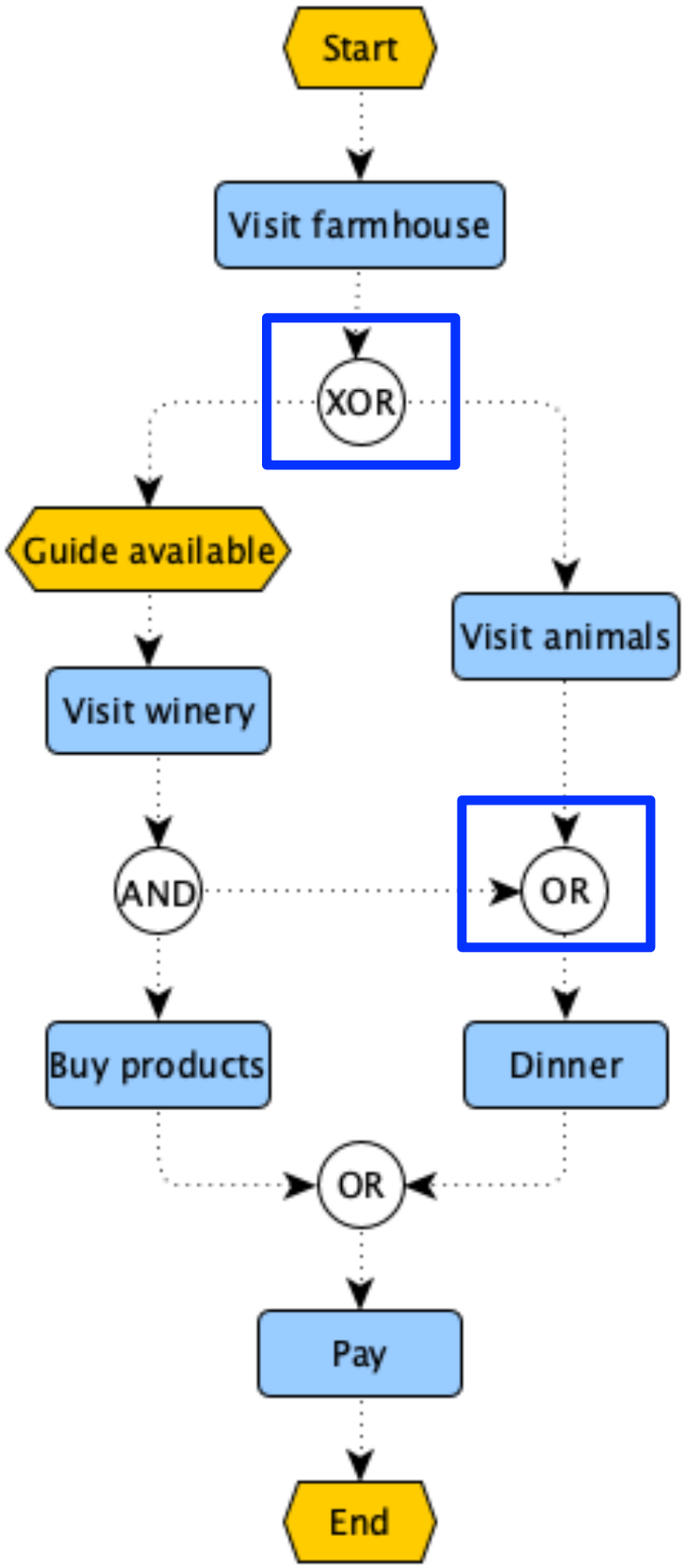
first-come: wait for the first input and ignore the second

Assumption: every OR join is tagged with a policy (some suggested to have different trapezoid symbols)

Example



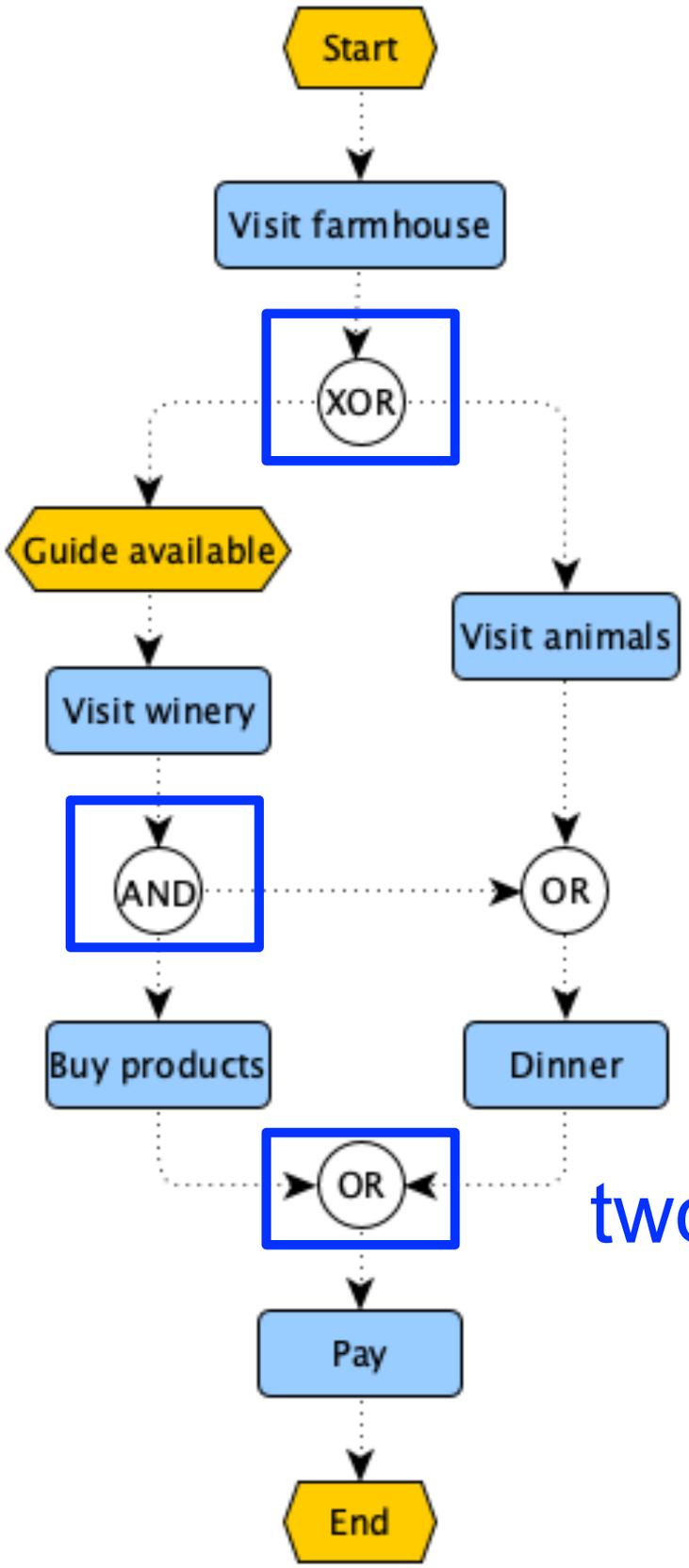
two OR joins
but no OR split



Example

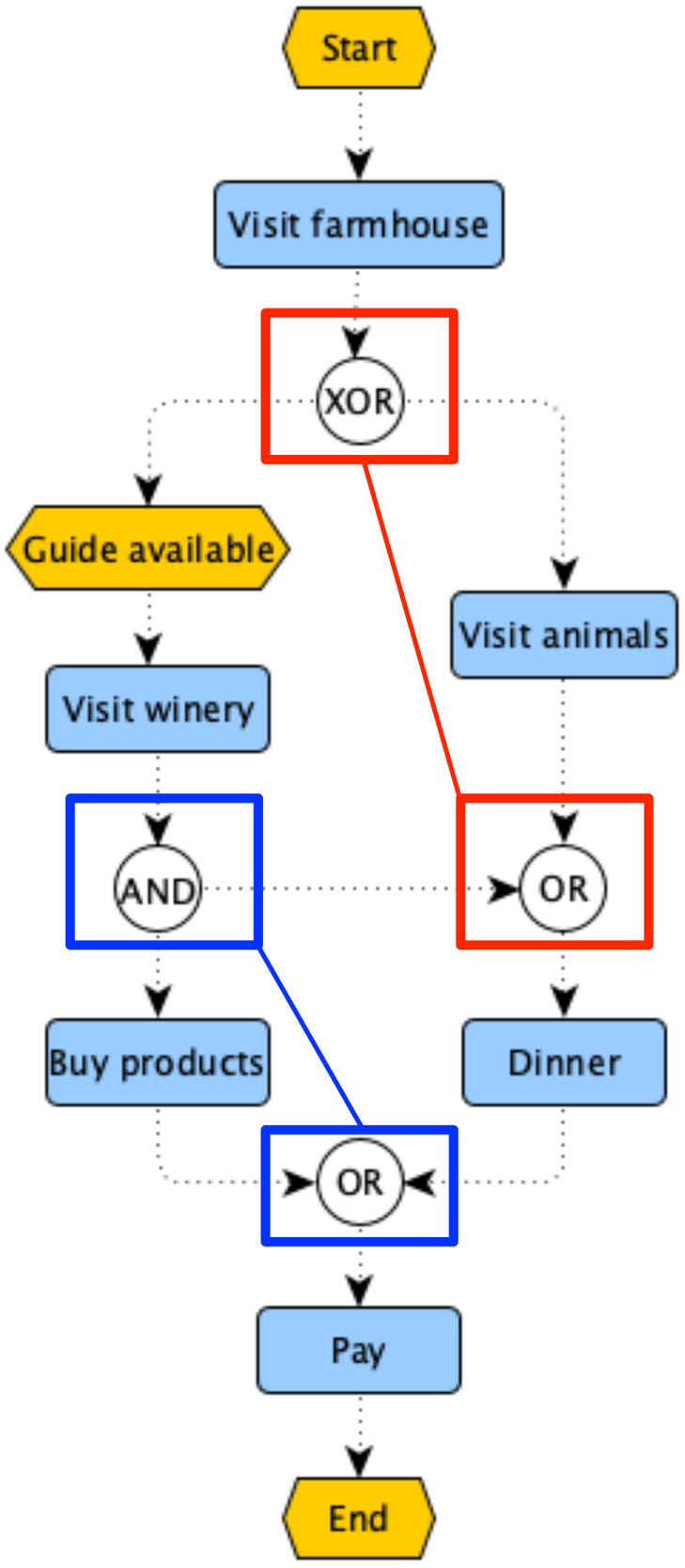
only one
candidate split

Example



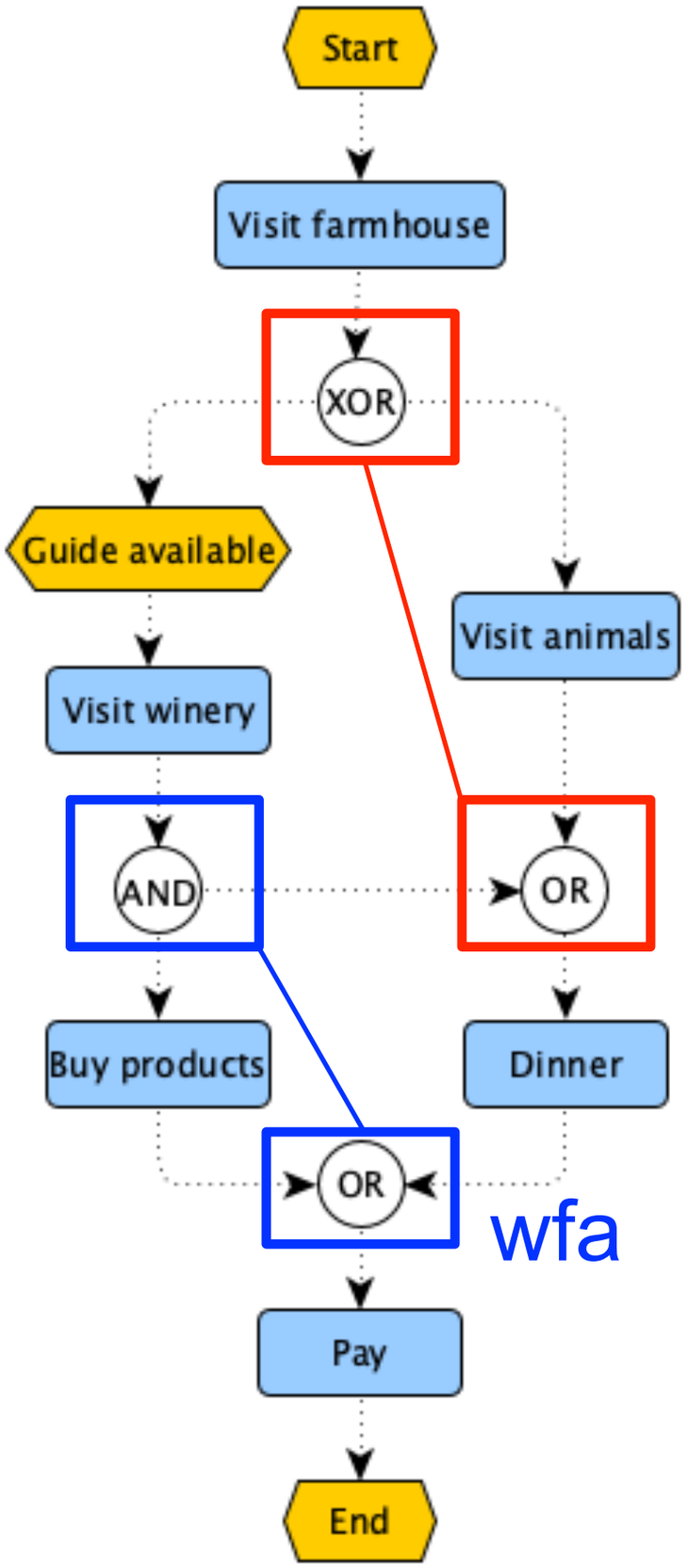
two candidate splits

Example



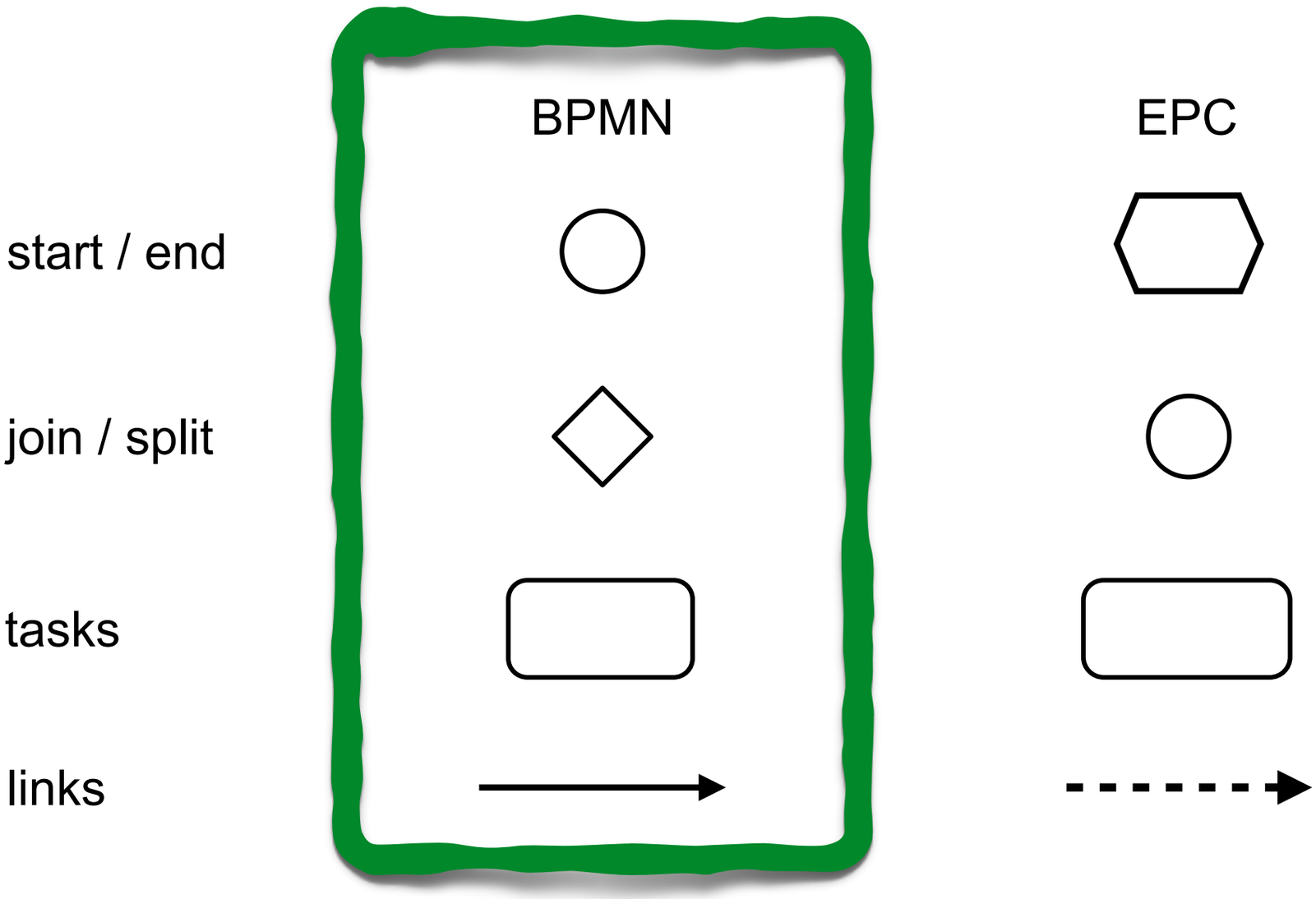
assign corresponding splits

Example



assign policies

A closer look at standards: EPC and BPMN



BPMN

Main goal:

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

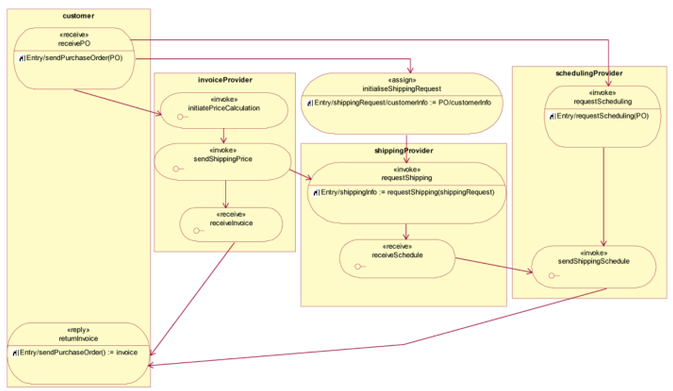
by **business analysts** (initial drafts of processes)

by **technical developers** (process implementation)

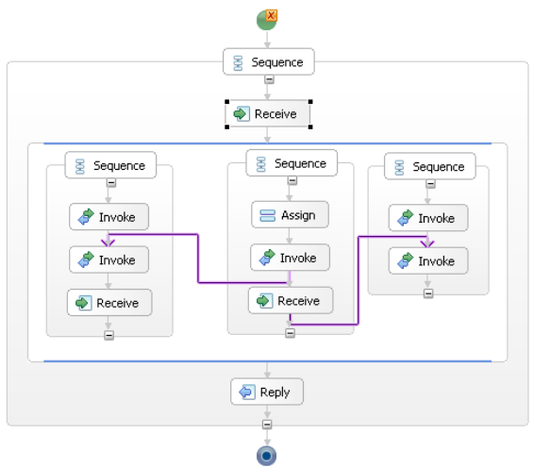
by **business people** (process management)

