ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Business Process Management

Procedural, Closed Languages

Federico Chesani
Department of
Computer Science and Engineering
University of Bologna

Credits and sources

These slides have been partly inspired by:

- Work of Prof. Marco Montali, Free University of Bozen http://www.inf.unibz.it/~montali/
- Mathias Weske, *Business Process Management,* Second Edition, Springer, 2012

- Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, Alistair P. Barros: Workflow Patterns. Distributed and Parallel Databases 14(1): 5-51 (2003)
- Russell, N.C.; ter Hofstede, A.H.M.; van der Aalst, W.M.P.; Mulyar, N.A. Workflow control-flow patterns : a revised view. Tech Rep. BPM Center BPM-06-22 https://pure.tue.nl/ws/files/2456784/710796038377571.pdf
- Wil M. P. van der Aalst, Arthur H. M. ter Hofstede. YAWL: yet another workflow language. Inf. Syst. 30(4): 245-275 (2005) https://www.sciencedirect.com/science/article/pii/S0306437904000304?via%3Dihub

- https://academic.signavio.com/p/login
- https://bpm-book.com/BpmBook/WebHome

- https://www.draw.io/

Petri Nets

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Petri Nets

- Proposed by Carl Adam Petri in his Ph.D. Thesis, to generalize automata theory with concurrency, in 1962 (in 1939, conceived for chemical processes)
- Abstract Formalism for representing processes, and processes languages in general.
- Graphical representation with an equivalent mathematical formalization

- Formal: semantics is *well-defined* and not ambiguos
- Abstract: no info about the execution environment

- Able to model dynamic systems with a static structure.

Petri Nets – basics concepts for the Static structure

- Places: indicated with an empty circle, that (possibly) will contain (at runtime) a token
- Transitions: rectangles, usually with a label. They have *input* and *output* places.
- Connections: directed arcs that connect places with transitions, and transitions with places.

- More formally, petri nets are *bipartite graphs:* eash arc connects a place and a transition, or a transition and a place.

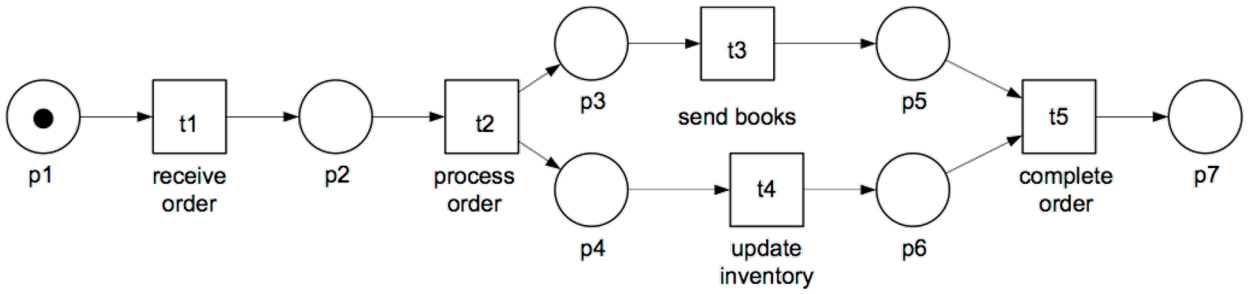


Fig. 4.26. Sample Petri net representing single process instance

M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets – basics concepts for the Dynamic structure

- Tokens: small, black filled circles, residing on places.
- Tokens change their position according to the static structure, and according to a set of fixed firing rules.
- The distribution of tokens in a petri net represent a current state of a system/process.

- Firing of a transition: it can fire if it is enabled... A transition is enabled if there is a token in each of its input places

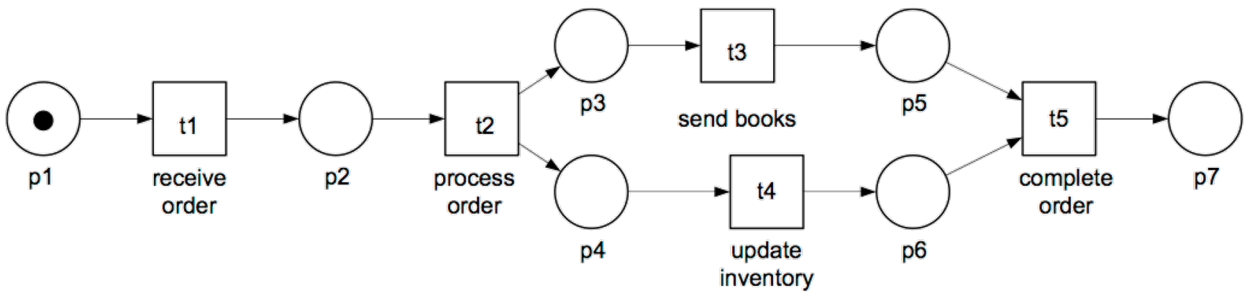


Fig. 4.26. Sample Petri net representing single process instance

M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets – basics concepts for the Dynamic structure

- Firing of a transition: it can fire if it is enabled... A transition is enabled if there is a token in each of its input places
- When a transition fires:
    - One token is removed from each input place
    - One token is added to each output place
- The movement of tokens, according to the structure and the firing rules, is called token play.

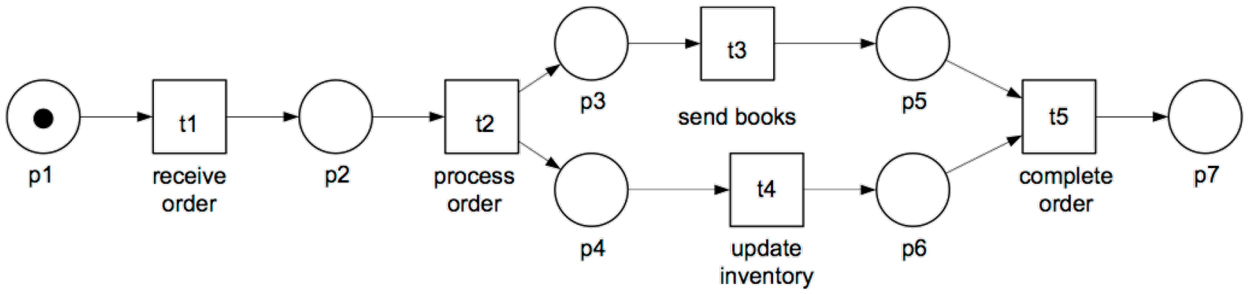


Fig. 4.26. Sample Petri net representing single process instance

M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets – basics concepts for the Dynamic structure

- Transitions change the state of a net... They could represent active components (events, activity executions, etc)
- Places are passive: they could represent mediums, buffers, states, conditions, etc.
- Tokens move around:
    - All together they represent the process state
    - Each token might represent physical or information objects

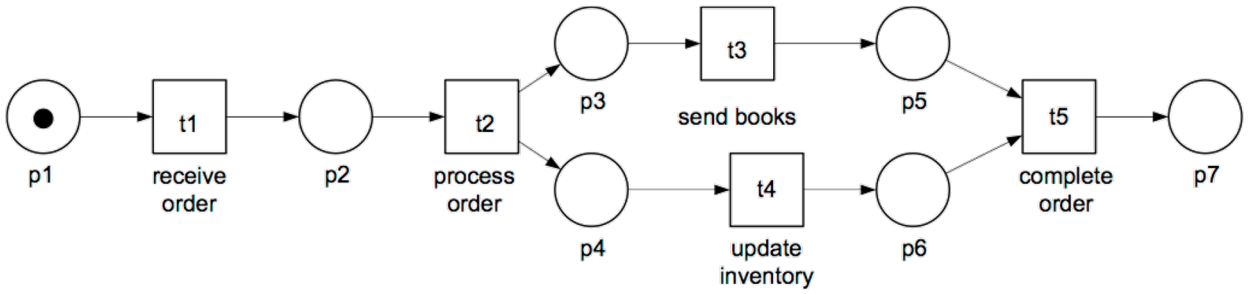


Fig. 4.26. Sample Petri net representing single process instance

M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets and BPM

- Petri Net → Process Model
- Transitions → Activity Models
- Instances → Tokens

- Firing of a transition → activity execution
- Process instance → at least one token (there might be more than one, however); (the number of token might vary along the same process instance execution)

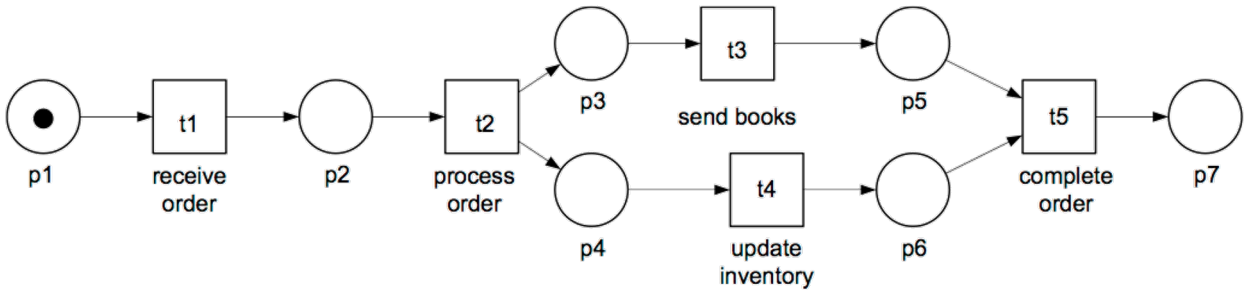


Fig. 4.26. Sample Petri net representing single process instance

M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets and BPM – Problems?

- In classical Petri Net tokens cannot be distinguished one from each other…
- … hence, they can host a single process instance at a time…

- … otherwise, confusion might happen…. how many instances in this process?

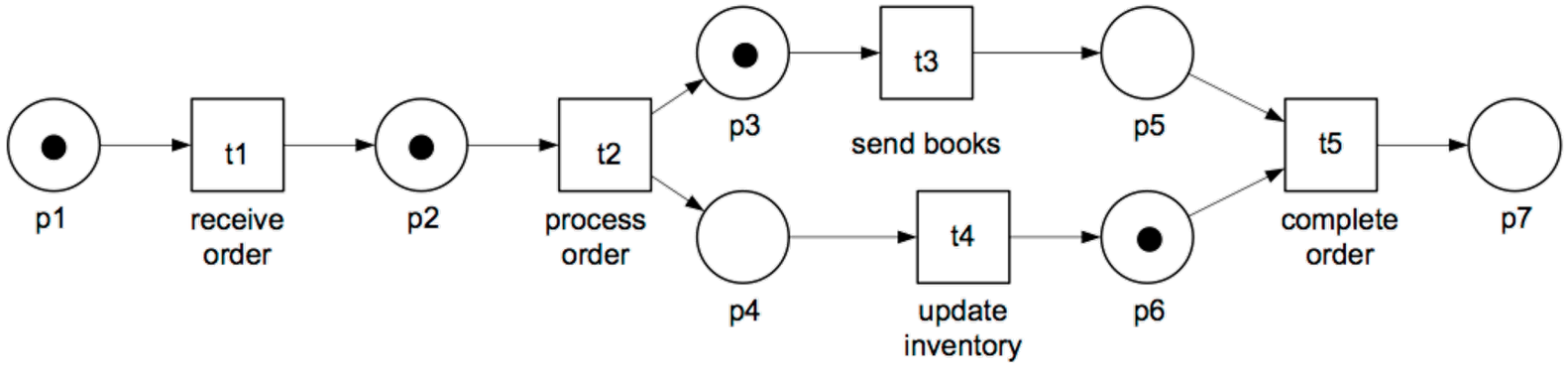
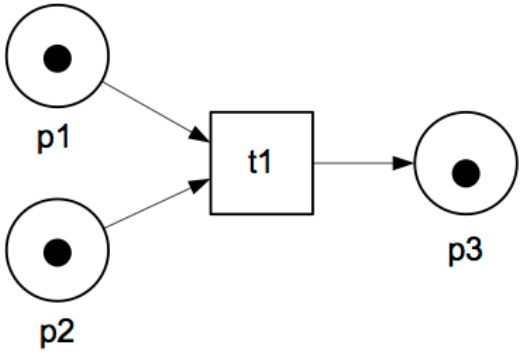


Fig. 4.27. Sample Petri net representing multiple process instances

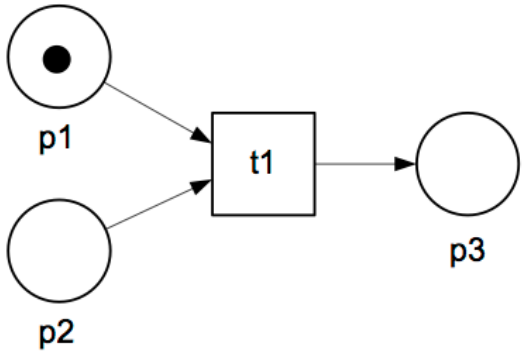
M. Weske: Business Process Management, Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets – Condition Event Nets

- At each moment, a place can have at most one token
- This holds also for output places: hence a transition is enabled only if its output places are free of tokens.



(c) t1 not enabled, since output condition is met



(d) t1 not enabled since not all input conditions are met

Petri Nets – Place Transition Nets

- A place can host one or more token
- Moreover, arcs have a weight (a natural), whose meaning must be interpreted as the number of tokens to be consumed or to be produced

- … still, tokens are indistinguishable… how many process instances below? What if t5 triggers?

