

# Business Processes Modelling

## MPB (6 cfu, 295AA)

Roberto Bruni

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

07 - Business process  
modelling notation



BPMN basics

BPMN key features

**More on BPMN**

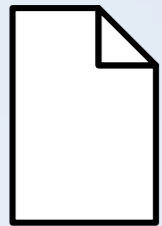
BPMN semantics

More artefacts  
(data-objects, groups)

# Data object

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

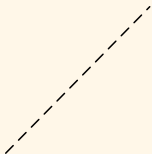
A data object is often represented by the usual file icon



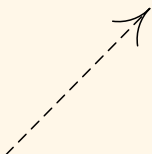
[state]

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.

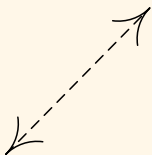
# Association, again



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



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



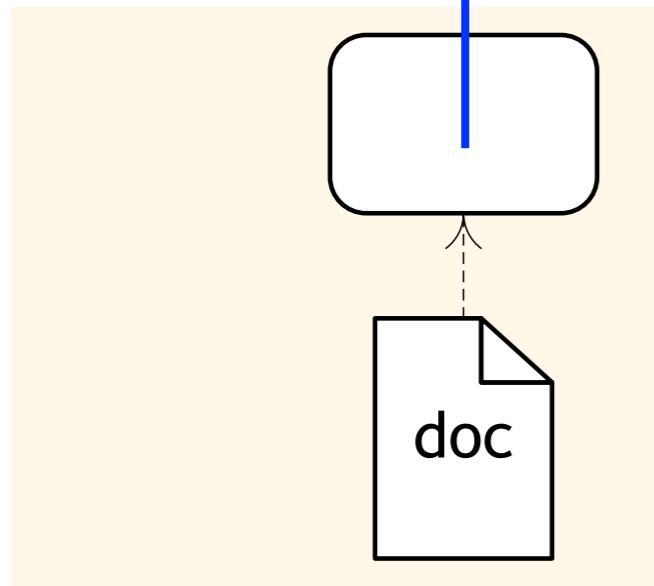
A **Bidirected Association** indicates that the data object is modified, i.e. read and written during the execution of an activity.

# Question time

read

write

modify

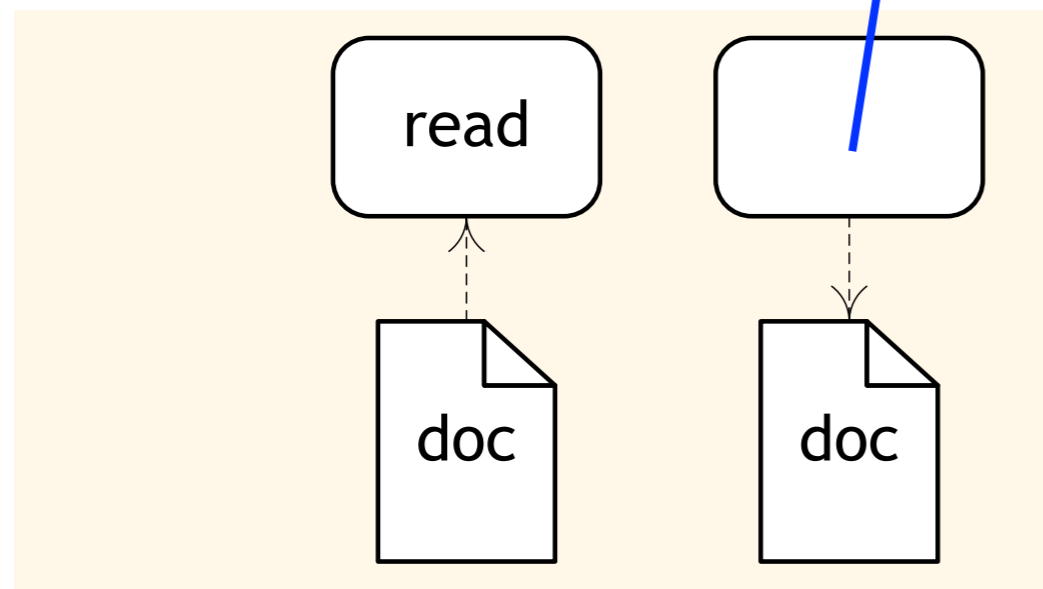


# Question time

read

write

modify

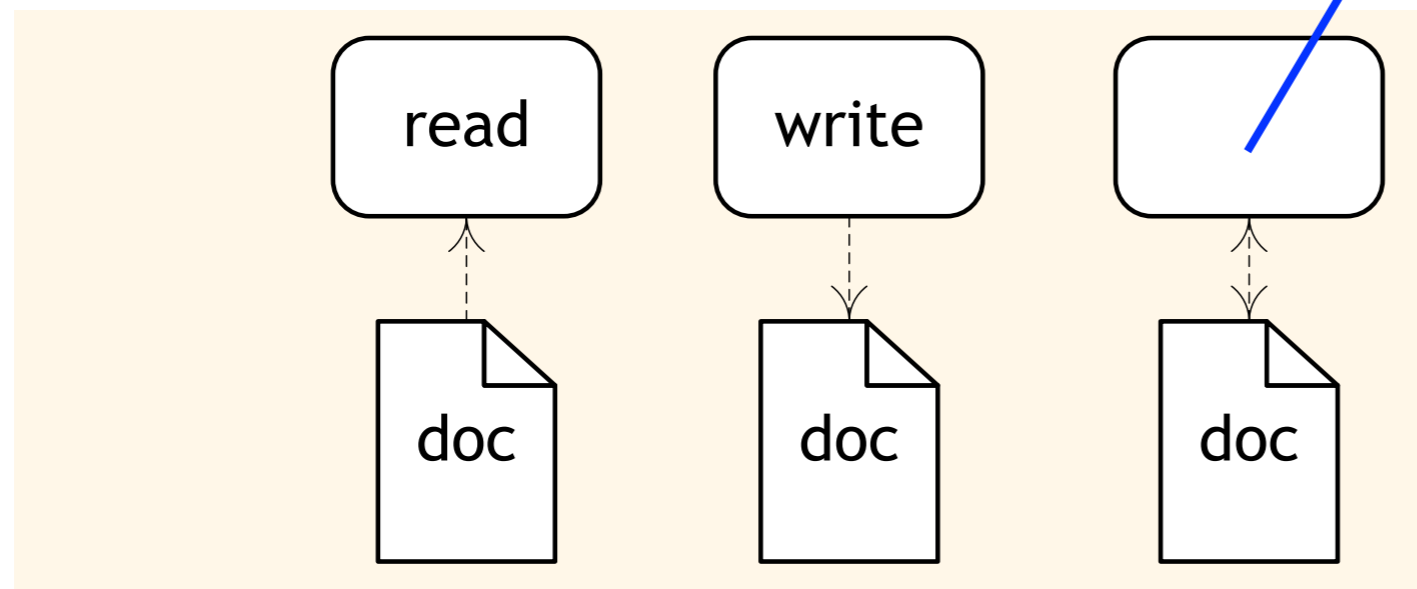


# Question time

read

write

modify



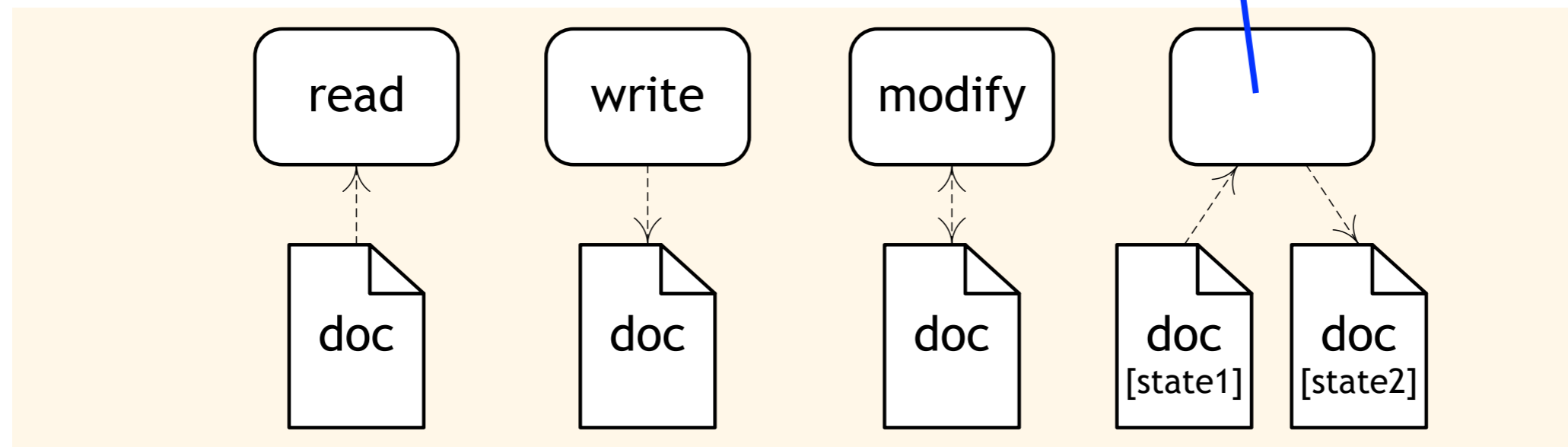


# Question time

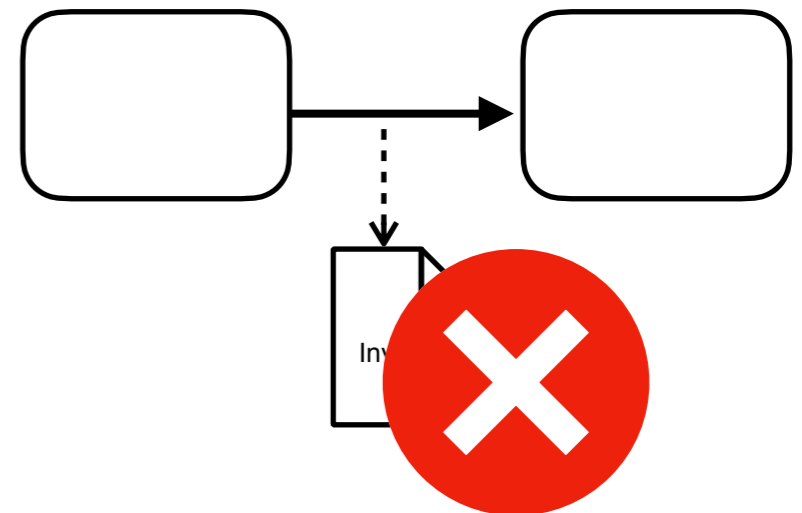
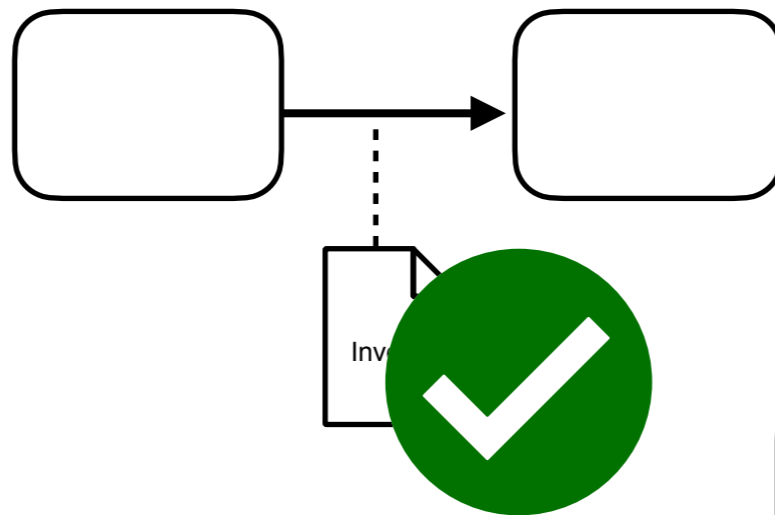
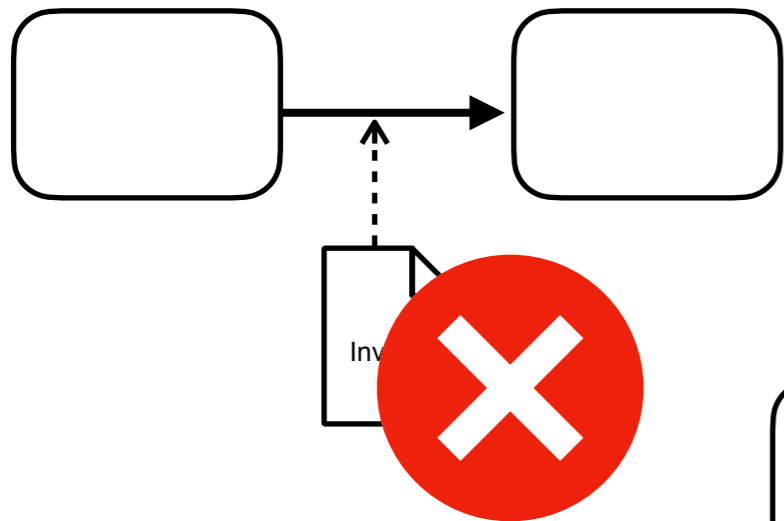
read

write

modify



# Question time

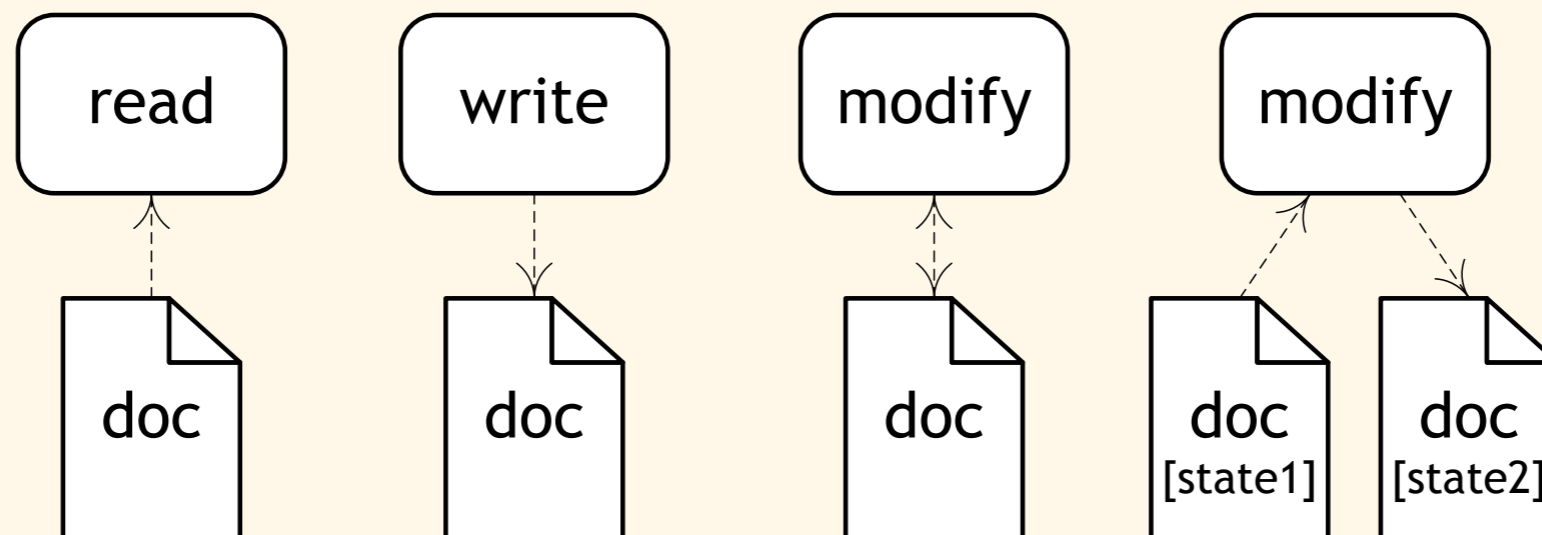


# Association, again

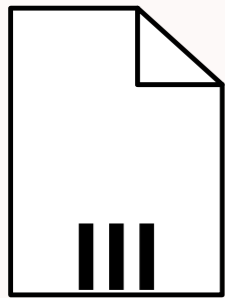
Attaching a data object with an **Undirected Association** to a sequence flow indicates hand-over of information between the activities involved.

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

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



# More data objects



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.