Before BPMN

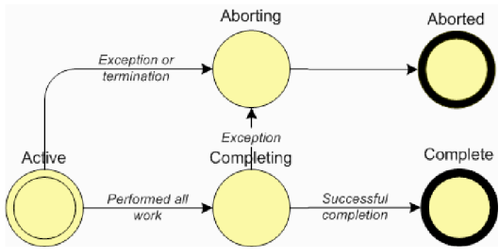
BPMD
BP Definition Metamodel



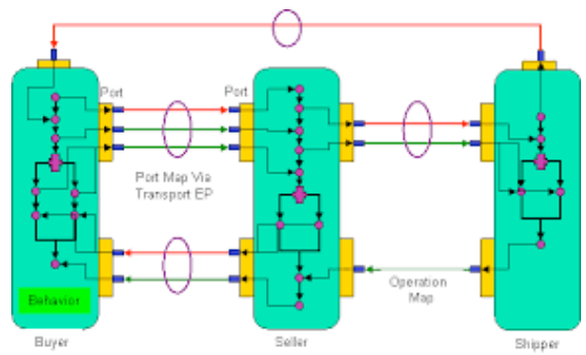
OASIS's BPEL
BP Execution Language



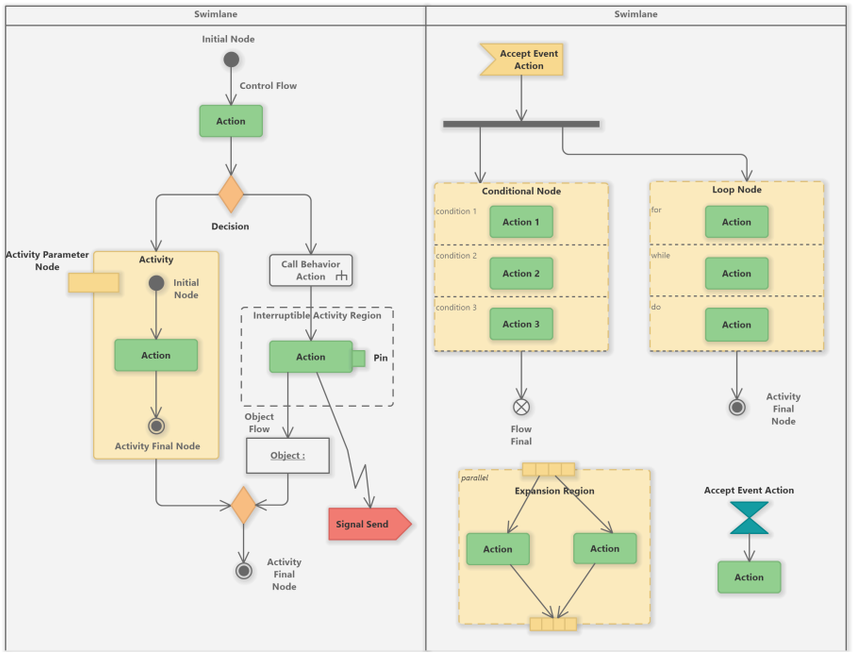
BPMI.org's BPML
BP Modelling Language



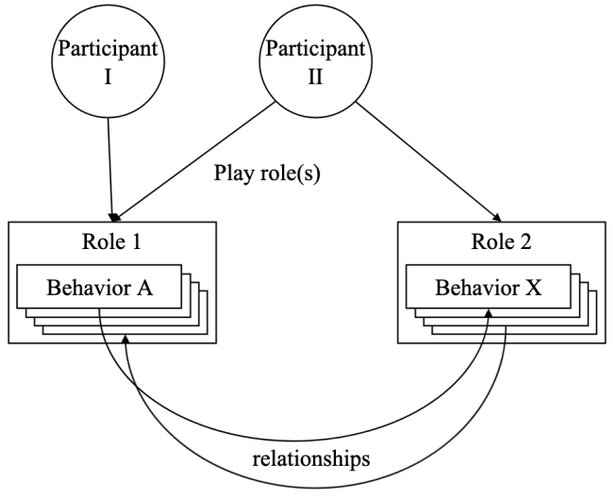
Microsoft's XLANG



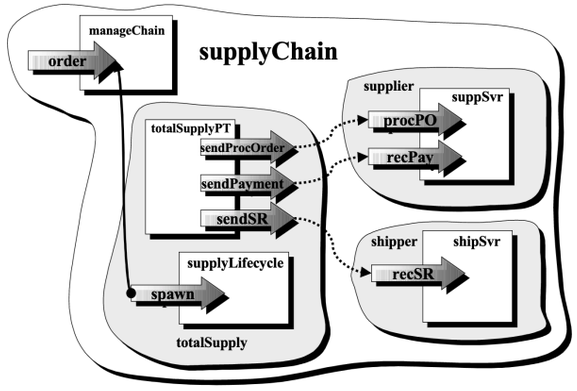
UML2 AD
Activity Diagram



W3C's WS CDL
Choreography Description Language



IBM's WSFL
WS Flow Language



Standardisation

In the context of graphical models for business processes

the development of BMPN 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

A joint effort!



Copyright © 2000-2001, BPMI.org

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**)

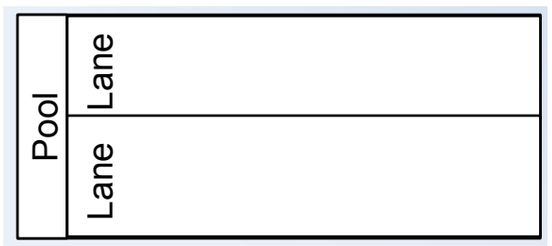
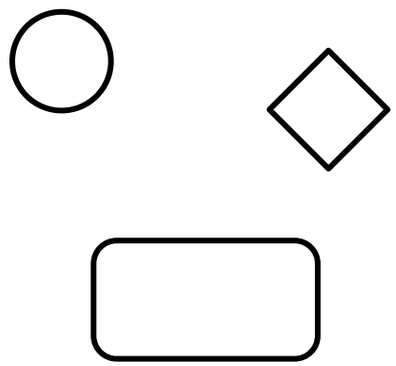
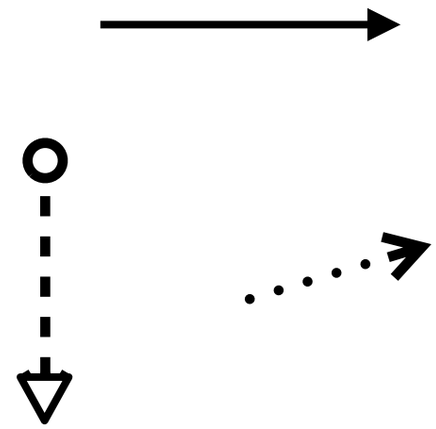
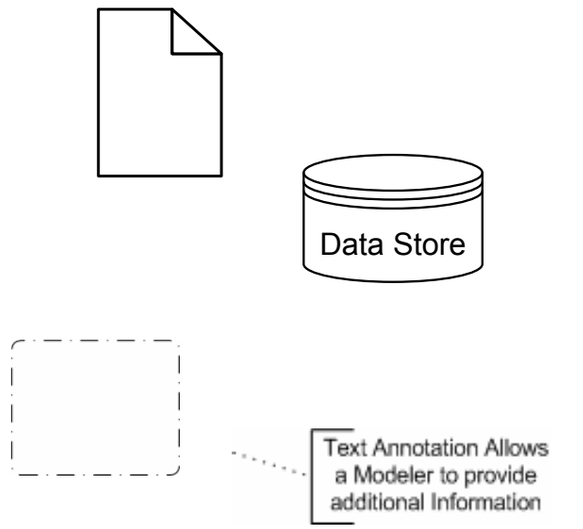
2006 - **BPMN 1.0 approved**
2007 - BPMN 1.1 approved
2009 - BPMN 1.2 approved

2009 - BPMN 2.0 Beta 1 proposed
2010 - BPMN 2.0 Beta 2 proposed
2011 **- BPMN 2.0 Final delivered**

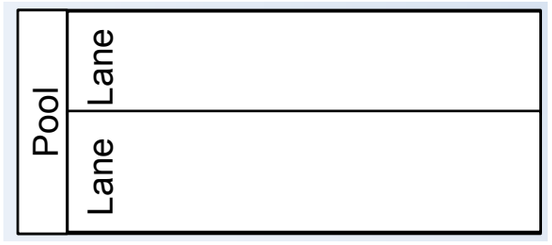
Business process diagrams

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

Four categories of elements

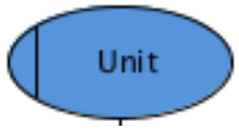
swimlanes	flow objects	connecting objects	artefacts
			

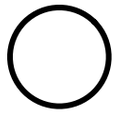
BPMN vs EPC (roughly)



swimlanes

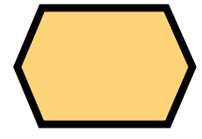
organization unit





event

event

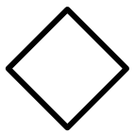




activity

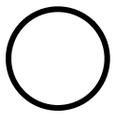
function





gateway

connector





sequence flow

control flow

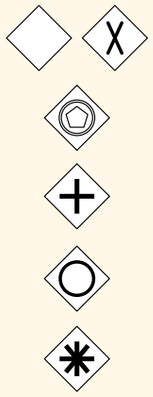




message flow

BPMN - Business Process Modeling Notation

Gateways



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.

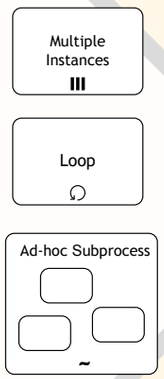
Event-based Exclusive Gateway
Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

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 based on branching conditions. When merging, it awaits all active incoming branches to complete.

Complex Gateway
It triggers one or more branches based on complex conditions or verbal descriptions. Use it sparingly as the semantics might not be clear.

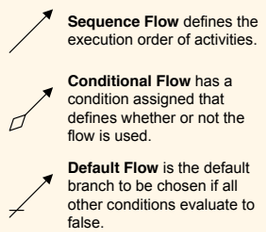
Activities



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.

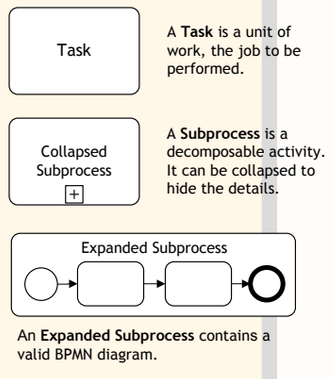
Ad-hoc Subprocesses contain tasks only. Each task can be executed arbitrarily often until a completion condition is fulfilled.



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.

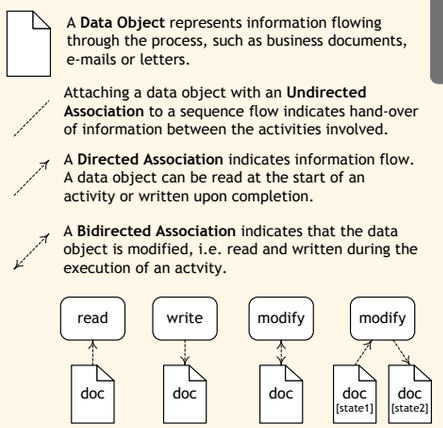


A **Task** is a unit of work, the job to be performed.

A **Subprocess** is a decomposable activity. It can be collapsed to hide the details.

An **Expanded Subprocess** contains a valid BPMN diagram.

Data



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

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.

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.
Compen-sation		◀◀	◀◀	◀◀		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.

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.

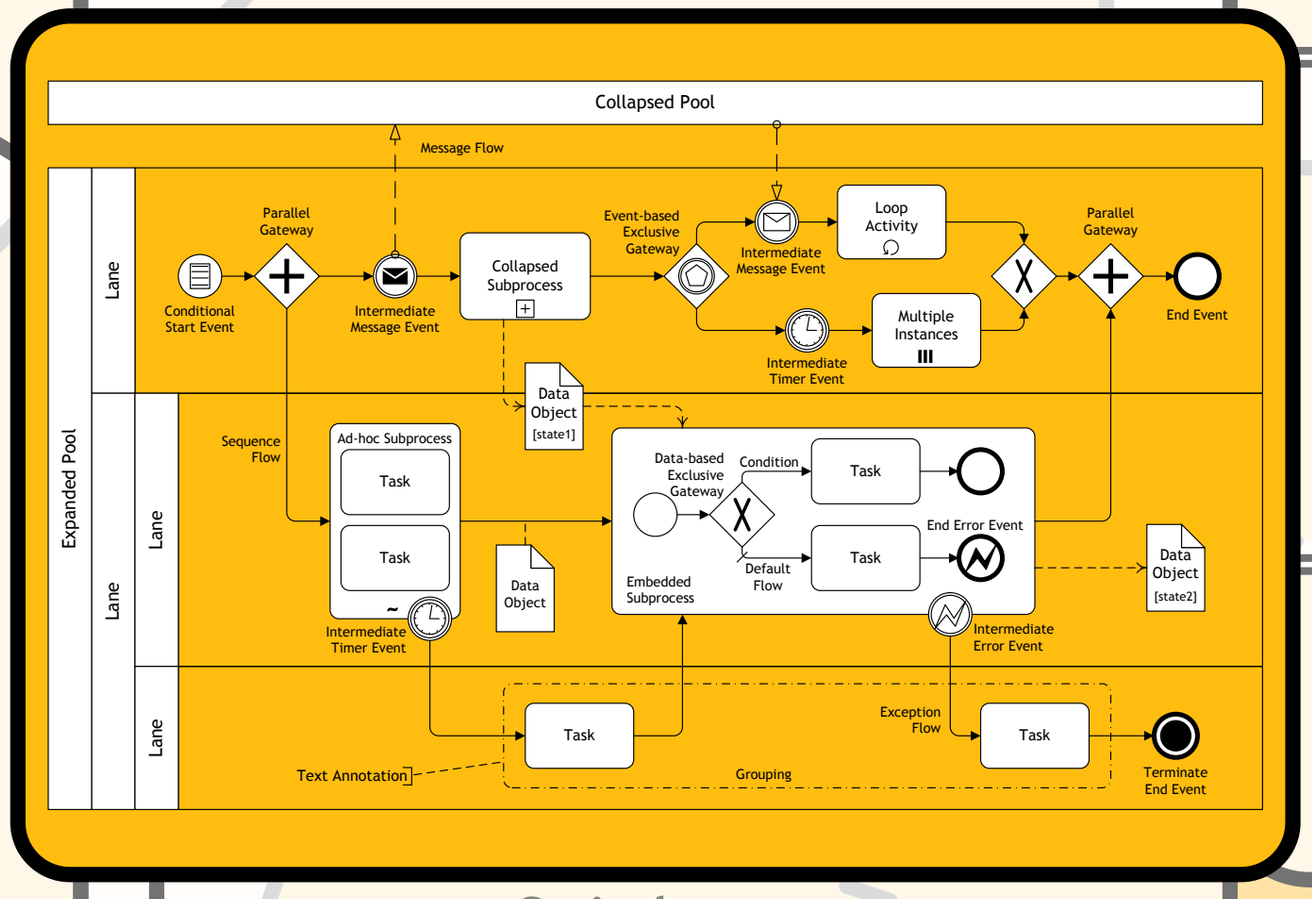
Throwing

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

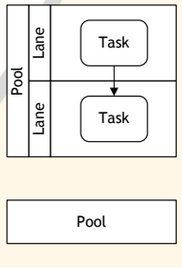
Intermediate Event (throwing): An event is thrown and the process continues.



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

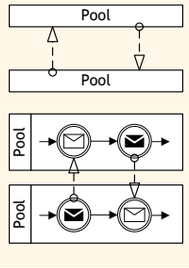


Swimlanes



Pools and **Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes sub-divide pools or other lanes hierarchically.

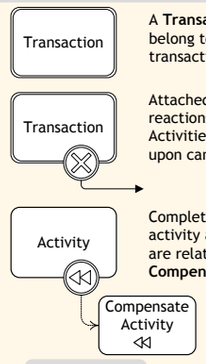
Collapsed Pools hide all internals of the contained processes.



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.

Transactions

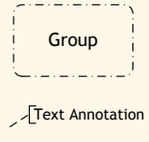


A **Transaction** is a set of activities that logically belong together; it might follow a specified transaction protocol.

Attached **Intermediate Cancel Events** indicate reactions to the cancellation of a transaction. Activities inside the transaction are compensated upon cancellation.

Completed activities can be compensated. An activity and the corresponding **Compensate Activity** are related using an attached **Intermediate Compensation Event**.

Documentation



An arbitrary set of objects can be defined as a **Group** to show that they logically belong together.

Any object can be associated with a **Text Annotation** to provide additional documentation.

Business **P**rocess **T**echnology
Prof. Dr. Mathias Weske

Web: bpt.hpi.uni-potsdam.de
Oryx: oryx-project.org
Blog: bpmn.info
BPMN Version 1.2

Authors

Gero Decker
Alexander Grosskopf
Sven Wagner-Boysen

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

ORYX

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)

BPMN 2.0 - Business Process Model and Notation

http://bpmb.de/poster

Activities

Task

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.

Transaction

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

Event Sub-Process

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.

Call Activity

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.

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.

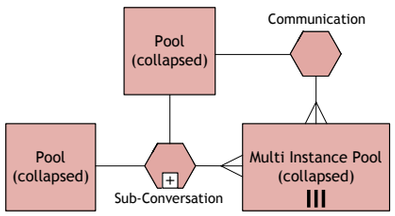
Conversations

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

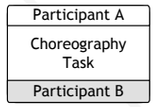
A **Conversation Link** connects Communications and Participants.

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

Conversation Diagram

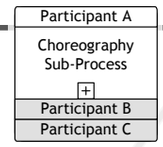


Choreographies



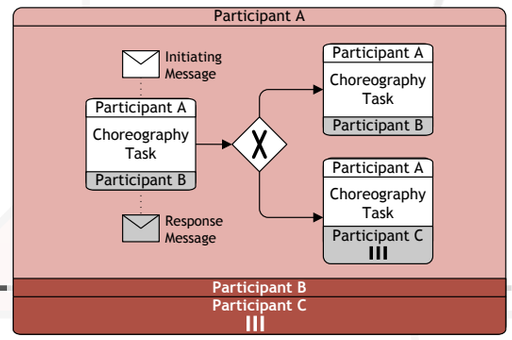
A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

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

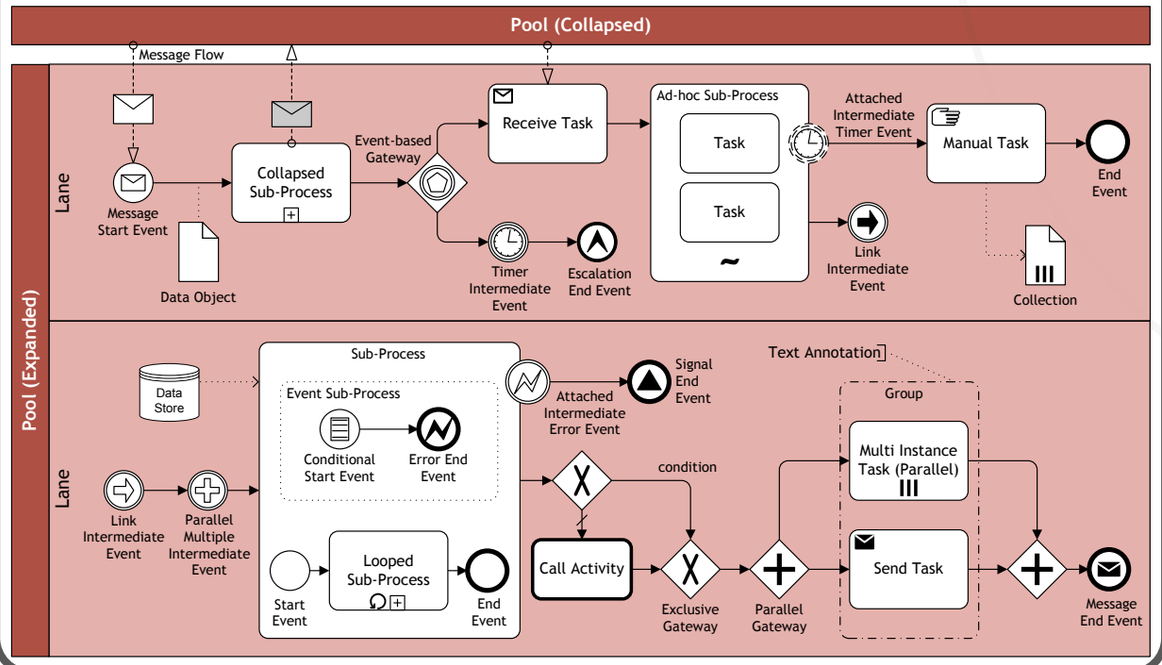


A **Choreography Sub-Process** contains a refined choreography with several Interactions.

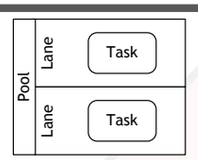
Choreography Diagram



Collaboration Diagram

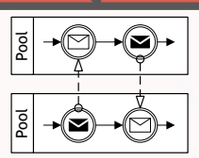


Swimlanes



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.

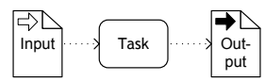


The order of message exchanges can be specified by combining message flow and sequence flow.

Events

	Start: Top-Level	Start: Event Sub-Process Interrupting	Start: Event Sub-Process Non-Interrupting	Intermediate: Catching	Intermediate: Boundary Interrupting	Intermediate: Boundary Non-Interrupting	Intermediate: Throwing	End
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.

BPM Offensive Berlin

HPI Hasso Plattner Institut

camunda the business process company

inubit integrating your business and IT

SIGNAVIO simply professional

BPMN 2.0 - Business Process Model and Notation

http://bpmb.de/poster

Tradotto da: dexea

Attività

Task

Un ***task*** è un unità di lavoro, cioè il lavoro da svolgere. Quando si annota con il simbolo [+] indica un sottoprocesso, cioè un'attività che può essere perfezionata.

Transazione

Una **transazione** è un insieme di attività che si legano logicamente; essa potrebbe seguire uno specifico protocollo.

Sottoprocesso basato su eventi

Un **sottoprocesso basato su eventi** si trova all'interno di un processo o sottoprocesso. Si avvia quando il suo evento di inizio viene attivato e può interrompere il processo di livello superiore oppure eseguire in parallelo (senza interruzioni) in base all'evento di inizio.

