

# Data and Process Modelling

## 8a. BPMN - Advanced Modelling

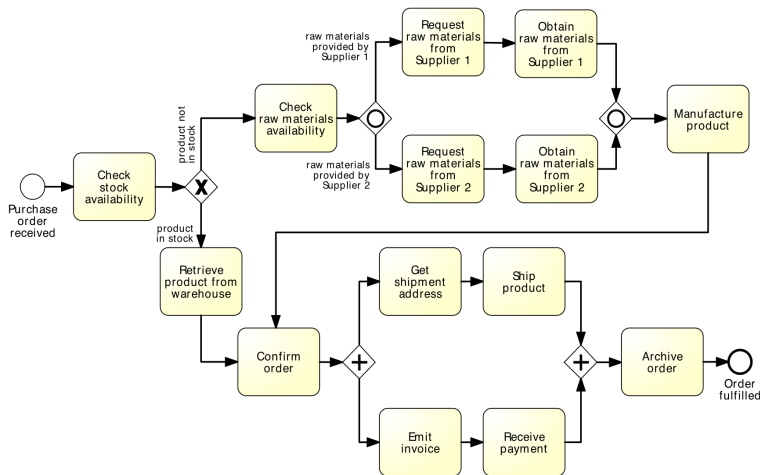
Marco Montali

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

A.Y. 2014/2015

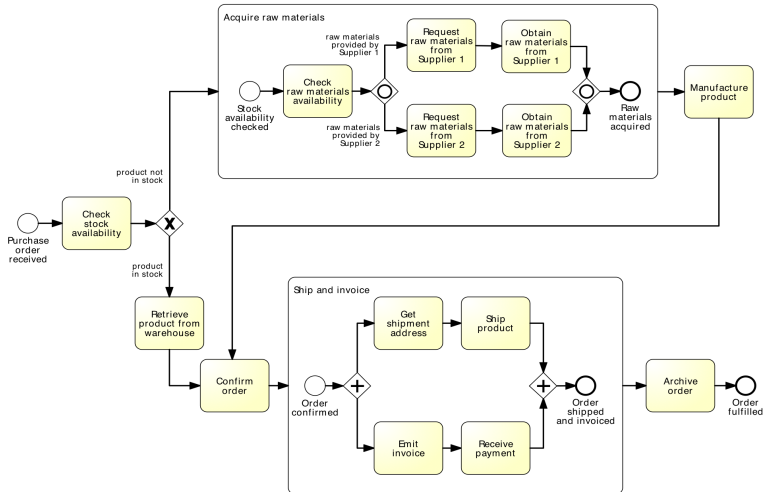


# The Need of Modularization



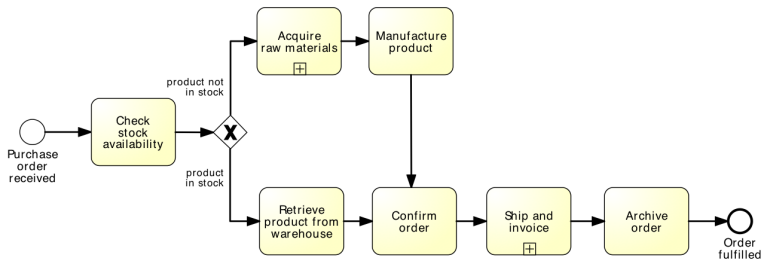
How to reduce the “complexity”?  
How to reuse parts of the process?

# BPMN Sub-Processes



Rounded rectangles represent generic **activities** (i.e., modules of work).  
Can be specialized in terms of tasks or sub-processes.

# BPMN Collapsed Sub-Processes



Collapsed view to hide the internal definition of (complex) activities.

# Activities and Control-Flow

- Normal activity without incoming sequence flow: instantiated when the process is instantiated (**implicit start**).
- Multiple incoming sequence flows: multi-merge semantics.
- Normal activity without outgoing sequence flow: marks the end of the path. If it is completed and no other parallel branch is active, the process is also completed (**implicit termination**).
- Multiple outgoing sequence flows: and-split semantics.
- Message flow rules: 0 or more incoming/outgoing message flows with separate pools involved.