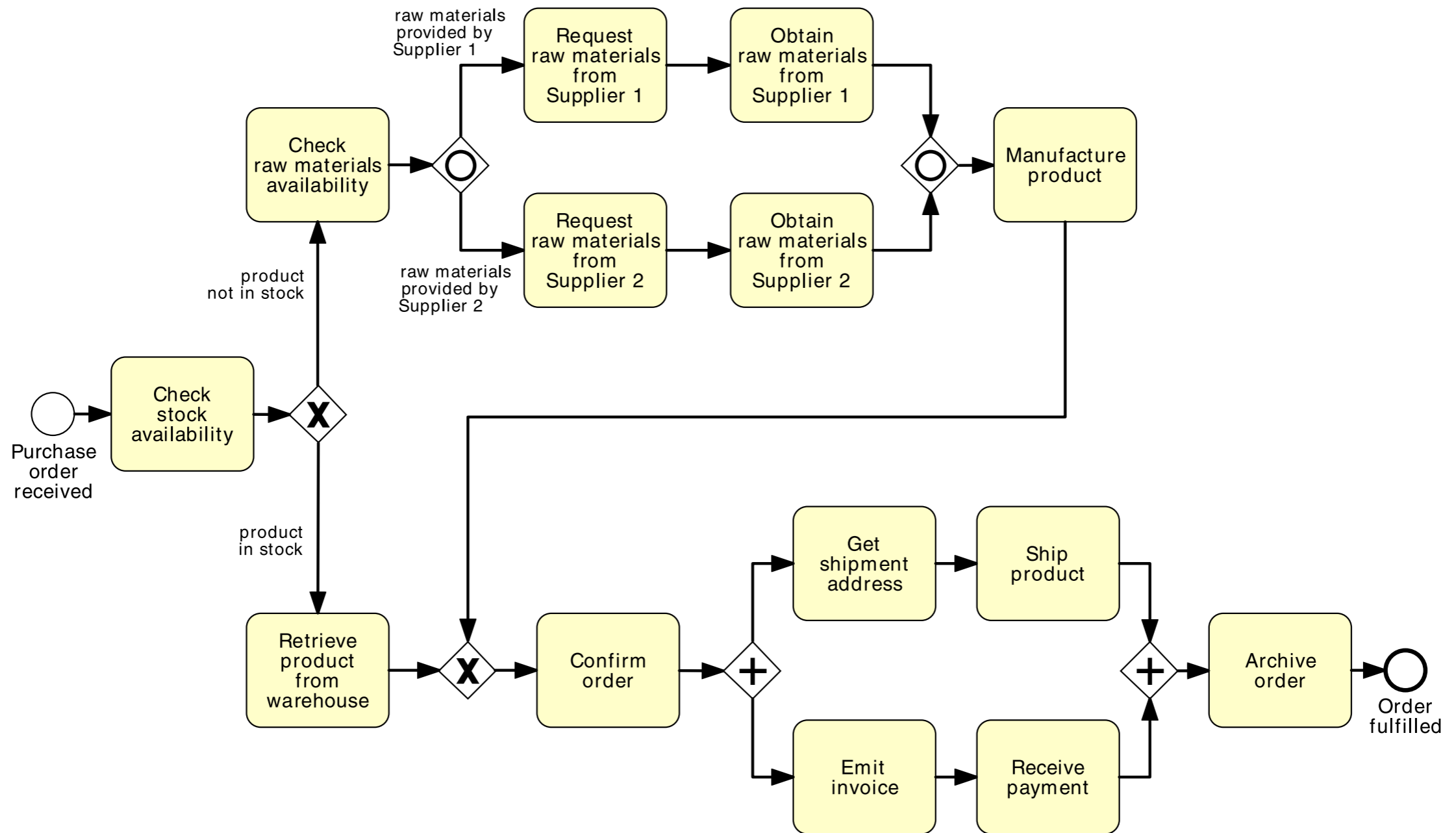
# Group

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

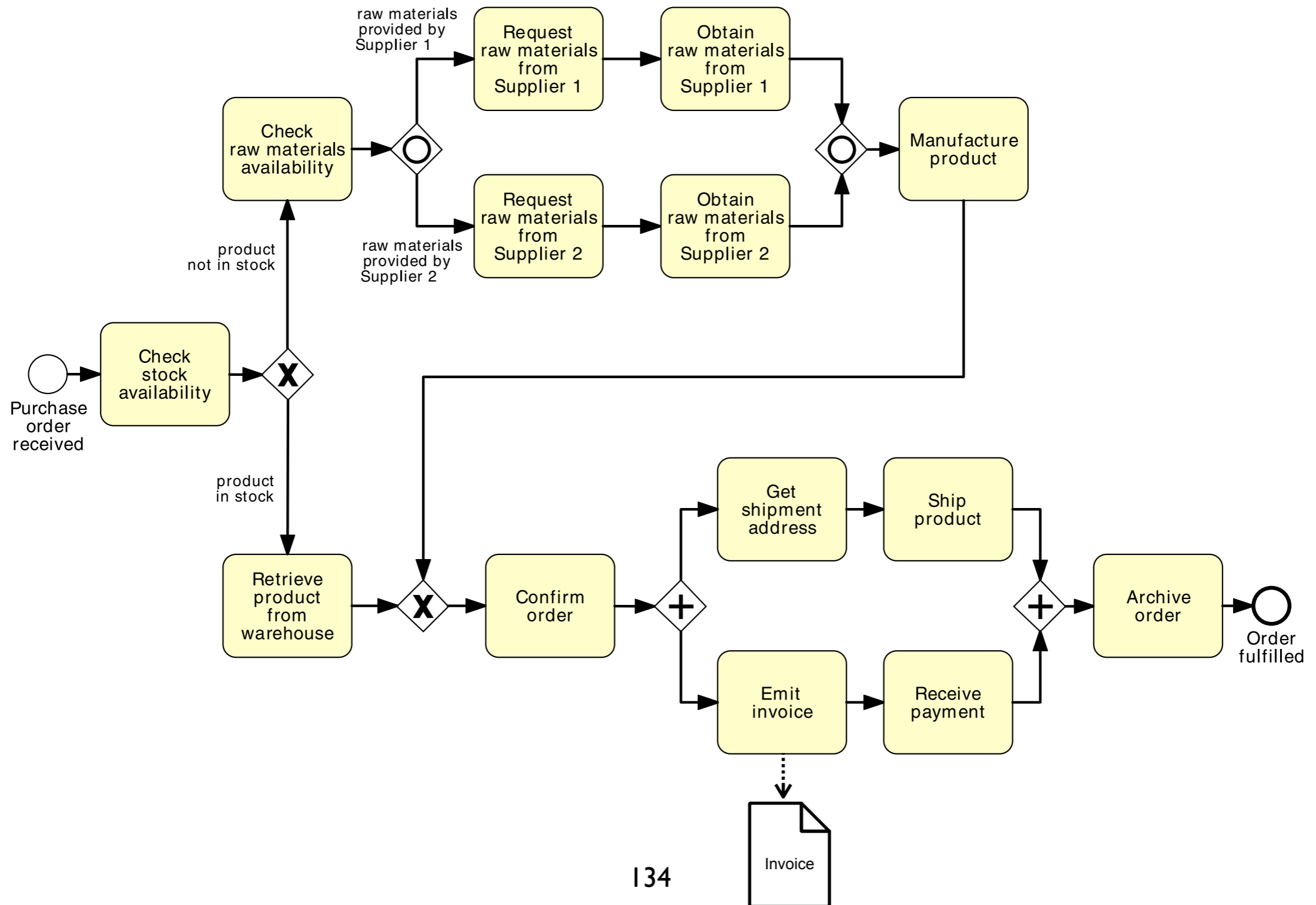
A group is represented by  
rounded corner rectangles with dashed lines