Call Activity

Una ***call activity*** è un contenitore di un sottoprocesso definito globalmente o un task che può essere riusato nel processo attuale.

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

definisce l'ordine di esecuzione delle attività.

Flusso predefinito

è il ramo predefinito da scegliere se tutte le altre condizioni vengono valutate come false.

Flusso condizionale

ha una condizione assegnata che definisce se usare o meno il flusso.

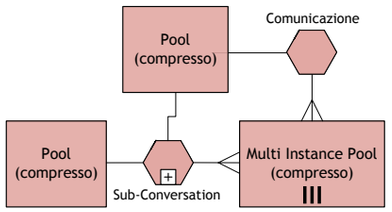
Conversazioni

Una **comunicazione** definisce un insieme di scambi di messaggi collegati logicamente. Se annotati con un simbolo [+] indicano una comunicazione interna ad un'altra conversazione.

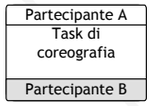
Un ***conversation link*** connette le comunicazioni ed i partecipanti.

Un ***forked conversation link*** connette le comunicazioni e molteplici partecipanti.

Diagramma di conversazione

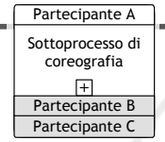


Coreografie



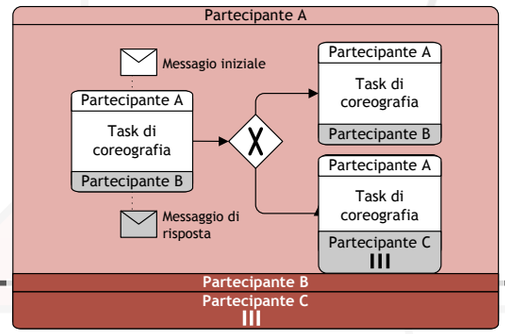
Un **Task di coreografia** rappresenta un'interazione(scambio di messaggi) tra due partecipanti.

Il simbolo ***Multiple Participants*** denota un insieme di partecipanti della stessa tipologia.

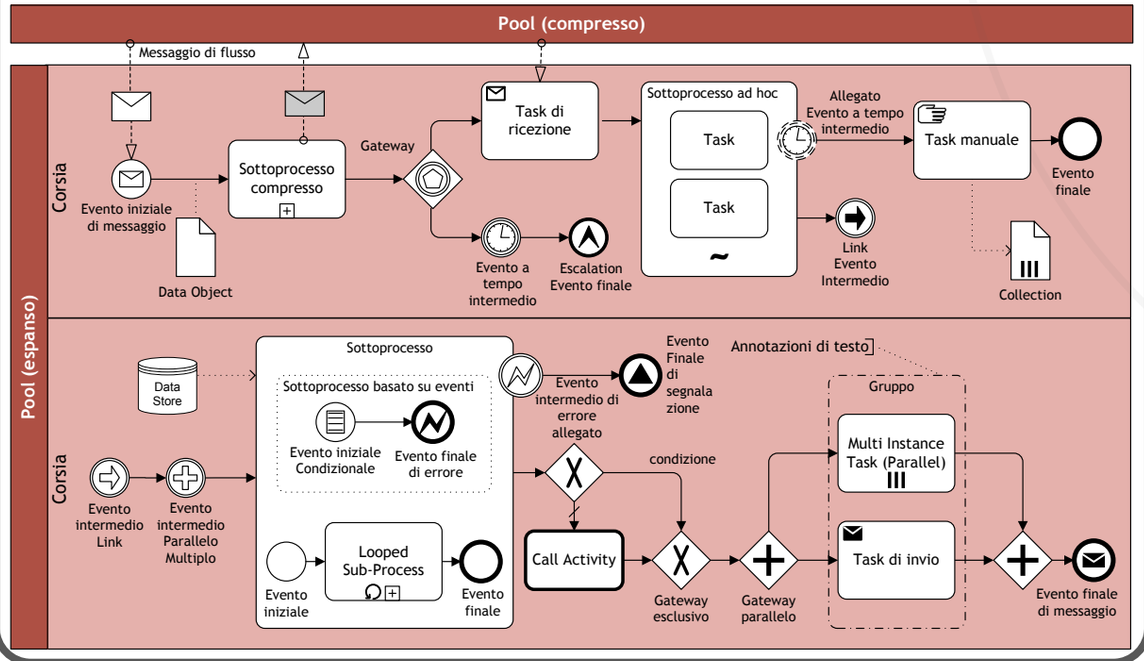


Un **Processo di coreografia** contiene una coreografia rifinita con molte interazioni.

Diagramma di coreografia



Collaboration Diagram



Eventi

	Inizio			Intermedio				Fine
	Alto livello	Interruzione di sottoprocessi	Non-interruzione di sottoprocessi	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	
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 *link events* 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.	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallelo Multiplo: intercetta tutti gli eventi.	✚	✚	✚	✚	✚	✚		
Terminate: causa la fine immediata di un processo.								●

Gateways

Esclusivo(xor)

Basato su eventi

Parallelo

In caso di *splitting*, il flusso sequenziale viene diretto esattamente verso uno dei rami in uscita. In caso di *merging*, il flusso aspetta che un ramo in entrata arrivi a termine prima di andare avanti.

Questo simbolo è sempre seguito da intercettazioni di eventi o *tasks* di ricezione. Il flusso sequenziale prosegue verso il sucessivo *task*/evento che accade per primo.

Quando viene usato per dividere il flusso sequenziale, tutti i rami in uscita sono attivati simultaneamente. Invece quando viene usato per unire rami paralleli, il flusso aspetta il completamento di tutti i rami in entrata prima di andare avanti.

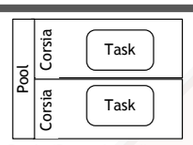
Inclusivo
In caso di *splitting*, uno o più rami sono attivati. Il flusso va avanti solamente quando l'esecuzione di tutti i rami è terminata.

Complesso
Gestioni di *merging* e *branching* che non sono gestite da altri gateways.

Esclusivo basato su eventi
All'attivazione di ogni evento successivo, viene avviata una nuova istanza di processo.

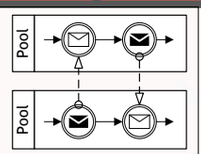
Parallelo basato su eventi
All'attivazione di tutti gli eventi successivi, viene avviata una nuova istanza di processo.

Swimlanes



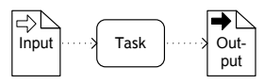
Pools (Partecipanti) e **Lanes(corsie)** rappresentano le responsabilità per le attività in un processo. Esse possono essere un'organizzazione, un ruolo o un sistema. Le corsie suddividono le *pools* o altre *corsie* gerarchicamente.

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



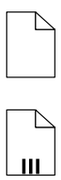
L' **ordine** degli **scambi di messaggi** può essere specificato associando il flusso di messaggi e il flusso sequenziale.

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.

BPM Offensive Berlin

HPI Hasso Plattner Institut — IT Systems Engineering | Universität Potsdam

Humboldt-Universität zu Berlin

camunda — the business process company

inubit — integrating your business and IT

SIGNAVIO — simply professional

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.

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.

Will there be a major rewrite?

Not for 2 or 3 years…

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

Pros and Cons

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

Swimlanes (pools, lanes)

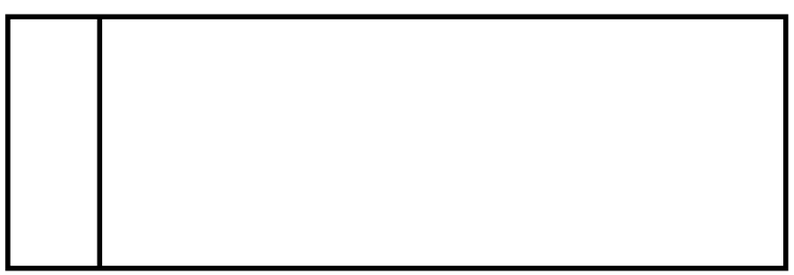
Swimlanes

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

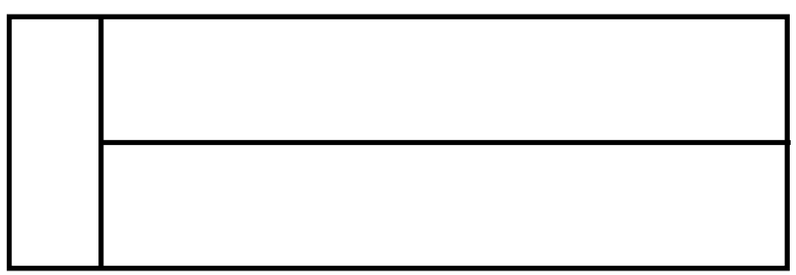
Present in many process modelling methodologies

BPMN supports two main swimlane objects:

pool

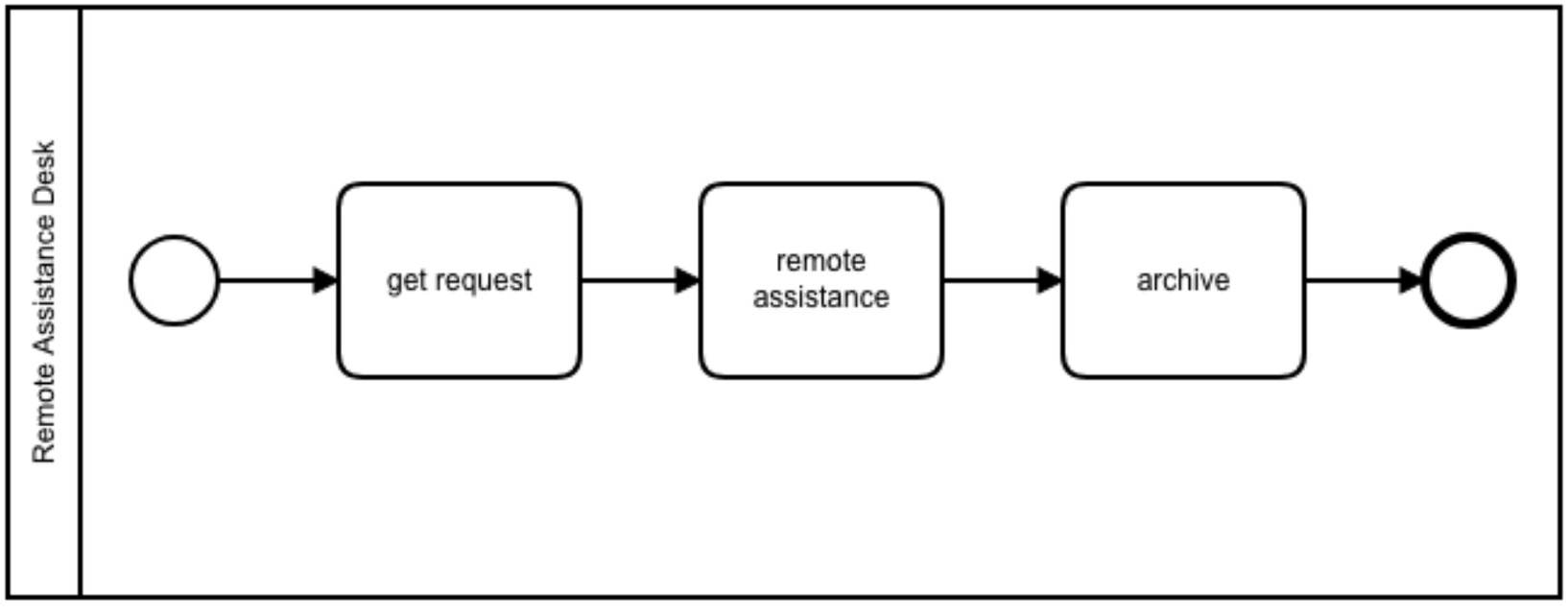


lanes



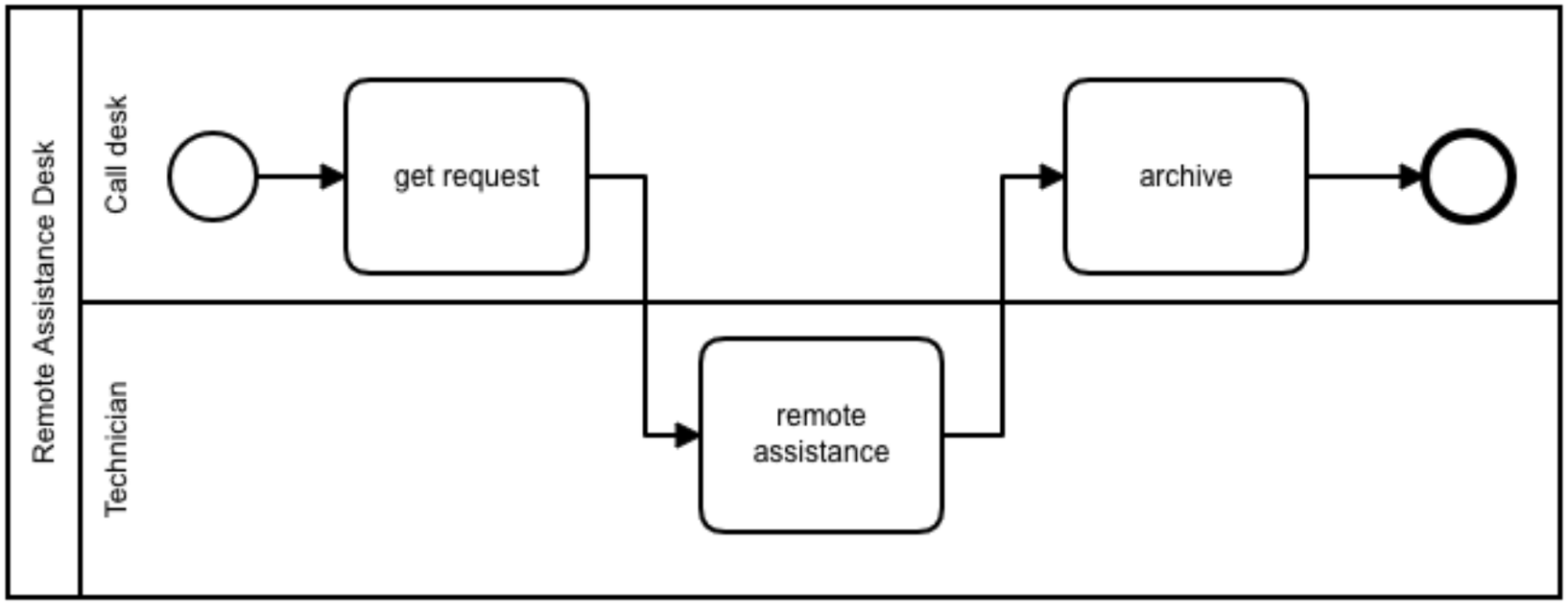
Pools

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



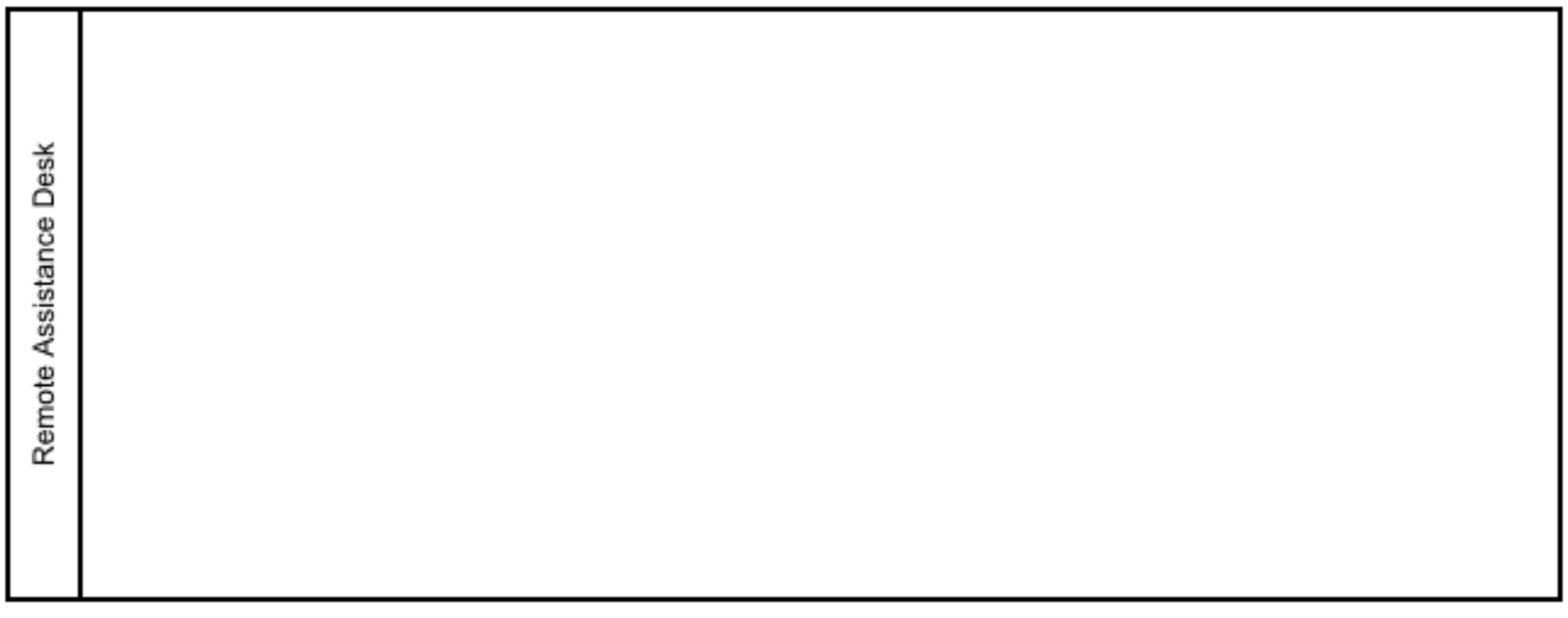
Lanes

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.

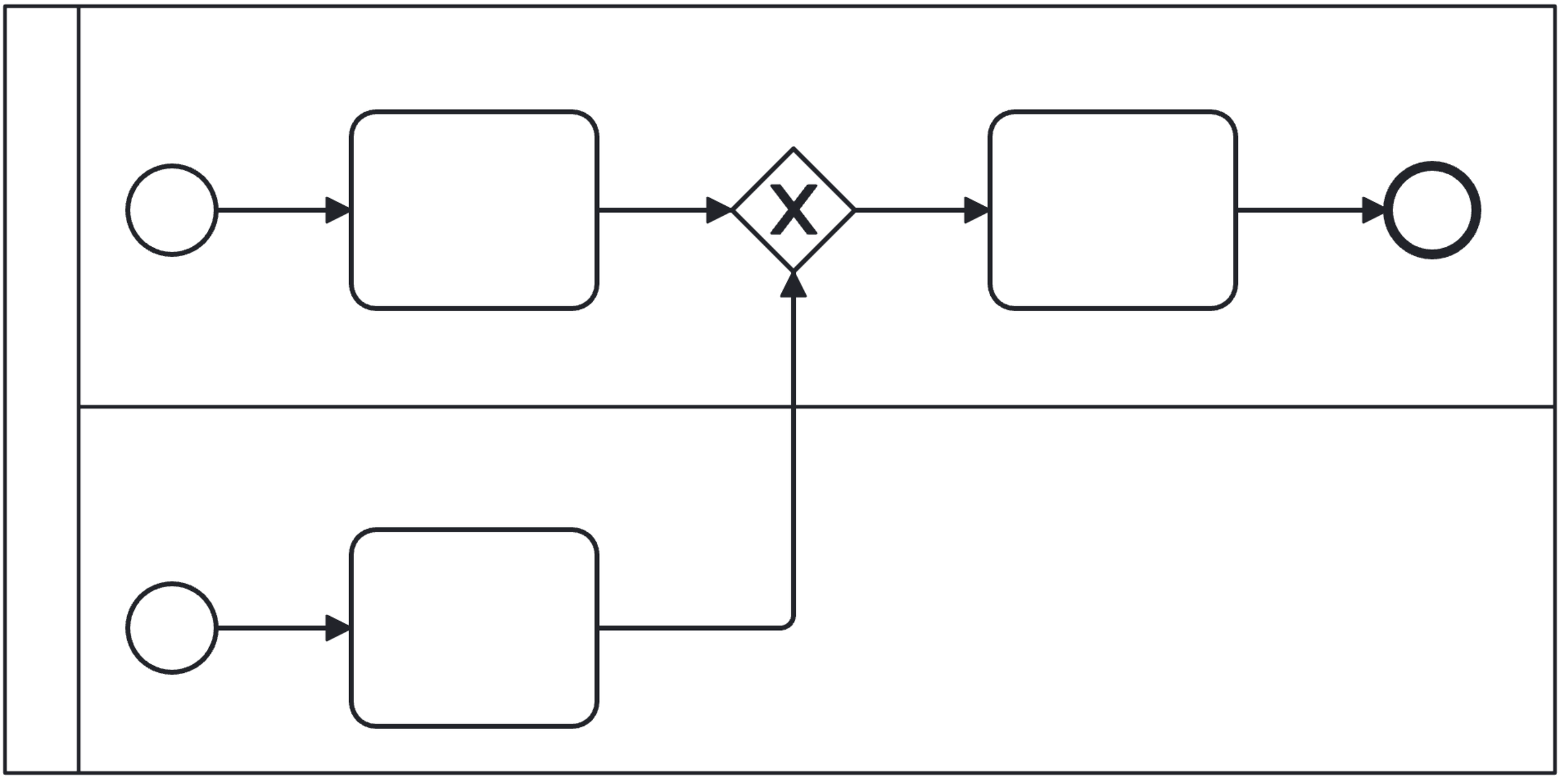
Naming conventions:

a noun possibly preceded by an adjective

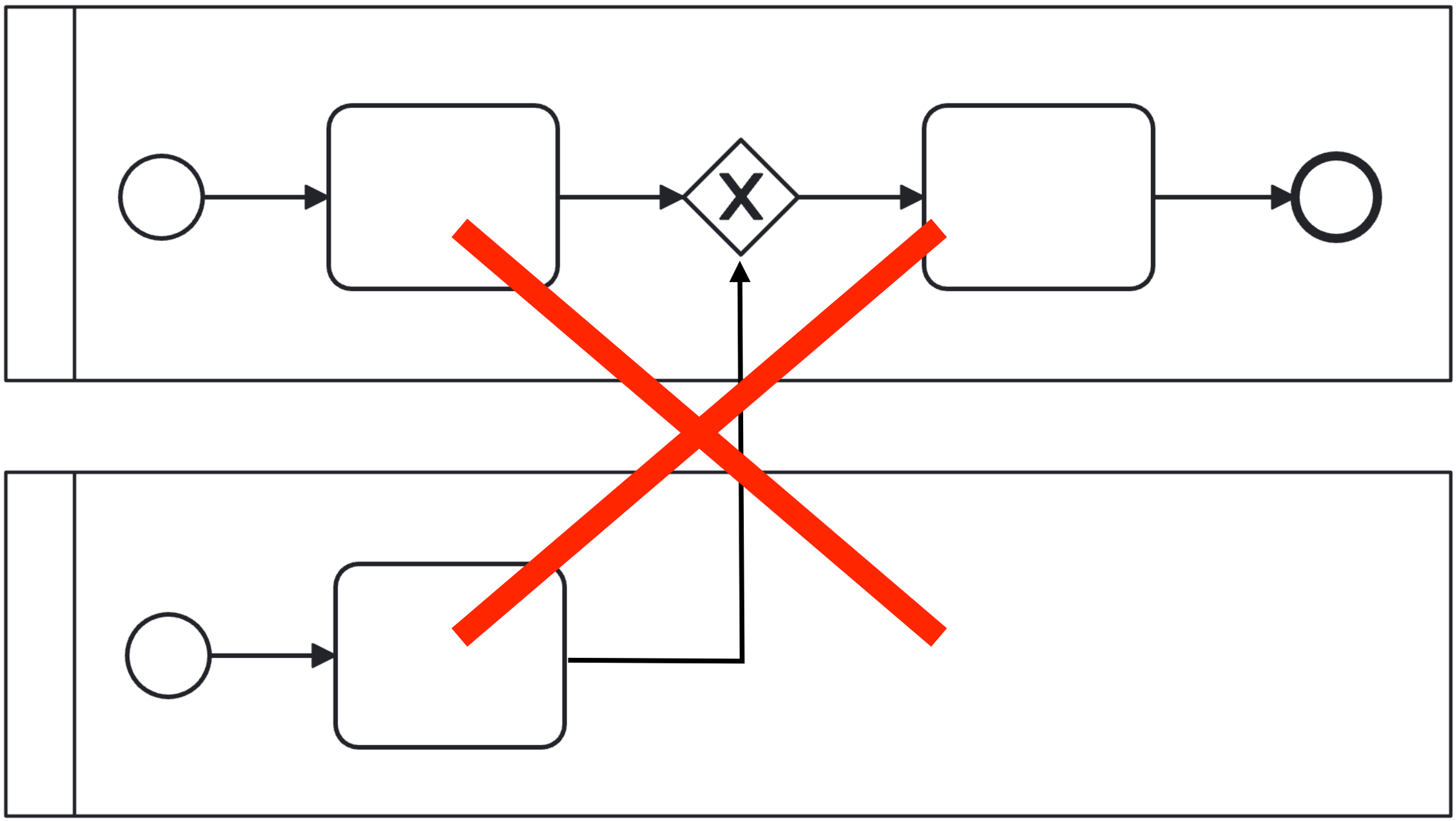
the label is often obtained by "nominalizing" the verb that describes the main action in the process
(e.g., claim handling, order fulfillment)

Avoid long labels
Articles are often omitted

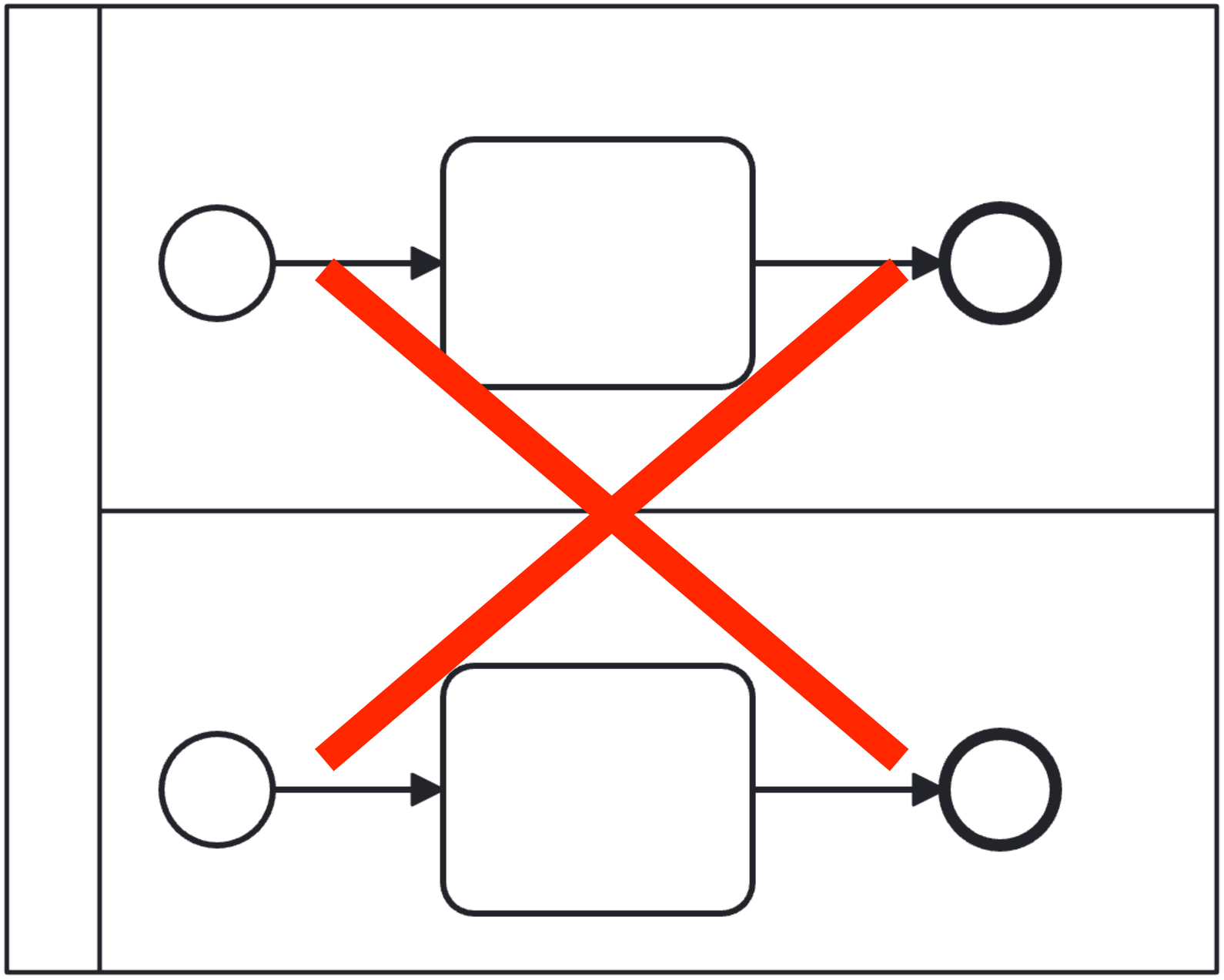
Example



Example



Example



Flow Objects

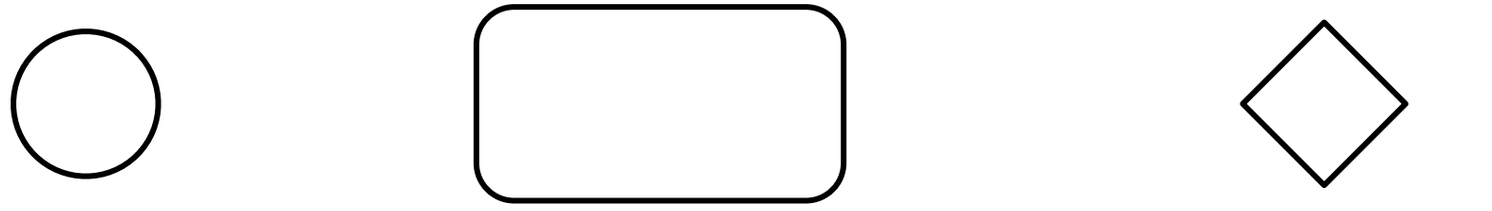
(events, activities, gateways)

Flow objects

Rationale:

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

events **activities** **gateways**

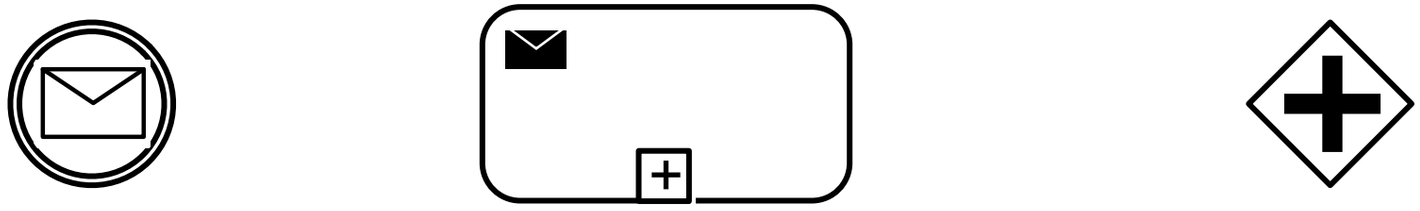


Flow objects

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**)

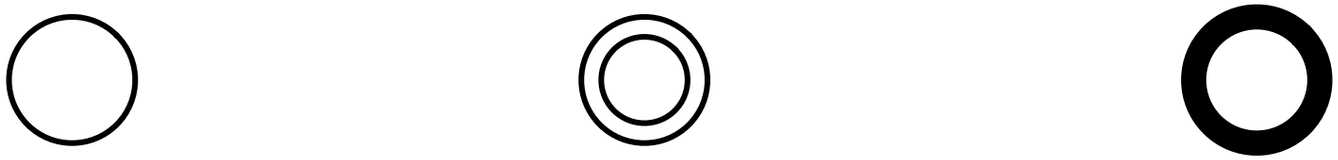
Flow objects: Events

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

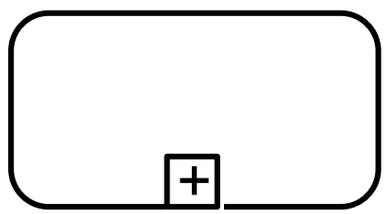
Flow objects: Activities

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**)

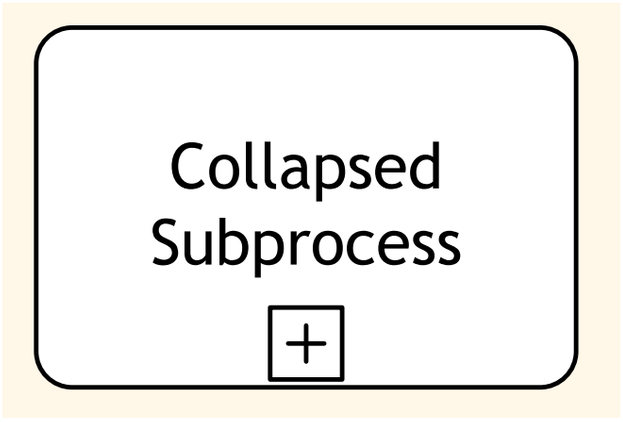


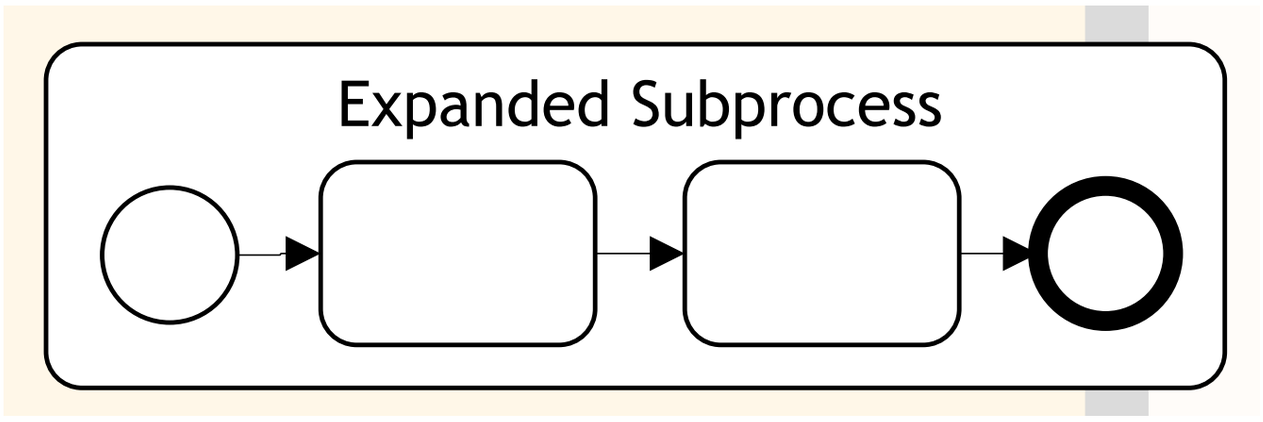


Sub-processes

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

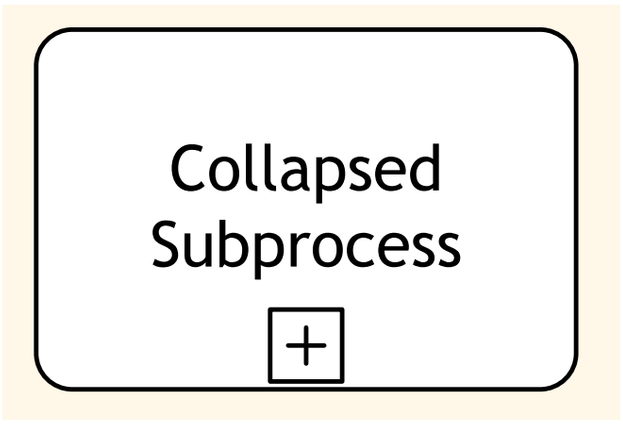


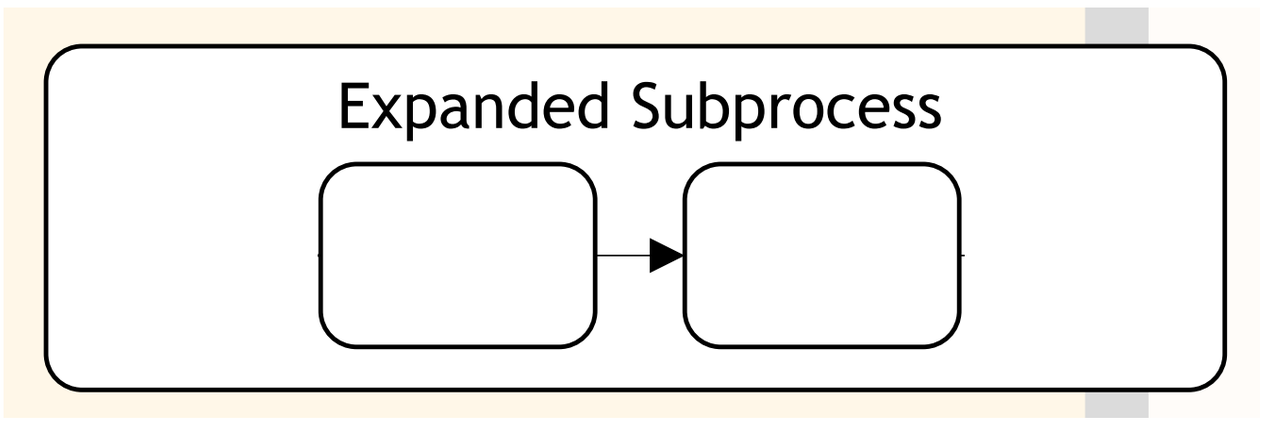


Sub-processes

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)

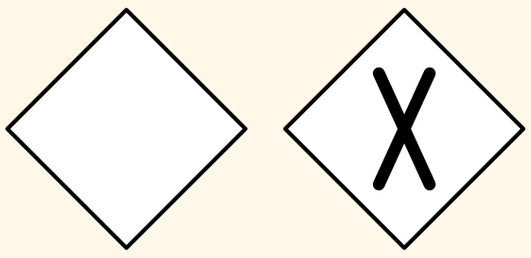
Avoid long labels
Articles are often omitted

Flow objects: Gateways

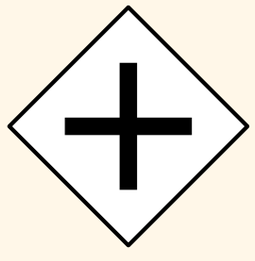
Gateways

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)

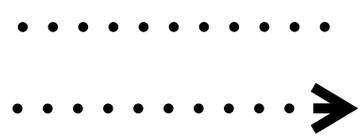
Message flow



connected objects must **reside in different pools**

to be discussed later

Association



connects flow objects with artefacts

to be discussed later

Sequence flow

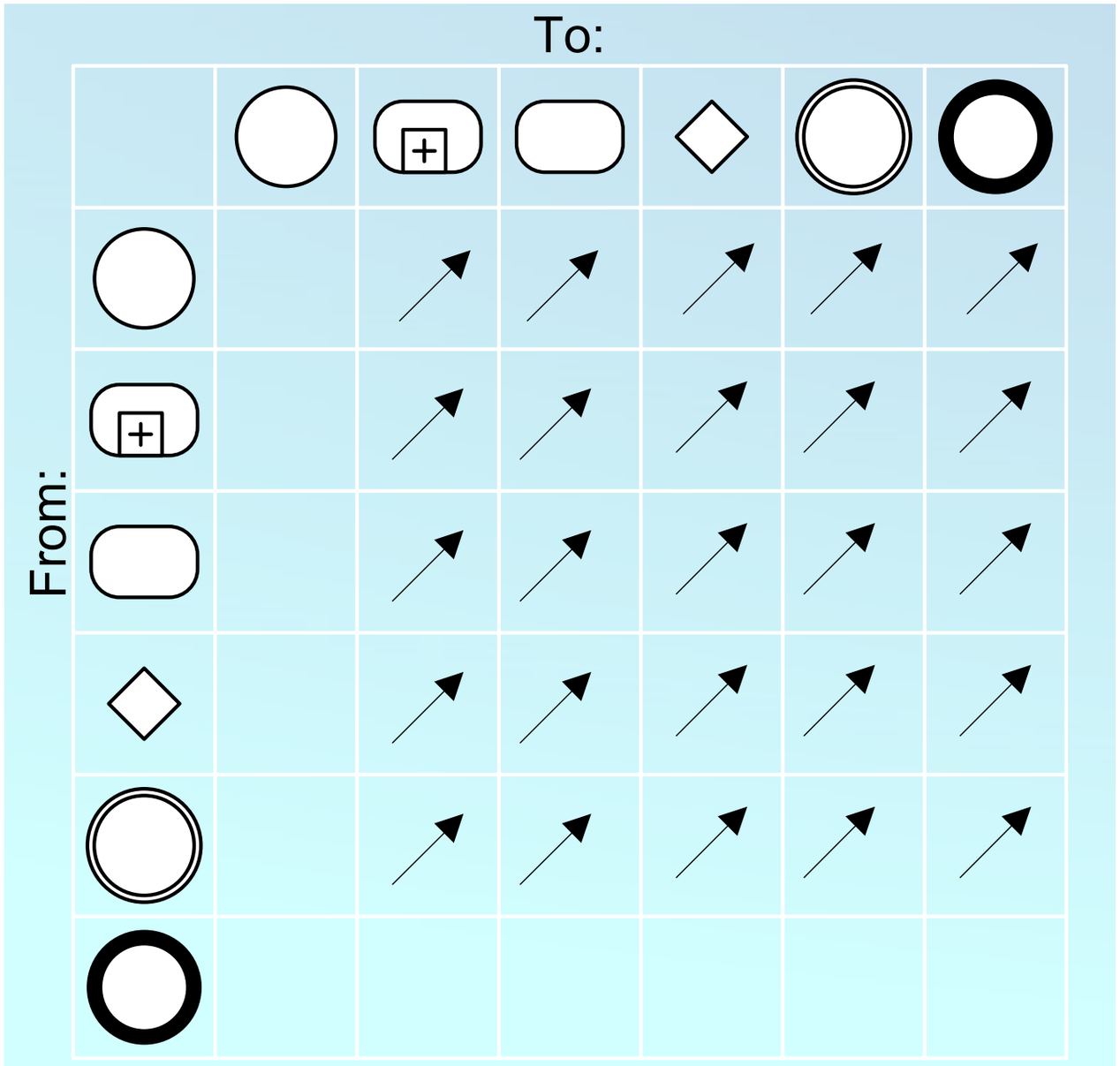
A **sequence flow** is used to show the order
in which activities are to be performed

the term “control flow” is generally avoided in BPMN

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



Requirements



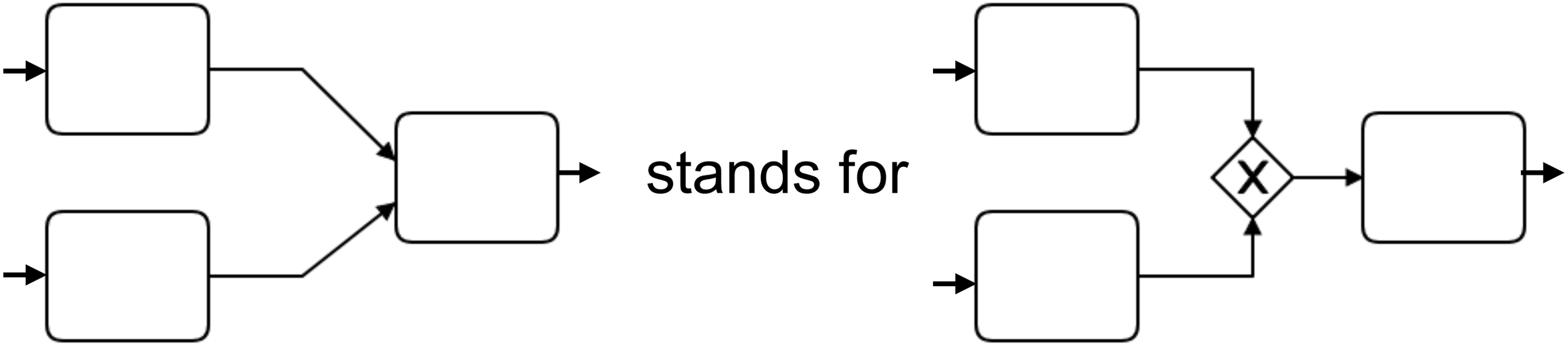
each event:
at most one incoming and
at most one outgoing
sequence flow