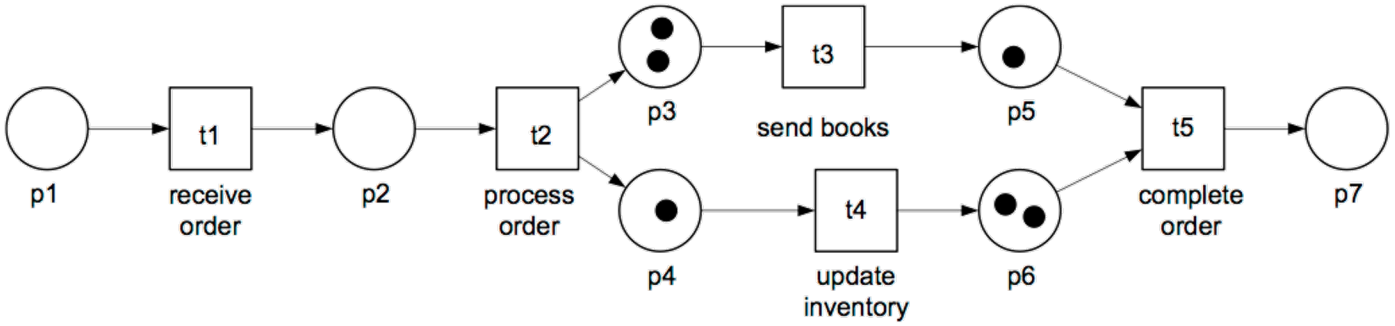


Fig. 4.29. Place transition net with multiple process instances

M. Weske: Business Process Management, Springer-Verlag Berlin Heidelberg 2012, 2007

Petri Nets – Coloured Petri Nets

- Token might have colours, i.e. they have values attached to them
- Values are typed (colour sets)
    - they have a domain
    - they have a set of allowed operations
- Transitions now are enabled also on the base of the values of the tokens, by means of arc conditions
    - pre-conditions on the enabling of a transition
    - post-conditions on the effects of a transition

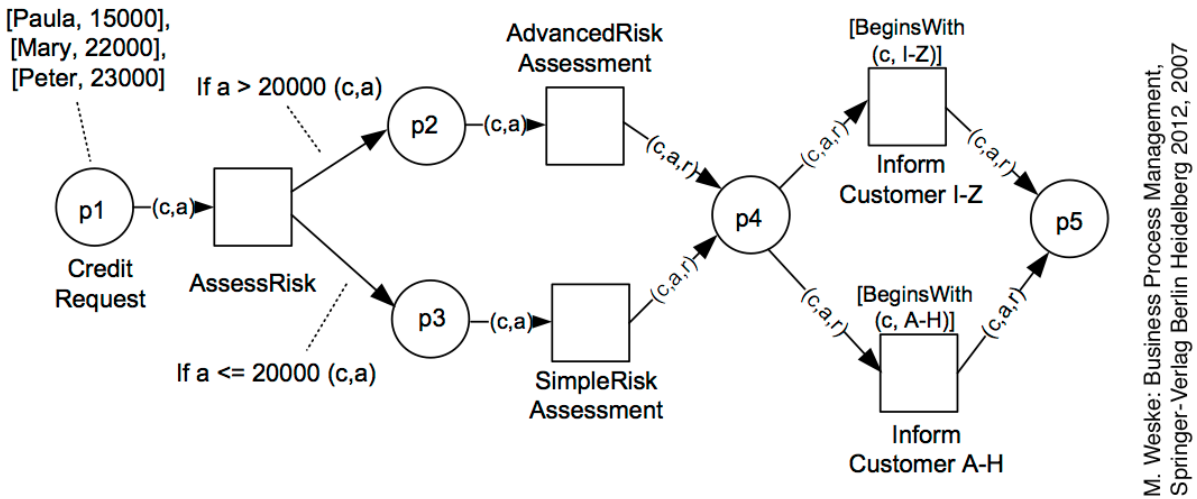


Fig. 4.30. Sample coloured Petri net

Summing up – advantages of Petri Nets

- Formal Semantics
- Graphical Representation and an immediate mapping into a mathematical representation
- Transition rules are mapped into (simple) mathematical functions as well
- Analysis of Process Properties is possible

- Tool Independence (from industrial viewpoint)

Workflow Nets

Workflow Nets

- They are Petri Nets, where:
  - Places represent conditions
  - Transitions represent activity instances
  - Tokens represent process instances
  - Tokens are **coloured**, but data and expression are not indicated in the graphical representation
  - **Hierarchical structuring** is fully supported

- Moreover, there is a **structural condition**:
  - There is only one distinguished place *i* (*initial place)* that has no incoming edges
  - There is only one distinguished place *o* (*final place)* that has no outgoing edges
  - Every place and every transition is located on a path from the initial place to the final place

Workflow Nets – what about control flow patterns?

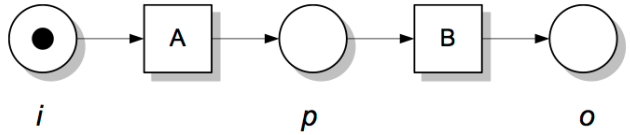


Fig. 4.42. Sequence pattern in workflow net

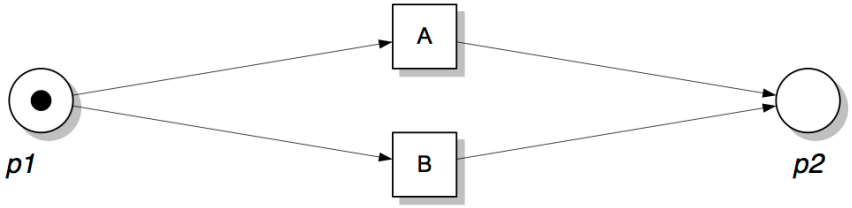


Fig. 4.44. *Implicit exclusive or split* also known as *deferred choice*

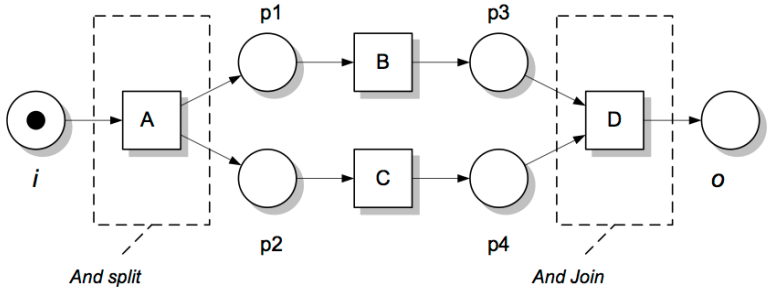


Fig. 4.43. *And split* and *and join* patterns in workflow nets

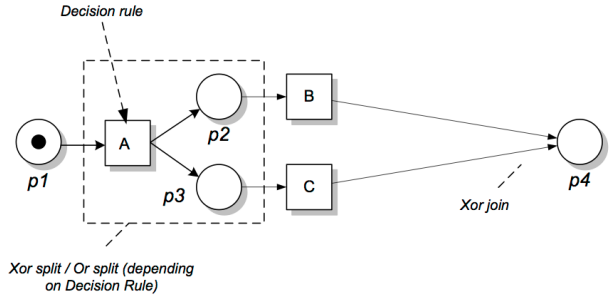


Fig. 4.45. Decision rule based split, can realize *or split*, *exclusive or split*, and *and split*

Workflow Nets – control flow patterns and decision criteria

- The exclusive or semantics is usually obtained through the definition of an expression criteria (a decision rule)
- Decision rules are defined by the process modeller
- It is up to the designer that:
  - always at most one output branch is chosen
  - always at least one output branch is chosen

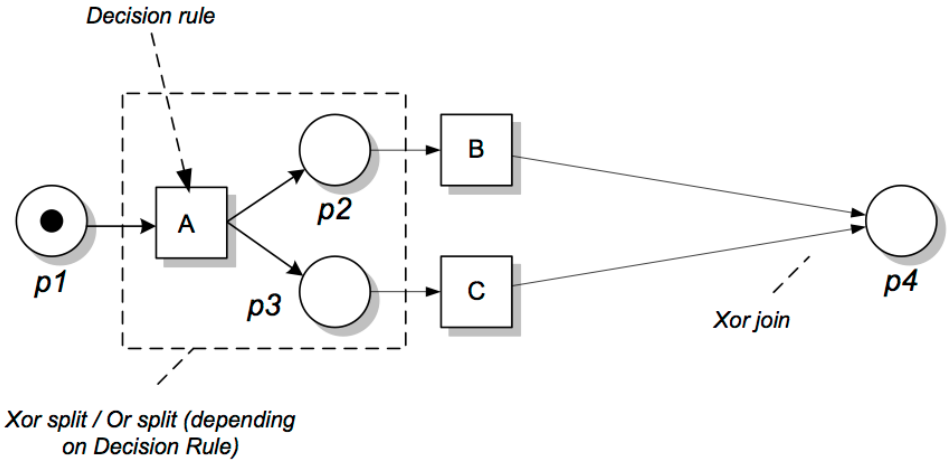


Fig. 4.45. Decision rule based split, can realize *or split*, *exclusive or split*, and *and split*

- What happens if the modeller writes a decision rules that does not enable any branch?
- How to avoid this, and avoid deadlocks?

