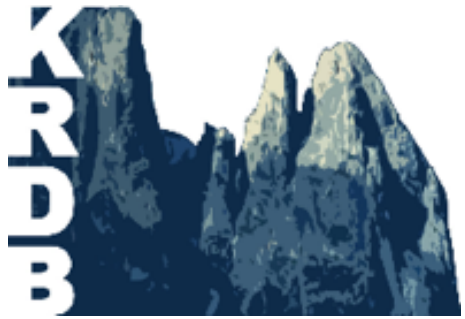
Data and Process Modelling

8a. BPMN - Basic Modelling

Marco Montali

KRDB Research Centre for Knowledge and Data
Faculty of Computer Science
Free University of Bozen-Bolzano

A.Y. 2014/2015



Business Process Model and Notation

OMG standardization initiative.

Charter (http://www.bpmn.org)

A standard Business Process Modeling Notation (BPMN) will provide businesses with the capability of understanding their internal business procedures in a graphical notation and will give organizations the ability to communicate these procedures in a standard manner. Furthermore, the graphical notation will facilitate the understanding of the performance collaborations and business transactions between the organizations. This will ensure that businesses will understand themselves and participants in their business and will enable organizations to adjust to new internal and B2B business circumstances quickly.

BPMN 2.0: Main Goals

- BPMN 2.0 as a single specification for notation, metamodel and interchange format.
- Enabling the exchange of BPs and their diagram layouts among process modeling tools to preserve semantic integrity.
- Support for model orchestrations and choreographies as stand-alone or integrated models.
- Support to the display and interchange of different perspectives on a model that allow a user to focus on specific concerns (internal, public, conversation, choreography).
- Provide an execution semantics via translation to executable WS-BPEL processes.
  - Achieved with strong assumptions on the shape of the acceptable BPMN models.

Uses of BPMN

Several coexisting modeling paradigms.

- Process (or orchestration): intra-organizational perspective.
  - Private, non-executable: intra-organizational, for documentation purposes (abstract).
  - Private, executable: intra-organizational, with fully specified information to enable executability (concrete languages for conditions, loops, choices, ...).
  - Public: interaction between a private BP and an external one. Only the internal activities involved in the interaction are shown.
- Collaboration: interaction between two or more business entities.
  - Multiple private processes with message exchange.
  - Choreography: contract (expected behavior) between interacting participants.
    - ★ No central orchestrator.
    - ★ Similar to a process, but each activity represents a message exchange.
  - Conversation: logical relation implied by message exchange.
    - ★ Focus on business artifacts.
    - ★ Elicitation of participants.
    - ★ Message exchange used by participants to manipulate artifacts.

We will focus on private, public, collaborative abstract processes.

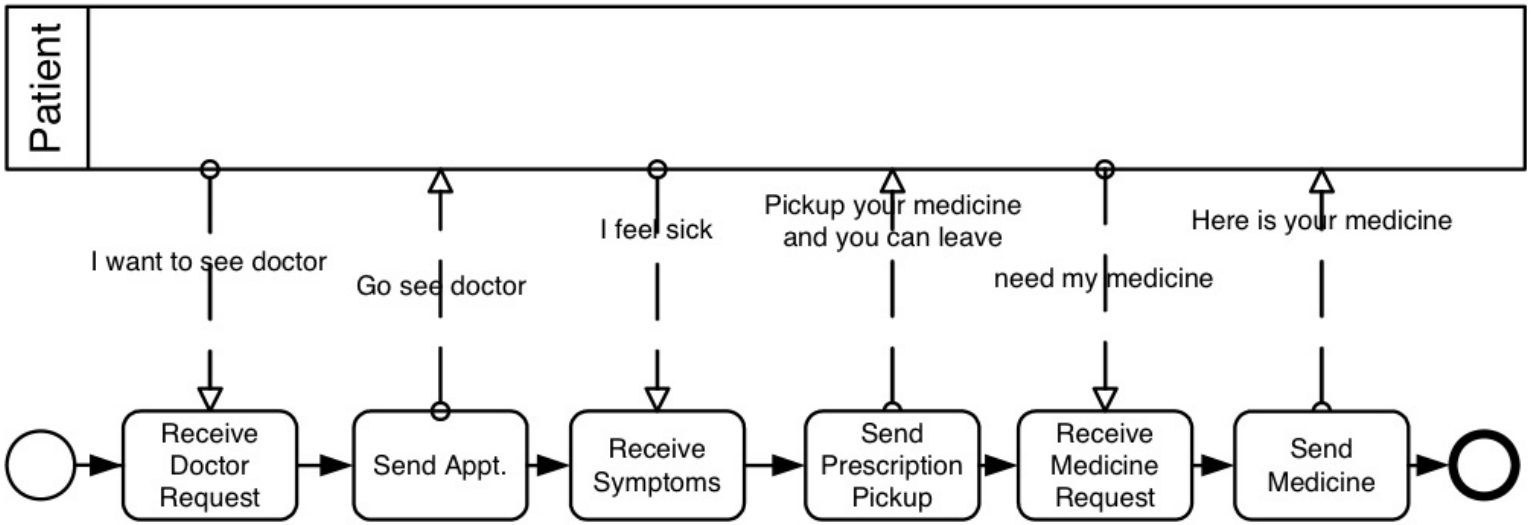
Examples - Control-Flow

Private process.

Examples - Control-Flow

Private process.