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

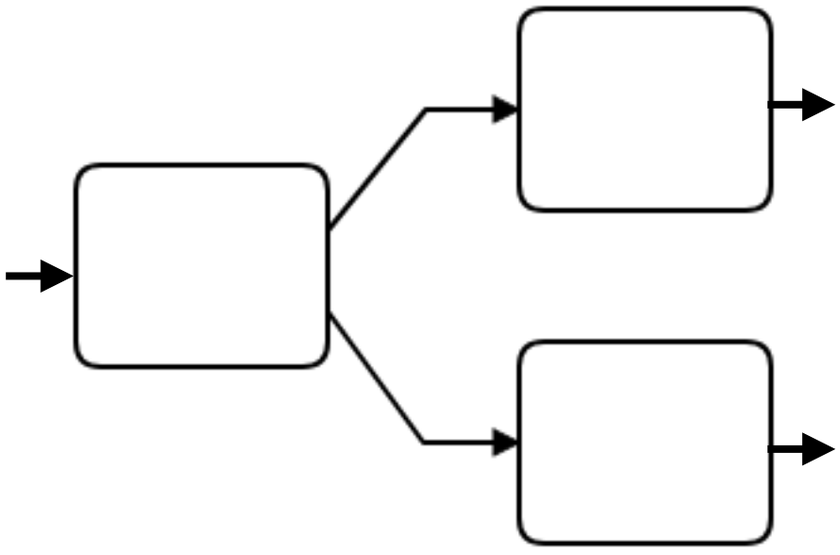
In principle each activity should have exactly:
one incoming arc, one outgoing arc
Be careful if this is not the case!



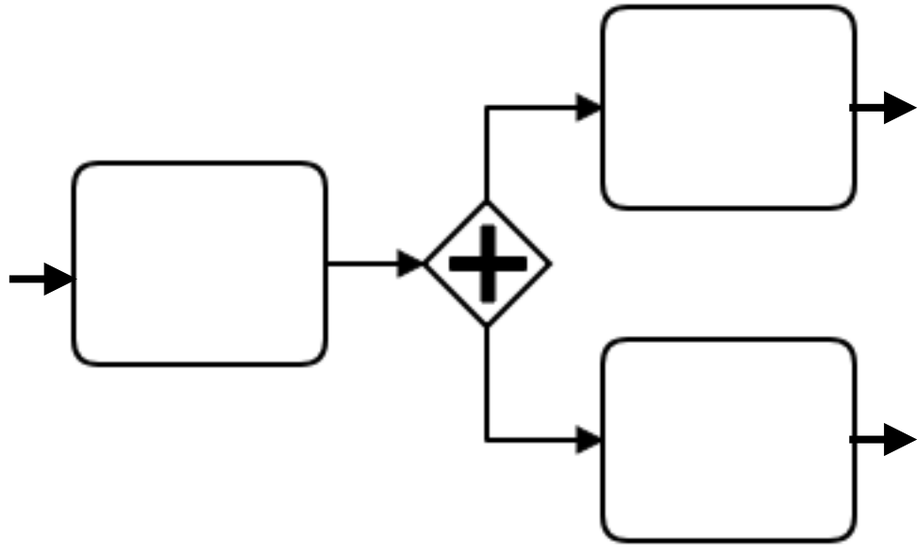
Multiple incoming flows are mutually exclusive

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!

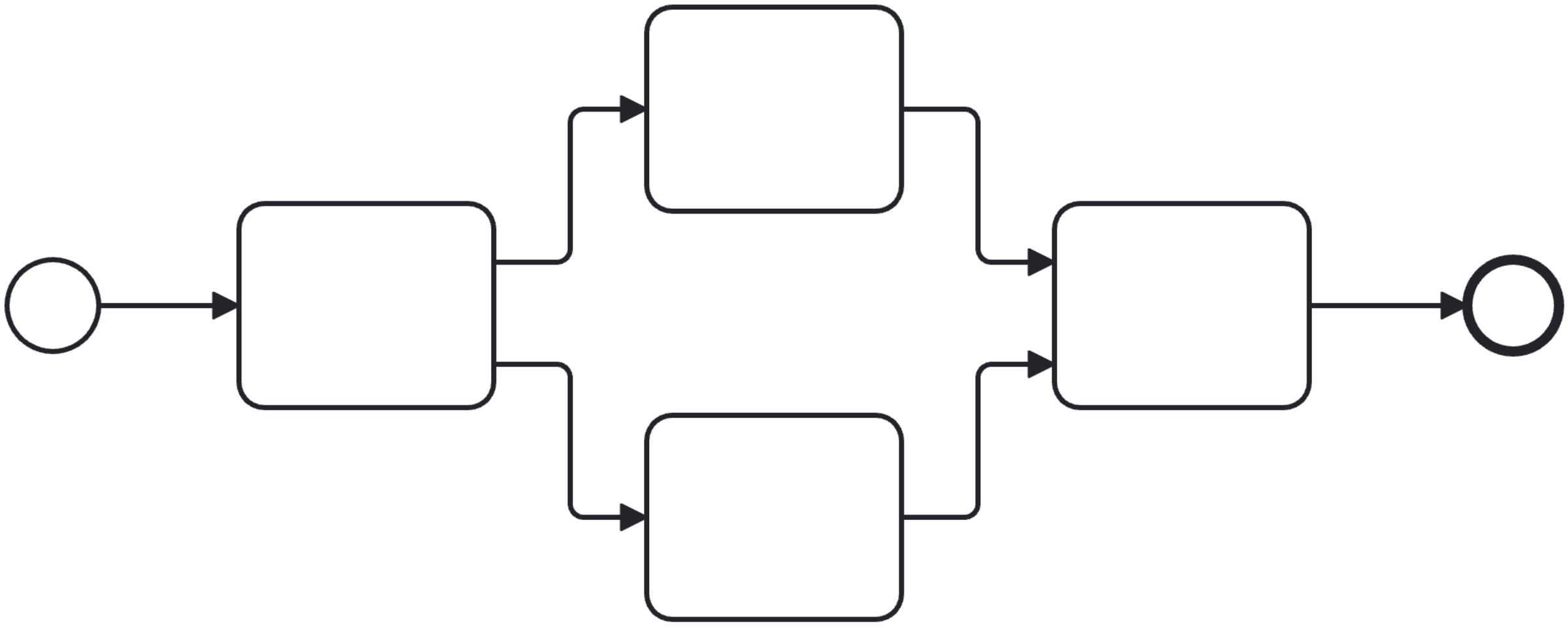


stands for



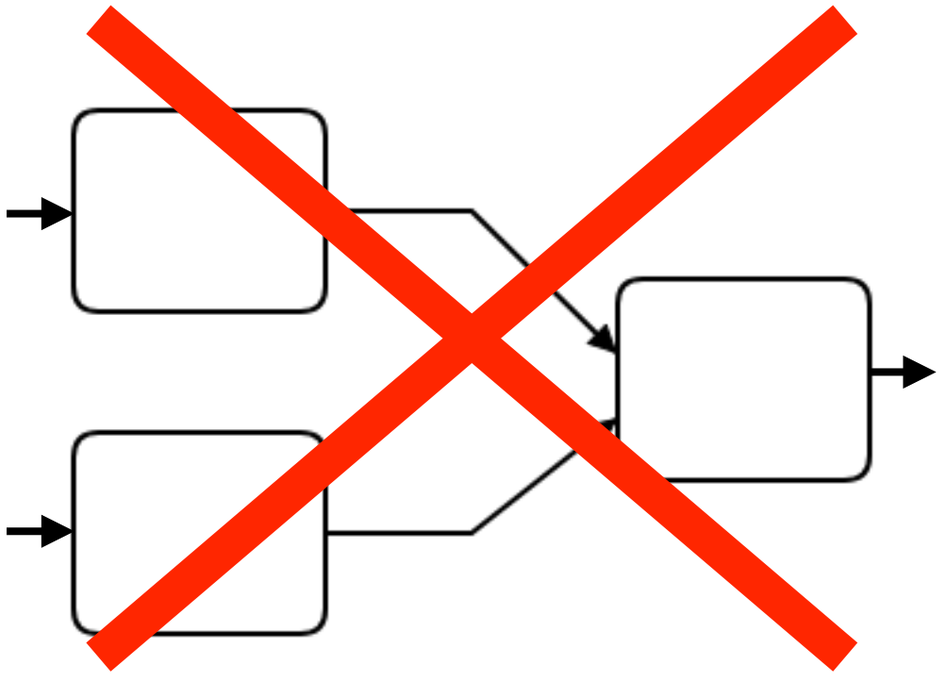
Multiple outgoing flows are activated in parallel
(unless conditions are attached to them)

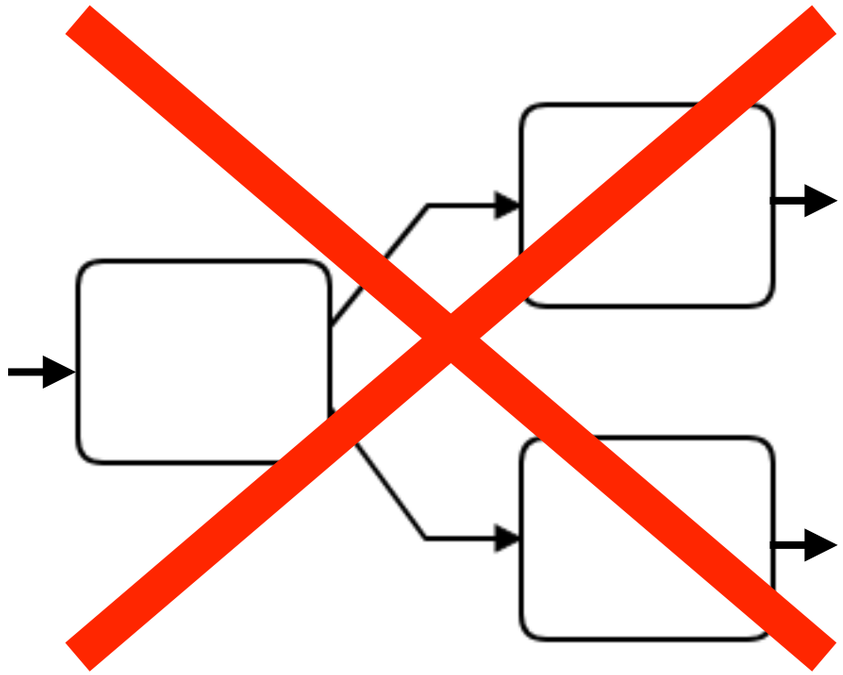
Hidden issue!



In your final projects

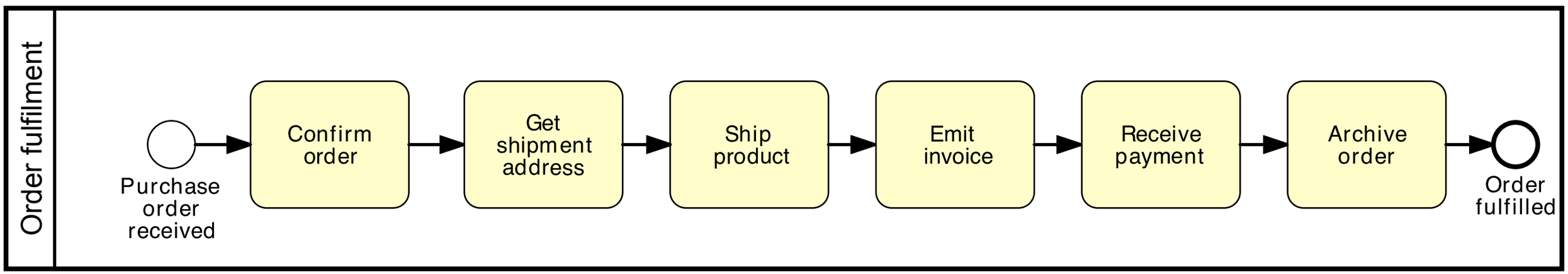
Please avoid



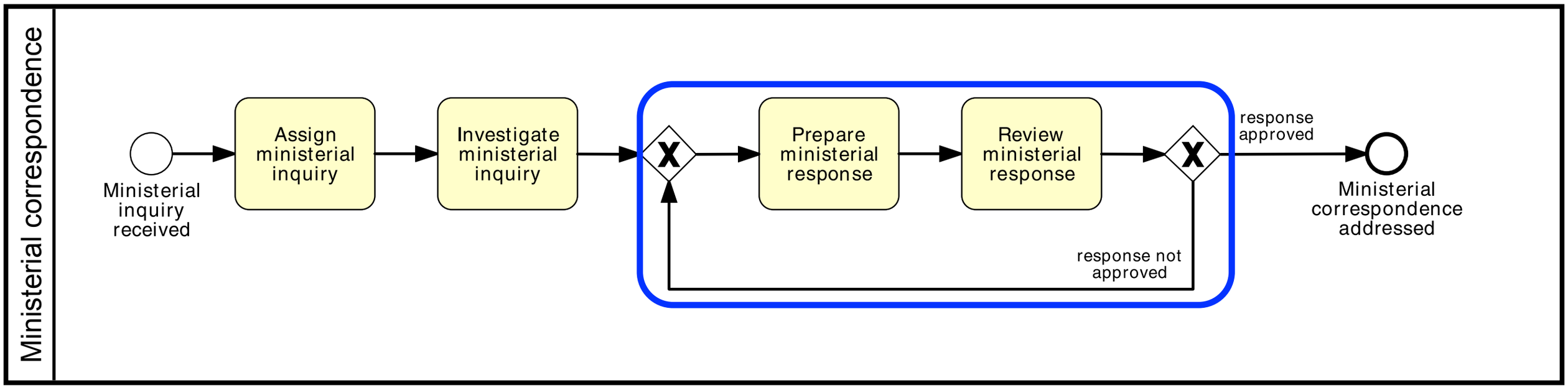


Typical patterns

Sequence:
order fulfilment

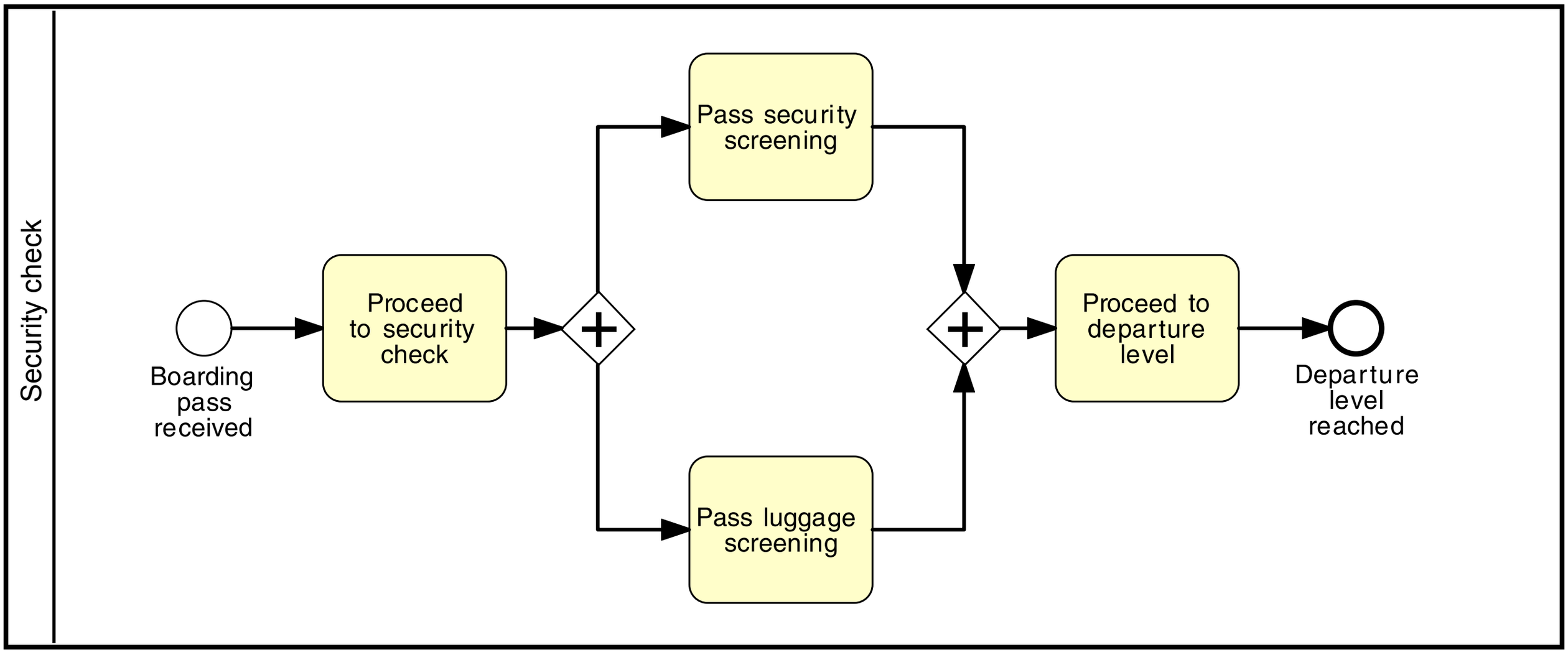


Rework and repetition: ministerial correspondence

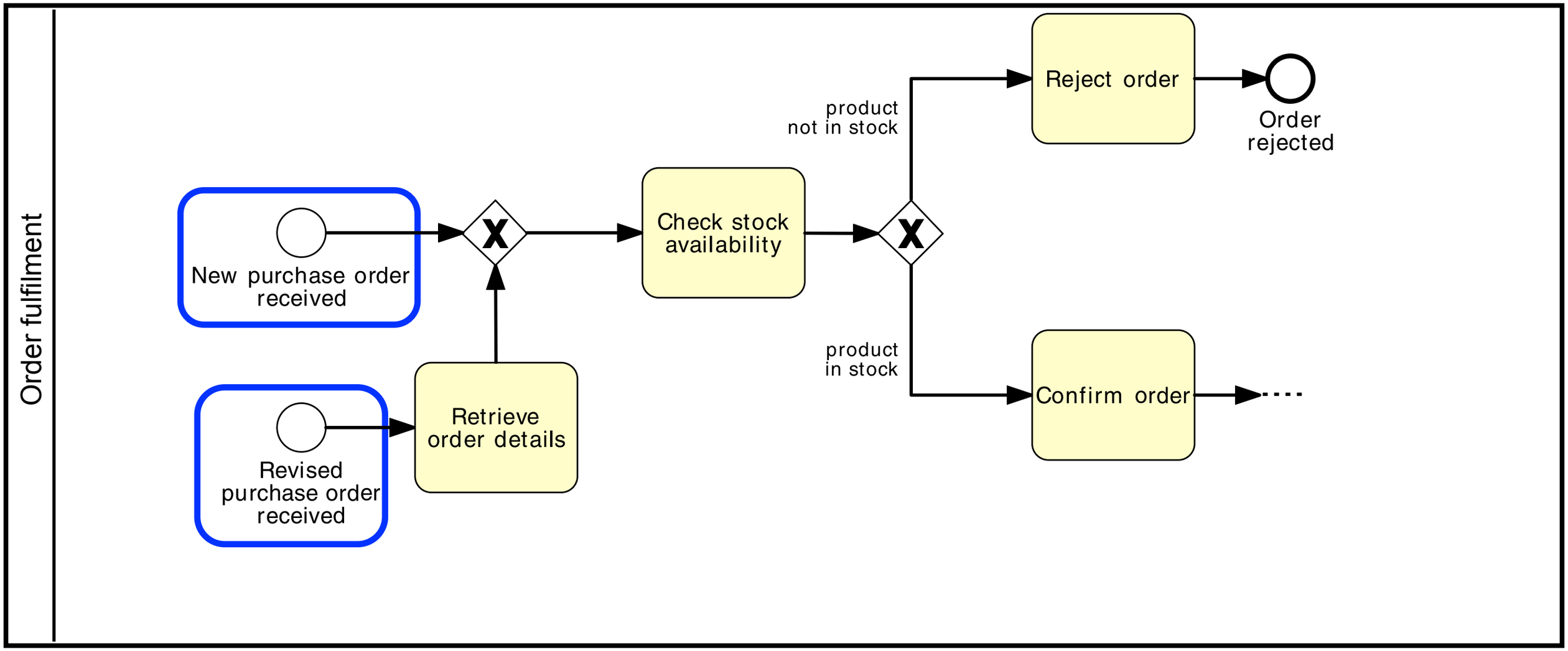


A repetition block starts with a XOR-join
and ends with a decision gateway (XOR-split)