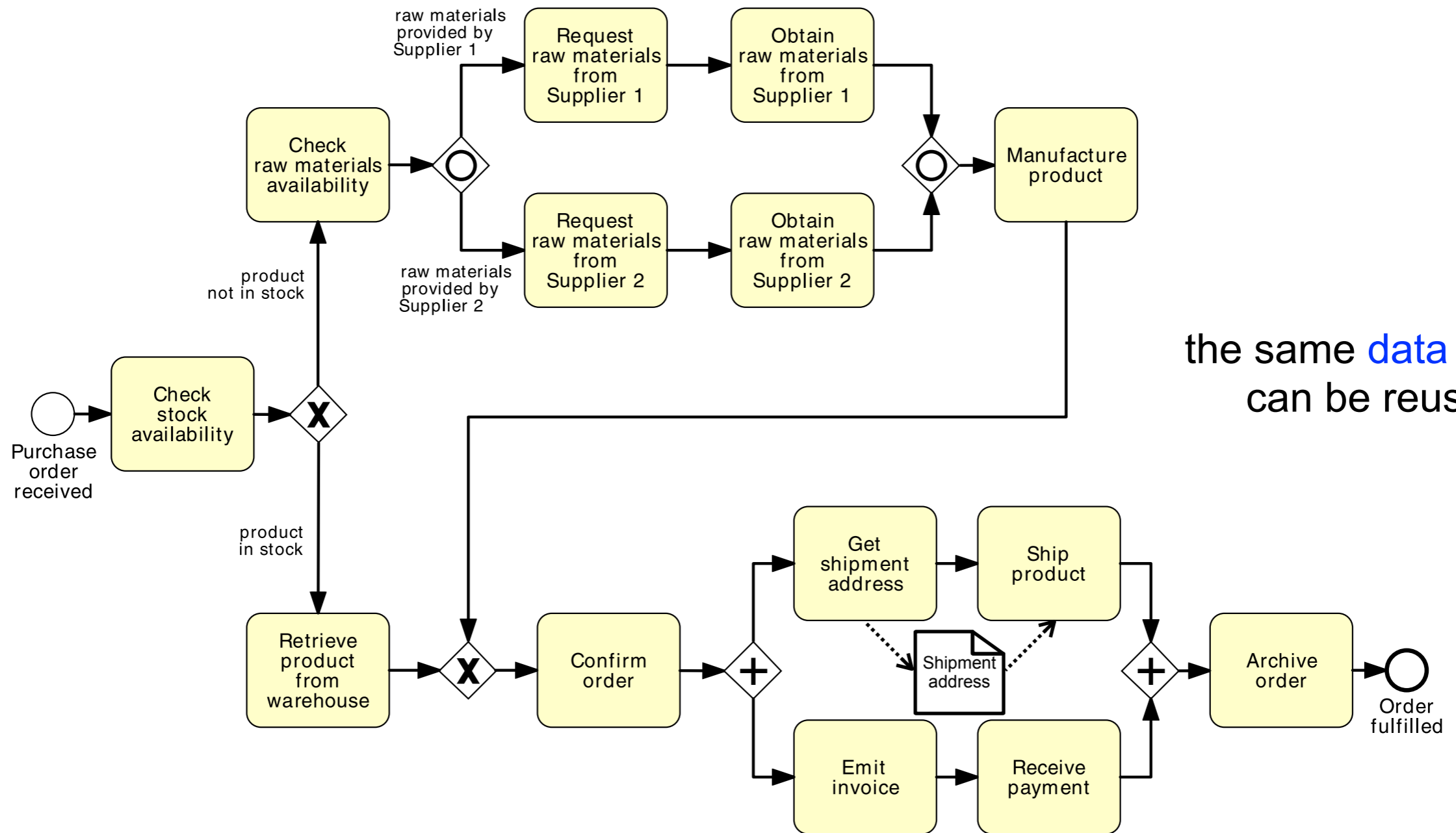
# Example: data objects



# Example: data objects

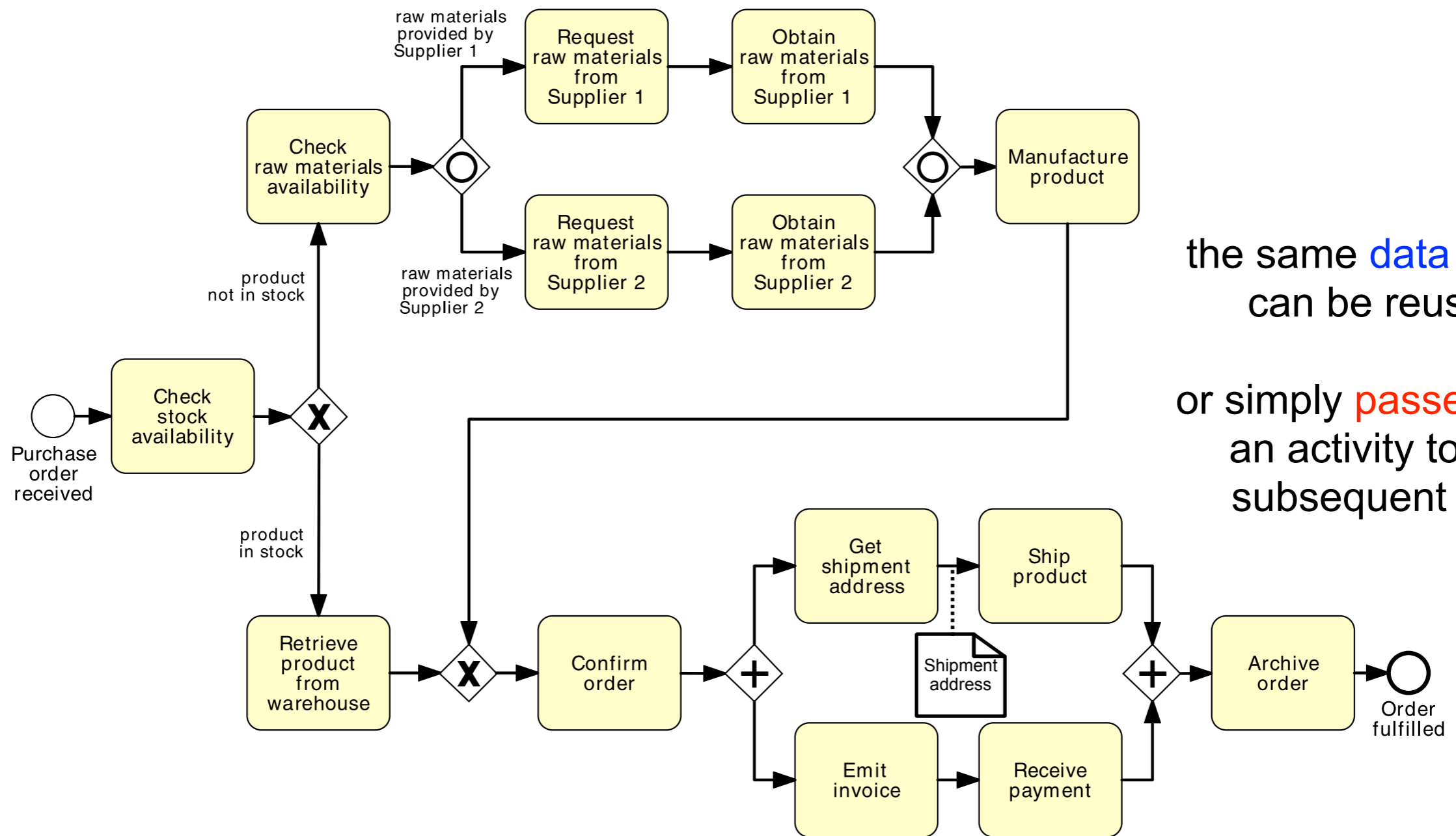


# Example: data objects





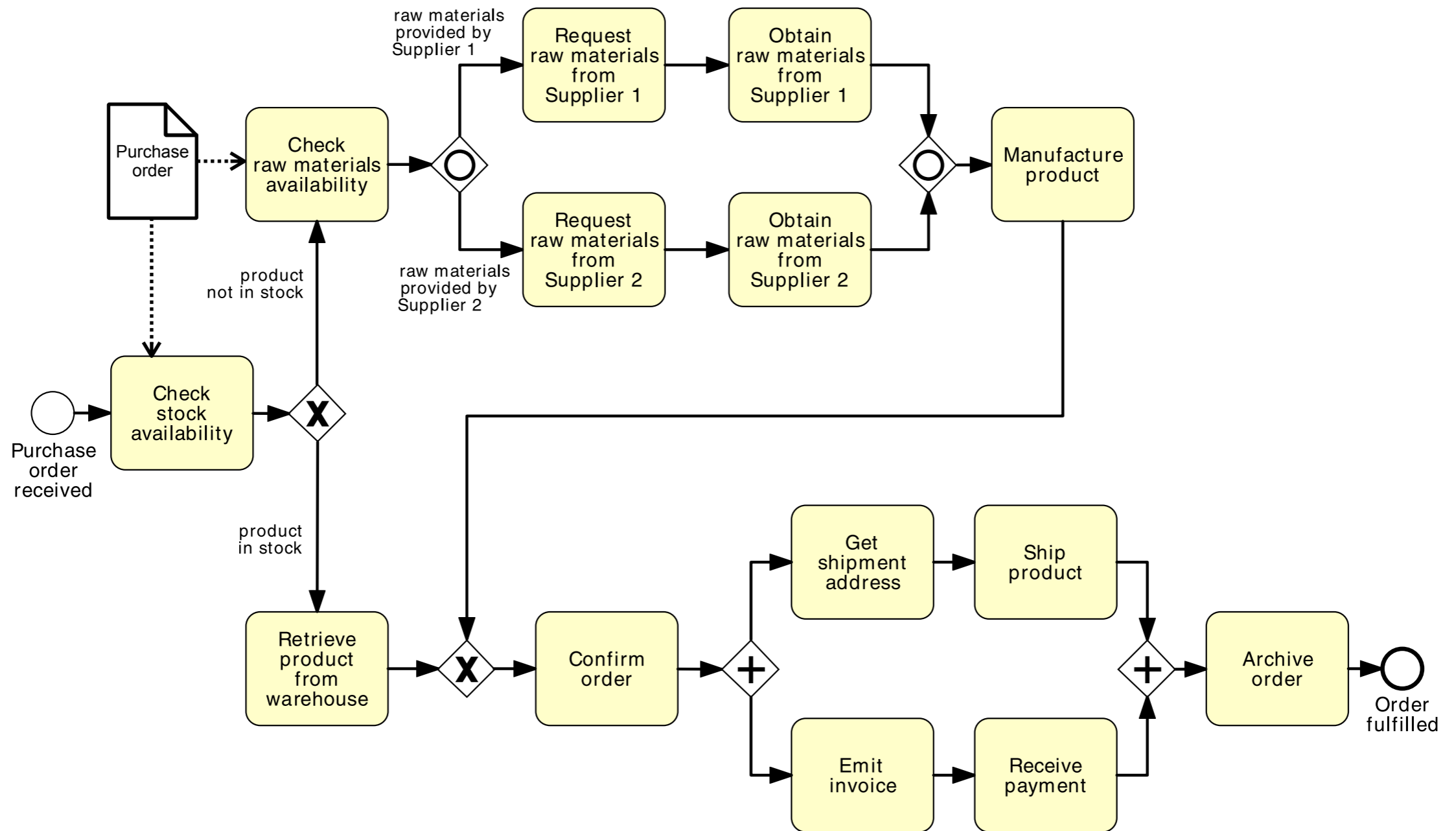
# Example: data objects



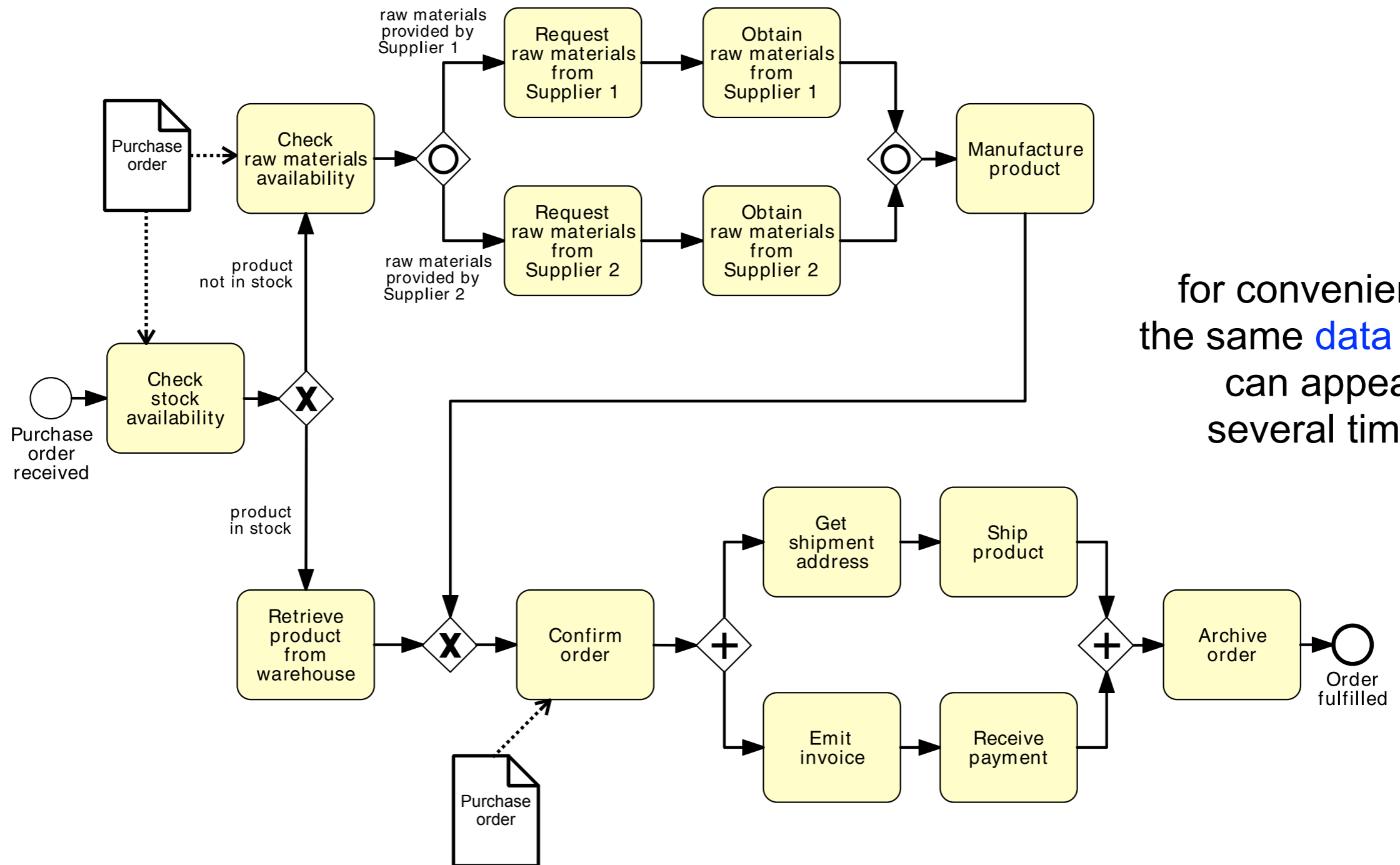
the same **data object** can be reused

or simply **passed** from an activity to the subsequent one

# Example: data objects

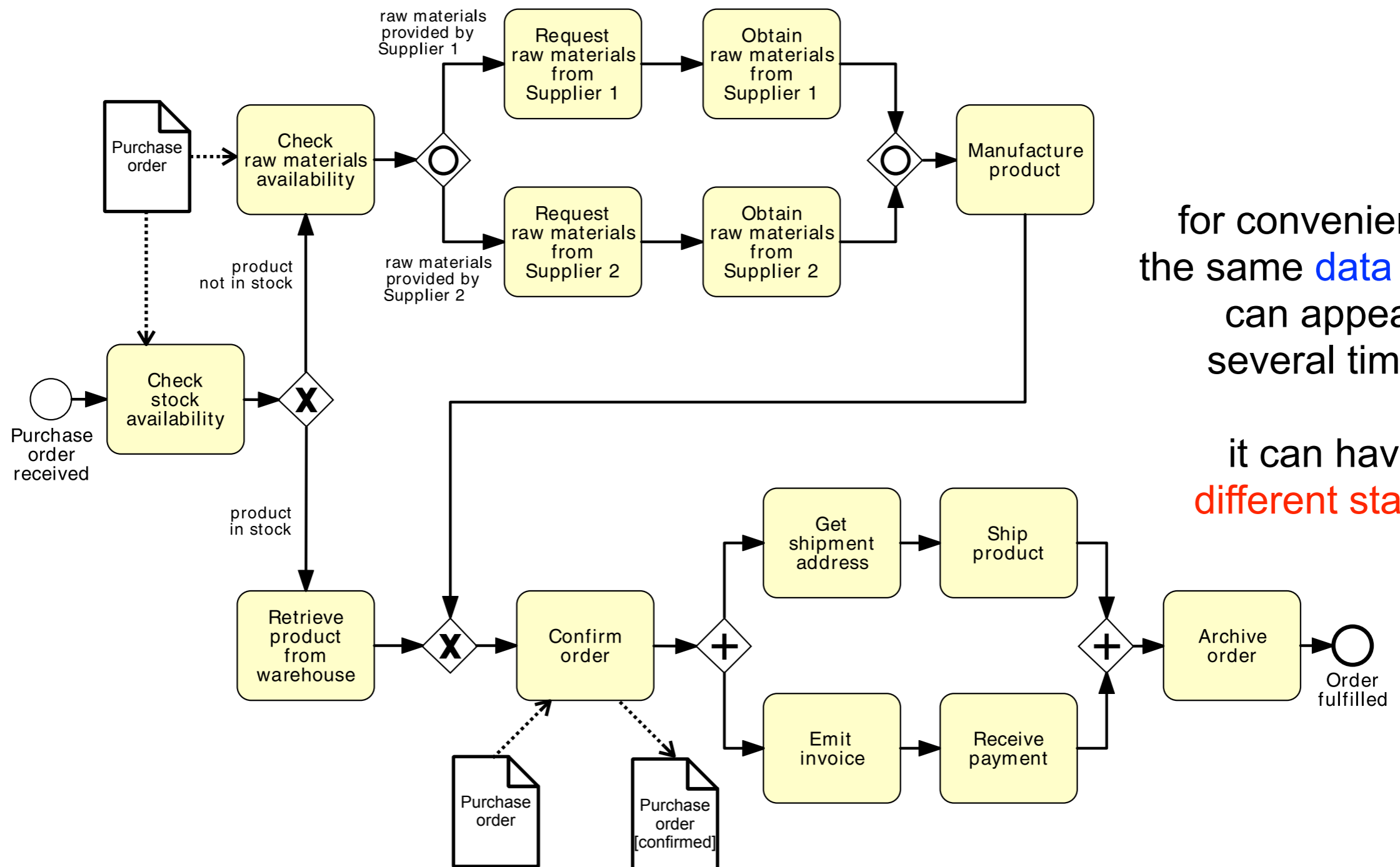


# Example: data objects



for convenience,  
the same **data object**  
can appear  
several times

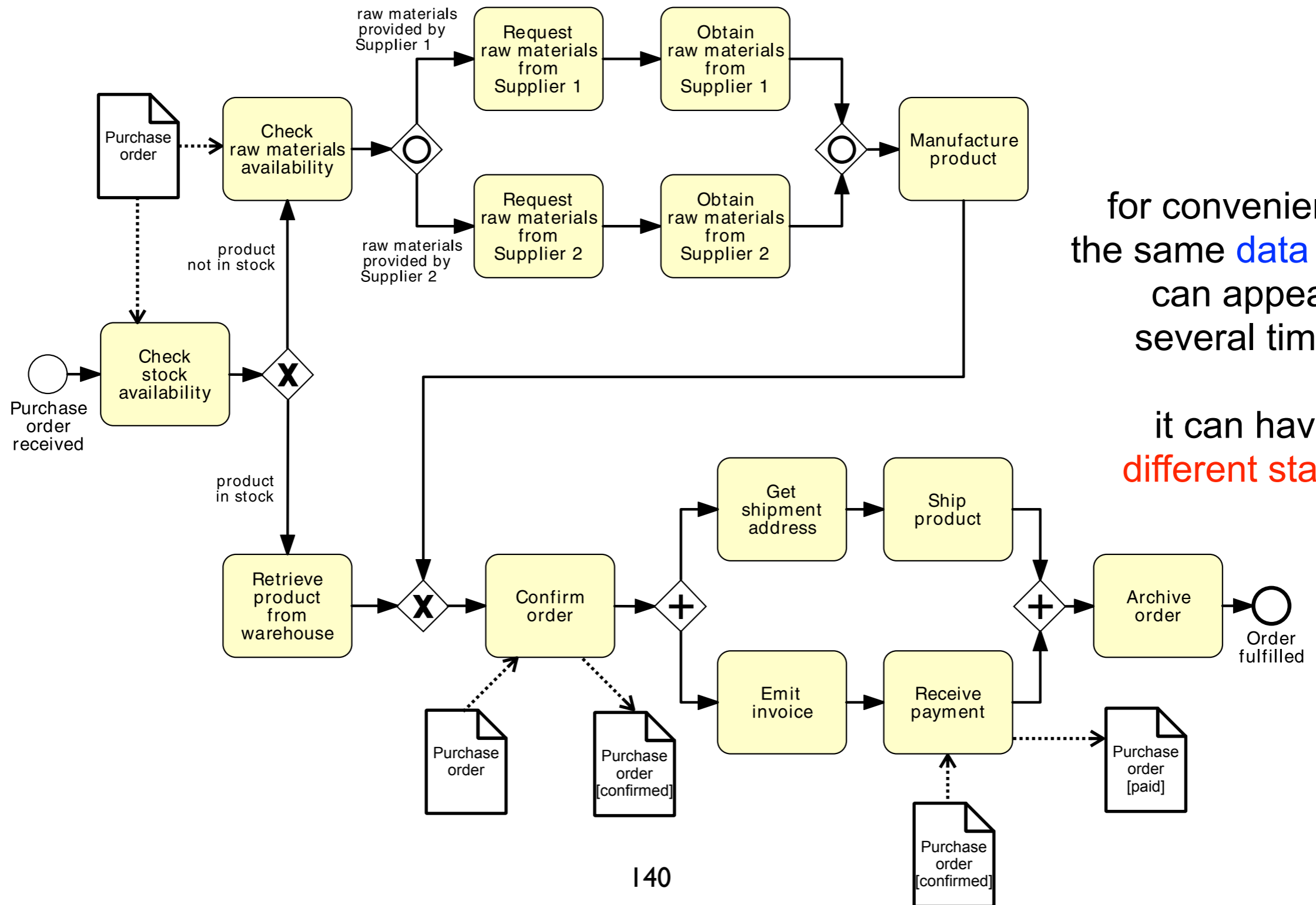
# Example: data objects



for convenience,  
the same **data object**  
can appear  
several times

it can have  
**different states**

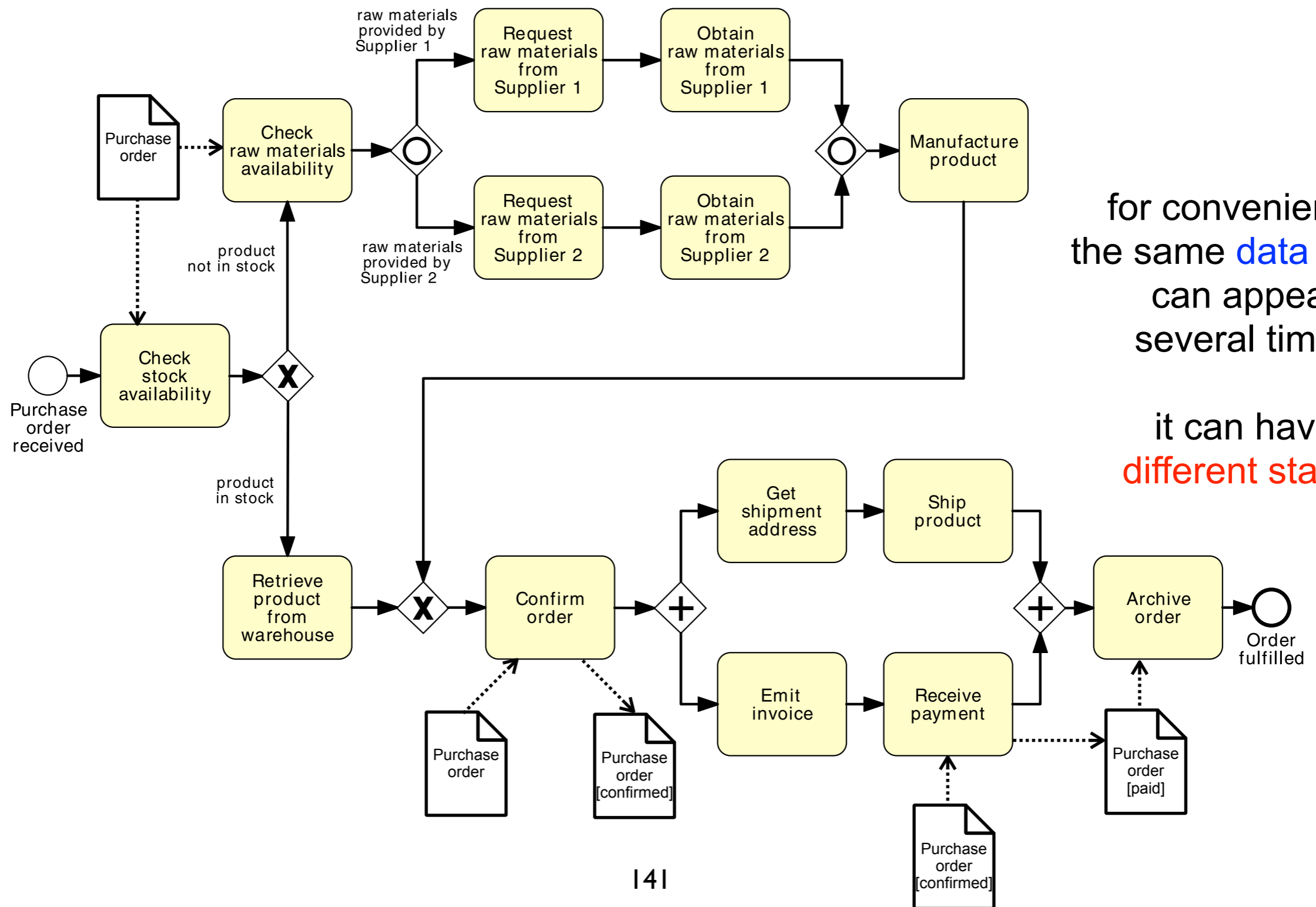
# Example: data objects



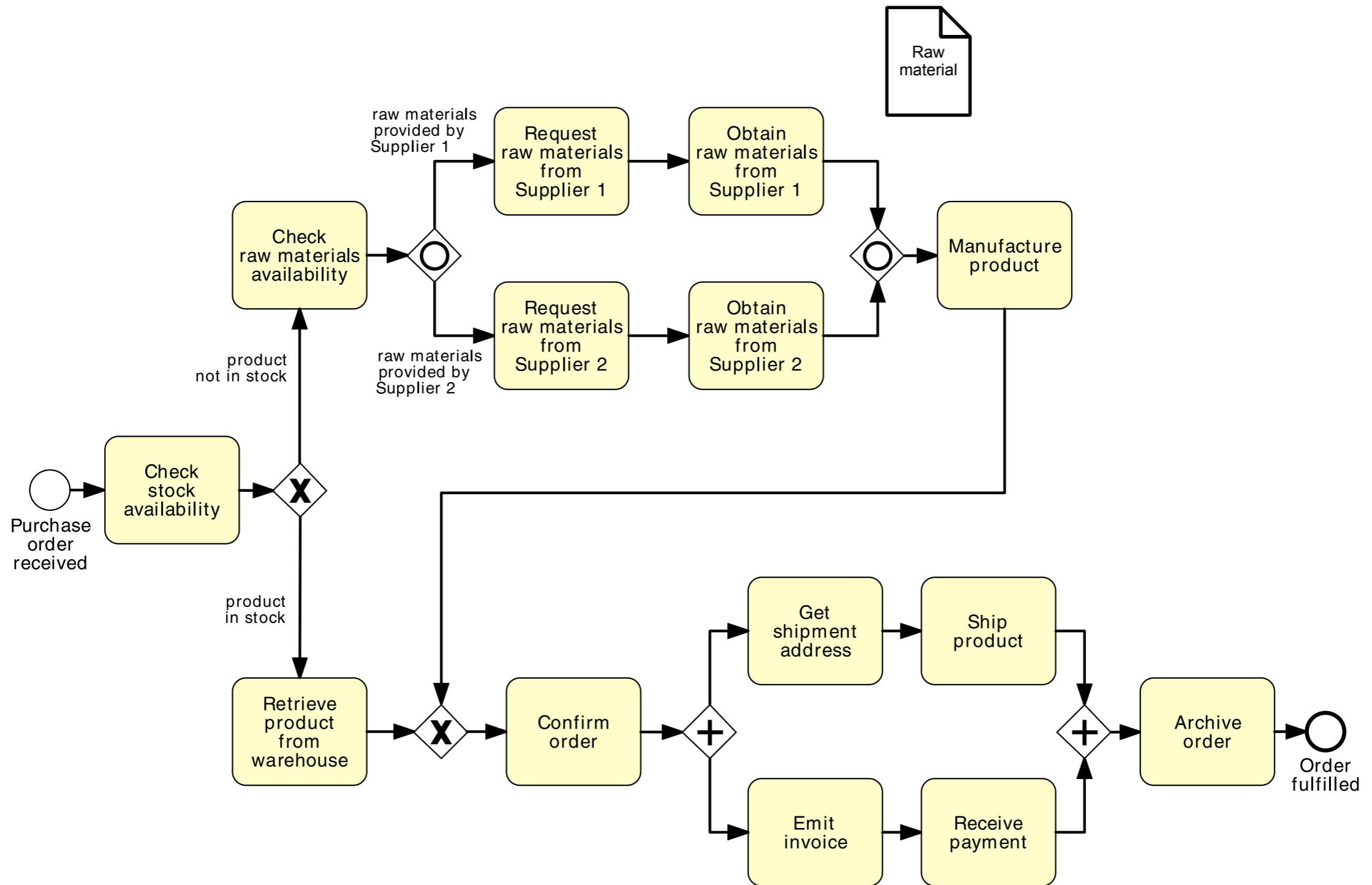
for convenience,  
the same **data object**  
can appear  
several times

