

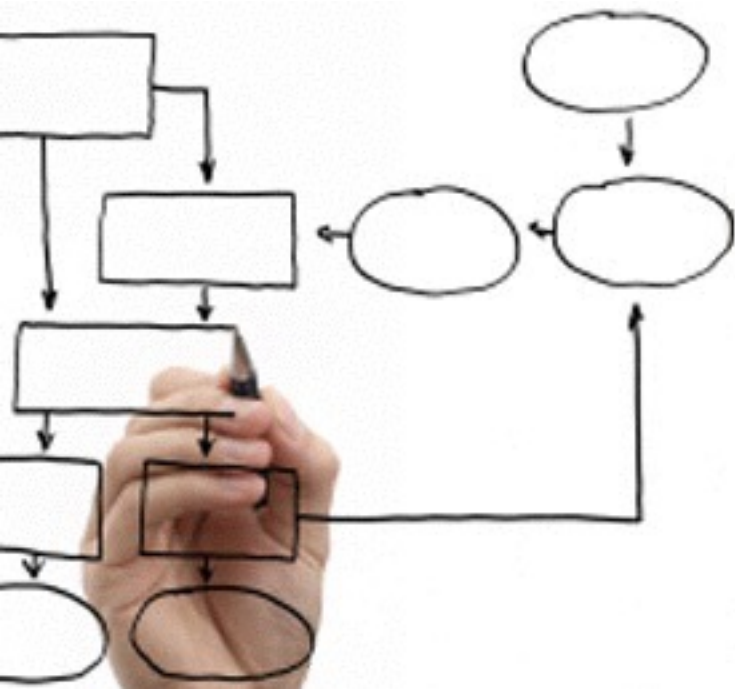
Methods for the specification and verification of business processes

MPB (6 cfu, 295AA)

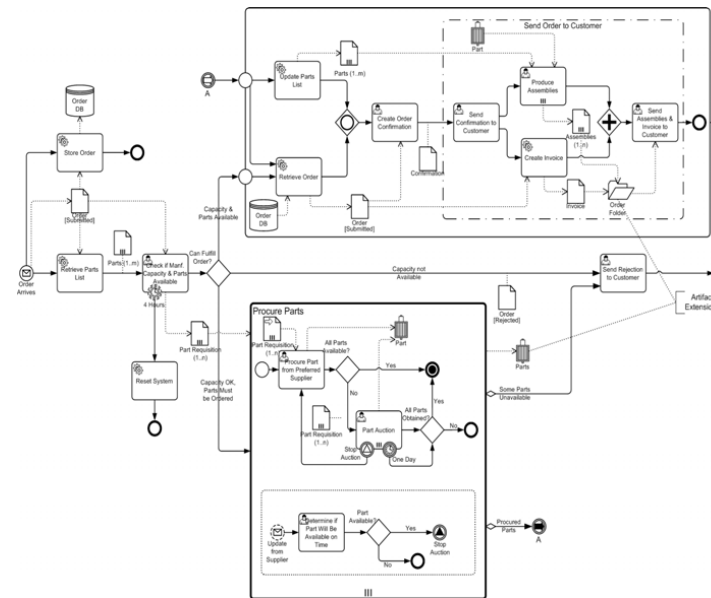
Roberto Bruni

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

21 - Business process
modelling notation



Object



We overview BPMN and their analysis
based on Petri nets

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.

Business Process Management Initiative

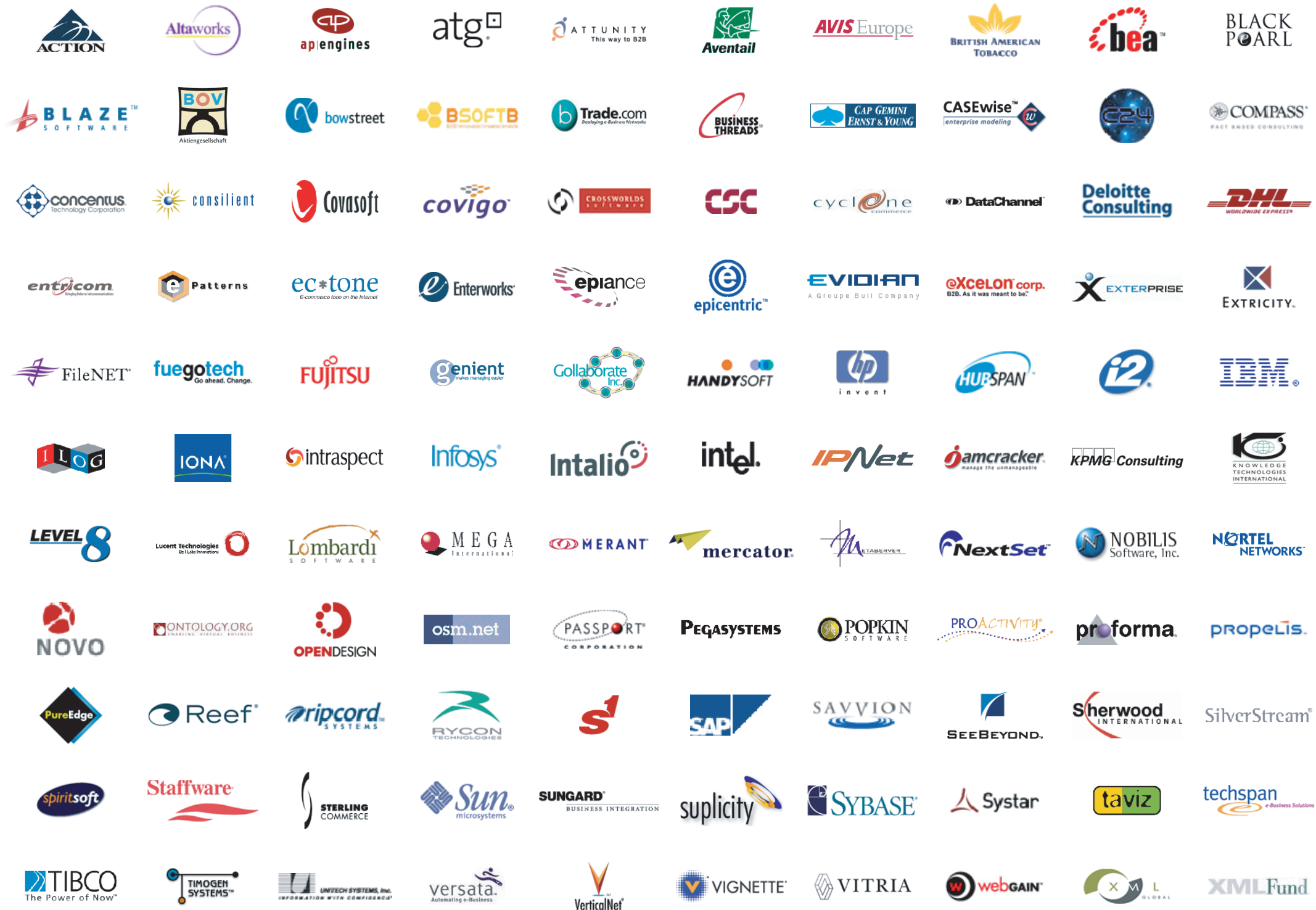
The **Business Process Management Initiative**

is an **independent organization** devoted to

the development of **open specifications**

for the management of **e-Business processes**
that span multiple applications, corporate departments, and
business partners, behind the firewall and over the Internet

The membership of the BPMI Notation Working Group represents a large segment of the BP modelling community



BMI-DTF

June 2005

The Business Process Management Initiative (BPMI.org)
and the Object Management Group™ (OMG™)
decided to merge their activities on
Business Process Management (BPM)
to provide thought leadership and industry standards for this
vital and growing industry.

The combined group has named itself the
Business Modeling & Integration Domain Task Force
(BMI -DTF)

Standardisation

The development of BPMN is an important step in

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

exploiting experiences with many divergent proposals to
consolidate the best ideas

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

reducing the confusion among business and IT end-users

Disclaim

Formal rigor and conciseness:
not primary concerns for BPMN specifications
(over 100 symbols, shorthands and alternative constructs
are often available)

The large number of object types
and their continuous evolution
makes it hard to define mappings
and to prove their consistency under all contexts

Inconsistencies and ambiguities in
BPMN standard are present but hard to detect

Business process diagram

BPMN defines a standard for
Business Process Diagrams (BPD)

based on **flowcharting technique**
tailored to graphical models of business process operations

Four basic categories of elements:

Swimlanes

Flow objects

Artefacts

Connecting objects

BPMN:

(some) key features

Key features of BPMN include:

sub-processes

exceptions

message flows

and also:

transactions

compensations

choreographies

...

Versioning

BPMN 1.0 approved 2006

BPMN 1.1 approved 2007

BPMN 1.2 approved 2009

BPMN 2.0 Beta 1 proposed 2009

BPMN 2.0 Beta 2 proposed 2010

BPMN 2.0 Final delivered 2011

BPMN 1.0 (2004/06)

Main goal:

provide a **notation** that is **readily understandable by all**
business users

from the **business analysts** who create initial drafts
of the processes

to the **technical developers** responsible for implementing
the technology that will perform those processes

to the **business people** who will manage those processes

BPMN 2.0 vs 1.0

Updated (new markers):

Tasks/SubProcesses

Events

Gateways

Artefacts

Added:

Choreographies

Full metamodel

XML Serialization

Diagram Interchange

BPMN Execution Semantics (verbal)

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.

The notation has been specifically designed to coordinate the sequence of processes and the messages that flow between different process 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

Who is BPMN targeted at?

BPMN is targeted at a **high level for business users** and at a **lower level for process implementers**.

The former should be able to easily read and understand a BPMN diagram.

The latter should be able to adorn a BPMN diagram with further details in order to represent the process in a physical implementation.

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

BPMN 2.0 (2009/11)

FAQ

What does this mean for UML users?

The unified modelling language (UML) takes an object-oriented approach to the modeling of applications, while BPMN takes a **process-oriented approach** to modelling of systems.

The BPMN and the UML are compatible with each other. Where BPMN has a focus on business processes, the UML has a focus on software design: the two are not competing notations but different views on systems.

BPMN 2.0 (2009/11)

FAQ

Will there be a major rewrite?

Not for 2 or 3 years...

(2015 and still no revision is planned)

Strong points 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 + choreography

Tool availability: exchange format

Weaknesses of BPMN

500 pages long (verbose) description

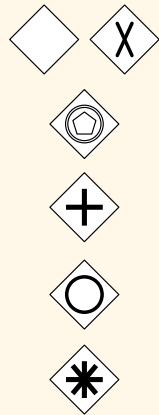
over 100 graphical elements

difficult to learn comprehensively,
with the risk of conflicting reading of the same diagram

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

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.

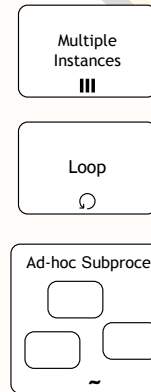
Inclusive Gateway

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

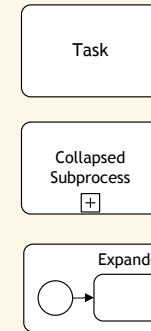
Loop Activity is iterated if a loop condition is true. The condition is either tested before or after the activity execution.

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.

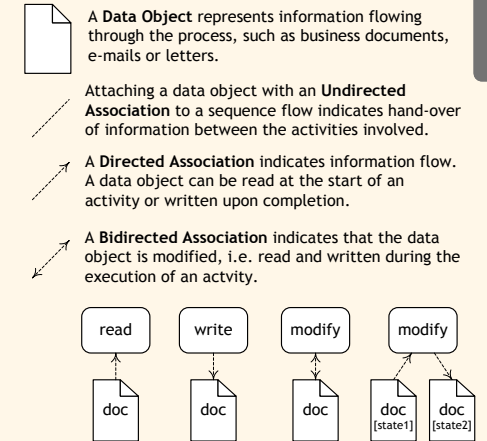


Task
A Task is a unit of work, the job to be performed.

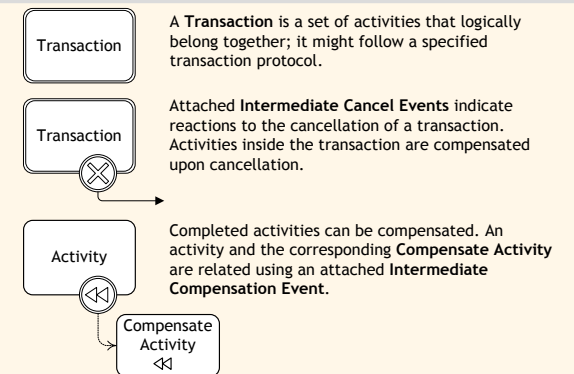
Collapsed Subprocess
A Subprocess is a decomposable activity. It can be collapsed to hide the details.

Expanded Subprocess
An Expanded Subprocess contains a valid BPMN diagram.

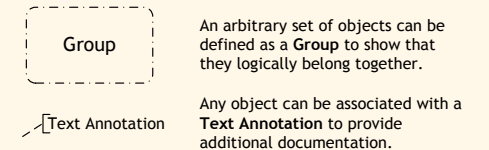
Data



Transactions



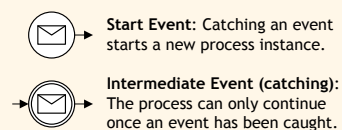
Documentation



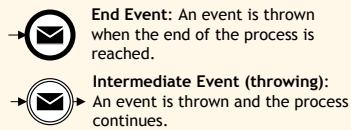
Events

	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.

Catching

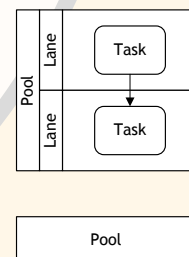


Throwing



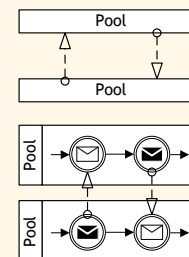
Attached Intermediate Event: The activity is aborted once an event is caught.

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.

Authors

Gero Decker
Alexander Grosskopf
Sven Wagner-Boysen

Business Process Diagram Graphical Objects

Events



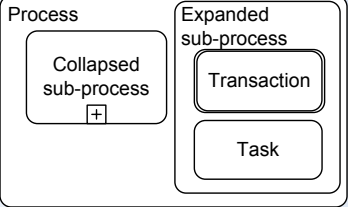
An event is something that »happens« during the process. These events affect the flow of the process and usually have a cause (trigger) or an impact (result).
Examples: 'Email received', '3 o'clock', 'Warehouse empty', 'Critical error',...

Event flow	Event type			Description
	Start	Intermediate	End	
General				The Start Event indicates where a particular process will start. Intermediate Events occur between a Start Event and an End Event. It will affect the flow of the process, but will not start or (directly) terminate the process. The End Event indicates where a process will end.
Message				A message arrives from a participant and triggers the Event. This causes process to {start, continue, end} if it was waiting for a message, or changes the flow if exception happens. End type of message event indicates that a message is sent to a participant at the conclusion of the process.
Timer				A specific time or cycle can be set that will trigger the start of the Process or continue the process. Intermediate timer can be used to model the time-based delays.
Error				This type of End indicates that a named Error should be generated. This Error will be caught by an Intermediate Event within the Event Context.
Cancel				This type of Event is used within a Transaction Sub-Process. This type of Event MUST be attached to the boundary of a Sub-Process. It SHALL be triggered if a Cancel End Event is reached within the Transaction Sub-Process.
Compensation				This is used for compensation handling--both setting and performing compensation. It calls for compensation if the Event is part of a Normal Flow. It reacts to a named compensation call when attached to the boundary of an activity. Very useful for modelling roll-back actions within the transaction.
Rule				This type of event is triggered when the conditions for a rule become true. Rules can be very useful to interrupt the loop process, for example: 'The number of repeats = N'. Intermediate rule is used only for exception handling.
Link				A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two Sub-Processes within the same parent Process. It can be used, for example, when the working area (page) is too small – go to another page.
Multiple				This type of event indicates that there are multiple ways of triggering the Process. Only one of them will be required to {start, continue, end} the Process.
Terminate				This type of End indicates that all activities in the Process should be immediately terminated. This includes all instances of Multi-Instances. The Process is terminated without compensation or event handling.

Activities



An activity is a generic type of work that a company performs. An activity can be atomic (task) or compound (process, sub-process).
Examples: 'Send a letter', 'Write a report', 'Calculate the interests',...



A task is used to represent the activity on the lowest abstraction level.



More information about the transaction and compensation attribute can be found under »Compensation Association«.

Task/Subprocess special attributes

Looping		The task or sub-process is repeated.
Ad Hoc		The tasks in the sub-process can not be connected with sequence flows at design time.
Multiple instances		Multiple instances of task or sub-process will be created.
Compensation		The symbol represents a compensation task or sub-process.

Artefacts



Artefacts are used to provide additional information about the process. If required, modellers and modelling tools are free to add new artefacts.
Examples of data objects: 'A letter', 'Email message', 'XML document', 'Confirmation',...

Set of standardized artefacts

Data object		Data objects provide information about what activities are required to be triggered and/or what they produce. They are considered as Artefacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process. The state of the data object should also be set.
Group		Grouping can be used for documentation or analysis purposes. Groups can also be used to identify the activities of a distributed transaction that is shown across Pools. Grouping of activities does not affect the Sequence or Message Flow.
Annotation		Text Annotations are a mechanism for a modeller to provide additional information for the reader of a BPMN Diagram.

Gateways



A gateway is used to split or merge multiple process flows. Thus it will determine branching, forking, merging and joining of paths. Examples: 'Condition true? – yes/no', 'Choose colour? – red/green/blue',...

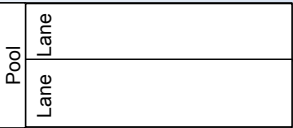
Gateway control types

XOR (DATA)			Data based exclusive decision or merging. Both symbols have equal meaning. See also Conditional flow.
XOR (EVENT)			Event based exclusive decision only.
OR			Data based inclusive decision or merging.
COM- PLEX			Complex condition (a combination of basic conditions)
AND			Parallel forking and joining (synchronization).

Swimlanes



Pools and lanes are used to represent organizations, roles, systems and responsibilities. Examples: 'University', 'Sales division', 'Warehouse', 'ERP system',...



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.

A **Pool** represents a participant in a process. It contains a business process and is used in B2B situations.

A **Lane** is a sub-partition within a pool used to organize and categorize activities.

Business Process Diagram Connecting Objects

Graphical connecting objects



There are three ways of connecting **Flow objects (Events, Activities, Gateways)** with each other or with other information – using sequence flows, message flows or associations.

Graphical connecting objects		
Normal sequence flow		A Sequence Flow is used to show the order In which the activities in a process will be performed.
Conditional sequence flow		A Sequence Flow can have condition expressions which are evaluated at runtime to determine whether or not the flow will be used.
Default sequence flow		For Data-Based Exclusive Decisions or Inclusive Decisions, one type of flow is the Default condition flow. This flow will be used only if all other outgoing conditional flows are NOT true at runtime.
Message flow		A Message Flow is used to show the flow of messages between two participants that are prepared to send and receive them. In BPMN, two separate Pools in a Diagram can represent the two participants.
Association		An Association (directed, non-directed) is used to associate information with Flow Objects. Text and graphical non-Flow Objects can be associated with Flow objects.

Sequence Flow and Message Flow rules

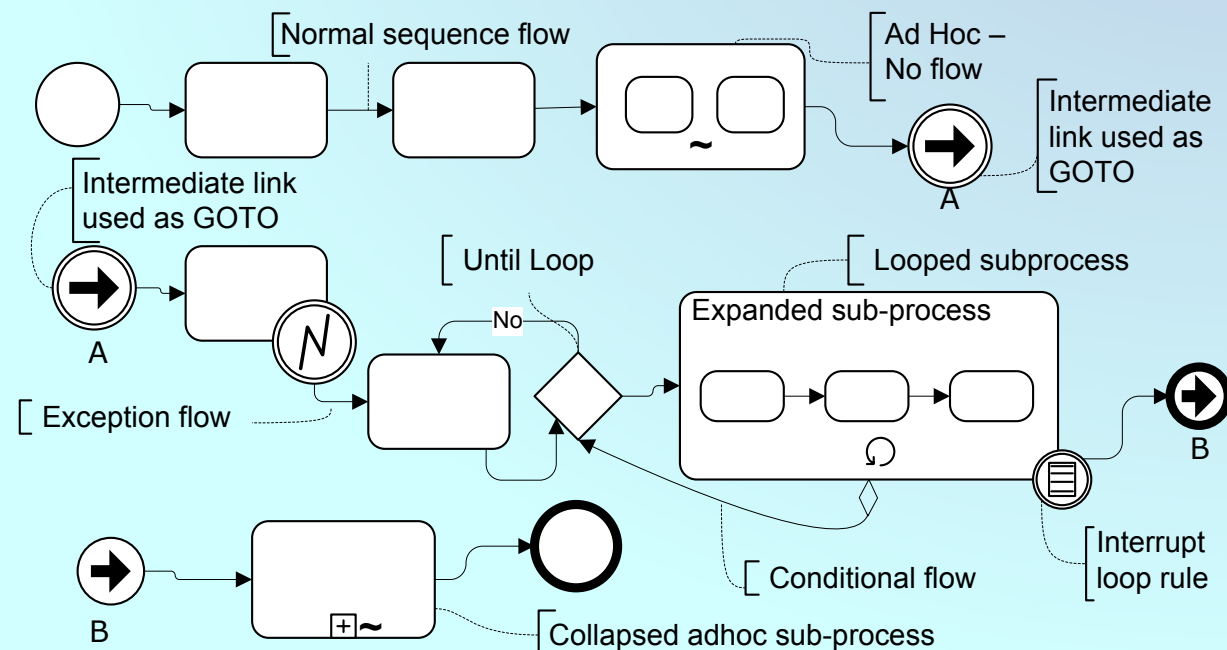
Only objects that can have an incoming and/or outgoing Sequence Flow / Message Flow are shown in the Tables Below.

		To:					
From:			→	→	→	→	→
			→	→	→	→	→
			→	→	→	→	→
			→	→	→	→	→
			→	→	→	→	→
		To:					
From:			→	→	→	→	→
			→	→	→	→	→
			→	→	→	→	→
			→	→	→	→	→
			→	→	→	→	→

Sequence flow mechanism

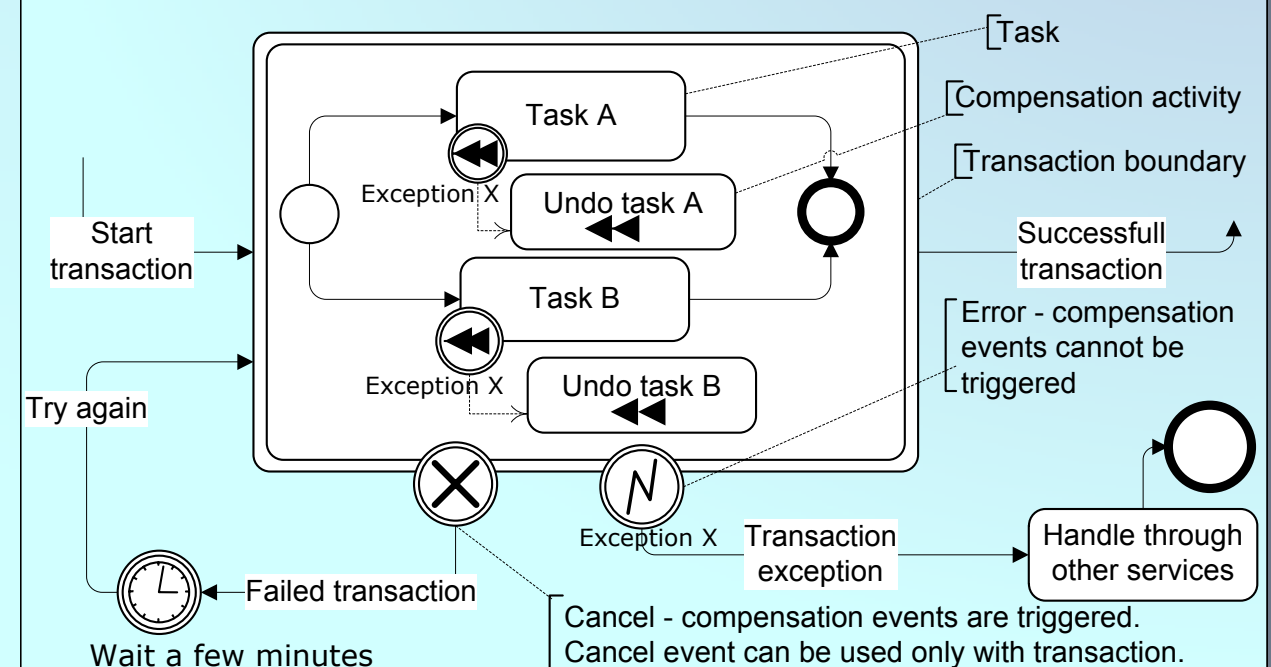


The Sequence Flow mechanisms is divided into types: Normal flow, Exception flow, Conditional flow, Link Events and Ad Hoc (no flow). Refer also to specific »Workflow Patterns«.

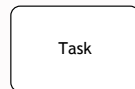


Compensation Association

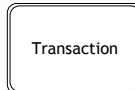
In case of transactions it is desired that all activities which constitute a transaction are finished successfully. Otherwise the transaction fails and rollback (compensation) activities occur which undo done activities.



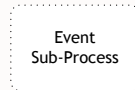
Activities



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.

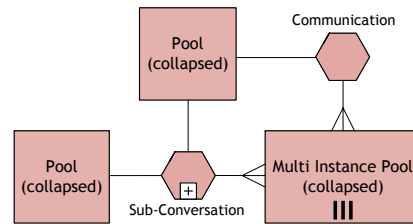


A **Conversation Link** connects Communications and Participants.

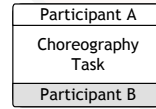


A **Forked Conversation Link** connects Communications and multiple Participants.

Conversation Diagram



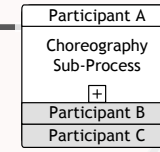
Choreographies



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

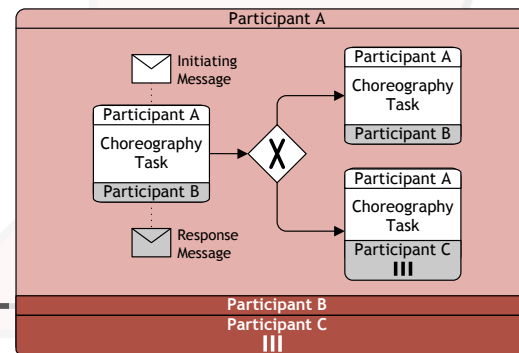
III

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

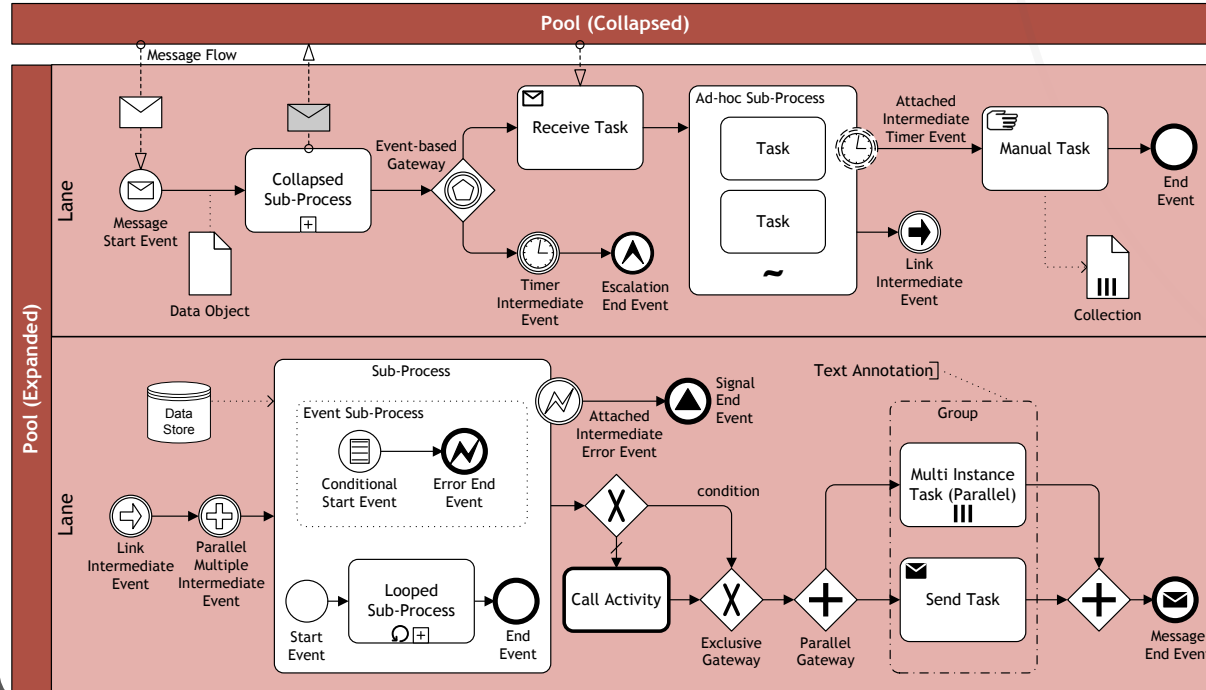


A **Choreography Sub-Process** contains a refined choreography with several Interactions.

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.

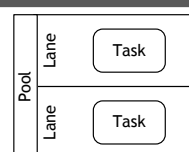


A **Message** is used to depict the contents of a communication between two Participants.

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.

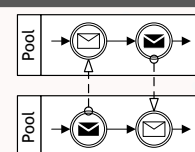
Swimlanes



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.



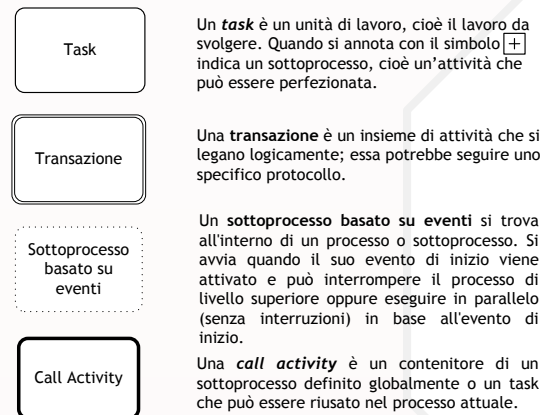
Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events.



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

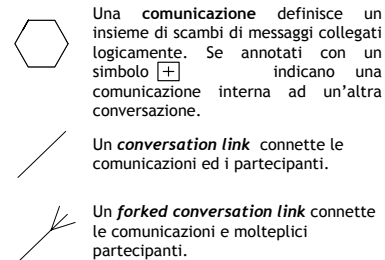
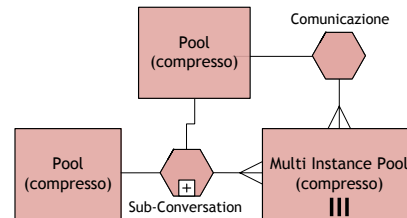


Diagramma di conversazione



Coreografie

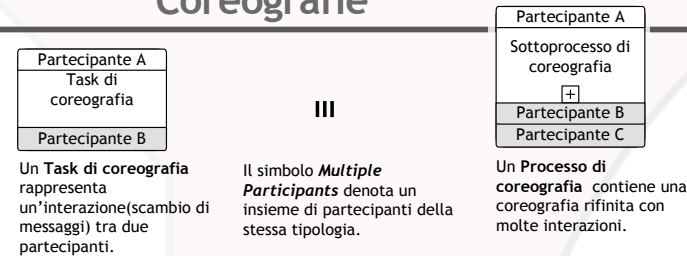
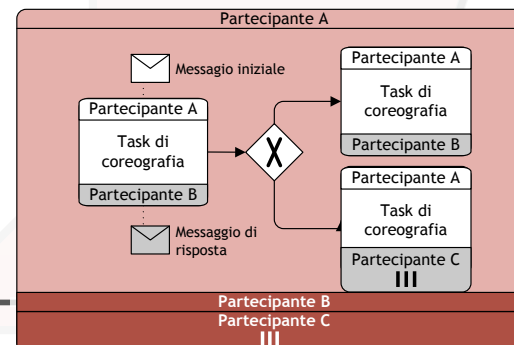
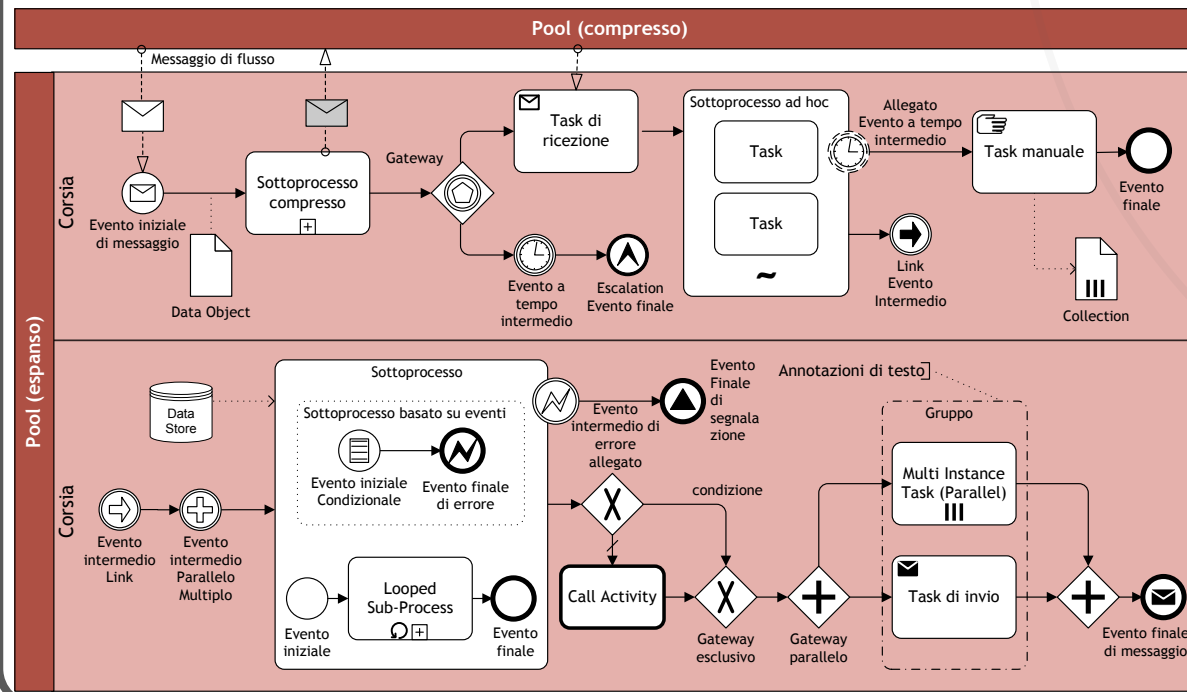


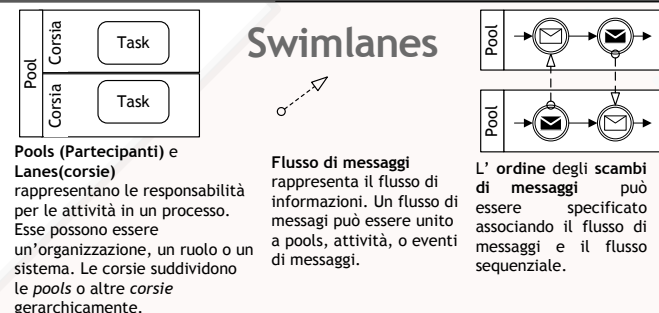
Diagramma di coreografia



Collaboration Diagram



Swimlanes



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.								
Parallelo 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 basics

Swimlanes

Swimlanes

Many process modelling methodologies utilise the concept of a **swimlane** as a mechanism to **organise activities into separate visual categories** in order to illustrate different functional capabilities or responsibilities

BPMN supports two main swimlane objects:

Pool

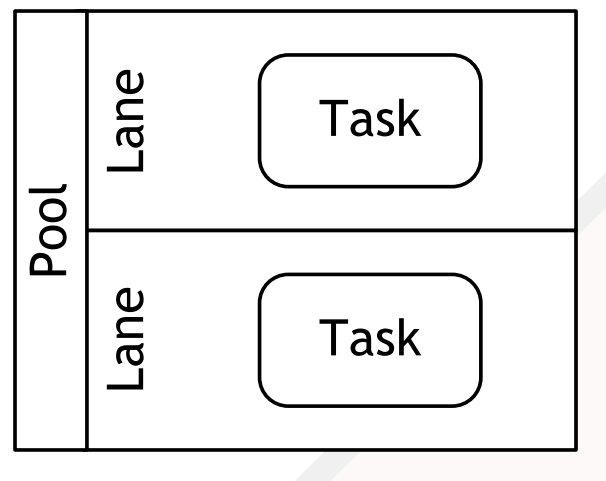
Lane

Pool and Lanes

A **pool** represents a participant (or role) in a process

A pool is represented as a rectangle with a name

A **lane** is a hierarchical sub-partition within a pool that is used to organise and categorise activities



A lane is an inner rectangle to the pool that extends to the entire length of the pool

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

Swimlanes



Pools and lanes are used to represent organizations, roles, systems and responsibilities. Examples: 'University', 'Sales division', 'Warehouse', 'ERP system',...



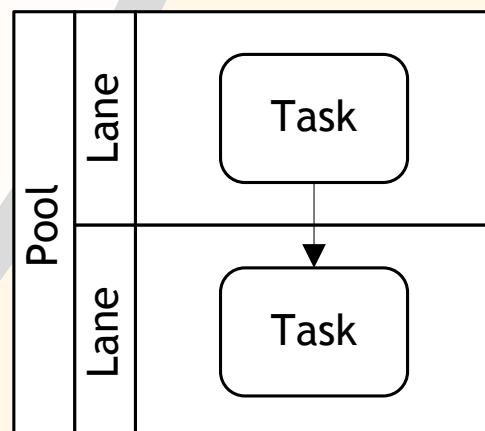
A Pool MUST contain 0 or 1 business process.

A Pool can contain 0 or more lanes.

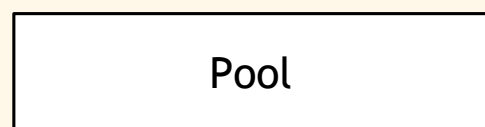
Two pools can only be connected with message flows.

A **Pool** represents a participant in a process. It contains a business process and is used in B2B situations.

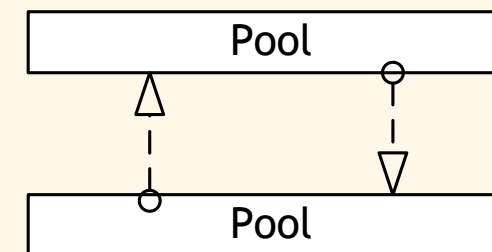
A **Lane** is a sub-partition within a pool used to organize and categorize activities.



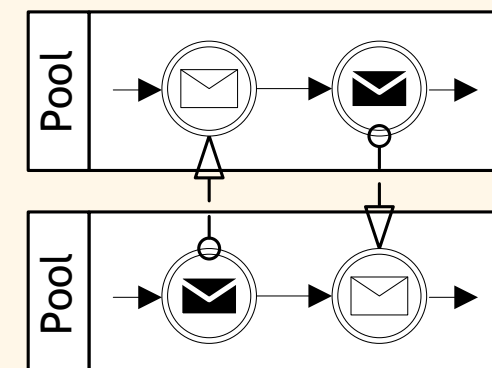
Pools and **Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes sub-divide pools or other lanes hierarchically.



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.