Parallel activities: airport security check

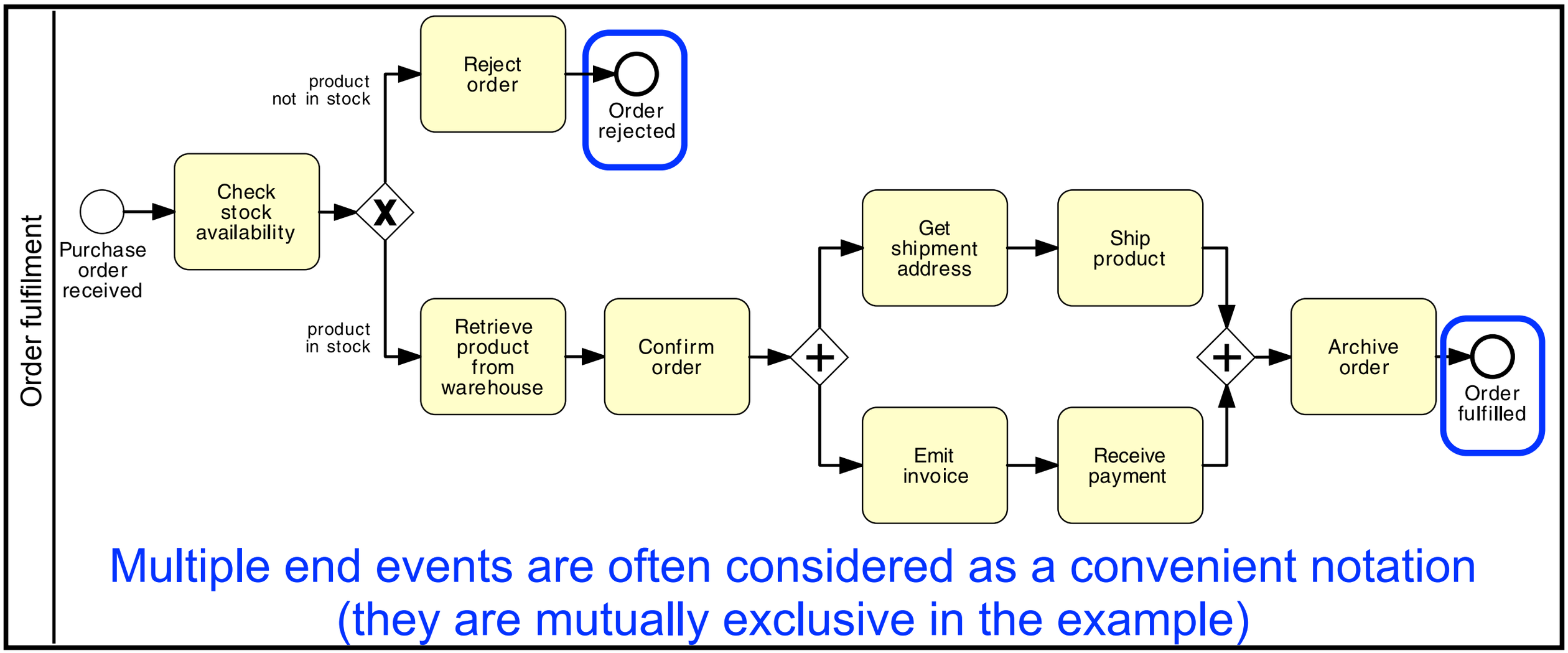


Multiple start events: order fulfilment



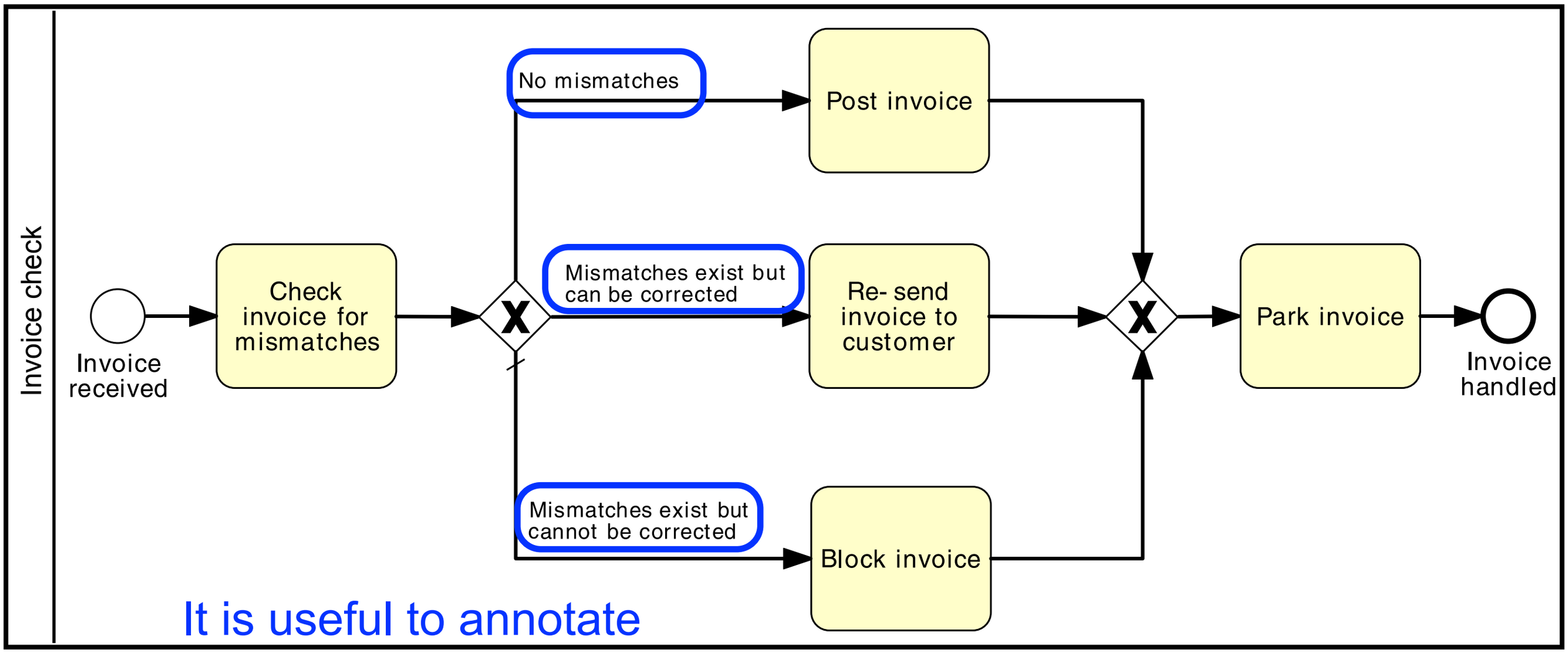
Multiple start events are often considered as a convenient notation (they capture mutually exclusive triggers to start a process instance)

Multiple end events: order fulfilment



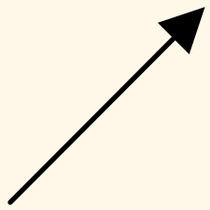
BPMN adopts **implicit termination** semantics:
a case ends only when each ``token'' reaches the end

Exclusive decisions: invoice checking process

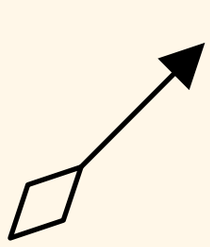


It is useful to annotate branches with the conditions under which they are taken

Annotated sequence flow

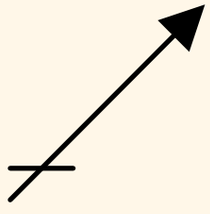


Sequence Flow defines the execution order of activities.



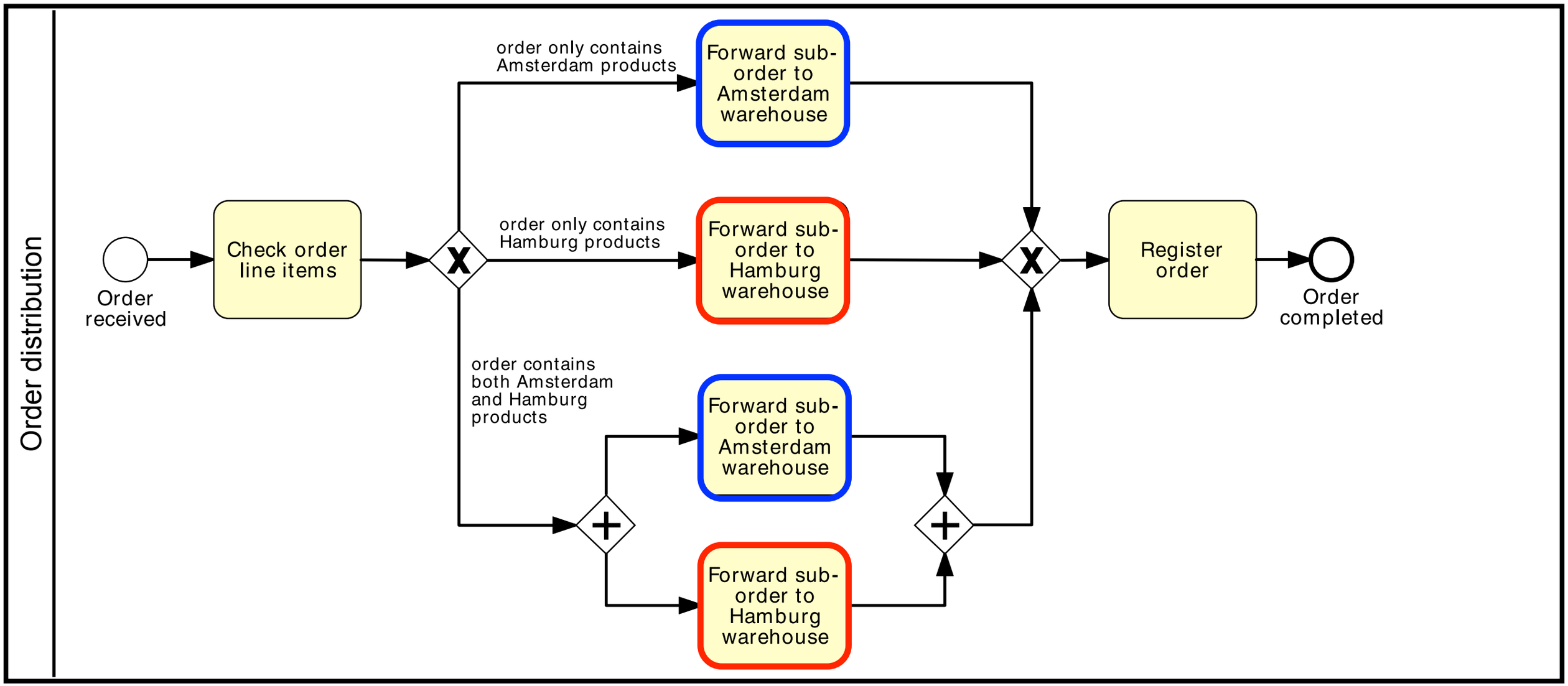
Conditional Flow has a condition assigned that defines whether or not the flow is used.

read as ``otherwise''



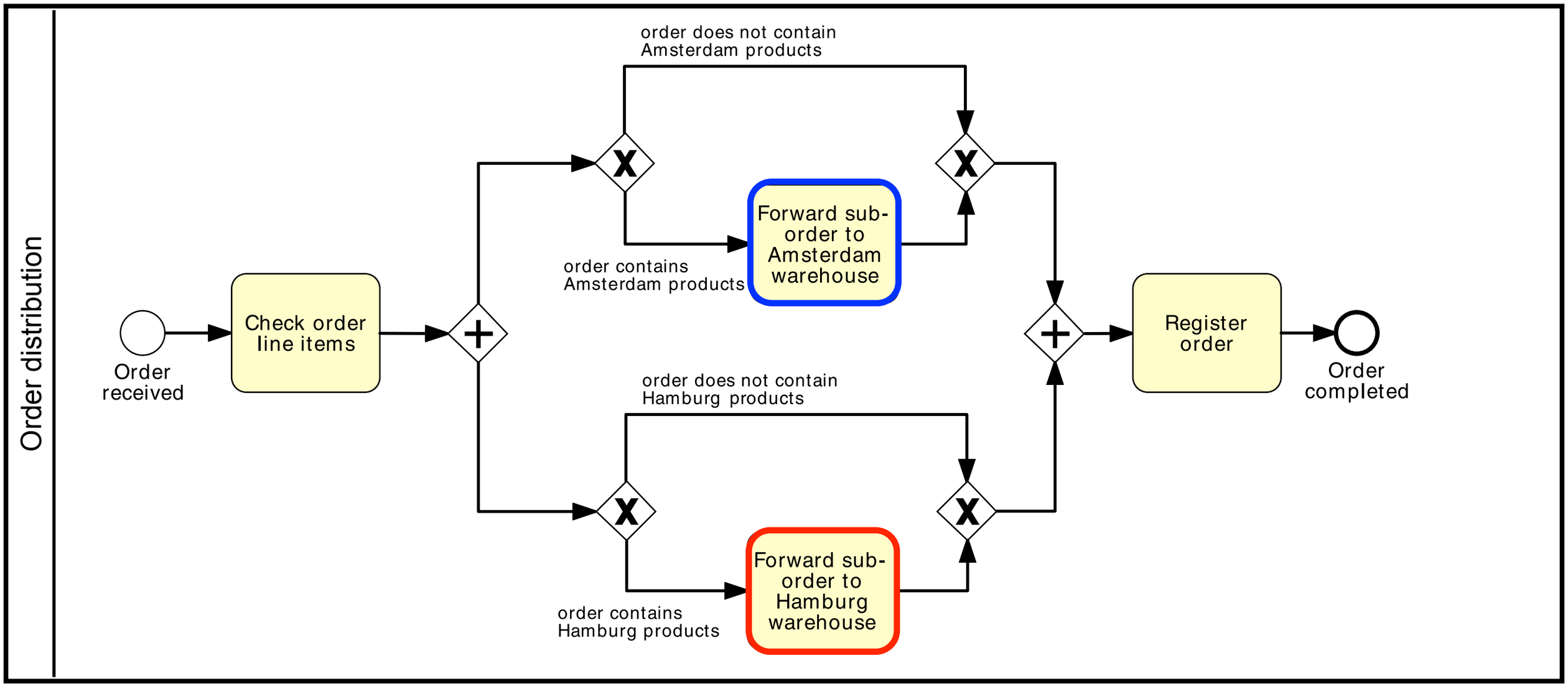
Default Flow is the default branch to be chosen if all other conditions evaluate to false.

Inclusive decisions: order distribution



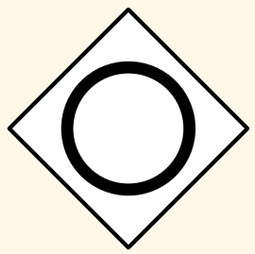
Only XOR / AND gateways, but the diagram is convoluted!
What if we had three or more warehouses? (does not scale)

Inclusive decisions: order distribution



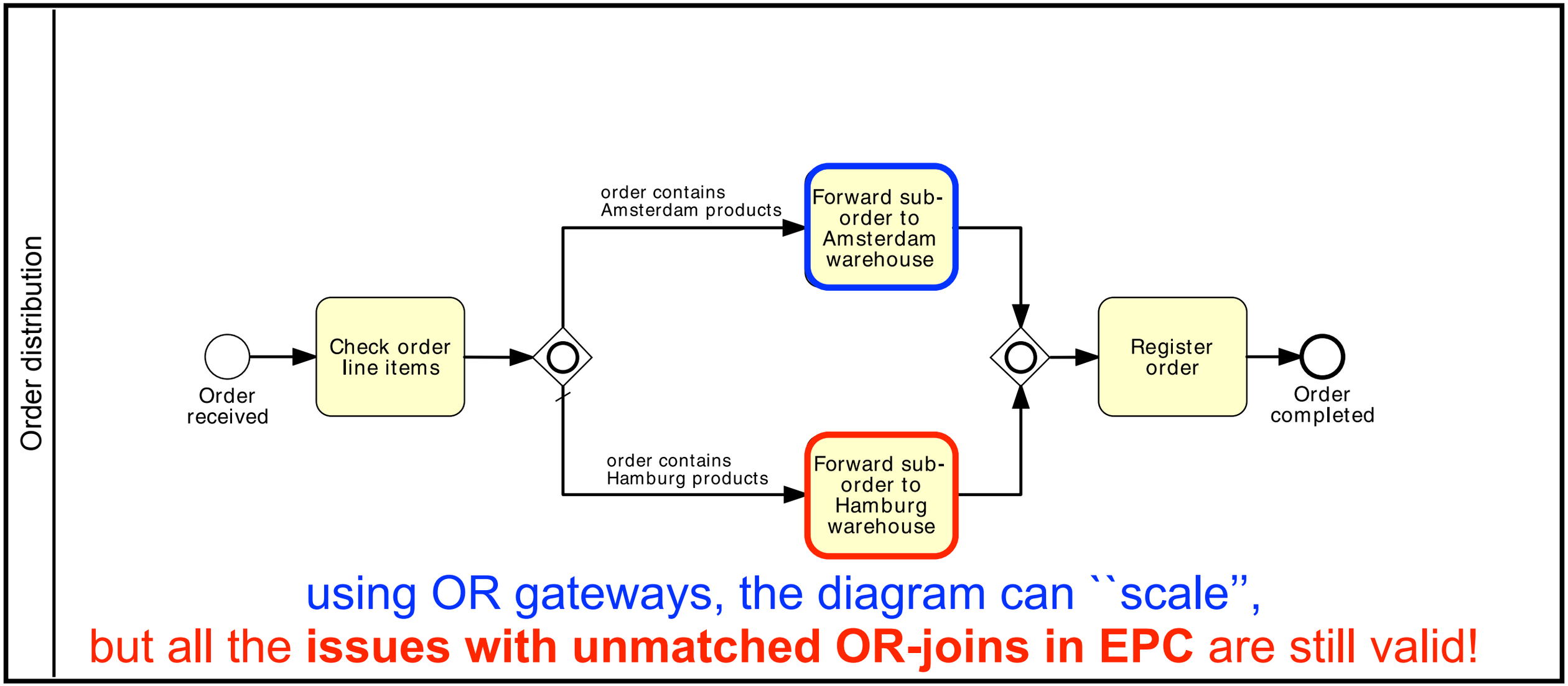
Only XOR / AND gateways, the diagram can ``scale'',
but is it correct? (also the case no-warehouse is now possible)

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.

