بسمه تعالی

دانشکده مهندسی صنایع

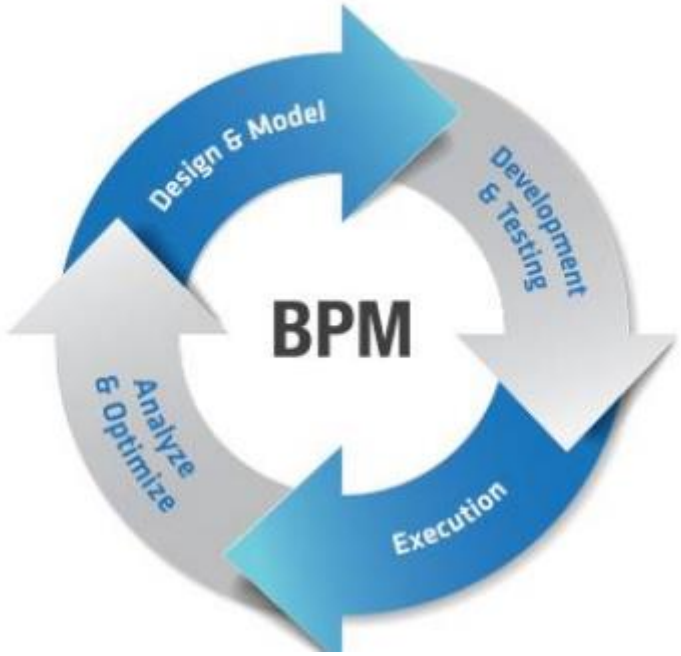
مدیریت فرآیندهای کسب و کار
Business Process Management (BPM)

هانیه وحیدچیرانی

آدرس ایمیل: HANIEHVAHID@YAHOO.COM

اهداف BPM

آشنایی با مفاهیم، مدل‌هاو ابزارهای مدیریت، بازمهندسی و مدل‌های فرآیندهای کسب‌وکار



Process modeling tools

Brand Name	Popular With	Process Design
visual-paradigm	Small and Midsize Business, SMBs, enterprise	BPMN 2.0
Nintex	Small and Midsize Business, SMBs, enterprise	Browser-based intuitive design
Process Maker	Corporations, governments, and organizations looking for an enterprise scale	BPMN 2.0
Kissflow	SMBs, mid-market	Human-centric drag and drop interface
Bizagi	**SMBs, medium businesses, enterprises**	**Bizagi BPMN Modeler**
Zoho Creator	Freelancers, small- and mid-size businesses, and enterprise	Script builder to drag and drop snippets of code

استاندارد BPMN

1. BPMN is aa graphical representation for specifying business processes in a business process model.
2. Originally developed by the Business Process Management Initiative(BPMI).

End-To-End business Process

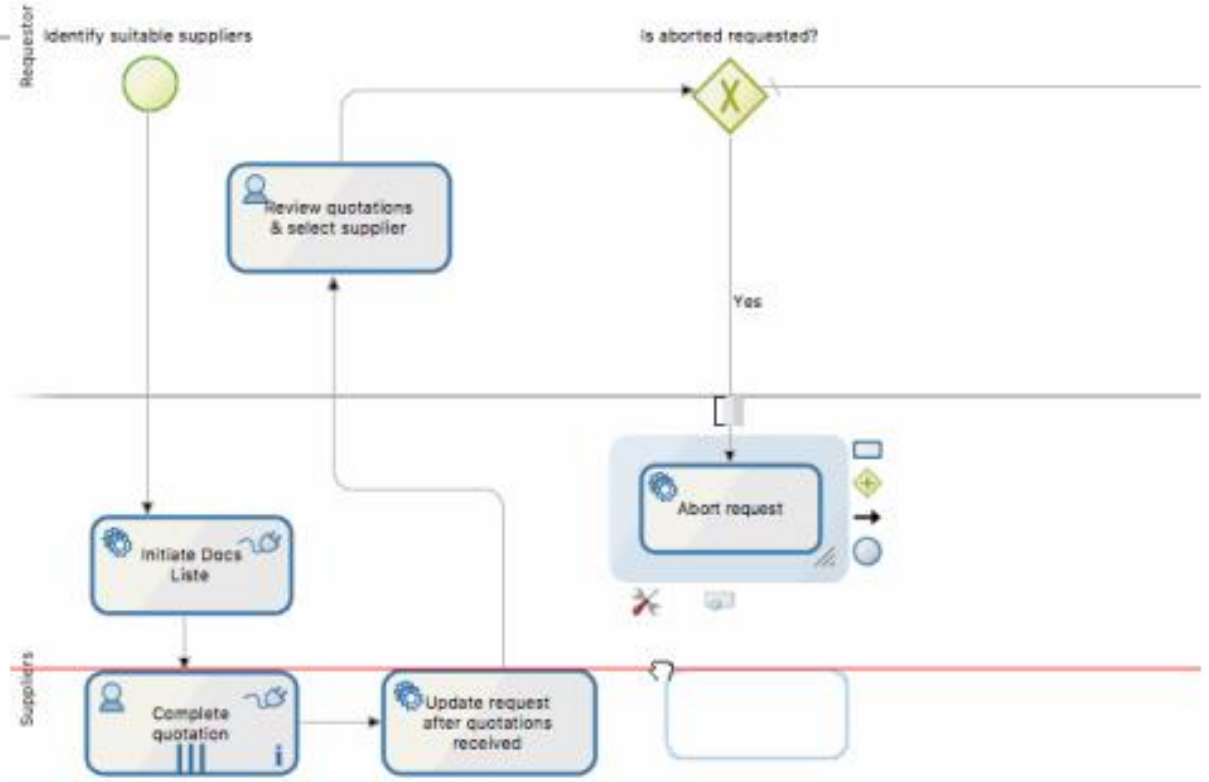
End-to-end means “from the very beginning to the very end”.

اشاره به پیوستگی فرآیند دارد.

1. A chain of process steps (or sub-processes) that **starts** as the result of a **customer trigger** and proceeds through **until** a **successful outcome for the customer is achieved**.
2. Examples of end-to-end processes
   1. **Order-to-Cash (Order-to-payment)**
   2. **Order-to-Delivery**
   3. **Procure to pay**

The steps that can be recognized in BPM are:

1. Analyze
2. Re-design and model
3. Implement
4. Monitor
5. Manage
6. Automate



BPM stages

Step 1: Design

Most processes include a form to collect data and a **workflow** to process it. Build your form and identify who will **own** each task in the workflow.

Step 2: Model

Represent the process in a visual layout. Fix details like **deadlines** and **conditions** to give a clear idea of the sequence of events, and the flow of data through the process.

BPM stages

Step 3: Execute

Execute the process by testing it live with a small group first and then open it up to all users. Make sure you restrict access to sensitive information.

Step 4: Monitor

Keep an eye on the process as it runs through the workflow. Use the right metrics to identify progress, **measure efficiency**, and locate **bottlenecks**.