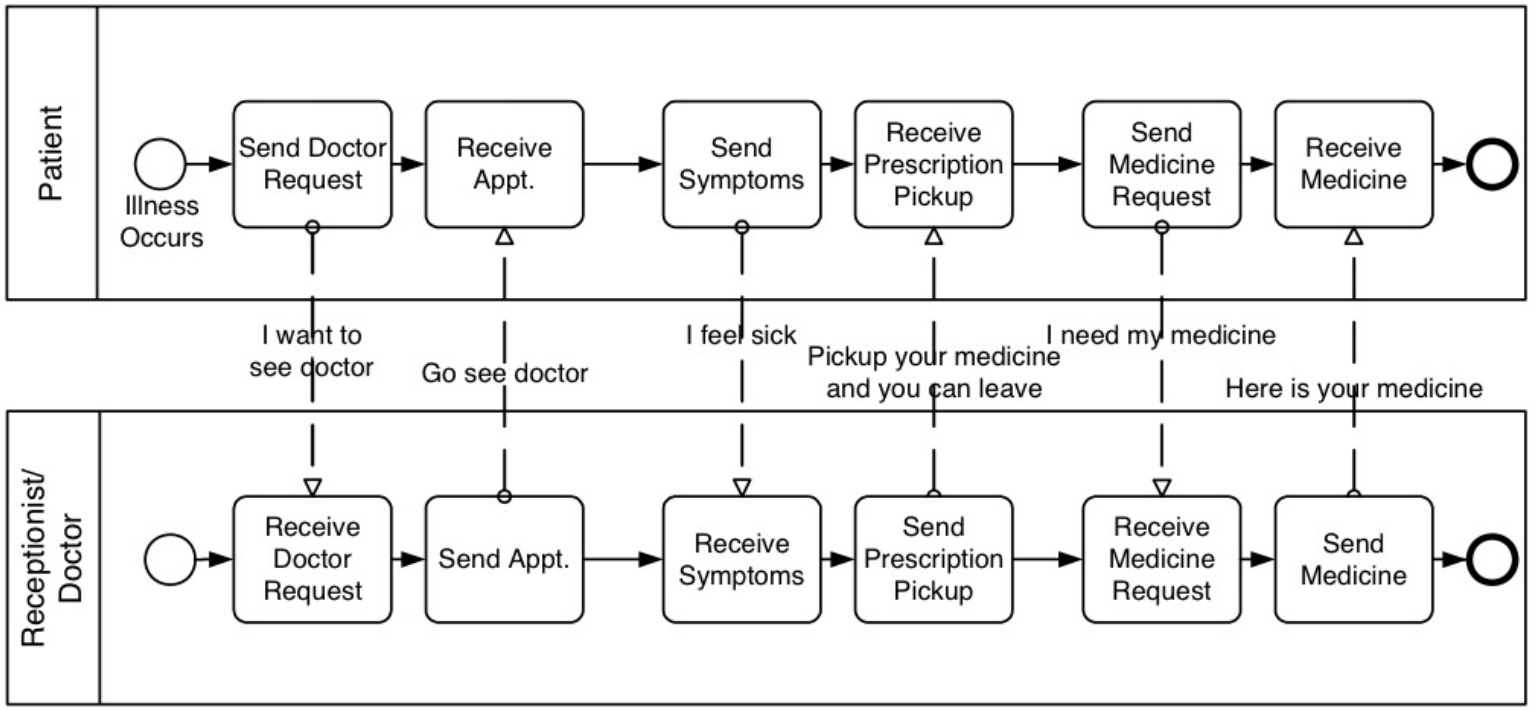
Public process.



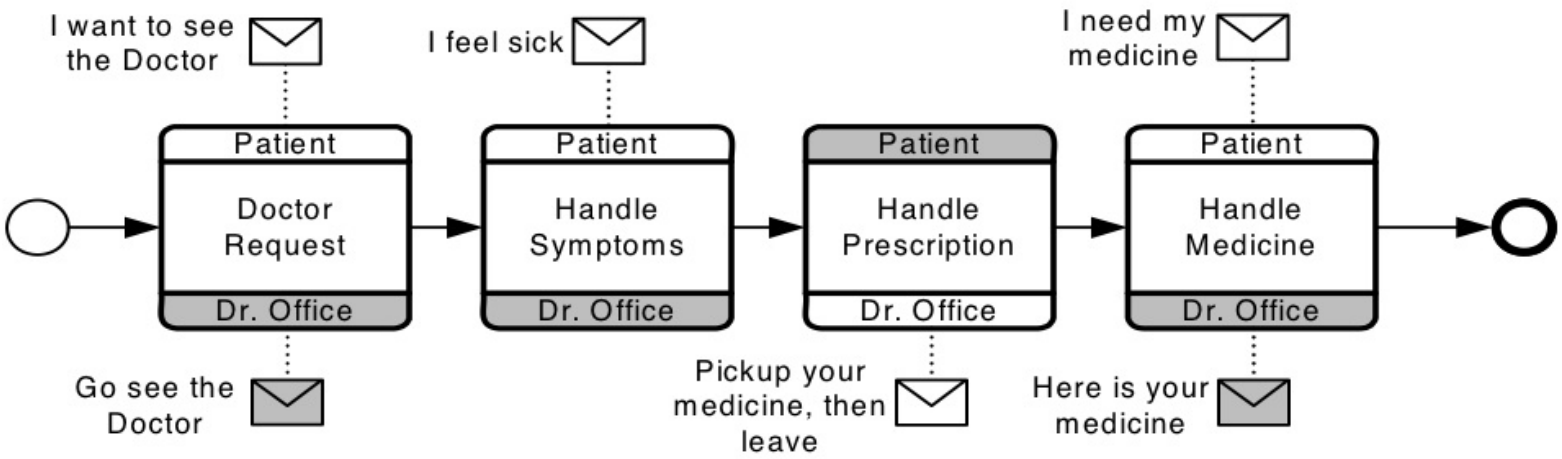
Examples - Control-Flow

Private process.