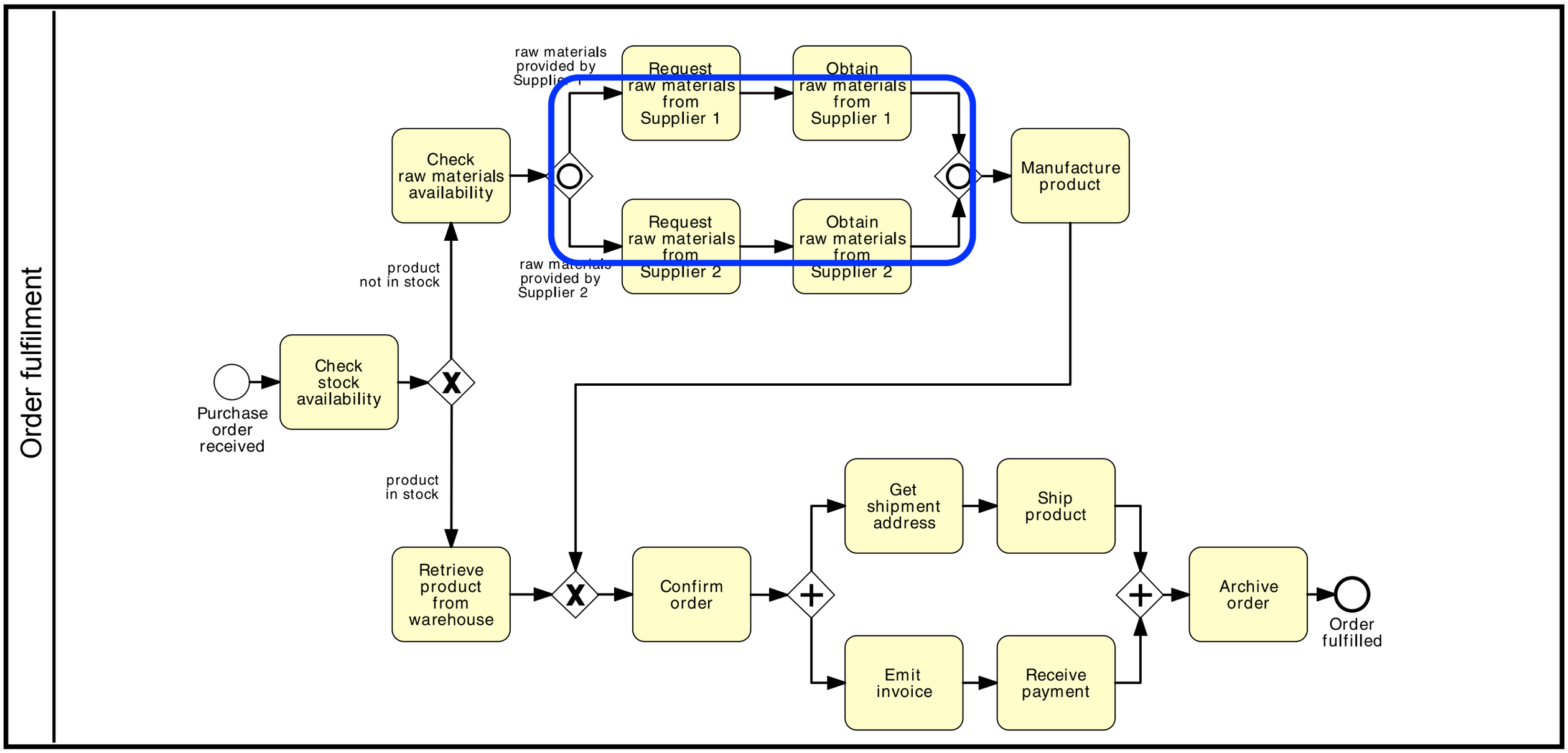
Inclusive decisions: order distribution



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

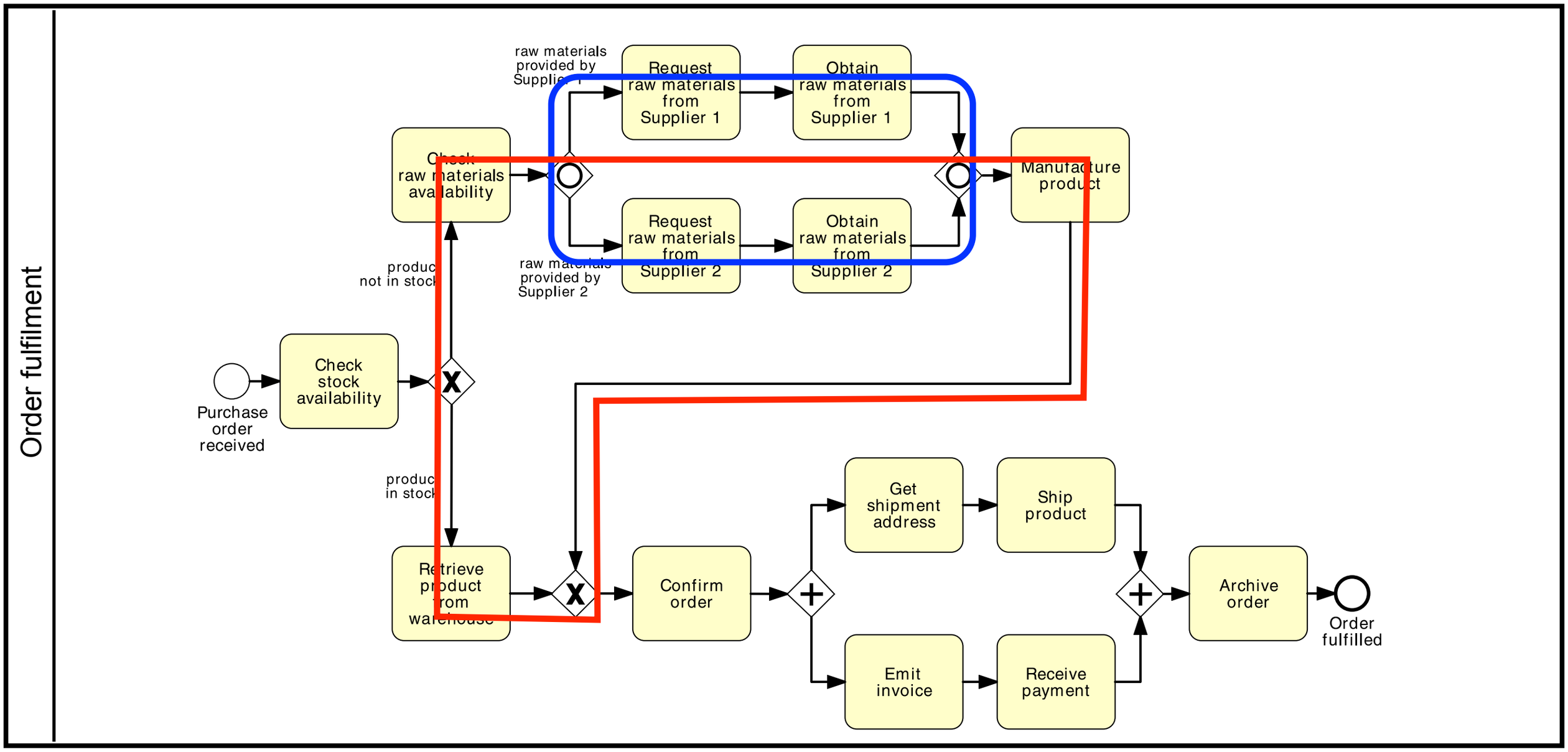
Use OR-gateways only when strictly necessary

XOR + AND + OR: order fulfilment



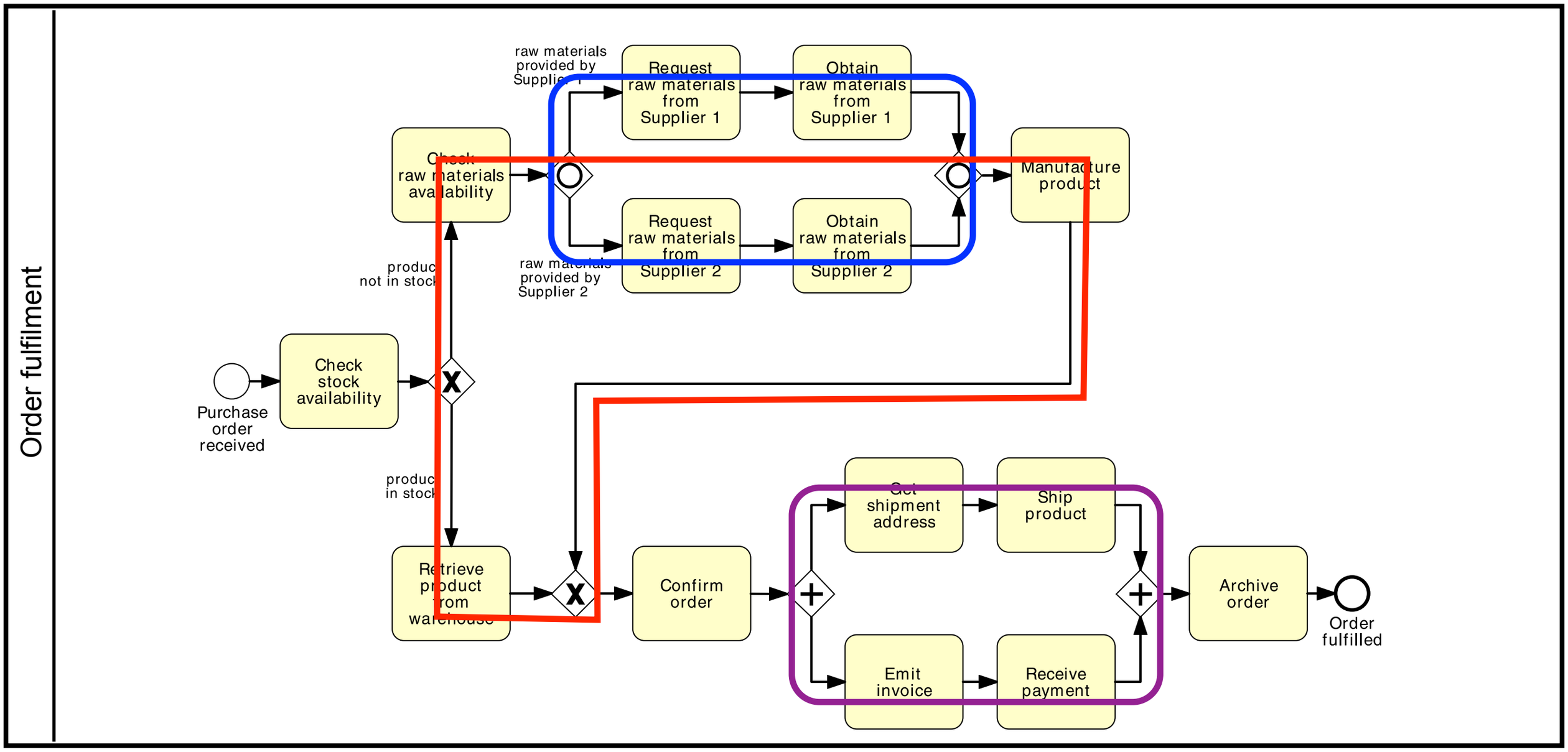
Better if gateways are balanced

XOR + AND + OR: order fulfillment



Better if gateways are balanced

XOR + AND + OR: order fulfillment



Better if gateways are balanced

Placing items in lanes

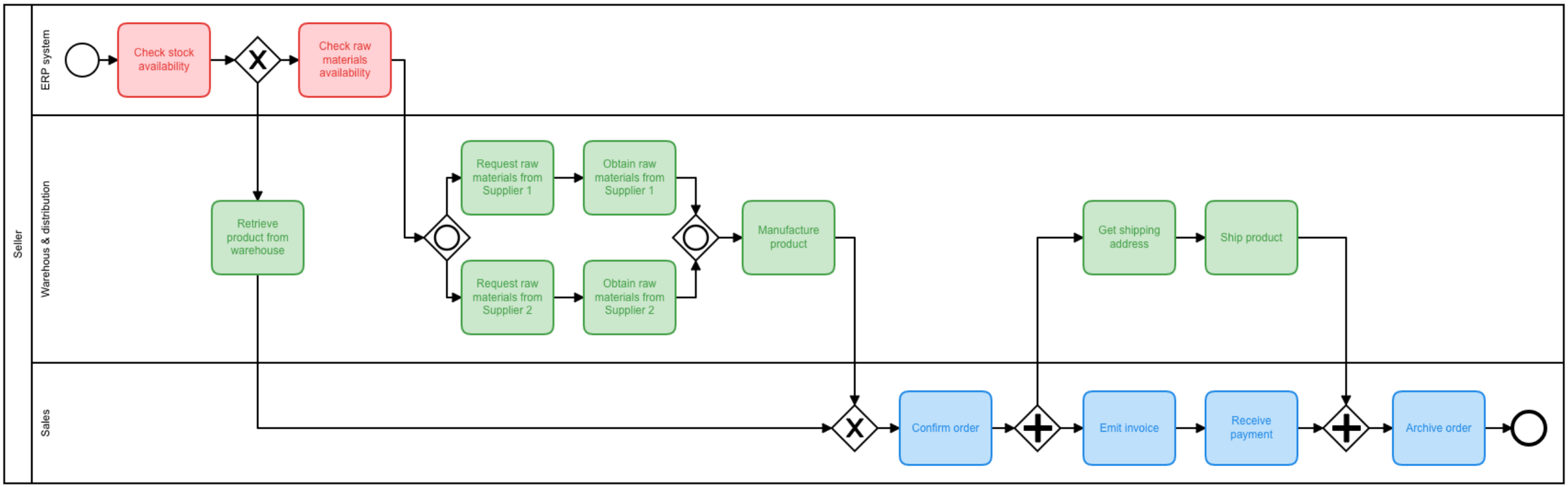
events: must be placed in the proper lane

activities: must be placed in the proper lane

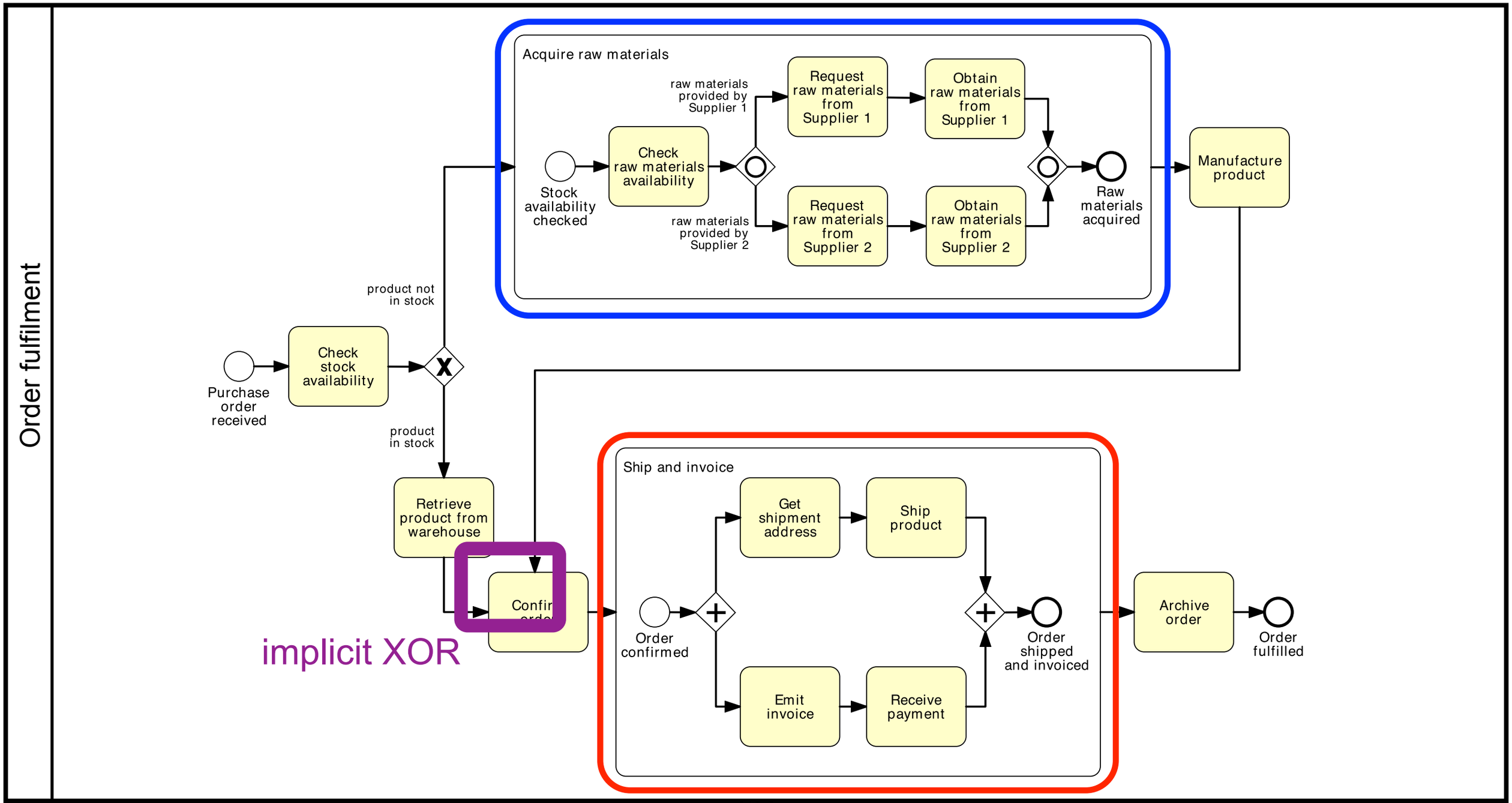
gateways:
(X)OR-splits: same lane as preceding decision activity
AND-split: placement is irrelevant
(any kind of) **join**: placement is irrelevant