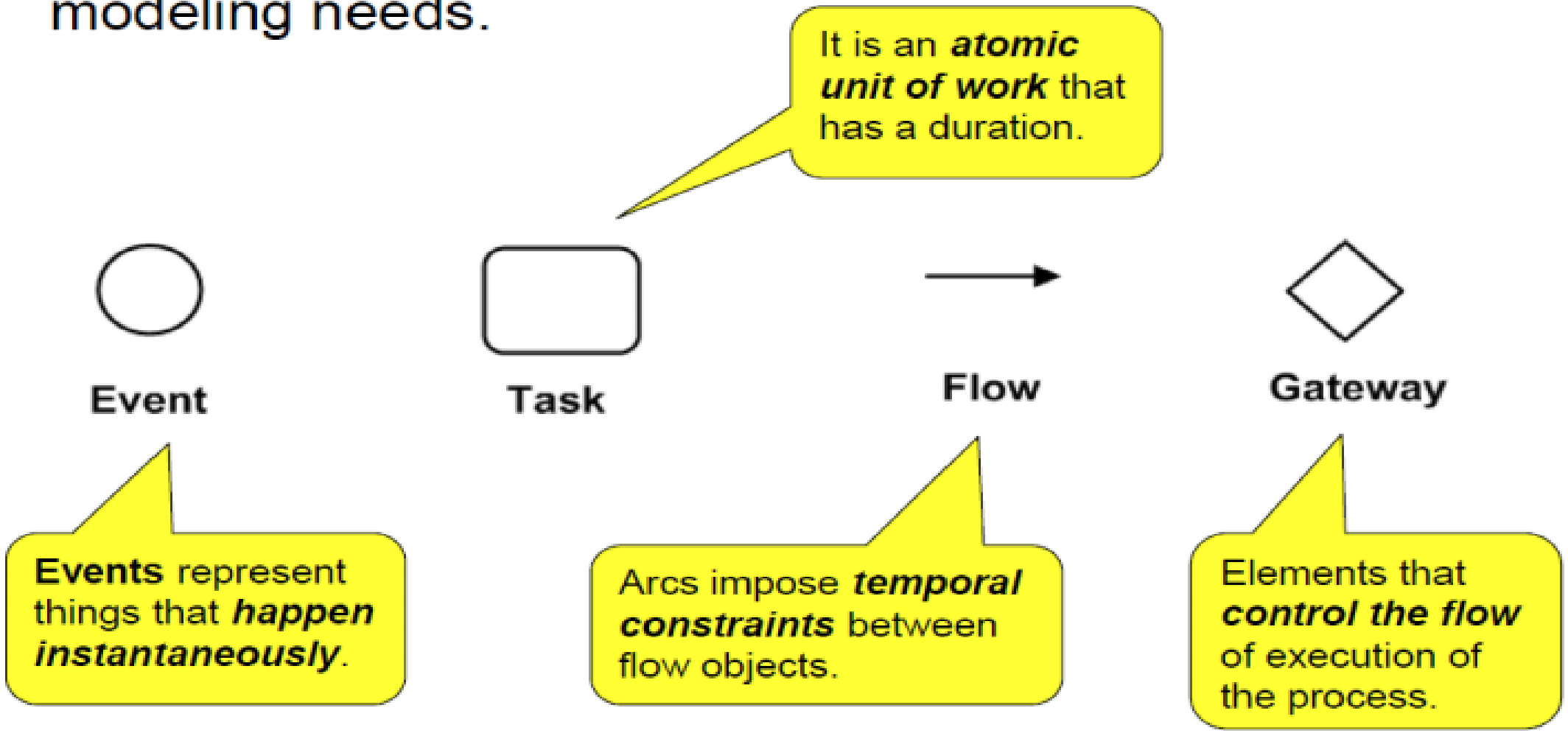
Step 5: Optimize

As you analyze, notice any changes that need to be done to your form or workflow to make them more efficient.

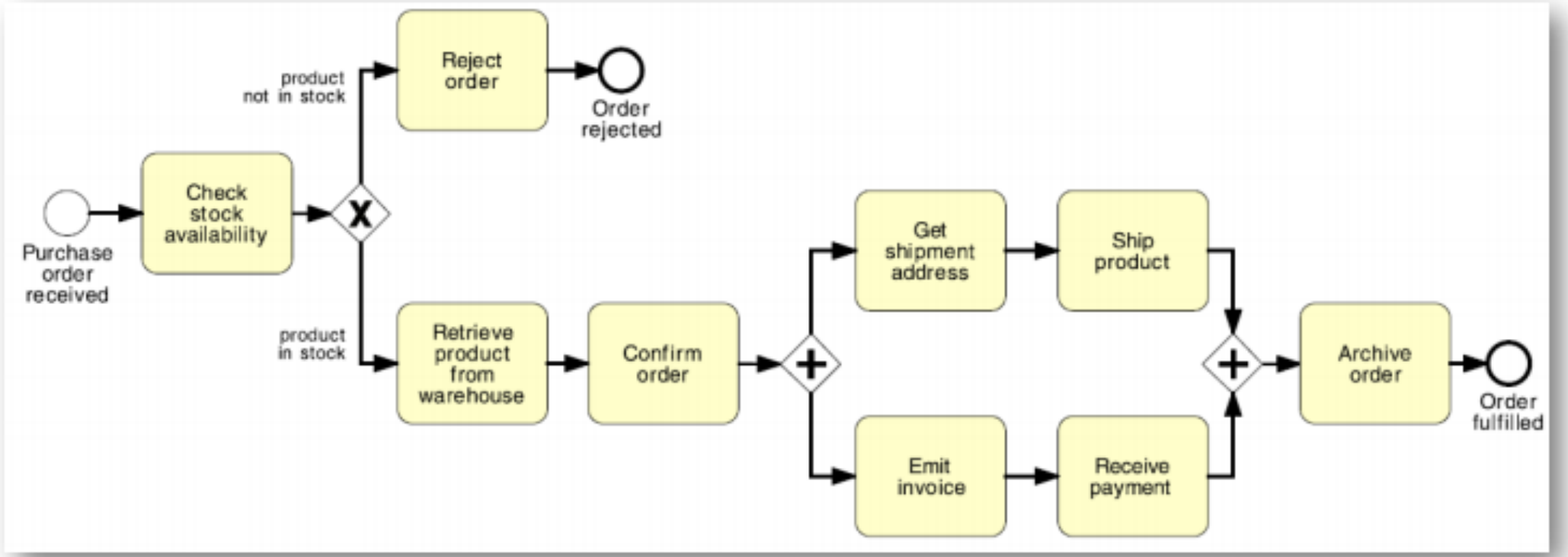
Business process modelling techniques:

1. **Business Process Model and Notation (BPMN)**
2. **Event-driven process chain (EPC)**
3. Life-cycle Modelling Language (LML)
4. Subject-oriented business process management (S-BPM)
5. Cognition enhanced Natural language Information Analysis Method (CogNIAM)
6. Extended Business Modelling Language (xBML)
7. Unified Modelling Language (UML)
8. Formalized Administrative Notation (FAN)
9. Harbarian process modeling (HPM)

4 basic elements, that usually cover the 80% of modeling needs.

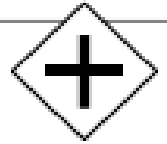


Example: Order fulfillment process diagram

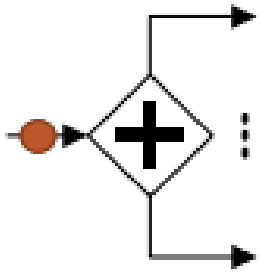


AND Gateway

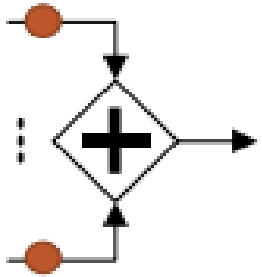
AND Gateway



An *AND Gateway* provides a mechanism to create and synchronize "parallel" flows.