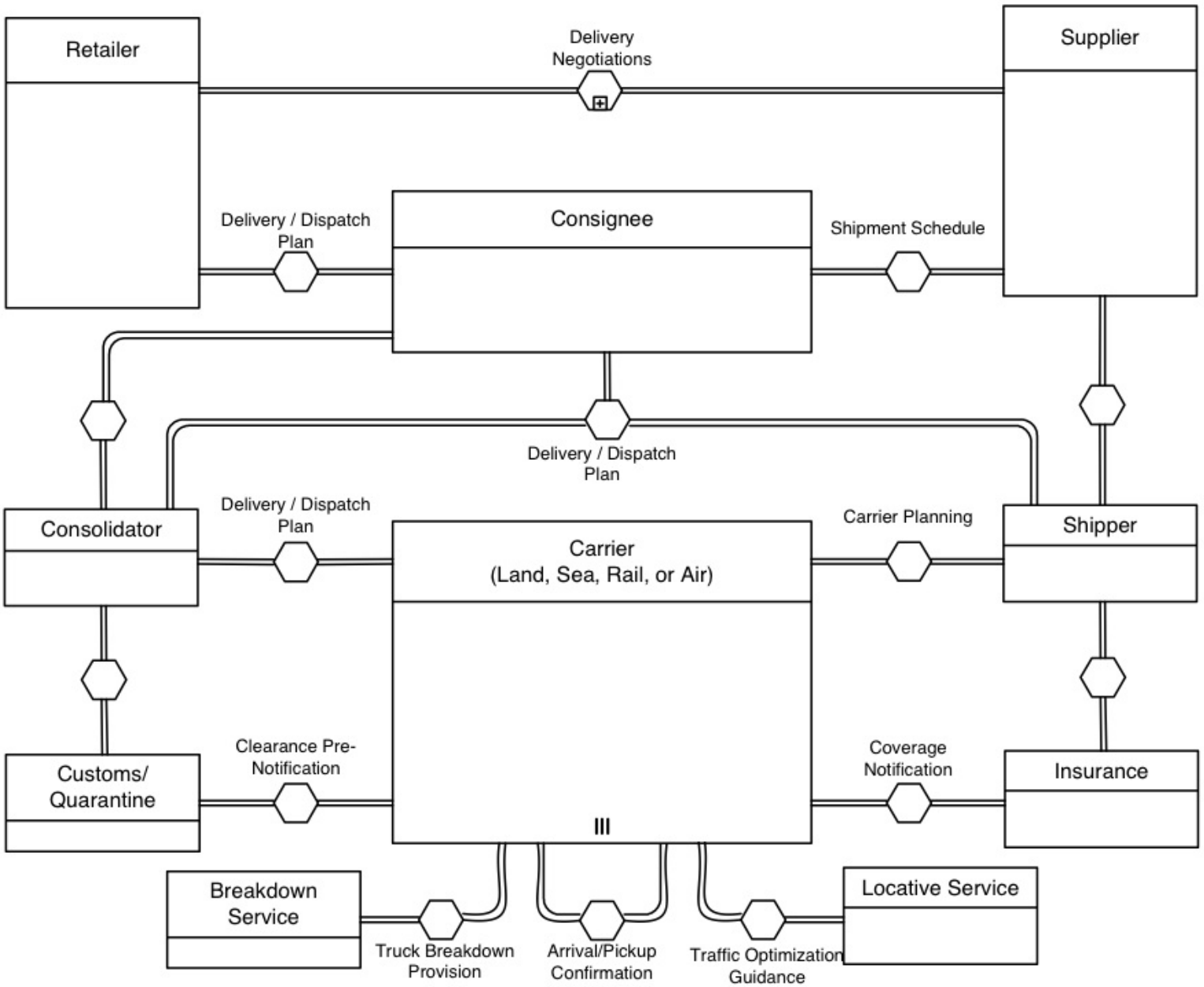
Collaborative process.



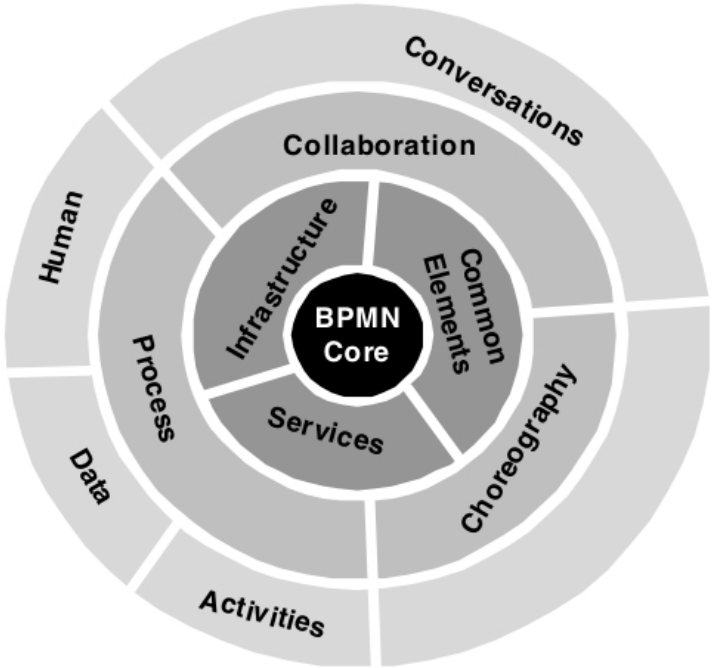
Example - Choreography



Example - Conversation



Core Structure



Every component of the structure:

- Is associated to a graphical notation.
- Is associated to a well-defined metamodel, capturing also the relationships with other components.
- Is associated to an XSD that corresponds to the metamodel and is used for validation, storage and interchange.