BPMN basics

Flow Objects

Flow objects

Theory:

fix a small set of core elements
so that modellers do not have to learn and recognise
a large number of different shapes:

Events

Activities

Gateways

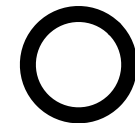
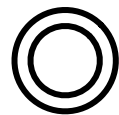
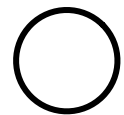
Practice:

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

Event

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

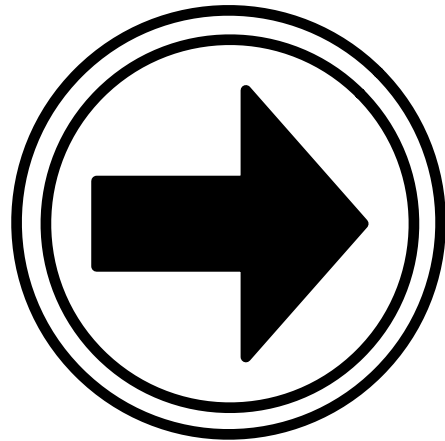
The type of an event is one among:
start, intermediate, end



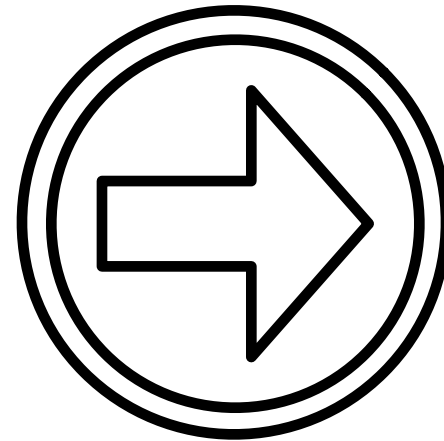
An event is represented as a circle
its type depends on the style of the border
(thin, double, thick)

An event can have a cause (**trigger**) or an impact (**result**)
Internal markers denote the trigger or result

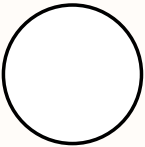
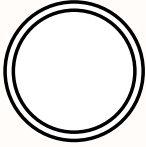
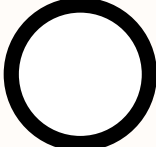








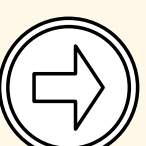
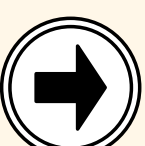
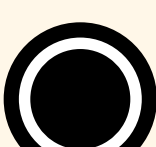
Off page connectors (printing / readability)



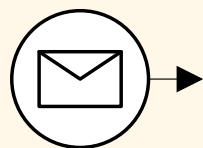
throw
(to page)



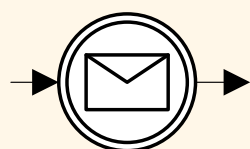
catch
(contd.)

	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.
Link					Off-page connectors. Two corresponding link events equal a sequence flow.
Terminate					Triggering the immediate termination of a process.

Catching

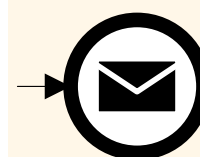


Start Event: Catching an event starts a new process instance.

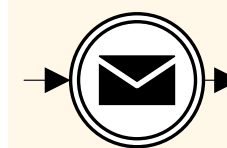


Intermediate Event (catching): The process can only continue once an event has been caught.

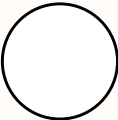
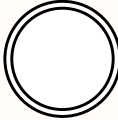




















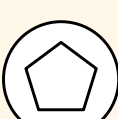
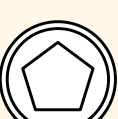





Throwing



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.

	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.
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.

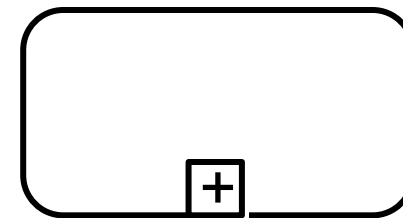
Events

Event flow		Event type		Description		Top-Level	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	
Start	Intermediate	End											
General				The Start Event indicates where a particular process will start. Intermediate Events occur between a Start Event and an End Event. It will affect the flow of the process, but will not start or (directly) terminate the process. The End Event indicates where a process will end.	None: Untyped events, indicate start point, state changes or final states.								
Message				A message arrives from a participant and triggers the Event. This causes process to {start, continue, end} if it was waiting for a message, or changes the flow if exception happens. End type of message event indicates that a message is sent to a participant at the conclusion of the process.	Message: Receiving and sending messages.								
Timer				A specific time or cycle can be set that will trigger the start of the Process or continue the process. Intermediate timer can be used to model the time-based delays.	Timer: Cyclic timer events, points in time, time spans or timeouts.								
Error				This type of End indicates that a named Error should be generated. This Error will be caught by an Intermediate Event within the Event Context.	Escalation: Escalating to an higher level of responsibility.								
Cancel				This type of Event is used within a Transaction Sub-Process. This type of Event MUST be attached to the boundary of a Sub-Process. It SHALL be triggered if a Cancel End Event is reached within the Transaction Sub-Process.	Conditional: Reacting to changed business conditions or integrating business rules.								
Compensation				This is used for compensation handling--both setting and performing compensation. It calls for compensation if the Event is part of a Normal Flow. It reacts to a named compensation call when attached to the boundary of an activity. Very useful for modelling roll-back actions within the transaction.	Link: Off-page connectors. Two corresponding link events equal a sequence flow.								
Rule				This type of event is triggered when the conditions for a rule become true. Rules can be very useful to interrupt the loop process, for example: 'The number of repeats = N'. Intermediate rule is used only for exception handling.	Error: Catching or throwing named errors.								
Link				A Link is a mechanism for connecting the end (Result) of one Process to the start (Trigger) of another. Typically, these are two Sub-Processes within the same parent Process. It can be used, for example, when the working area (page) is too small – go to another page.	Cancel: Reacting to cancelled transactions or triggering cancellation.								
Multiple				This type of event indicates that there are multiple ways of triggering the Process. Only one of them will be required to {start, continue, end} the Process.	Compensation: Handling or triggering compensation.								
Terminate				This type of End indicates that all activities in the Process should be immediately terminated. This includes all instances of Multi-Instances. The Process is terminated without compensation or event handling.	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.								

Activity

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

An activity can be atomic (**task**) or compound (**sub-process**)



An activity is represented as a rounded box,
Suitable markers are used to indicate
the nature of the action to be performed (**task type**)
and the execution behaviour (**activity marker**)

Events vs Activities

Events are instantaneous

Activities take time (have a duration)

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

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)

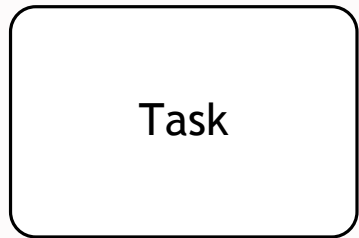
Avoid long labels
Articles are often omitted


Sub-processes

Process models tend to be too large
to be understood at once

Hiding certain parts within sub-processes
we improve readability

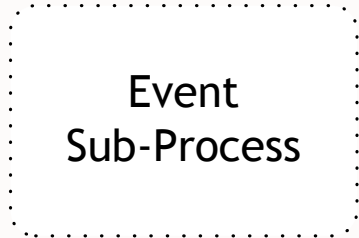
A **sub-process** is a self-contained, composite activity
that can be broken into smaller units of work



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

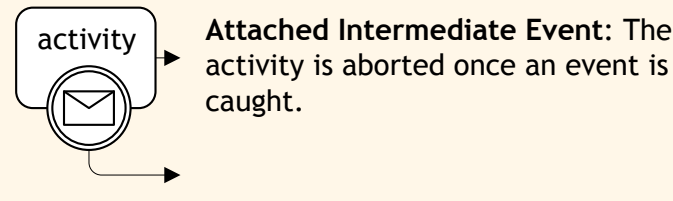
Markers indicate execution behavior of activities:

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

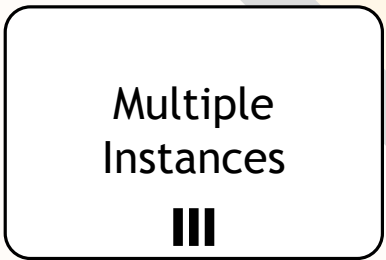
Task Types

Types specify the nature of the action to be performed:

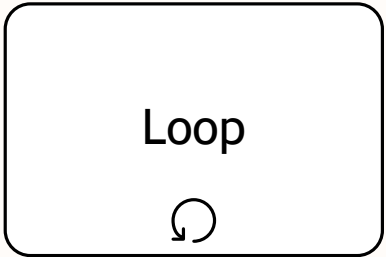
-  Send Task
-  Receive Task
-  User Task
-  Manual Task
-  Business Rule Task
-  Service Task
-  Script Task



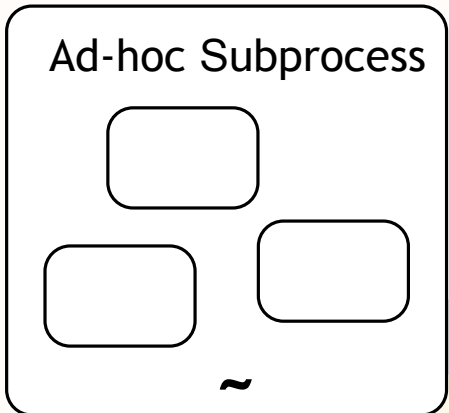
Attached Intermediate Event: The activity is aborted once an event is caught.



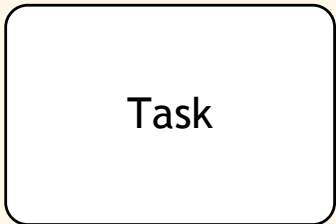
Multiple Instances of the same activity are started in parallel or sequentially, e.g. for each line item in an order.



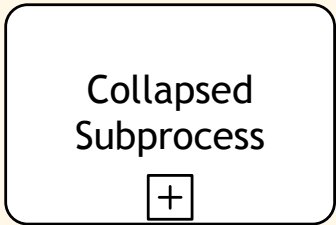
Loop Activity is iterated if a loop condition is true. The condition is either tested before or after the activity execution.



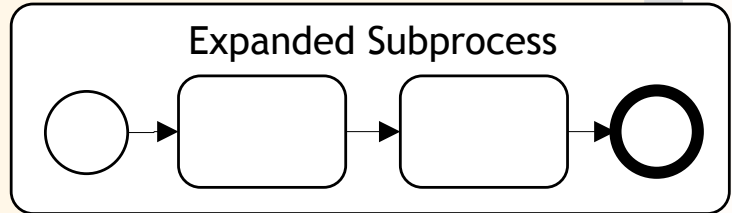
Ad-hoc Subprocesses contain tasks only. Each task can be executed arbitrarily often until a completion condition is fulfilled.



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



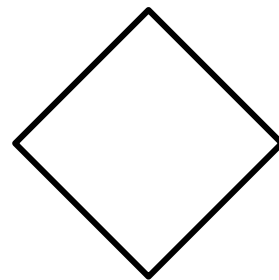
A **Subprocess** is a decomposable activity. It can be collapsed to hide the details.



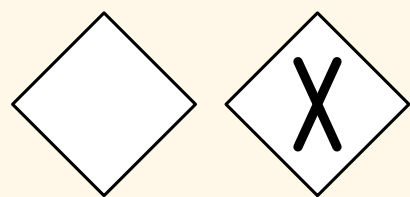
An **Expanded Subprocess** contains a valid BPMN diagram.

Gateway

A **gateway** is used to control the splitting and joining of paths in the sequence flow
(conditional, fork, wait)

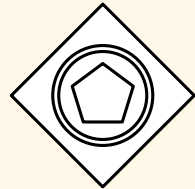


A gateway is represented as a diamond shape
Suitable markers are used to indicate
the nature of behaviour control



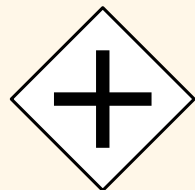
Data-based Exclusive Gateway

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.



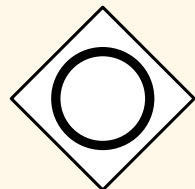
Event-based Exclusive Gateway

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



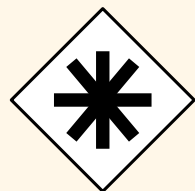
Parallel Gateway

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

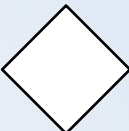



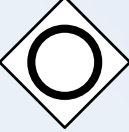

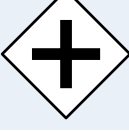
When splitting, one or more branches are activated based on branching conditions. When merging, it awaits all active incoming branches to complete.

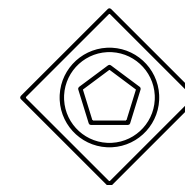


Complex Gateway

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

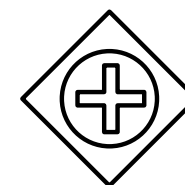
Gateway control types

XOR (DATA)	 	Data based exclusive decision or merging. Both symbols have equal meaning. See also Conditional flow. 
XOR (EVENT)		Event based exclusive decision only.
OR		Data based inclusive decision or merging.
COM- PLEX		Complex condition (a combination of basic conditions)
AND		Parallel forking and joining (synchronization).



Exclusive Event-based Gateway (instantiate)

Each occurrence of a subsequent event starts a new process instance.



Parallel Event-based Gateway (instantiate)

The occurrence of all subsequent events starts a new process instance.

BPMN basics

Artefacts

Artefacts

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

Any number of artefacts can be added to a diagram as appropriate for the specific context of the business process being modelled

BPMN includes three pre-defined types of artefacts:

Data object

Group

Text annotation

Data object

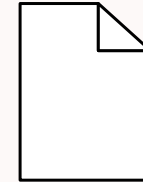
A **data object** specifies the data that are required or produced by an activity

A data object is often represented by the usual file icon

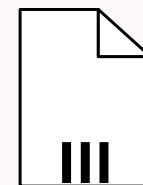


A **Data Input** is an external input for the entire process. It can be read by an activity.

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.

Group

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



A group is represented by rounded corner rectangles with dashed lines

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

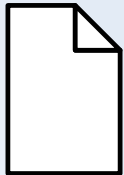

Artefacts

Artefacts are used to provide additional information about the process. If required, modellers and modelling tools are free to add new artefacts.

Examples of data objects: 'A letter', 'Email message', 'XML document', 'Confirmation',...



Set of standardized artefacts

Data object	 [state]	Data objects provide information about what activities are required to be triggered and/or what they produce. They are considered as Artefacts because they do not have any direct effect on the Sequence Flow or Message Flow of the Process. The state of the data object should also be set.
Group		Grouping can be used for documentation or analysis purposes. Groups can also be used to identify the activities of a distributed transaction that is shown across Pools. Grouping of activities does not affect the Sequence or Message Flow.
Annotation	[Description]	Text Annotations are a mechanism for a modeller to provide additional information for the reader of a BPMN Diagram.

BPMN basics

Connecting objects

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
Message flow
Association

Sequence flow

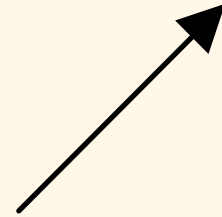
A **sequence flow** is used to show the order in which activities are to be performed

Note: connected objects must reside in the same pool
(but they can be in different lanes)
the term “control flow” is generally avoided in BPMN

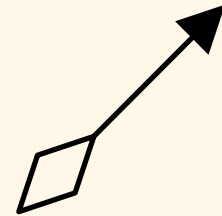


A sequence flow is represented by
a solid line with a solid arrowhead

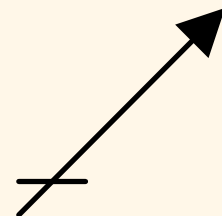
read as
``otherwise''



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.

Message flow

A **message flow** is used to show the flow of messages between two separate process participants (business entities or business roles) that send and receive them

Note: in BPMN the participants reside in separate pools



A message flow is represented by a dashed line with a open arrowheads (see above)

Sequence Flow and Message Flow rules

Only objects that can have an incoming and/or outgoing Sequence Flow / Message Flow are shown in the Tables Below.

		To:					
From:							

		To:					
From:							

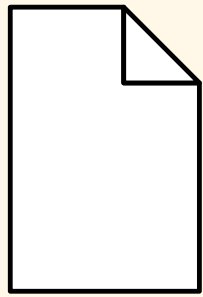
Association

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

Note: in particular, input and output of activities



An association is represented by
a dotted line with a line arrowhead

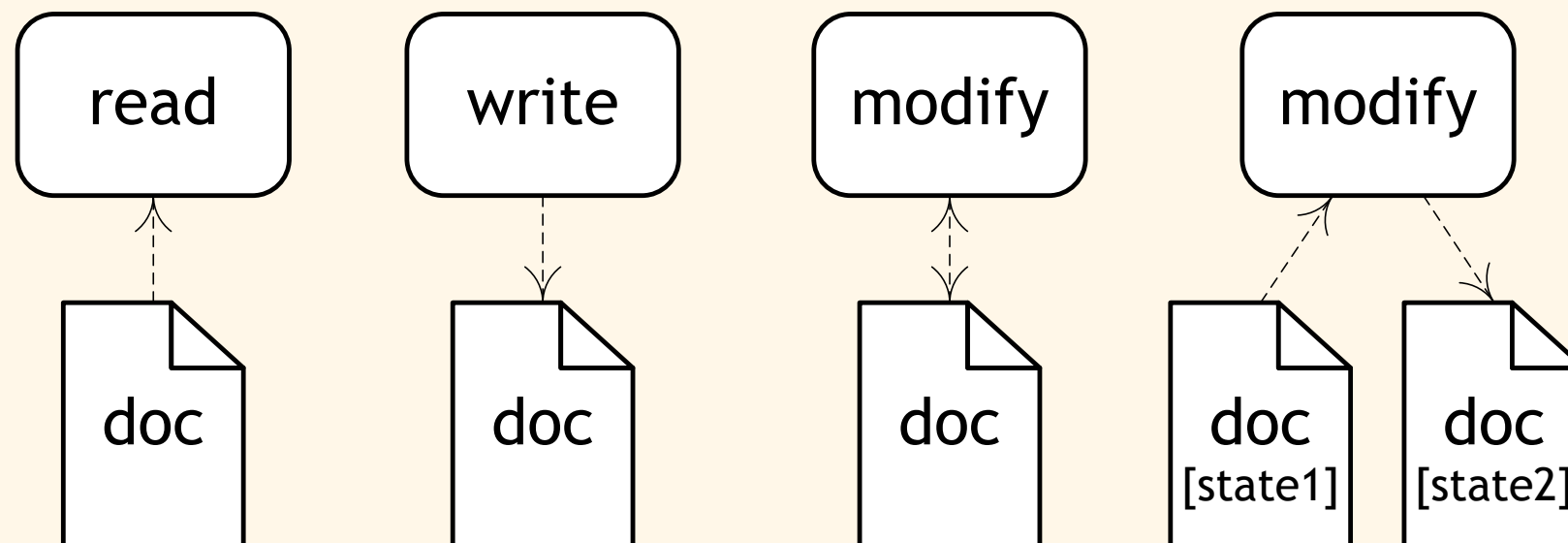


A **Data Object** represents information flowing through the process, such as business documents, e-mails or letters.

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.

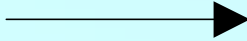
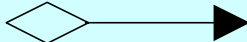
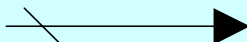
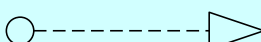
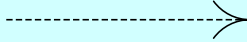
A **Bidirected Association** indicates that the data object is modified, i.e. read and written during the execution of an activity.



Graphical connecting objects

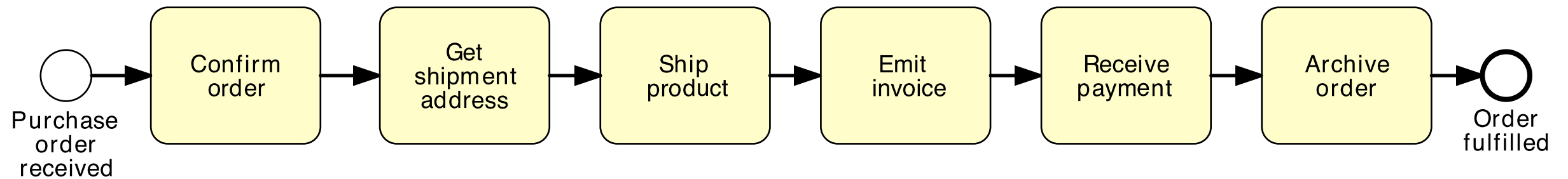


There are three ways of connecting **Flow objects (Events, Activities, Gateways)** with each other or with other information – using sequence flows, message flows or associations.

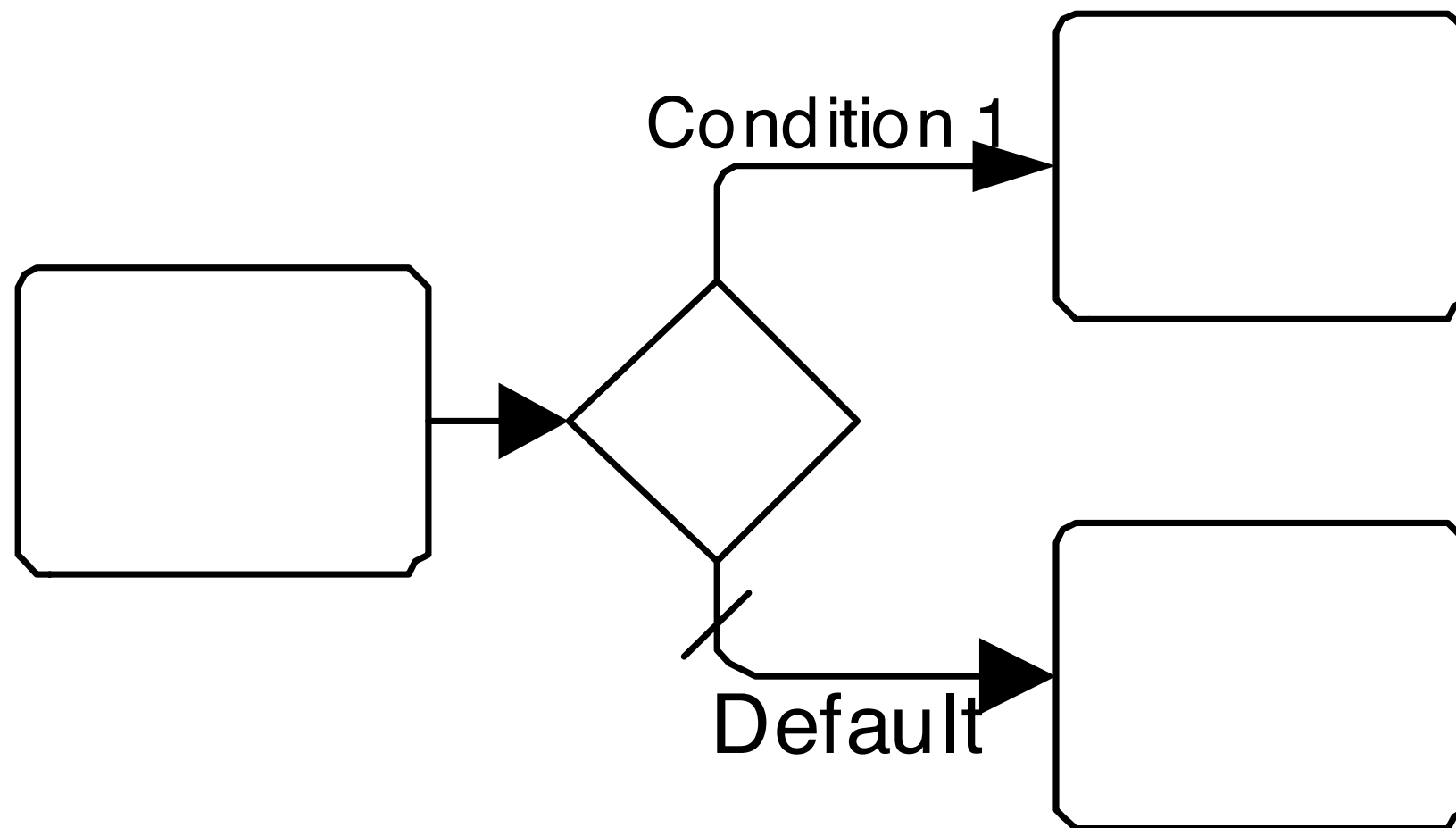
Graphical connecting objects		
Normal sequence flow		A Sequence Flow is used to show the order In which the activities in a process will be performed.
Conditional sequence flow		A Sequence Flow can have condition expressions which are evaluated at runtime to determine whether or not the flow will be used.
Default sequence flow		For Data-Based Exclusive Decisions or Inclusive Decisions, one type of flow is the Default condition flow. This flow will be used only if all other outgoing conditional flows are NOT true at runtime.
Message flow		A Message Flow is used to show the flow of messages between two participants that are prepared to send and receive them. In BPMN, two separate Pools in a Diagram can represent the two participants.
Association		An Association (directed, non-directed) is used to associate information with Flow Objects. Text and graphical non-Flow Objects can be associated with Flow objects.

A few patterns

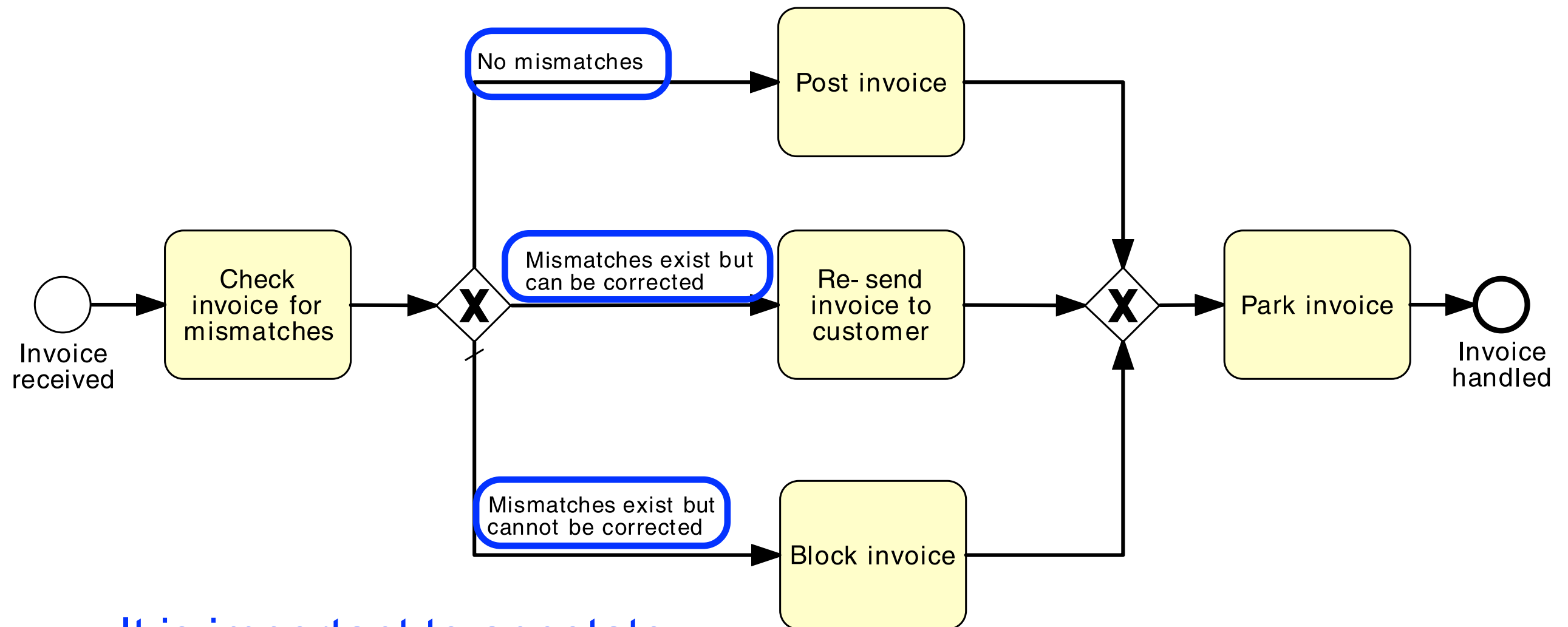
Sequence: order fulfillment



Exclusive decisions (exactly one)

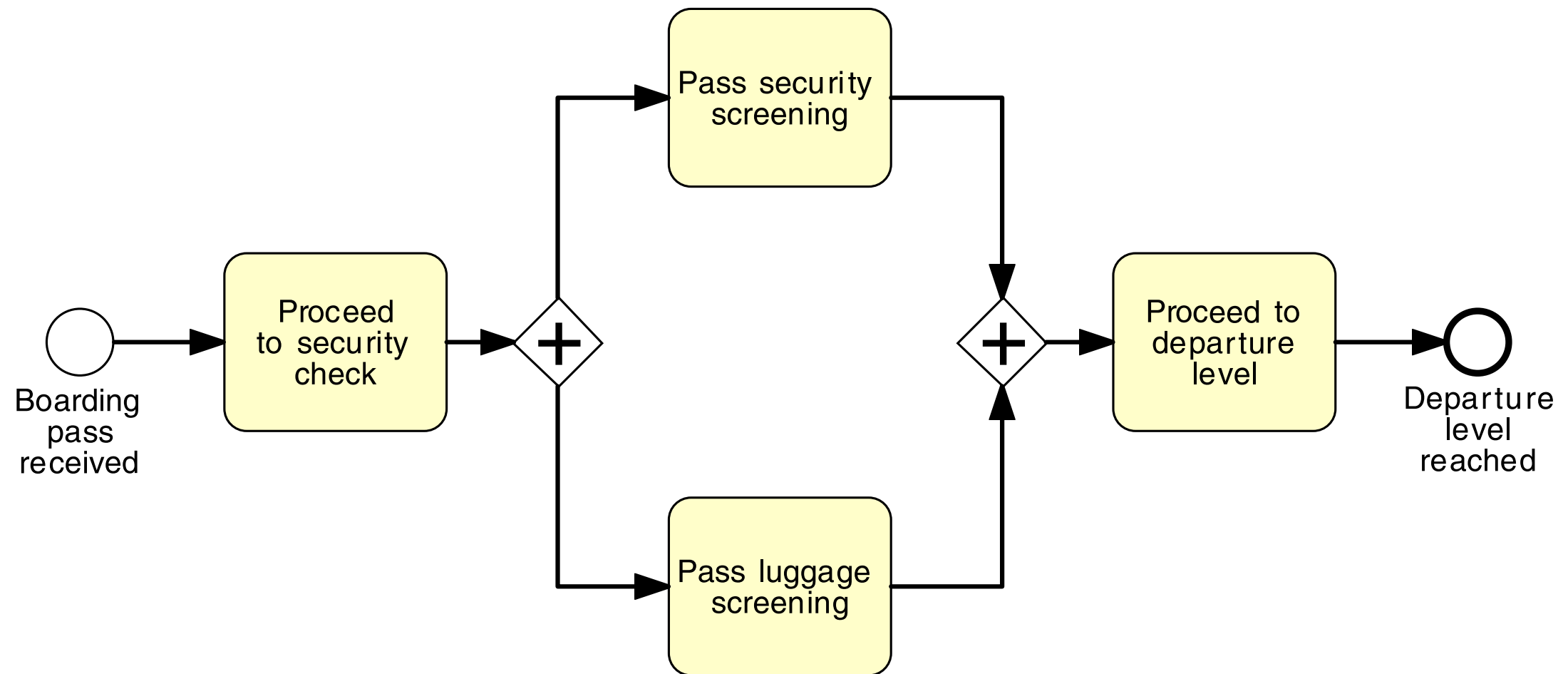


Exclusive decisions: invoice checking process

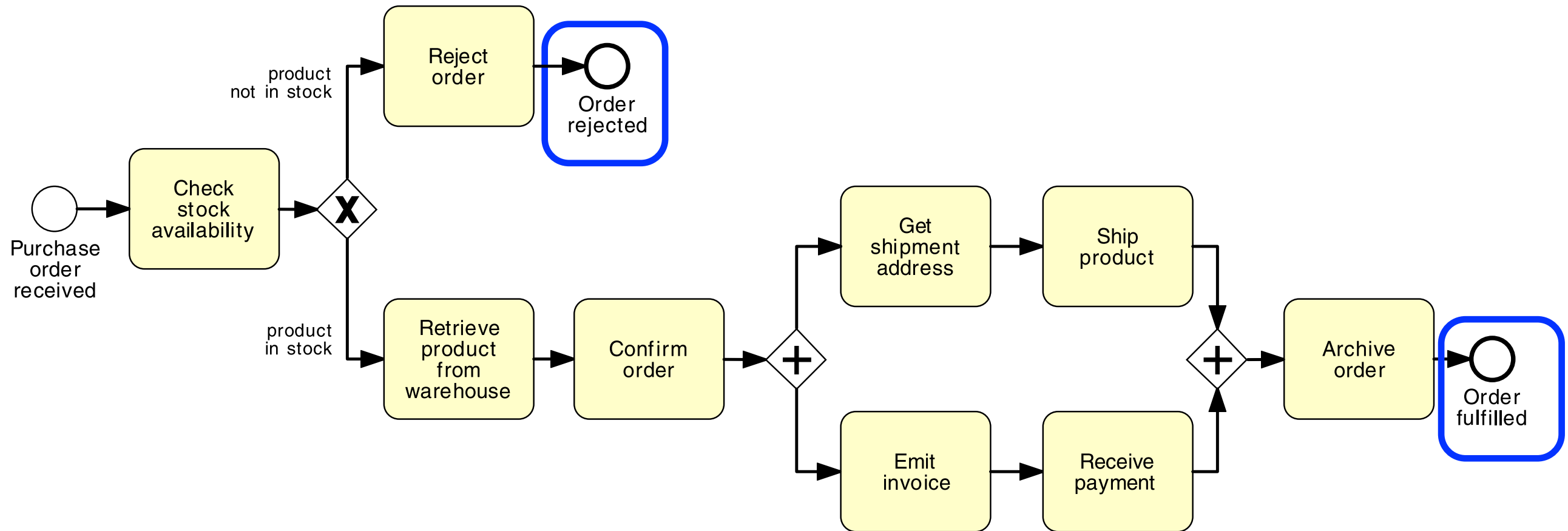


It is important to annotate branches with the conditions under which they are taken

Parallel activities: airport security check



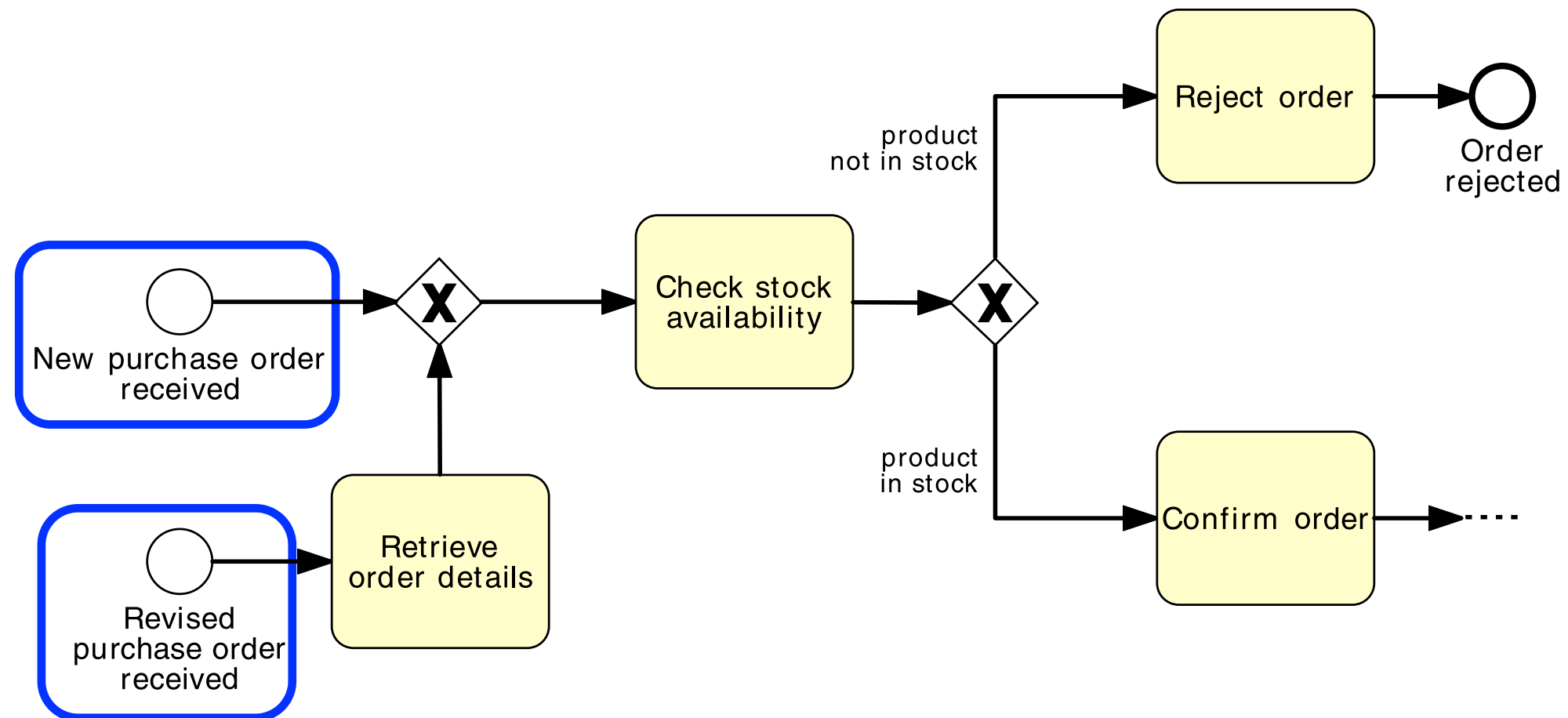
XOR + AND: order fulfillment



Multiple end events are often considered as a convenient notation (they are mutually exclusive in the example)

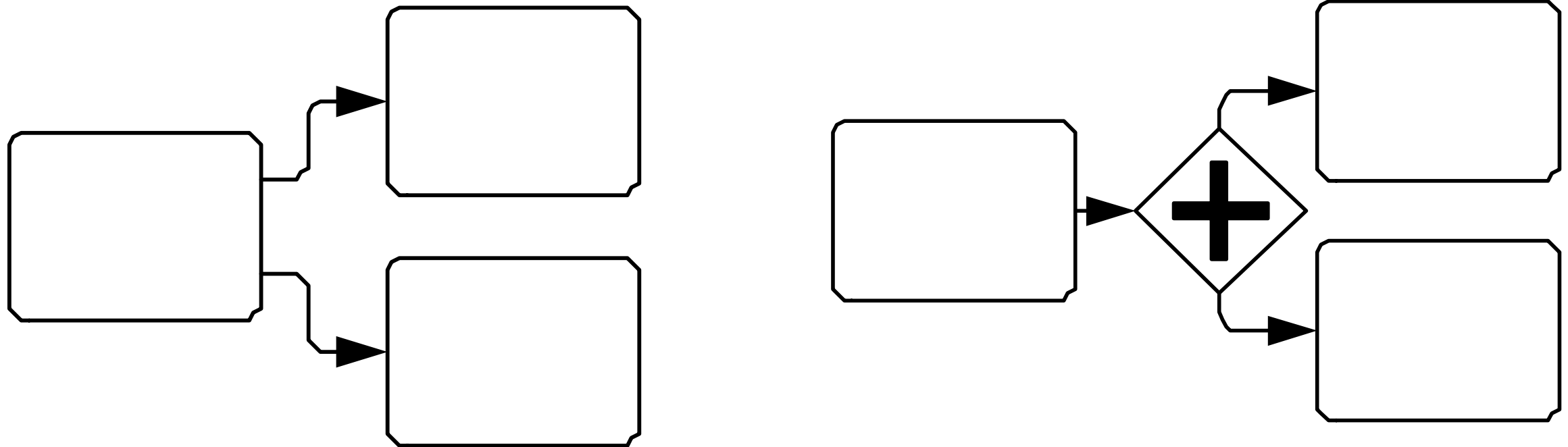
BPMN adopts **implicit termination** semantics:
a case ends only when each ``token'' reaches the end