# Activity Decorators



Basic task.



Compensation task.



Loop task (looping information attached to the activity).



Multi-instance task with parallel composition (expression attached to the activity to calculate the number of instances).

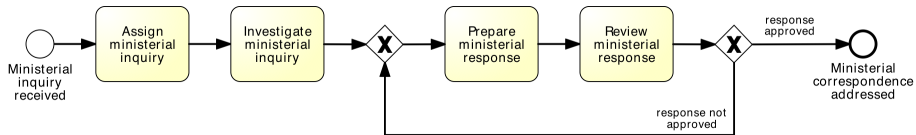


Multi-instance task with sequential composition.

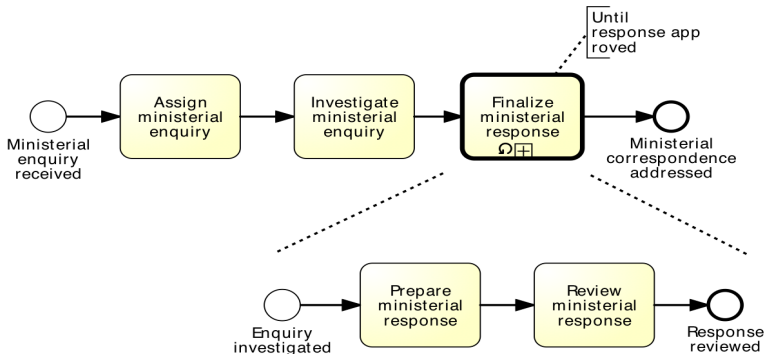
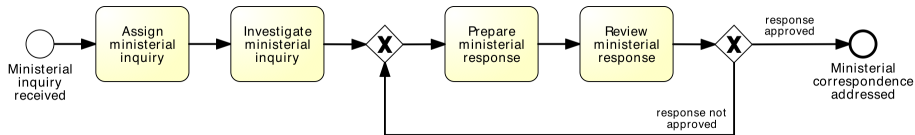
---

For sub-processes only: **ad-hoc** (tilde marker) - flexible execution of the inner activities, without a complete specification of the process.

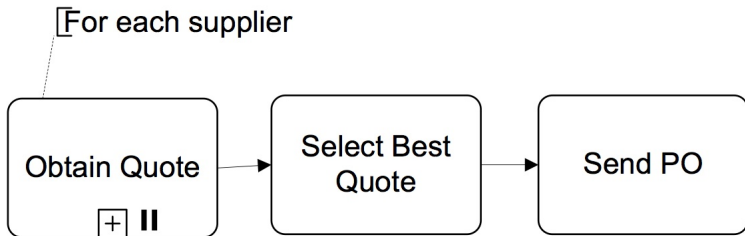
# Example: Loops



# Example: Loops



## Example: Multi-Instances



# Event





























Something that occurs during the execution of the process. Represented with a circle, whose decorations determine the specific semantics.

- **Start event** (thin line): indicates where the process starts.
- **End event** (thick line): indicates where a path of the process ends.
- **Intermediate event** (double line): indicates that something happens during the execution of the process.






























Two modalities:

- **Catch** a trigger.
- **Throw** a result (explicitly or implicitly), possibly caught by another event. Strategies:
  - ▶ publication (e.g., message);
  - ▶ direct resolution;
  - ▶ propagation (to the innermost enclosing scope instance with an event able to catch the trigger);
  - ▶ compensation (triggers a compensation handler);
  - ▶ cancellation (terminates all running activities in a subprocess, and compensates all the completed ones).

# Events

	Start			Intermediate			End
	Standard	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing
<b>None:</b> Untyped events, indicate start point, state changes or final states.							
<b>Message:</b> Receiving and sending messages.							
<b>Timer:</b> Cyclic timer events, points in time, time spans or timeouts.							
<b>Escalation:</b> Escalating to an higher level of responsibility.							
<b>Conditional:</b> Reacting to changed business conditions or integrating business rules.							
<b>Link:</b> Off-page connectors. Two corresponding link events equal a sequence flow.							