Metamodel and XSD: `http://www.bpmn.org`.

BPMN Graphical Elements

Strategy:

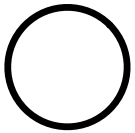
- elements grouped into 5 families;
- each family organized in two strata - basic and advanced elements.

Families:

- Flow objects: behavior of the BP.
- Data: manipulated information.
- Connecting objects: connection between flow objects and other elements.
- Swimlanes: organizational grouping of modeling elements.
- Artifacts: additional infos.

The Main Basic Elements

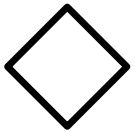
Task.



Event.

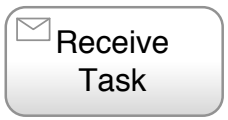
- Start event: thin solid line.
- Intermediate event: double line (with variants).
- End event: thick solid line.

Flow.

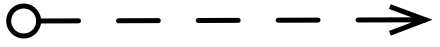
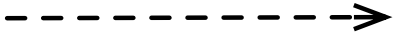


Gateway.

Task Types

	Generic task.		Executed by human operator on a terminal (workitem).
	Invokes a service, not shown or provided by other pool (message flow).		Human task executed without the support of the IS.
	Sends a message to an external participant, not shown or in another pool (message flow).		Interaction with a business rule engine.
	Waits for a message from another participant, possibly explicitly identified (another pool, message flow).		Script interpreted and executed by the BP engine.
	The incoming message has the effect of instantiating a new process.		

Connecting Objects

	Sequence flow.
	Message flow.
	Directional association: link from/to a data item (data flow).
	Connection with other artifacts (e.g., notes, textual annotations).

Main Gateways

Elements used to control the interaction of sequence flows. Correspond to some of the control-flow patterns.

- Exclusive (none or X): xor-split/join (and multi-merge).
- Parallel (+): and-split/join.
- Inclusive (thick circle): or-split/join.
- Complex (∗): complex split conditions (like n-out of-m join, discriminator, . . .).

Outgoing sequence flows of exclusive, inclusive and complex gateways can be associated to condition expressions. A "default" condition is represented by a bar on the sequence flow.

Execution Semantics of BPMN

Can be found in the official documentation, Chapter 13.

Token game

Provides an intuition of the execution semantics (on the blackboard).

Exclusive Gateway

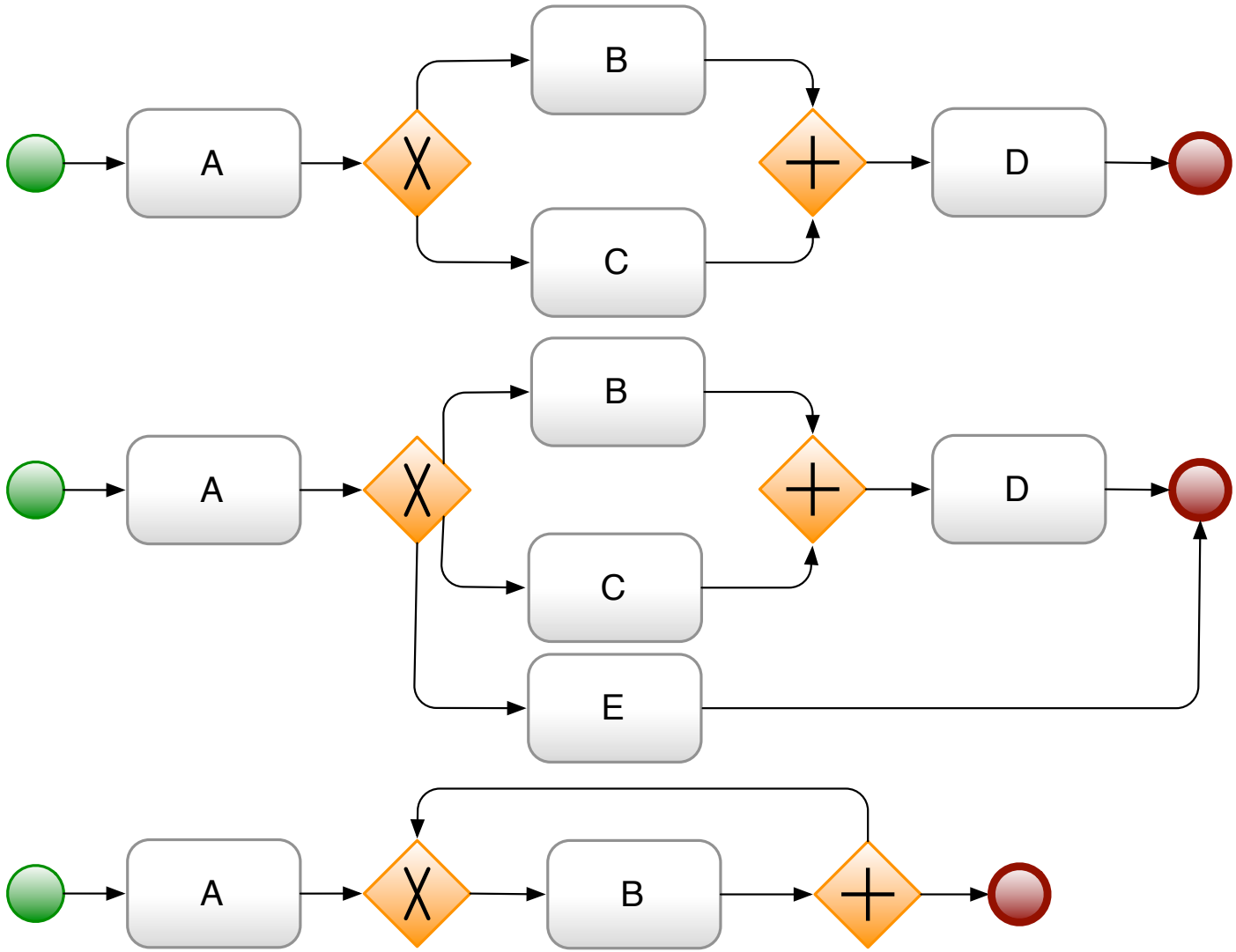
- Split: represents a decision point.
  - One and only one outgoing branch is selected.
  - If conditions are not present, the choice is under the responsibility of the executors (deferred choice).
  - If conditions are present, they must be mutually exclusive, and the one that evaluates to true determines the chosen path (exclusive choice). Default condition needed if the other conditions do not cover all the possible cases.
- Join: factorizes common process parts.
  - Whenever a token is received from one of the incoming branches, it is forwarded to the outgoing branch.
  - Corresponds to the simple/multi-merge in the control-flow patterns terminology.
  - Can be also realized by directly attaching multiple incoming flows to a task.