Workflow Nets – syntactic sugar

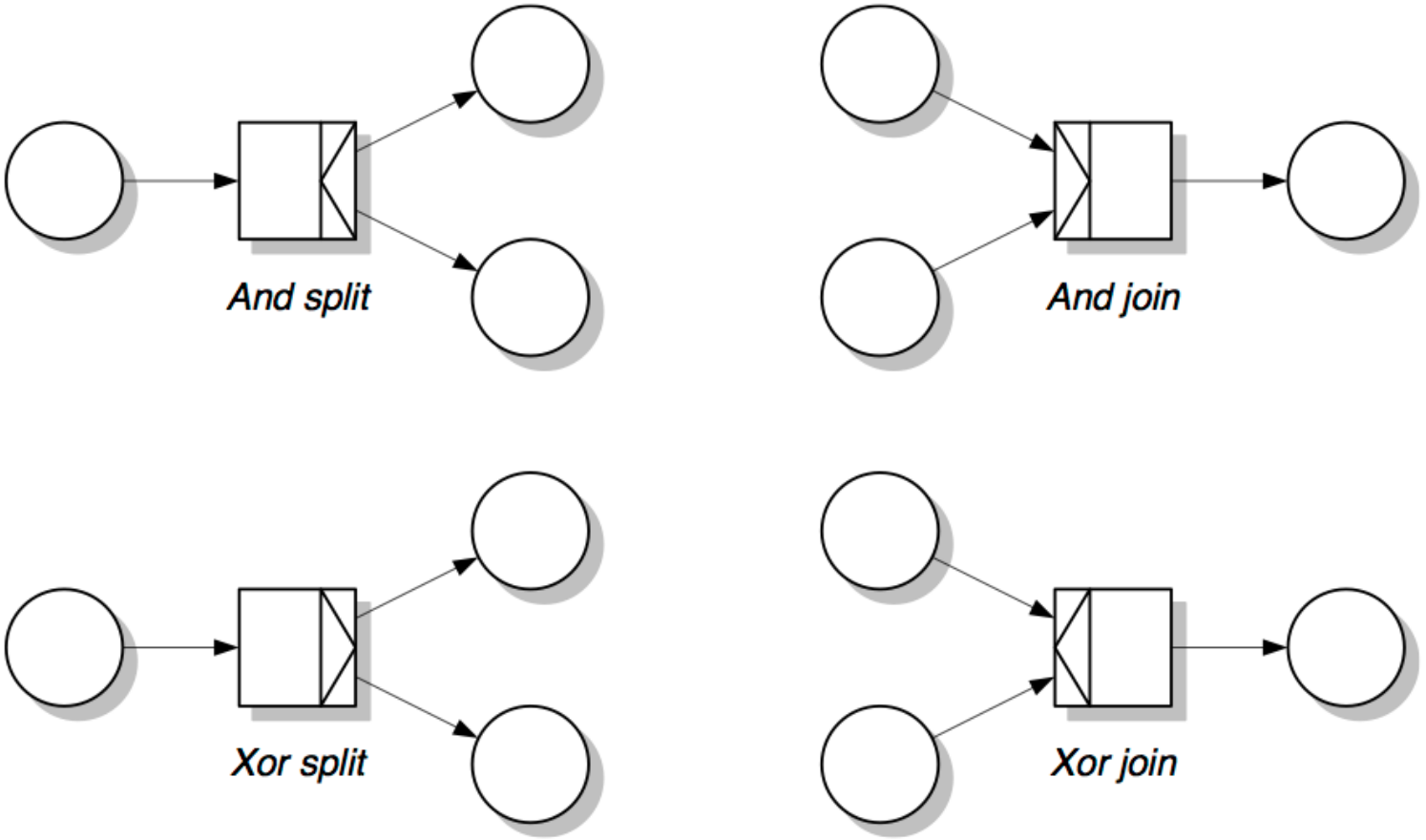


Fig. 4.46. Syntactic sugaring of transitions in workflow nets

Workflow Nets – the environment represented through triggers

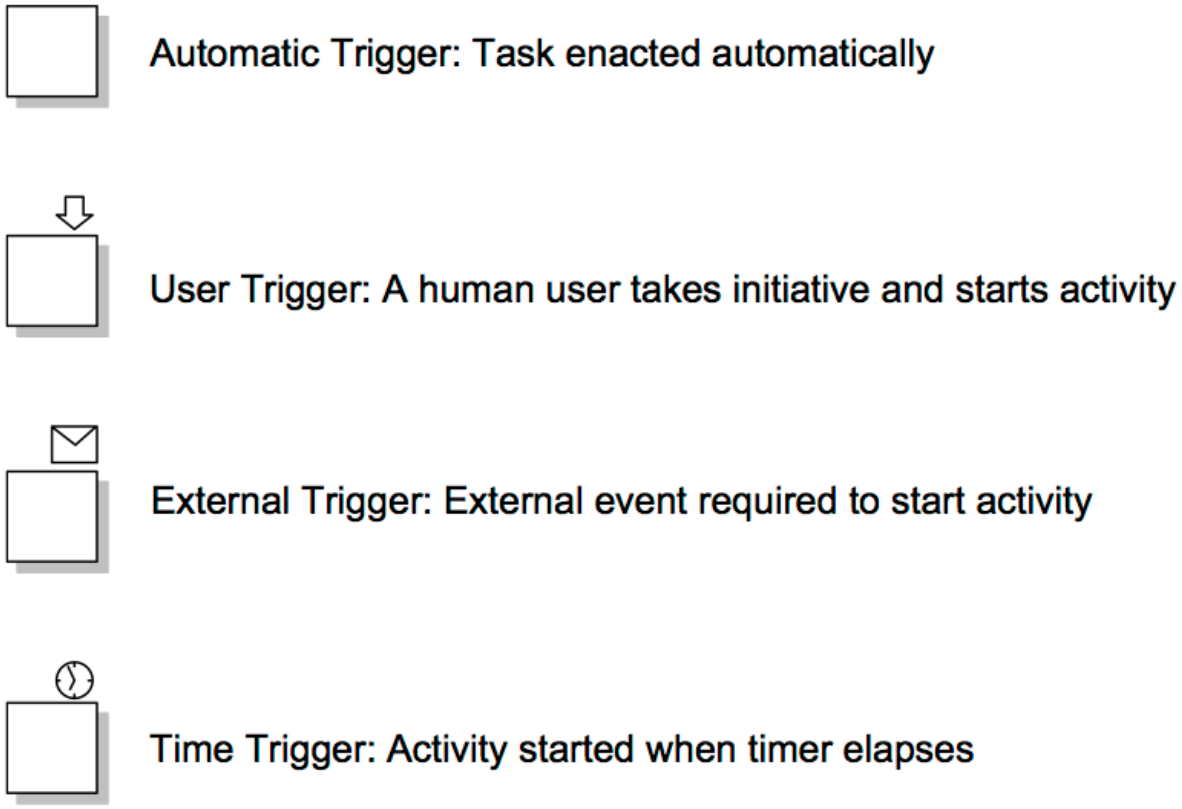


Fig. 4.47. Triggers in workflow nets

- Problem: how to formally represent triggers?
- If done by adding a place and an ingoing arc into the transition, the workflow net violates the structural constraint!!!

Workflow Nets – an example

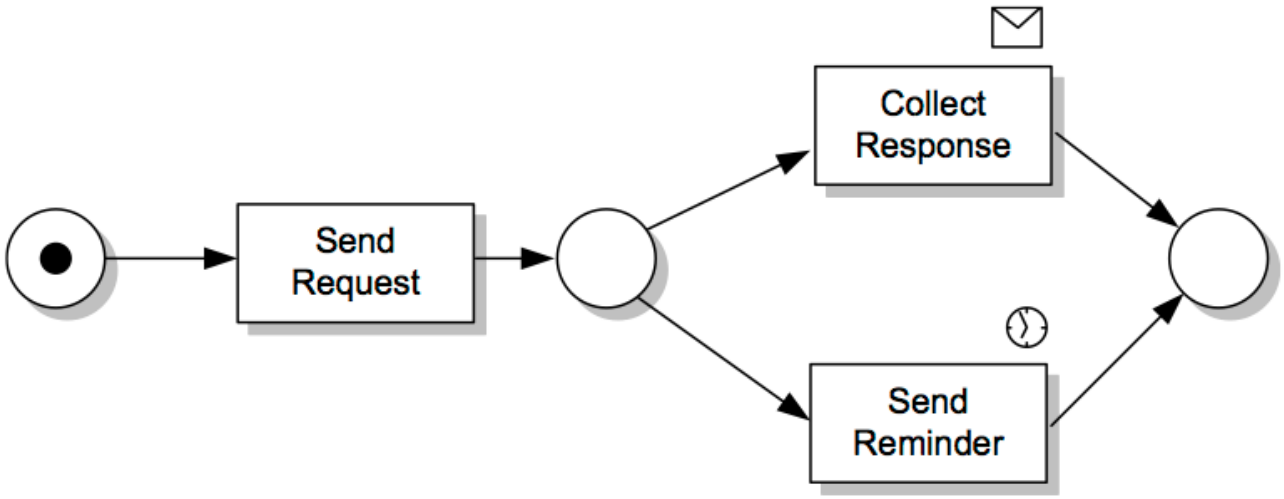
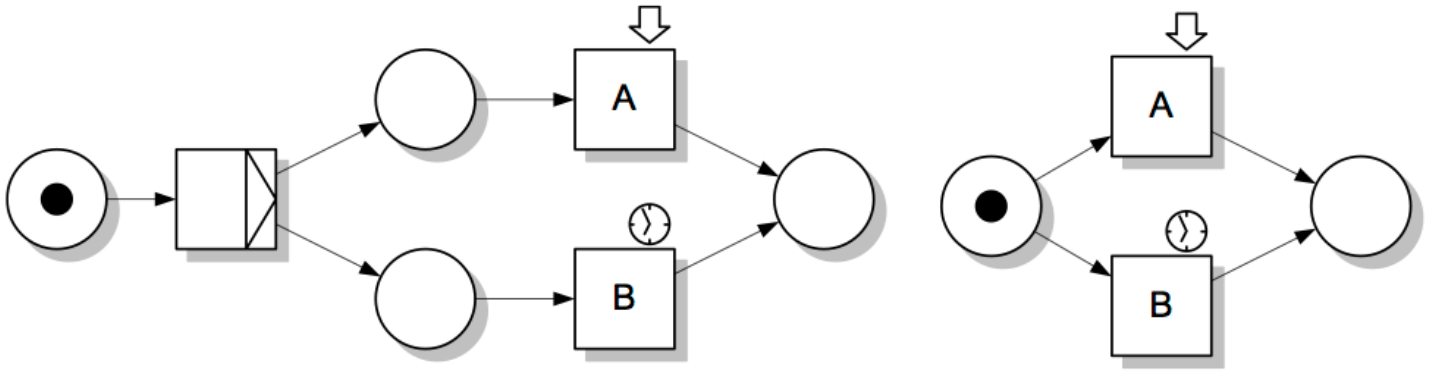


Fig. 4.48. Sample workflow net with external trigger and time trigger

Workflow Nets – the difference between explicit and implicit xor split



(a) *Explicit xor split* does not enable A and B concurrently

(b) *Implicit xor split* enables A and B concurrently

Fig. 4.49. Sample workflow nets illustrate the difference between *explicit xor split* (a) and *implicit xor split* (b)

Workflow Nets – final considerations

- Tokens need to be coloured, otherwise no way to distinguish different process instances
    - Tokens need to include at least a process instance identifier
    - As a consequence, enabling conditions are based also on the colour of the token
- Data is handled in an abstract way
- Decision criteria are supported, but only implicitly: no way to express them, no way to reason on them and on their properties
- Transitions fire instantly (the evolution of the net is instantaneous). Real activity instances instead have a duration
    - No events can occur during activity execution, since transitions take zero time…

YAWL: Yet Another Workflow Language

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

YAWL

YAWL: Yet Another Workflow Language
http://www.yawlfoundation.org/

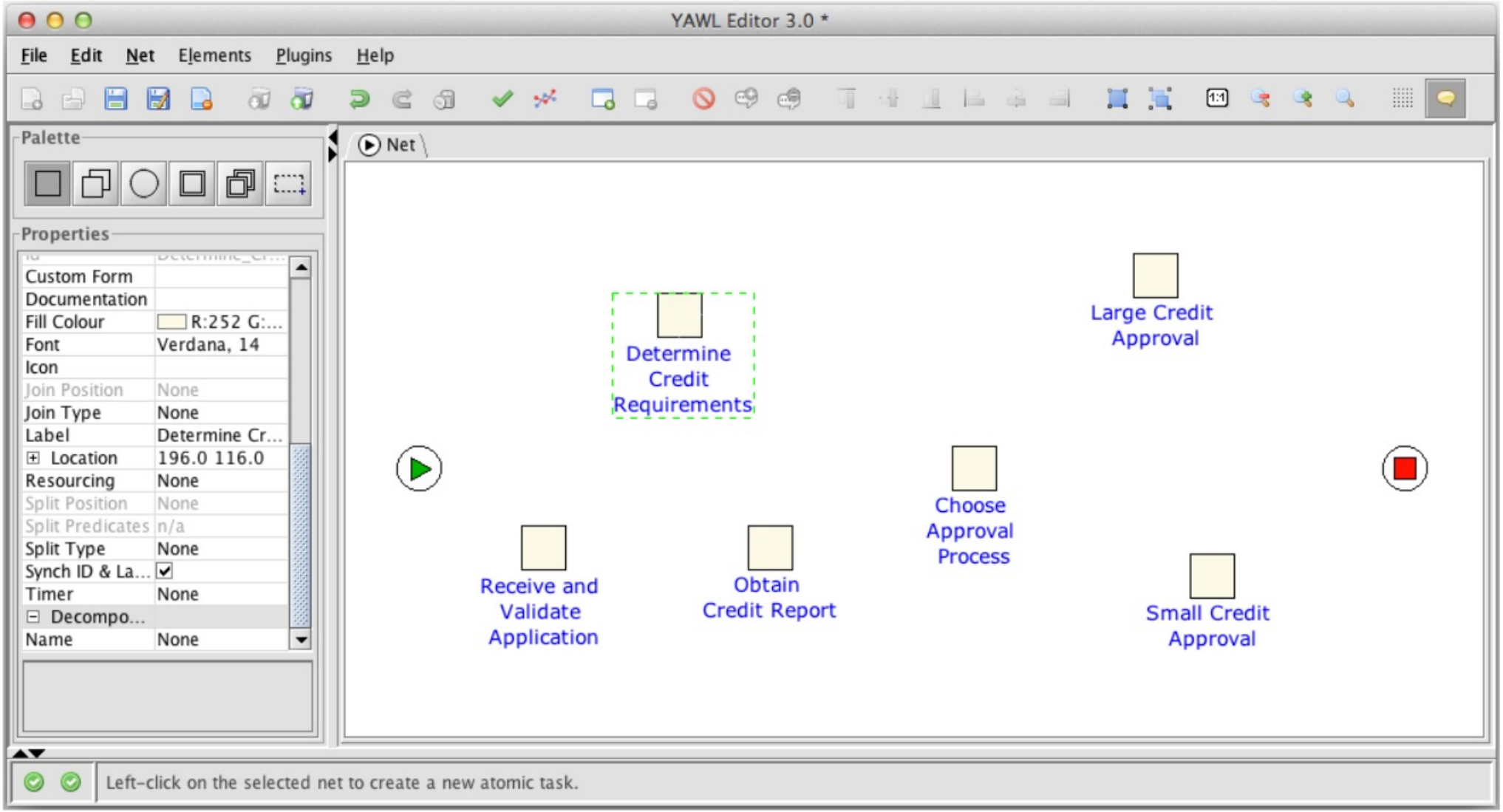
Supports:

- Control-flow dependencies
- Resourcing requirements
- Data support

- Nice (and free) editor
- Based on an execution semantics (state transition system)
- But also formal (petri-net based) verifiable semantics (but also WofYAWL)
- Supports (through a plug-in) to the Declare declarative approach (later on this)

- Workflow Engine fully available

YAWL



Yet Another … Why?