# Events

	Start			Intermediate			End
	Standard	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing
<b>Error:</b> Catching or throwing named errors.							
<b>Cancel:</b> Reacting to cancelled transactions or triggering cancellation.							
<b>Compensation:</b> Handling or triggering compensation.							
<b>Signal:</b> Signalling across different processes. A signal thrown can be caught multiple times.							
<b>Multiple:</b> Catching one out of a set of events. Throwing all events defined							
<b>Parallel Multiple:</b> Catching all out of a set of parallel events.							
<b>Terminate:</b> Triggering the immediate termination of a process.							

# Top-Level Start Events

Determine the creation of a new process instance (top-level).

- **None** (no symbol): generic start.
- **Message** (letter): start upon receiving a message.
- **Timer** (clock): time expression implicitly triggering a start.
- **Conditional** (written paper): implicit start when the attached business rule evaluates to true.
- **Signal** (triangle): start upon receiving a broadcasted signal.
- **Multiple** (pentagon): multiple possible starts.
- **Parallel multiple** (+): start determined by the presence of multiple conditions.

The generic start is the only one usable in subprocesses.

Start events can be used also for *event sub-processes*. Two possibilities:

- **Interrupting** (solid line): containing process is interrupted.
- **Non-interrupting** (dashed line): containing process continues.

# End Event Types

Determine the termination of a path in the process instance (token). The instance terminates only when all tokens have been consumed

- **None** (none).
- **Message** (black letter): sends a message before terminating.
- **Error** (black lightning): generates a named error, terminating all active threads in the process. The error is caught by a corresponding catch event in the nearest enclosing parent activity, or its treatment is unspecified.
- **Escalation** (up arrow): like error, but active threads continue.
- **Cancel** (X): used for transactions, triggers a cancelation and alerts all the entities involved in the transaction.
- **Compensation** (rewind): indicates the need for compensation of a named visible activity or all visible activities (compensated in reversed order). For visibility rules check the documentation (p. 248).
- **Signal** (black triangle): broadcasting of a signal.
- **Terminate** (black circle): forces a sudden termination of all threads.
- **Multiple** (black pentagon): multiple consequences.

# Intermediate Events

## Two versions

- **Throw**: see end events (immediate execution).
- **Catch**: see start events (wait semantics).

Catch events can be attached to the boundary of an activity for exception and compensation handling. Possibilities:

- **Interrupting**: error, cancel.
- **Interrupting/non-interrupting**: message, timer, escalation, conditional, signal, multiple, parallel multiple.
- **Compensation**: executed only if the activity instance is already completed.

Link events (with arrows as icon) can be used as **off-page connectors** for large process models.

# Working with Events

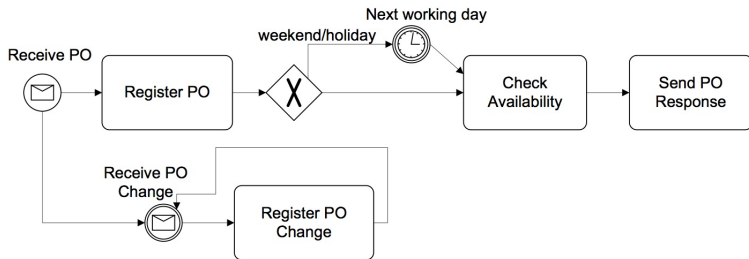
## Purchase Order Handling

A PO handling process starts when a PO is received. The PO is first registered. If the current date is not a working day, the process waits until the following working day before proceeding. Otherwise, an availability check is performed and a “PO response” is sent back to the customer. Anytime during the process, the customer may send a “PO change request”. When such a request is received, it is just registered, without further action.