Parallel Gateway

- Split: represents a fork point.
  - All the outgoing branches are followed in parallel.
  - Can also be realized by directly attaching multiple outgoing flows from a task.
- Join: synchronizes multiple running parallel threads of control.
  - Whenever *all* incoming branches are activated, the gateway forwards the execution to the outgoing branch.
  - "Reset" of the gateway depending on the context (multiple activation of the same branch possible?).

Gateway Combinations and Process Correctness

Are these models correct?

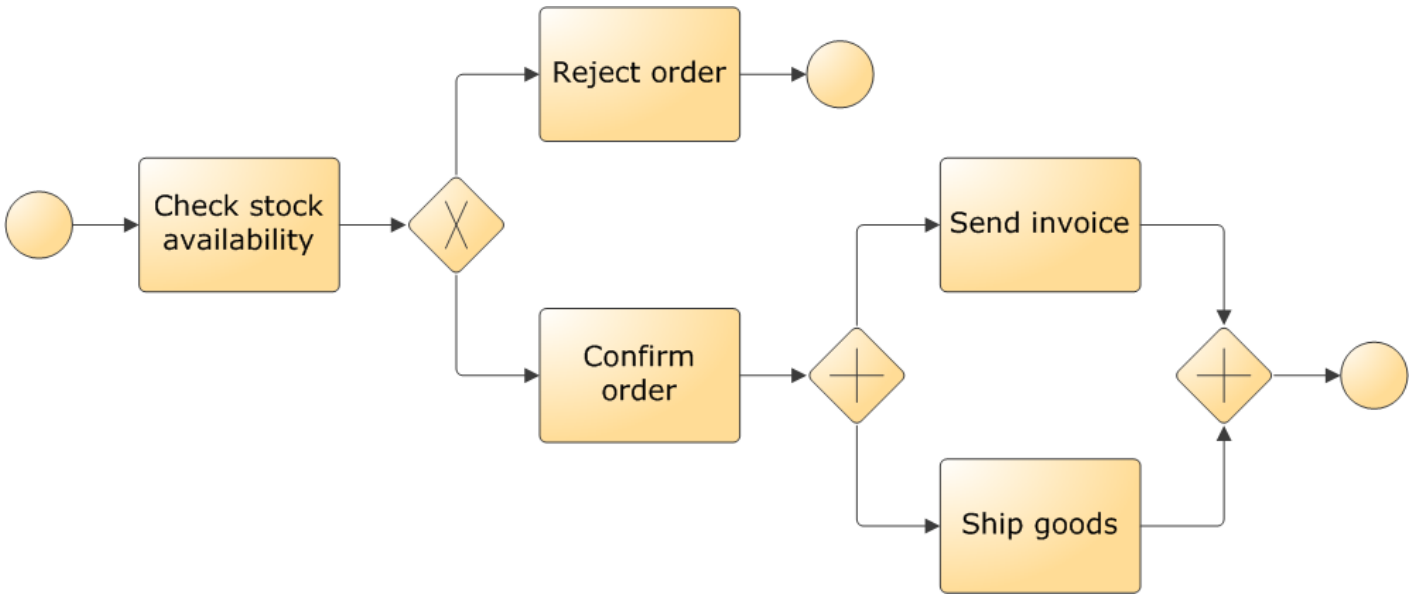


Example: Order Management Process

- The order management process starts by checking whether the goods requested in the order are available.
- If not, the order is rejected.
- If so, the order is confirmed.
- After the confirmation, an invoice is sent, and at the same time the requested goods are shipped. Then the process terminates.

Example: Order Management Process

- The order management process starts by checking whether the goods requested in the order are available.
- If not, the order is rejected.
- If so, the order is confirmed.
- After the confirmation, an invoice is sent, and at the same time the requested goods are shipped. Then the process terminates.