Multiple start events: order fulfillment



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

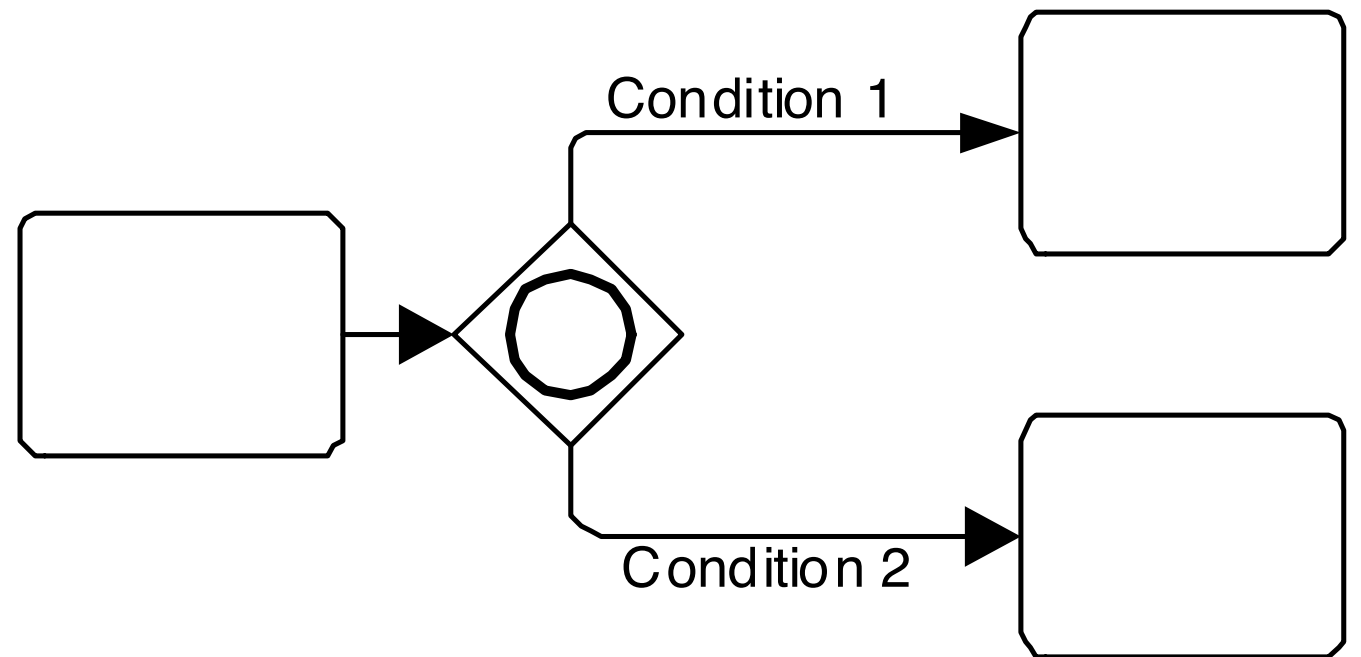
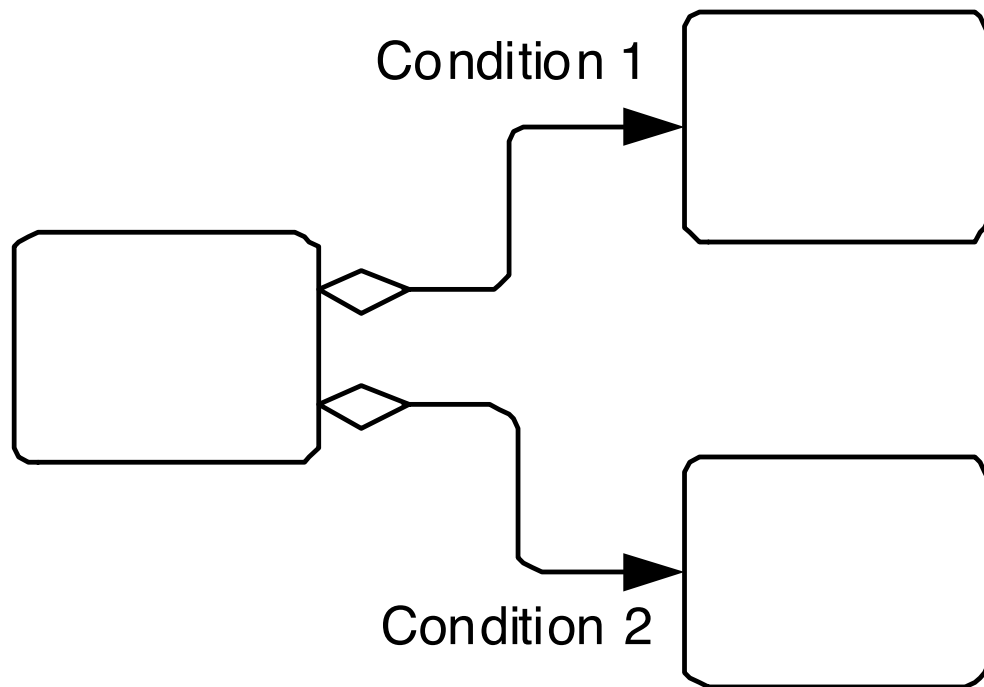
Omitting gateways



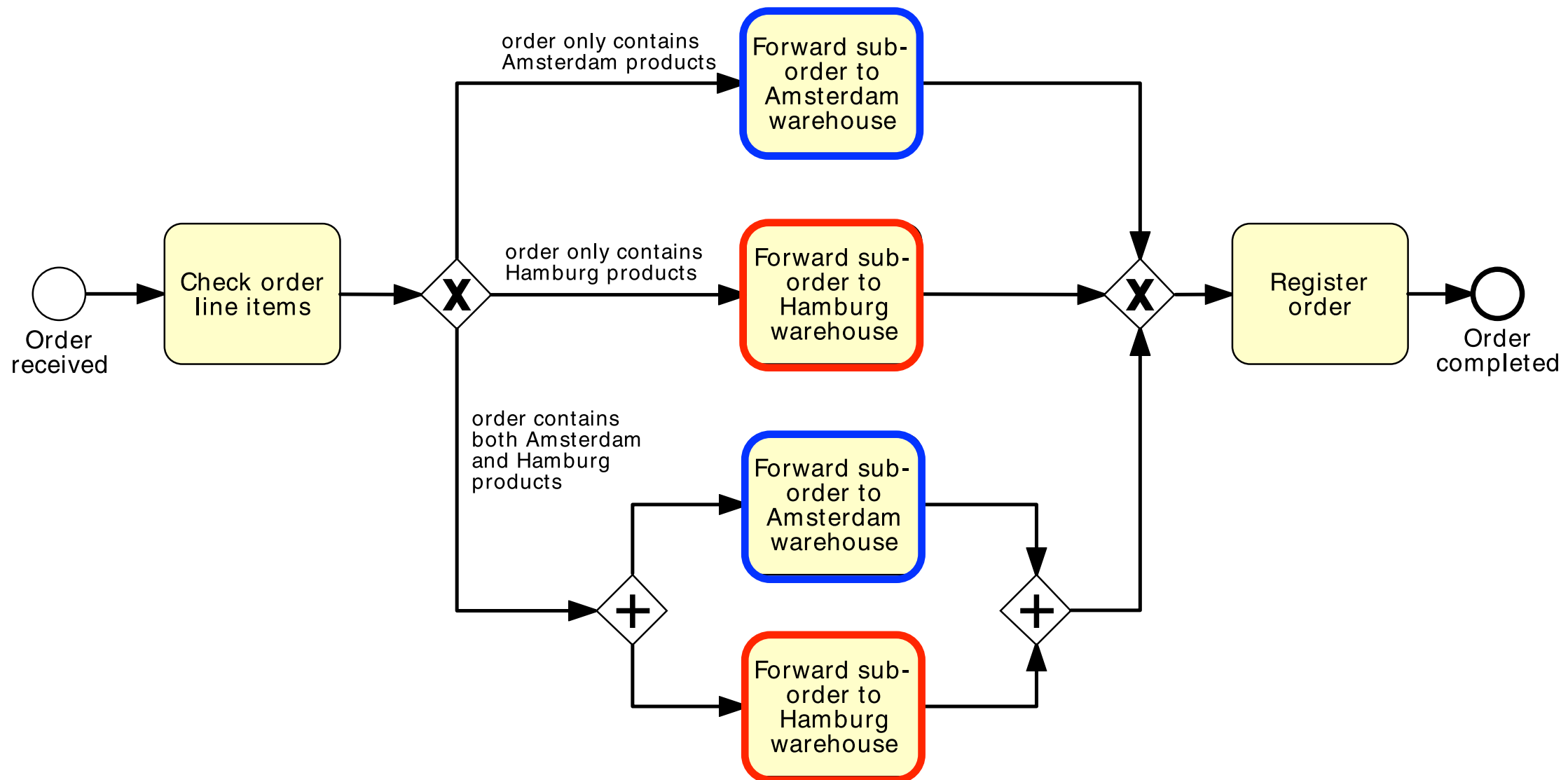
An AND-gateway can be omitted when it follows an activity or event

Similarly, a XOR-gateway before an activity or event can be omitted

Inclusive decisions (none, one, many)

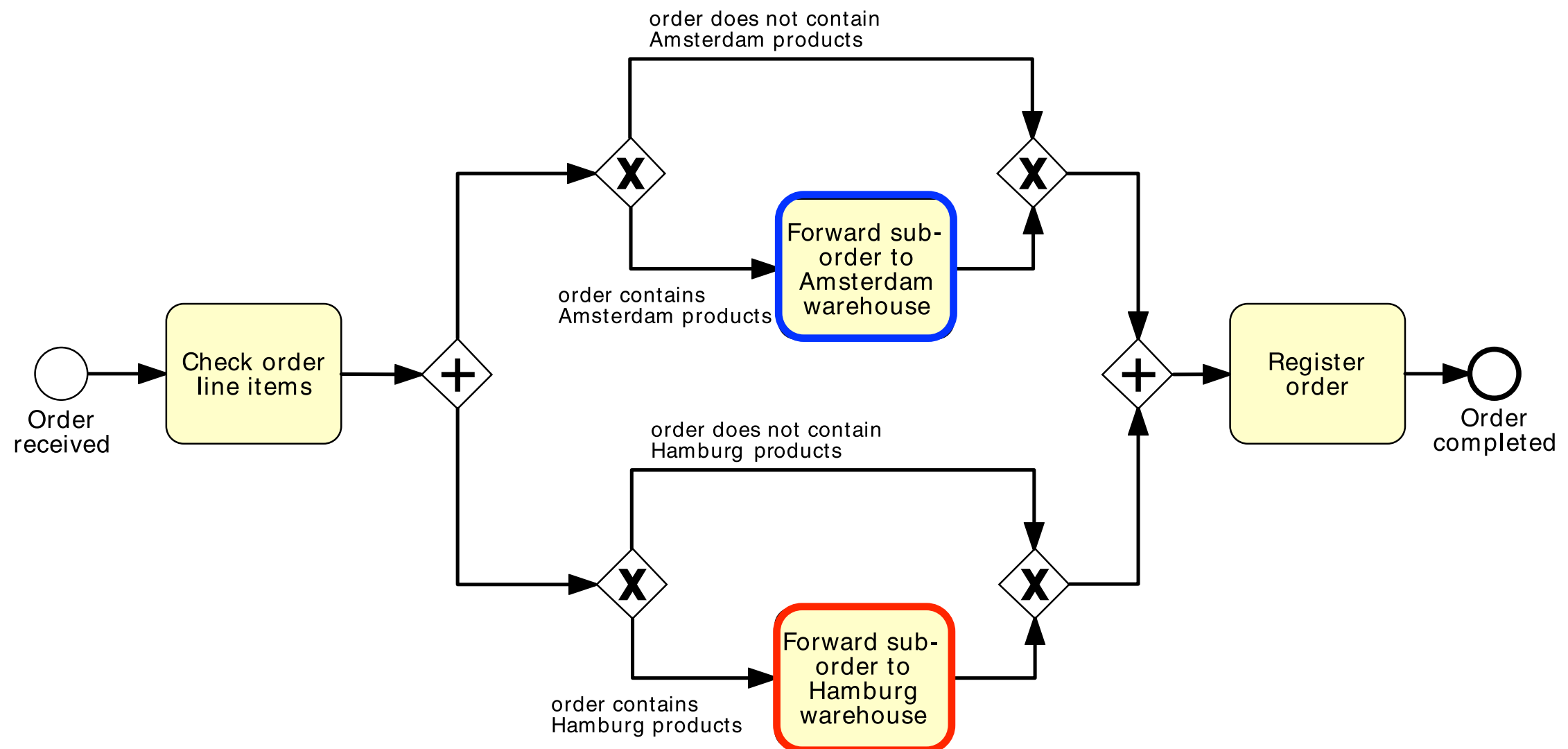


Inclusive decisions: order distribution



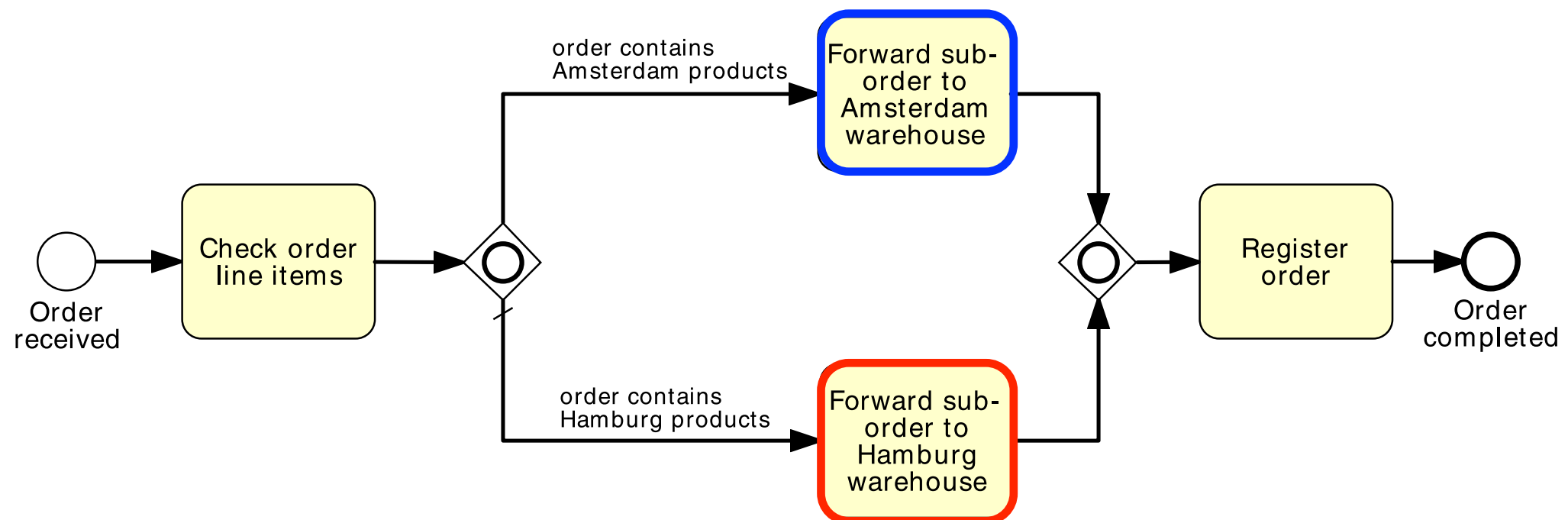
Only XOR / AND gateways, but the diagram is convoluted!
What if we had three or more warehouses? (does not scale)

Inclusive decisions: order distribution



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

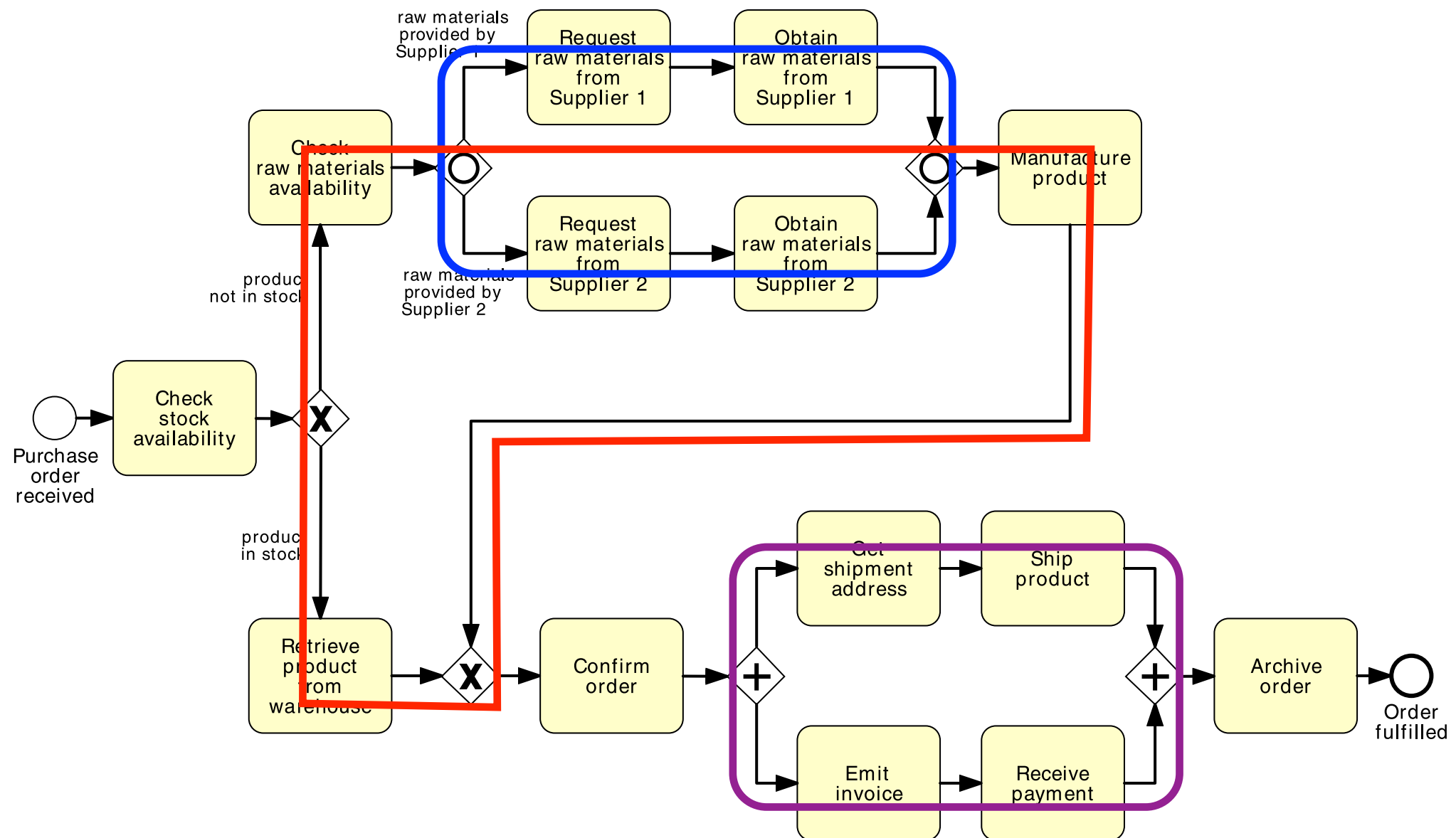
Inclusive decisions: order distribution



OR gateways, the diagram can ``scale'',
but remember all the issues with unmatched OR-joins: they are still valid!

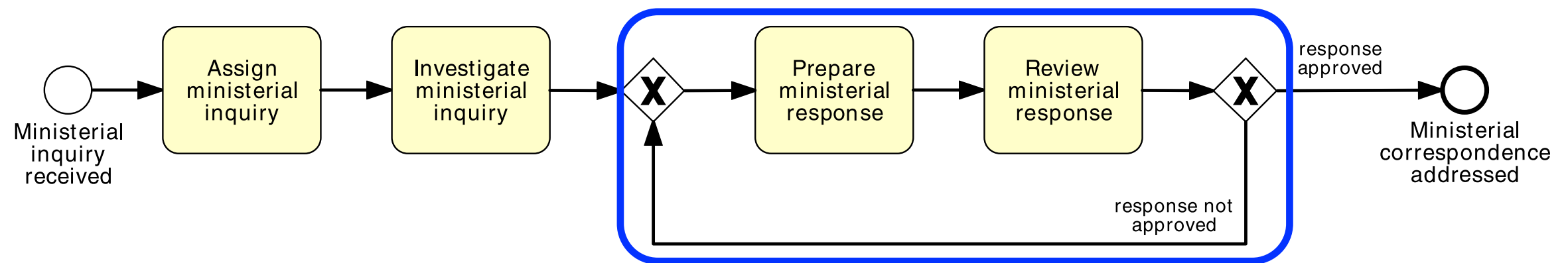
Use OR-gateways only when strictly necessary

XOR + AND + OR: order fulfillment



Better if gateways are balanced

Rework and repetition: ministerial correspondence

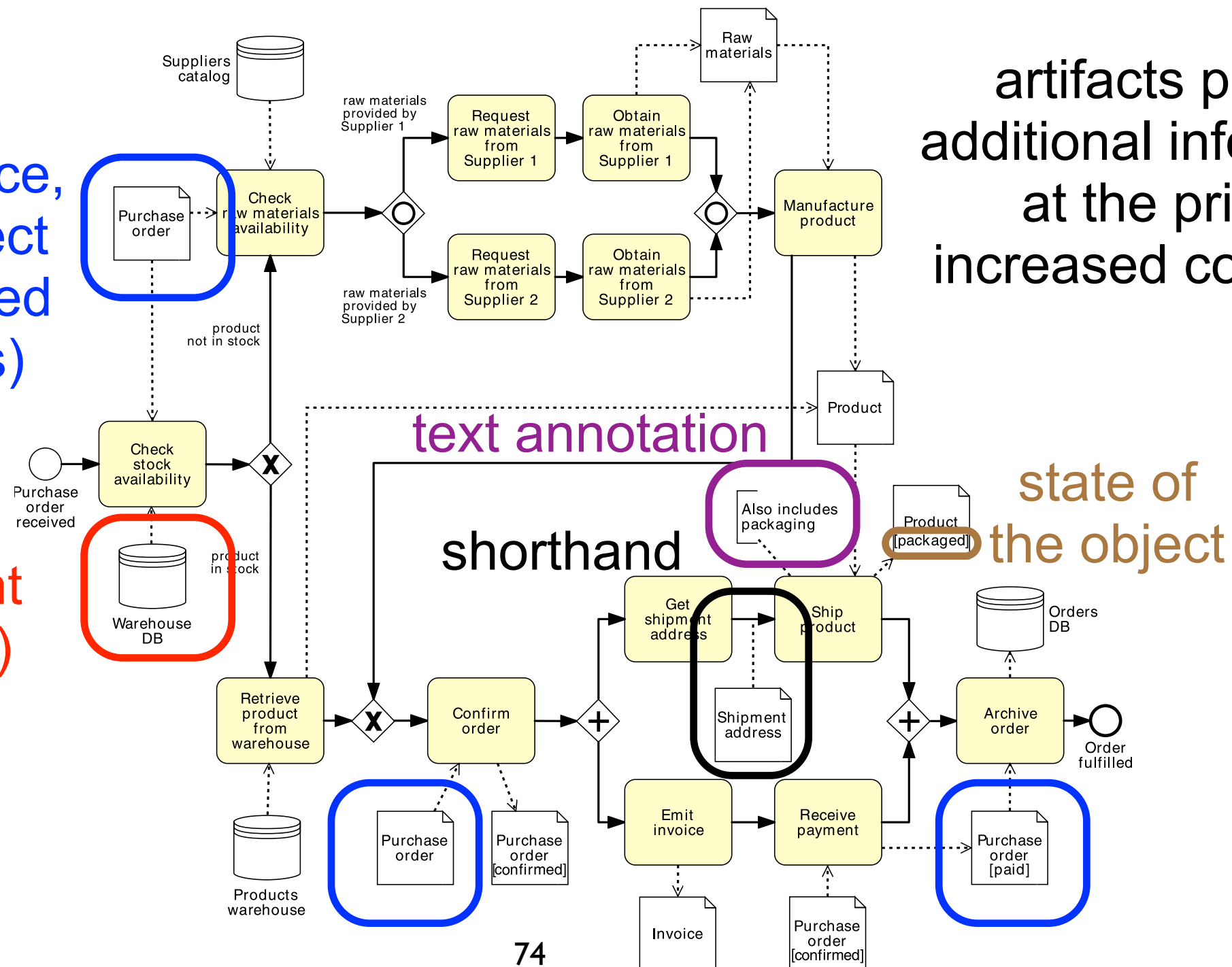


A repetition block starts with a XOR-join
and ends with a decision gateway (XOR-split)

Information artifacts: order fulfillment

data object
(for convenience,
the same object
can be repeated
several times)

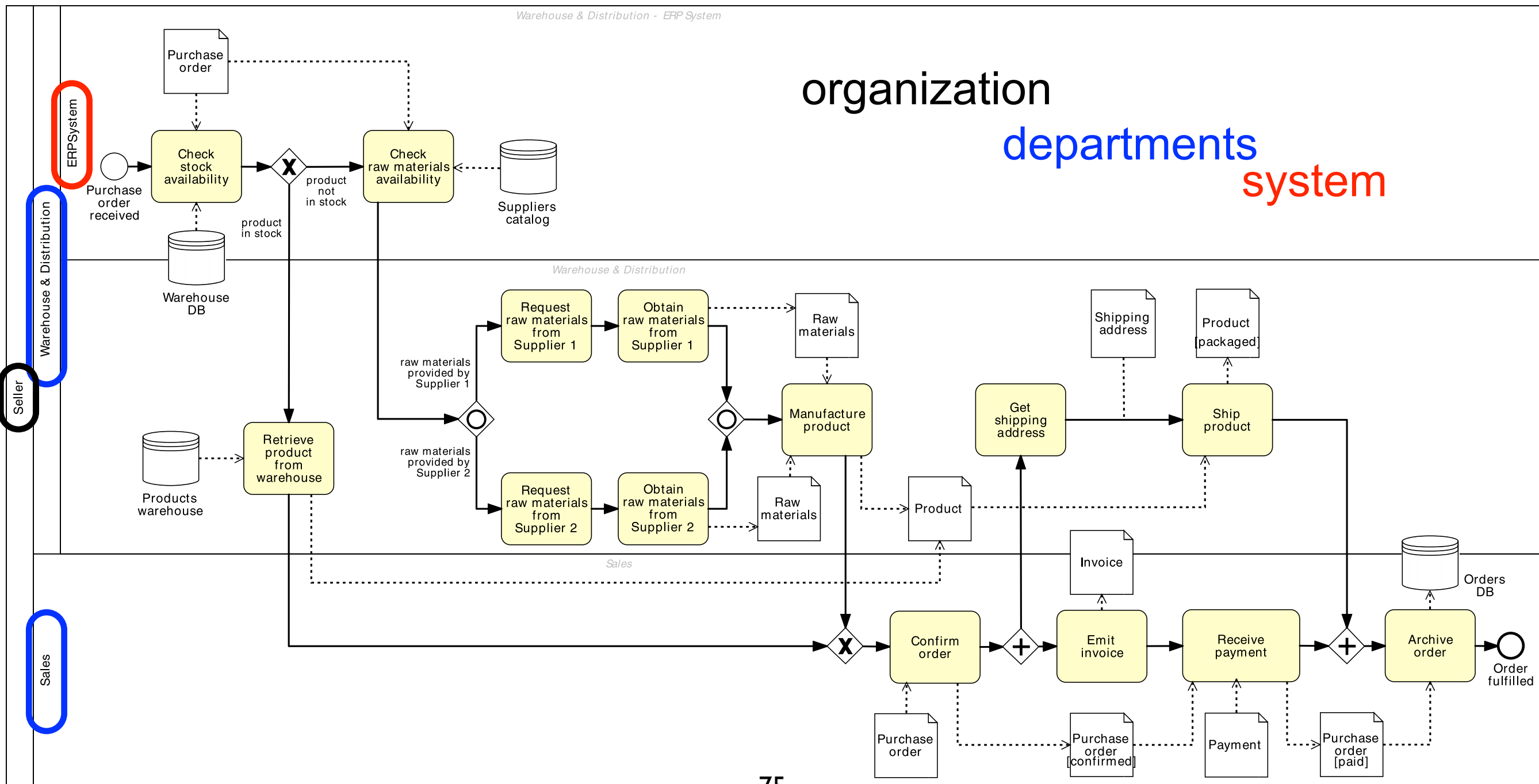
data store
(for persistent
data objects)



artifacts provide
additional information,
at the price of
increased complexity

Resources as lanes:

order fulfillment



Placing items

events: must be placed in the proper lane

activities: must be placed in the proper lane

data-objects: placement is irrelevant

gateways:

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

AND-split, joins: placement is irrelevant

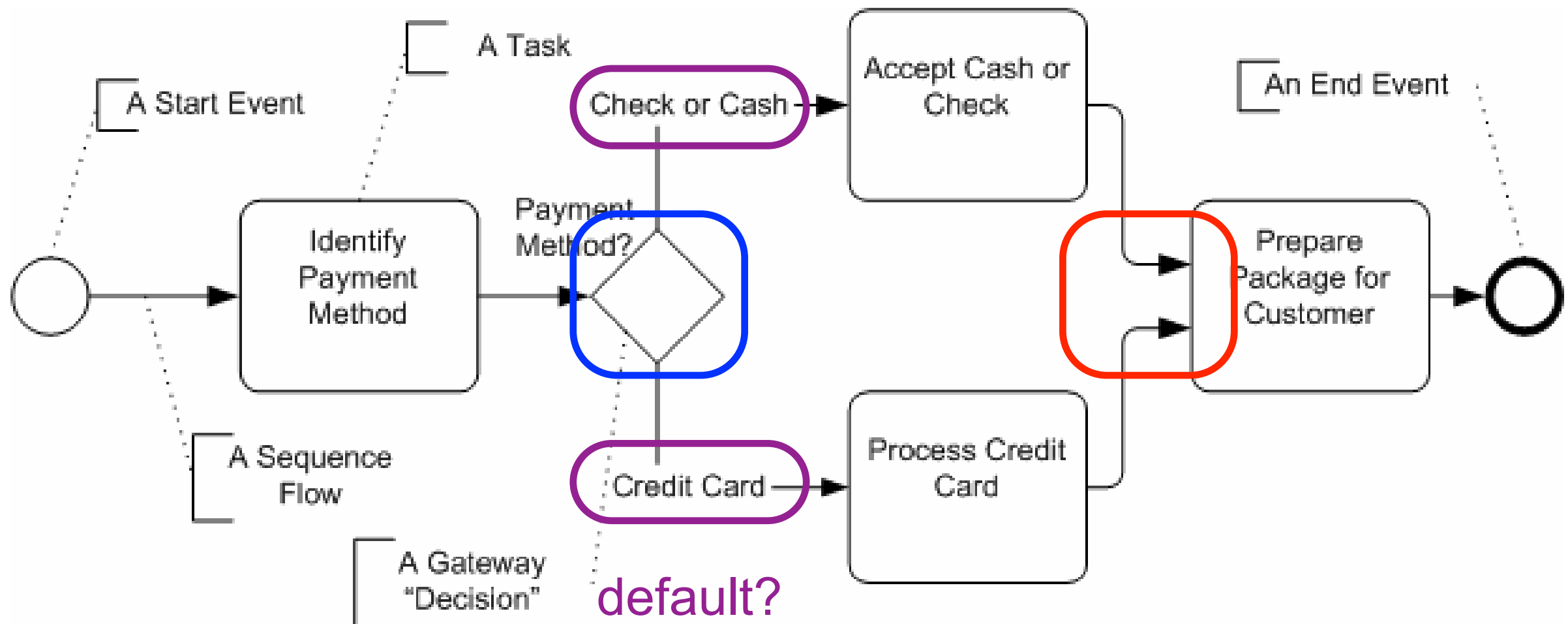
Some remarks

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

Sequence flow may cross the boundaries of Lanes within the same Pool

Message flow may not be used between Flow objects in Lanes of the same Pool

Question time

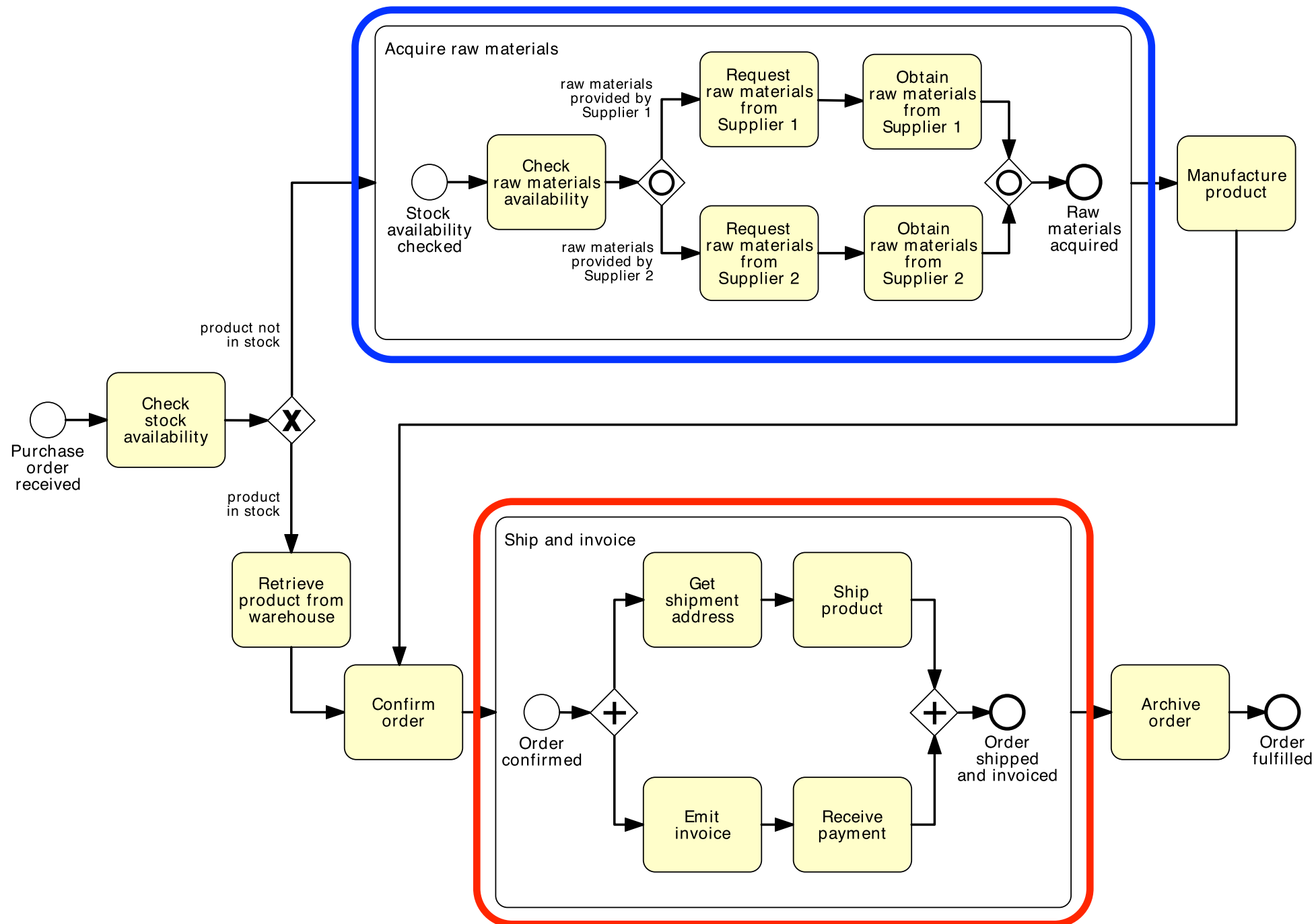


which symbol?

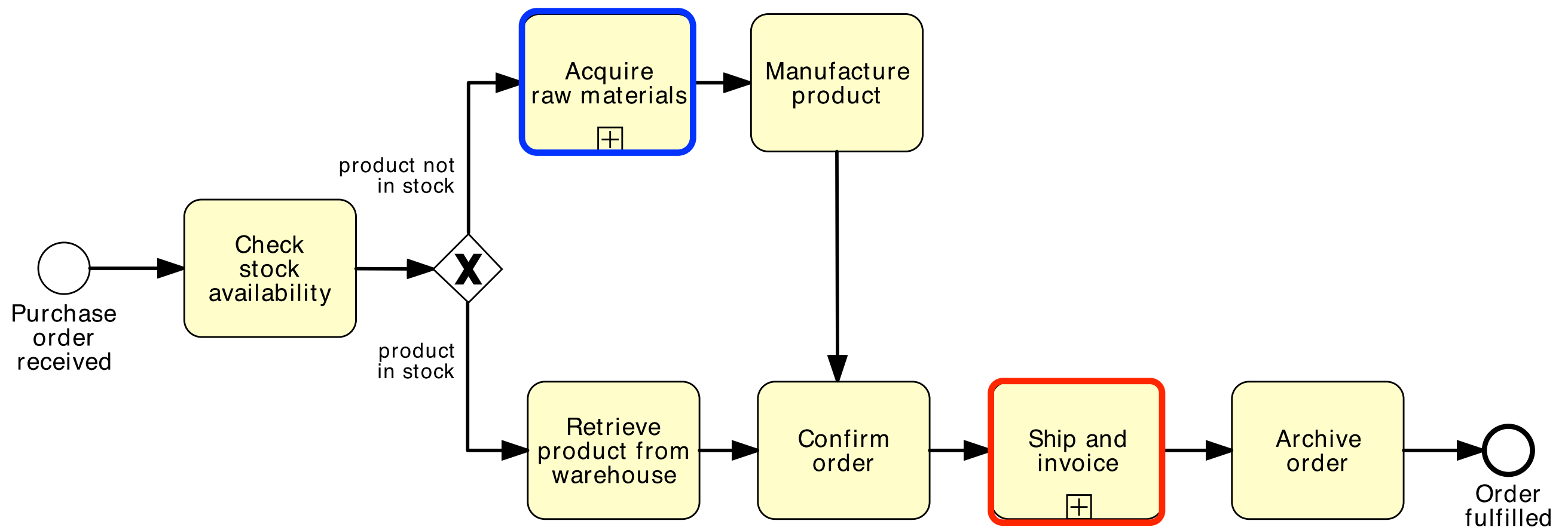
which implicit gateway?