# Working with Events

## Purchase Order Handling

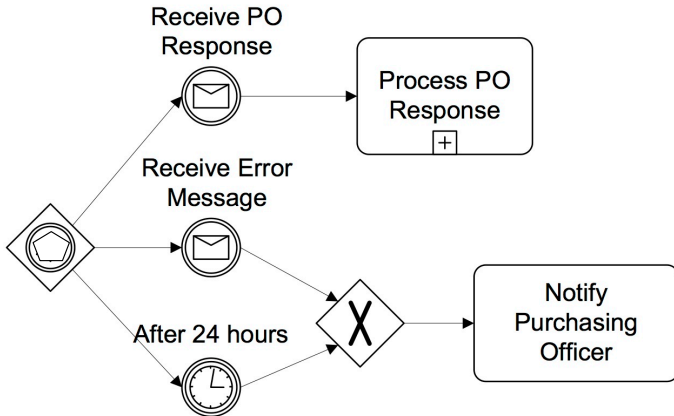
A PO handling process starts when a PO is received. The PO is first registered. If the current date is not a working day, the process waits until the following working day before proceeding. Otherwise, an availability check is performed and a “PO response” is sent back to the customer. Anytime during the process, the customer may send a “PO change request”. When such a request is received, it is just registered, without further action.



# Event-Based Gateway

How to model choices that are not under the orchestrator's control?

- Typical problem: take a path depending on the first incoming event among possible events.
- A race condition determines the winner.
- The **event-based gateway** exactly captures this issue.

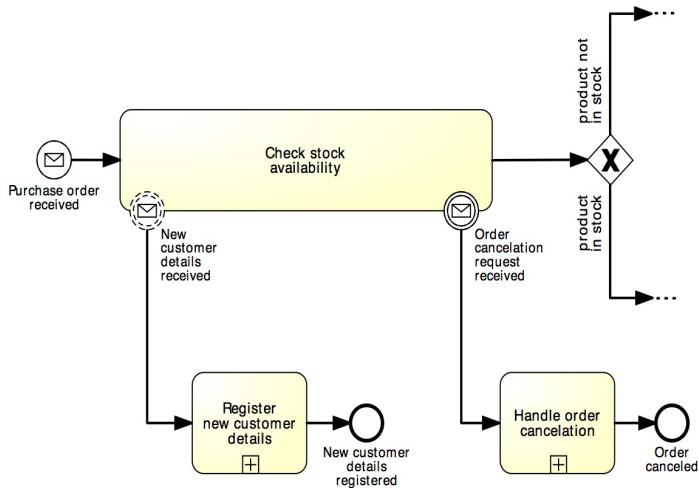


# Reaction to Events

How to specify reactions to internally-generated or incoming events?

- **Event sub-process** (dashed boundary): part of the process triggered by an event, interrupting or non-interrupting the normal flow.
- **Boundary events**: events attached to an activity, determining reactions when corresponding events are triggered inside the corresponding sub-process. Again: interrupting vs non-interrupting behavior.

# Boundary Events - Example

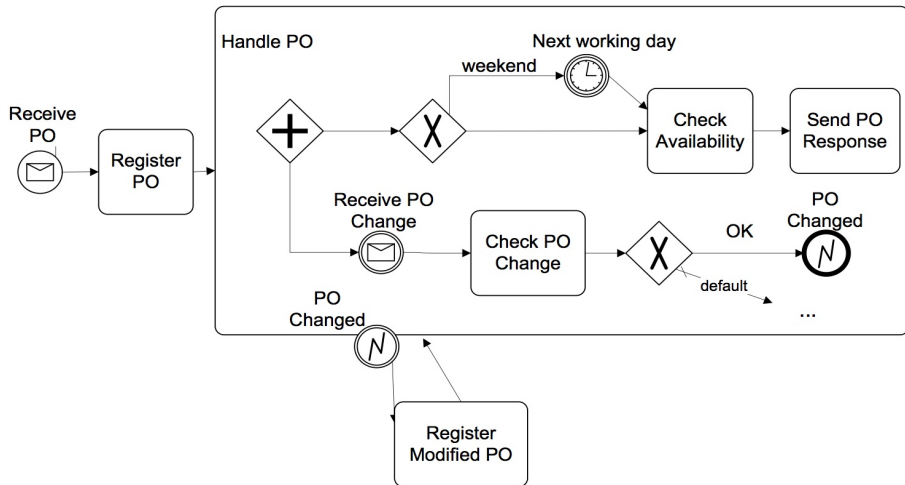


# Exception Handling

Exceptions are events that deviate a process from its normal course.