- Lack of support for a number of control flow patterns
  - Multiple instances of specific activities
  - Or split / join
  - Nonlocal firing behaviour: for example, the cancellation pattern would affect different parts/paths of a petri net

- Direct arcs between transitions (when places are anonymous conditions)
- Explicit split and join behaviour attached to the transitions
- Support to nonlocal behaviour
- Handling of multiple instance activities

- Wil M. P. van der Aalst, Arthur H. M. ter Hofstede. YAWL: yet another workflow language. Inf. Syst. 30(4): 245-275 (2005) https://www.sciencedirect.com/science/article/pii/S0306437904000304?via%3Dihub
- Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, Michael Adams, Nick Russell. Modern Business Process Automation - YAWL and its Support Environment. Springer 2010, ISBN 978-3-642-03120-5 https://www.springer.com/gp/book/9783642031205

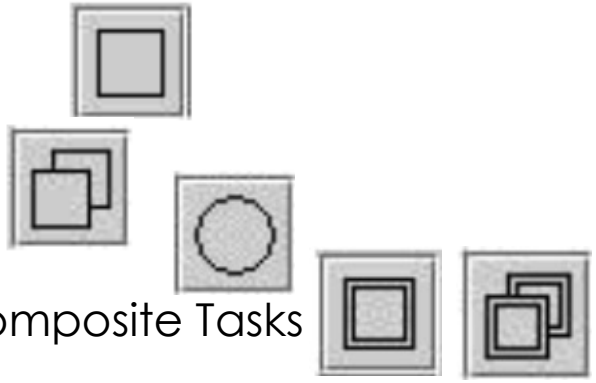
YAWL Nets

- YAWL is formally based on YAWL Nets, i.e. Workflow nets with these extensions:
    - Arcs directly connect transitions
    - Split / join flavours are directly indicated through specific symbols
    - For each task, a cancellation region can be defined
    - Tokens reside on transitions for the activity instance duration
    - Tasks can generate multiple instances (through definition of min, max, threshold, dynamic/static creation)

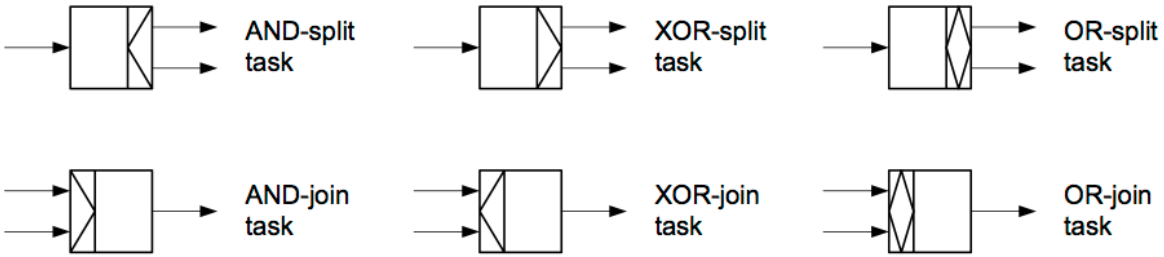
YAWL

- Two special symbols for start and end of a process instance

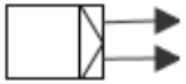
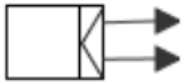
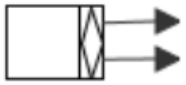
- Tasks/activities represented as simple blocks
  - Atomic Tasks
  - Multiple Instance Atomic Tasks
  - Condition
  - Composite Tasks and Multiple Instance Composite Tasks



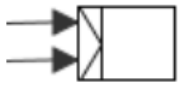
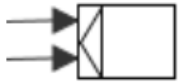
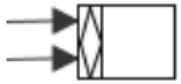
- Entrance/exit form blocks can be decorated with join/split elements
  - XOR split/join
  - AND split/join
  - OR split/join



Procedural, Closed Process Modeling: YAWL

Name:	Symbol:	Description:
Split Types:		
XOR-Split		The XOR-Split is used to trigger *only one* outgoing flow. It is best used for automatically choosing between a number of possible exclusive alternatives once a task completes.
AND-Split		The AND-Split is used to start a number of task instances simultaneously. It can be viewed as a specialisation of the OR-Split, where work will be triggered to start on *all* outgoing flows.
OR-Split		The OR-Split is used to trigger some, but *not necessarily all* outgoing flows to other tasks. It is best used when we won't know until run-time exactly what concurrent resultant work can lead from the completion of a task.

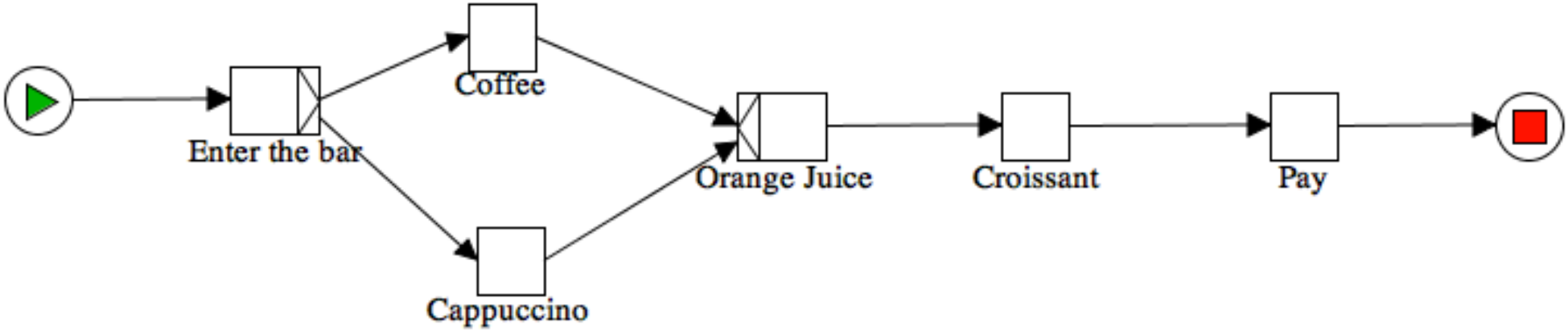
Procedural, Closed Process Modeling: YAWL

Join Types:		
AND-Join		A task with an AND-Join will wait to receive completed work from all of its incoming flows before beginning. It is typically used to synchronise pre-requisite activities that must be completed before some new piece of work may begin.
XOR-Join		Once *any* work has completed on an incoming flow, a task with an XOR-Join will be capable of beginning work. It is typically used to allow new work to start so long as one of several different pieces of earlier work have been completed.
OR-Join		The OR-Join ensures that a task waits until all incoming flows have either finished, or will never finish. OR-Joins are "smart": they will only wait for something if it is necessary to wait. However, understanding models with OR-joins can be tricky and therefore OR-joins should be used sparingly.

Procedural, Closed Process Modeling: YAWL

Model the following process:

A customer enters a bar, she **will** have a coffee or a cappuccino, (but not both), a croissant, an orange juice, and she is expected to pay. After paying and consuming, the customer exits the bar.



Problem:

The order of the activities is fixed

- Good for machines (easy to automatize)
- **Really bad** for humans

Business Process Model and Notation

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

BPMN – Business Process Model and Notation

- http://www.bpmn.org

- Silver, B. BPMN Method and Style, 2nd Edition. Cody-Cassidy Press, 2009.
- Weske, M. Business Process Management: Concepts, Languages, Architectures. Springer, 2007.
- Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. Fundamentals of Business Process Management. Springer 2013.
- Grosskopf, A., Decker, G., Weske, M. The Process: Business Process Modeling using BPMN. Meghan-Kiffer, 2009.

BPMN – Business Process Model and Notation

- OMG standardization initiative.

- 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 (strong assumptions on the shape of the acceptable BPMN models).

BPMN vs. Flowcharts

Very similar to flowcharts, but…

Difference #1: BPMN is a formal specification.

- It has a metamodel and rules of usage.
- BPMN models can be validated.

Difference #2: BPMN describes event-triggered behavior

- Accounts for how the process should react to events and exceptions.

Difference #3: BPMN captures business collaborations

- Tackles how different process communicate with each other.

What BPMN is…

A model and notation to represent the logic behind "standard" business processes.

Process Logic: Definition of all possible sequences of activities, so that, when the process knows

1. which events have occurred so far,
2. which activities have been completed,
3. which data have been produced,

it also knows **what has to be done next** (orchestration).

Focus on making the process **control-flow explicit**: BPMN reveals the allowed order of activities and when they happen. Nothing is said about **where** or **why**.

- Atomic tasks are not internally specified: BPMN tackles the process logic, not the task logic.
- Connection with data/resources quite weak.

BPMN – several uses…

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.

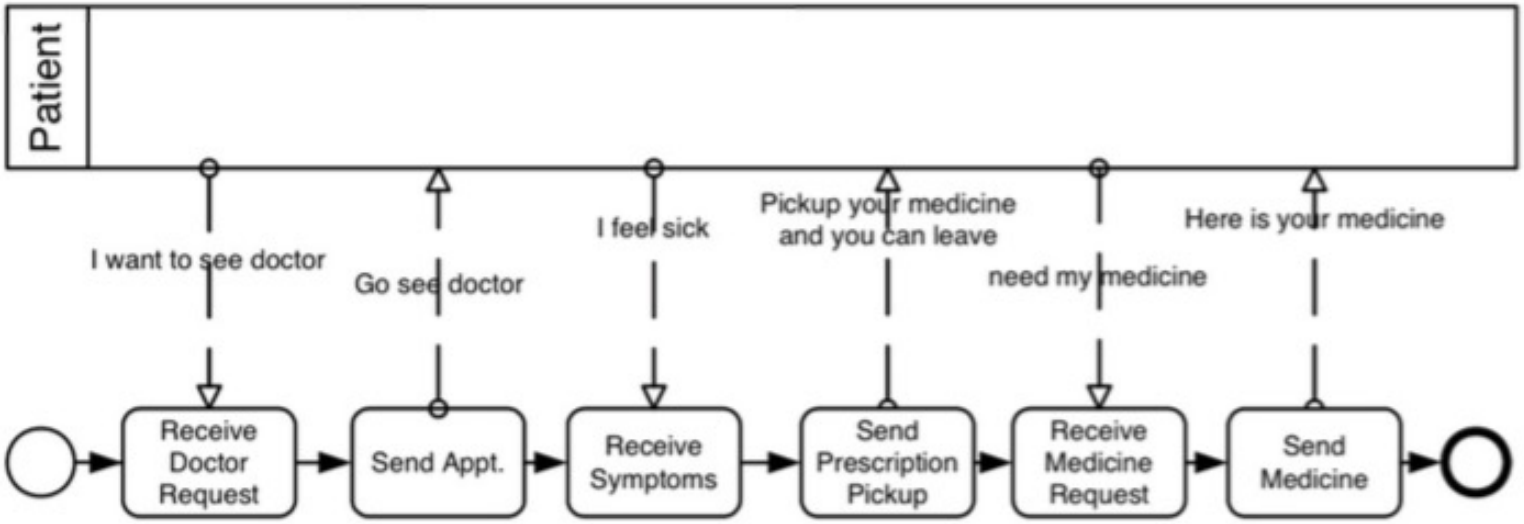
Collaboration: interaction between two or more business entities.