Identify sub-processes:

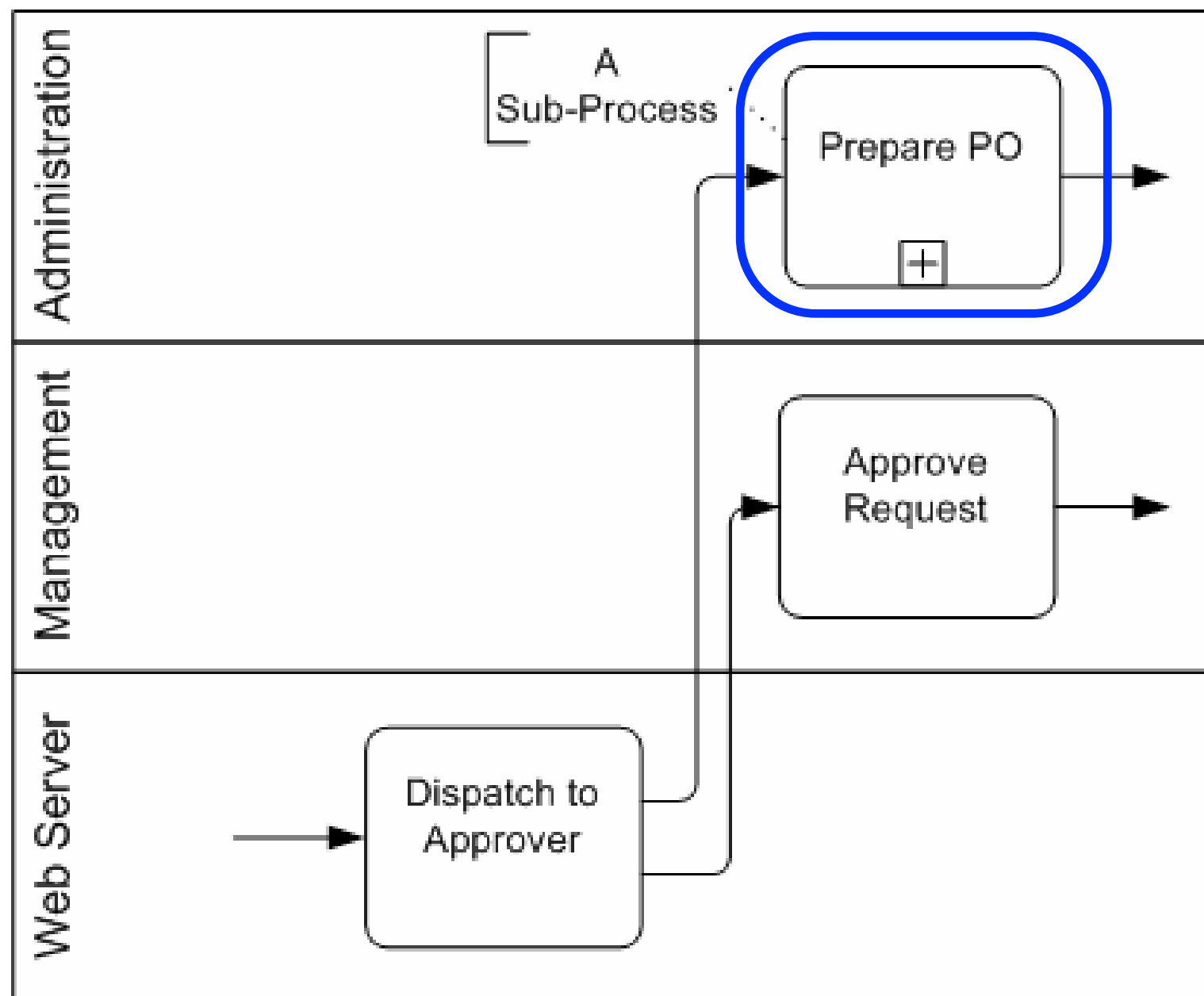
order fulfillment



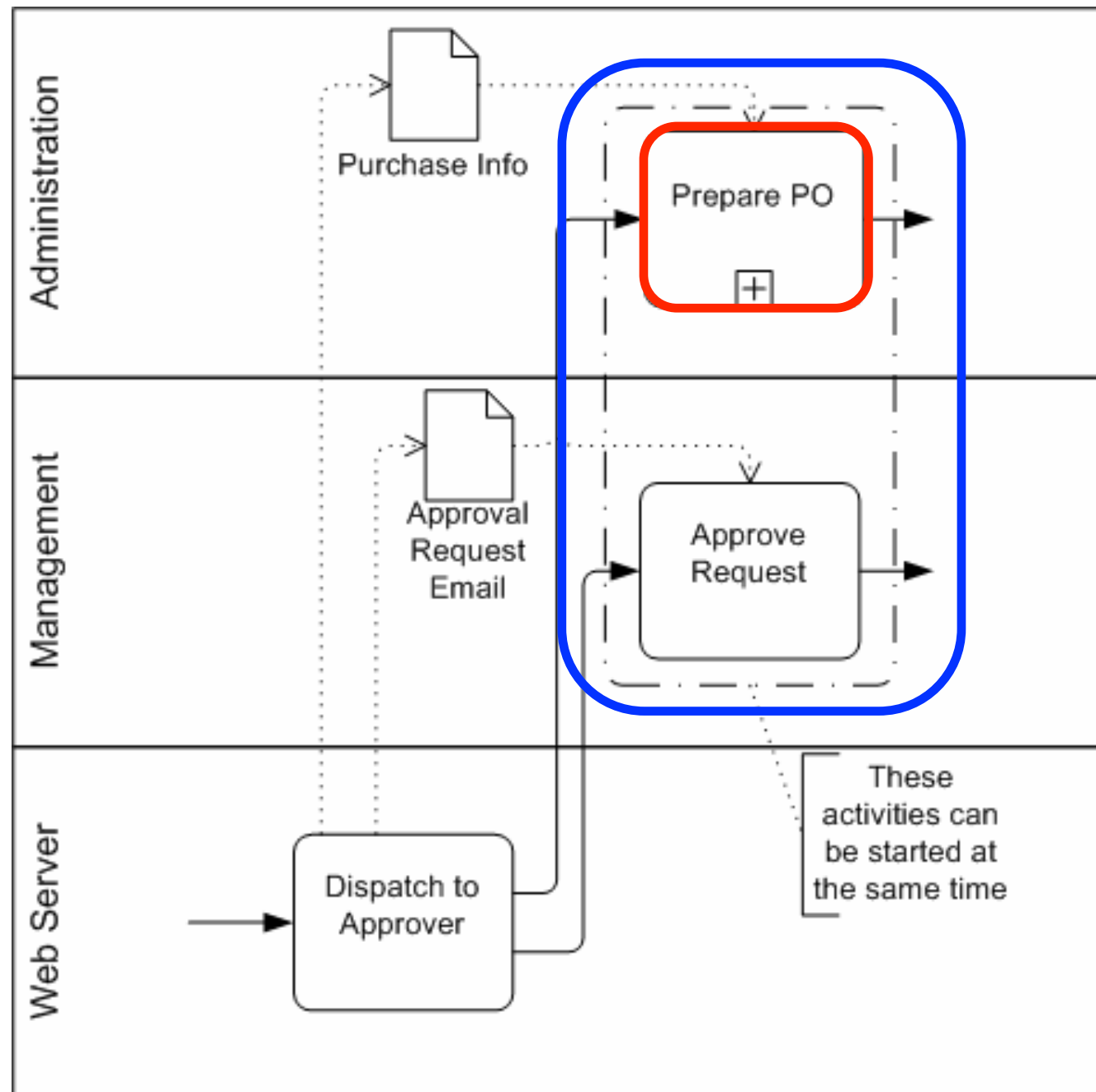
Hiding sub-processes: order fulfillment



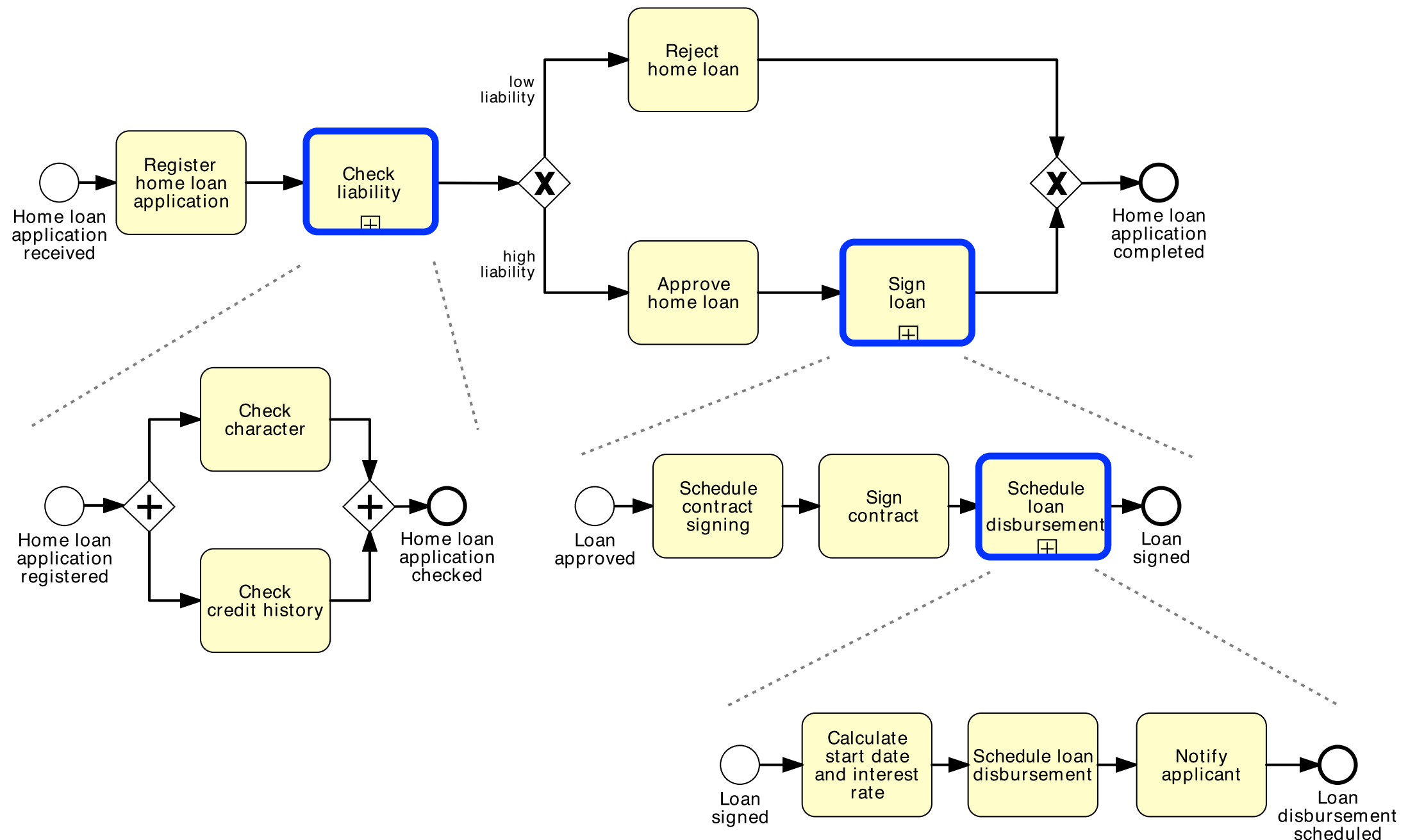
Three lanes and a sub-process



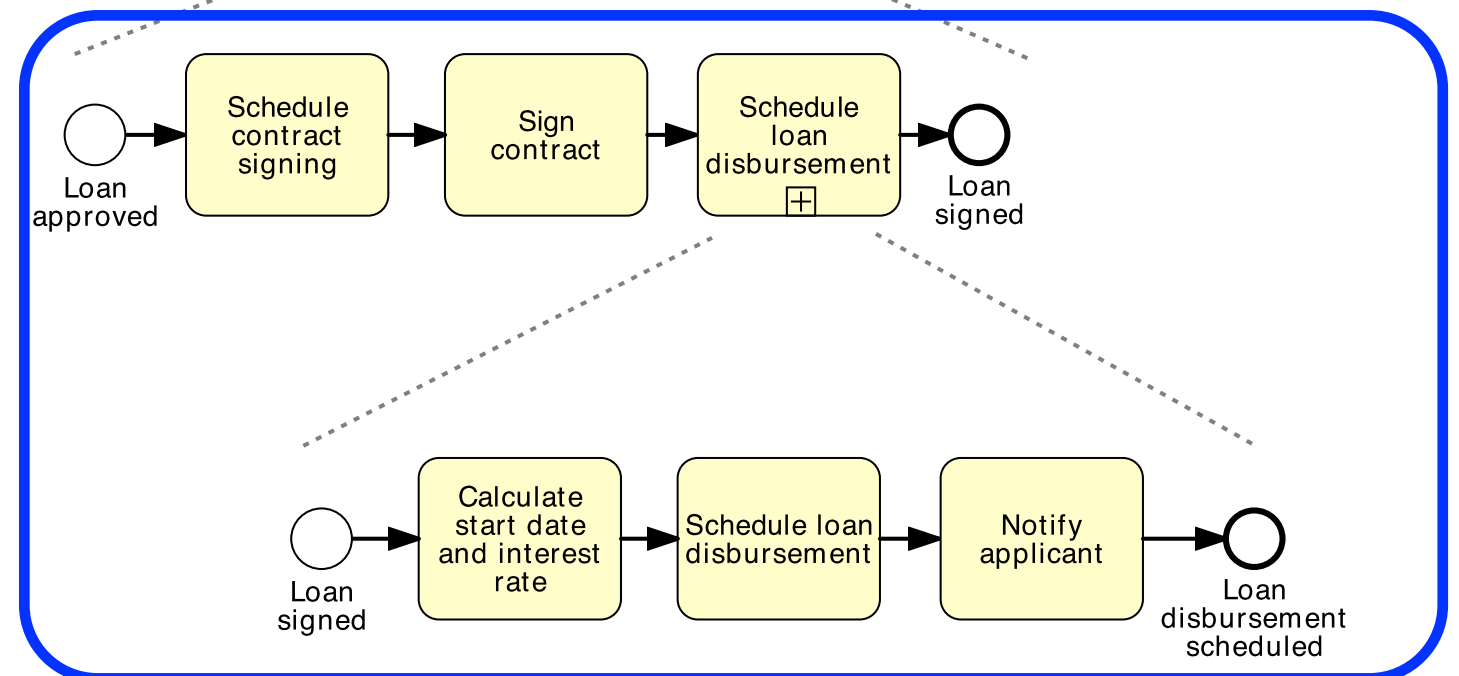
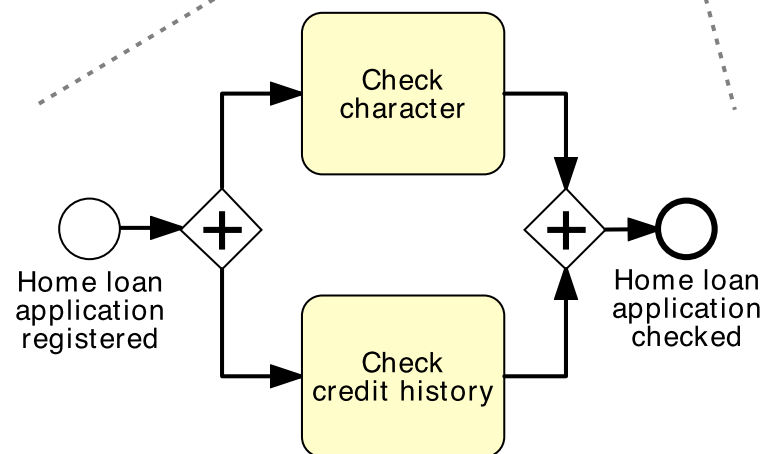
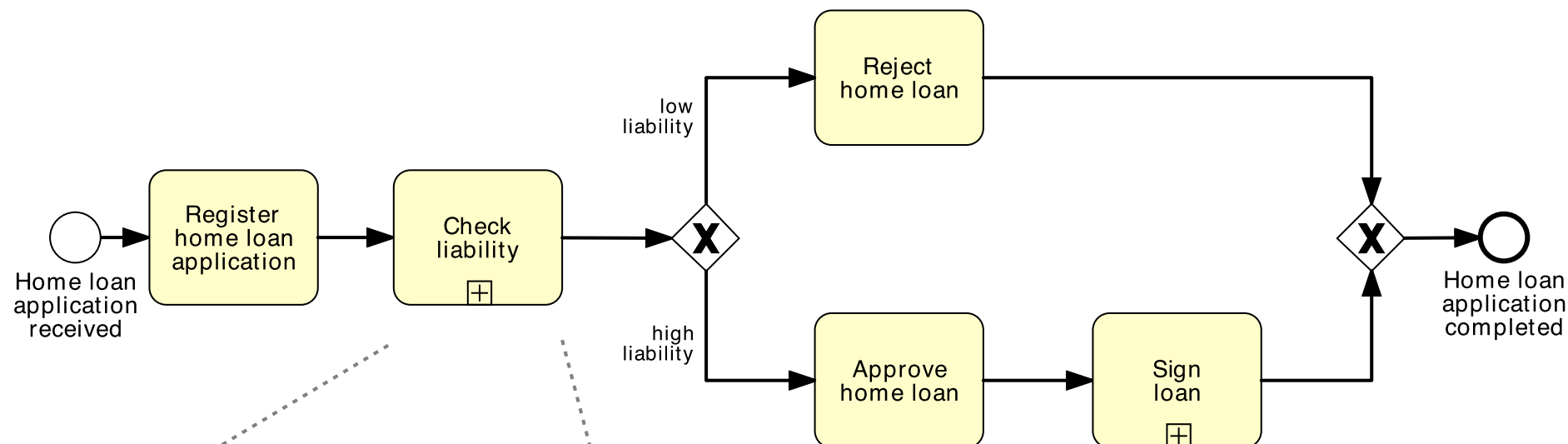
Three lanes, a sub-process and a group



Nesting sub-processes: home loans

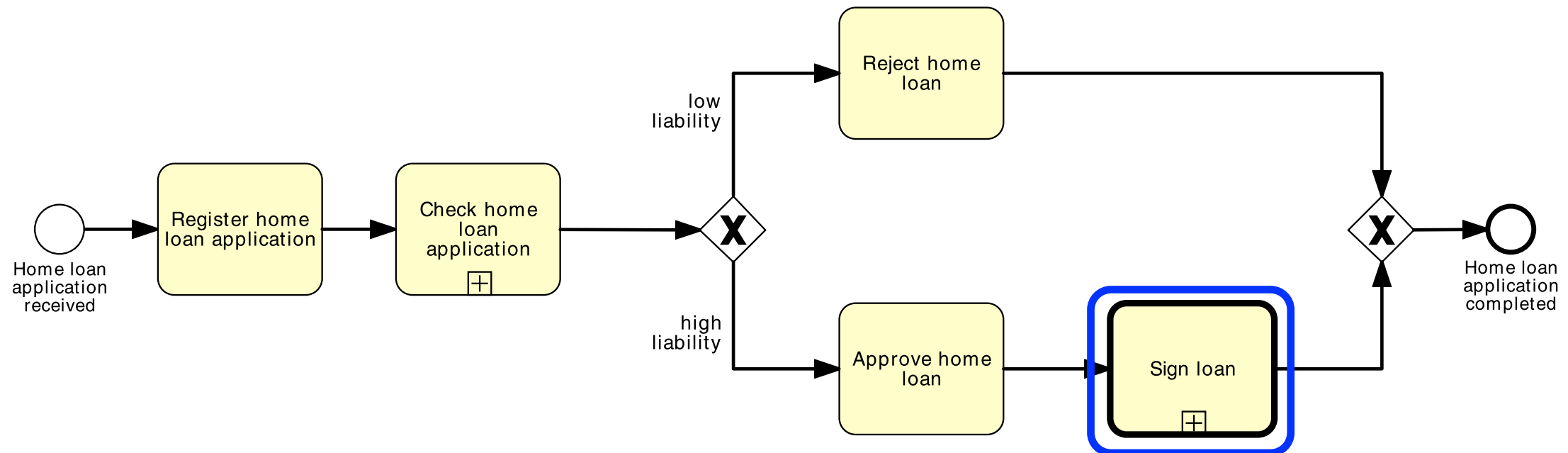


Global sub-processes: home / student loans

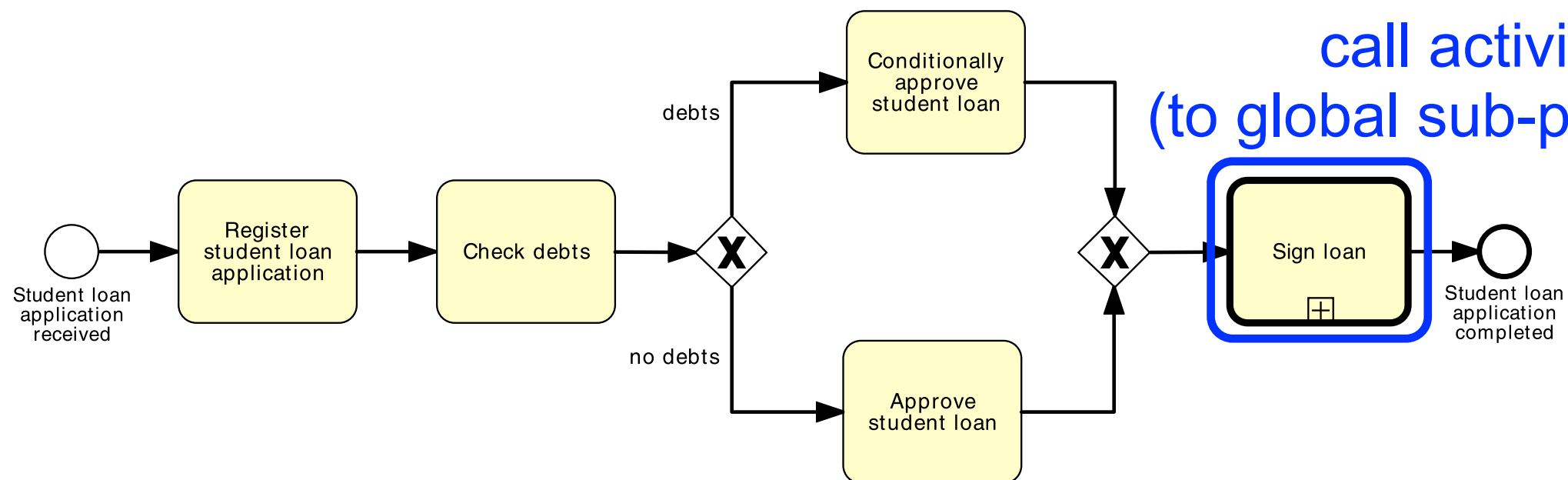


suppose the "Sign loan" process is defined as a separate model: it can be reused

Call activities: home / student loans



thick borders denote
call activities
(to global sub-processes)



Global processes: advantages

Readability: processes tend to be smaller

Reusability: define once, use many time

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

Exercises

Model the following fragments of business processes
for assessing loan applications:

Exercise: loan application 1

Once a loan application has been **approved** by the loan provider, an acceptance pack is **prepared** and **sent** to the customer.

The acceptance pack includes a repayment schedule which the customer needs to agree upon by **sending the signed documents** back to the loan provider.

The latter then verifies the repayment agreement:

if the applicant disagreed with the repayment schedule, the loan provider **cancel**s the application;

if the applicant agreed, the loan provider **approve**s the application.

In either case, the process completes with the loan provider notifying the applicant of the application status.

Exercise: loan application 2

A loan application is **approved** if it passes **two checks**:

- (i) the applicant's loan **risk assessment**, which is done automatically by a system, and
- (ii) the **appraisal** of the property for which the loan has been asked, carried out by a property appraiser.

The risk assessment requires a **credit history check** on the applicant, which is performed by a financial officer.

Once both the loan risk assessment and the property appraisal have been performed, a loan officer can **assess** the applicant's eligibility.

If the applicant is not eligible, the application is **rejected**, **otherwise** the acceptance pack is **prepared and sent** to the applicant.

Exercise: loan application 3

A loan application may be coupled with a home insurance which is offered at discounted prices.

The applicant may express their interest in a home insurance plan at the time of submitting their loan application to the loan provider.

Based on this information, **if** the loan application is **approved**, the loan provider may **either only** send an **acceptance pack** to the applicant, **or also** send a **home insurance quote**.

The process then continues with the **verification** of the repayment agreement.

Exercise: loan application 4

Once a loan application is **received** by the loan provider, and before proceeding with its assessment, the application itself needs to be **checked** for completeness.

If the application is incomplete, it is **returned** to the applicant, so that they can **fill out** the missing information and **send it back** to the loan provider.

This process is **repeated** until the application is complete.

Exercise: loan application 5

Put together the four fragments of the loan assessment process that you created in previous Exercises.

Then extend the resulting model by adding all the required artifacts.

Moreover, attach annotations to specify the business rules behind:

- (i) checking an application completeness,
- (ii) assessing an application eligibility, and
- (iii) verifying a repayment agreement.

Exercise: loan application 6

Extend the business process for assessing loan applications that you created in previous exercises by considering the following resource aspects.

The process for assessing loan applications is executed by four roles within the **loan provider**:

a **financial officer** takes care of checking the applicant's credit history;

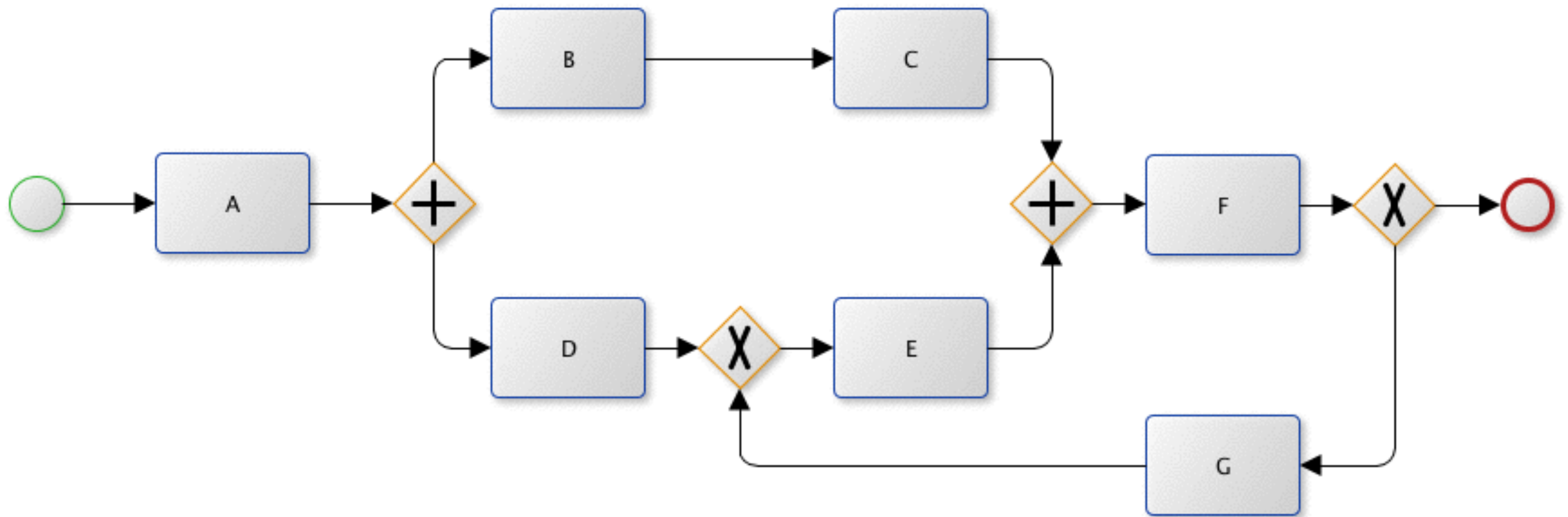
a **property appraiser** is responsible for appraising the property;

an **insurance sales representative** sends the home insurance quote to the applicant if this is required.

All other activities are performed by the **loan officer** who is the main point of contact with the applicant.

Exercises: refactoring

Can the process model below execute correctly?
If not, how can it be fixed without affecting the cycle, i.e.
such that F, G, and E all remain in a cycle?



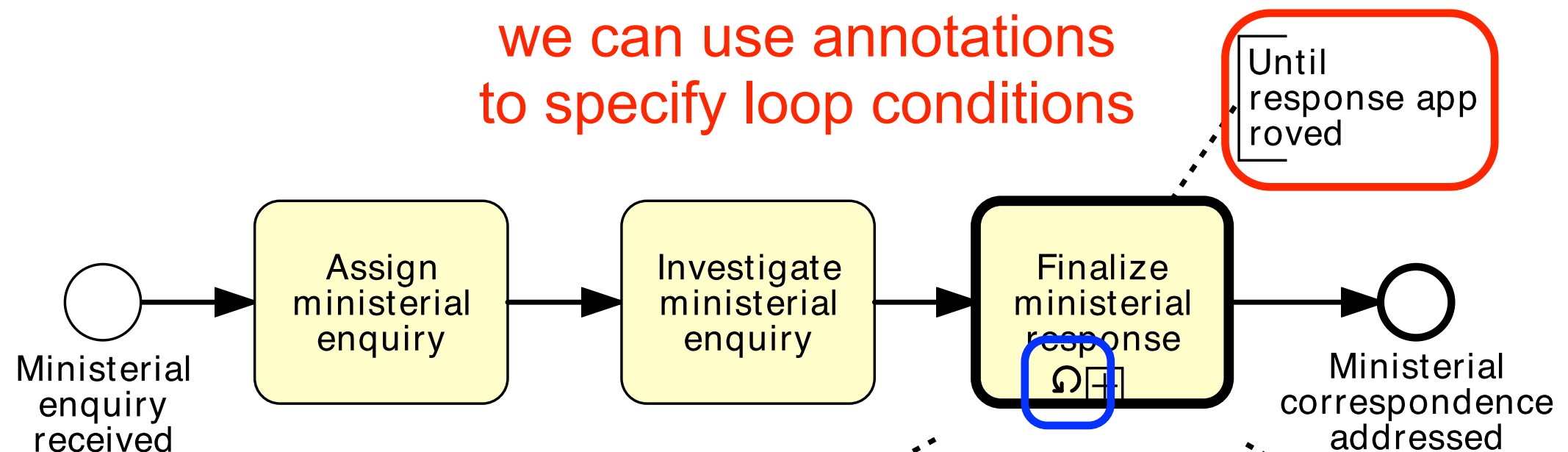
Semantics annotations

The graphical syntax is not expressive enough
to model exactly all interesting situations

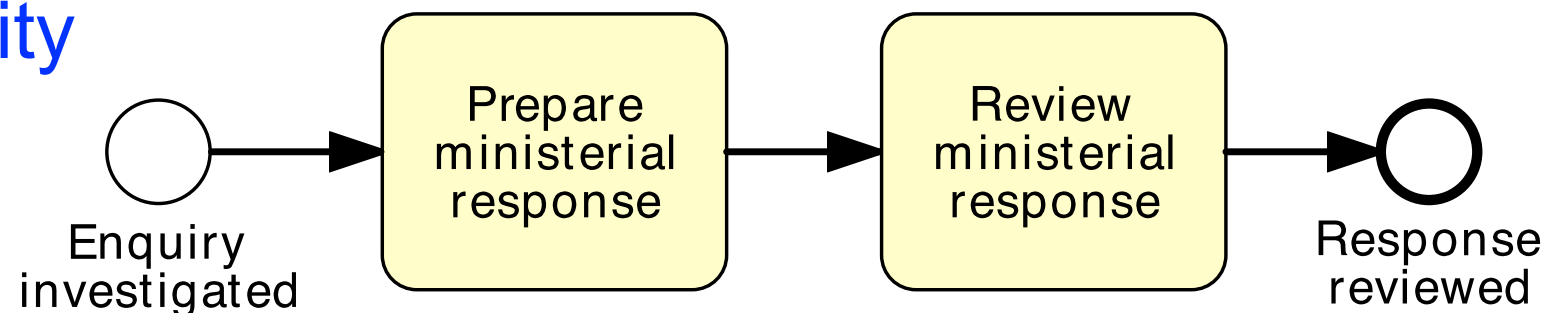
In many cases part of the behaviour is moved
to decorations and annotations
(i.e., without considering them the implementation
is not possible, as well as providing formal semantics)

Loop annotation: ministerial correspondence

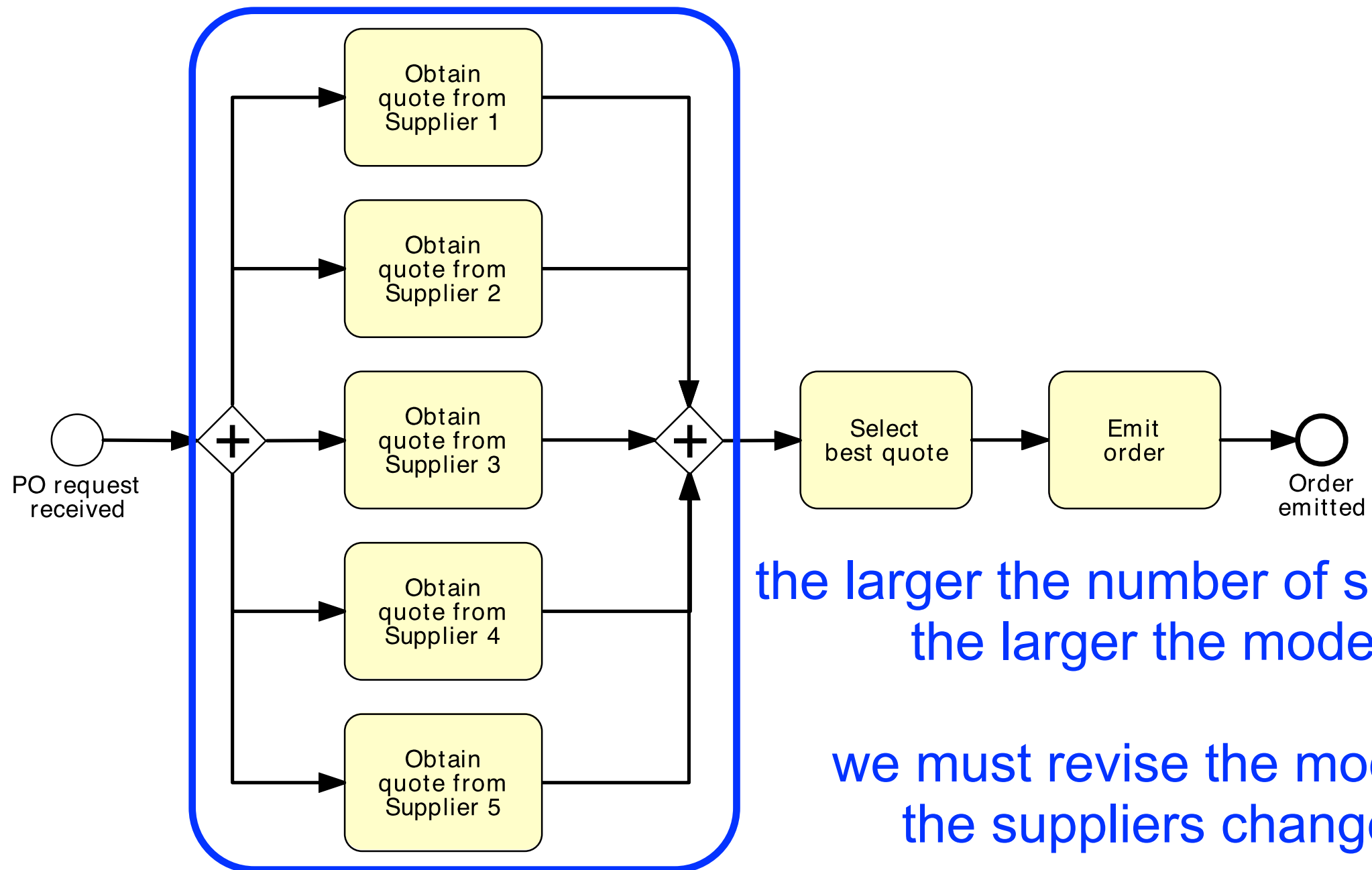
we can use annotations
to specify loop conditions



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



Parallel repetition: procurement process

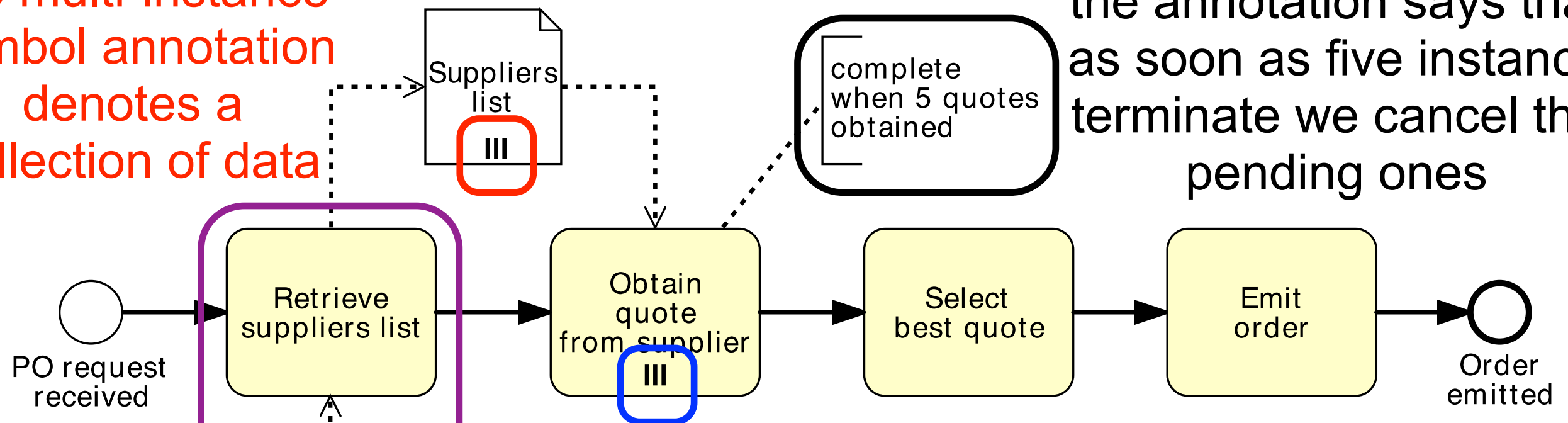


the larger the number of suppliers
the larger the model!

we must revise the model if
the suppliers change!