AND-split → takes **all** outgoing branches

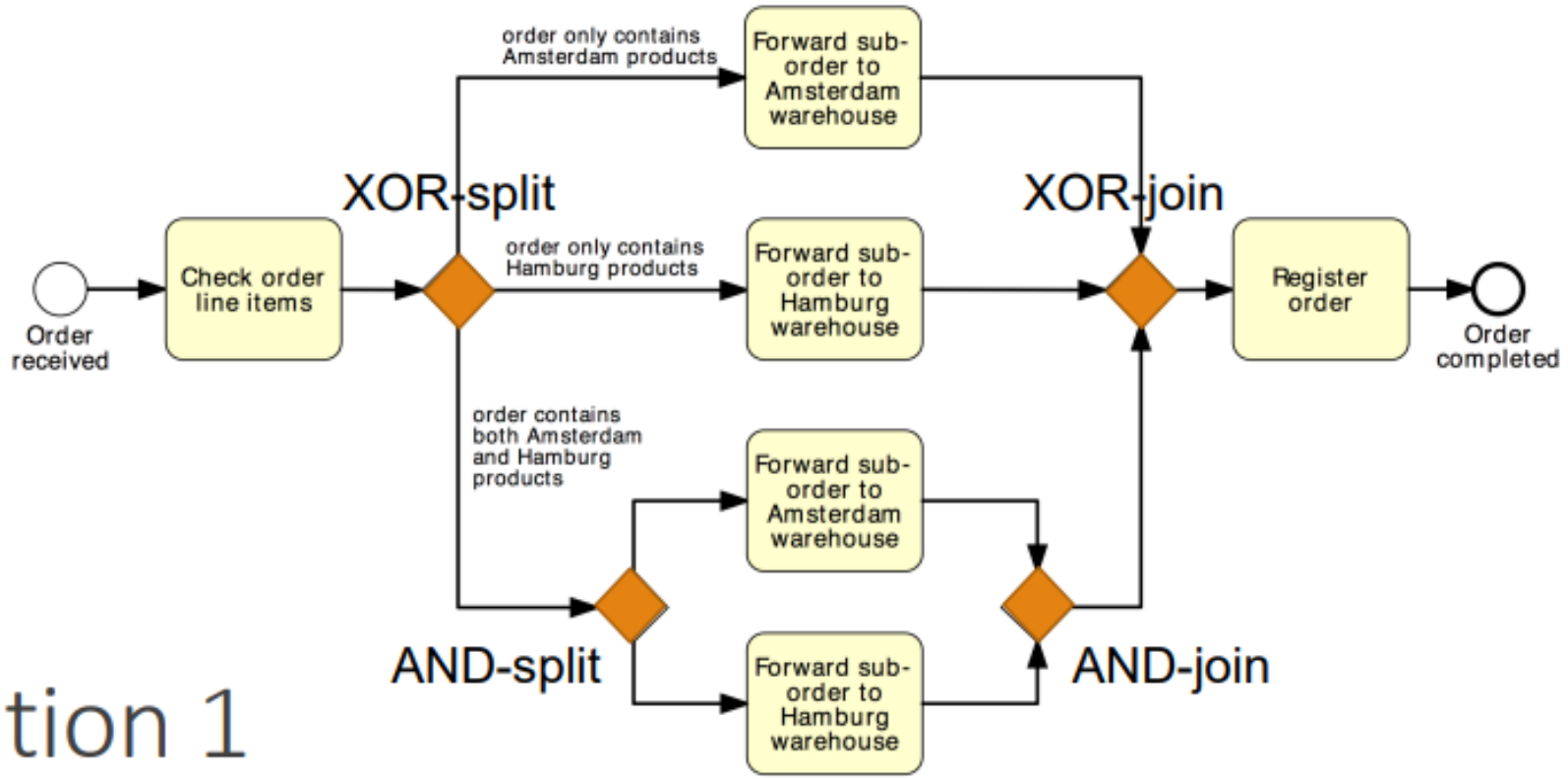


AND-join → proceeds when **all (both activated or non-activated)** incoming branches have completed

Order distribution process

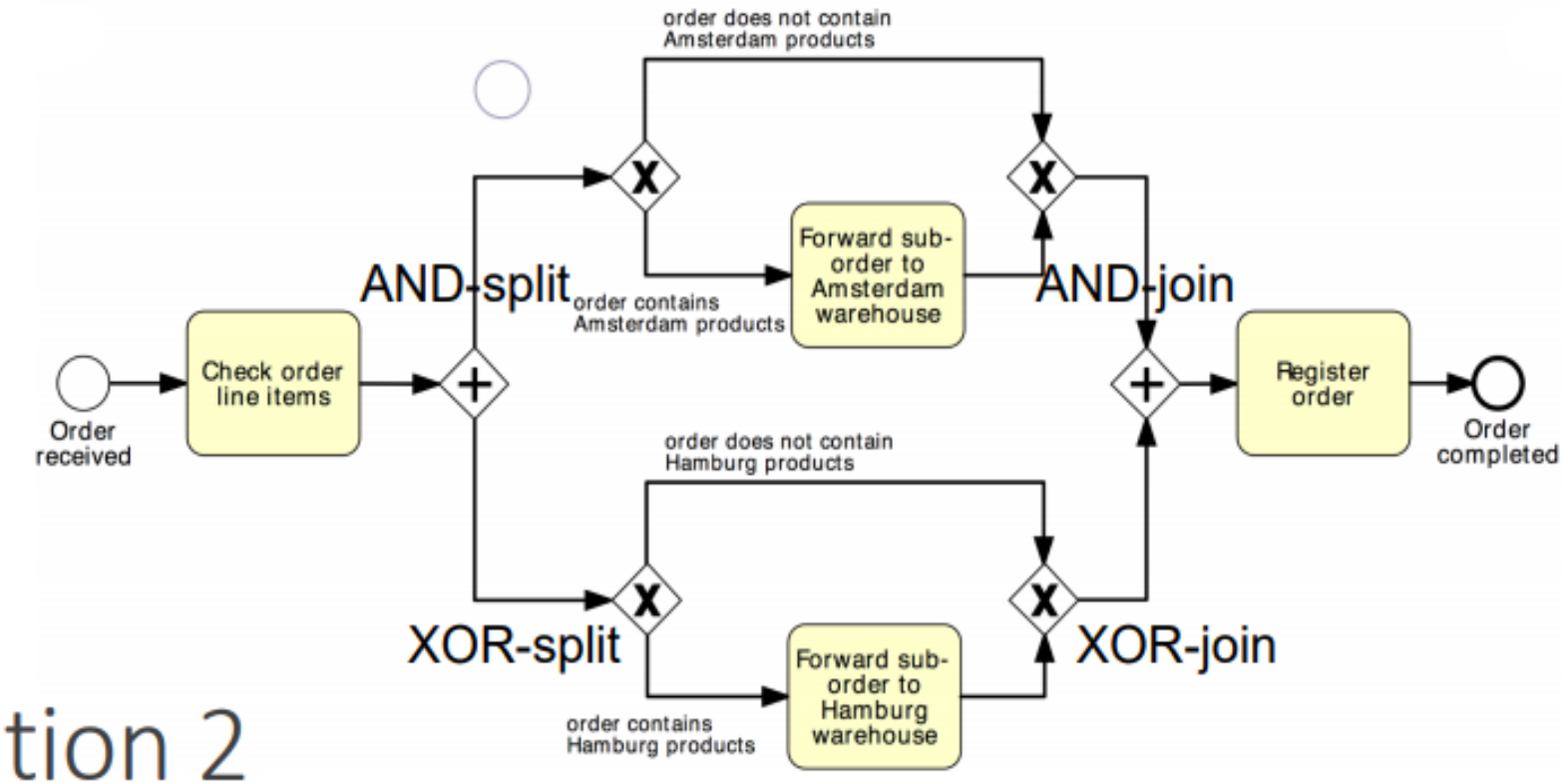
A company has **two warehouses** that store different products: Amsterdam and Hamburg. When an order is received, it is distributed across these warehouses: if some of the relevant products are maintained in Amsterdam, a sub-order is sent there; likewise, if some relevant products are maintained in Hamburg, a sub-order is sent there. Afterwards, the order is registered and the process completes.