- 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

BPMN – Examples

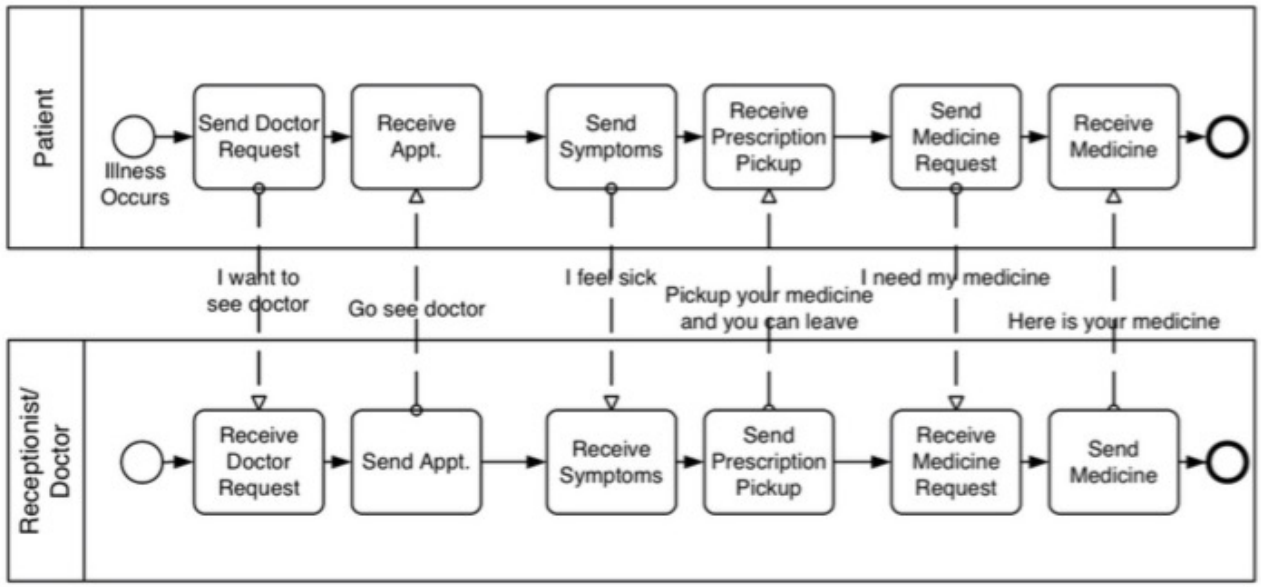
Private Process

Public Process

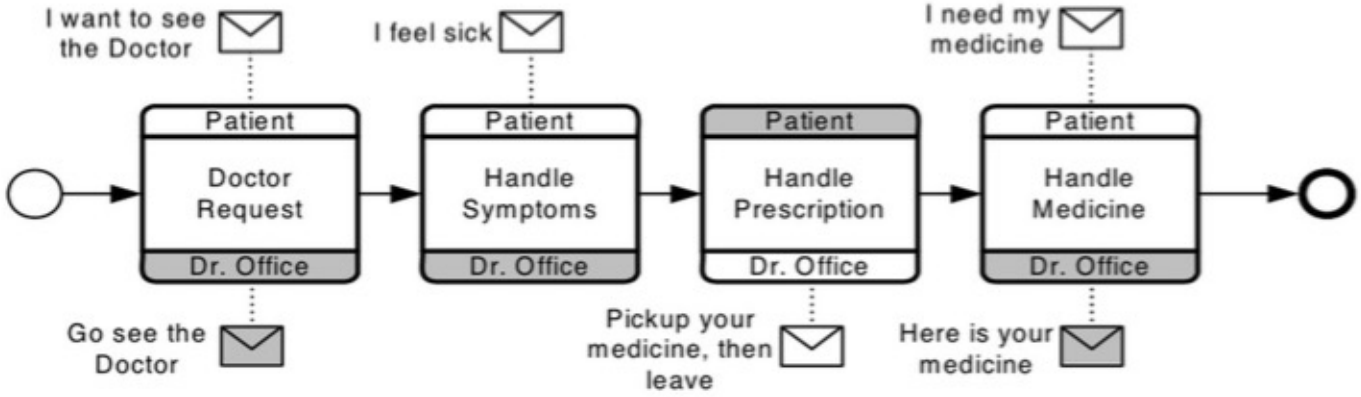


BPMN – Examples

Collaborative Process

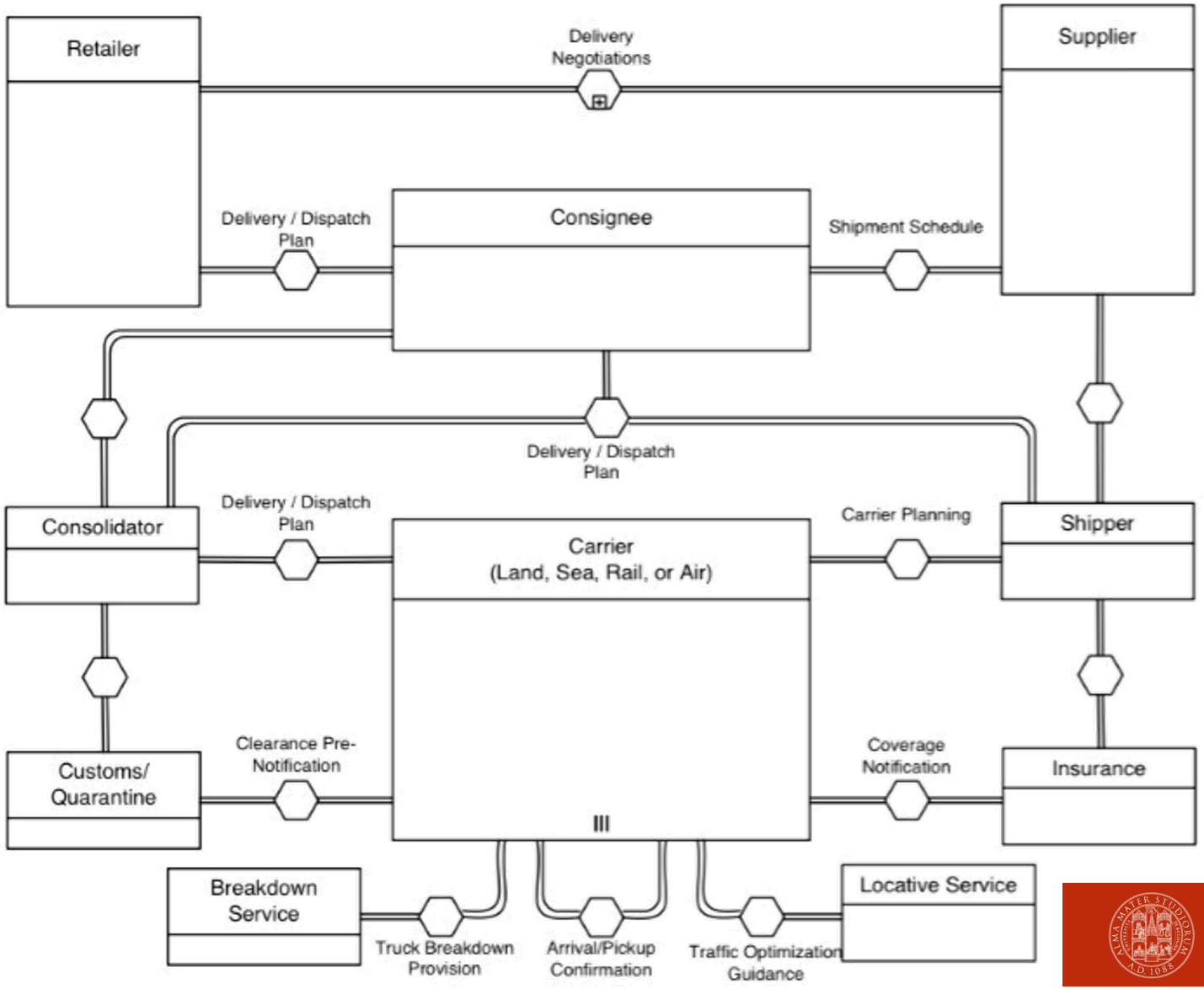


Choreography



BPMN – Examples

Conversation



BPMN – but…

- **No Method** – how to make a good model?
- **No Styling rules**

Why do we need them?

- **Clarity**: the BPMN process is unambiguous and self-explanatory.
- **Completeness**: all details have been properly considered.
- **Structural consistency**: different modelers come up with similar solutions to the same problem, facilitating understanding and communication.
- **Shareability**: BPMN as a mediator language for business and IT people.

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 second strata is very rich!

BPMN – Basic Elements

	Activity.
	Event.
	Flow.
	Gateway.

BPMN – Modeling Levels

1. **Descriptive process modeling**. Traditional flowcharting for main processes, focusing on the internal orchestration of activities.
   Uses a minimal subset of the BPMN notation.

2. **Analytic process modeling**. Expansion of level 1 with reactive behaviors and exception handling.
   Uses an extensive palette of constructs and decorators to enrich the level 1 modeling.

3. **Executable BPMN**. Enrichment of the process model with all necessary details to enable process orchestration. Typically done at the XML level.

BPMN – Initial questions

- What is the goal of the process? What is the main "object" evolved by the process? What do process instances represent? **case**
- How does the process actually start? In response to which event? Are there multiple ways of starting the process? **start events**
- When does the process end? Are there multiple ways to complete the process? To which completion state are they associated? **end events**
- How does the process get from X to Y? Is there only one way to reach Y from X? Are there alternative paths? Are there tasks that can be executed in parallel? **tasks, gateways, sequences**
- How is it known when X is done? Are there abnormal/exceptional/ peculiar conditions that end X and lead to a different continuation for the process? Are there rules governing such different outcomes? **events, tasks, gateways, sequences**

BPMN – 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).

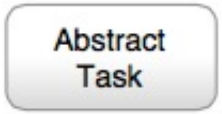
BPMN – Activity

Activity: A **unit of work** performed in the process by a *performer*. Each activity is either a task or a subprocess.

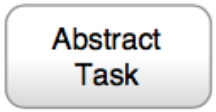
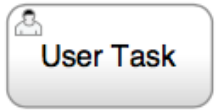
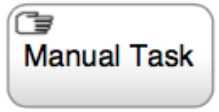
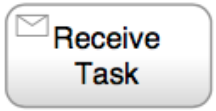
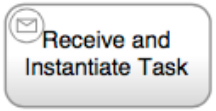
- Task
  Atomic unit of work. Its internal specification is *not* tackled by BPMN. Each task implicitly has a well-defined starting and completion point. Typically labeled using a **verb-noun** form.

- Subprocess
  Compound unit of work, whose components can be described as a new, *child BPMN process*.

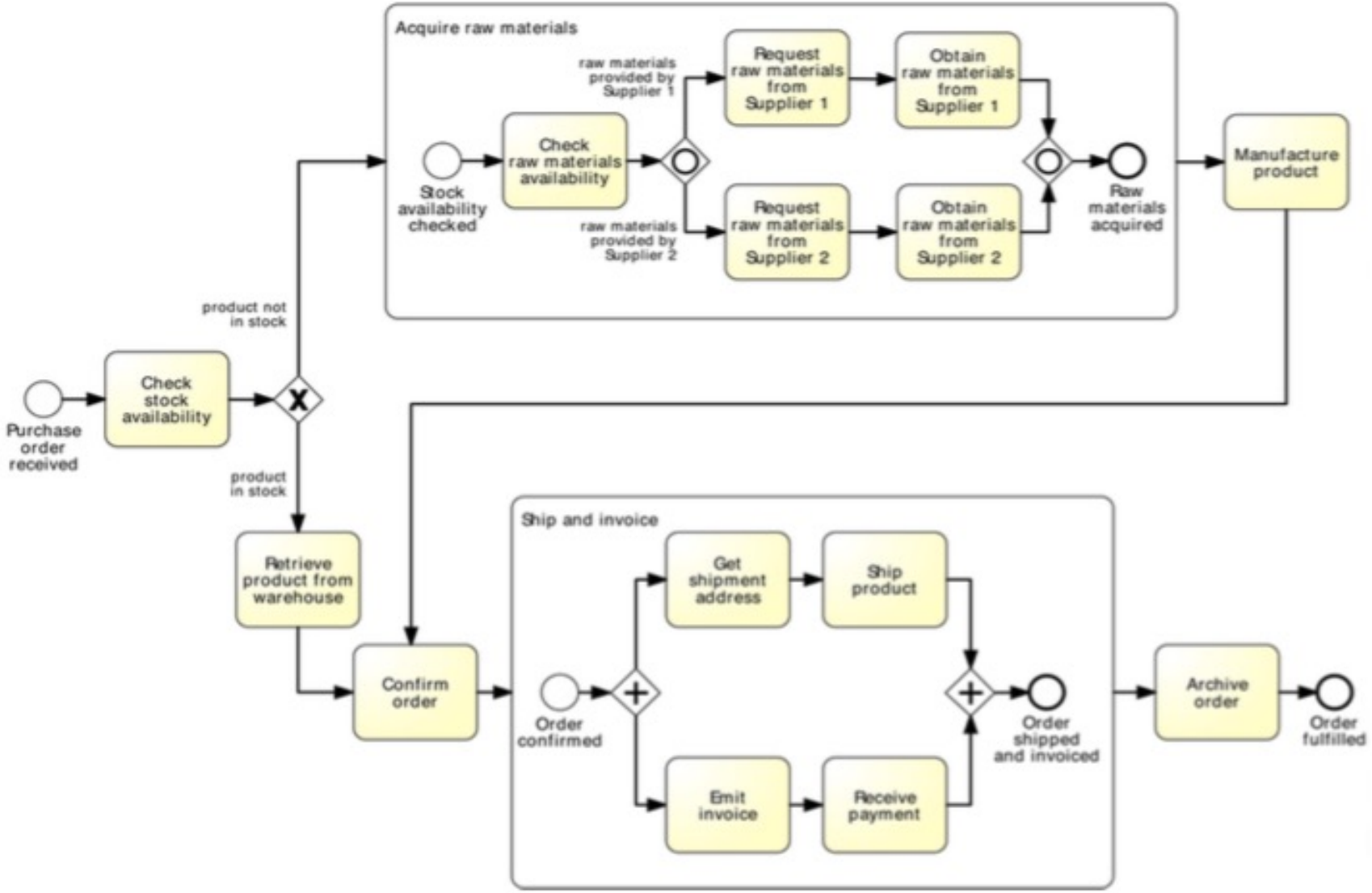
BPMN – Basic Task Types

	A generic task, whose type is not specified.
	A task under the responsibility of a human (interacting with the information system).
	An automated task, managed autonomously by the system without human intervention.

BPMN – Other 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.		

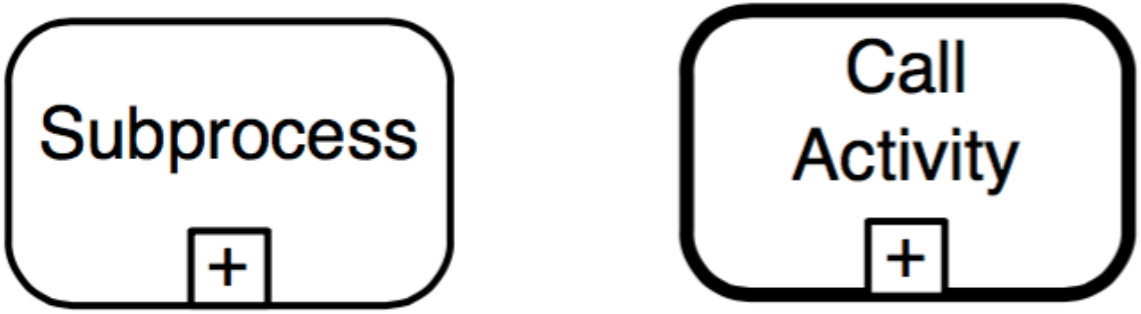
BPMN – Processes and subprocesses



BPMN – About subprocesses…

- A **subprocess** is always defined in the context of (i.e., embedded in) its parent process.

- BPMN supports also the notion of **call activity**: a compound activity whose definition is independent.



BPMN – Gateways

Manage the

- divergence of one control flow into many subsequent flows. (**split gateway**)
- convergence of multiple flows into a single flow. (**join gateway**)

Split gateways have a single input sequence flow, multiple output sequence flows (vice-versa for join gateways).

BPMN – XOR Split

Empty diamond or diamond with an inner X.

Choice in the process. Leads to select one and only one output flow.

- *Data-based XOR split*: decision taken by checking data-based conditions and evaluating which one is true.
  Conditions are expressed at this stage using natural language.
  Default flow: represented with a bar on the sequence flow.
- *Deferred choice*: external decision under the responsibility of the process executors.