data-objects: placement is irrelevant

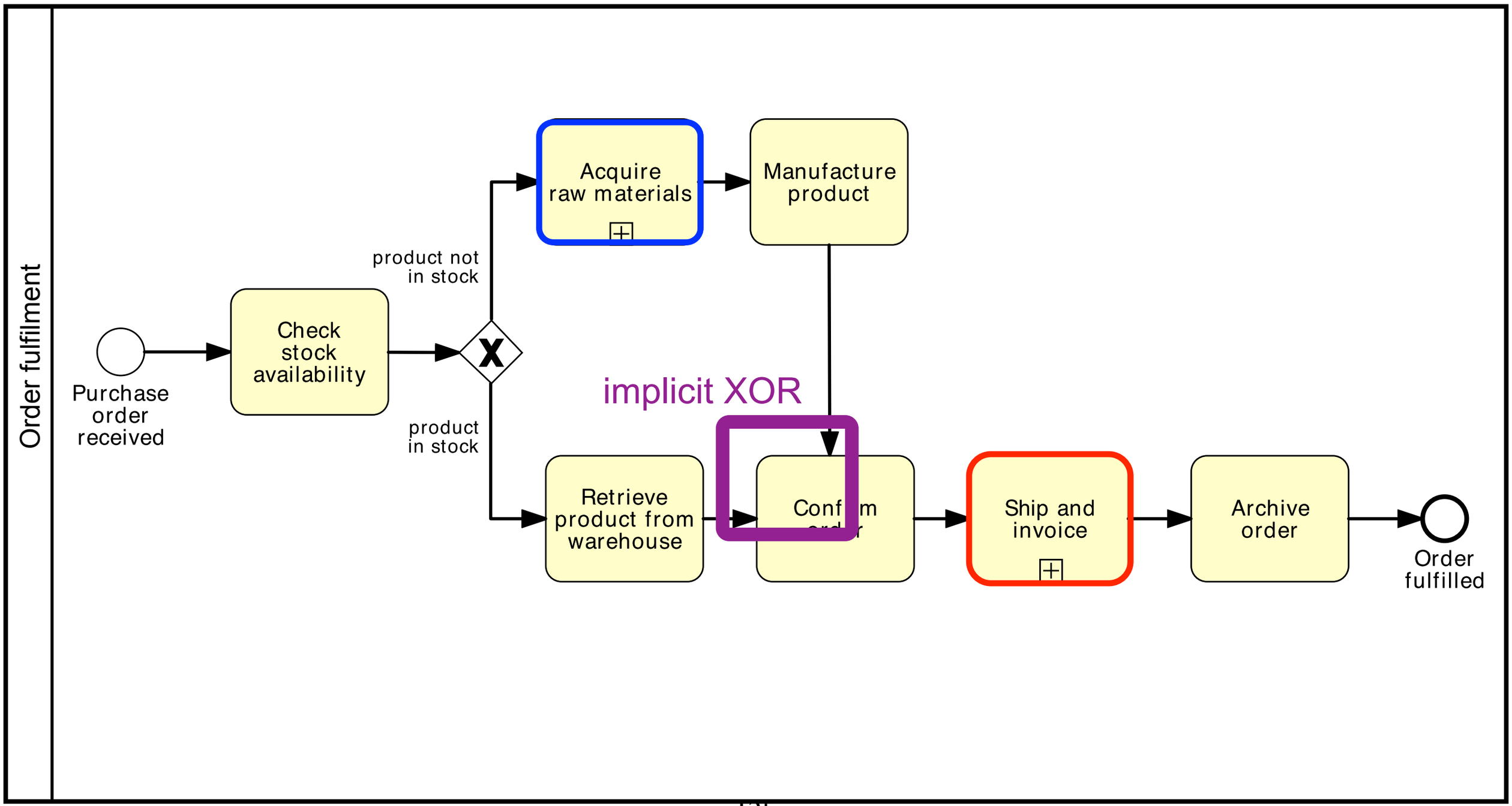
Resources as lanes: order fulfillment



Identify sub-processes

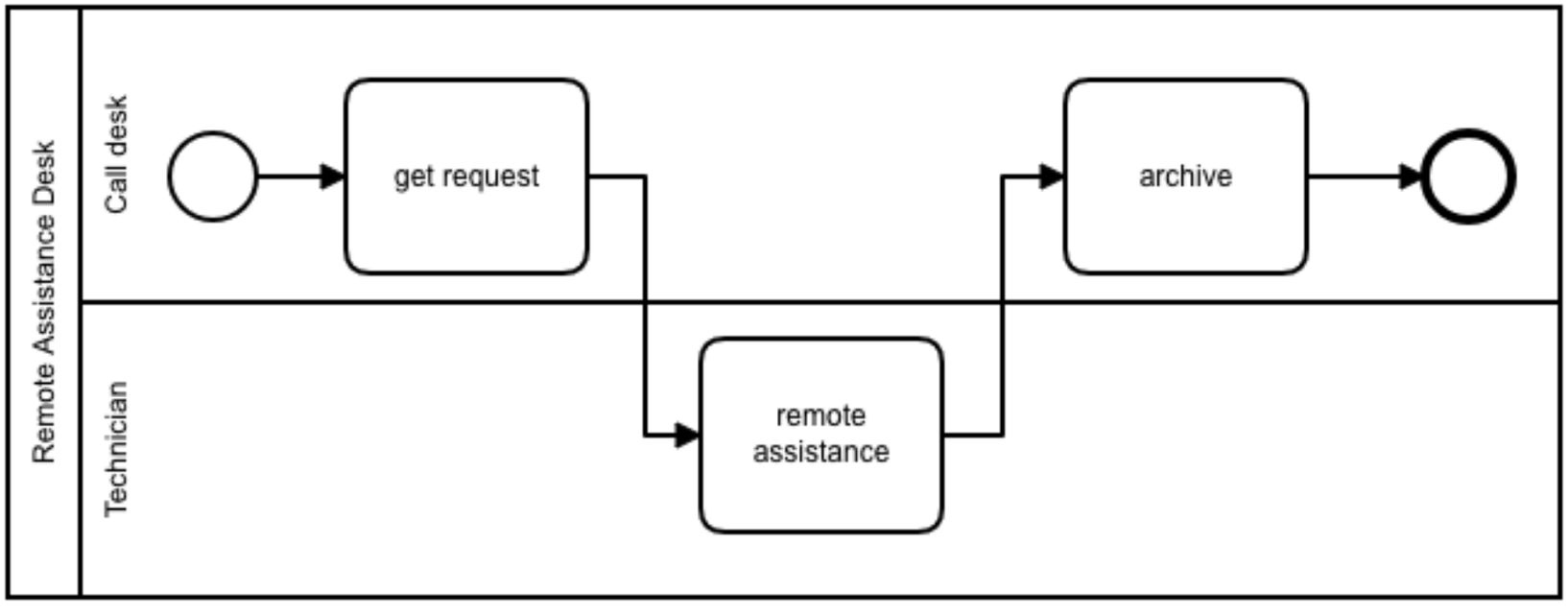


Collapsed sub-processes



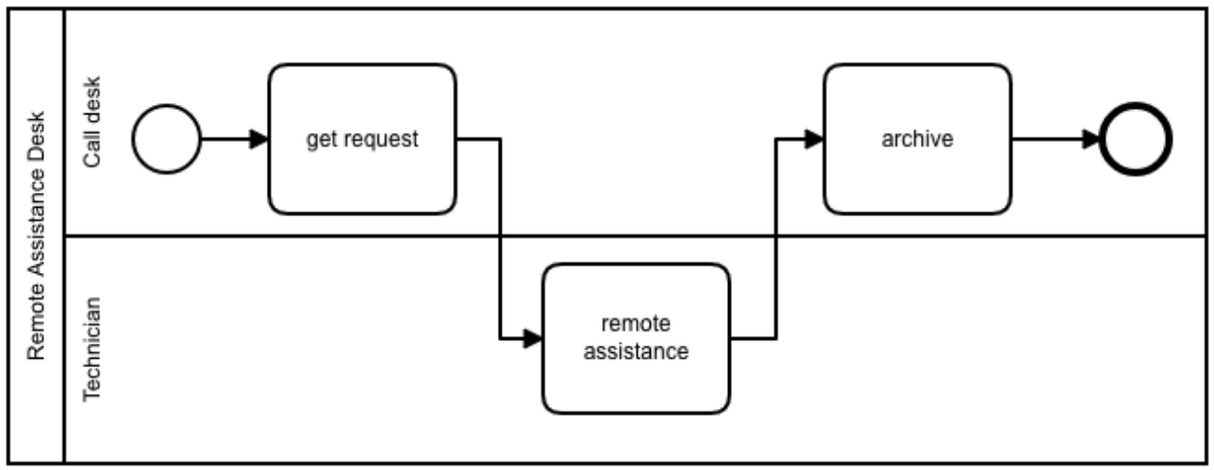
Exercise: basics

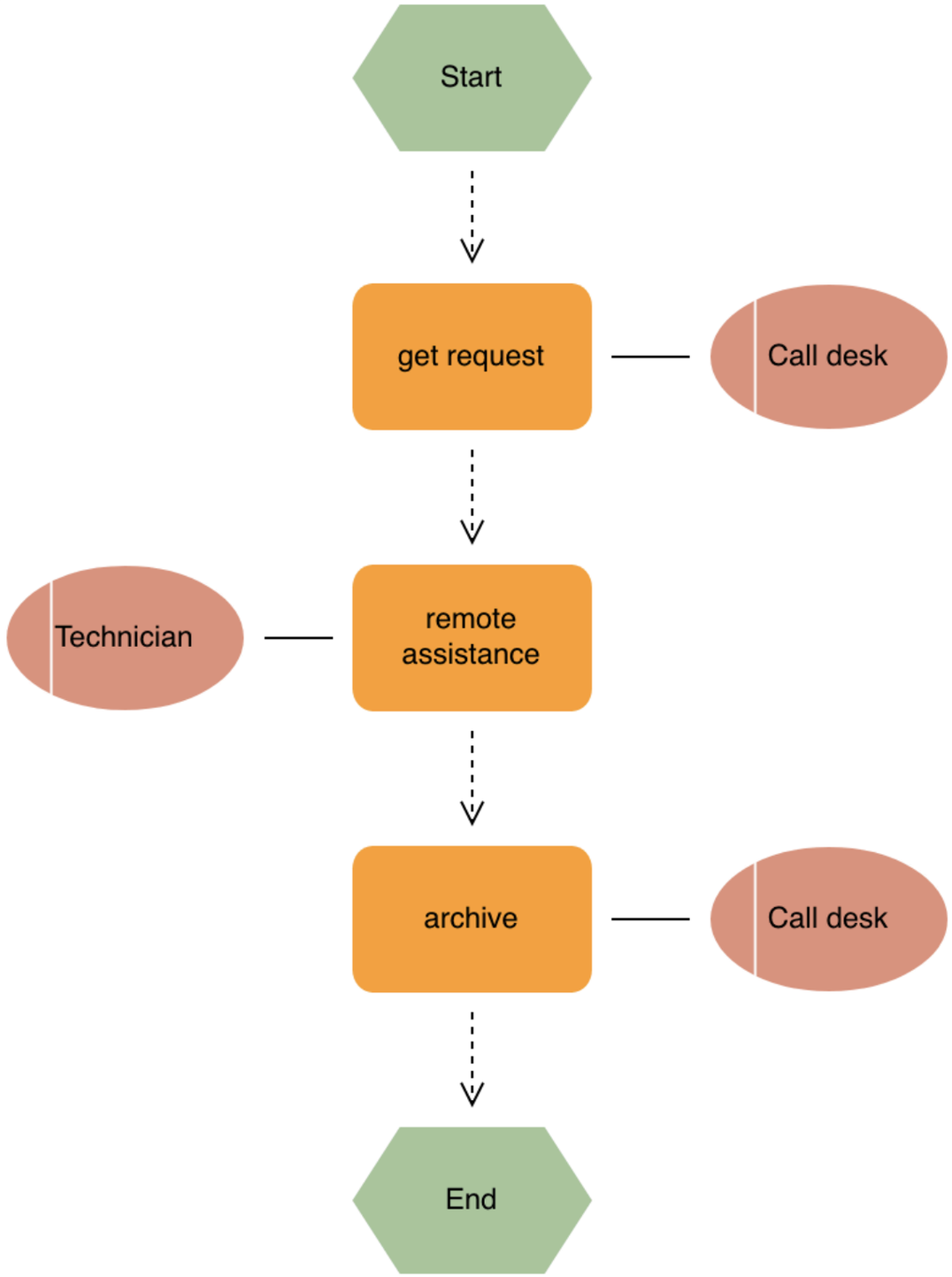
Draw the EPC diagram that corresponds to the BPMN diagram below



Exercise: basics

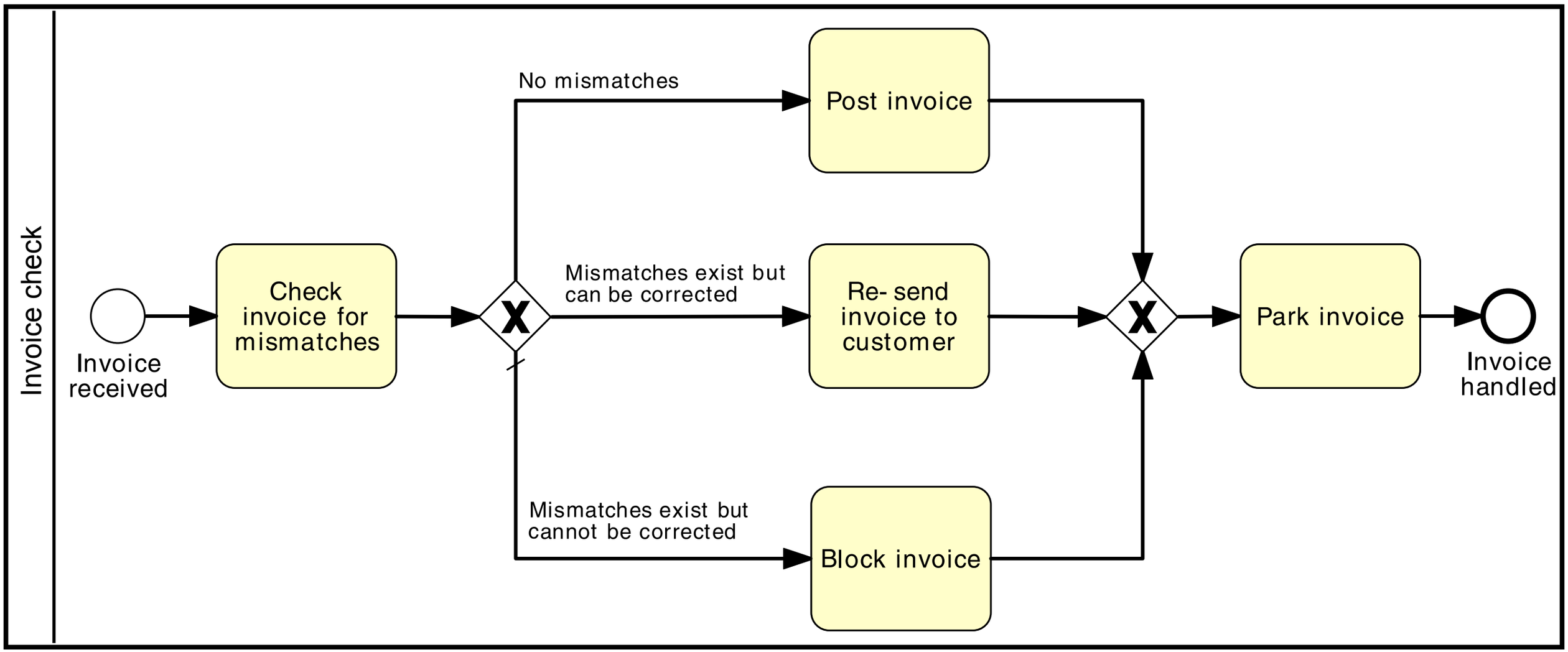
Draw the EPC diagram that corresponds to the BPMN diagram below





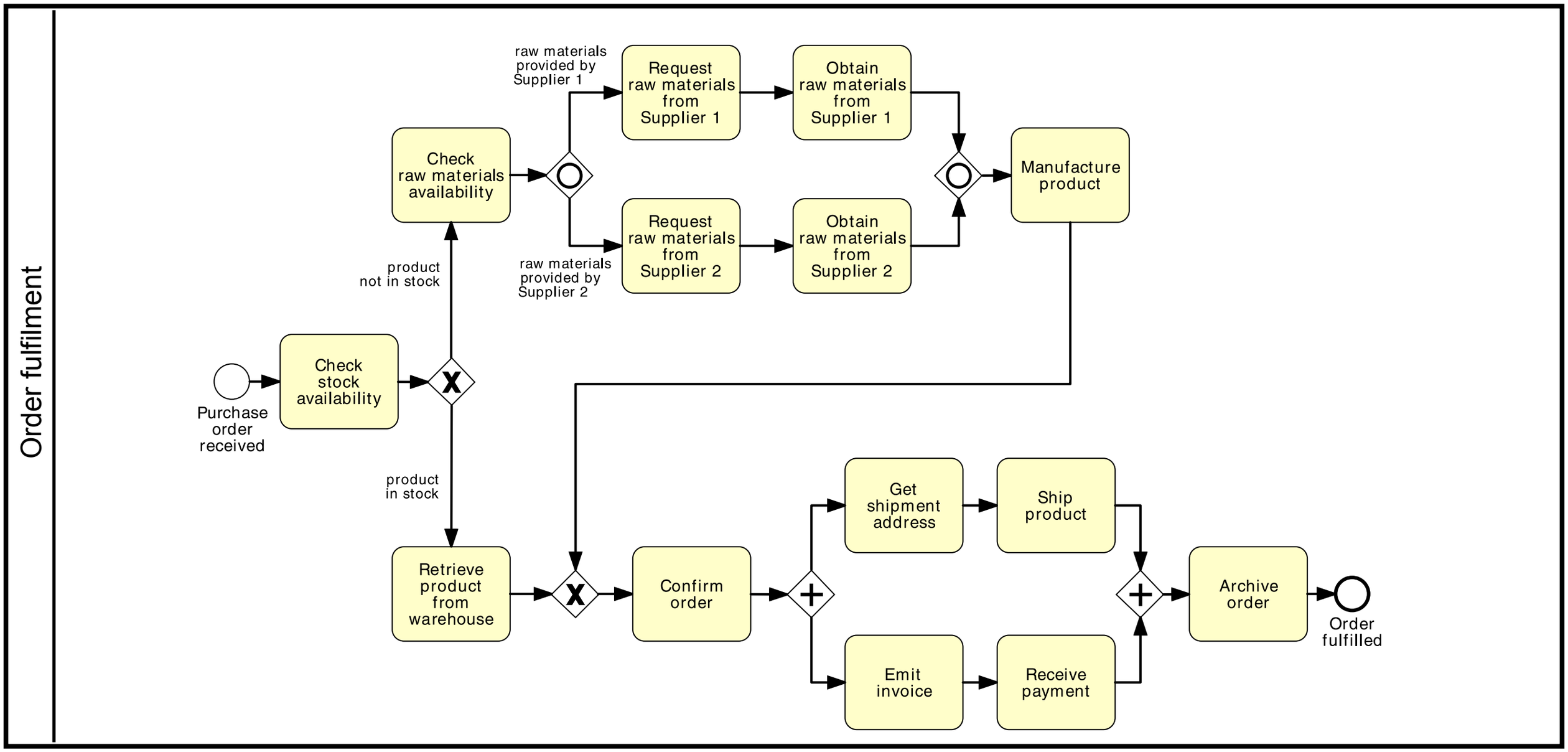
Exercise: basics

Draw the EPC diagram that corresponds to the BPMN diagram below



Exercise: basics

Draw the EPC diagram that corresponds to the BPMN diagram below



Exercises

Transfer the following verbal description into an EPC diagram

You are tasked with modeling the Customer Order Process of a small e-commerce company.
The process starts when a customer places an order online and ends when the order is successfully delivered.
The process must involve at least the following activities:
checking if the items are available in stock,
a notification to the customer if the items are not available,
the preparation of the order for shipment,
and the processing of the payment.

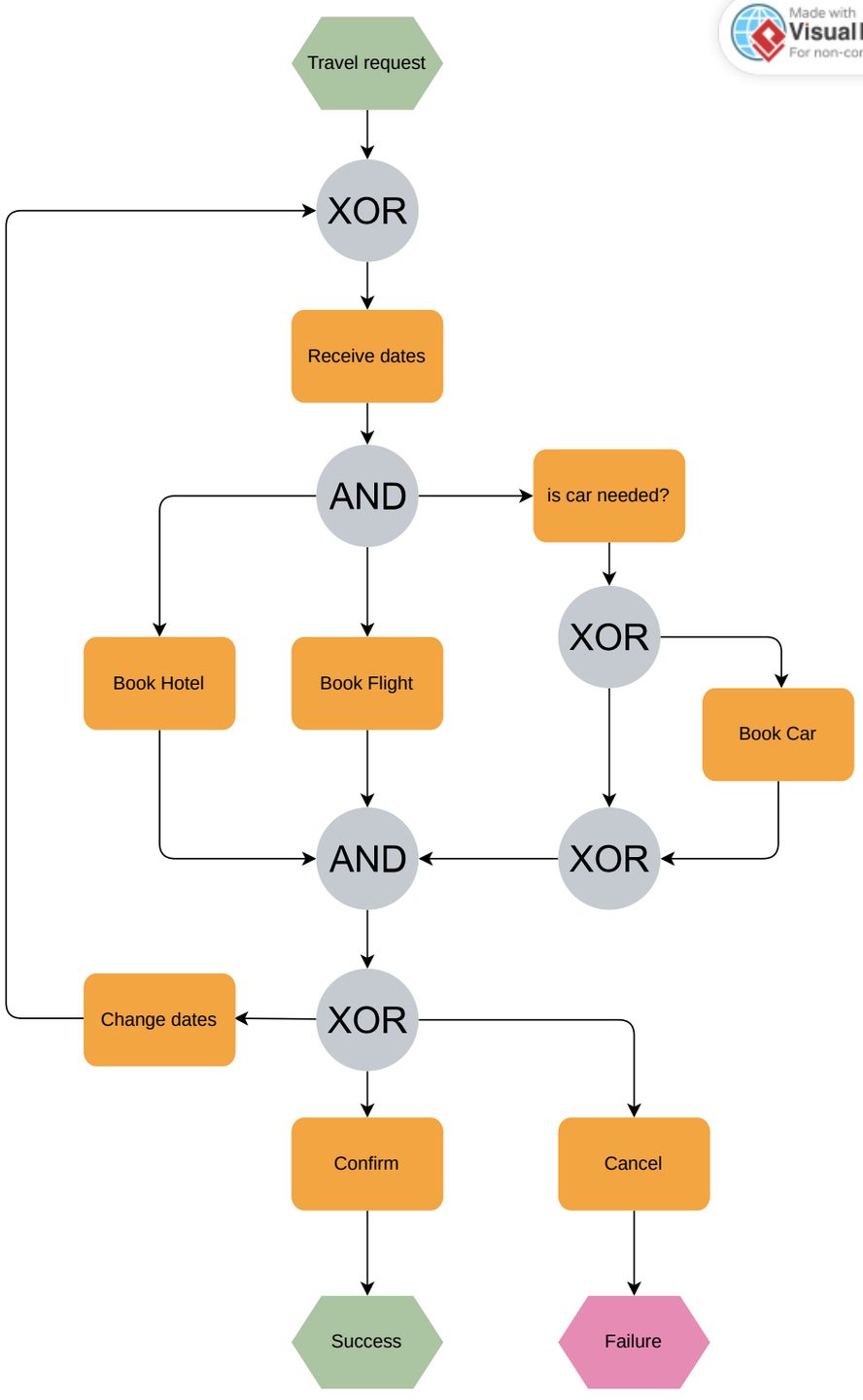


Send your solutions to: bruni@di.unipi.it

Exercises

Draw the BPMN diagram that corresponds to the EPC diagram below

Made with VisualParadigm
For non-commercial use



Send your solutions to:

bruni@di.unipi.it