Order distribution process



Solution 1

Order distribution process



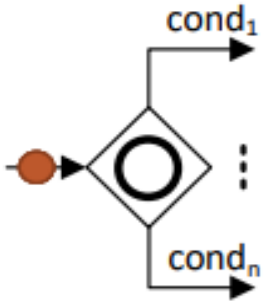
Solution 2

OR Gateway
Inclusive gateway

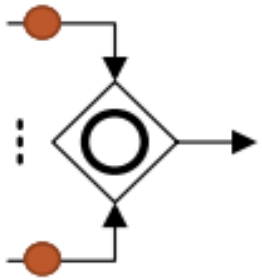
OR Gateway/Inclusive gateway

An *OR Gateway* provides a mechanism to create and synchronize **n out of m** parallel flows.

flows which can proceed **independently** of each other

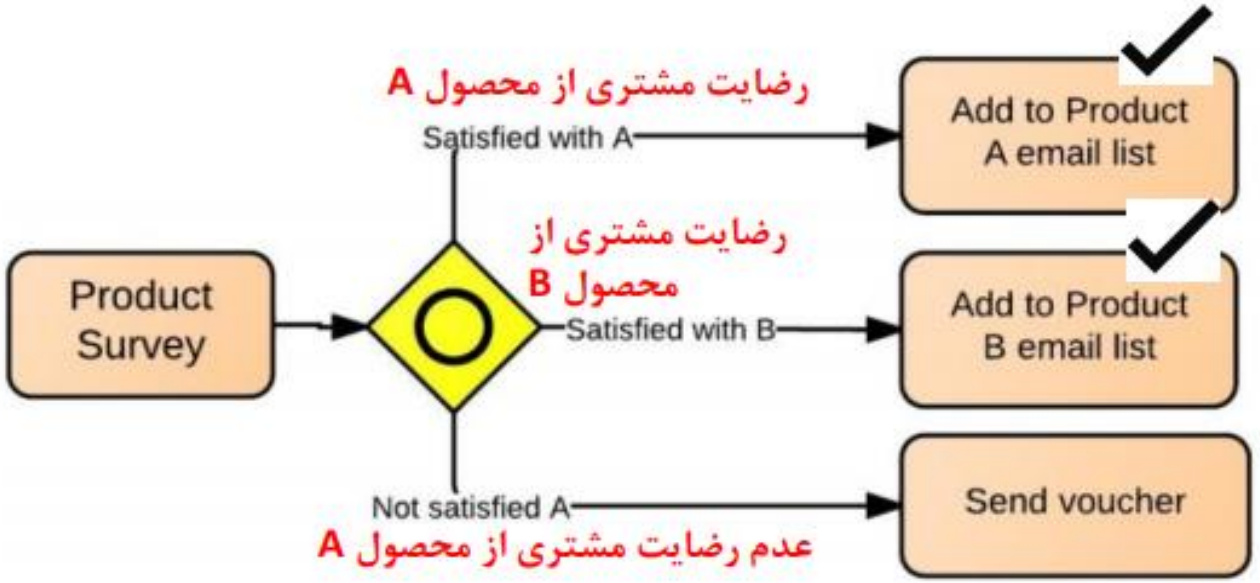


OR-split ➔ takes one or more branches depending on conditions

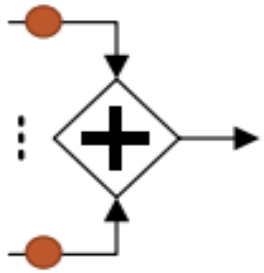


OR-join ➔ proceeds when all **active** incoming branches have completed

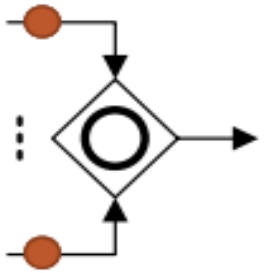
The example below shows an inclusive gateway that triggers different processes based on the way customers responded to a product survey. If the customer is satisfied with A, they are added to the Product A email list. If the customer is satisfied with B, they are added to the Product B email list. And if the customer is not satisfied with A, they are sent a voucher.



Difference between *AND-join & OR-join*



AND-join ➔ proceeds when **all (both activated or non-activated)** incoming branches have completed

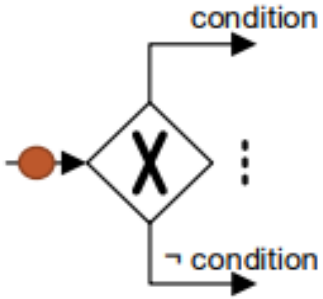


OR-join ➔ proceeds when all **active** incoming branches have completed

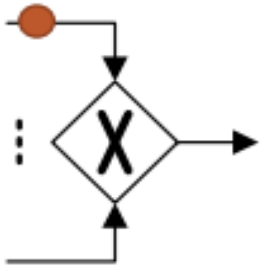
Exclusive (XOR) Gateway

Exclusive (XOR) Gateway

An *XOR Gateway* captures decision points (XOR-split) and points where alternative flows are merged (XOR-join)



XOR-split ➔ takes **one** outgoing branch



XOR-join ➔ proceeds when **one** incoming branch has completed

Exclusive Gateway (XOR)

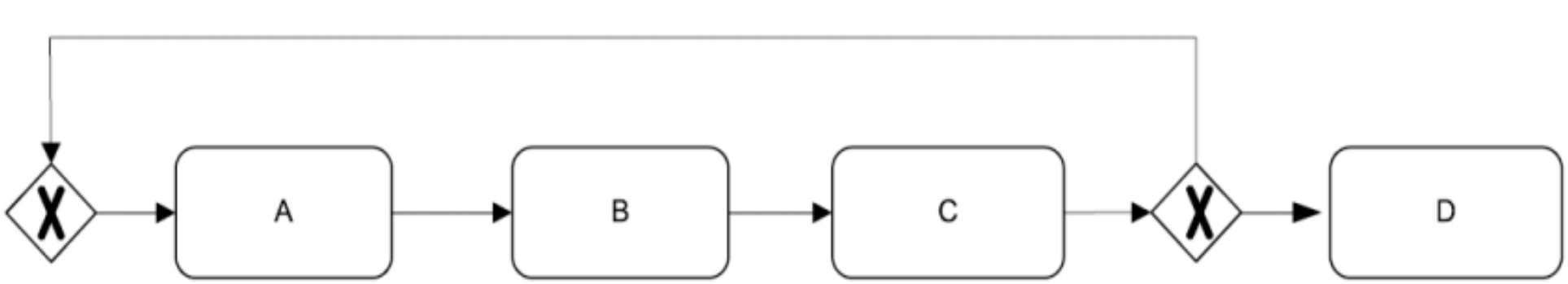
1. When the execution arrives at this gateway, **all outgoing sequence flows are evaluated** in the order in which they have been defined.
2. The sequence flow which condition evaluates to 'true' (**or** which doesn't have a condition set, conceptually having a 'true' value defined on the sequence flow) is selected for continuing the process.
3. **only one** sequence flow is selected
4. In case multiple sequence flow have a condition that evaluates to 'true', the **first one** is selected for continuing the process

Note

A gateway **without an icon inside** it defaults to an exclusive gateway

Arbitrary Cycles

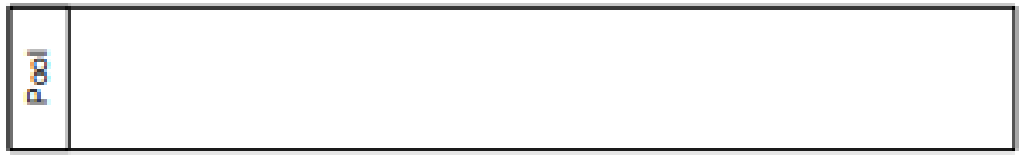
An *arbitrary cycle* is a point in a process model where one or more activities can be executed **repeatedly**.