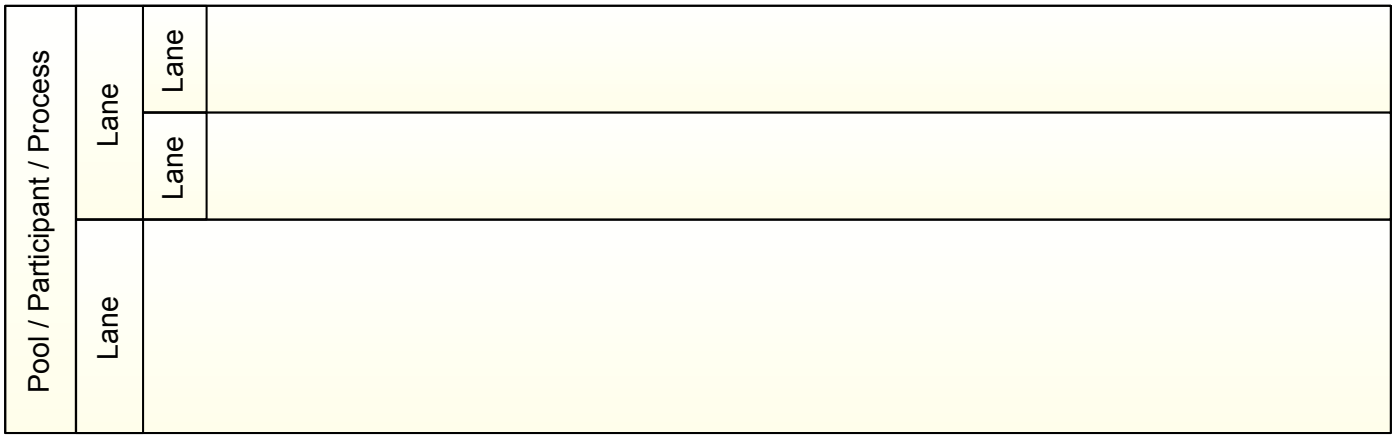
Organizational Modeling

Pool: representation of an independent resource class, with its own BP specification. It can be "implicit", i.e., not shown.

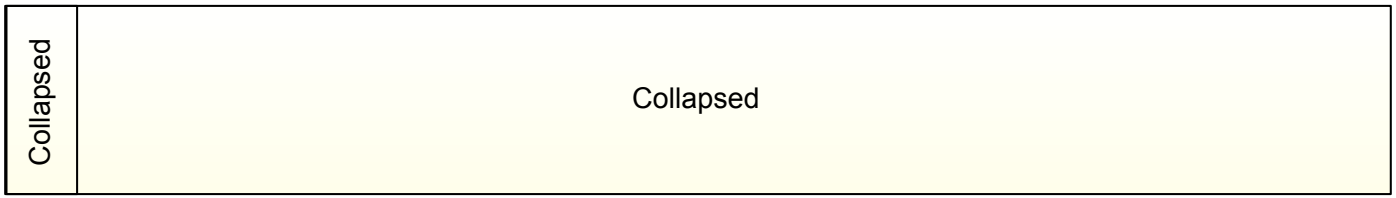
- Example: Customer, Supplier, Lab, Warehouse, ...

Lane: resource class in a given organizational space (pool), which shares the same process as other internal resource classes.

- Example: Manager, Sales Department, Engineer, ...



An "external" pool can also be treated as a black-box (no elements inside).



Pools and Message Exchange

Pools communicate with each other by means of message exchange.

- For a task to start, both the incoming control and message flows must be active!
- Messages could either be attached to tasks, or to the pools (abstraction).

Order Management Process with Pools

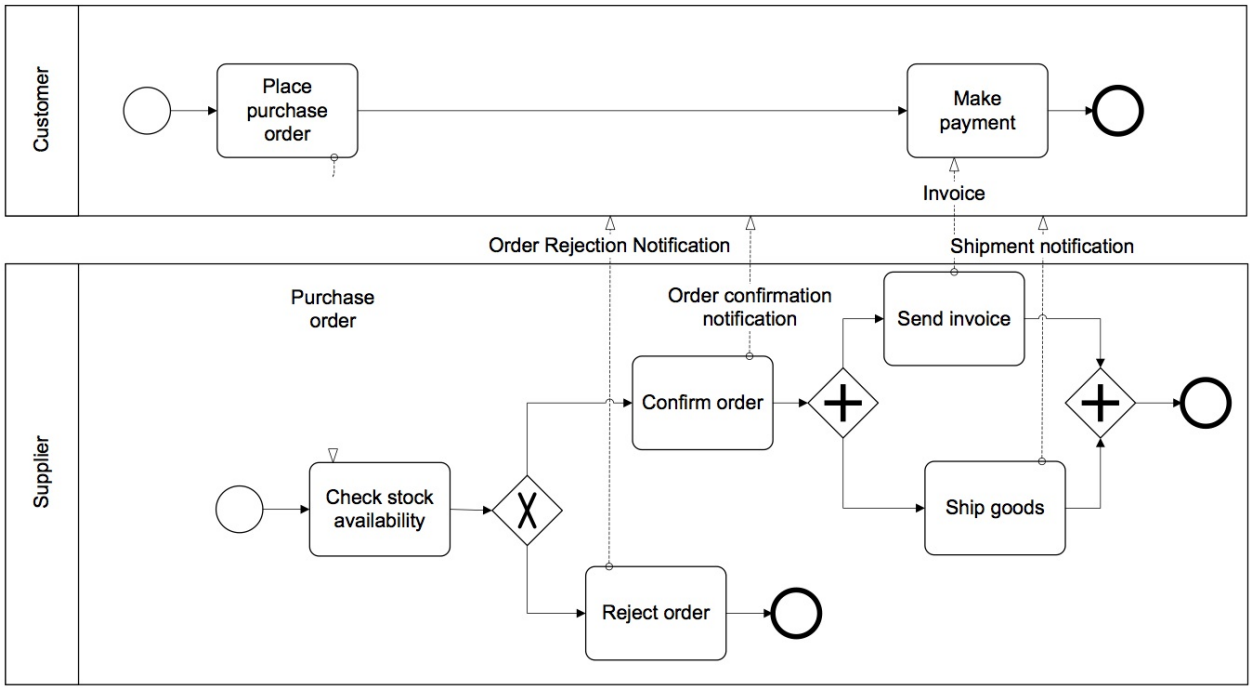
Example

Extend the Order Management Process by including the Customer. The Cusomer initially places the order, and if everything goes fine, performs the payment when the invoice is received.