Multi-instance activities: procurement process

the multi-instance
symbol annotation
denotes a
collection of data

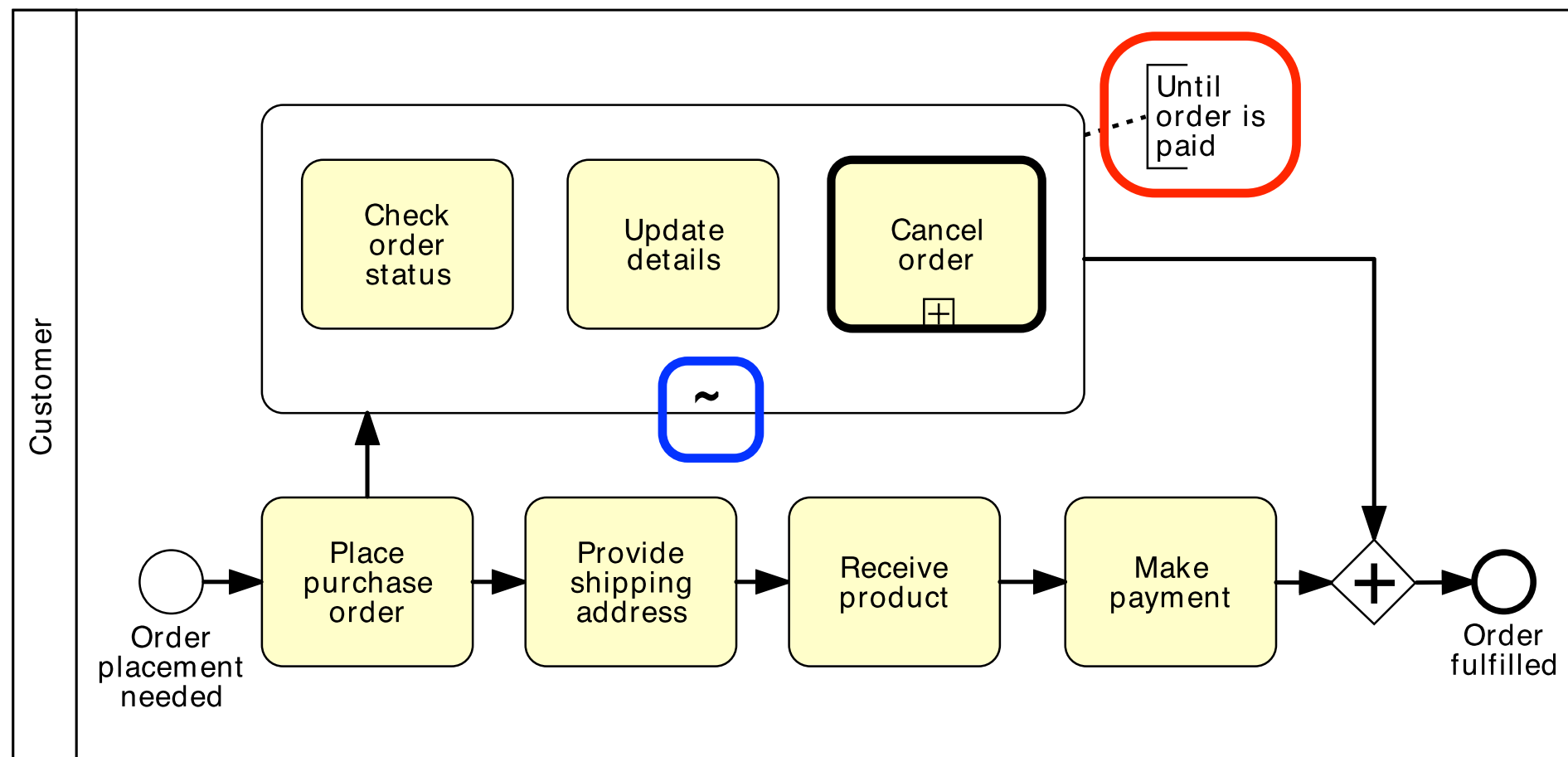


the annotation says that
as soon as five instance
terminate we cancel the
pending ones

the multi-instance symbol annotation
denotes an activity that is executed
multiple times concurrently
(e.g. repeated activity for multiple entries
or data-items)

the list of instances
is determined
dynamically

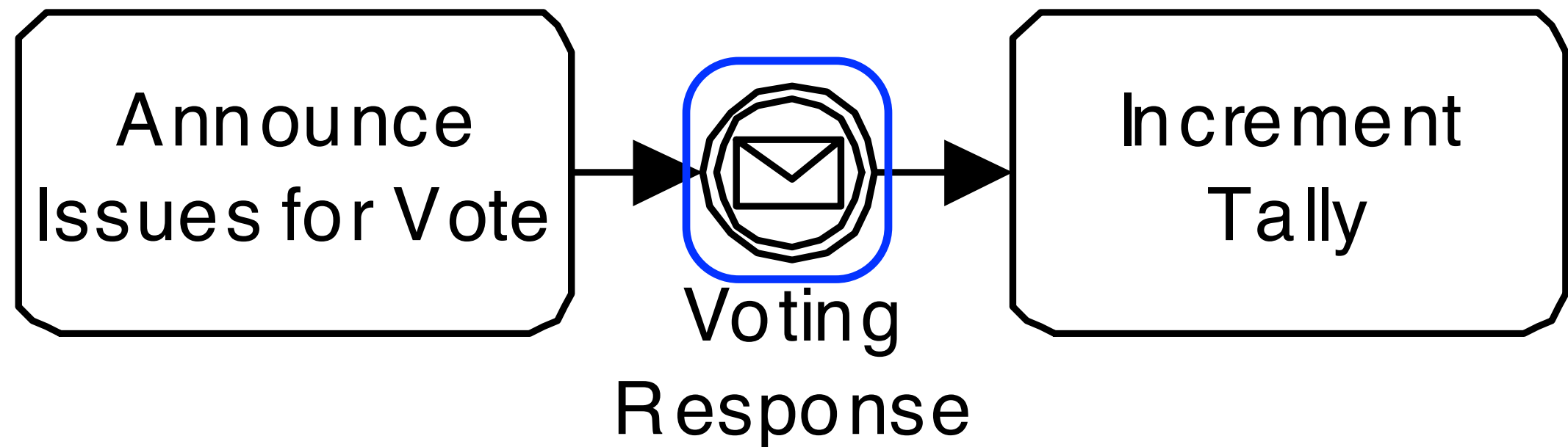
Ad-hoc sub-processes: customer process



we can use
annotations
to specify
loop conditions

the ad-hoc symbol annotation
denotes an uncontrolled repetition of activities:
they may be repeated multiple times with no specific order
or not occur at all, until a condition is met

Process break (event waiting)



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

a black-filled envelope would denote that a message has been sent

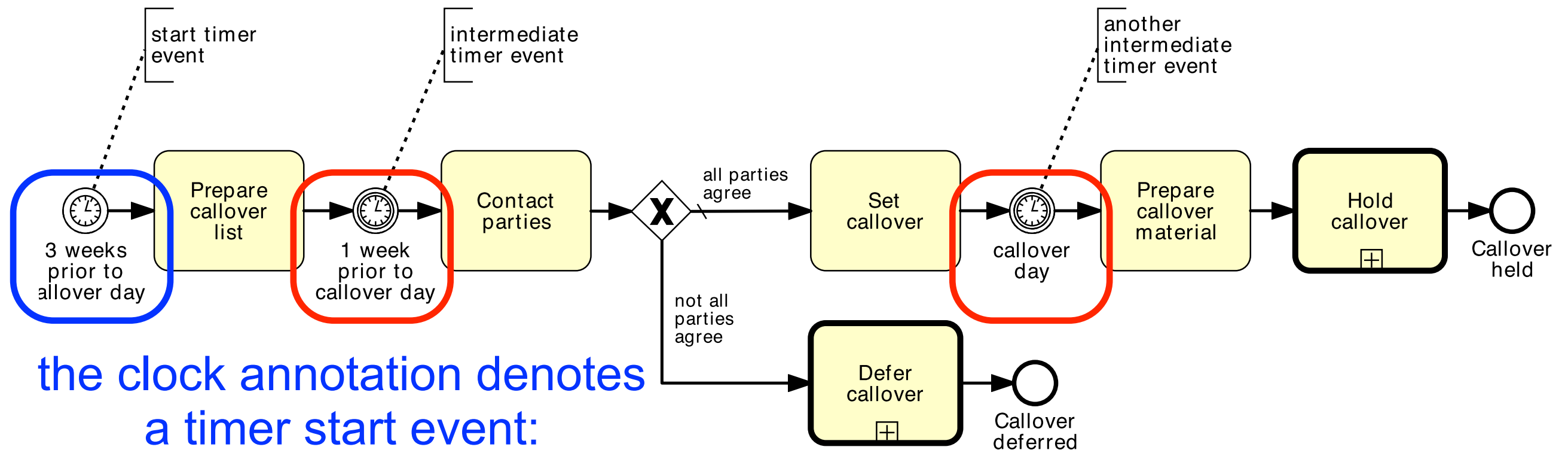
Message annotated events and activities

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:
the process concludes by sending a message

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

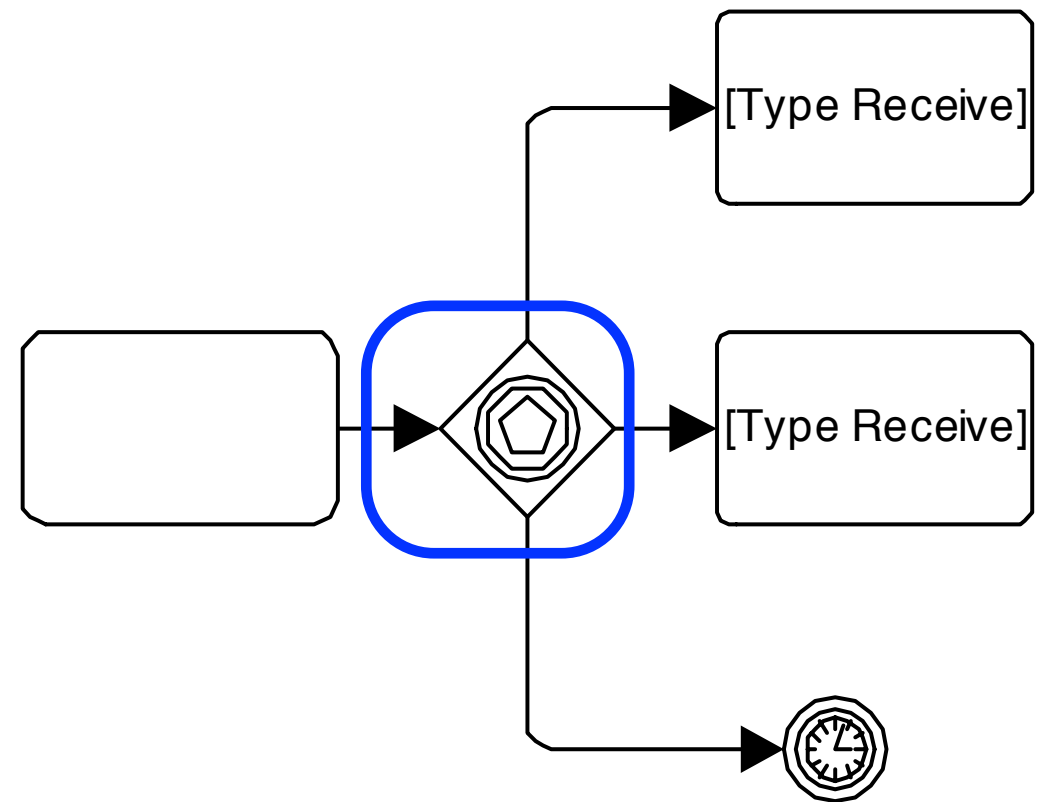
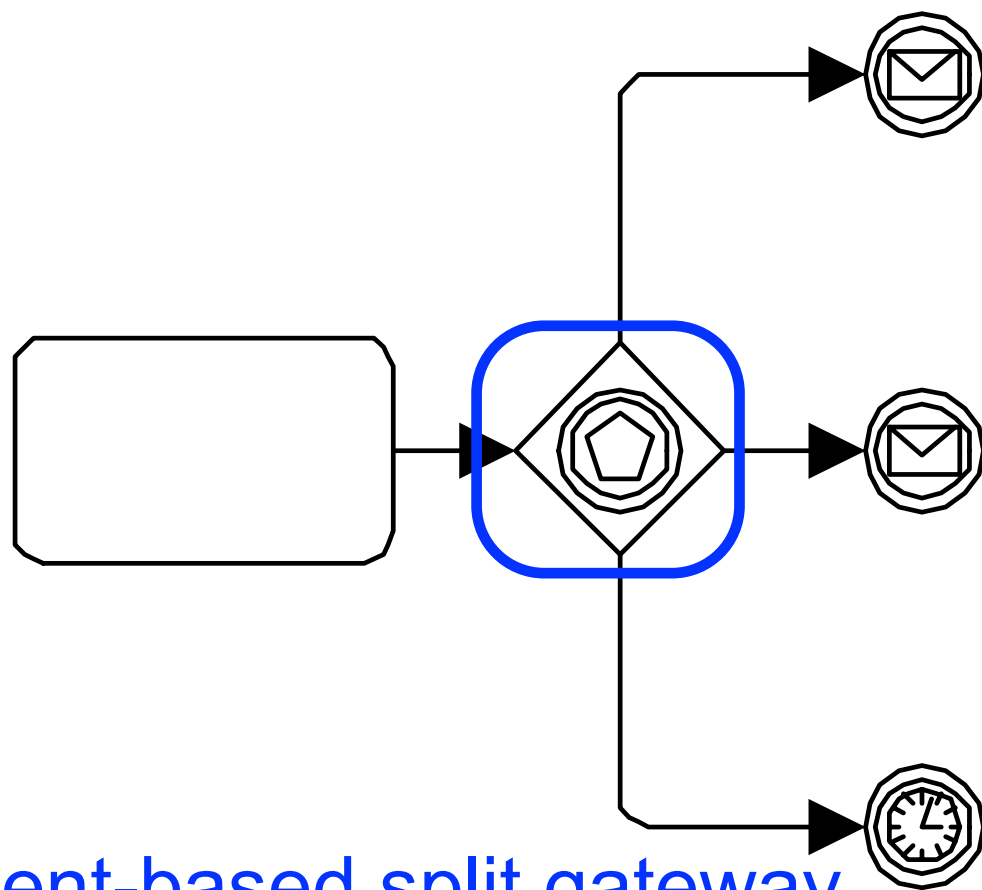
Timer events: small claims tribunal



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

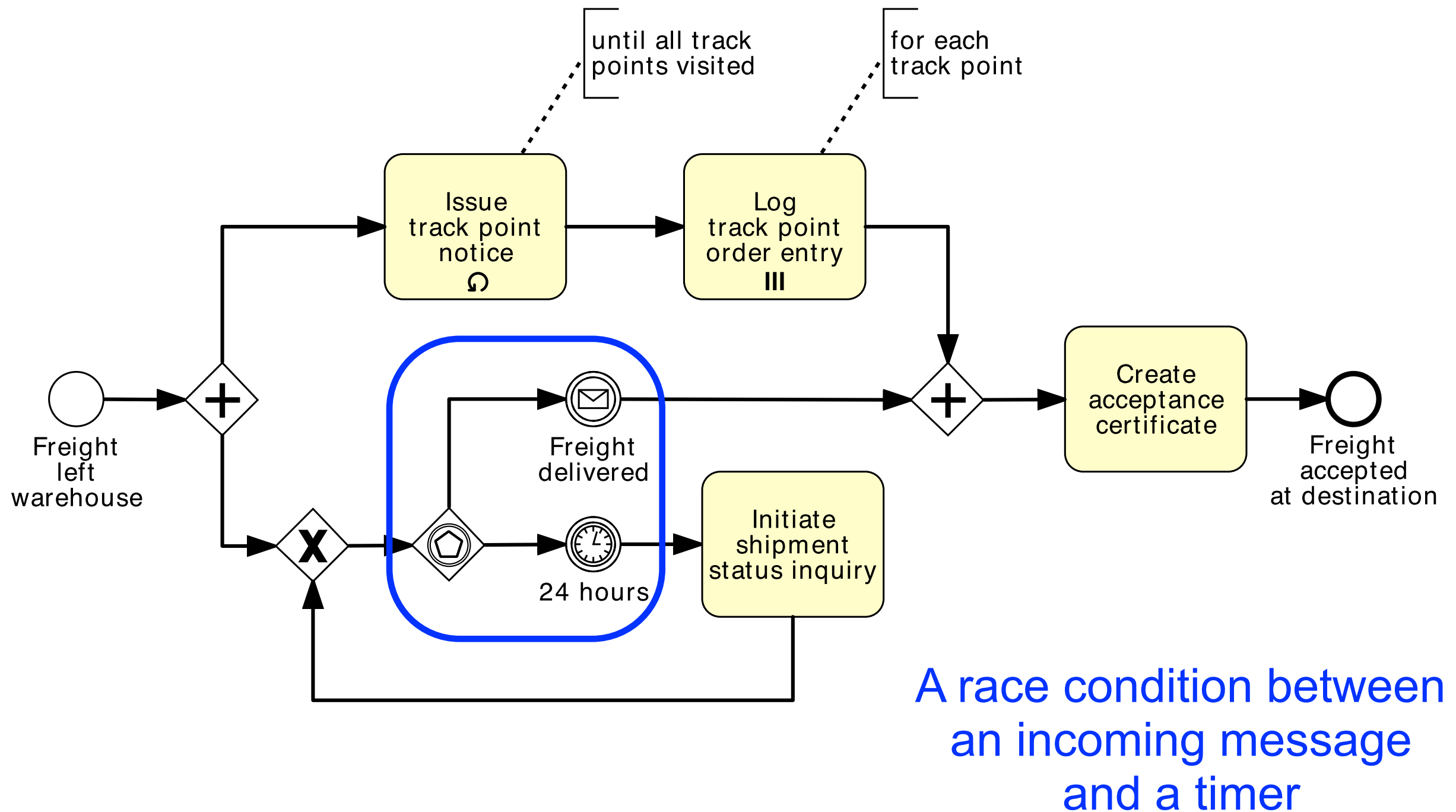
the clock annotation denotes
a timer intermediate event:
the process is blocked until
a time-out expires

Event-based decisions (also deferred choice)



Event-based split gateway
can be used to select
a branch based on some
external event

Deferred choice



Exceptions:

rainy-days vs sunny-days

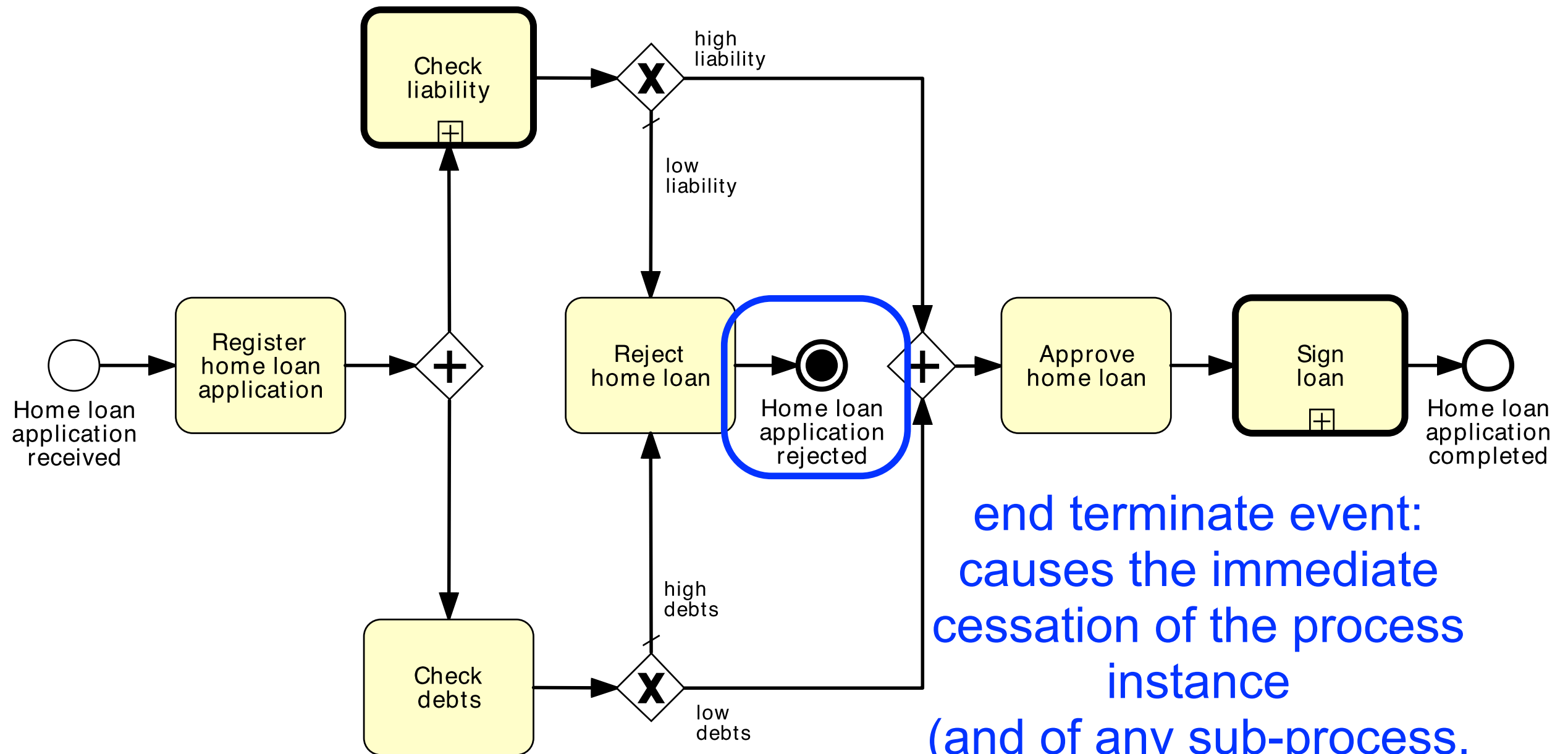
Exceptions are events
that deviate a process from its normal course

They include: business faults (e.g., out of stock),
technology faults (e.g., database crash)

Exceptions provoke the interruption or abortion
of the running process instance

Before adding exceptions it is important to have
the sunny-day scenario well understood

Process abortion: home loan



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

Handling exceptions: rainy-days vs sunny-days

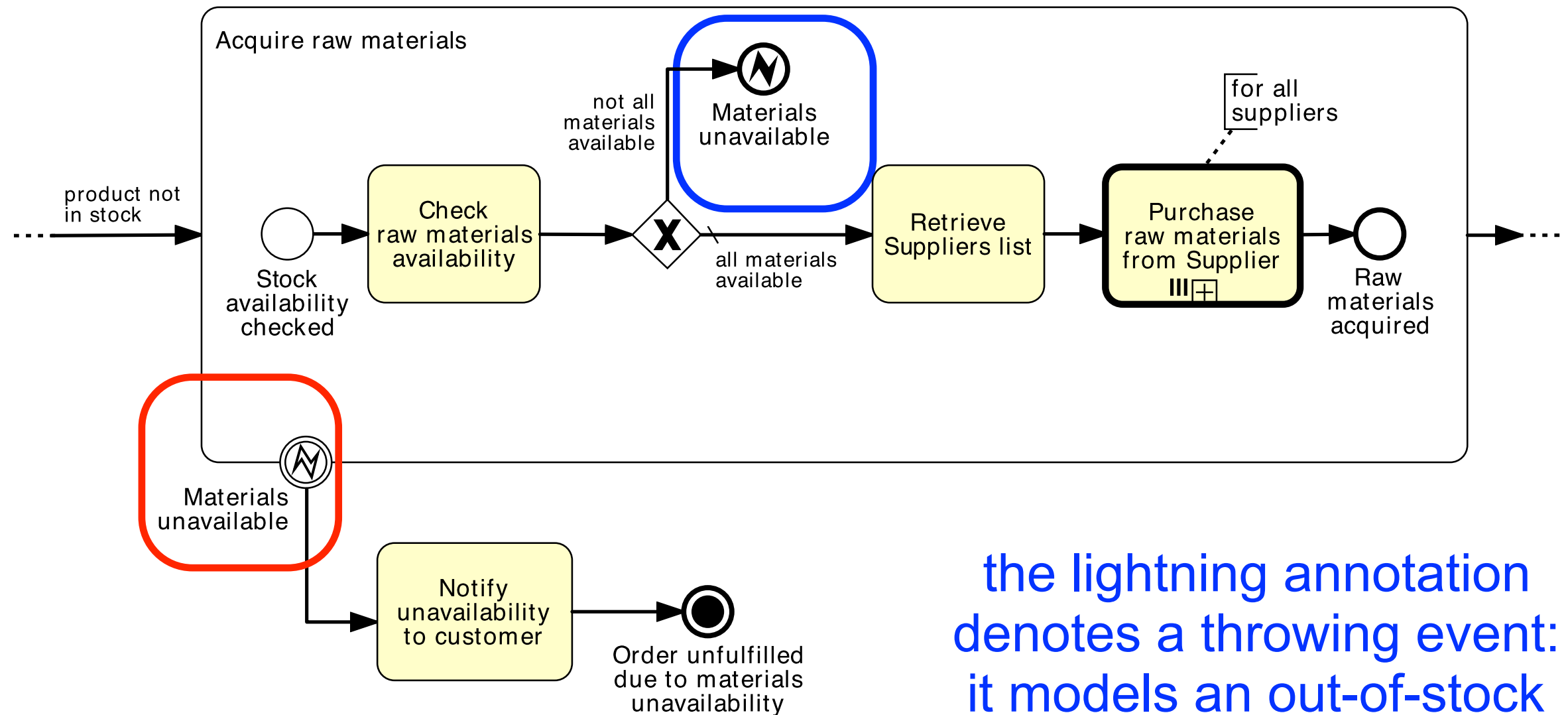
We can handle exceptions of sub-processes by interrupting the activity that caused the exception and moving the control flow to another process

The recovery procedure can try to bring the process back to a consistent state

Error end events are used to interrupt the execution

Boundary events trigger the recovery procedure
(called exception flow)

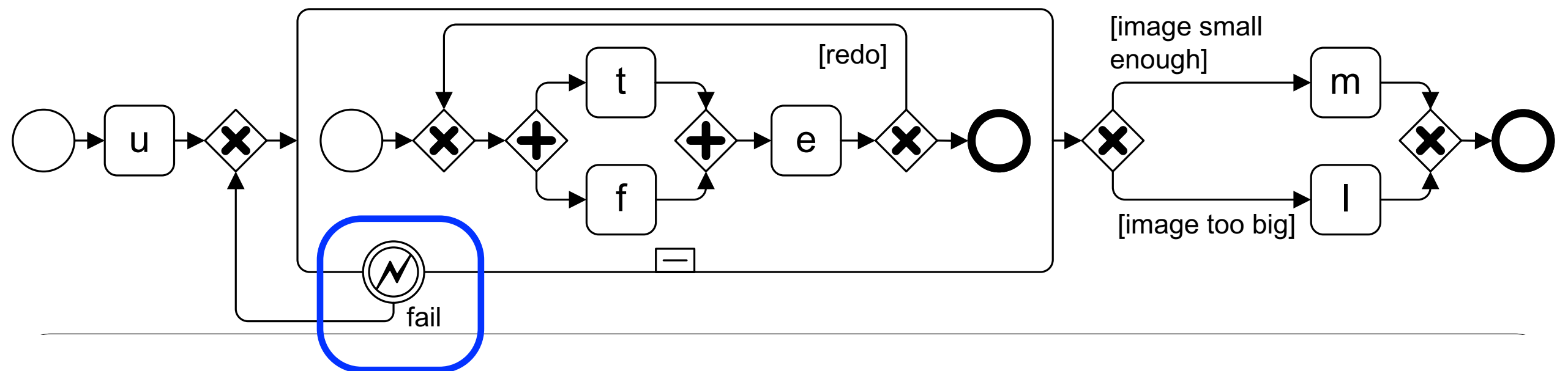
Throwing and catching: order fulfillment



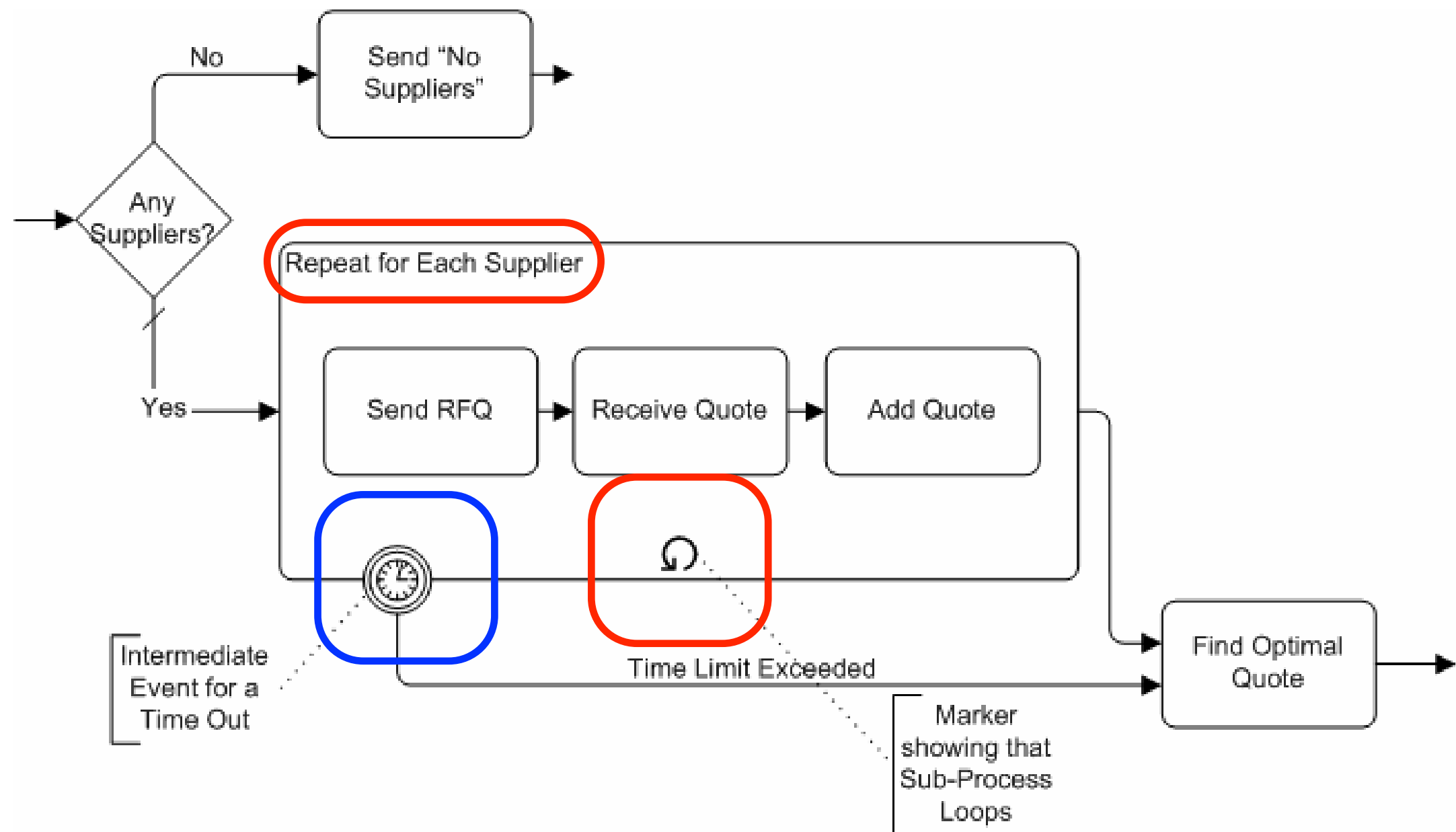
the lightning annotation
denotes an error-catching event

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

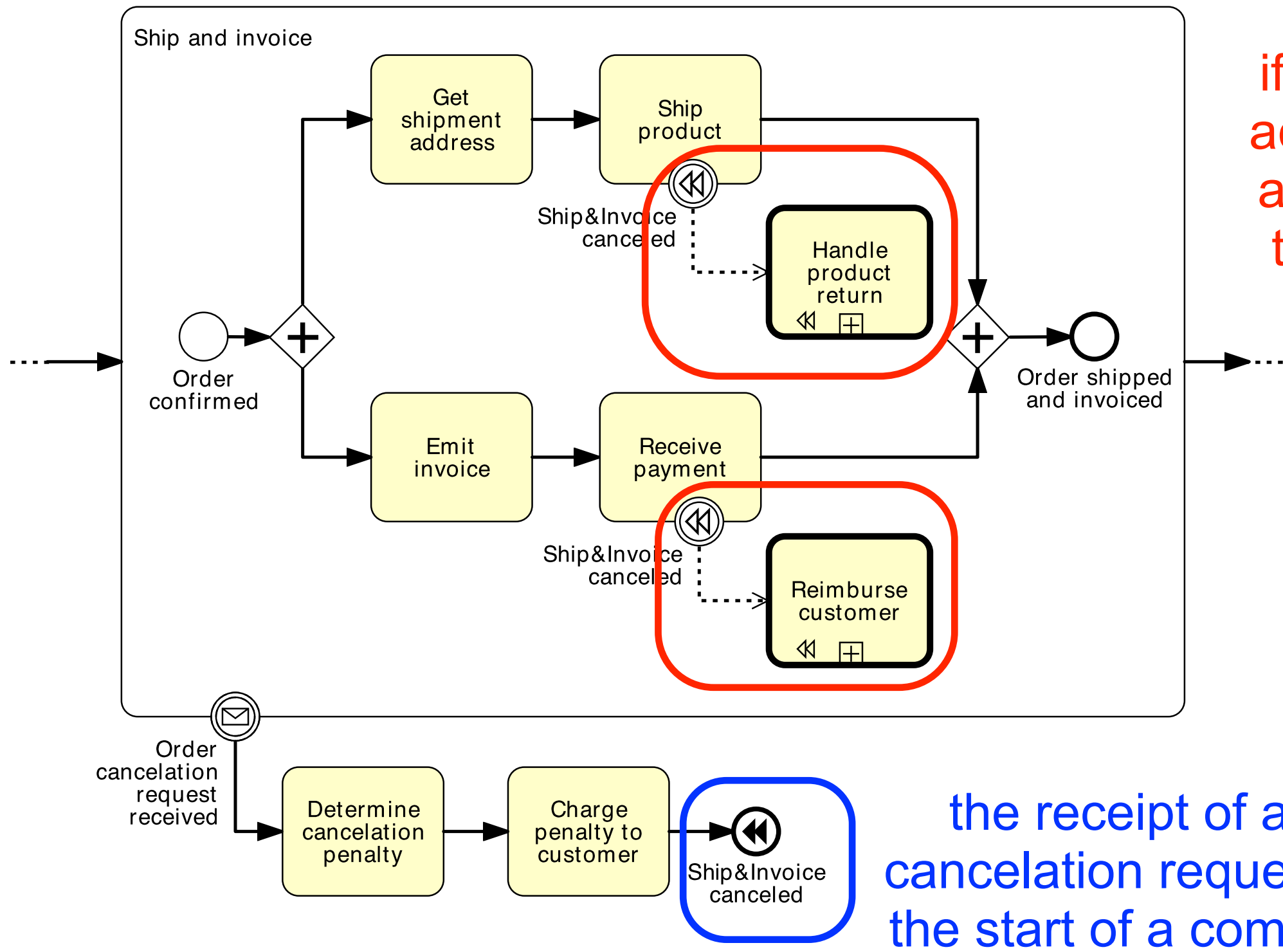
Recovery from faults: image manipulation



Intermediate time out and a loop



Compensations



if the compensable activities have been already completed, then they must be compensated

the receipt of an order cancellation request triggers the start of a compensation

Exercises

Model the following process fragment:

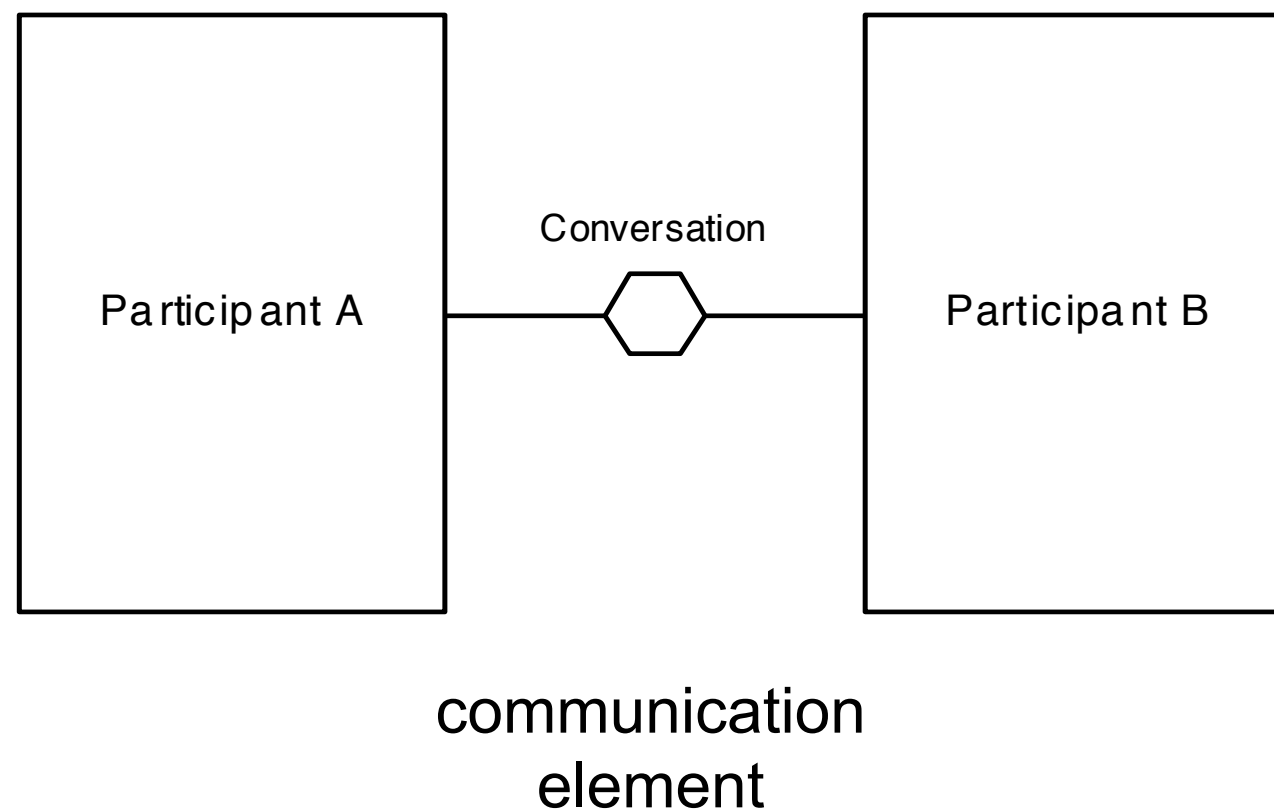
After a car accident, a statement is sought from two witnesses out of the five that were present, in order to lodge the insurance claim.

As soon as the first two statements are received, the claim can be lodged with the insurance company without waiting for the other statements.

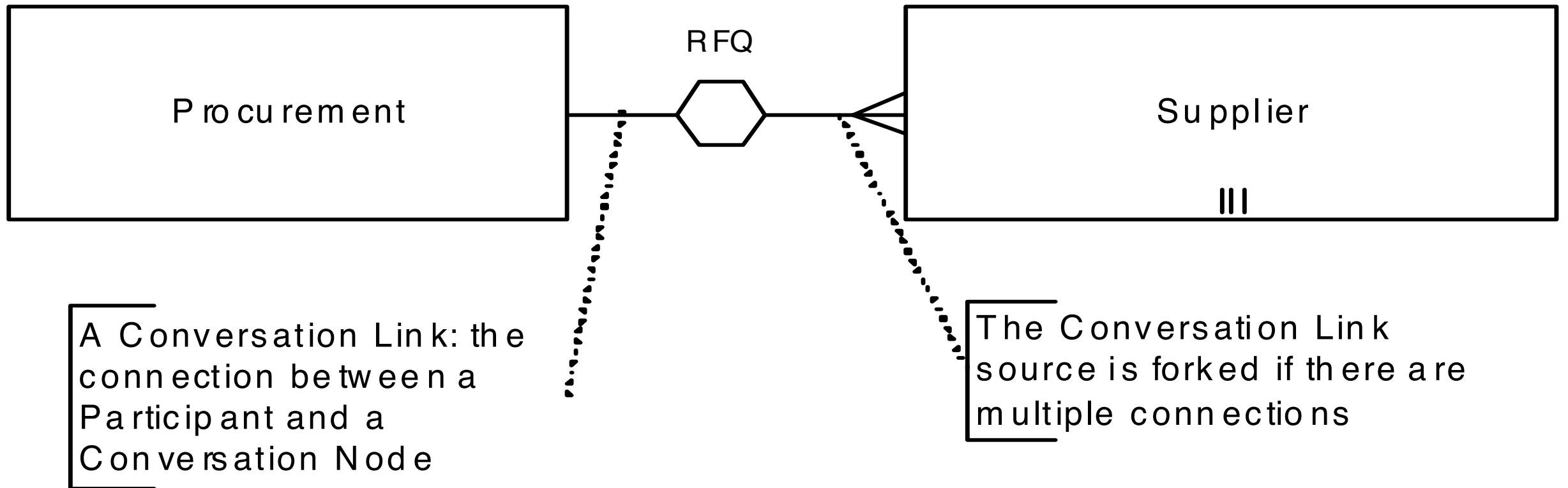
Conversations,
choreographies, and
collaborations

Conversation

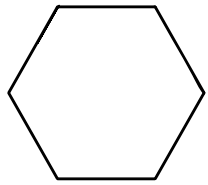
A **Conversation** is the logical relation of (correlated) Message exchanges



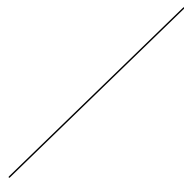
Conversation links



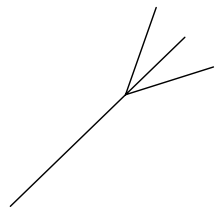
Conversation diagram



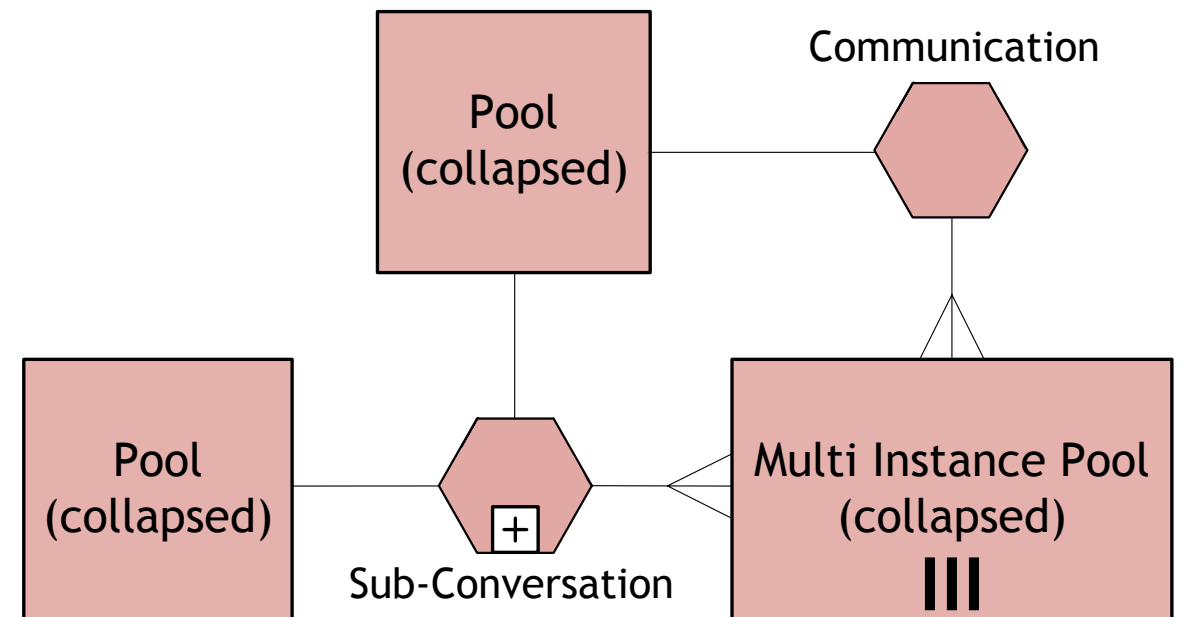
A **Communication** defines a set of logically related message exchanges. When marked with a **+** symbol it indicates a Sub-Conversation, a compound conversation element.