it can have  
**different states**

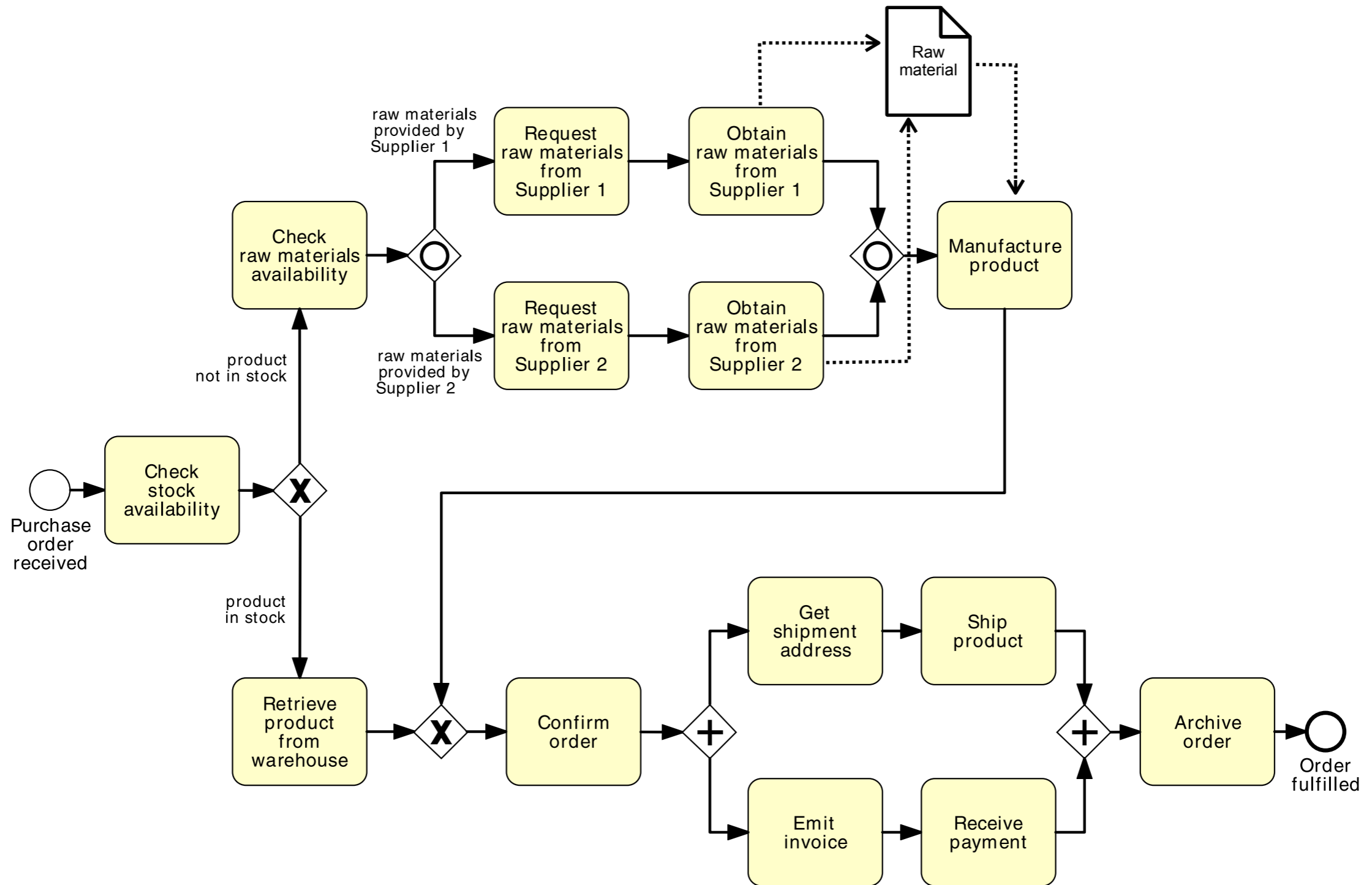
# Example: data objects



# Question time: connections?

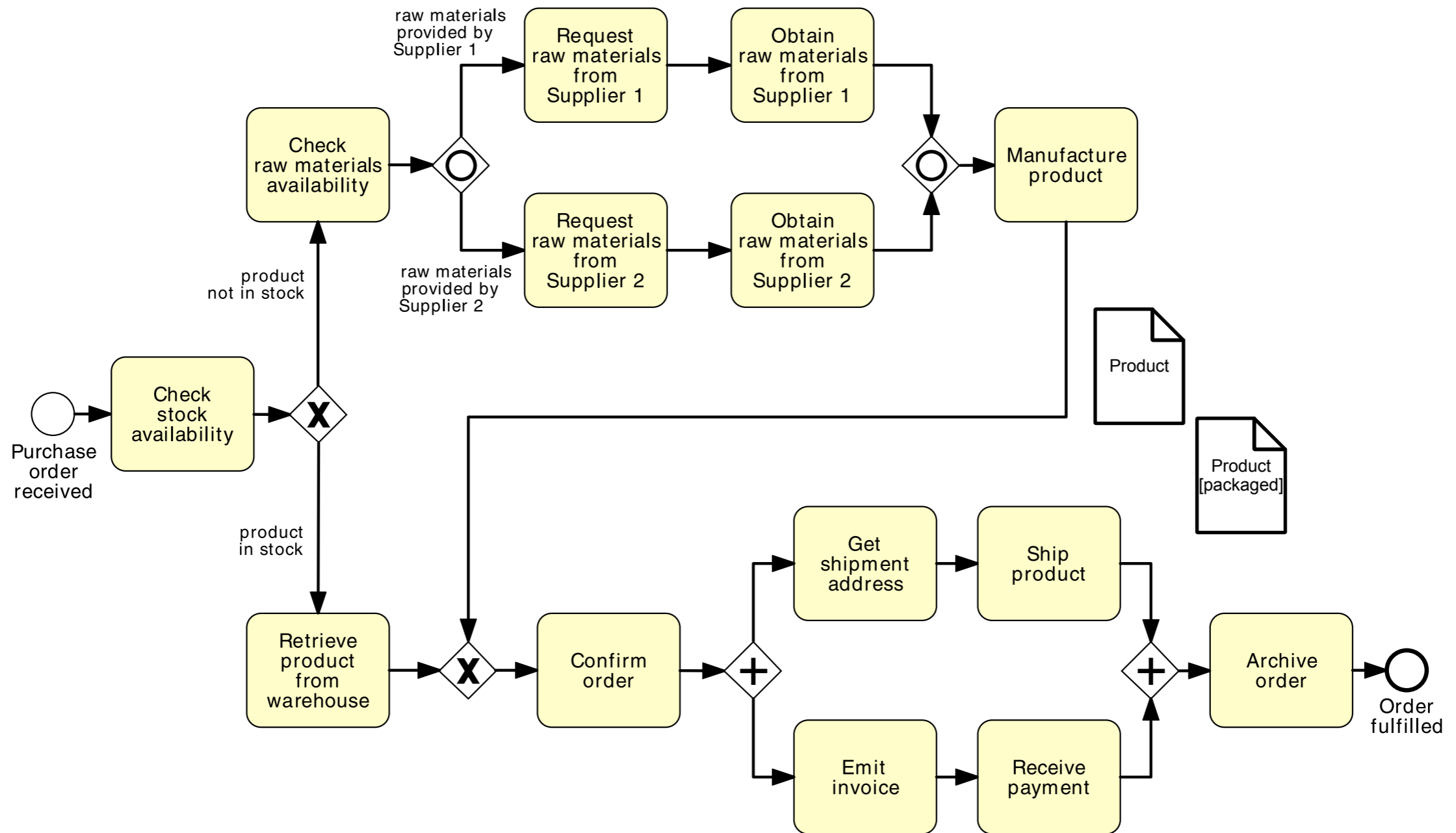


# Question time: connections?

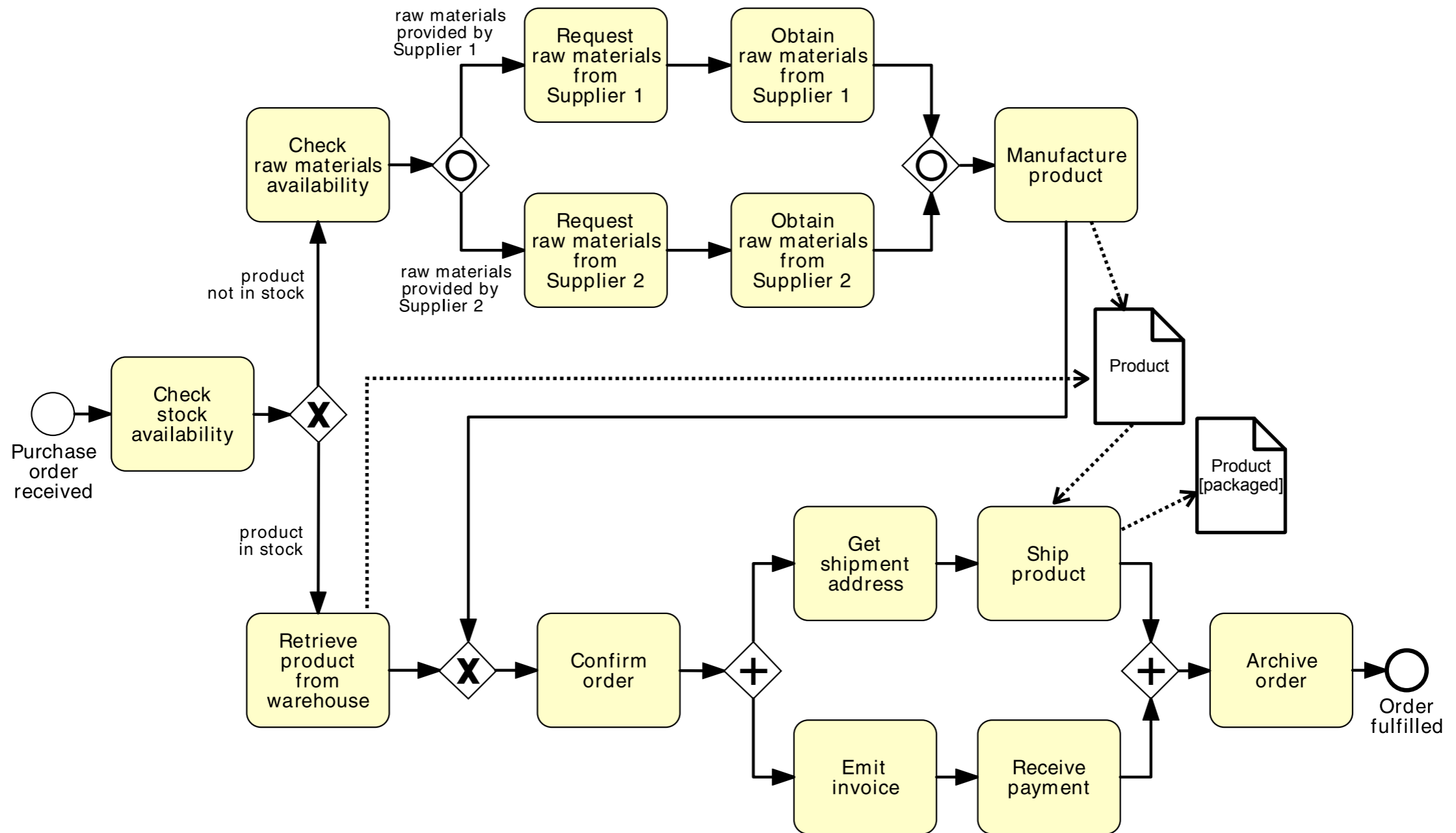




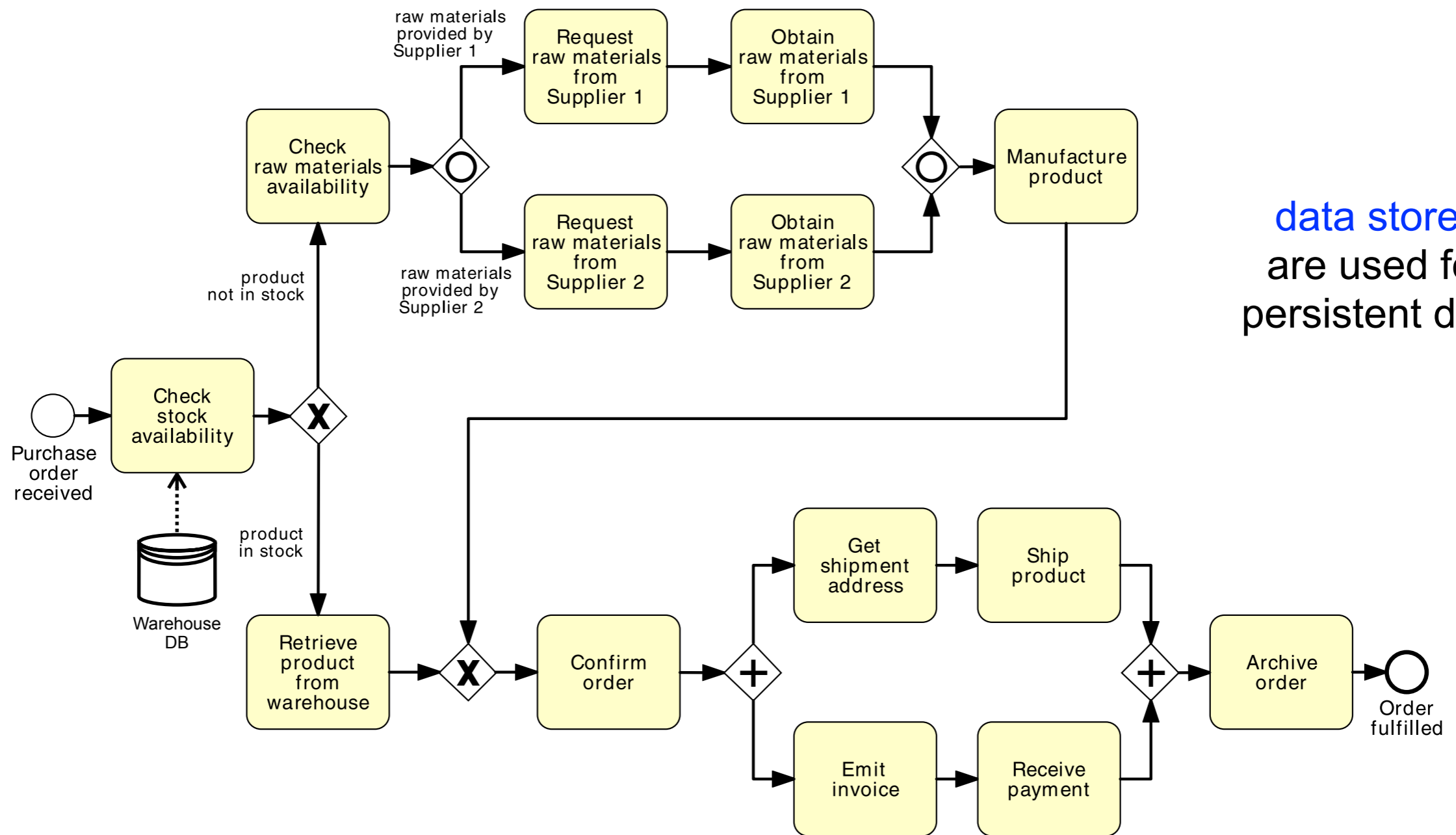
# Question time: connections?



# Question time: connections?

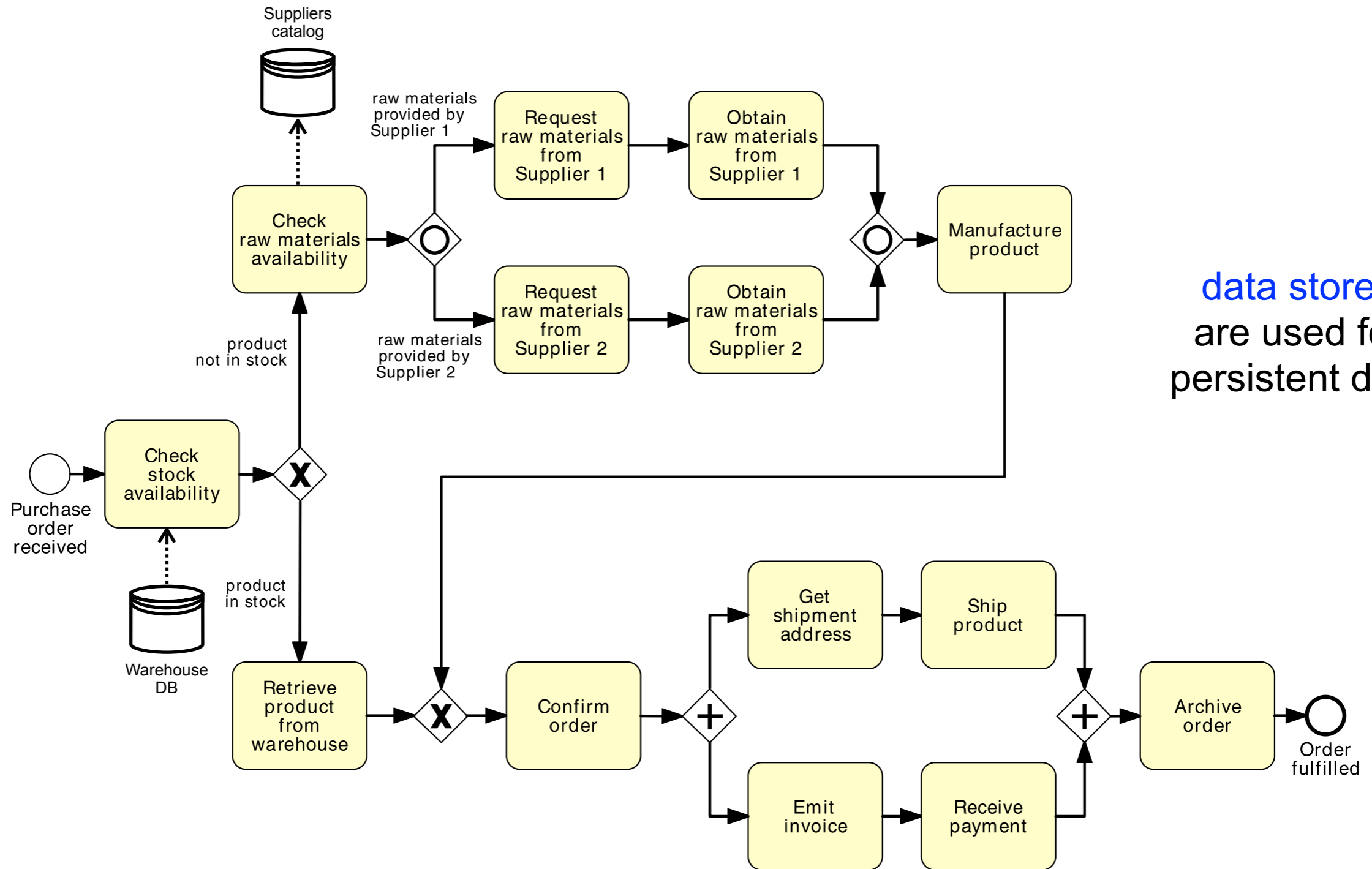


# Example: data stores



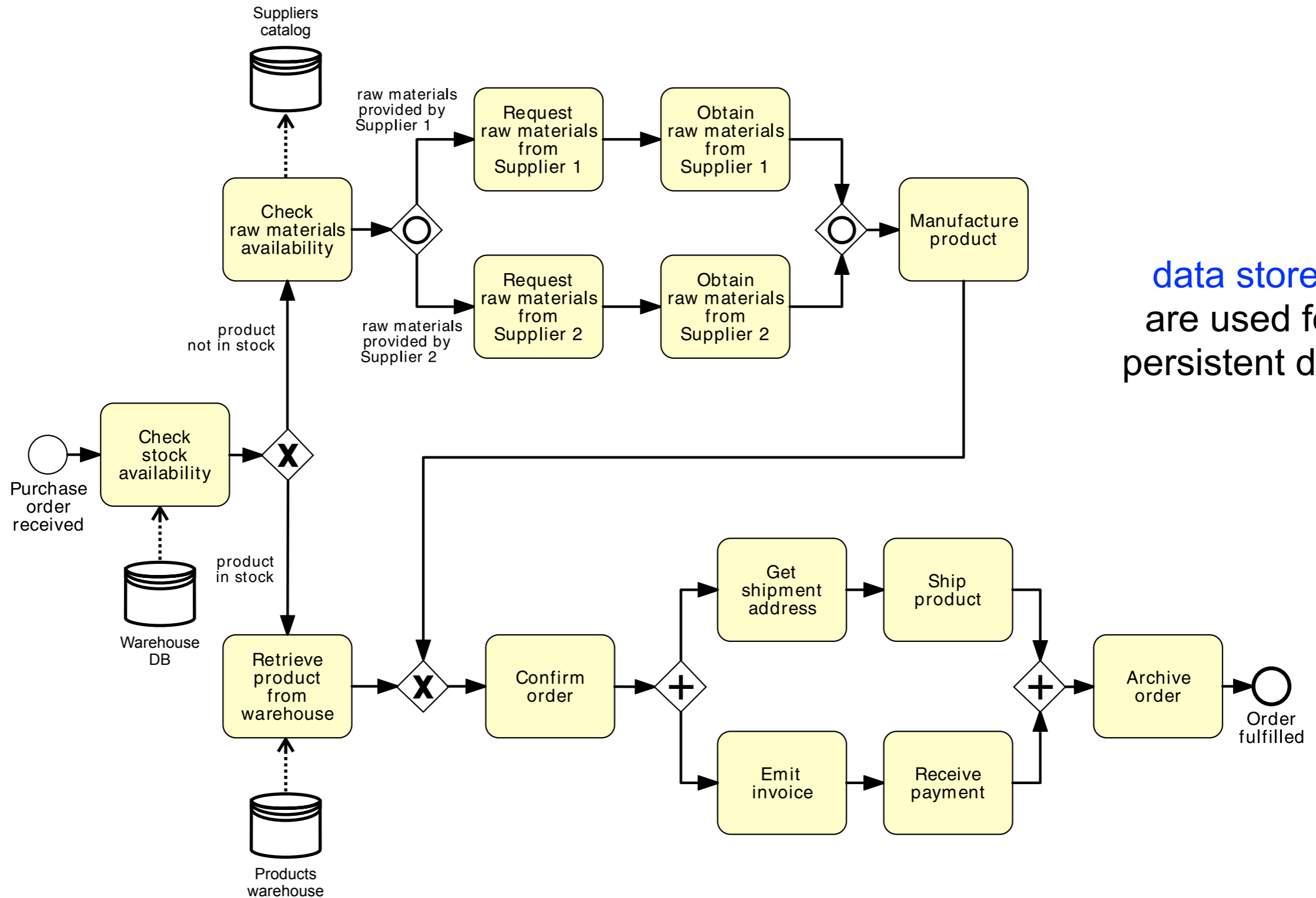
data stores  
are used for  
persistent data

# Example: data stores



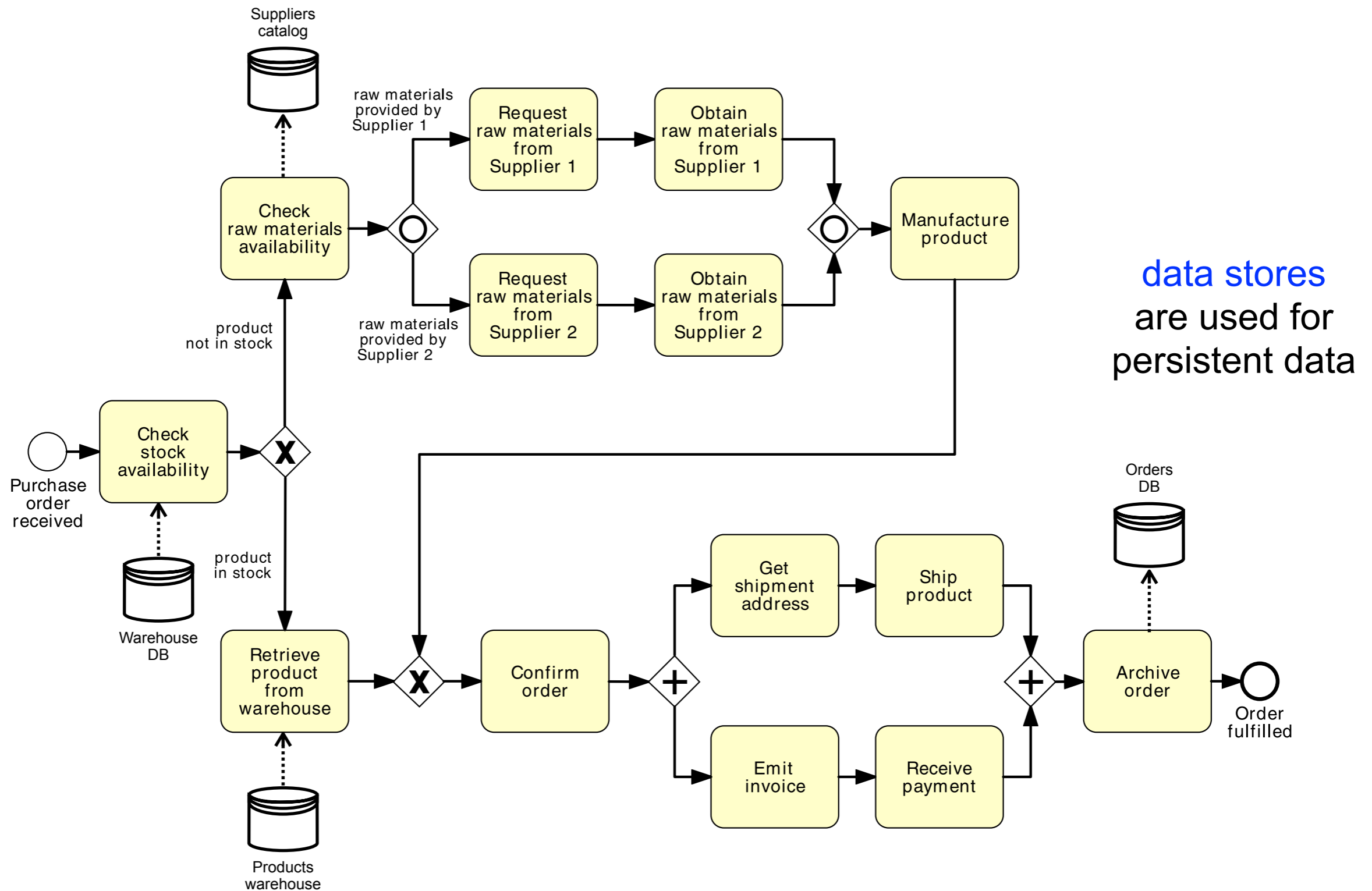
data stores  
are used for  
persistent data