Order Management Process with Pools

Example

Extend the Order Management Process by including the Customer. The Cusomer initially places the order, and if everything goes fine, performs the payment when the invoice is received.



We can extend the Order Management Process by modeling that the Supplier contains two departments: Sales and Warehouse.

Data

BPMN is domain agnostic: it does not provide a data model nor a pre-defined language to query and manipulate data.

Data Object

Data object: unit of information. Can be associated to a *state*.

Data Object Collection

Data object representing a collection of items.

Data Object [Object State]

Data object reference: refers to a data object in some state.

Data Object Input

Data Object Output

Input/output data required by a process when it starts.

Data Store

Data store: container for persisted data objects.

Order Management Process with Data Artifacts

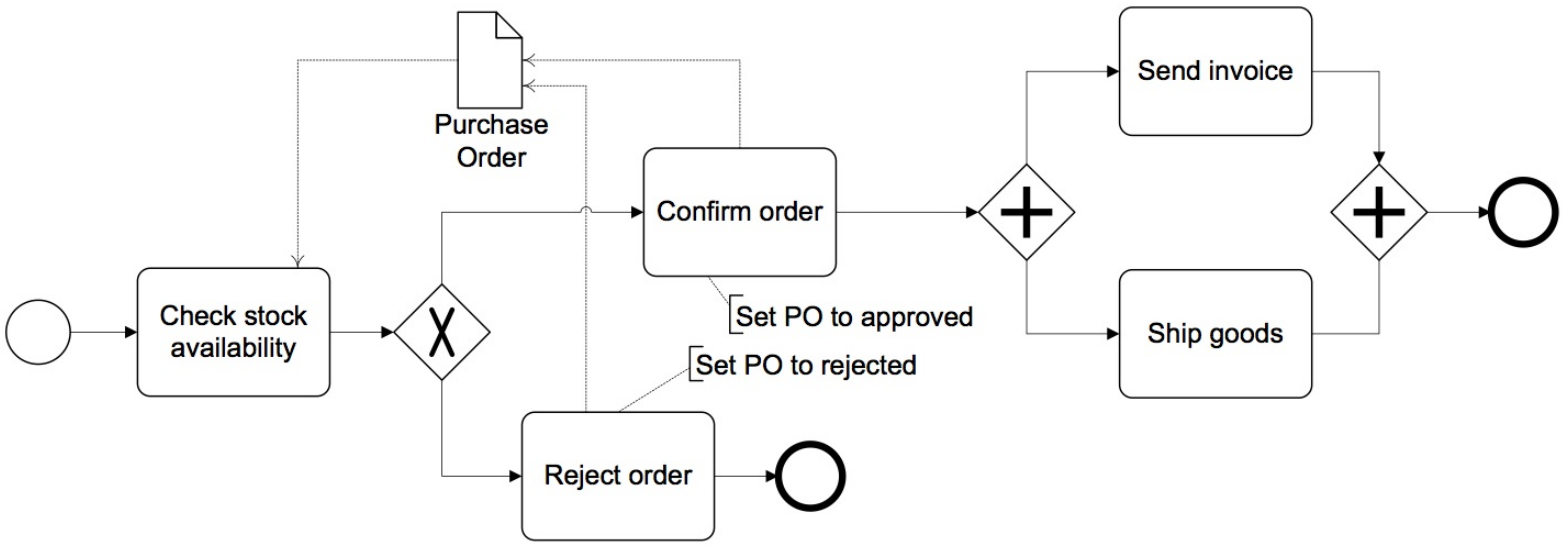
Example

Extend the Order Management Process by including the *purchase order* data object, used by the check stock activity and manipulated ("state" update) when the order is rejected/confirmed.

Order Management Process with Data Artifacts

Example

Extend the Order Management Process by including the *purchase order* data object, used by the check stock activity and manipulated ("state" update) when the order is rejected/confirmed.



It is also possible to use explicit state information.