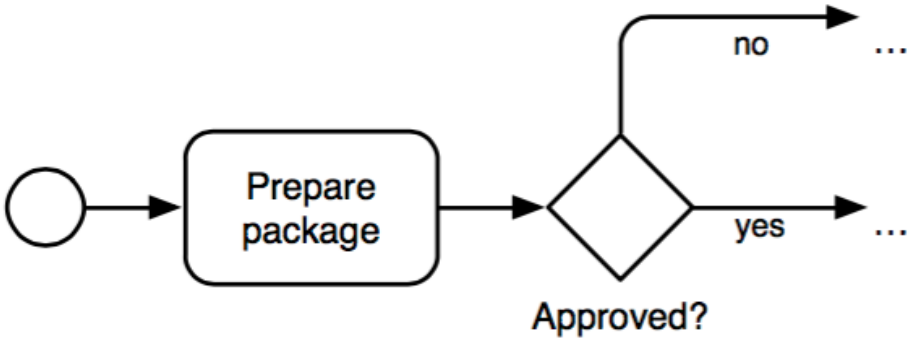
Typical guideline:

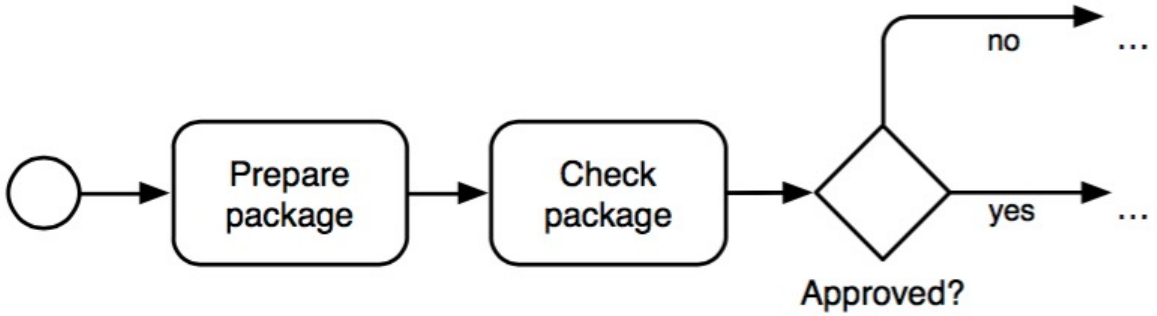
- Attach a **question label** to the split (e.g., approved?).
- Draw an output for each of the **answers** (e.g., yes/no).

In the case of data-based choice, the modeler must ensure that one and only one output will be activated for a certain data configuration.

BPMN – XOR Split

An XOR split does not make a decision. It just tests conditions…





BPMN – XOR Join

Merges common process continuations into a unique BPMN fragment.

Whenever an input is enabled, the output is enabled.

BPMN – AND Gateway (parallel gateway)

Diamond with an inner +.

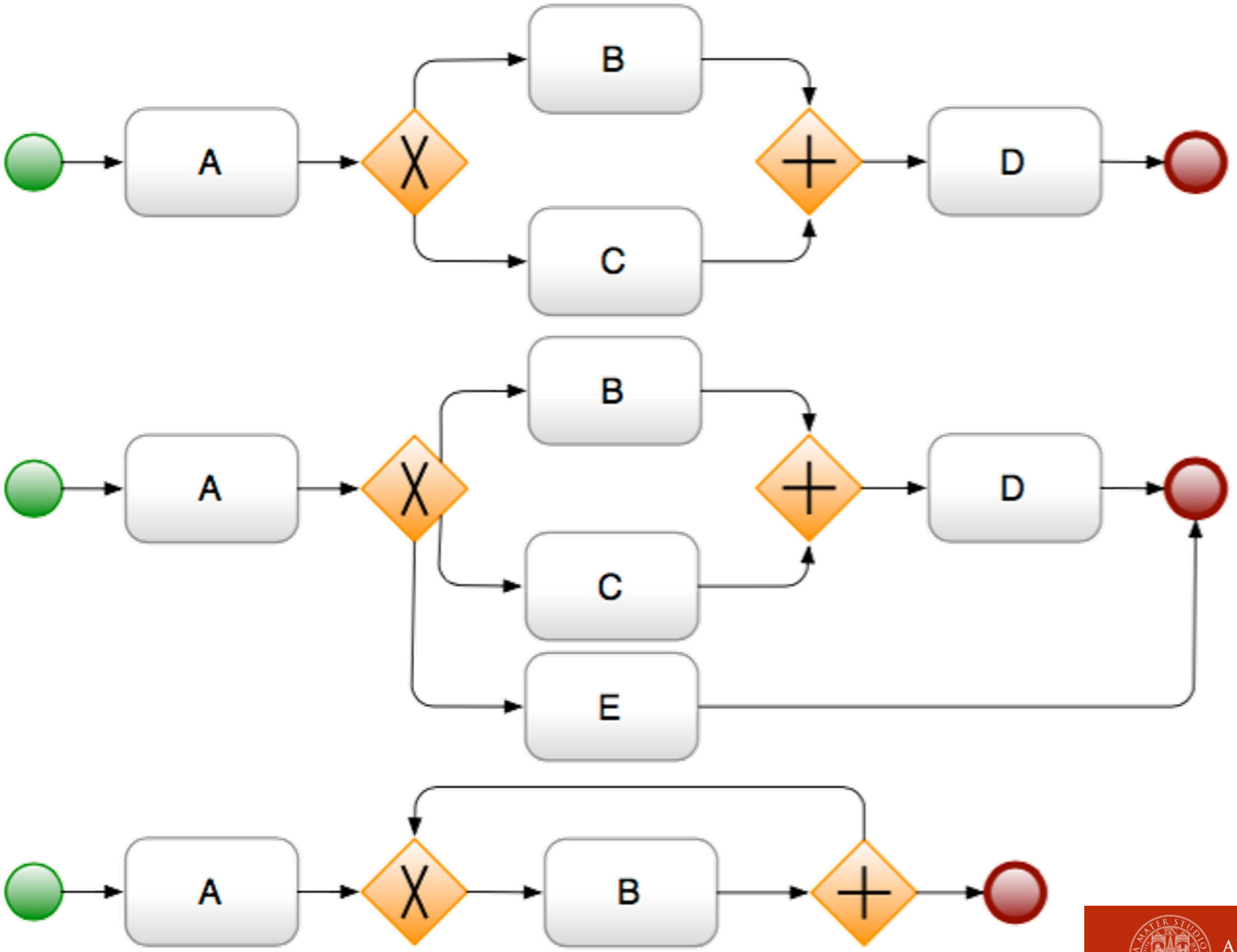
Models a **fork** in the process: all the output sequence flows are to be followed in parallel, unconditionally.

Such parallel threads:

- may be later joined into a unique thread, or
- may reach separate end events.

Models a **synchronization point** in the process: the output is enabled as soon as *all* the incoming flows have been activated.

BPMN – Gateways… mind how you use it…



BPMN – Start event

Circle with a thin border.

Indicates **where** and **how** a process starts.

- Every subprocess has a single, generic start event.
- Top-level processes may have multiple start events.

- Top-level processes may associate a **trigger** to the event.

The trigger indicates the semantics of the event, and identifies the meaning of the process as the handling of such an event.

Each case starts executing a process because an instance of that event has been caught.

Basic triggers:

- none -- Just a thin circle
- message -- Thin circle with a white envelope inside
- timer -- Thin circle with a white clock inside
- multiple -- Thin circle with a pentagon or a plus inside

BPMN – End event

Circle with a thick border.

Indicates the **end of a path** in a process or subprocess, possibly indicating a **result**.

- Unlike start events, multiple end events are common.
- Typically, each end event indicates a different end state for the (sub)process.

The result indicates the semantics of the event, and identifies what is produced when the process reaches that end point.

Basic results:

- none -- Just a thick circle
- message -- A thick circle with a black envelope inside
- termination -- A thick circle with a bulls-eye (black-filled circle) inside (kills the other threads still alive)
- multiple -- A thick circle with a pentagon inside

BPMN – Sequence vs Message Flows

Sequence flow

- Denotes orchestration: the flow of control within a process.
- General best practice: all activities, gateways, events in a process must lie on a continuous chain of sequence flows.
- Sequence flow is confined in a specific process level.
  No sequence flow can cross a subprocess boundary.
- No sequence flow can cross a participant boundary.

Message flow

- Denotes communication between the process and an external entity.
- Can connect to any type of activity, a message/multiple event, a black-box participant (cf. pool).
- No two elements belonging to the same process (i.e., jointly orchestrated) can be interconnected through a message flow.
- When starting from a generic or user task, the message flow only indicates possibility of communication.

BPMN – Organizational Modeling

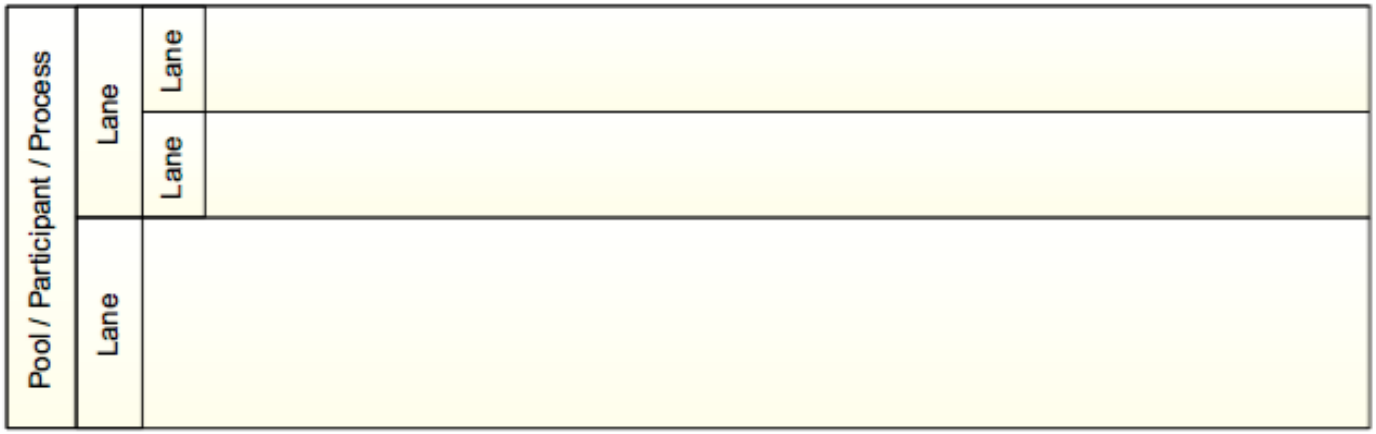
Pool: representation of an independent participant (i.e., resource class), with its own BP specification. It can be "implicit", i.e., not shown.

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, but also PurhcasingDept.

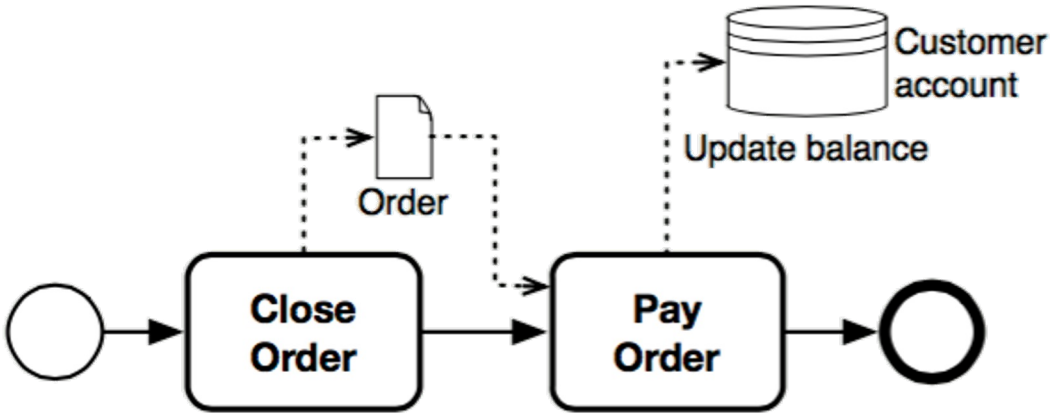
. . .

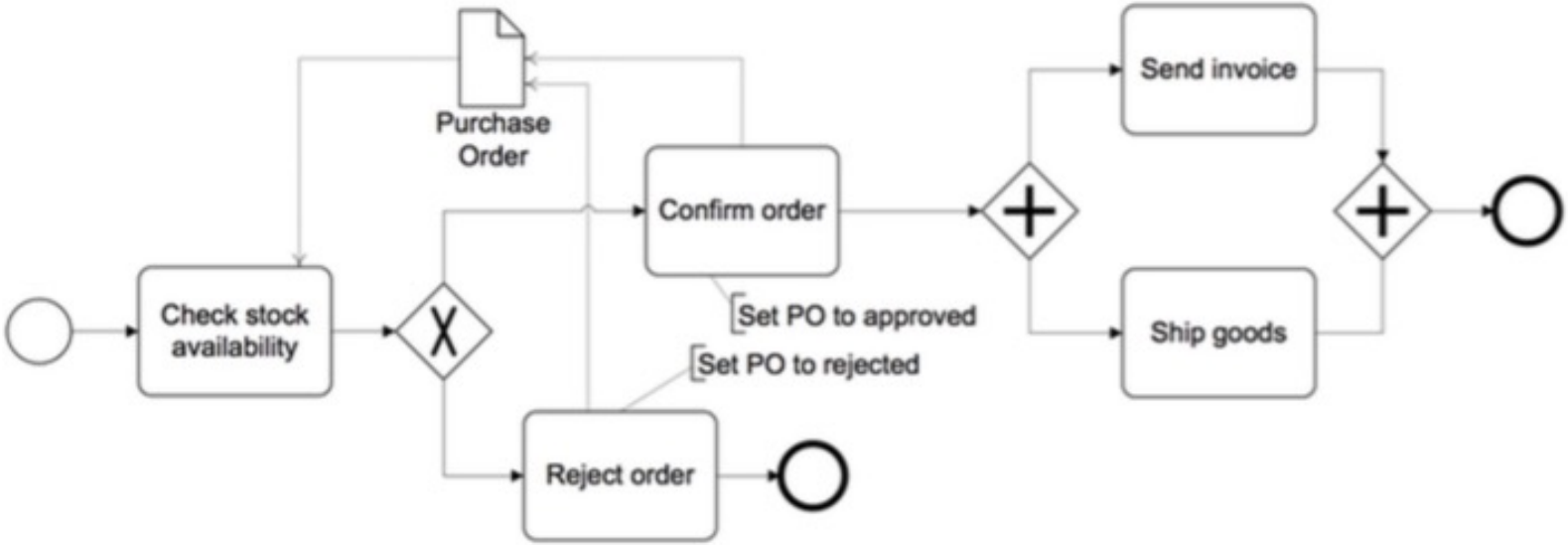


BPMN – Data

 Data Object	Data object: **local variable** inside a process level, pointing to a **temporary unit of information**.
 Data Object [Object State]	Data object reference: refers to a data object in some state.
 Data Object Collection	Variable represeting a collection of data objects.
 Data Store	Data store reference: reference to **persistent unit of information**, manipulated by the process but also external entities.