Organizational Elements in BPMN – Pools & Lanes

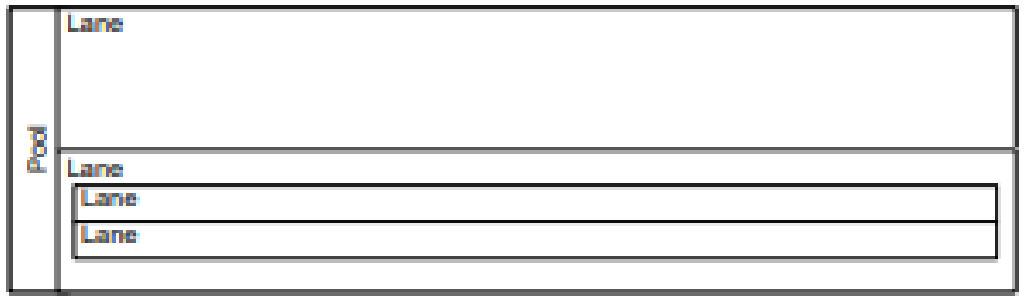
Pool

Captures a resource class. Generally used to model a **business party** (e.g. a whole company)



Lane

A *resource sub-class* within a pool. Generally used to model departments (e.g. shipping, finance), internal roles (e.g. Manager, Associate), software systems (e.g. ERP, CRM)



Lanes can be nested

هفت راهنما برای مدلسازی

G1 Use as **few elements** in the model as possible

G2 **Minimize** the overlaps between routing paths per element

G3 Use one **start** and one end **event** if possible

G4 Model as **structured** as possible

G5 **Avoid**, where possible, **OR** gateway elements

G6 Use **verb-object** activity labels

G7 **Decompose** a model with more than 30 elements

Guidelines: Naming rules

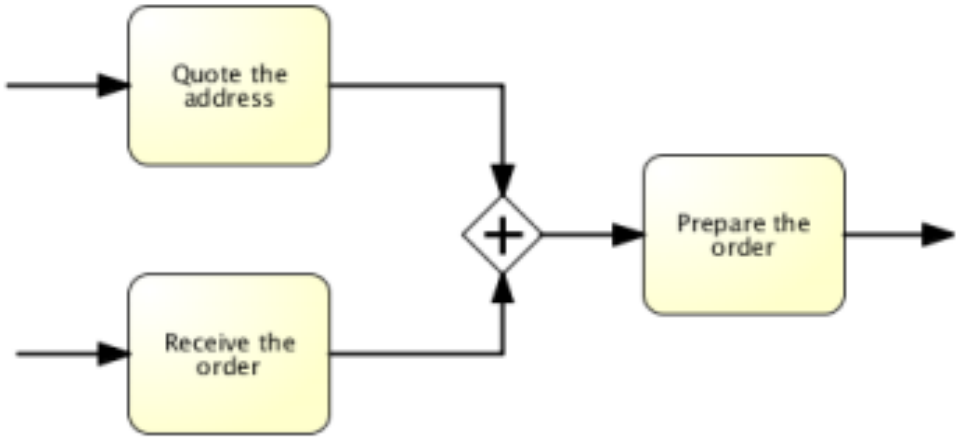
1. Give a name to every event and task
2. **For tasks:** verb followed by business object name and possibly complement
   - Issue Driver Licence, Renew Licence via Agency
3. **For message events:** object + past participle
   - Invoice received, Claim settled
4. **Be specific:** Avoid generic verbs such as Handle, Record…
5. Label each XOR-split with a condition
   - Policy is invalid, Claim is inadmissible

Some common mistakes in BPMN modeling

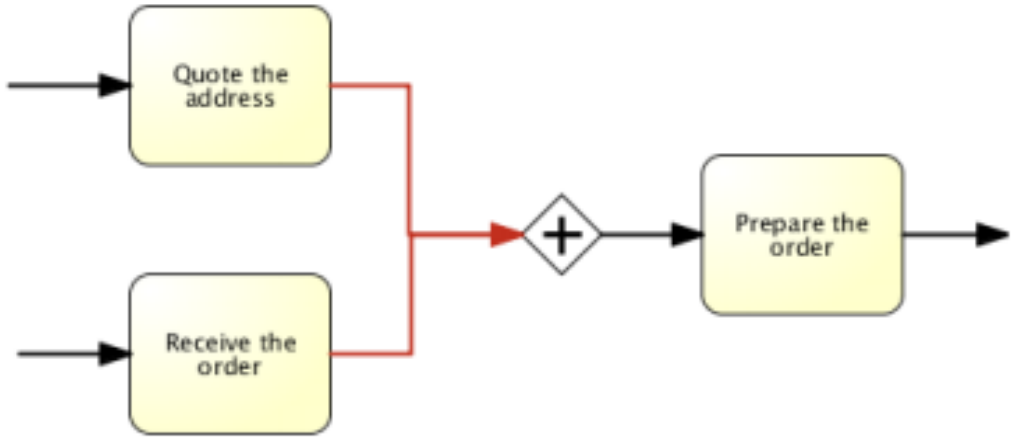
Improper structuring

Guidelines: edge overlays

Connectors (edges) should not overlap. If this rule is violated, the model might become unclear.



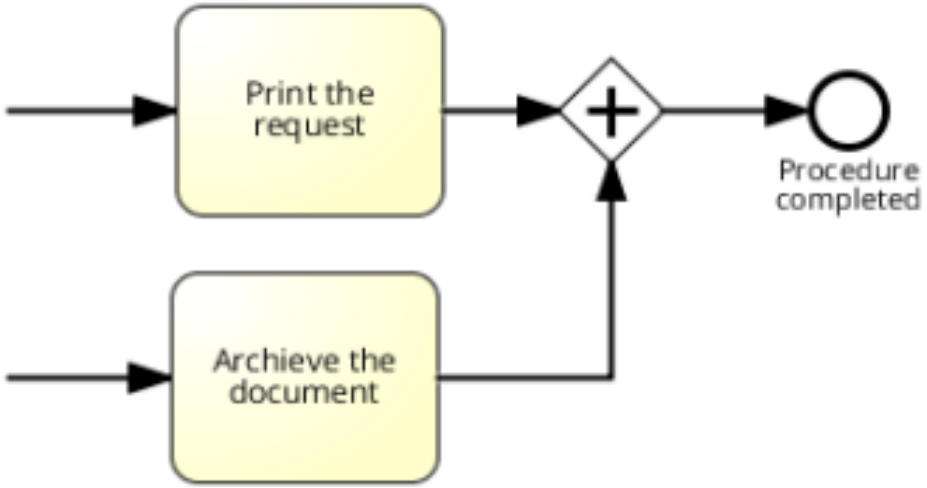
Positive example



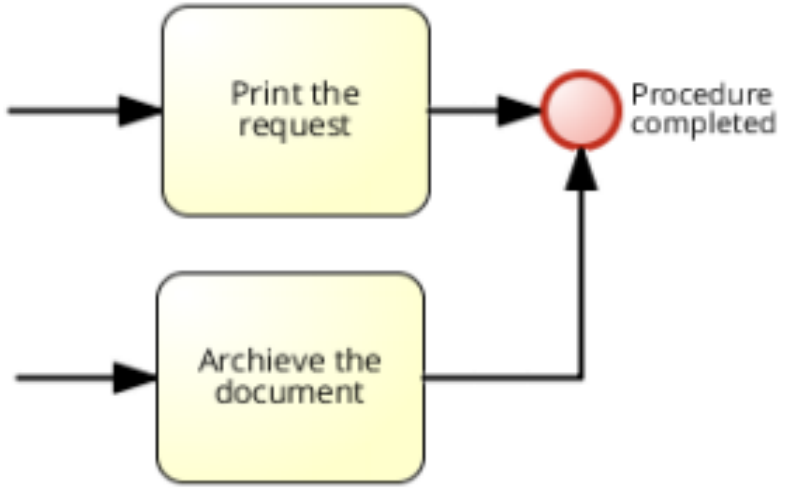
Negative sample

Usage of explicit joins before end events

1. Joins should be modeled explicitly using gateways. This is important for clarity.
2. Elements (with the exception of gateways) should **only have one** incoming sequence flow. By modeling more than one incoming sequence flow to an element, the diagram may become harder to read. Failure to do so can result in errors such as multi-merges or deadlocks.