# Example: data stores

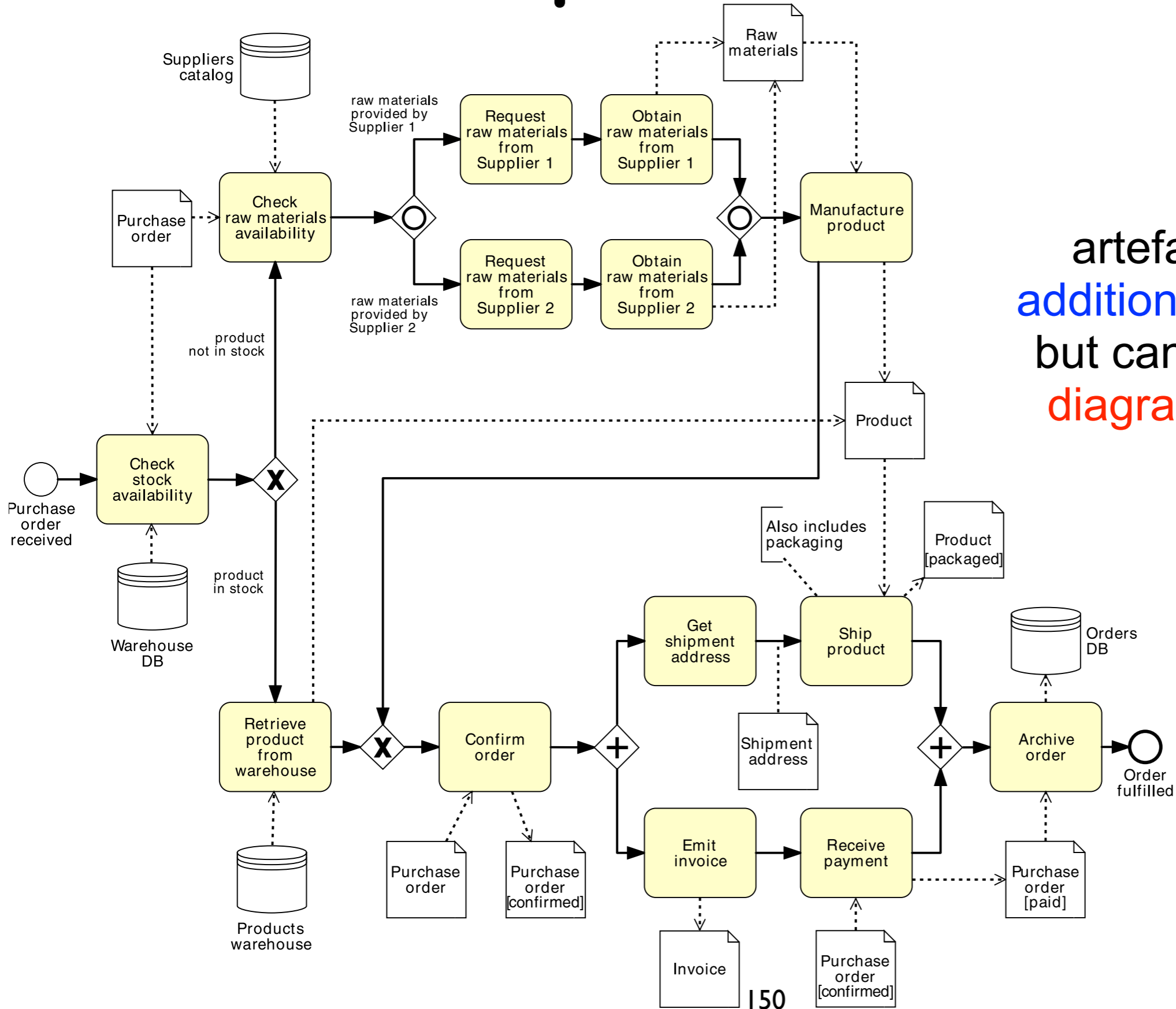


data stores  
are used for  
persistent data

# Example: data stores

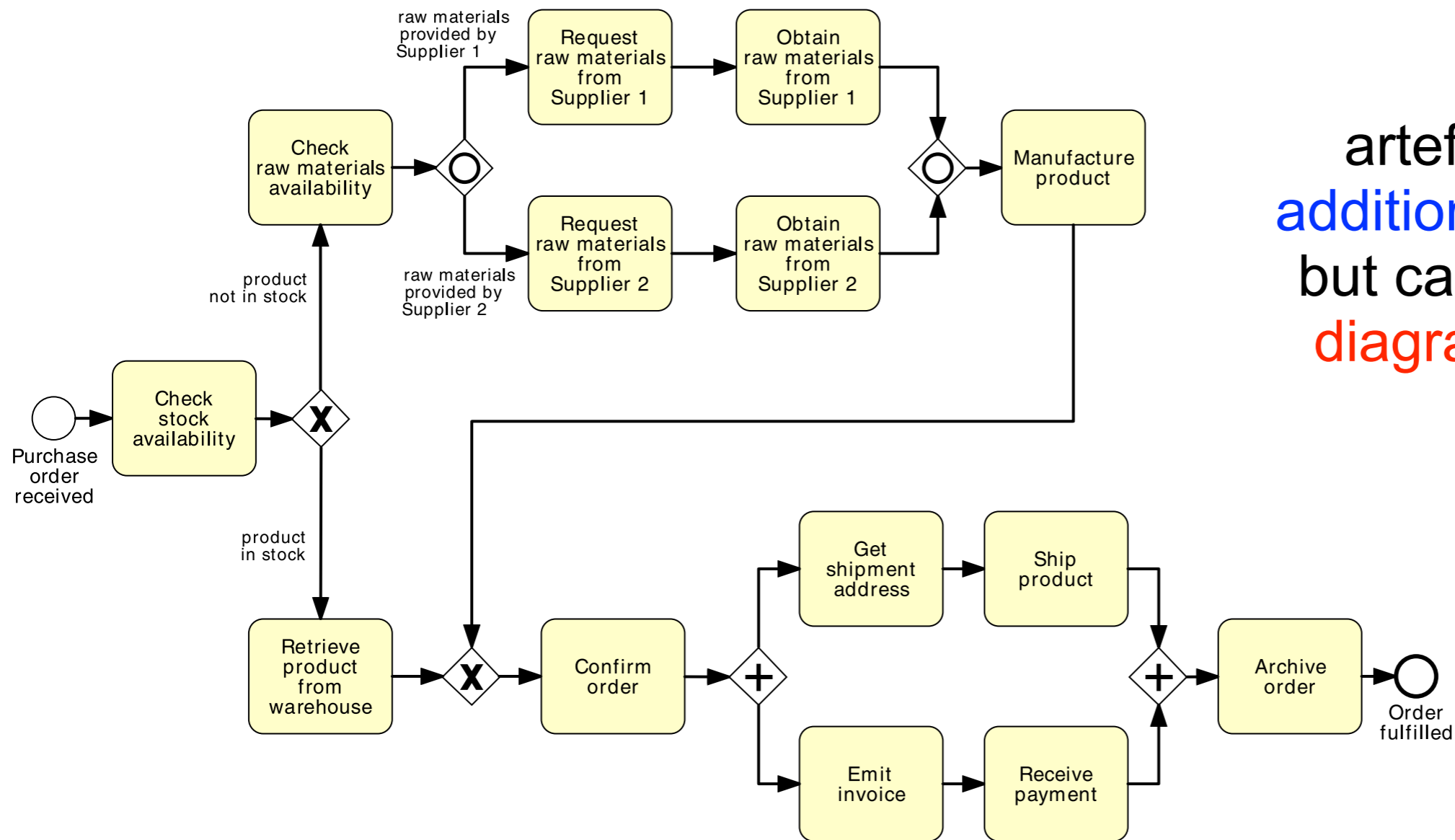


# Example: artefacts



artefacts provide additional information, but can compromise diagram readability

# Example: artefacts

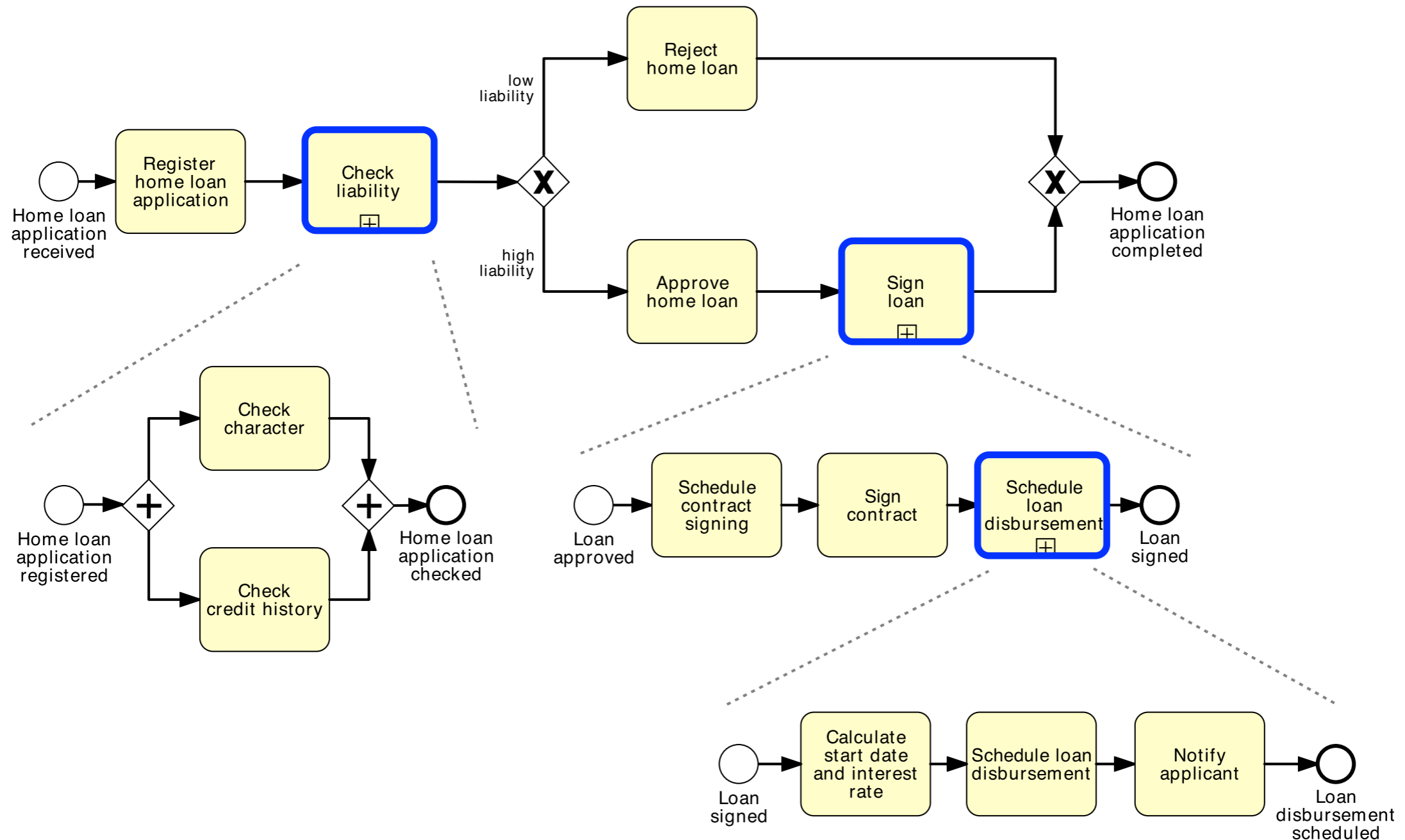


artefacts provide  
additional information,  
but can compromise  
diagram readability

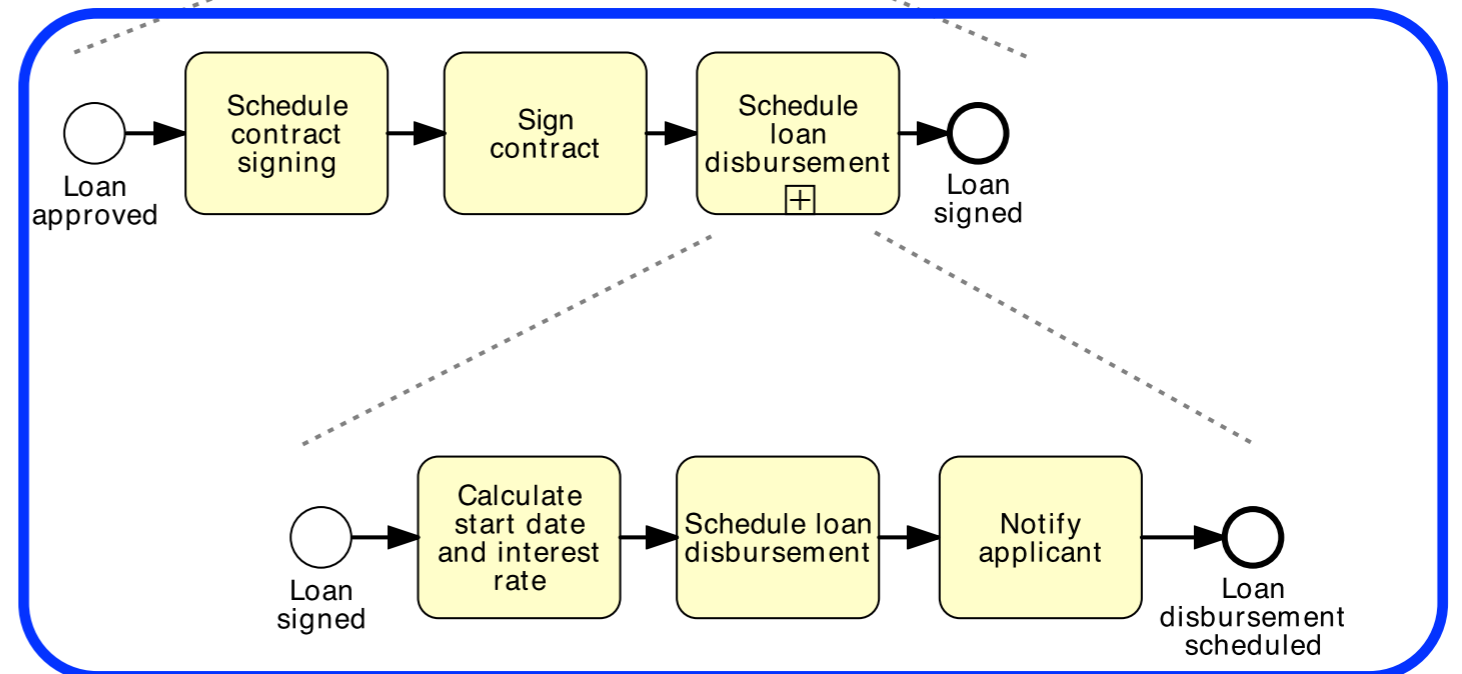
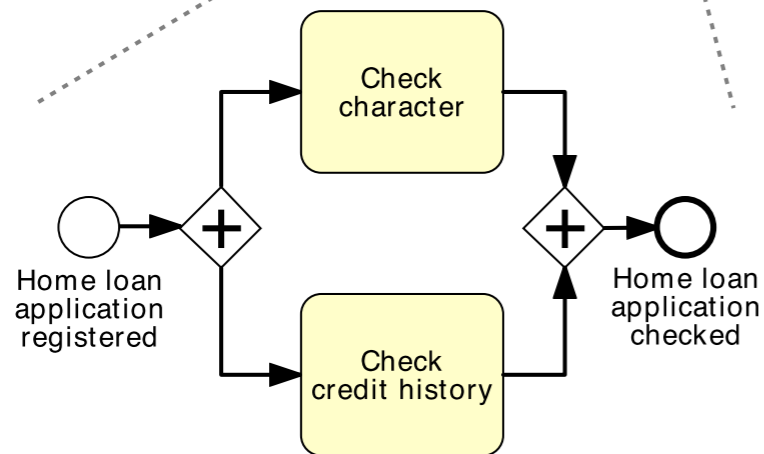
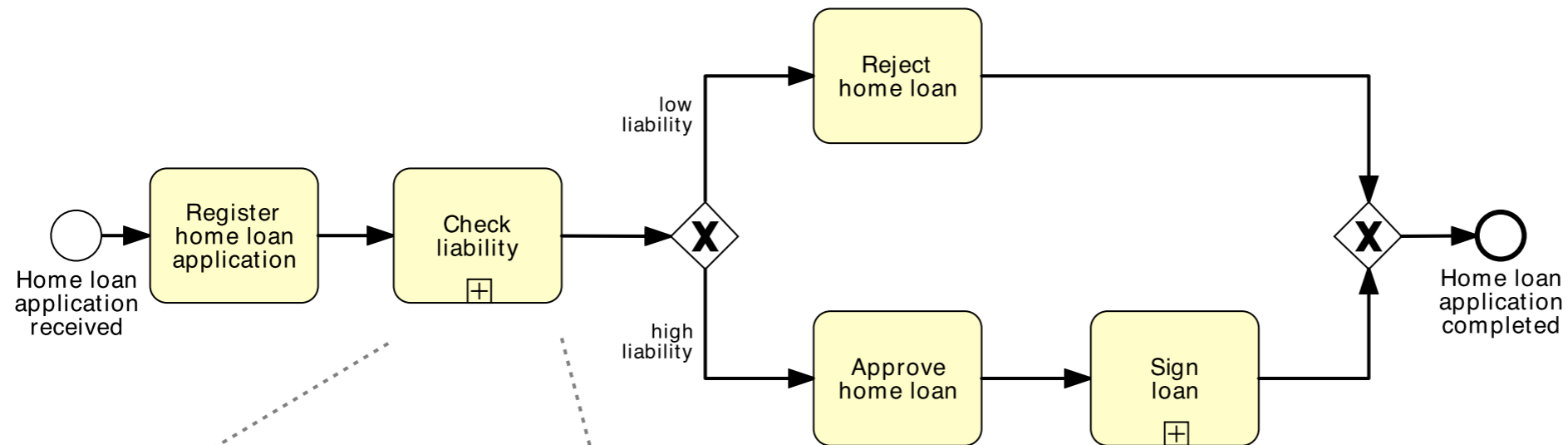