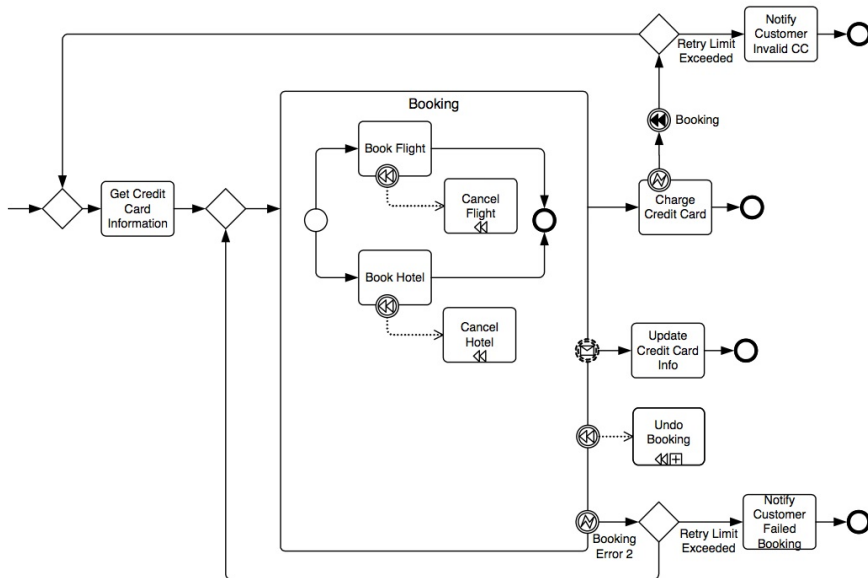
- Handling exceptions often involves stopping a sub-process and performing a special activity.
- Achieved using two event nodes:
  - ▶ An **end error event** that stops the enclosing subprocess execution.
  - ▶ An **intermediate error event** attached to the enclosing subprocess: this is where the process execution will continue after the error.

# Exception Handling - Example



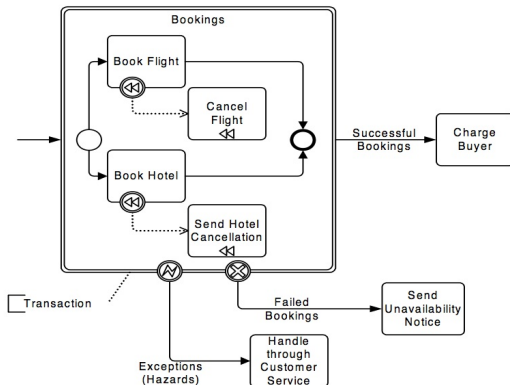
# Boundary Events with Compensation

From the BPMN official documentation, page 279.