BPMN – Data – Examples





BPMN Level 2: Reacting to events…

Event: Something that happen in a process, at a specific point in time.

Reaction to events: depends on whether the process is throwing or catching the event.

- **Reaction to throwing an event**: how the process generates a signal that something happened.
  Type of signal represents the trigger.

- **Reaction to catching an event**: how the process responds to a signal that something happened.
  Type of signal represents the result.

The type of signal is shown as an icon inside the event circle.

BPMN – Event Types

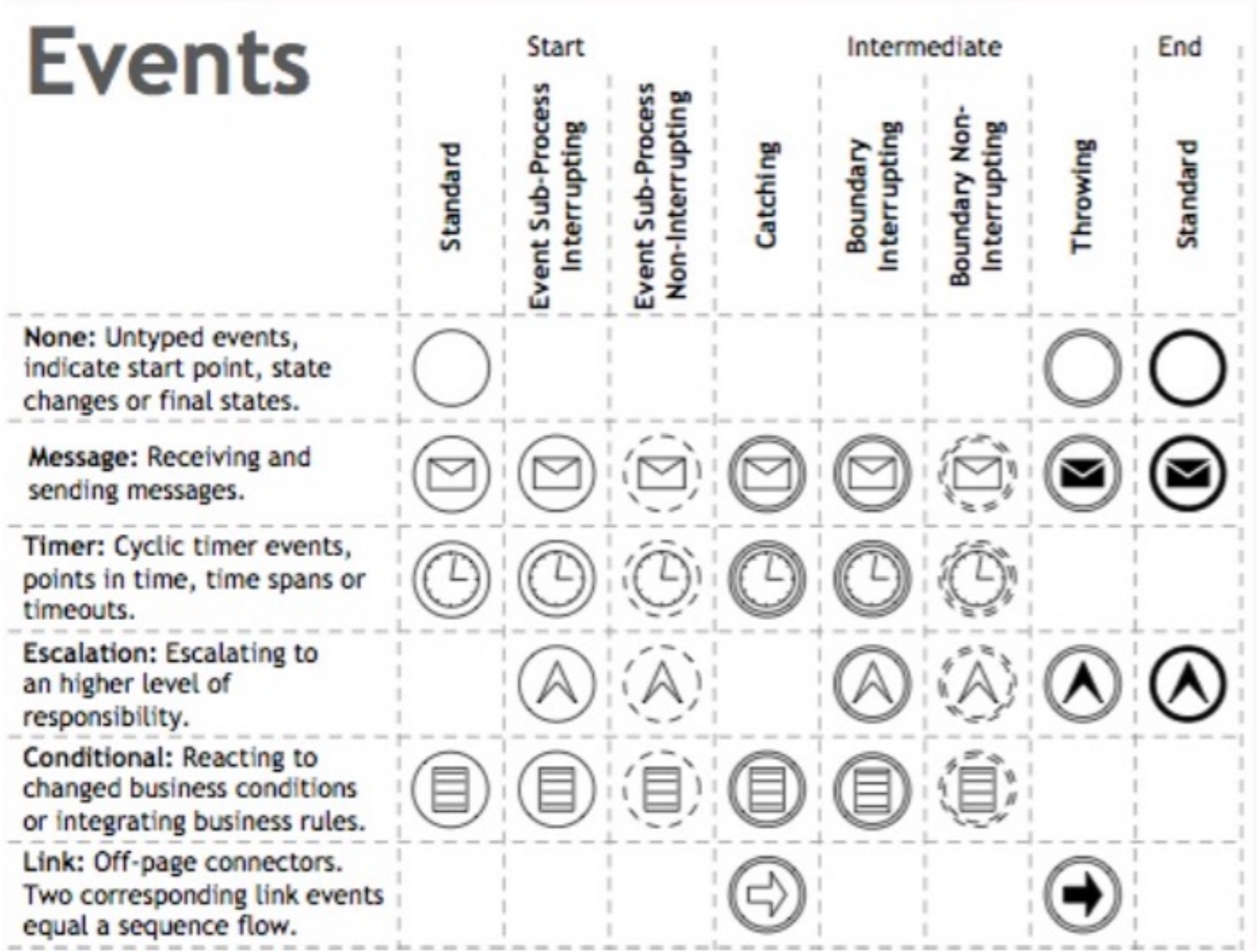
- **Start event** (thin line): indicates where the process starts.
- **End event** (thick line): indicates where a path of the process ends.
- **Intermediate event** (double ring): indicates that something happens during the execution of the process.

It is now possible to model:

- Reaction to external events.
- Exception handling.
- Parallel event handlers.
- Complex indirect interactions between different process parts, where one triggers an event and the other catches it.
- Cancelations and compensations.
- …

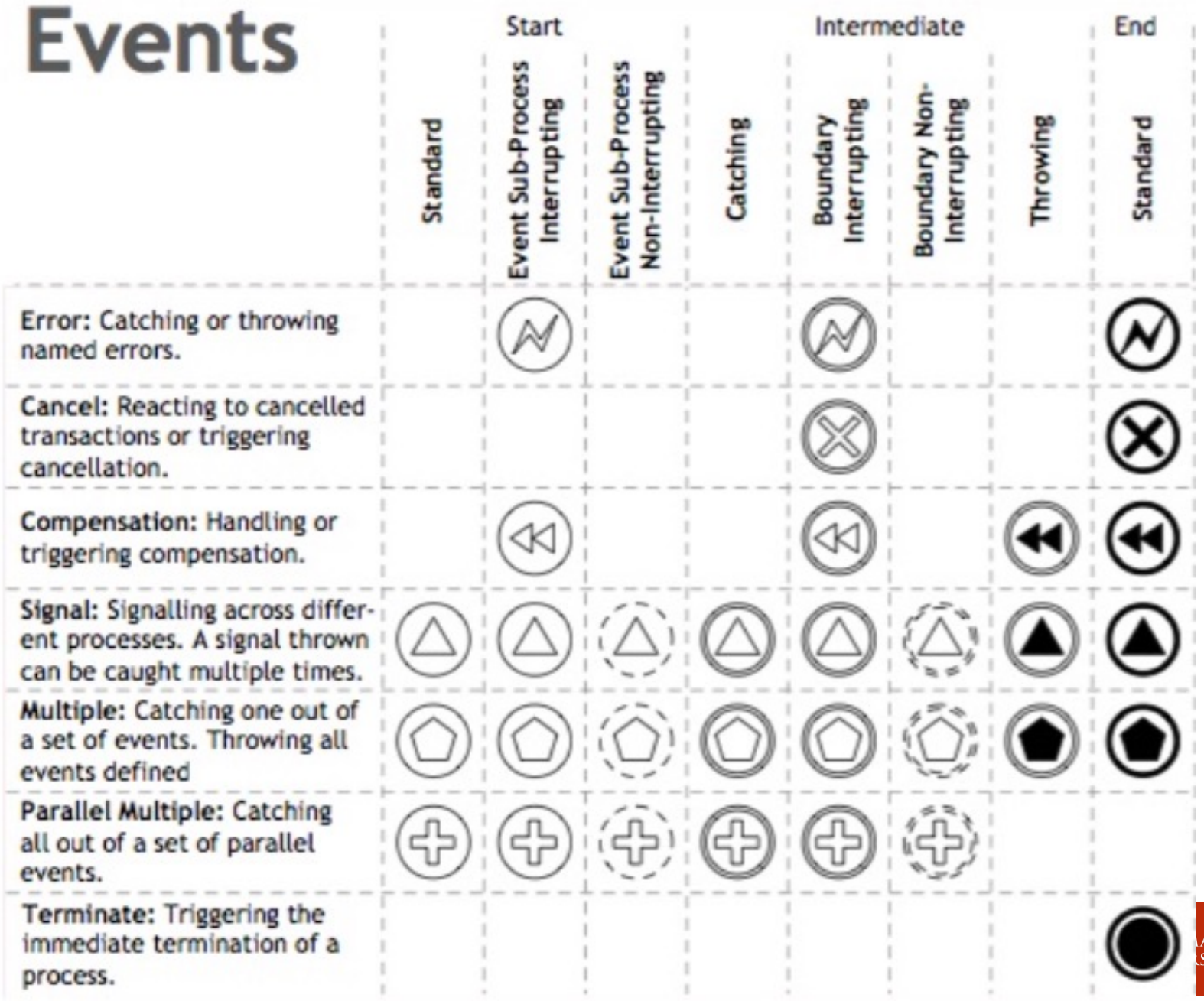
… and the standard get fearsome…

BPMN – Events…



DIORUM
OLOGNA

BPMN – Events…



BPMN – Intermediate Events

Intermediate Events are events occurring after the start of a process level, but before its end. Four basic contexts of use:

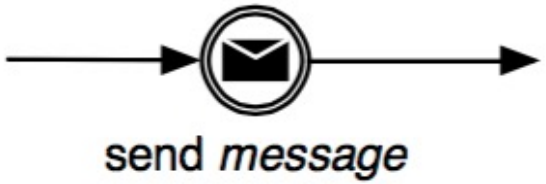
1. Throw an intermediate event.
2. Catch an intermediate event.
3. Catch an intermediate event in a specific process level, by interrupting it.
4. Catch an intermediate event in a specific process level, without interrupting it.

BPMN – Intermediate Events

Throwing

Intermediate event with a black icon inside. Semantics:

1. As soon as the sequence flow reaches the event, the corresponding signal is **thrown**.
2. The process continues immediately.

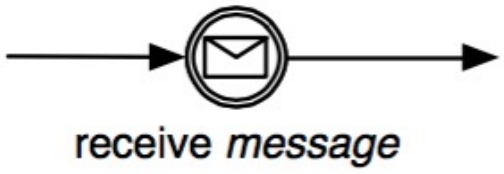


Catching

Intermediate event with a black icon inside. Semantics:

1. As soon as the sequence flow reaches the event, the process **waits** for the corresponding trigger signal.
2. When the trigger signal is caught, the process resumes immediately.

Supported by many event types, but not error events.



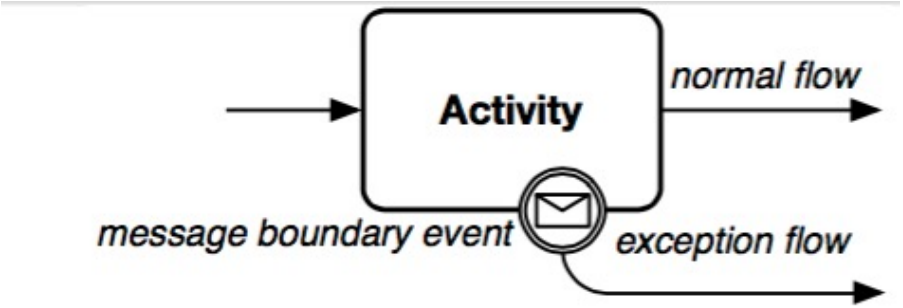
BPMN – Intermediate: Boundary Events

Catching intermediate event drawn on the boundary of an activity.

Semantics:

1. **While** the activity is running, it **listens** to the signal attached to the event.
2. If the activity completes without the occurrence of the boundary event signal, the process continues on the standard execution path (normal flow).
3. If the signal occurs before the activity completes, the sequence flow out of the event is triggered (exception flow).

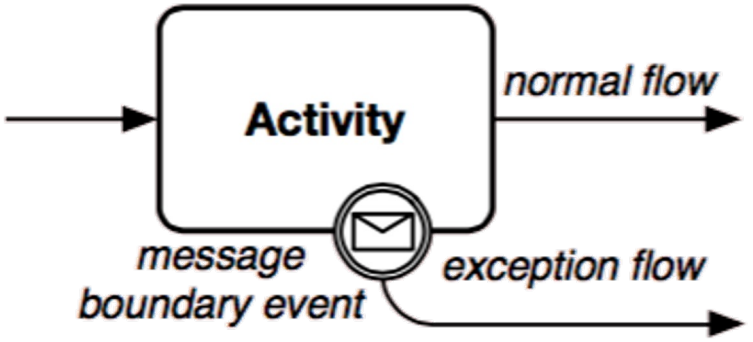
How this is actually done depends on the *type* of the boundary event.