Call activities

# Nesting sub-processes: home loans



# Global sub-processes: home / student loans



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

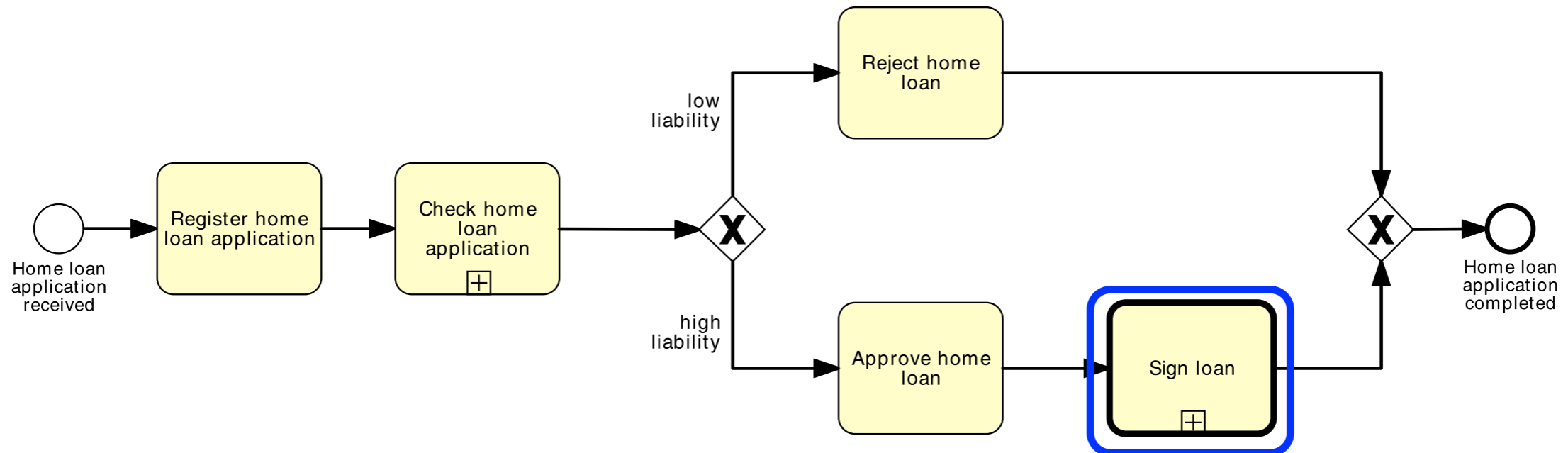
# Call activities



Call Activity

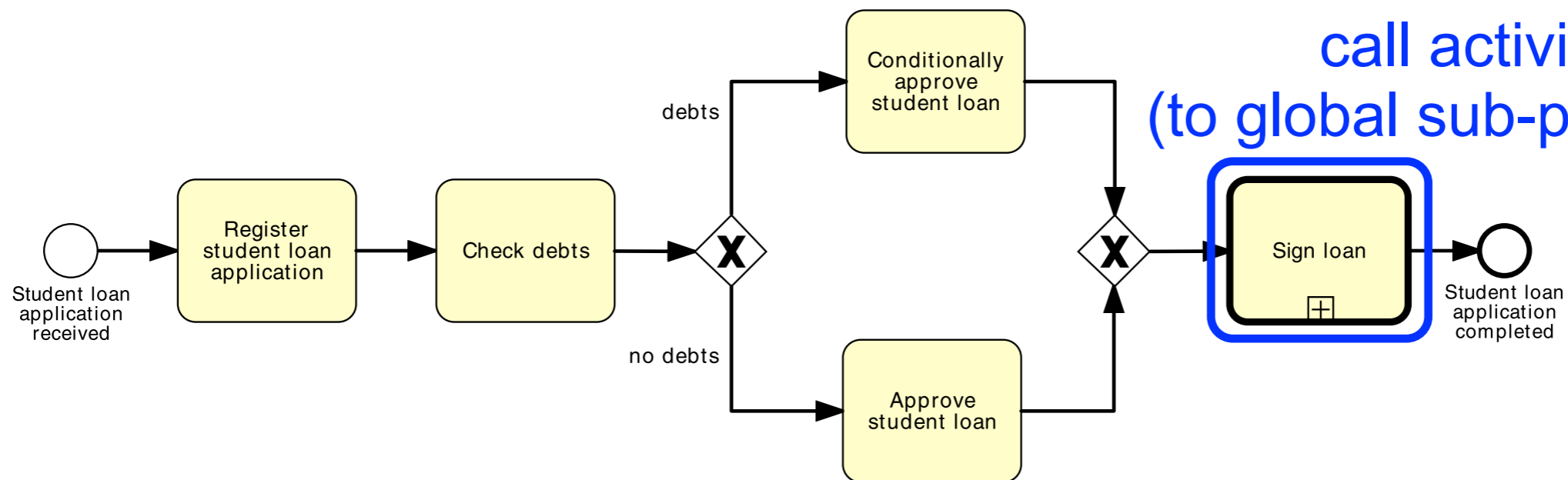
A **Call Activity** is a wrapper for a globally defined Sub-Process or Task that is reused in the current process.

# Call activities: home / student loans



thick borders denote  
call activities

(to global sub-processes)



# Global processes: advantages

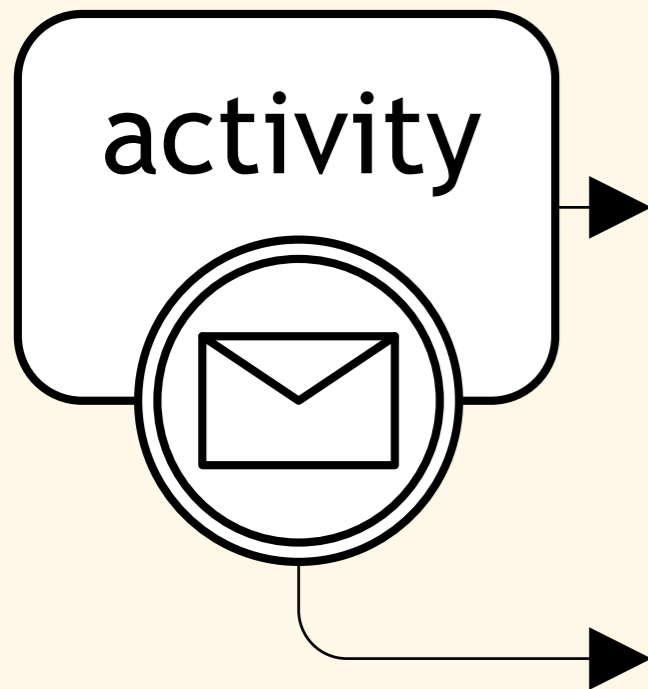
Readability: processes tend to be smaller

Reusability: define once, use many time

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

Throwing and catching

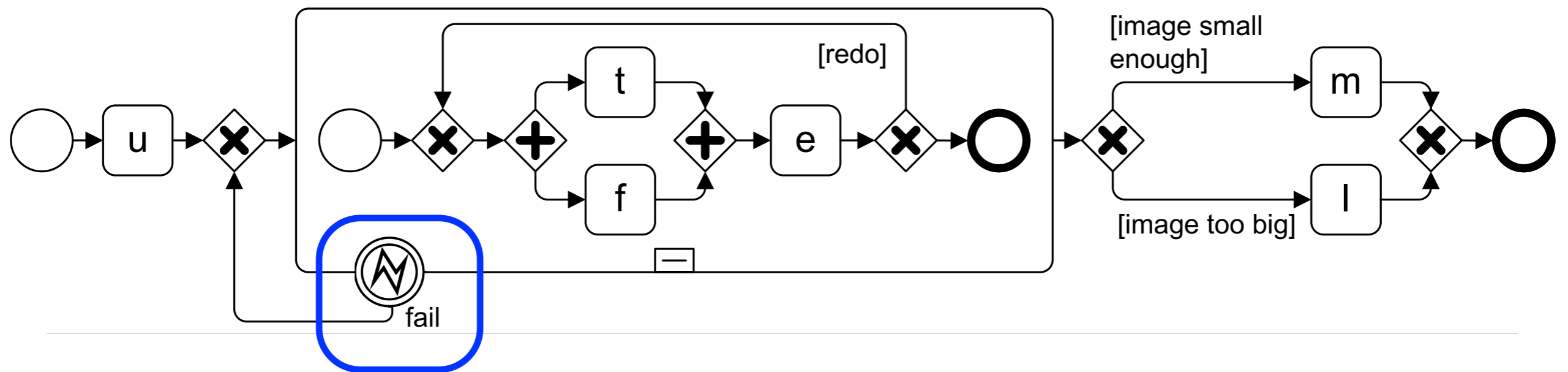
# Attached events



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

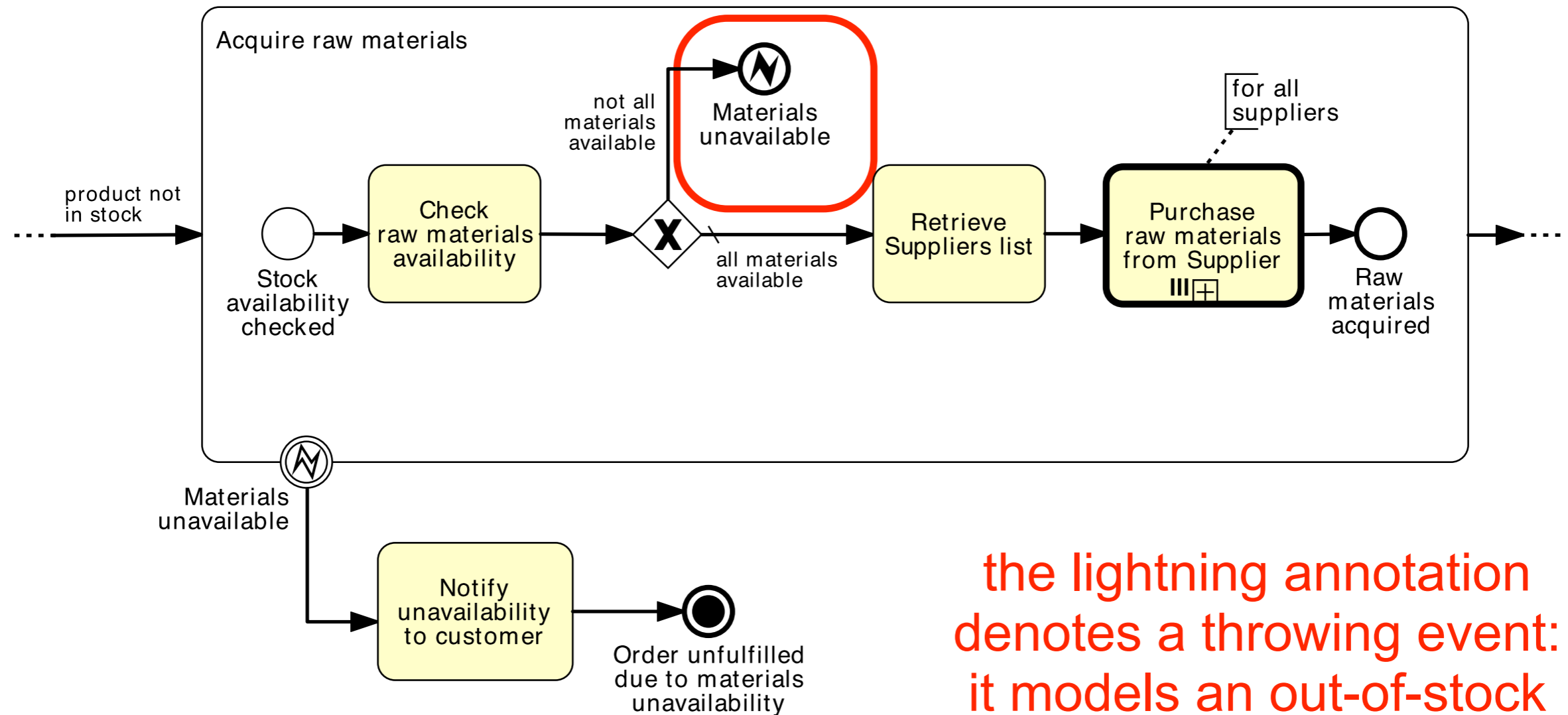


# Recovery from faults: image manipulation



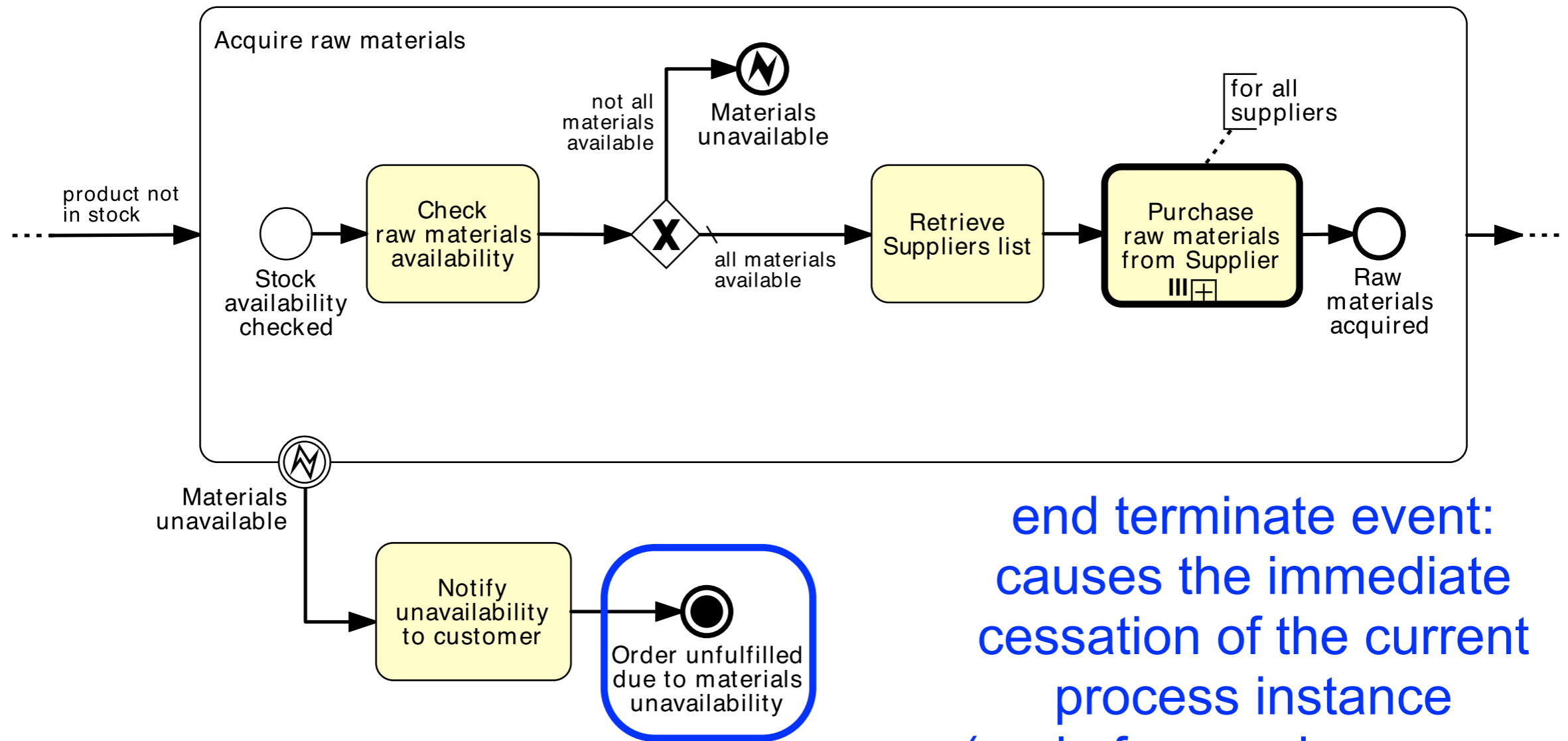
the lightning annotation  
denotes an error-catching event

# Throwing and catching: order fulfillment



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

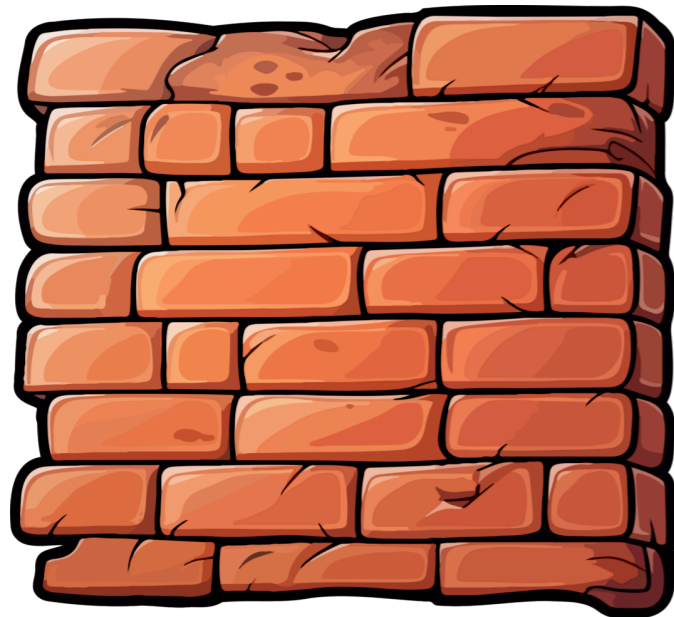
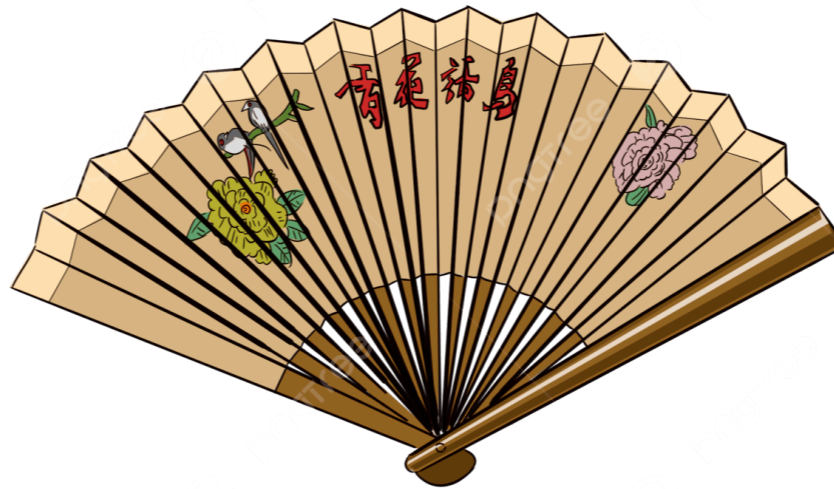
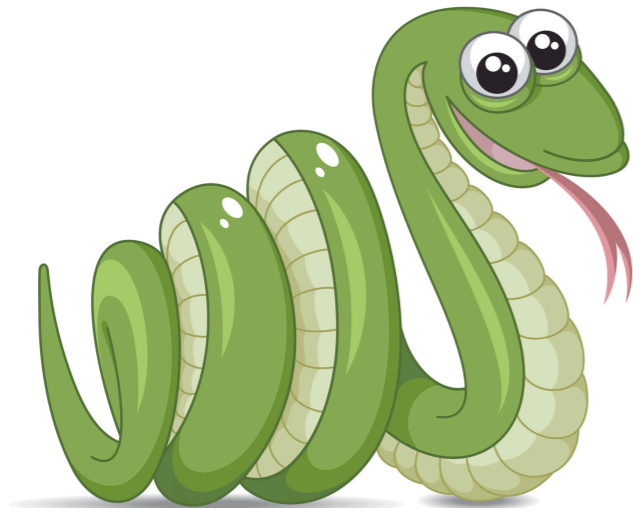
# Throwing and catching: order fulfillment



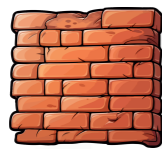
end terminate event:  
causes the immediate  
cessation of the current  
process instance  
(and of any sub-process,  
but not of the parent process if any)

# Choreographies

# Guess the animal



# Single views



# Global view



# Global view



A group of blind men heard that a strange animal, called an elephant, had been brought to the town, but none of them were aware of its shape and form. Out of curiosity, they said: "We must inspect and know it by touch, of which we are capable". So, they sought it out, and when they found it they groped about it. The first person, whose hand landed on the trunk, said, "This being is like a thick snake". For another one whose hand reached its ear, it seemed like a kind of fan. As for another person, whose hand was upon its leg, said, the elephant is a pillar like a tree-trunk. The blind man who placed his hand upon its side said the elephant, "is a wall". Another who felt its tail, described it as a rope. The last felt its tusk, stating the elephant is that which is hard, smooth and like a spear.