Positive example

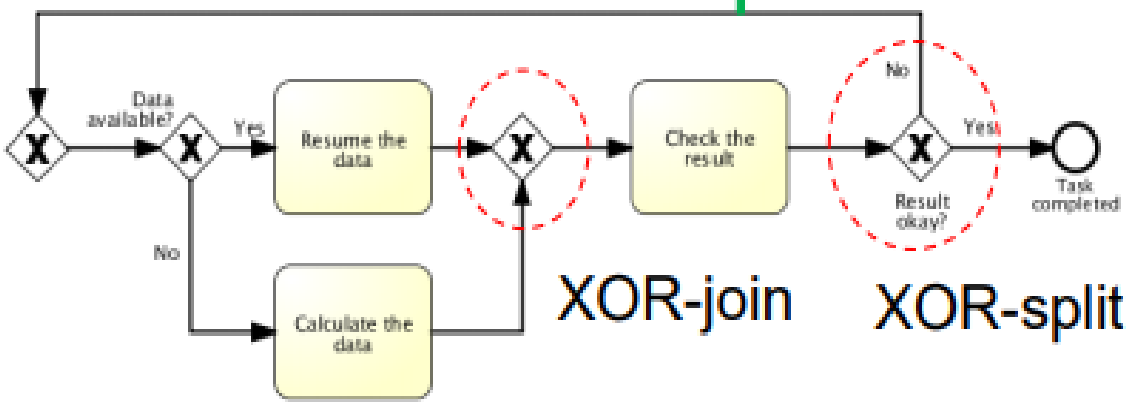


Negative sample

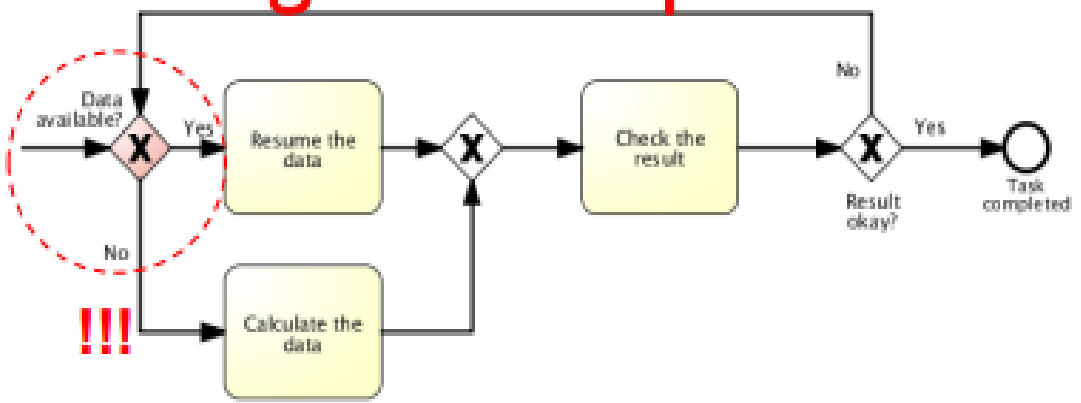
Absence of split and join behavior on one element

1. The strict separation of branched and merging gateways should be observed, since it leads to better transparency and clarity in the process.
2. In case a sequence flow is merged and then immediately branched again, separate gateways need to be modeled for both actions. A gateway should not have a **jointing and splitting** function **at the same time**.

Positive example

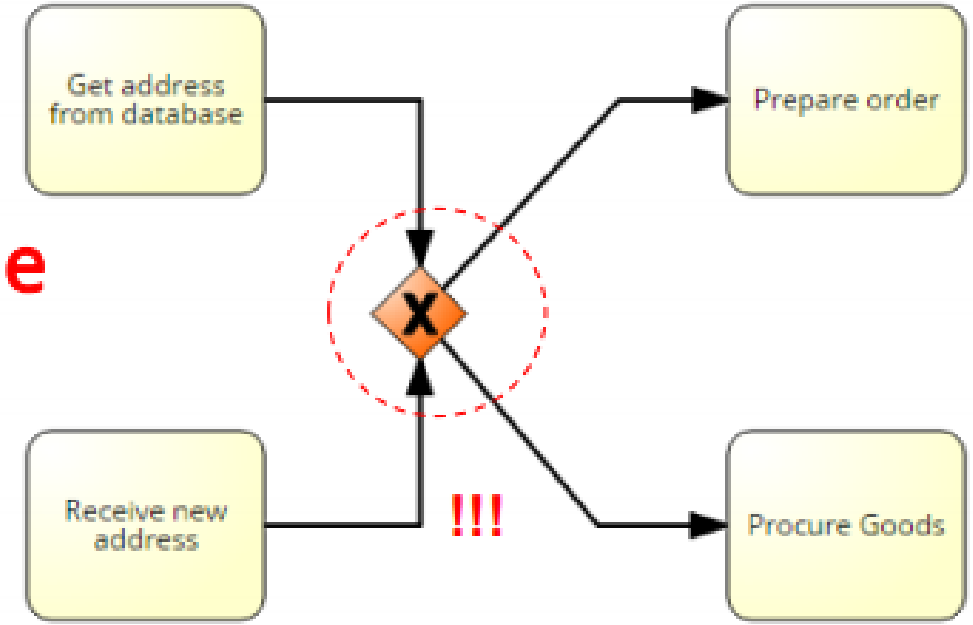


Negative sample



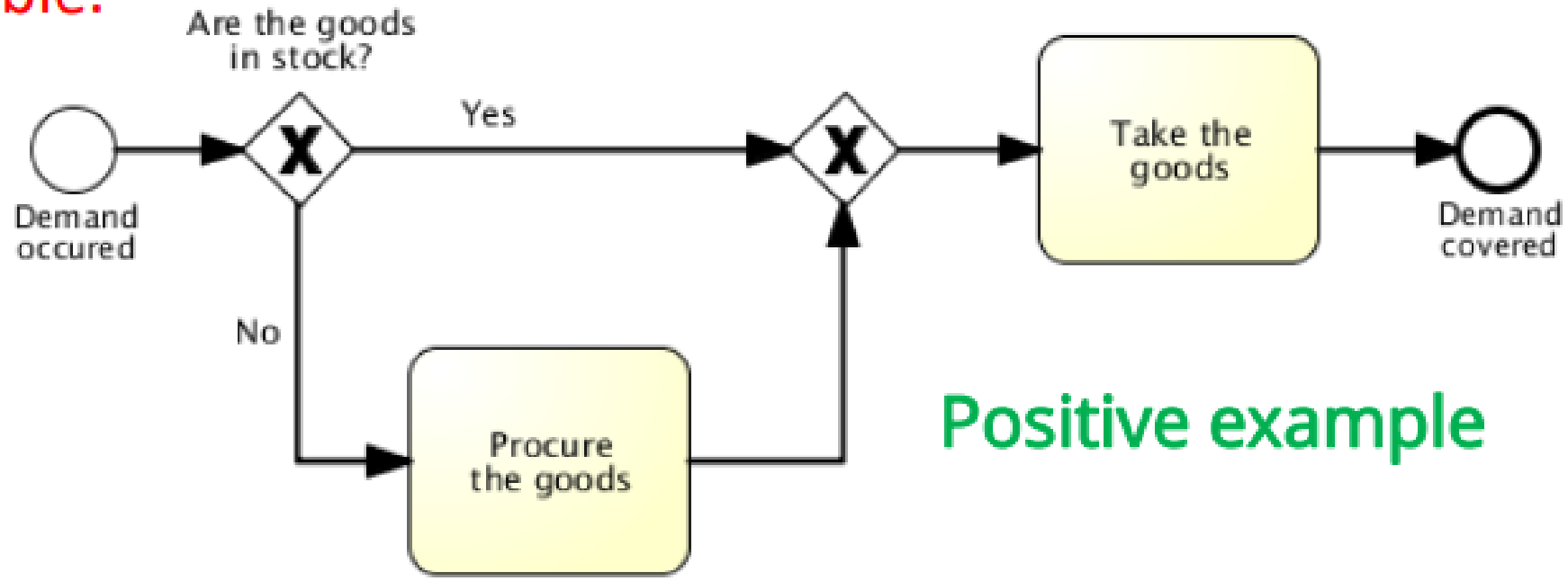
Absence of split and join behavior on one element