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



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



Choreography

The behaviour of different Conversations is modelled through separate Choreographies

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

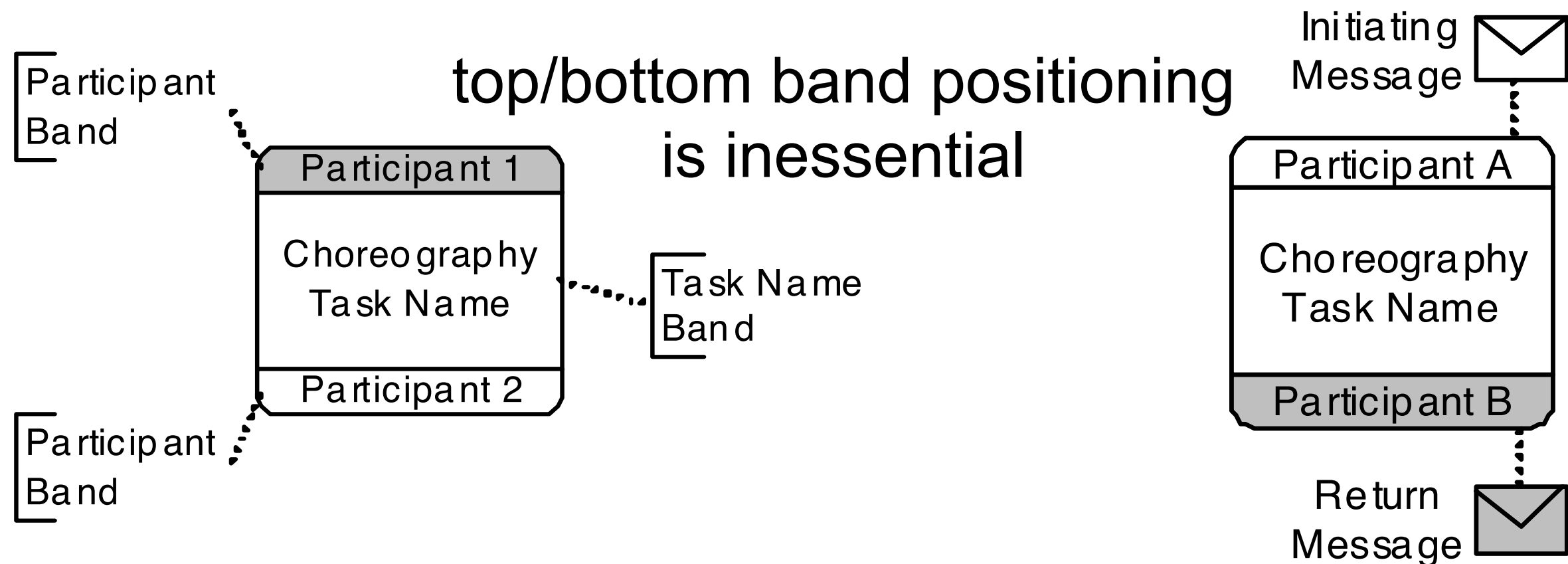
A choreography does not exist in a pool and it is not executable

It describes how the participants are supposed to behave

Choreography task

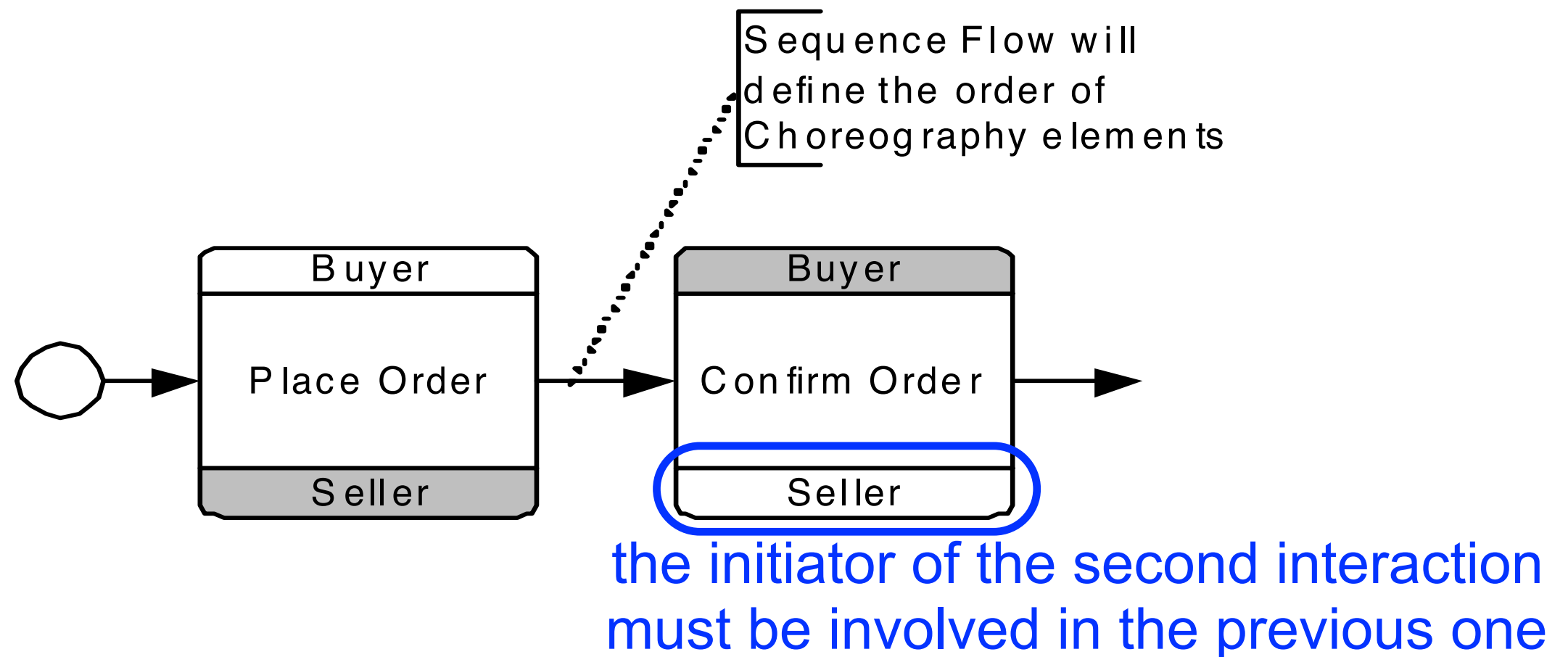
A **Choreography task** is an activity in a choreography that consists of a set (one or more) Message exchanges

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

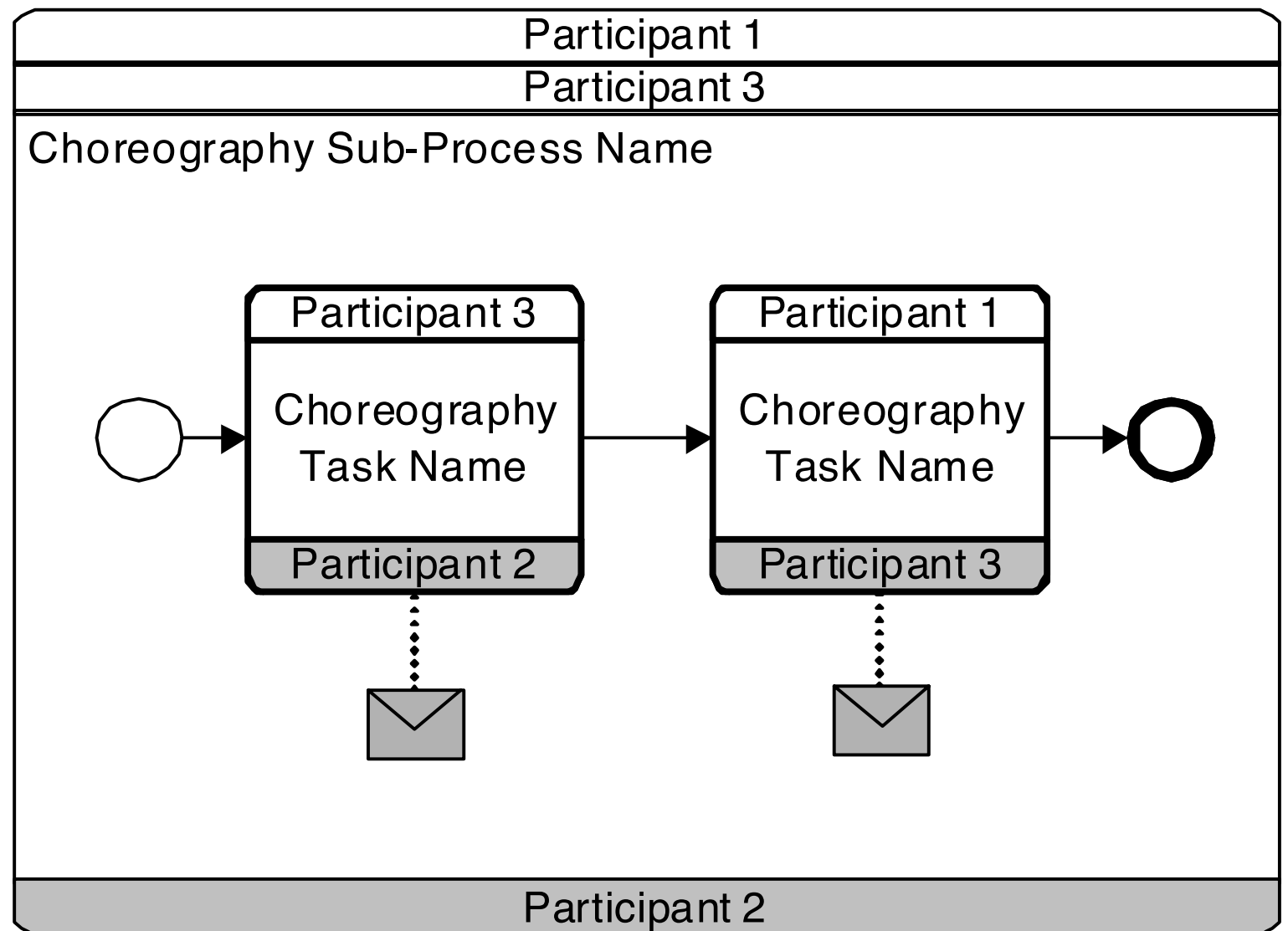
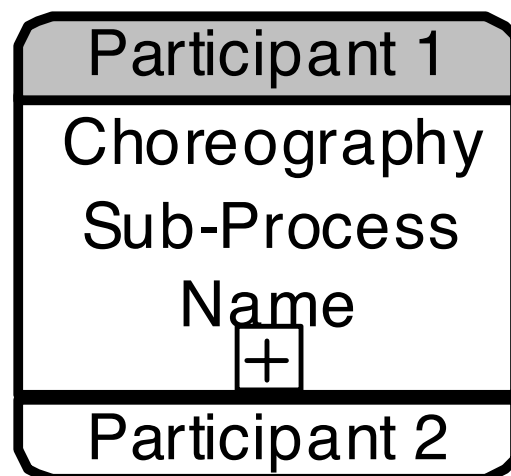


Sequence flow in a choreography

Sequence Flow are used within Choreographies to show the sequence of the Choreography Activities, Events, and Gateways

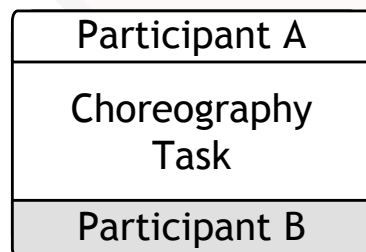


Choreography task collapsed / expanded



Choreography diagram

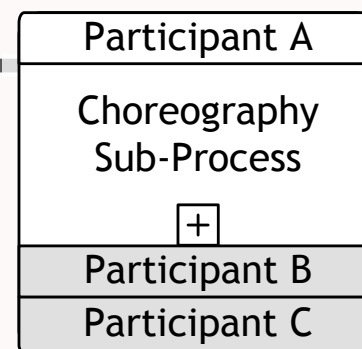
Choreographies



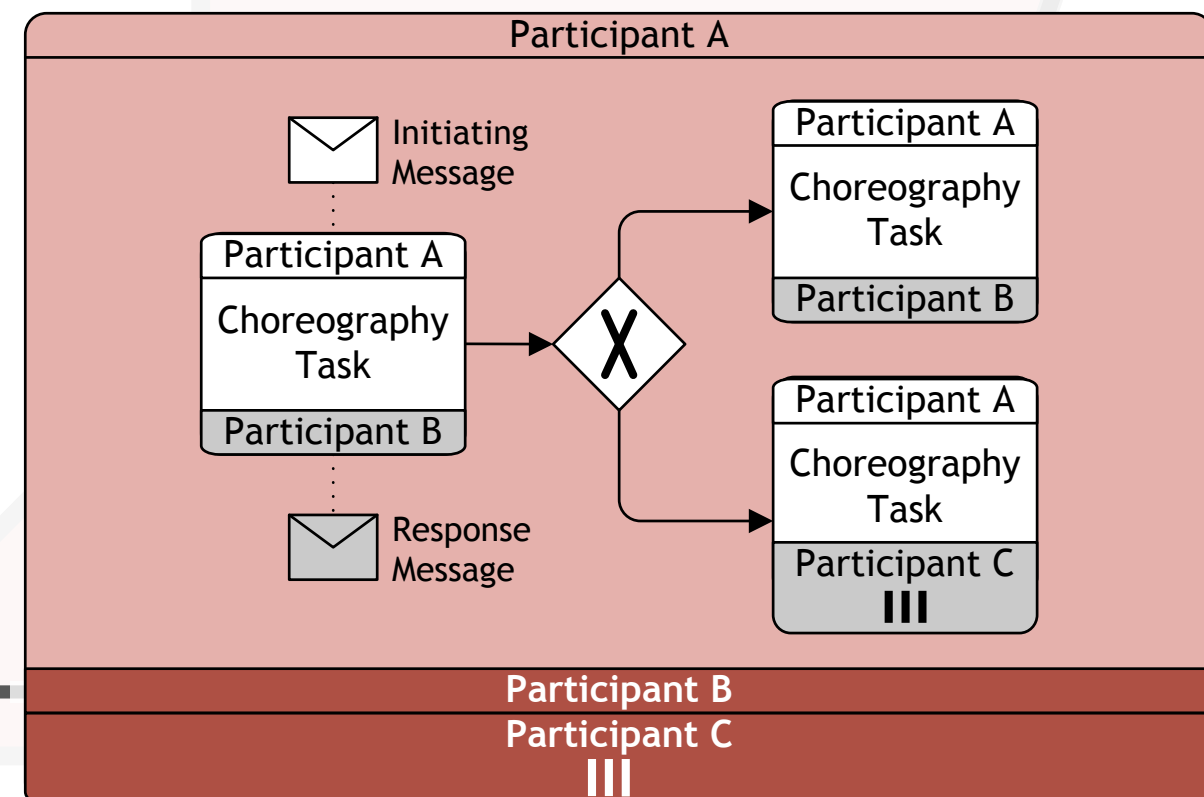
A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

III

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



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



Collaboration

A **Collaboration** contains two or more Pools, representing the Participants in the Collaboration

A Pool may be empty or show a Process within

The Message exchange is shown by a Message Flow that connects Pools or the objects within the Pools
The Messages associated with the Message Flow may also be shown

Choreographies may be shown “in between” the Pools as they bisect the Message Flow

Public processes (collaborative B2B)

Public processes (also called **abstract** processes) depicts the interaction between two or more business entities

Typically designed as a global view:

Sequence of activities and
the message exchange patterns between participants

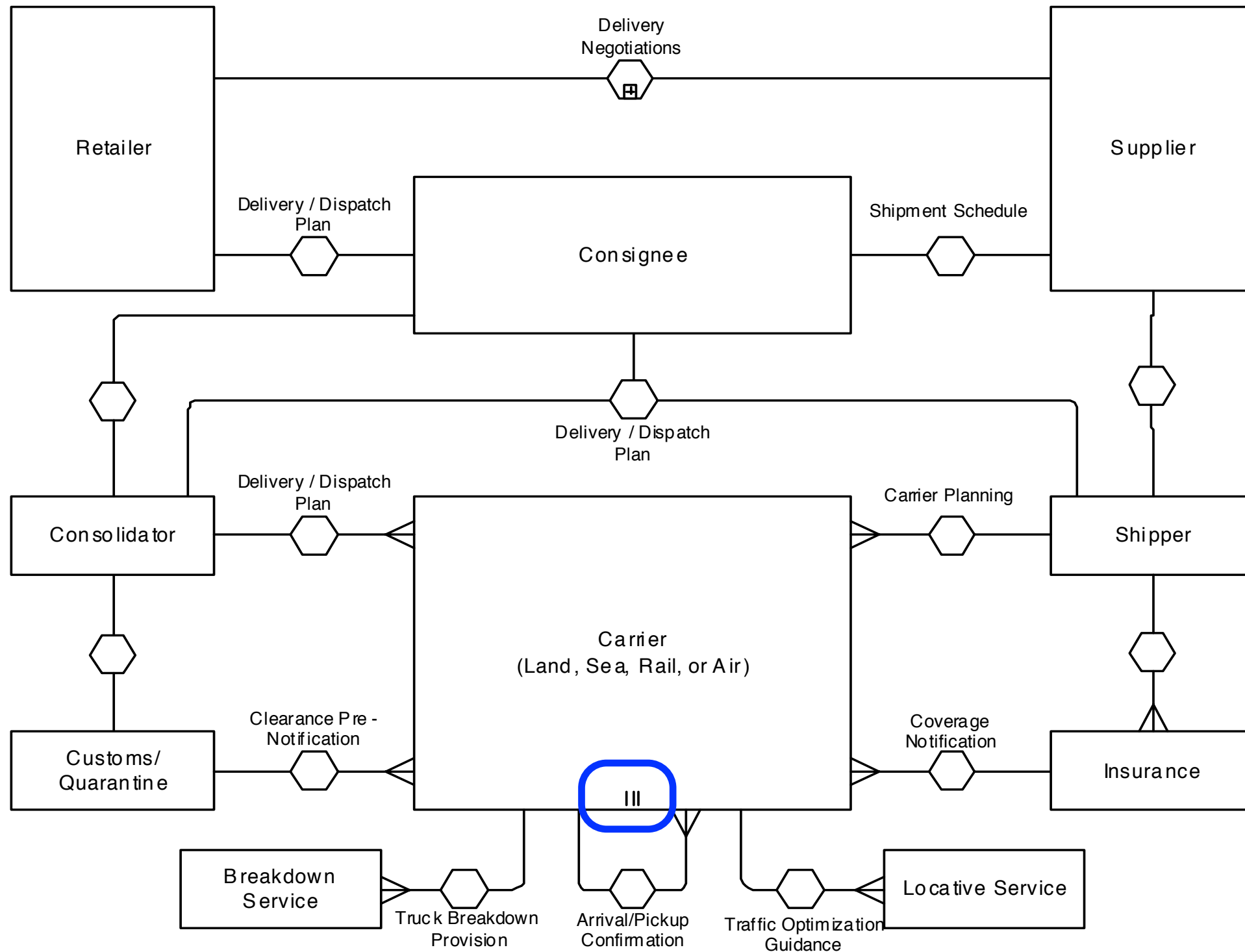
Activities of collaboration participants as “touch points”
(visible to the public)

Actual processes are likely to have more activities
(private or internal view)

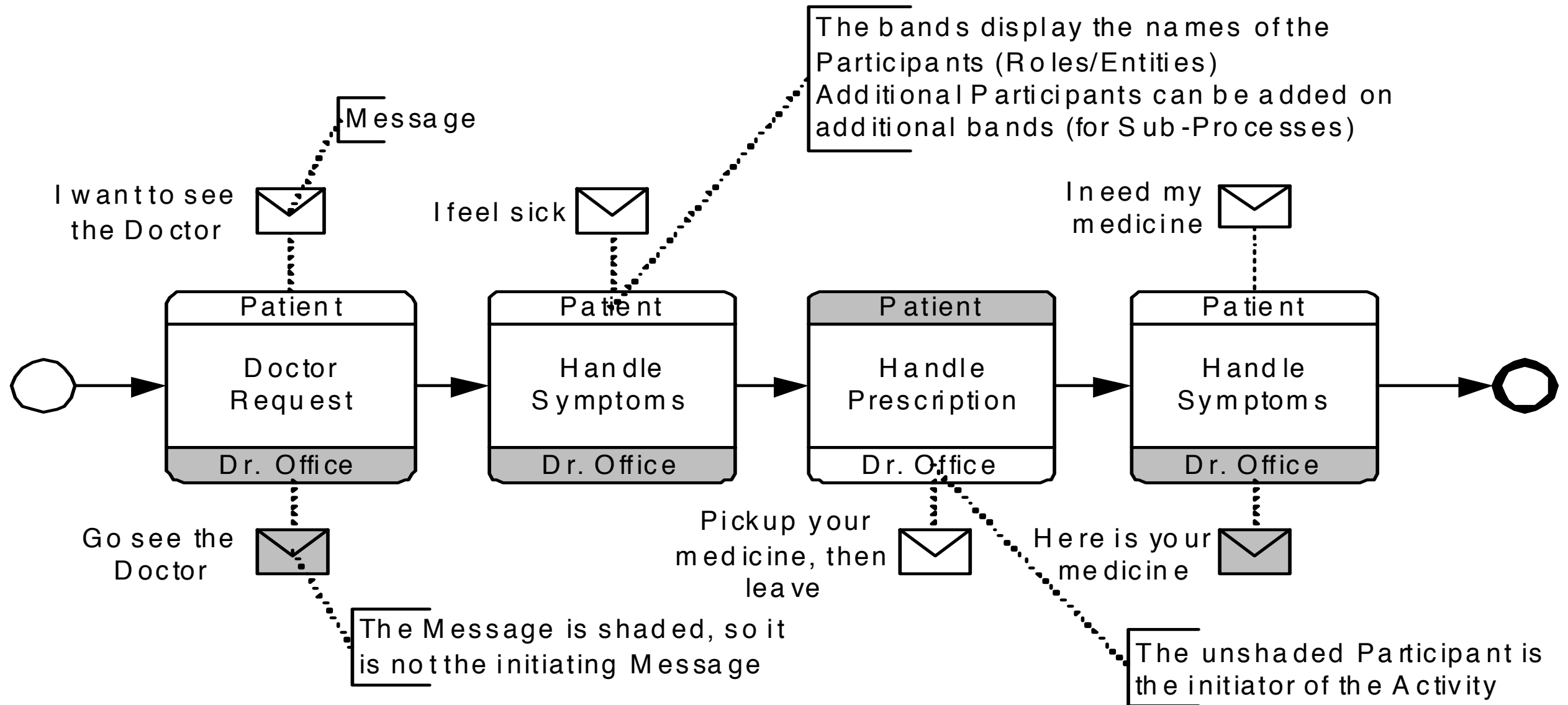
Examples

(a taste of BPMN)

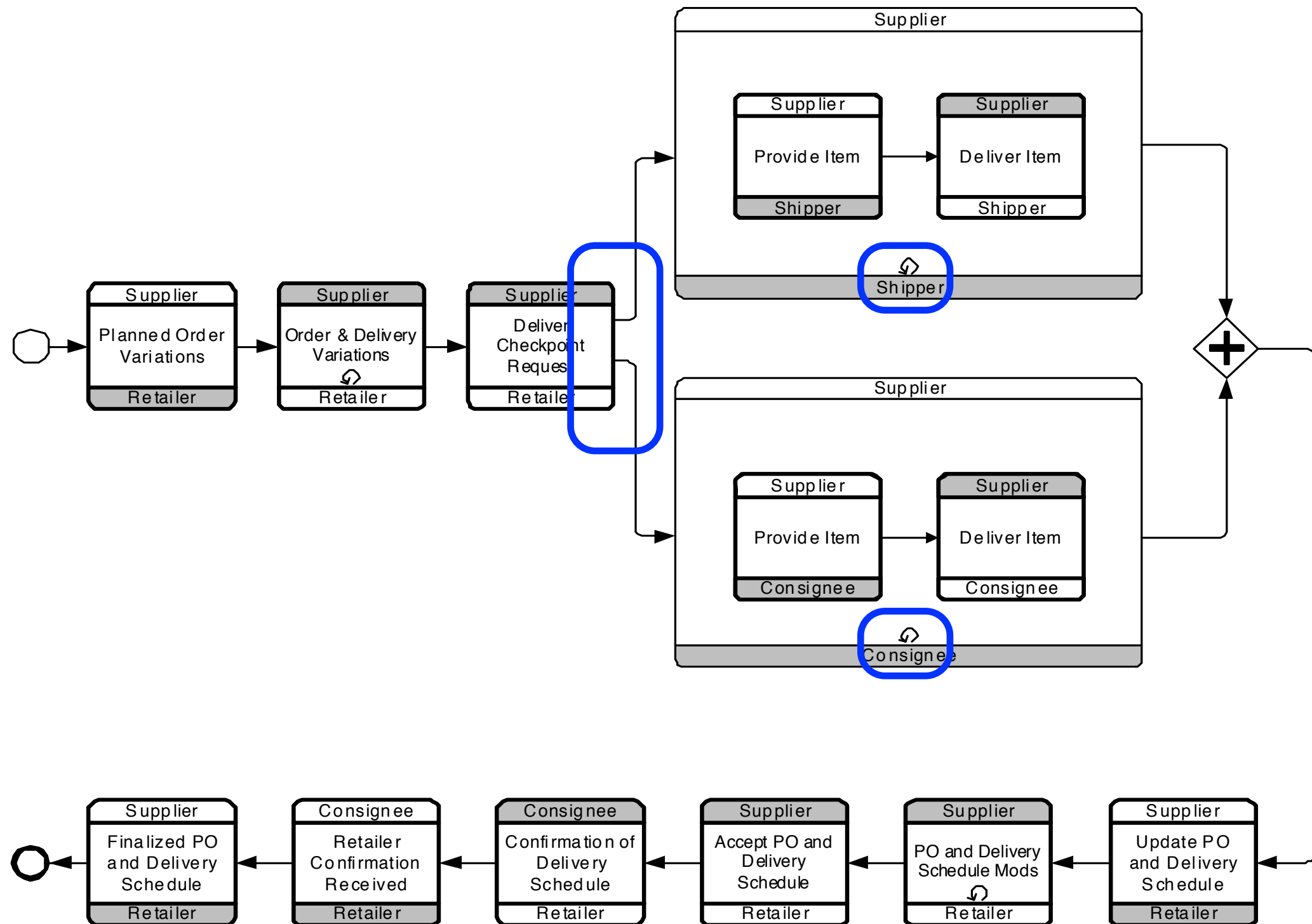
A conversation



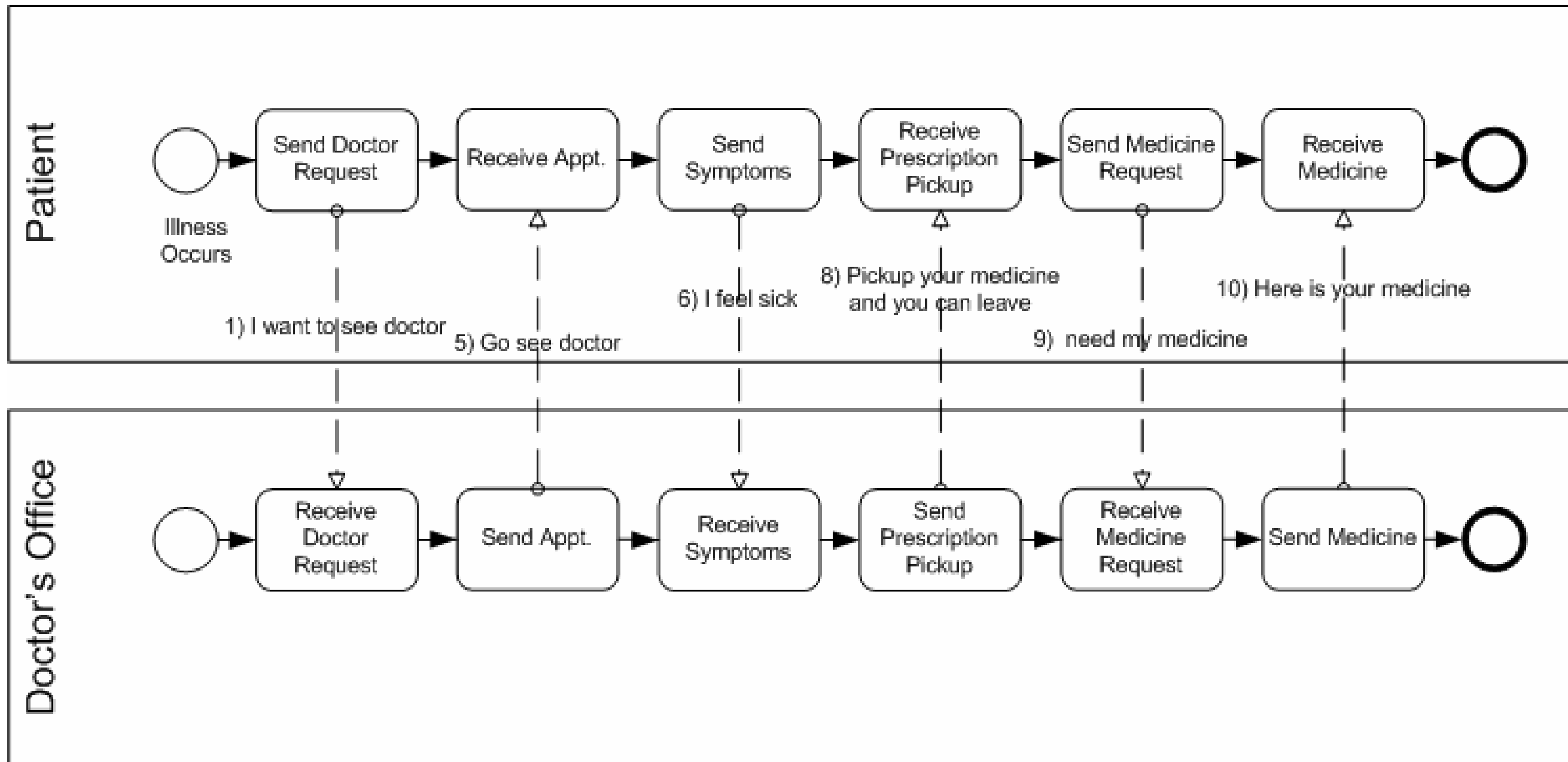
A choreography



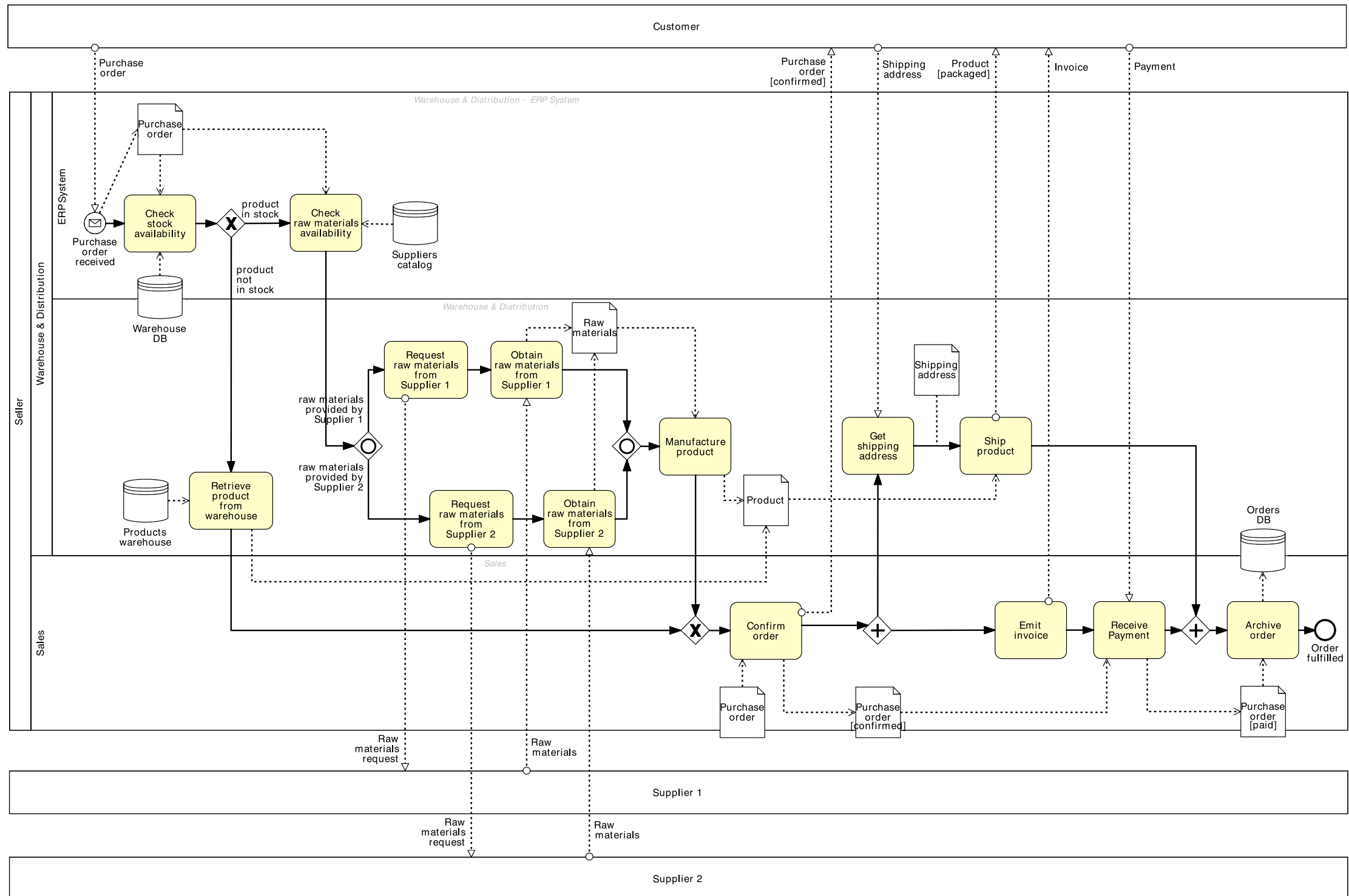
Another choreography



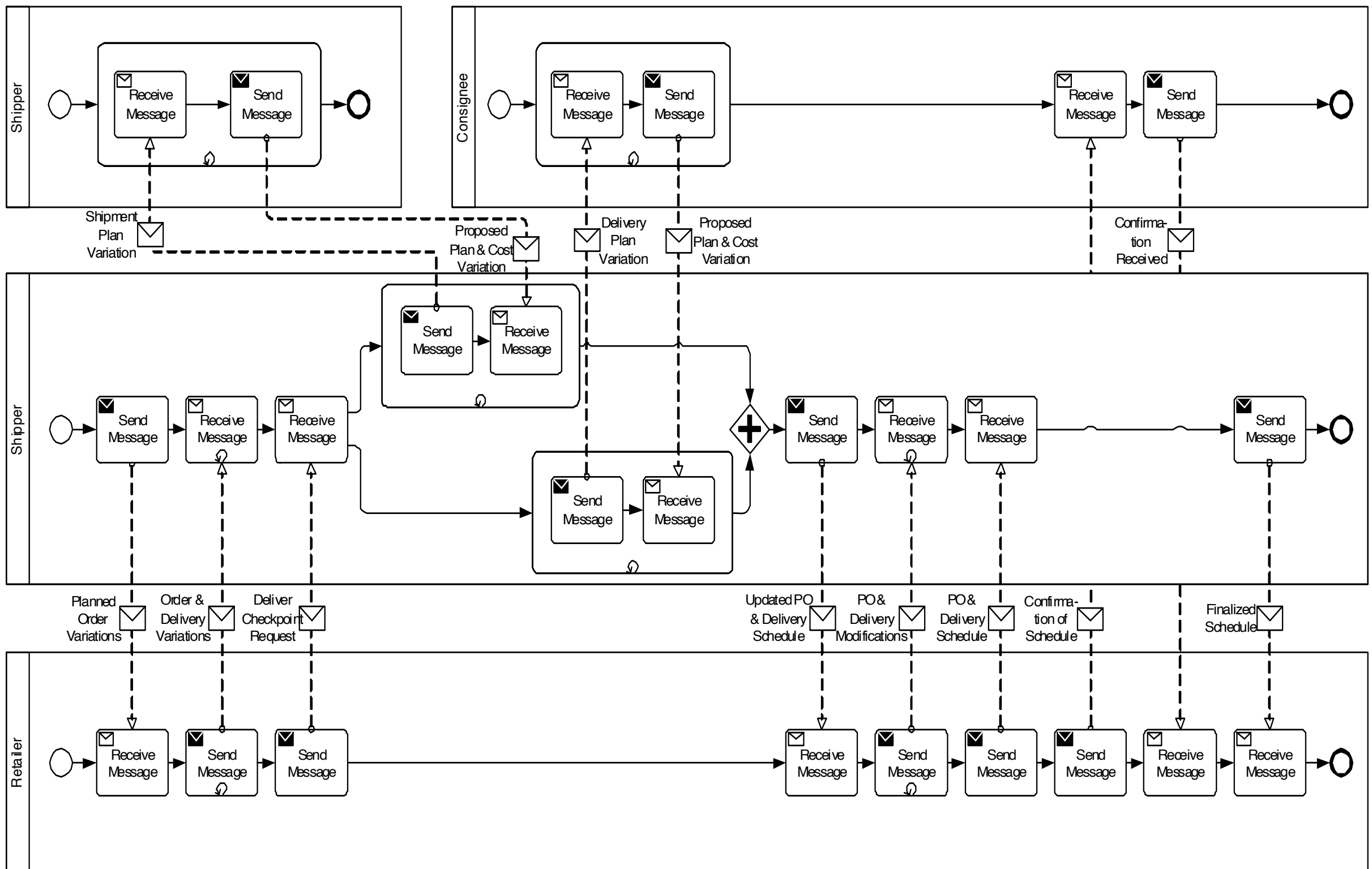
A collaboration with two pools



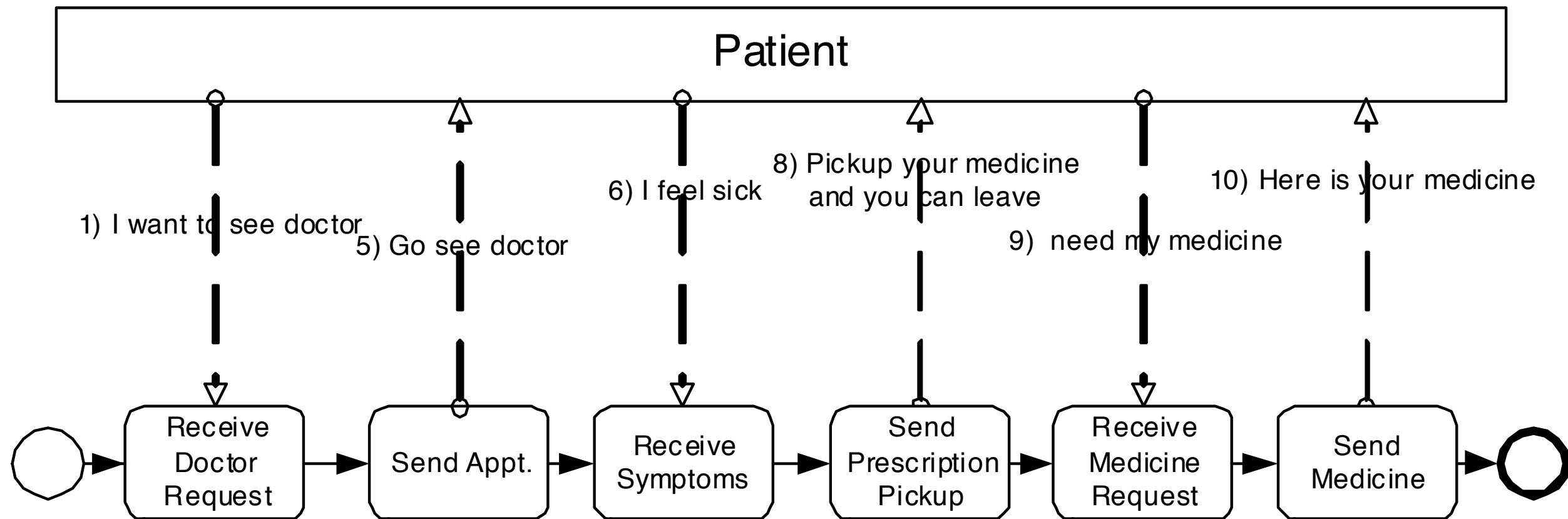
A collaboration diagram: order fulfillment