# Choreography

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

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

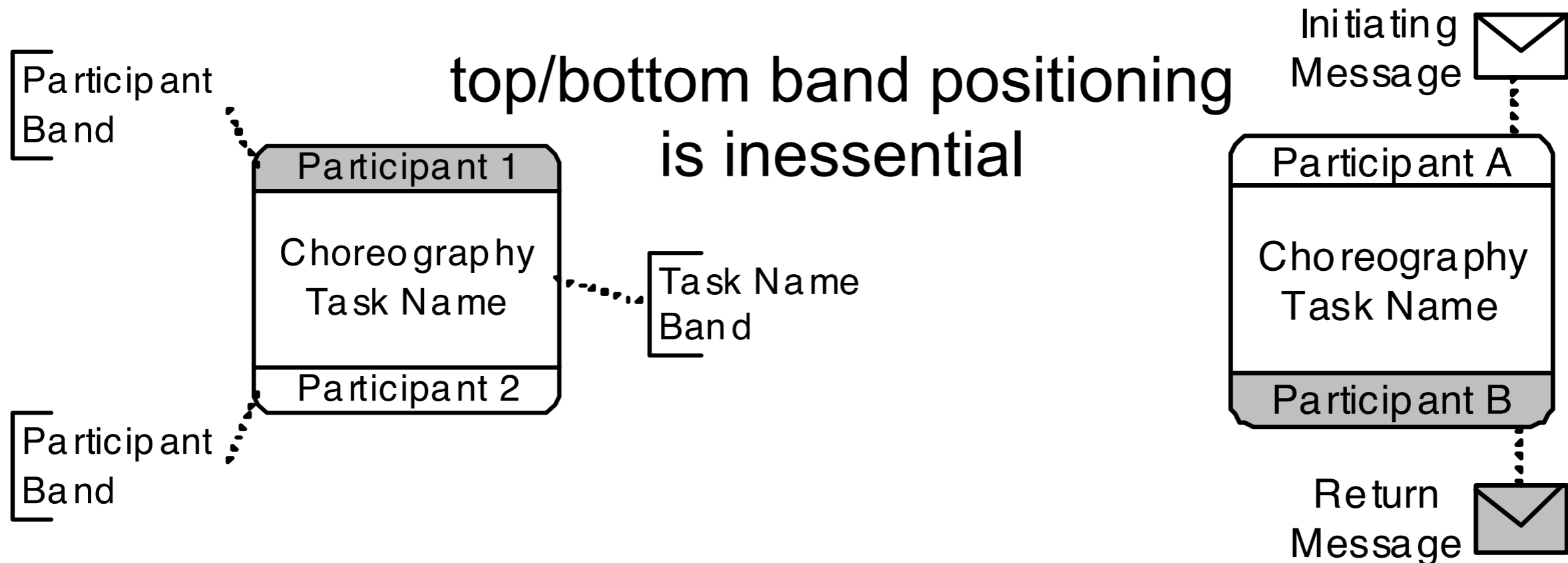
It describes how the participants are supposed to behave

a choreography can also use message data objects

# Choreography task

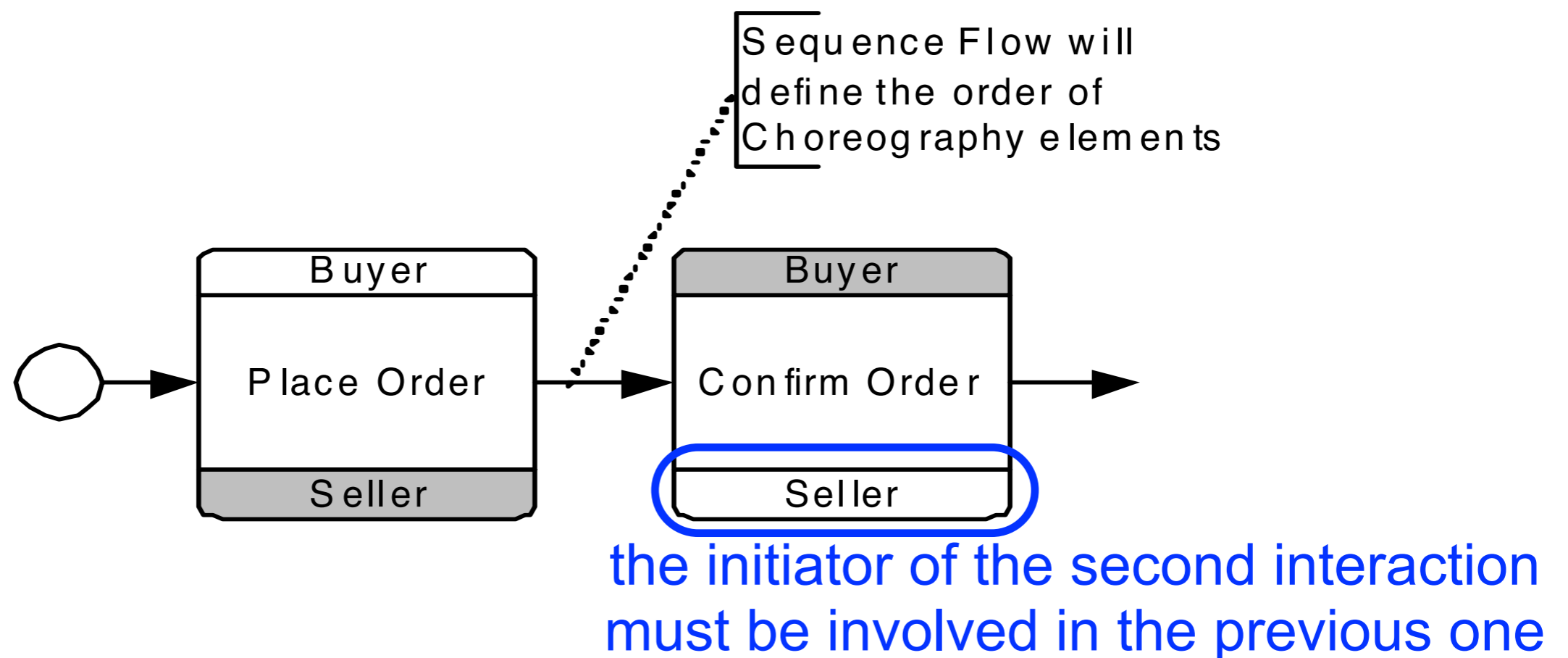
A **choreography task** is an activity in a choreography that consists of a set (one or more) communications

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

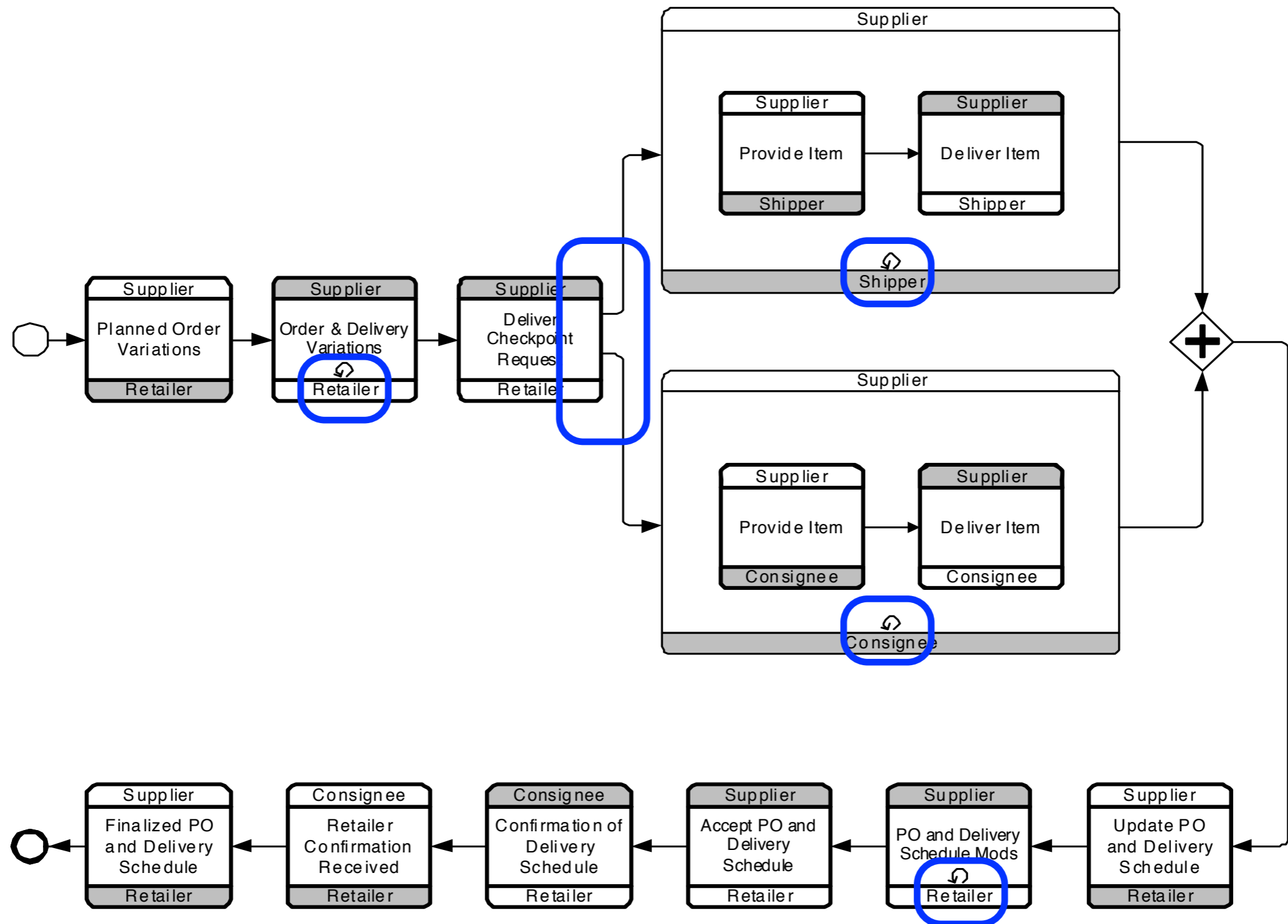


# Choreography flow

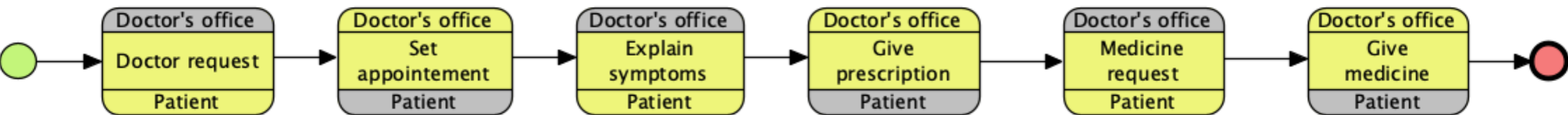
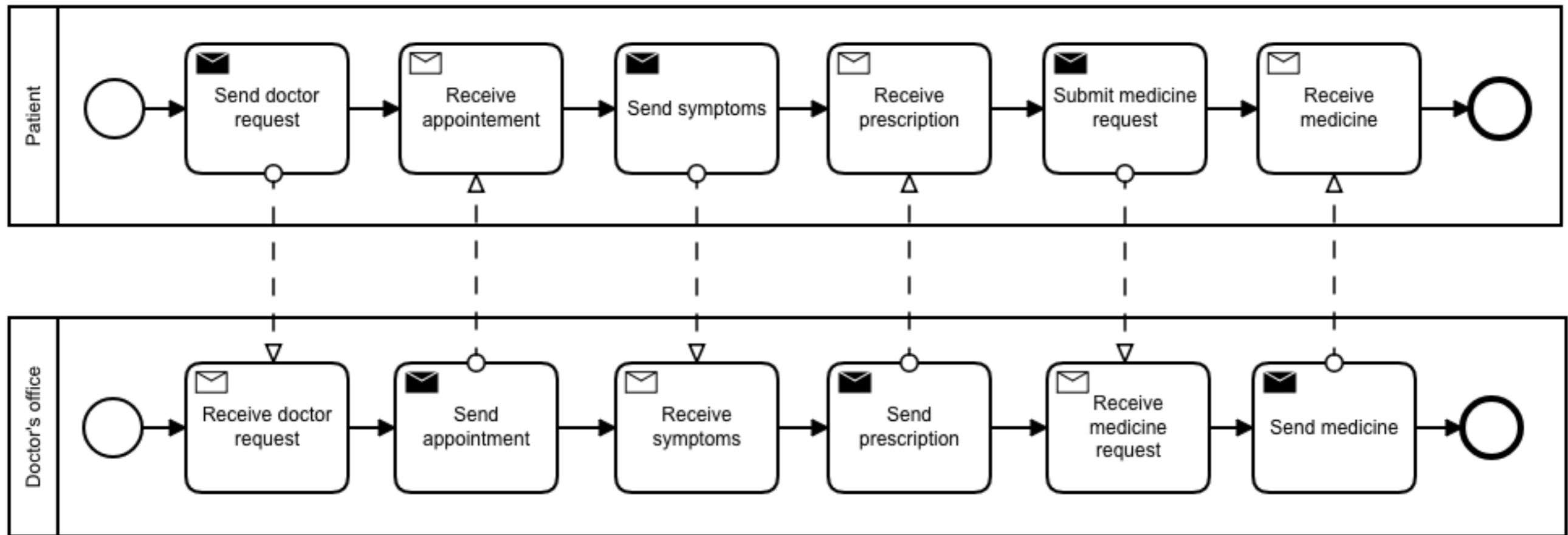
Ordinary sequence flow and gateways are used within choreographies to show the sequence of tasks involved



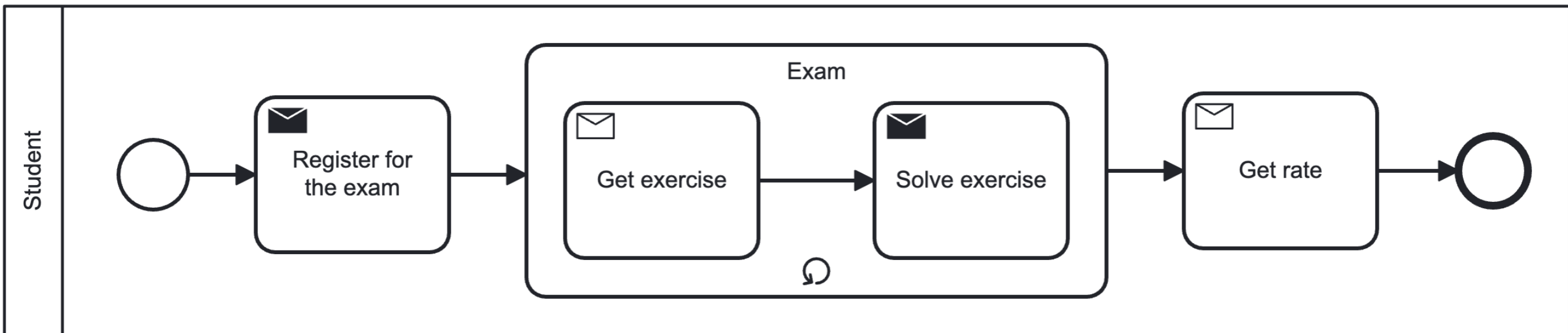
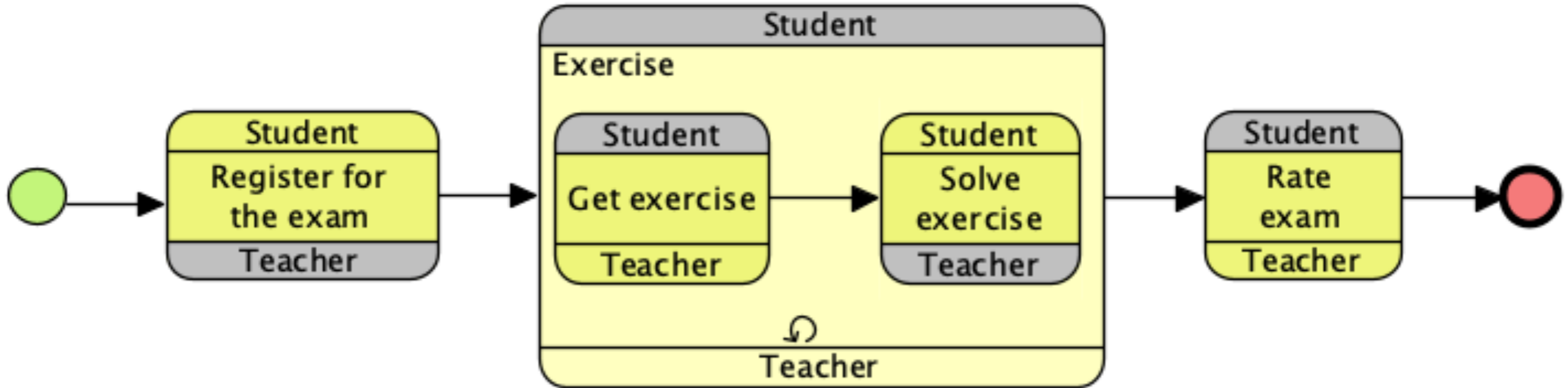
# A large choreography



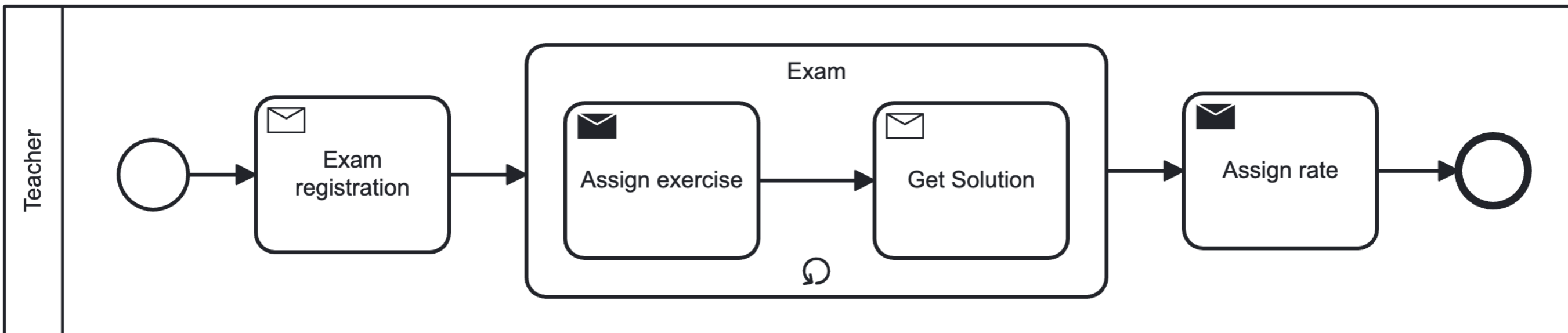
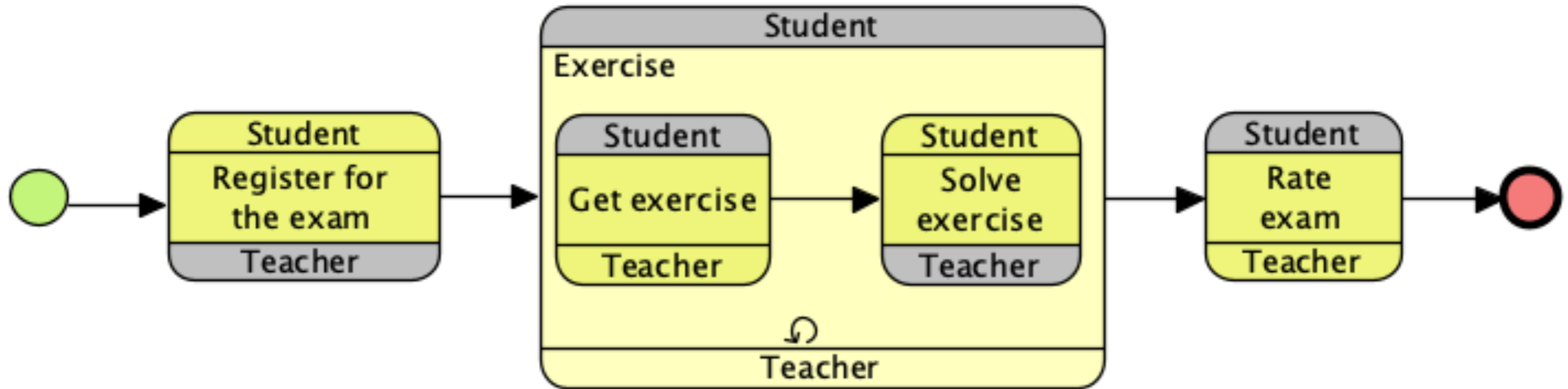
# From collaborations to choreographies