Negative sample



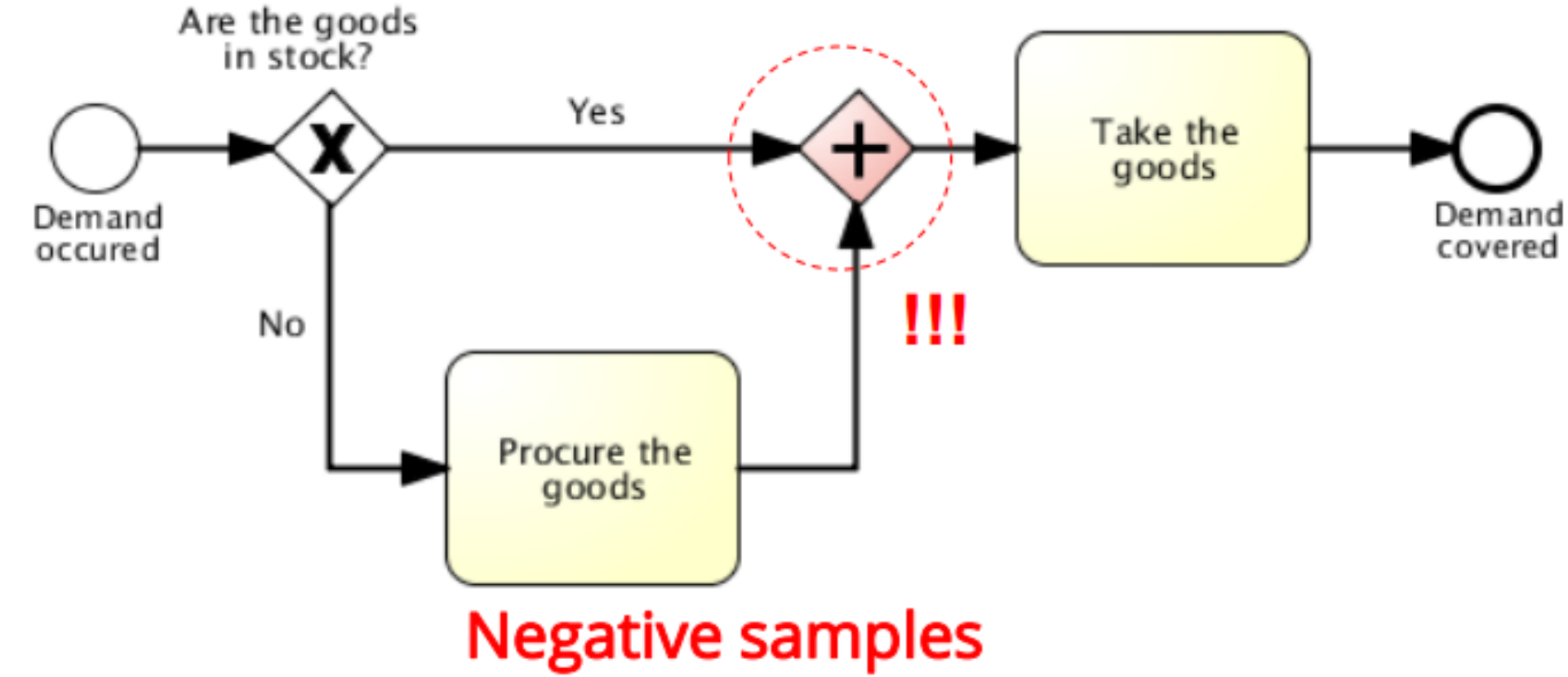
Deadlocks

Deadlocks arise due to a wrong combination of gateways and block the further process procedure. A process with a deadlock is not executable.

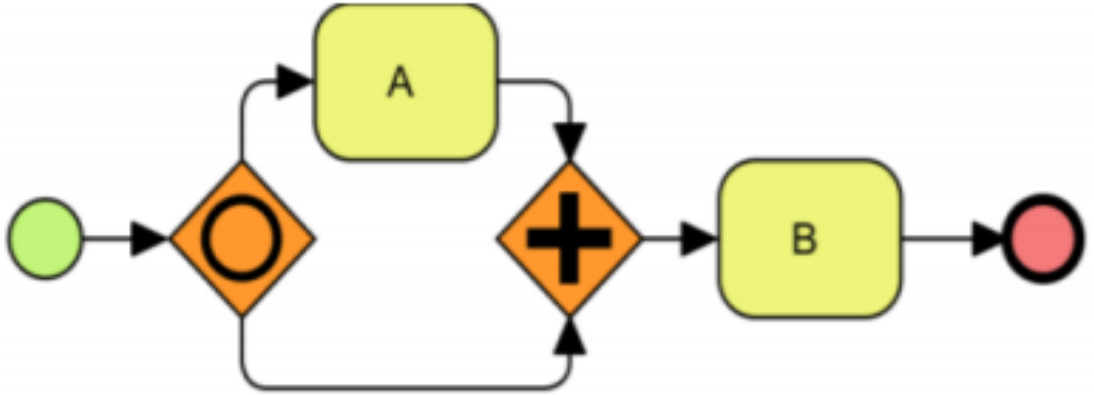


Positive example

Deadlocks

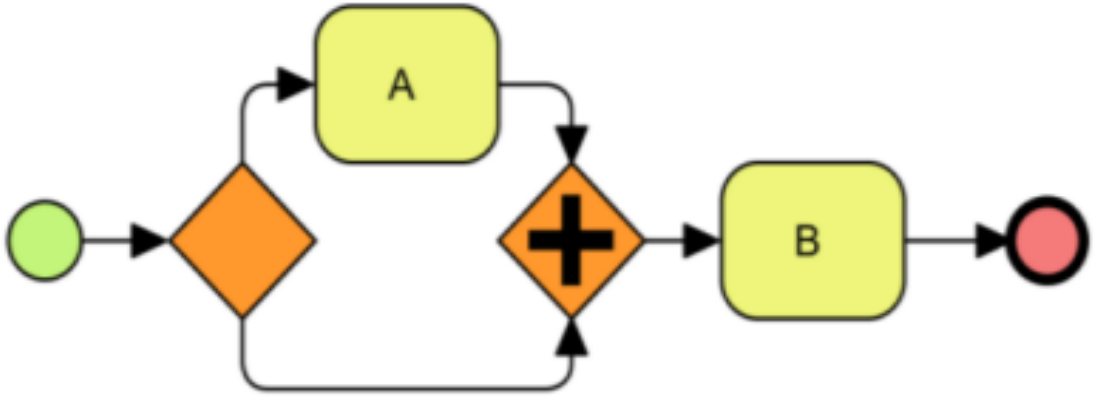

Are the goods in stock?
Yes
No
Demand occured
Take the goods
Demand covered
Procure the goods
!!!
Negative samples