An Interesting Example

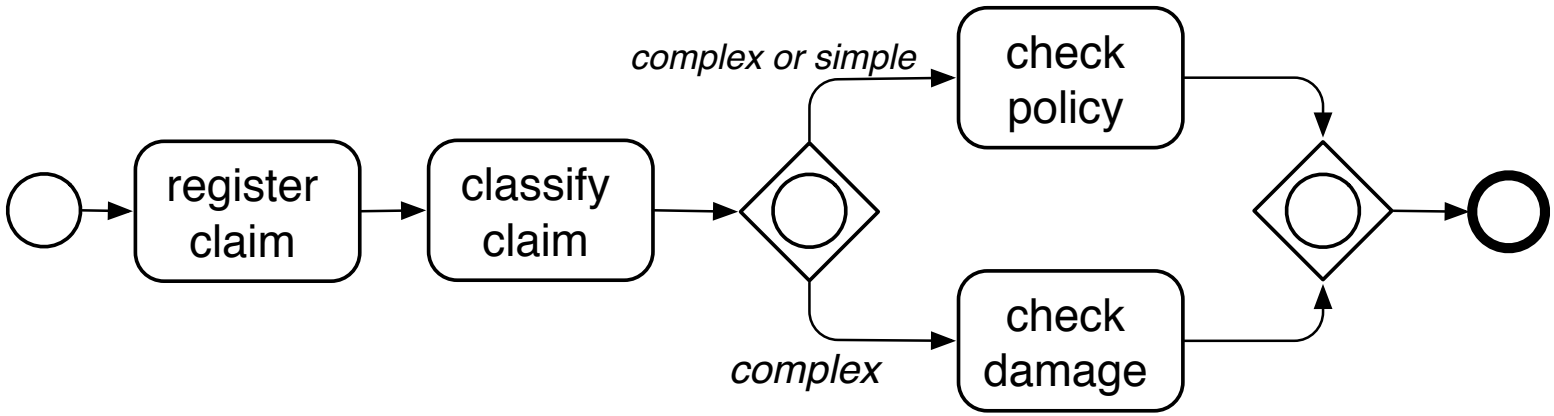
Claim handling

When a claim is received, it is registered. After registration, the claim is classified leading to two possible outcomes: simple or complex. If the claim is simple, the policy is checked. For complex claims, both the policy and the damage are checked independently.

- Can we model the example using the constructs seen so far?
- Can we model it in a compact way?

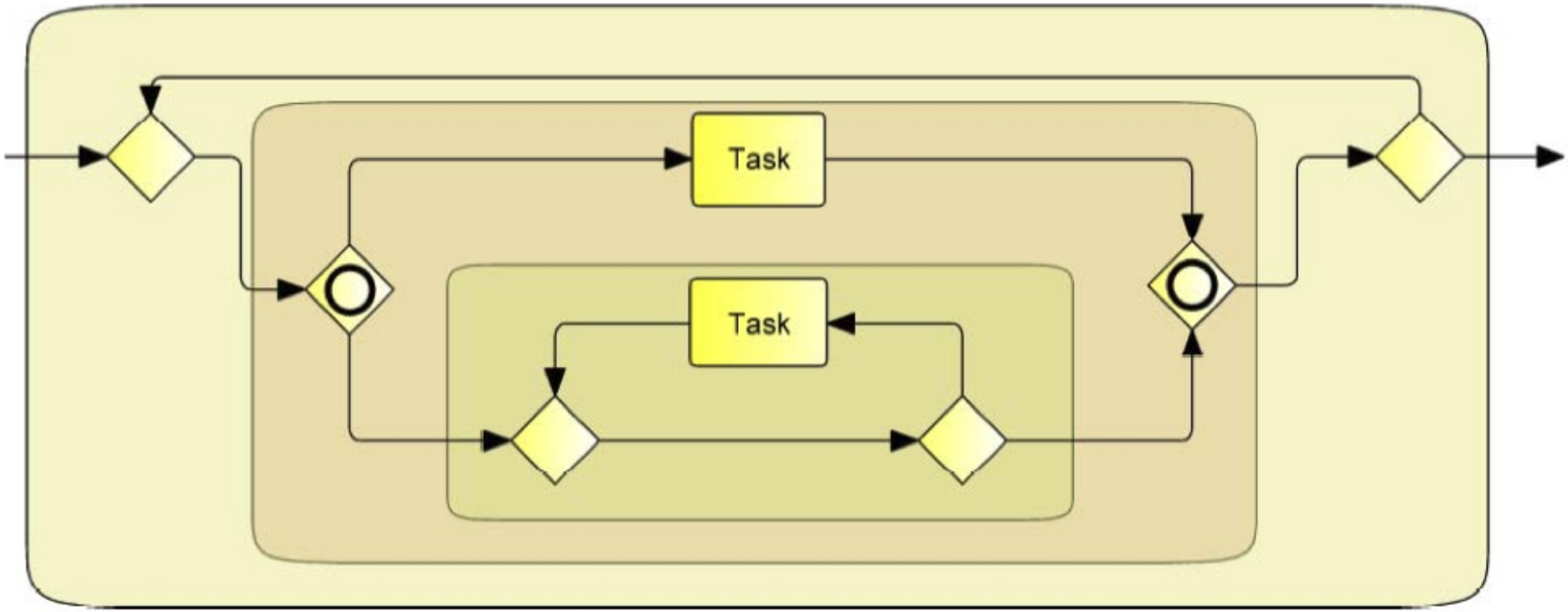
Inclusive Gateway

- Split: selects one or more paths depending on the (possibly overlapping) data conditions, or the user choice.
  - The selected outgoing branches are followed in parallel.
- Join: synchronizes the *activated* branches.
  - Non-local semantics.
  - Intuitive in the case of *block-structured* models, cryptic in the general case.
  - Can generate (apparently) paradoxical cases (see, e.g., vicious circles in EPCs). Many solutions provided in the literature.



Block-Structured Models - Intuition

Only apparent graph-based structure.



General Execution Semantics for Inclusive Gateway

Remember: the semantics is non local!

An Inclusive Gateway is activated if:

- **At least one** incoming Sequence Flow has *at least one* token **and**
- **For every** directed path formed by sequence flow that
  - starts with a Sequence Flow f of the diagram that has a token,
  - ends with an incoming Sequence Flow of the inclusive gateway that has *no token*,
  - does not visit the Inclusive Gateway

  then there is also a directed path formed by Sequence Flow that
  - starts with f,
  - ends with an incoming Sequence Flow of the inclusive gateway that has a token, and
  - does not visit the Inclusive Gateway.

Another Interesting Example

Is this model correct? How can we fix it?