BPMN – Types of Boundary Events

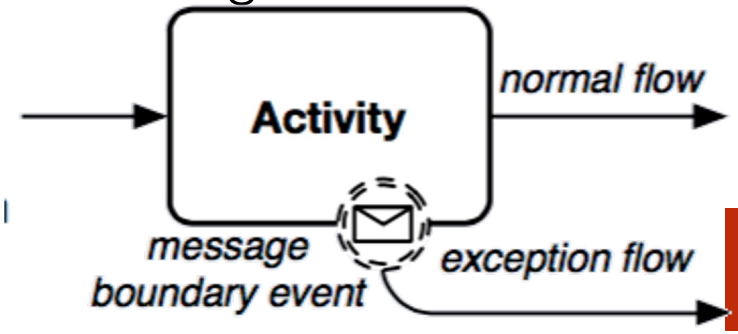
Interrupting boundary event: If the trigger signal occurs during the activity execution:

- The activity is **immediately terminated**.
- The exception flow is activated.
- Drawn with a **solid** double ring.

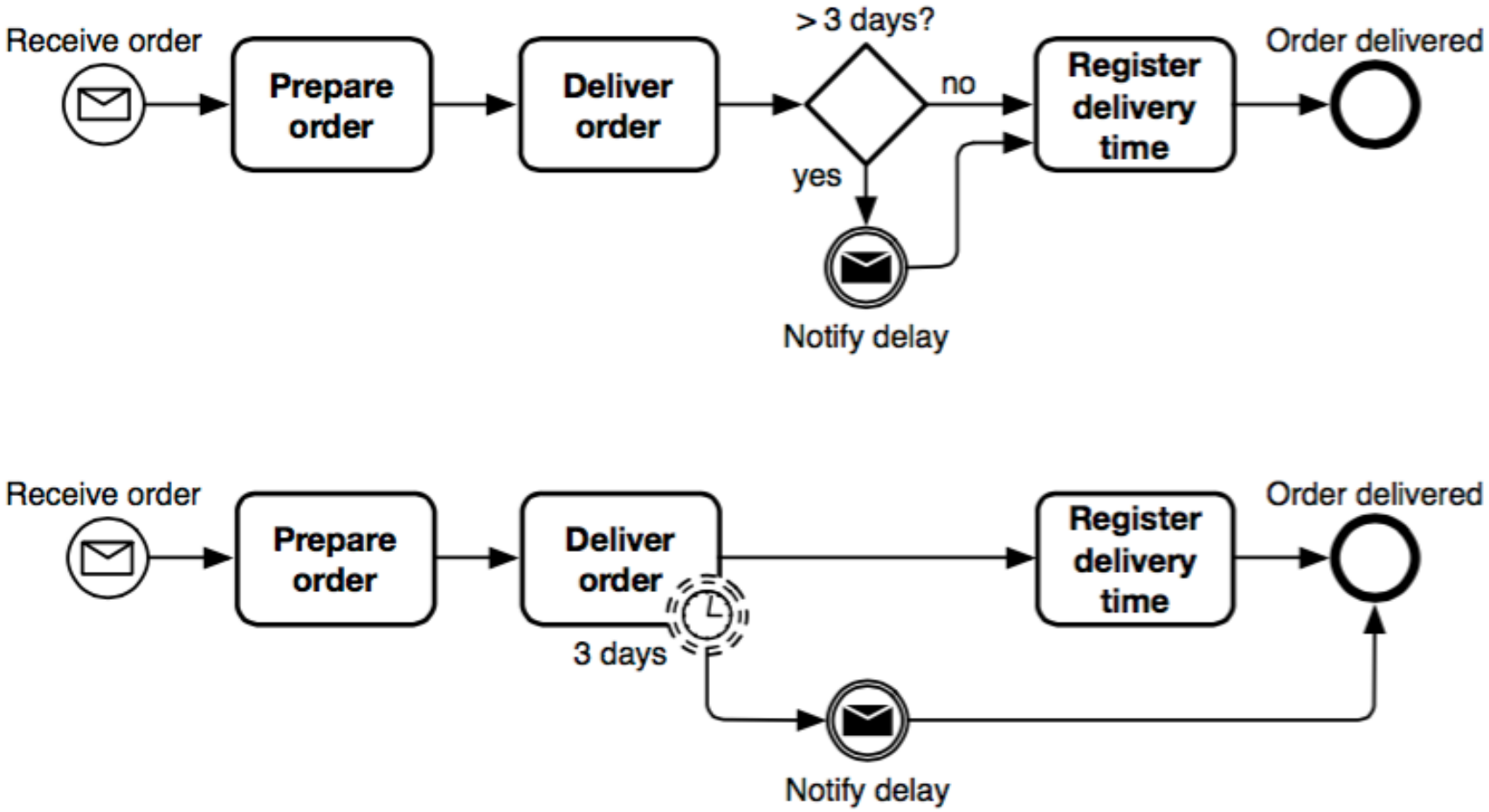


Non-interrupting boundary event: If the trigger signal occurs during the activity execution:

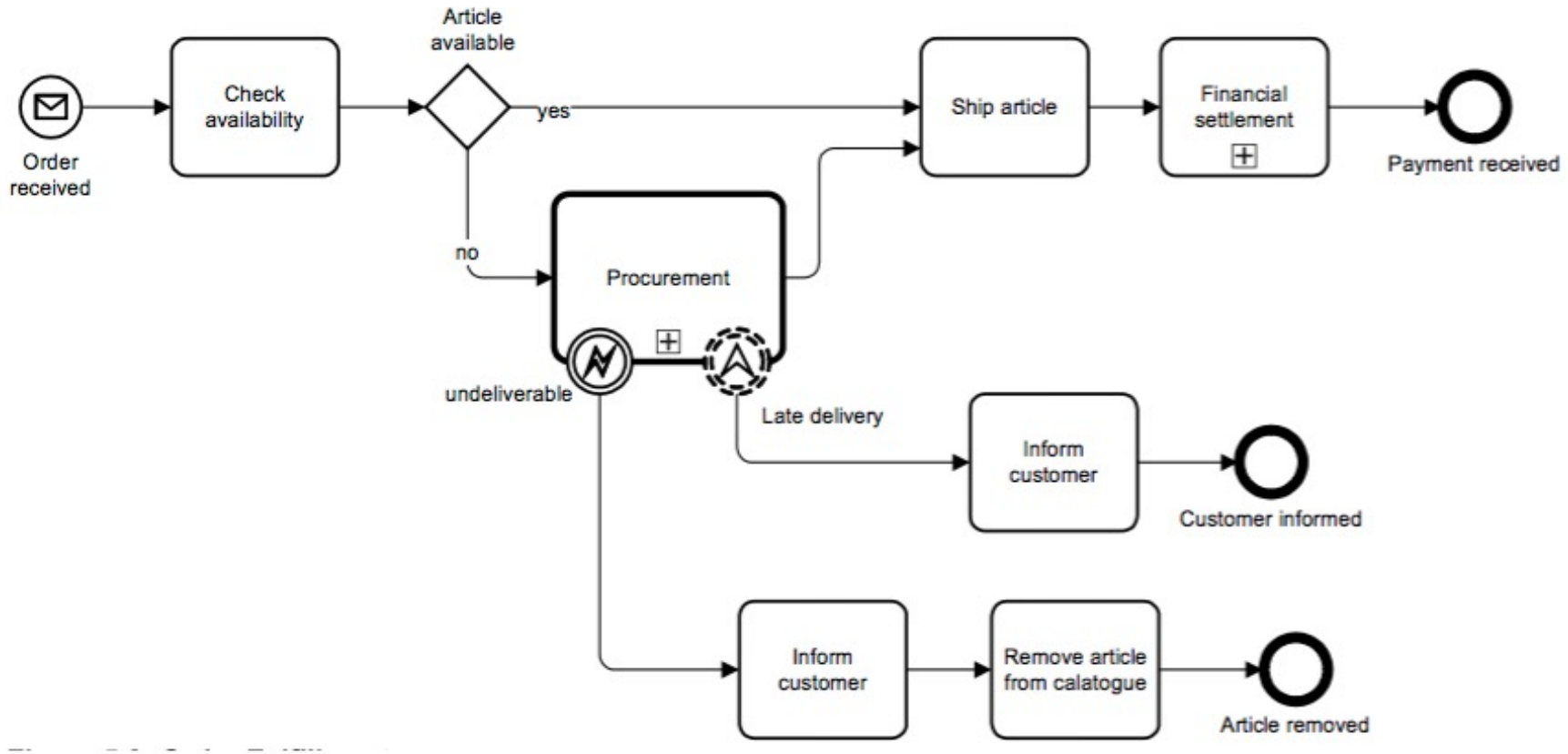
- The activity **normally continues**
- The exception flow is activated (with a new parallel thread).
- Drawn with a **dashed** double ring.



BPMN – Differences?

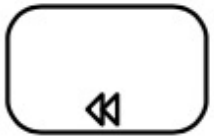
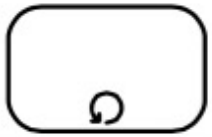
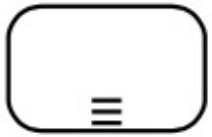


BPMN – And many other event types…

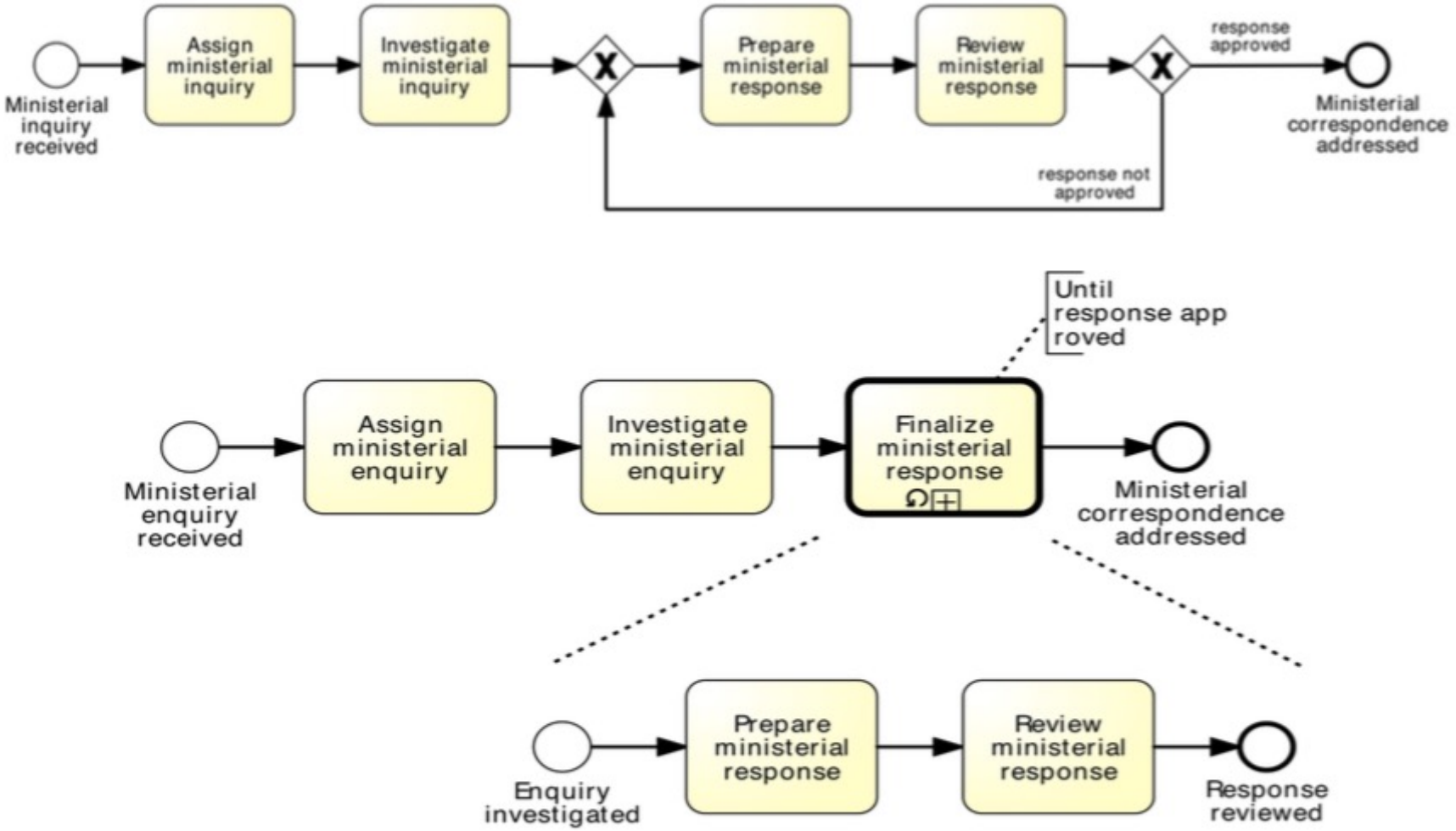


From *BPMN 2.0 by Example* - `http://www.bpmn.org/`

BPMN – 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.

BPMN – Loop Example



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Federico Chesani

DISI – Department of Computer Science and Engineering

federico.chesani@unibo.it

www.unibo.it