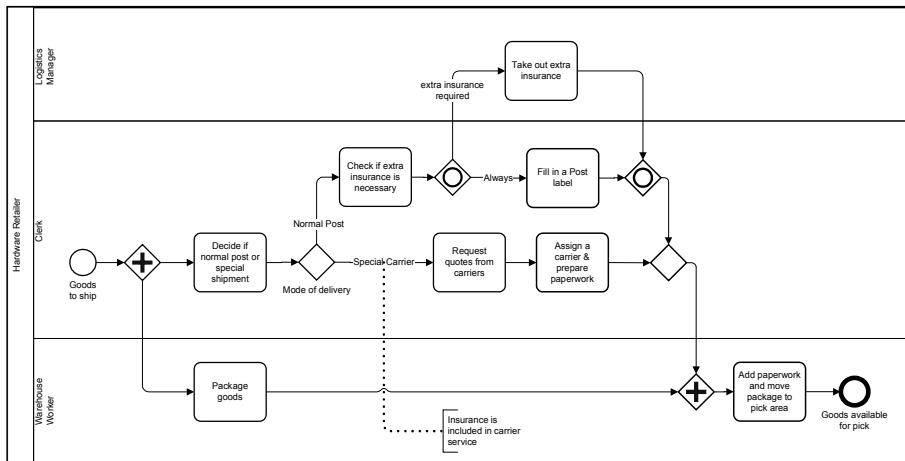
# BPMN Transactions

From the BPMN official documentation, page 179.



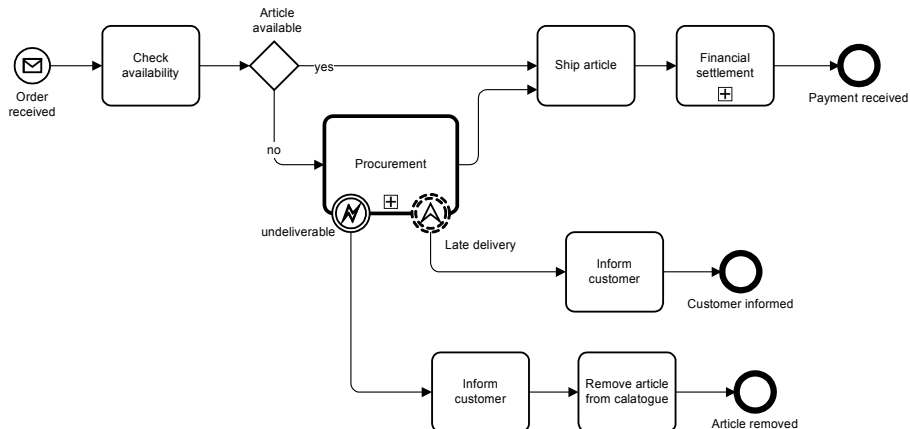
**Cancellation** triggers compensation for the already completed activities (process-oriented way of modeling a “roll-back”).

# BPMN - Hardware Retailer Example



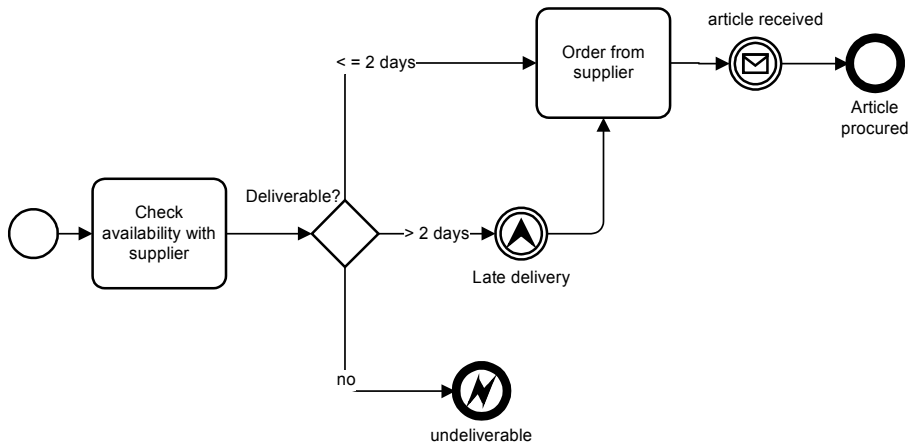
From *BPMN 2.0 by Example* - <http://www.bpmn.org/>

# BPMN - Order Example (Main Process)



From *BPMN 2.0 by Example* - <http://www.bpmn.org/>

# BPMN - Order Example (Procurement Sub-Process)



From *BPMN 2.0 by Example* - <http://www.bpmn.org/>

# BPMN - Pizza Example