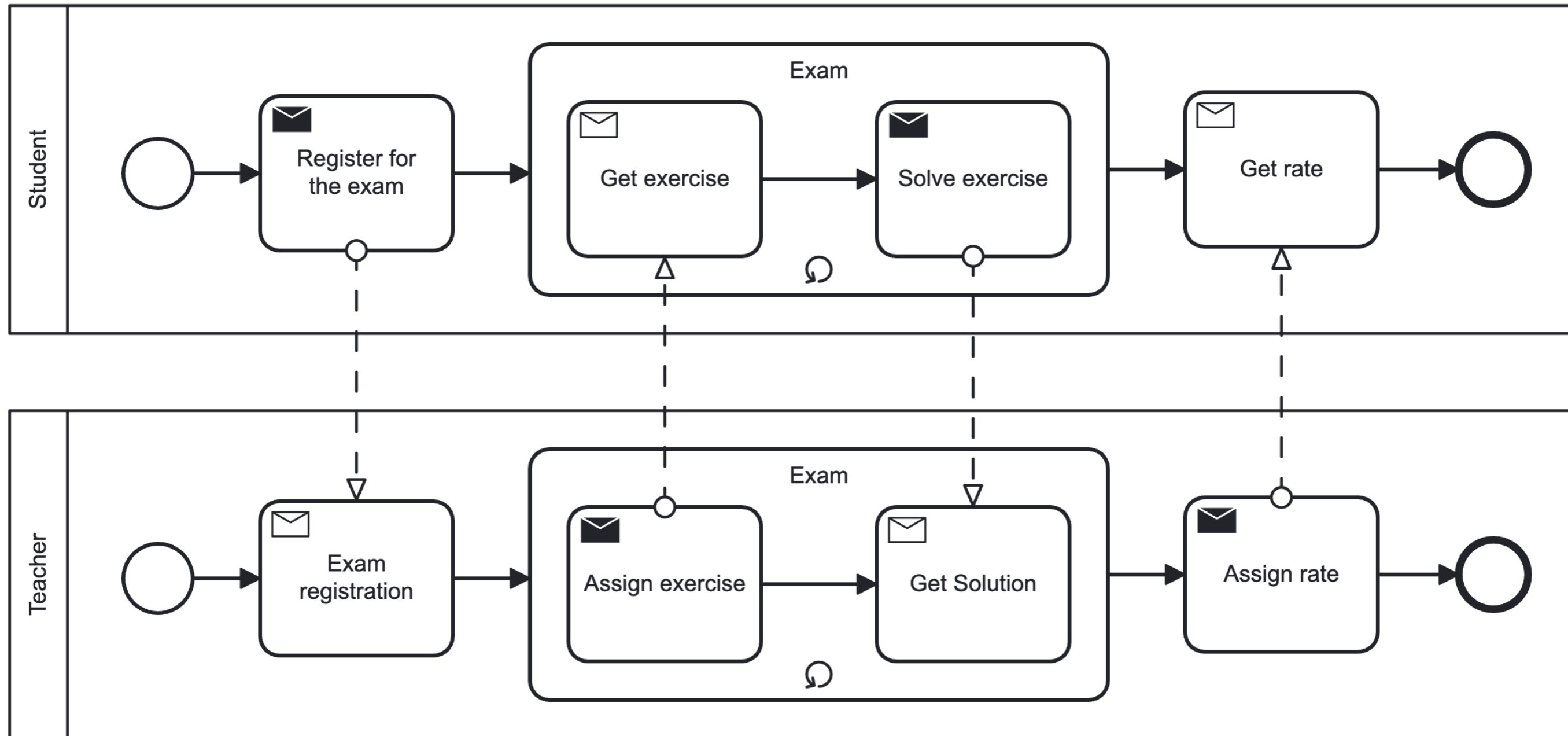
# Projection (on Student)



# Projection (on Teacher)

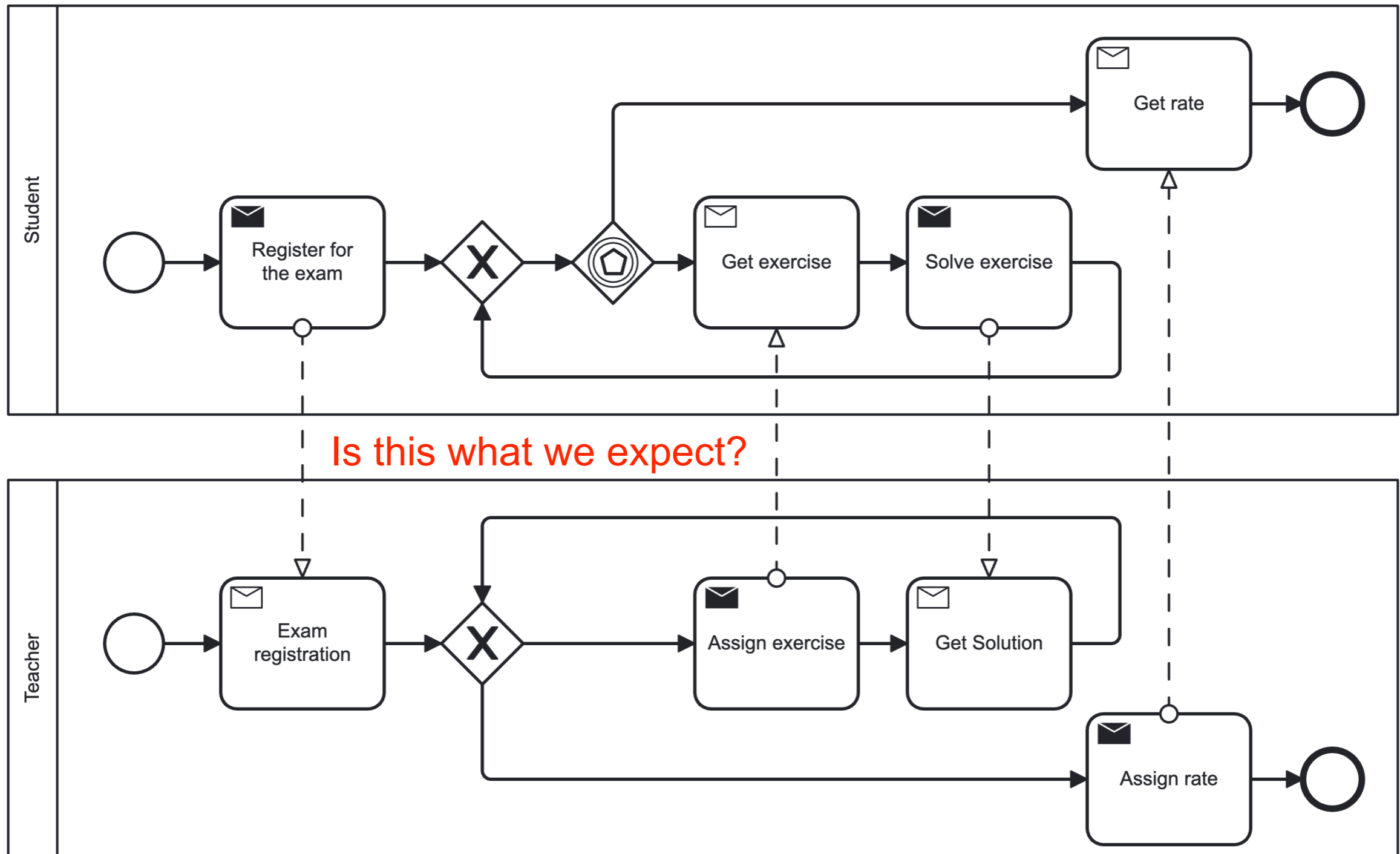


# From choreographies to collaborations



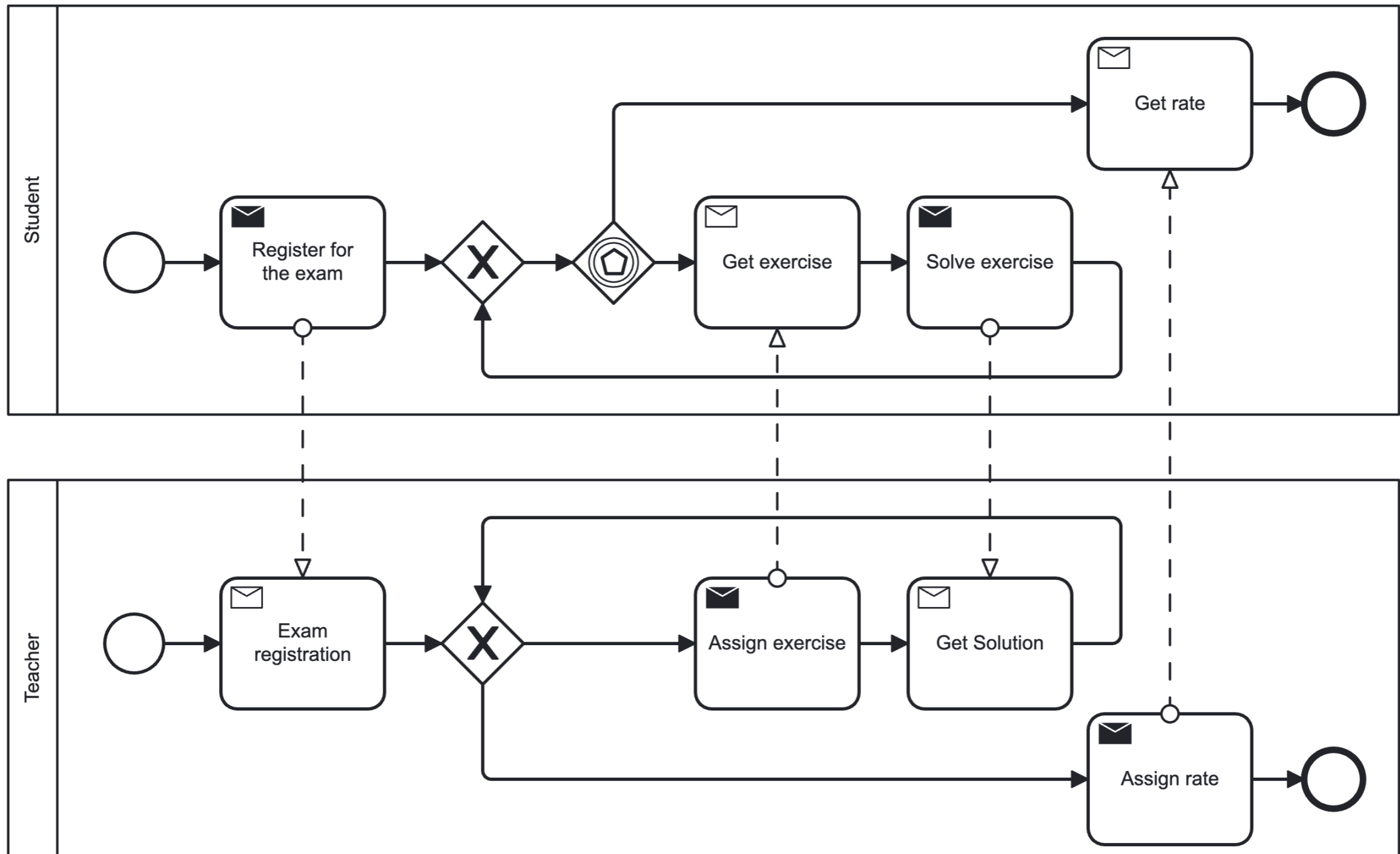


# From choreographies to collaborations



# Exercise

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



BPMN basics

BPMN key features

More on BPMN

**BPMN semantics**

# BPMN execution semantics



Date: January 2011



## Business Process Model and Notation (BPMN)

*Version 2.0*

Standard document URL: <http://www.omg.org/spec/BPMN/2.0>

13 BPMN Execution Semantics .....	425
13.1 Process Instantiation and Termination .....	426
13.2 Activities .....	426
13.2.1 Sequence Flow Considerations .....	427
13.2.2 Activity .....	428
13.2.3 Task .....	430
13.2.4 Sub-Process/Call Activity .....	430
13.2.5 Ad-Hoc Sub-Process .....	431
13.2.6 Loop Activity.....	432
13.2.7 Multiple Instances Activity .....	432
13.3 Gateways .....	434
13.3.1 Parallel Gateway (Fork and Join) .....	434
13.3.2 Exclusive Gateway (Exclusive Decision (data-based) and Exclusive Merge) ...	435
13.3.3 Inclusive Gateway (Inclusive Decision and Inclusive Merge) .....	435
13.3.4 Event-based Gateway (Exclusive Decision (event-based)) .....	437
13.3.5 Complex Gateway (related to Complex Condition and Complex Merge) .....	437
13.4 Events .....	439
13.4.1 Start Events .....	439
13.4.2 Intermediate Events .....	440
13.4.3 Intermediate Boundary Events .....	440
13.4.4 Event Sub-Processes .....	440
13.4.5 Compensation .....	441
13.4.6 End Events .....	443

# Some sample paragraphs

The execution semantics are described informally (textually), and this is based on prior research involving the formalization of execution semantics using mathematical formalisms.

A **Process** is instantiated when one of its **Start Events** occurs.

A **Process** can also be started via an **Event-Based Gateway** or a **Receive Task** that has no incoming **Sequence Flows**

Each **Start Event** that occurs creates a *token* on its outgoing **Sequence Flows**, which is followed as described by the semantics of the other **Process** elements.

A **Process** *instance* is completed, if and only if the following three conditions hold:

- If the *instance* was created through an instantiating **Parallel Gateway**, then all subsequent **Events** (of that **Gateway**) **MUST** have occurred.
- There is no *token* remaining within the **Process** *instance*.
- No **Activity** of the **Process** is still active.

For a **Process** *instance* to become completed, all *tokens* in that instance **MUST** reach an end node.

A *token* reaching an **End Event** triggers the behavior associated with the **Event** type.

If a *token* reaches a **Terminate End Event**, the entire **Process** is abnormally terminated.

# BPMN formal semantics?

Many attempts:

Abstract State Machines (ASM)

Term Rewriting Systems

Graph Rewrite Systems

Process Algebras

Temporal Logic

...

**Petri nets**

(Usual difficulties with OR-join semantics)

# Exercises

(modelling with BPMN)

# Exercises

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



# Example: loan application

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:

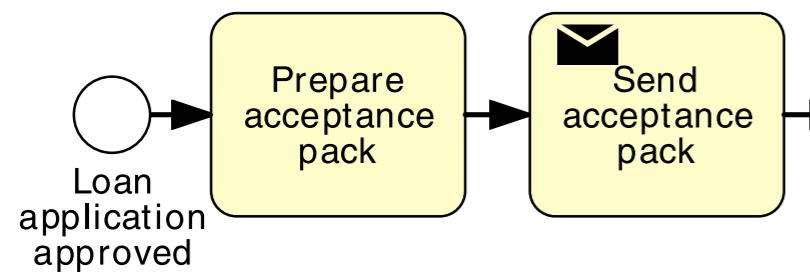
**if** the applicant disagreed with the repayment schedule, the loan provider  **Cancels** the application;

**if** the applicant agreed, the loan provider  **approves** the application.

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

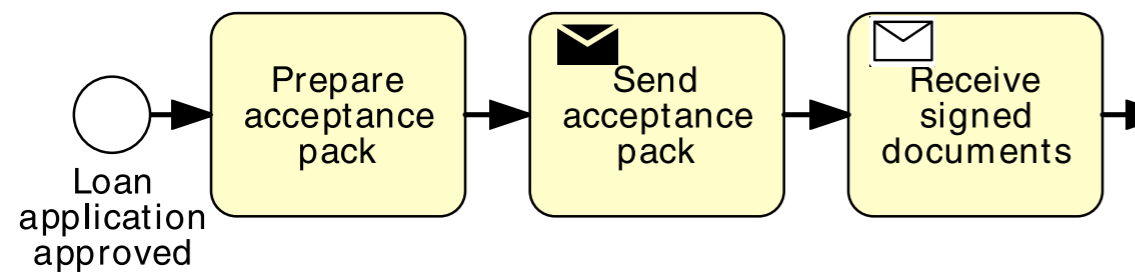
# Example: loan application

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

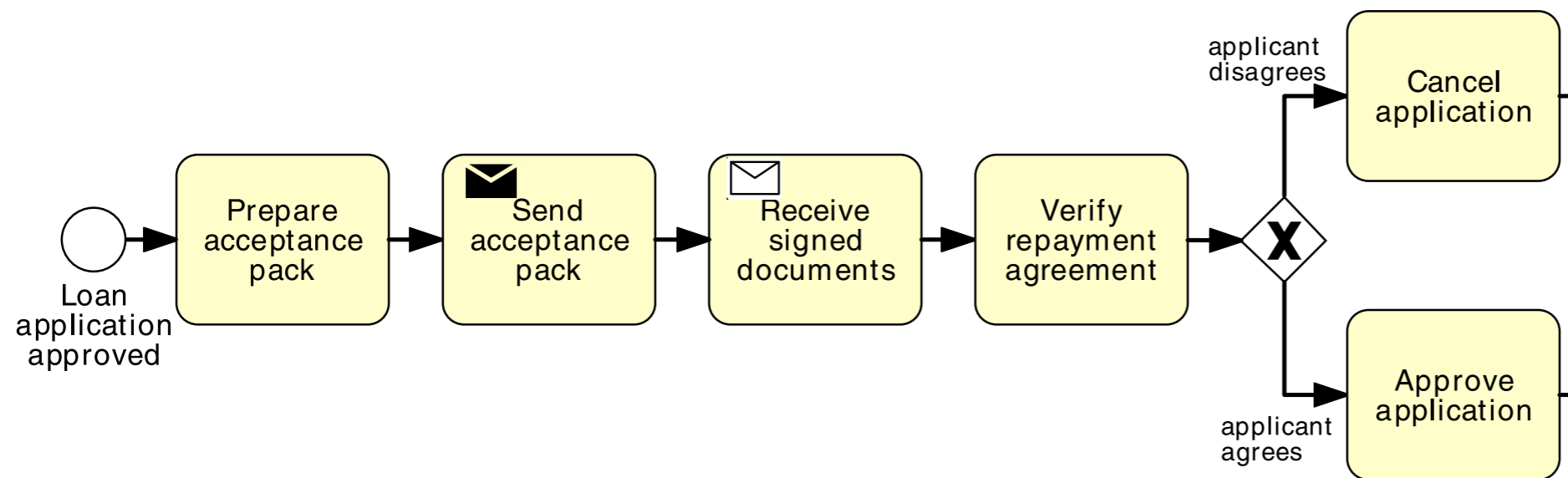


# Example: loan application

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



# Example: loan application

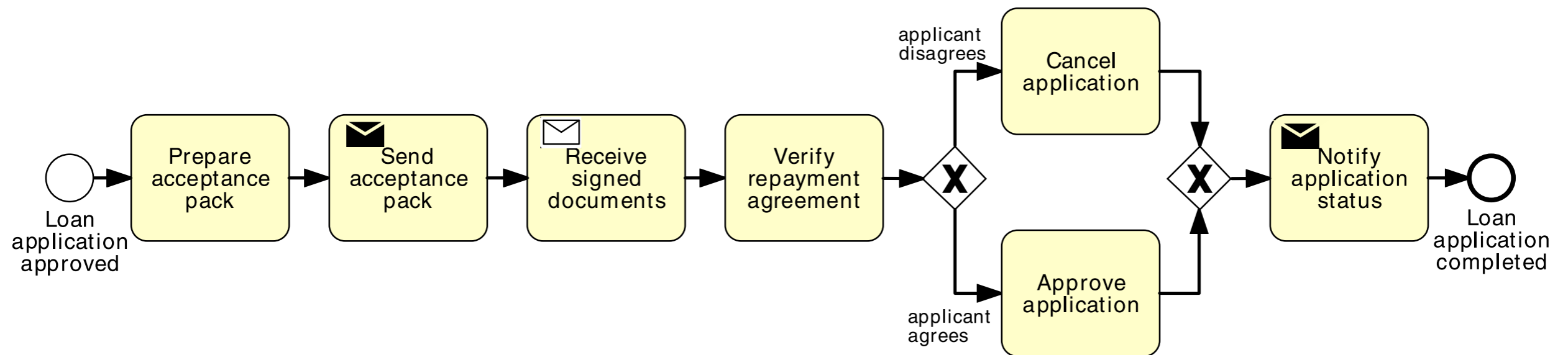


The latter then verifies the repayment agreement:

**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.

# Example: loan application



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

# Exercise: loan application 1

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 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

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 5

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: loan application 6

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