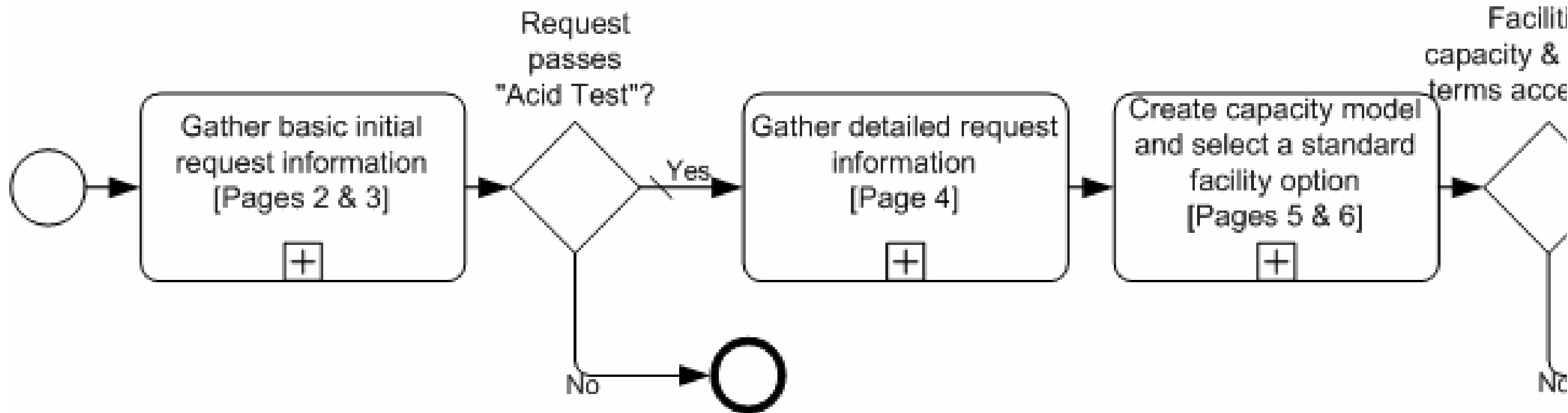
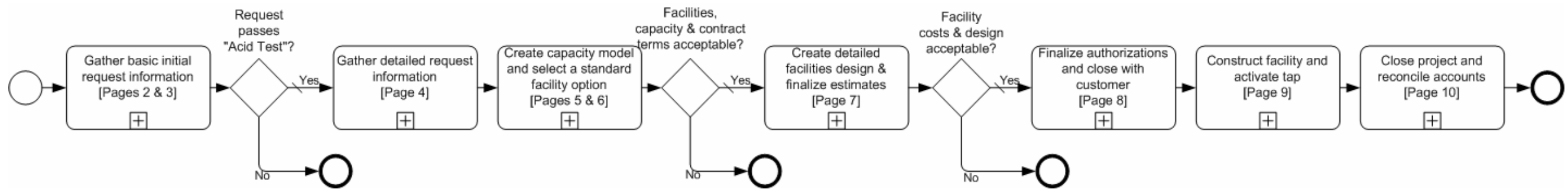
A collaboration diagram



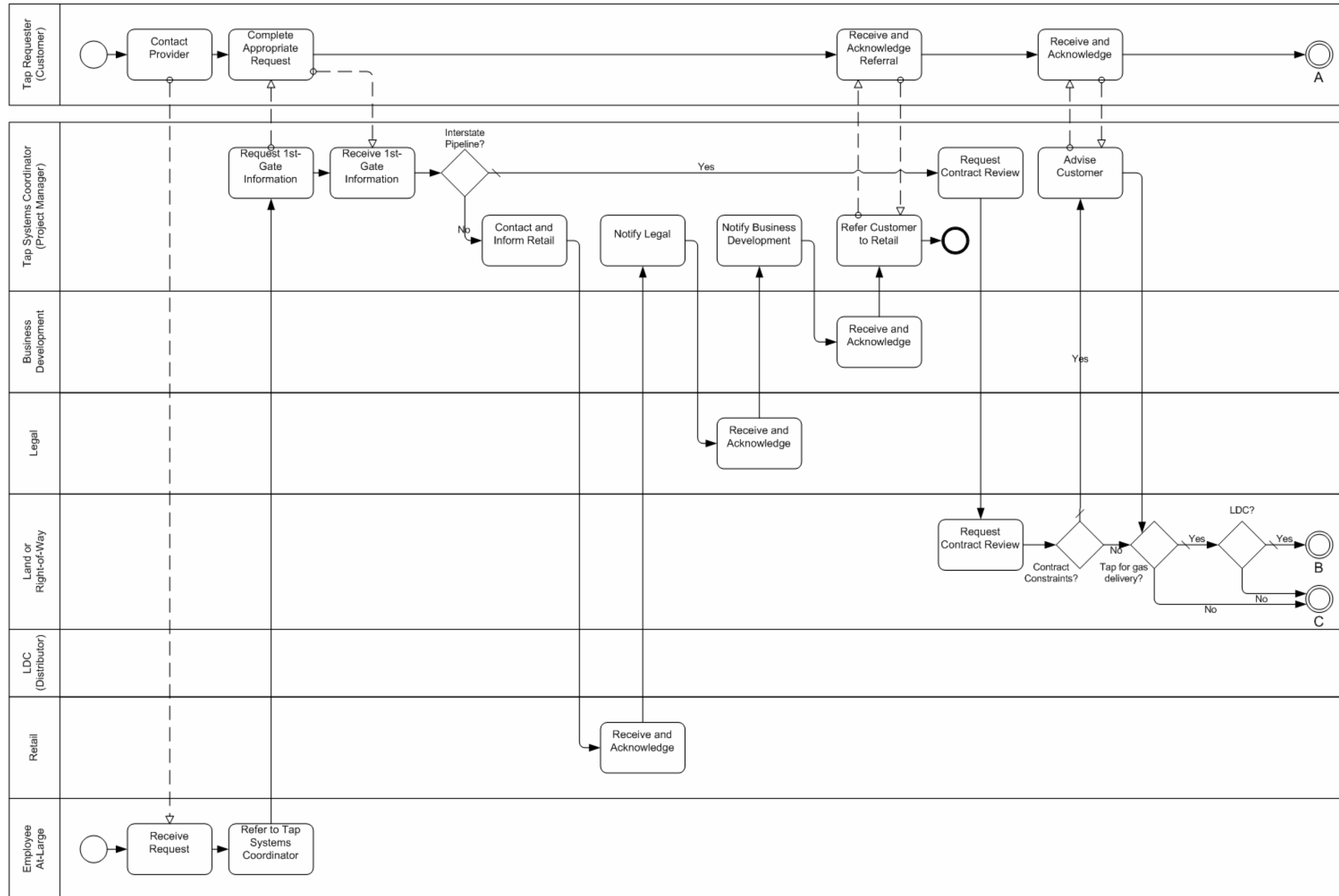
A public process



An abstract process



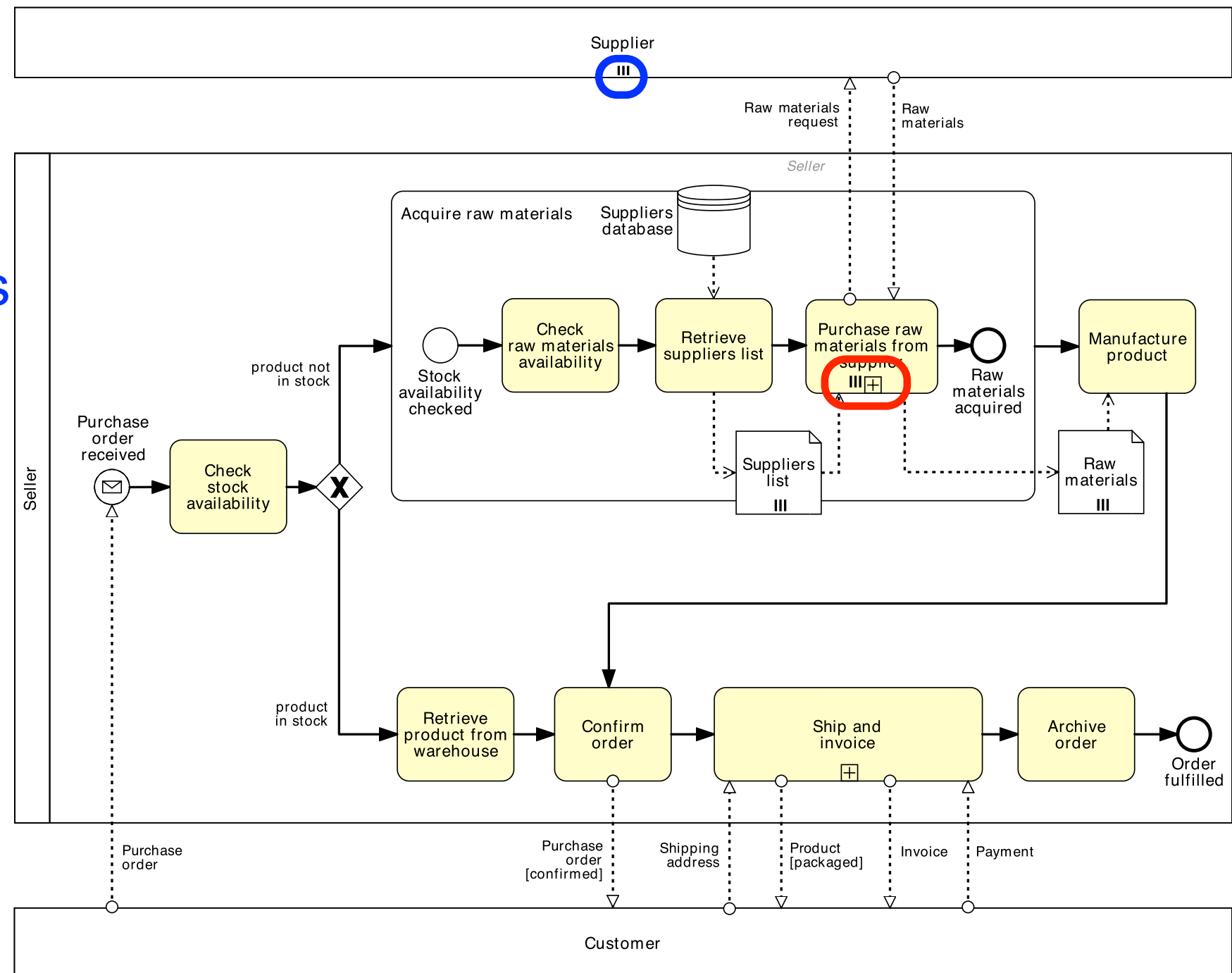
An internal, interactive process



Multi-instance pools: order fulfillment

the multi-instance
symbol annotation
denotes
a set of resources
with similar characteristics

multi-instance
sub-process



Exercises: loan application

Extend the loan application model by representing the interactions between the loan provider and the applicant.

Semantics and analysis of business process models in BPMN

Remco M. Dijkman^a, Marlon Dumas^{b,c}, Chun Ouyang^{c,*}

^a *Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, 5600 MB, The Netherlands*

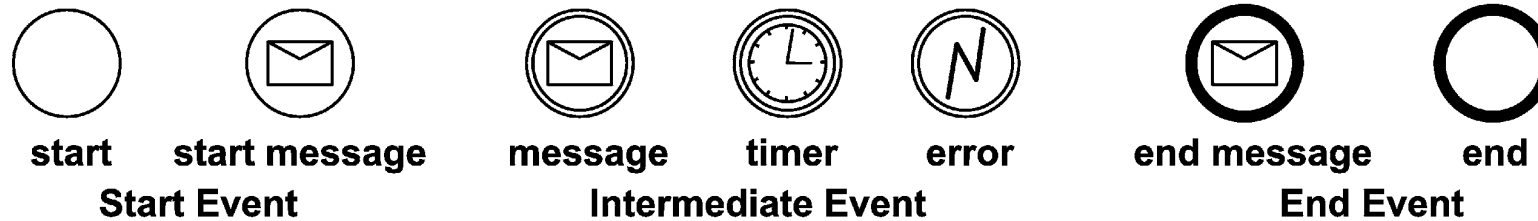
^b *Institute of Computer Science, University of Tartu, J Liivi 2, Tartu 50409, Estonia*

^c *Faculty of Information Technology, Queensland University of Technology, G.P.O. Box 2434, Brisbane, Qld 4001, Australia*

From BPMN to Petri nets

Overview

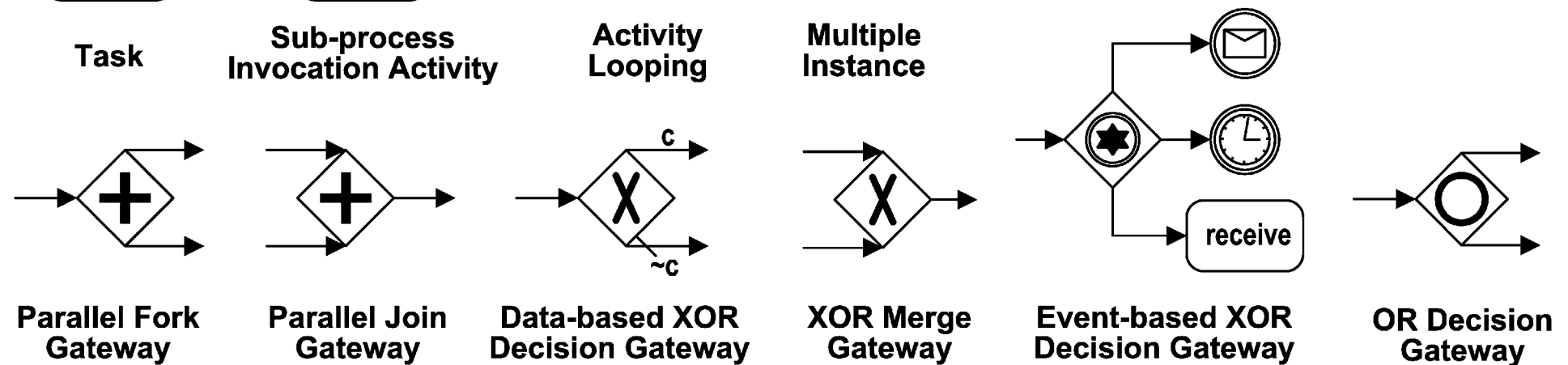
EVENT



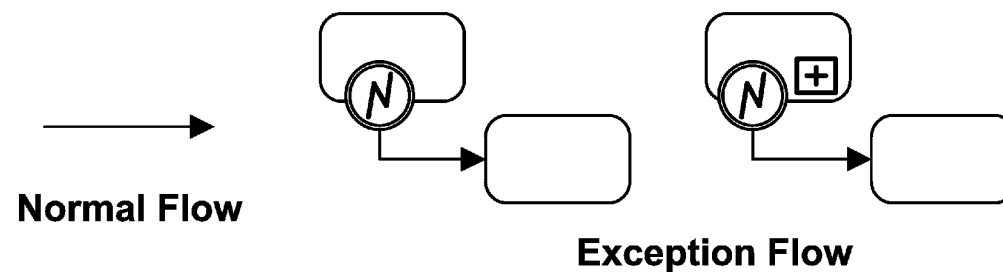
ACTIVITY



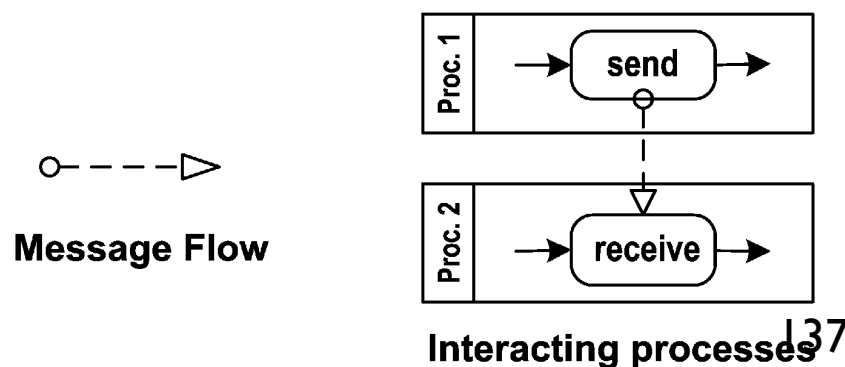
GATEWAY



SEQUENCE FLOW



MESSAGE FLOW



[Note]:

1. Apart from intermediate error events, intermediate message or timer events may also be the source of exception flows.
2. A message flow may link task to task, end event to task, task to start event, and end event to start event.

Simplified BPMN

a start / exception event has just one outgoing flow
and no incoming flow

an end event has just one incoming flow
and no outgoing flow

all activities and intermediate events have exactly
one incoming flow and one outgoing flow

all gateways have either
one incoming flow (and multiple outgoing)
or one outgoing flow (and multiple incoming)

Simplified BPMN

The previous constraints are no real limitation:

event or activities with multiple incoming flows
insert a preceding XOR-join gateway

event or activities with multiple outgoing flows
insert a following AND-split gateway

gateways with multiple incoming and outgoing flows
decompose in two gateways

insert start / end event if needed

Simplified BPMN

No link events

they are just a notational convenience
to spread a model into several pages
(no effect on the semantics)

No transactions and compensations

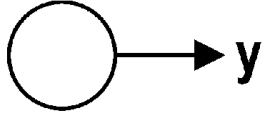
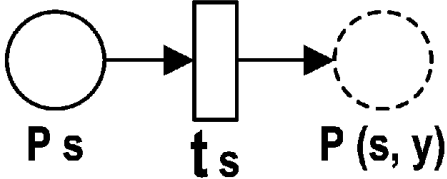
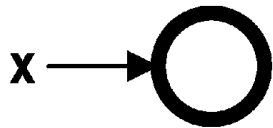
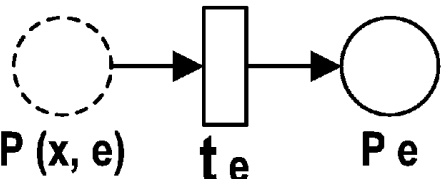
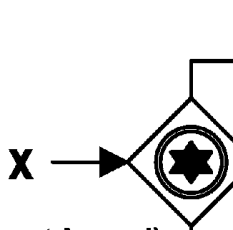
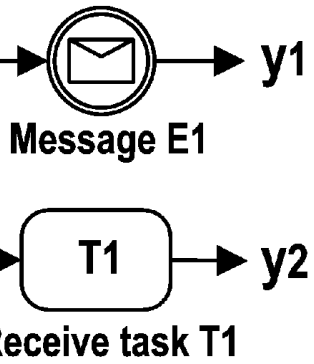

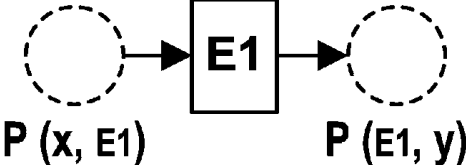

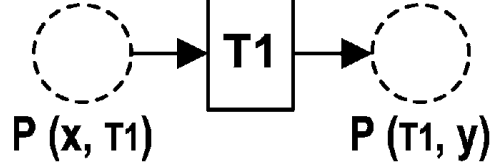
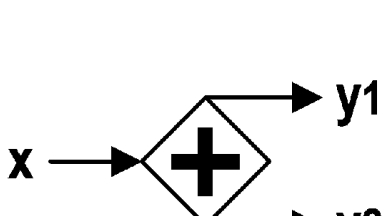
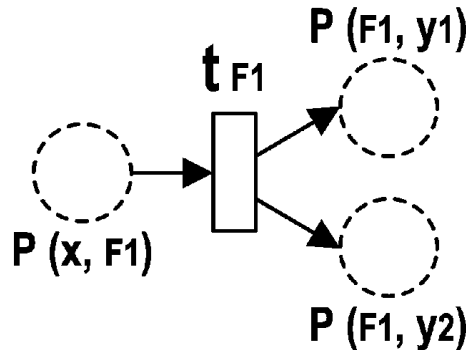
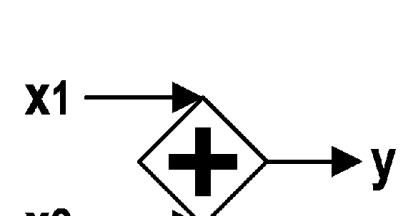
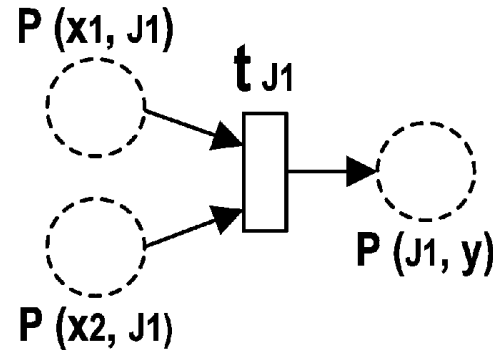
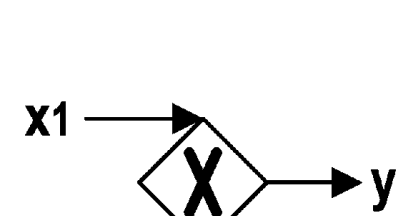
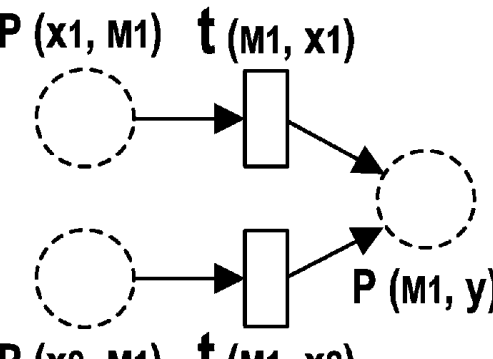
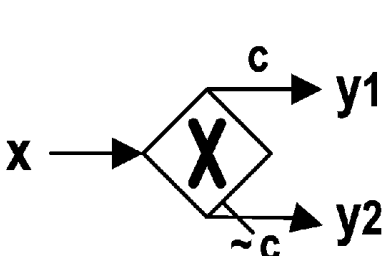
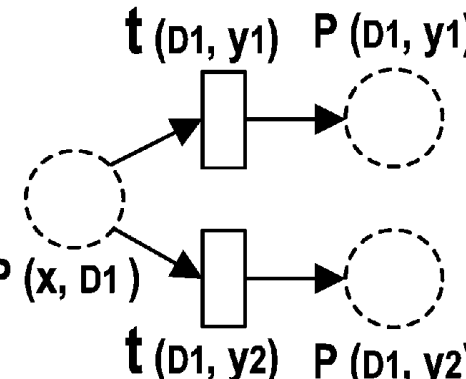
Limited form of sub-processing

no OR-split

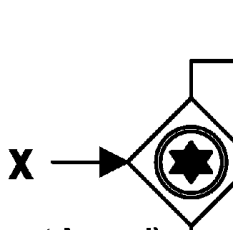
(can be expressed in terms of AND-split and XOR-split)

no OR-join

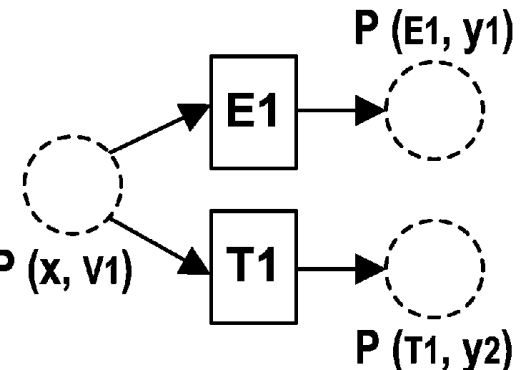
Task, events and gateways as nets

BPMN Object	Petri-net Module	BPMN Object	Petri-net Module	BPMN Object	Petri-net Module
 Start s	 P_s t_s $P(s, y)$	 End e	 $P(x, e)$ t_e P_e	 (Event-based) Decision V1	 $P(x, V1)$ t_{V1} $P(V1, y1)$ $T1$ $P(T1, y2)$
 Message E	 $P(x, E1)$ $E1$ $P(E1, y)$	 Task T	 $P(x, T1)$ $T1$ $P(T1, y)$	 Fork F1	 $P(x, F1)$ t_{F1} $P(F1, y1)$ $P(F1, y2)$
 Join J1	 $P(x1, J1)$ $P(x2, J1)$ t_{J1} $P(J1, y)$	 Merge M1	 $P(x1, M1)$ $P(x2, M1)$ t_{M1} $P(M1, y)$	 (Data-based) Decision D1	 $P(x, D1)$ t_{D1y1} $P(D1, y1)$ t_{D1y2} $P(D1, y2)$

BPMN Object

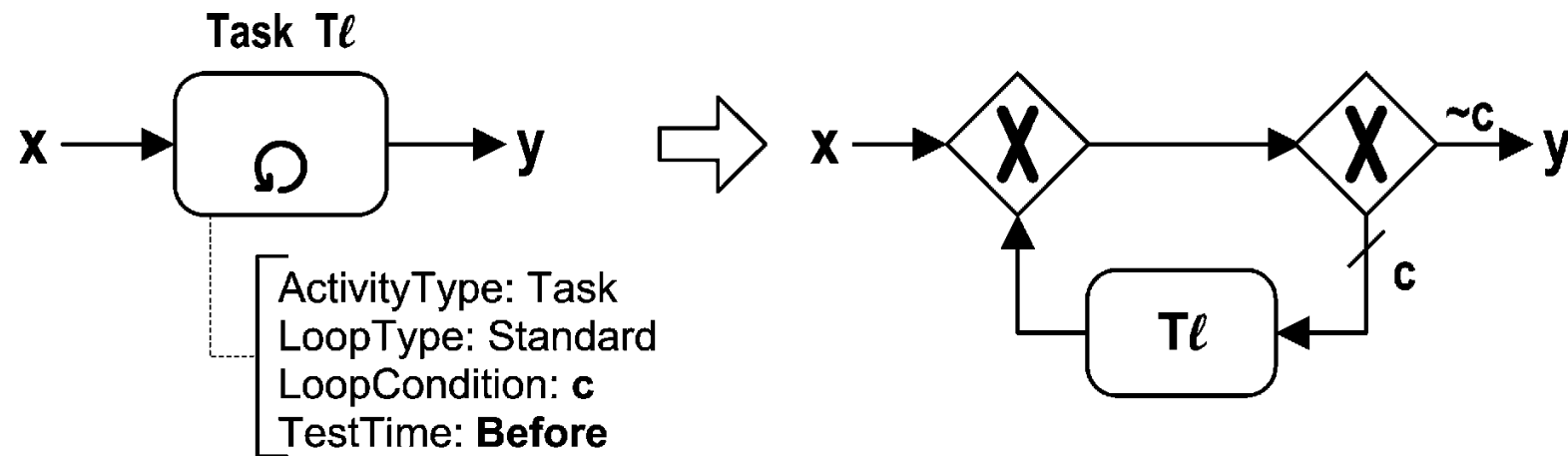

 (Event-based) Decision V1

Petri-net Module

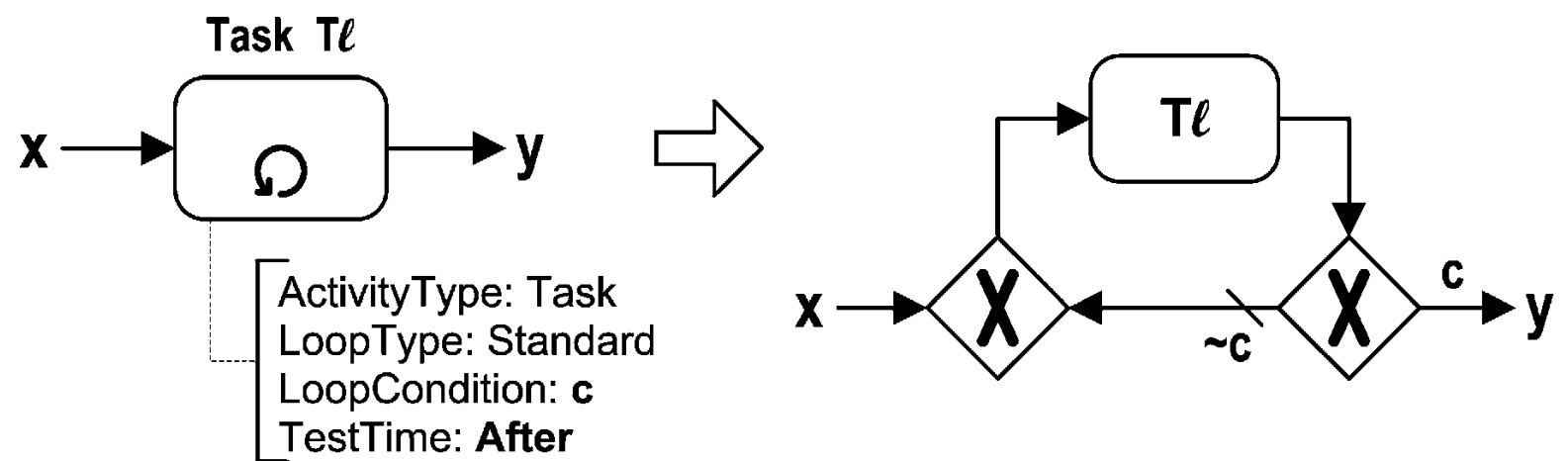

 $P(x, V1)$ $E1$ $P(E1, y1)$ $T1$ $P(T1, y2)$

[Note]:
 x, x1 or x2 represents an input object, and y, y1 or y2 represents an output object.

Activity looping

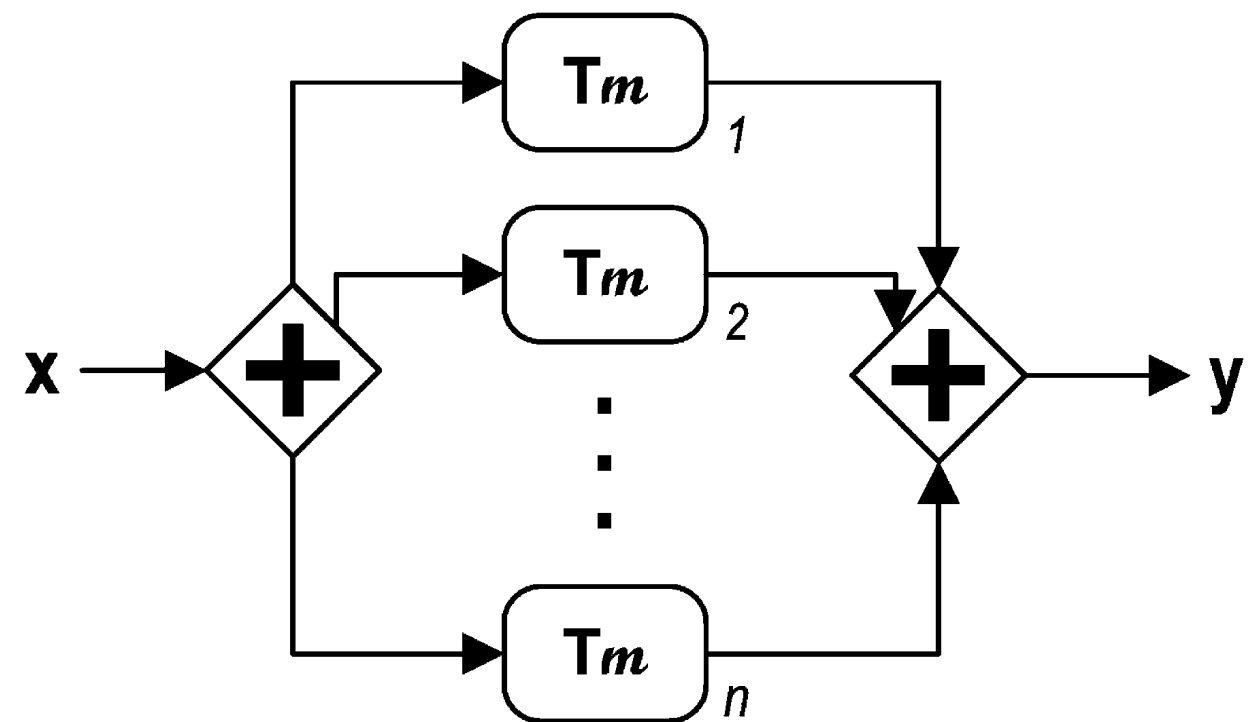
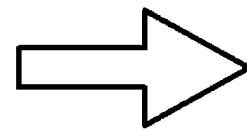
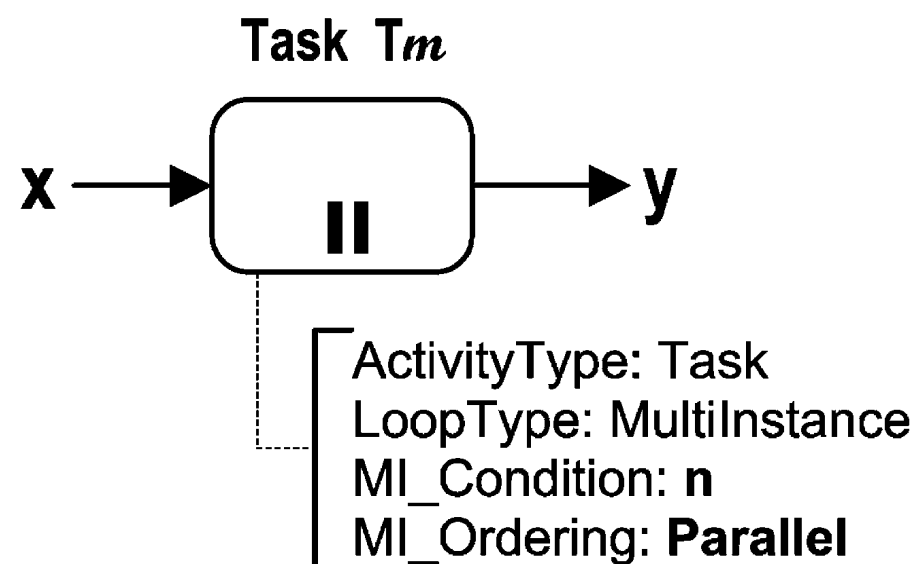


(a) “while-do” loop

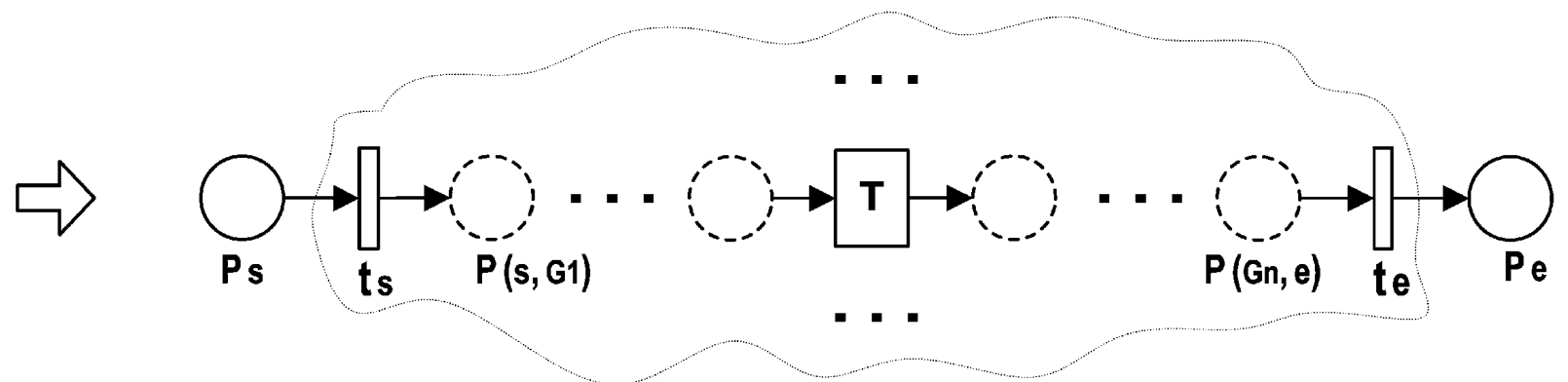
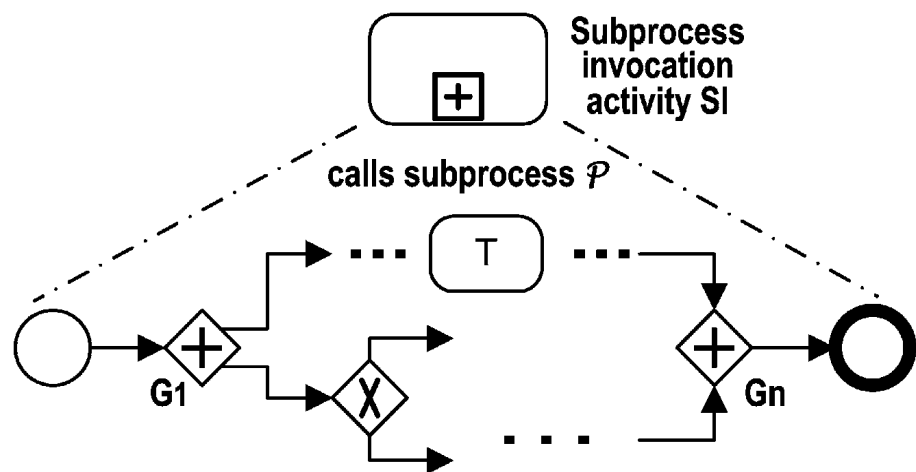
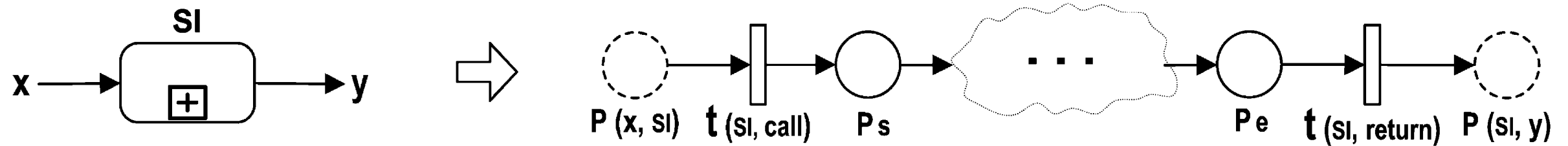


(b) “do-until” loop

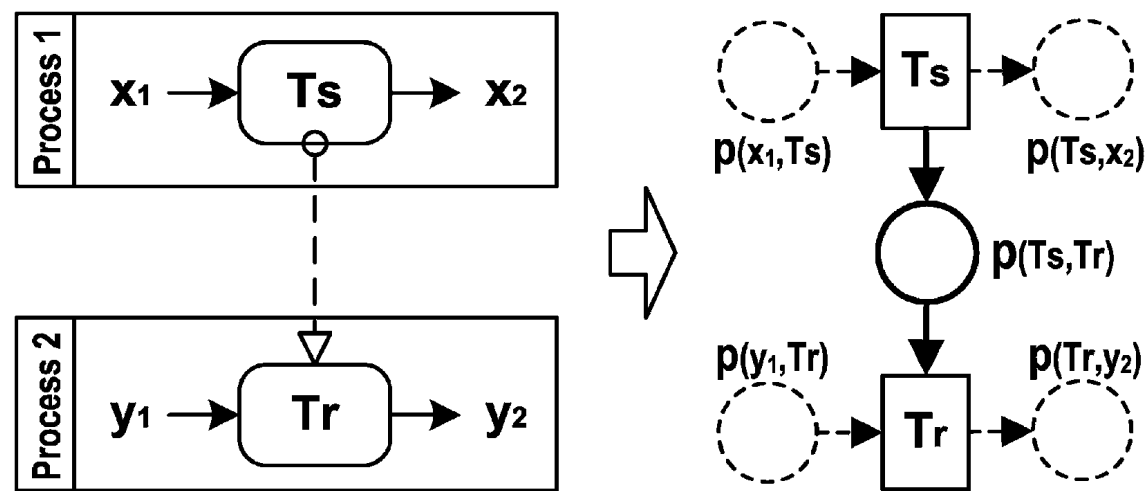
Multiple instances (design-time bounded)



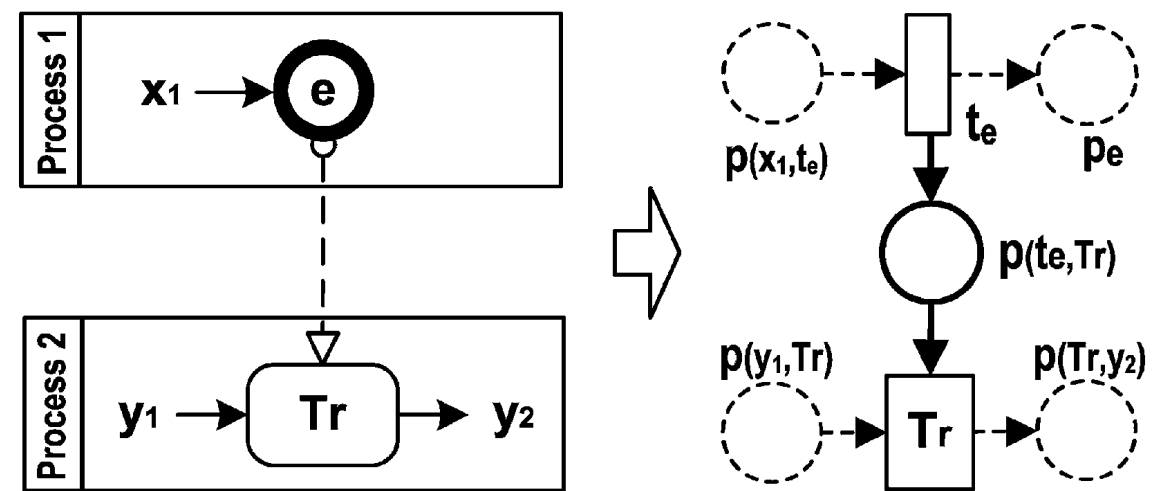
Sub-processes



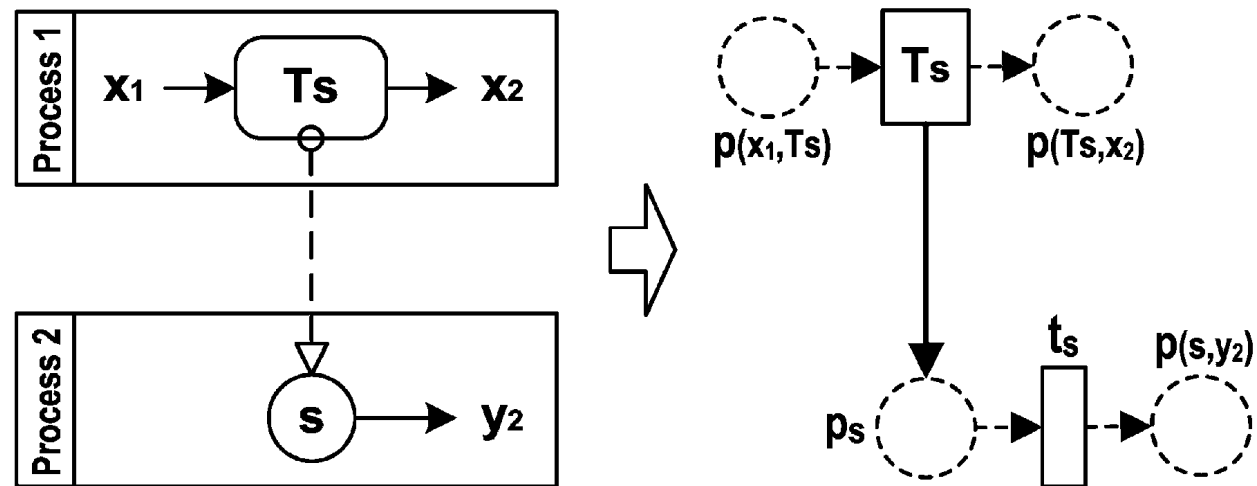
Message flow



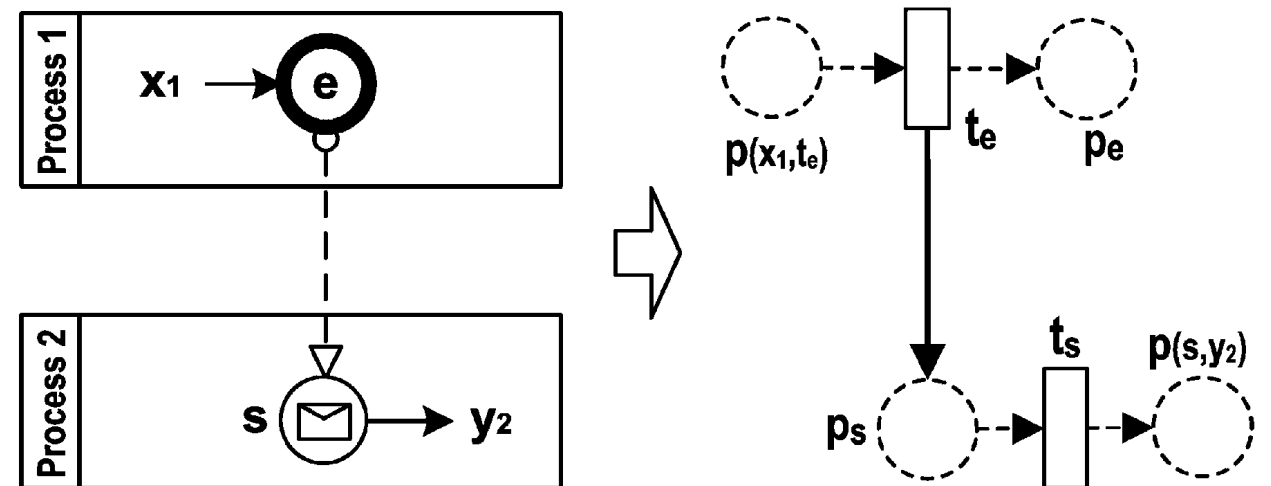
(a) task to task



(b) end event to task

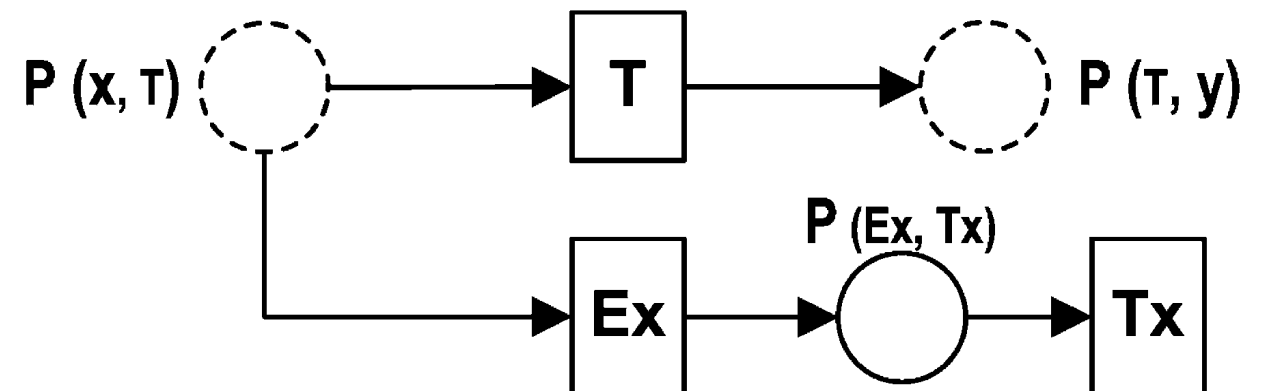
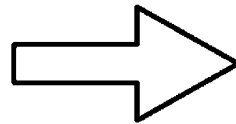
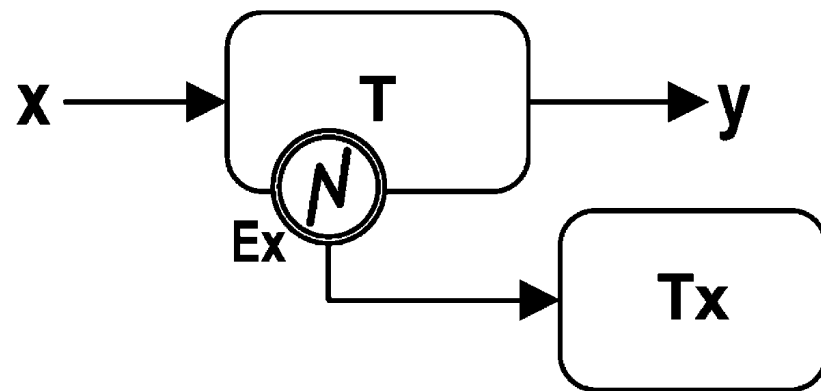


(c) task to start event



(d) end event to start event

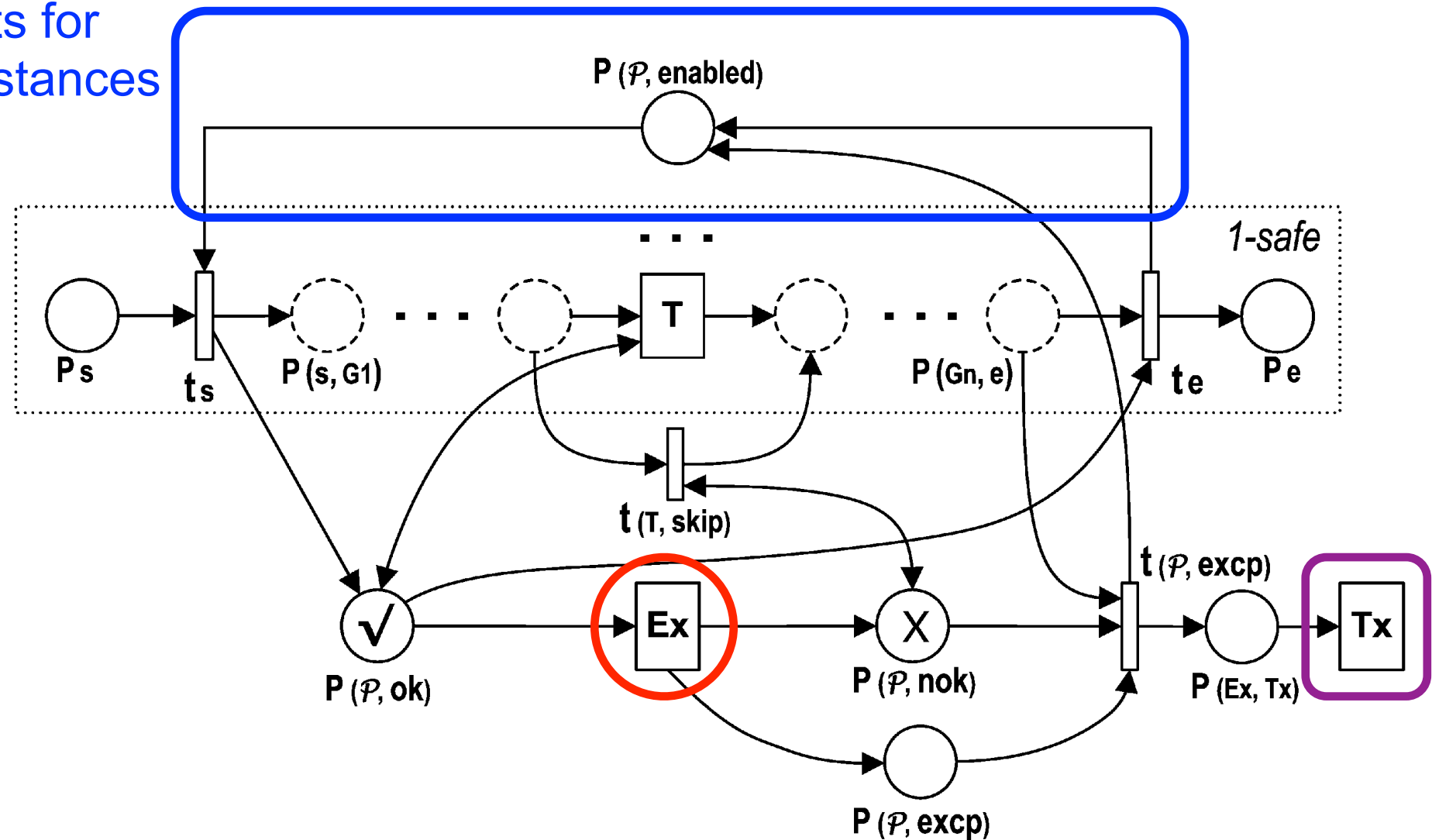
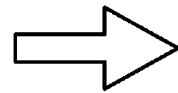
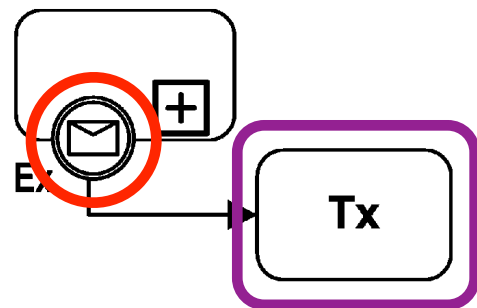
Exception handling: single task



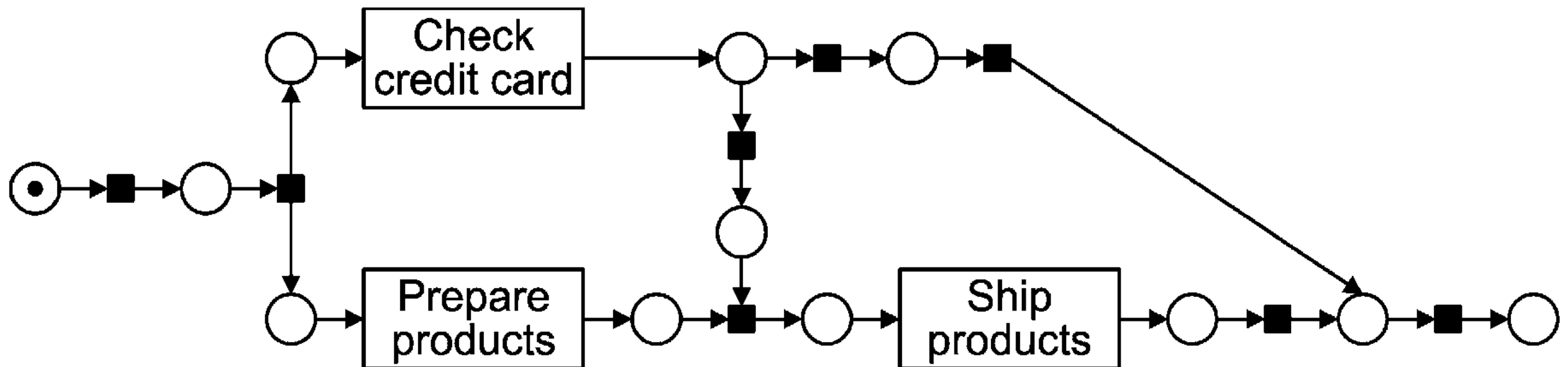
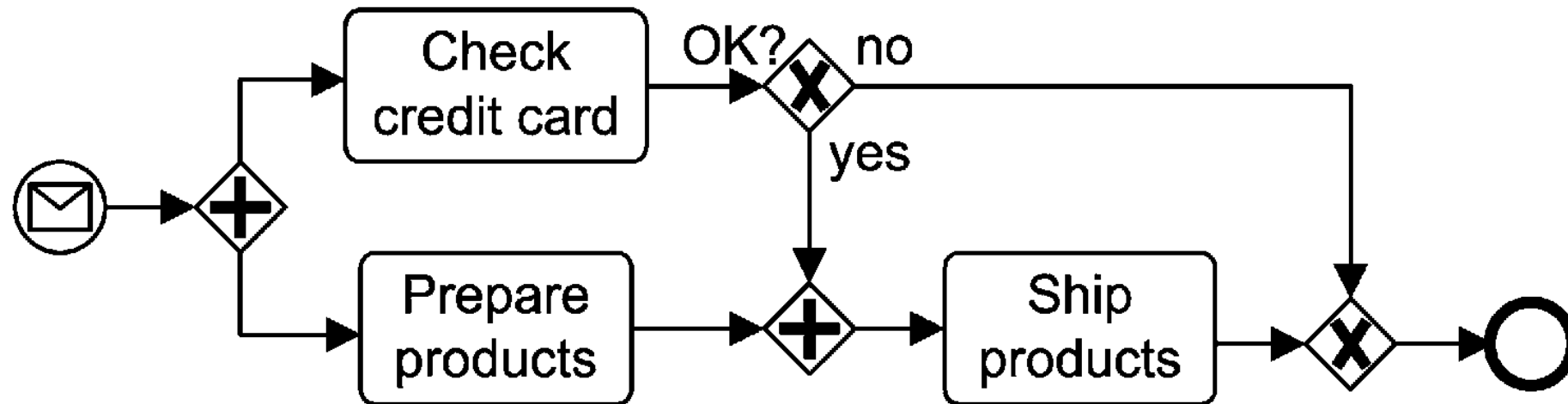
Exception handling: sub-processes

accounts for
multiple instances

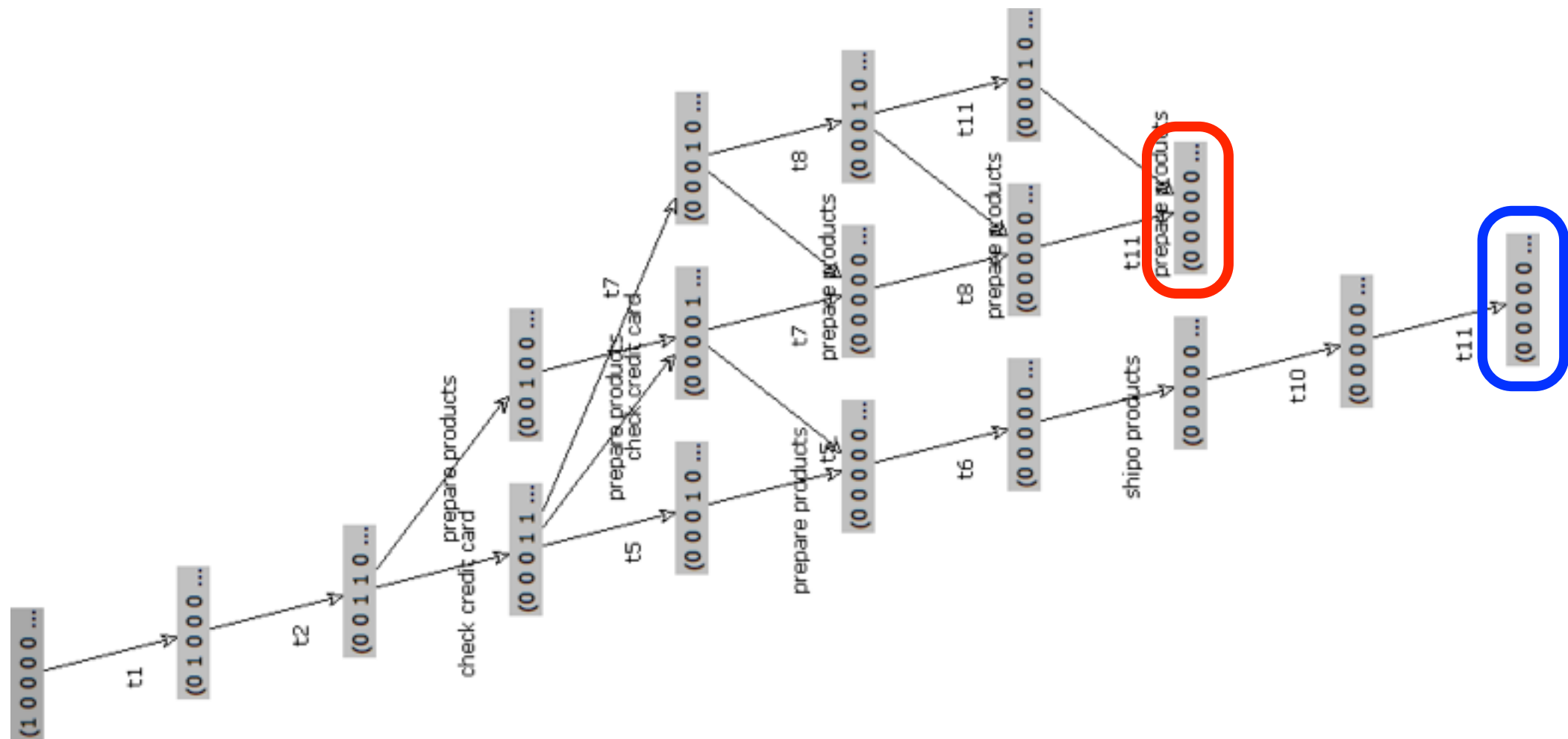
Call subprocess \mathcal{P}



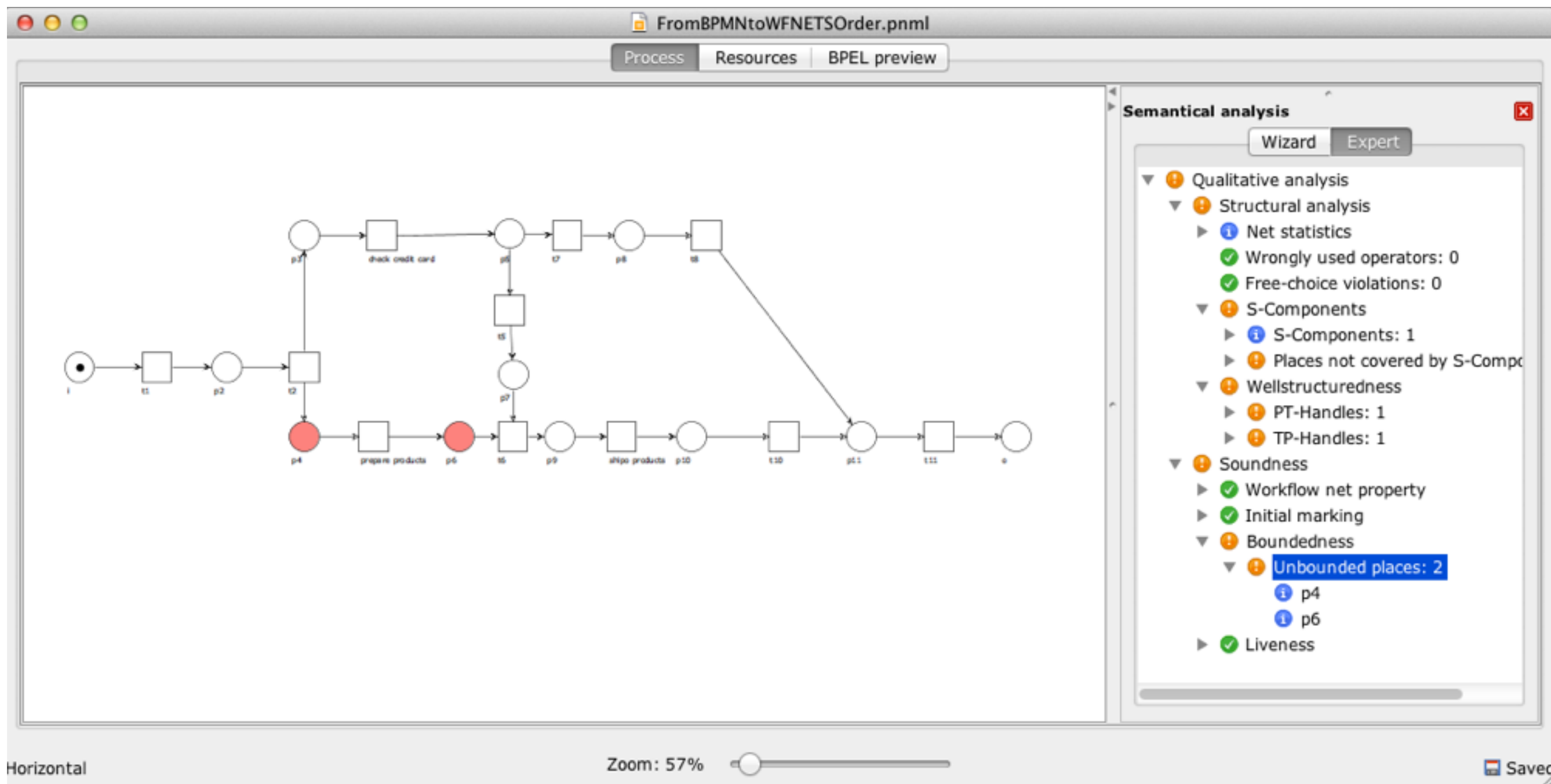
Example: Order process



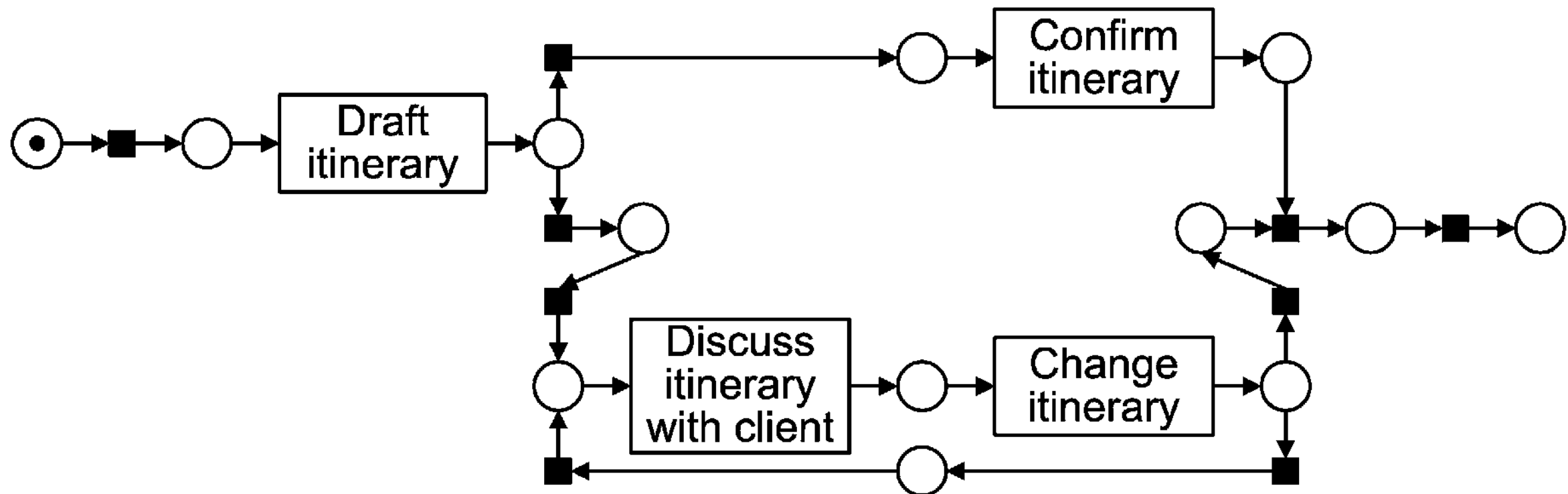
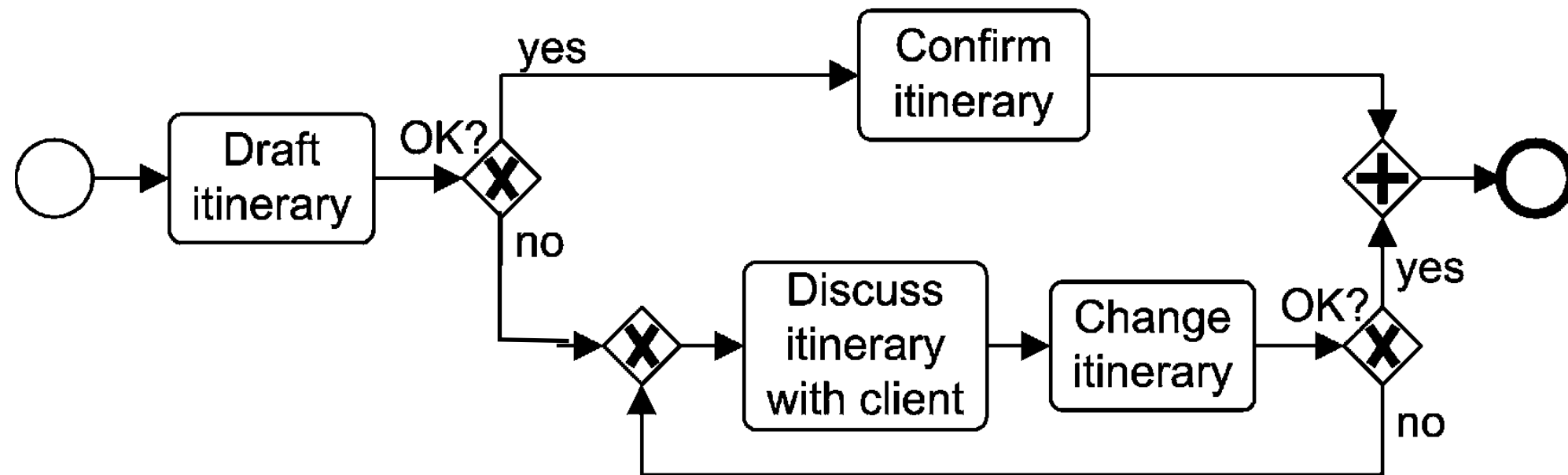
Example: Order process



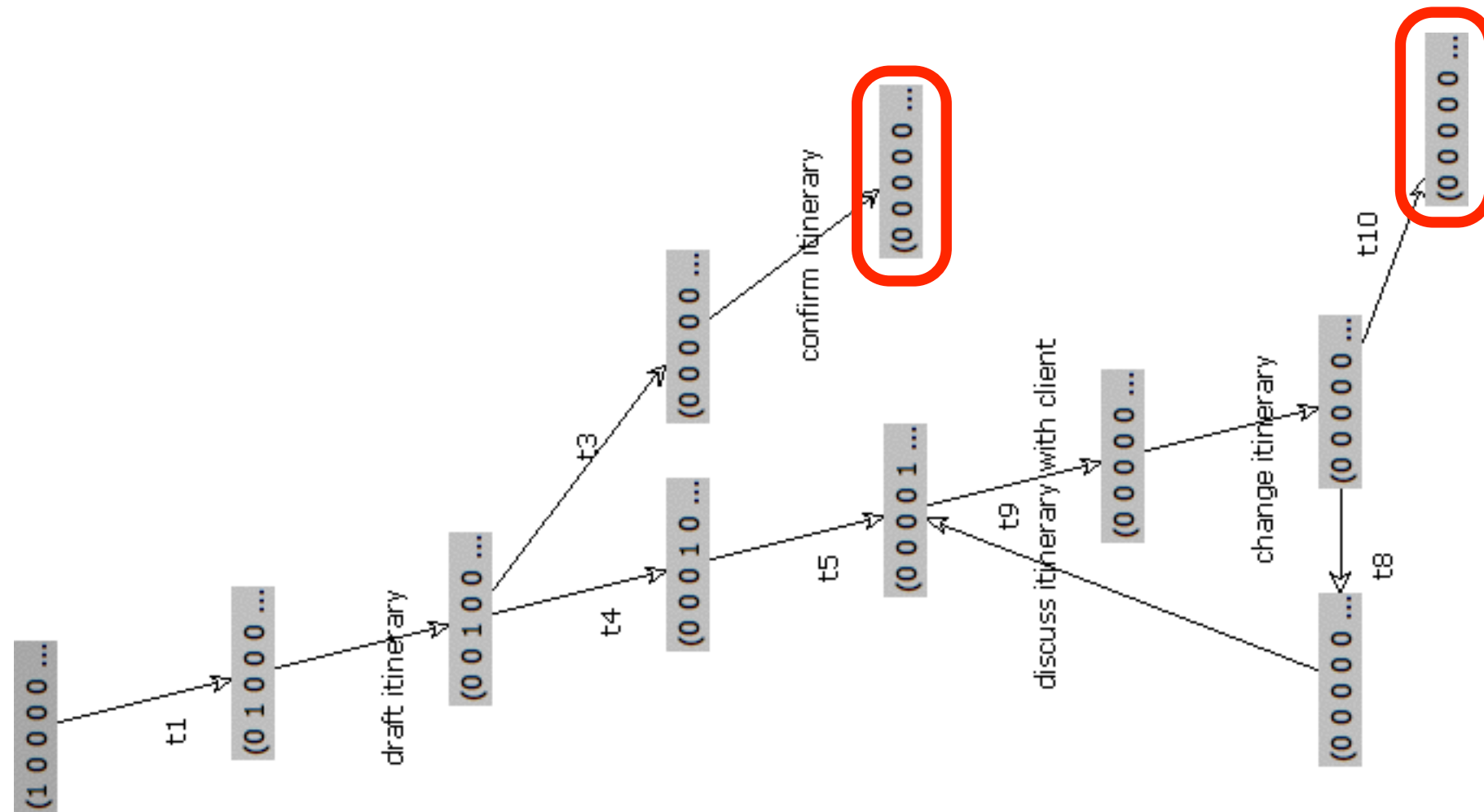
Example: Order process



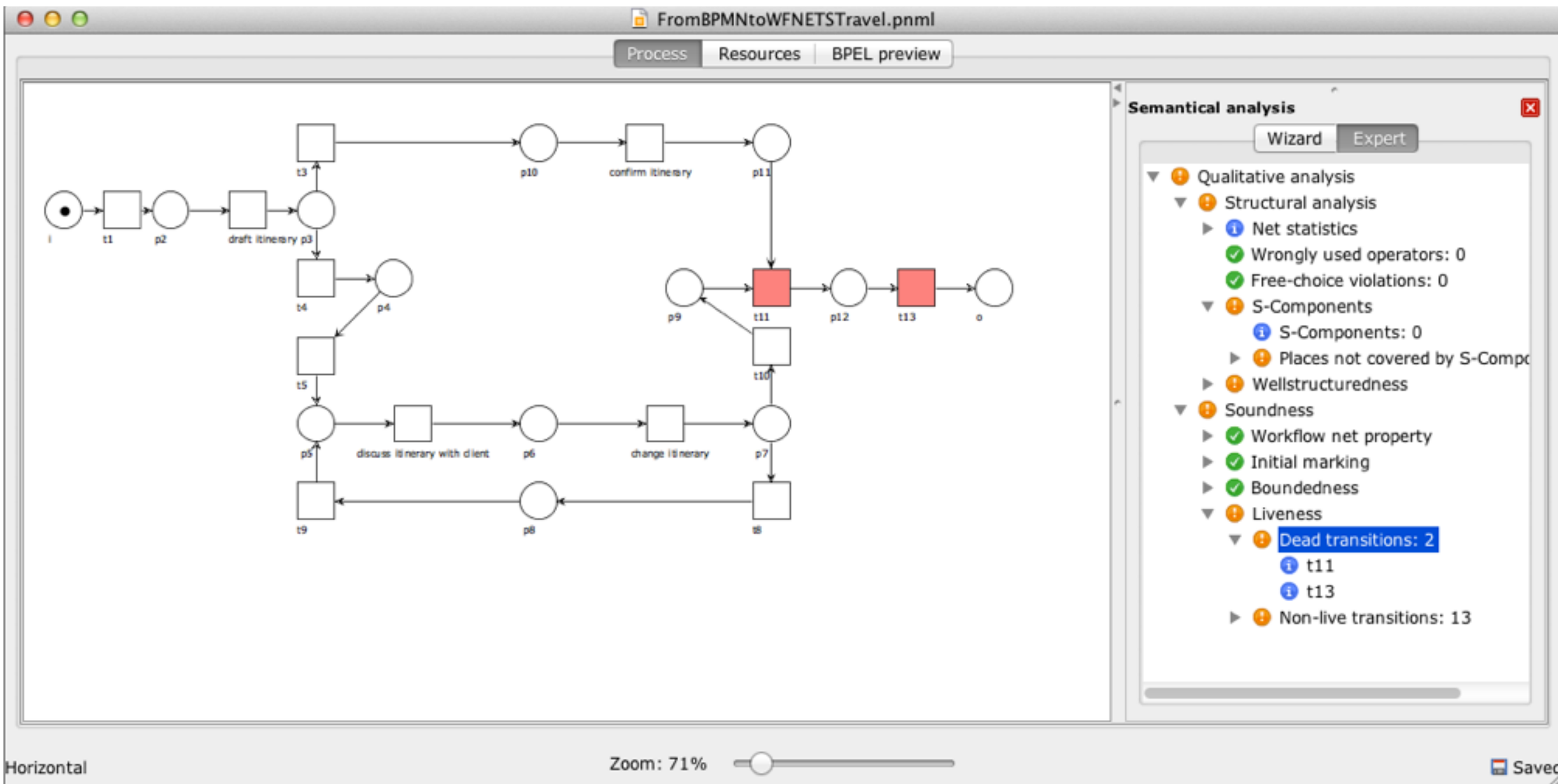
Example: Travel itinerary



Example: Travel itinerary



Example: Travel itinerary



Exercise

Translate the BPMN collaboration diagram to nets and discuss problematic issues