Deterministic vs. Non-deterministic deadlock



Non-deterministic deadlock

ممکن است حالت بن بست
اتفاق بیافتد



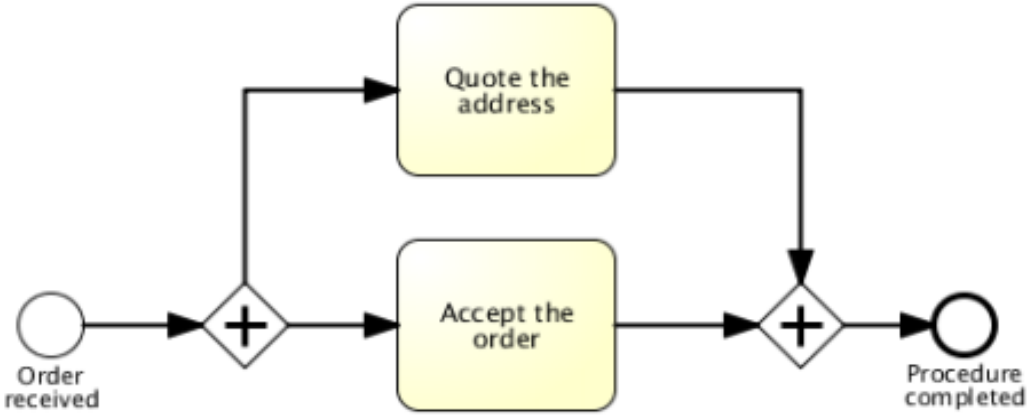
Deterministic deadlock

در تمام حالات بن بست اتفاق می افتد

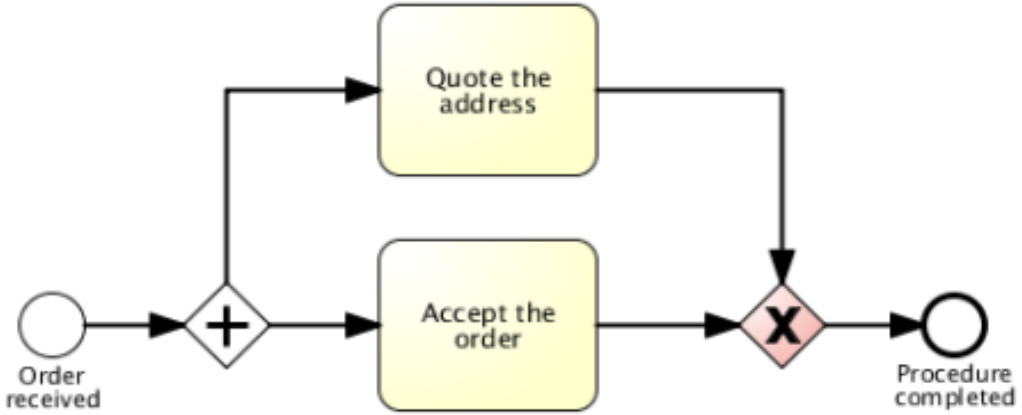
Guidelines: Absence of multi-merges

Multi-merges are the opposite of deadlocks. They often occur when Gateways are used incorrectly and lead to multiple sequence flows merging at a single-flow element. Avoiding this is necessary, since the process often shows an unexpected behavior within a multi-merge.

Positive example

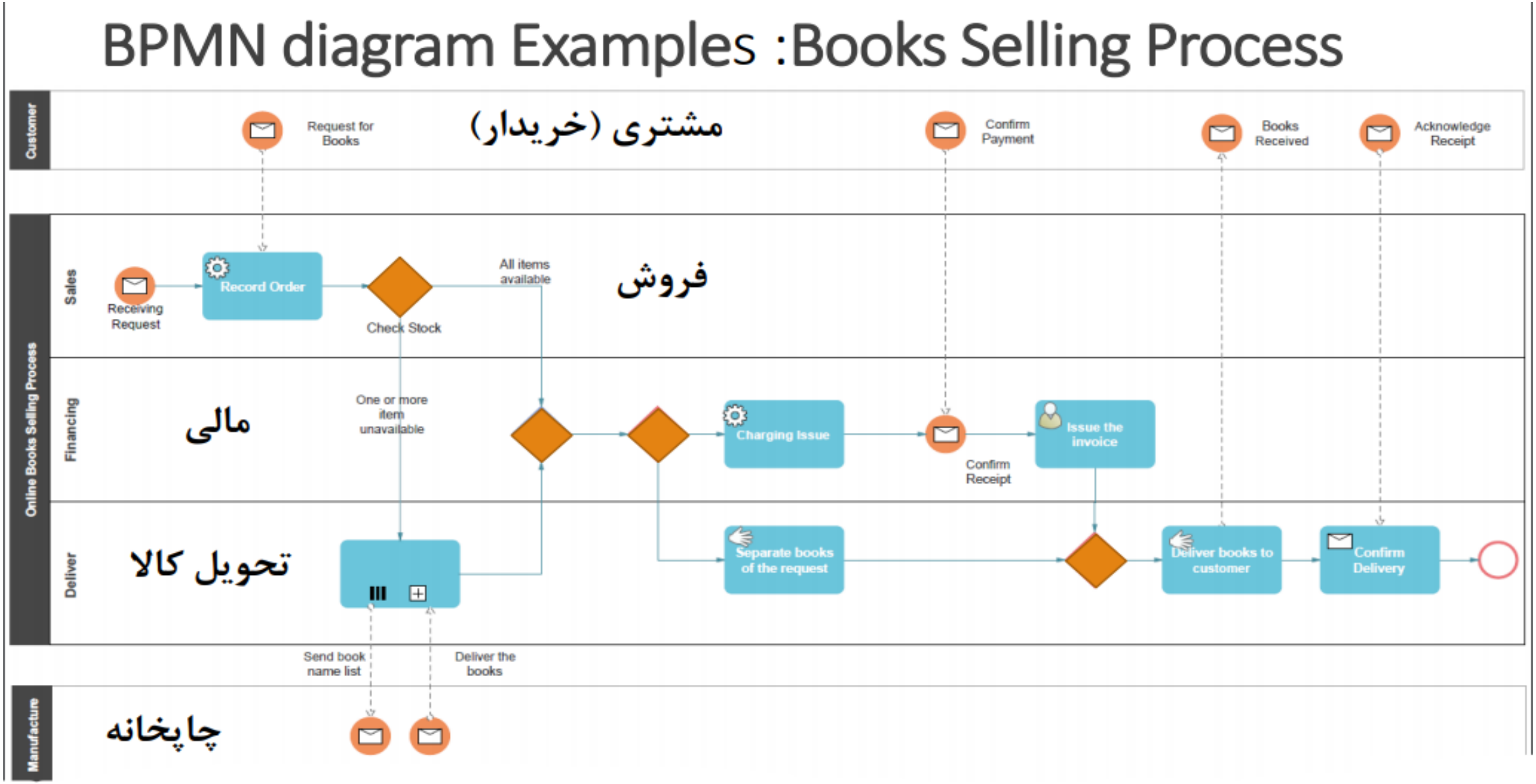


Negative sample



Lack of synchronization

BPMN diagram Examples :Books Selling Process

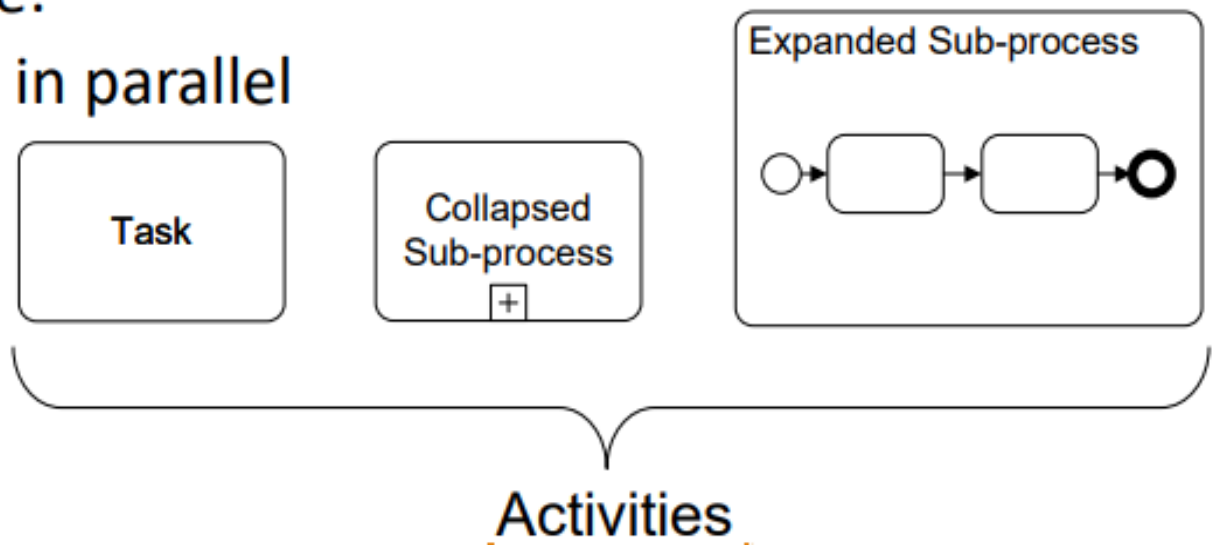


Process decomposition

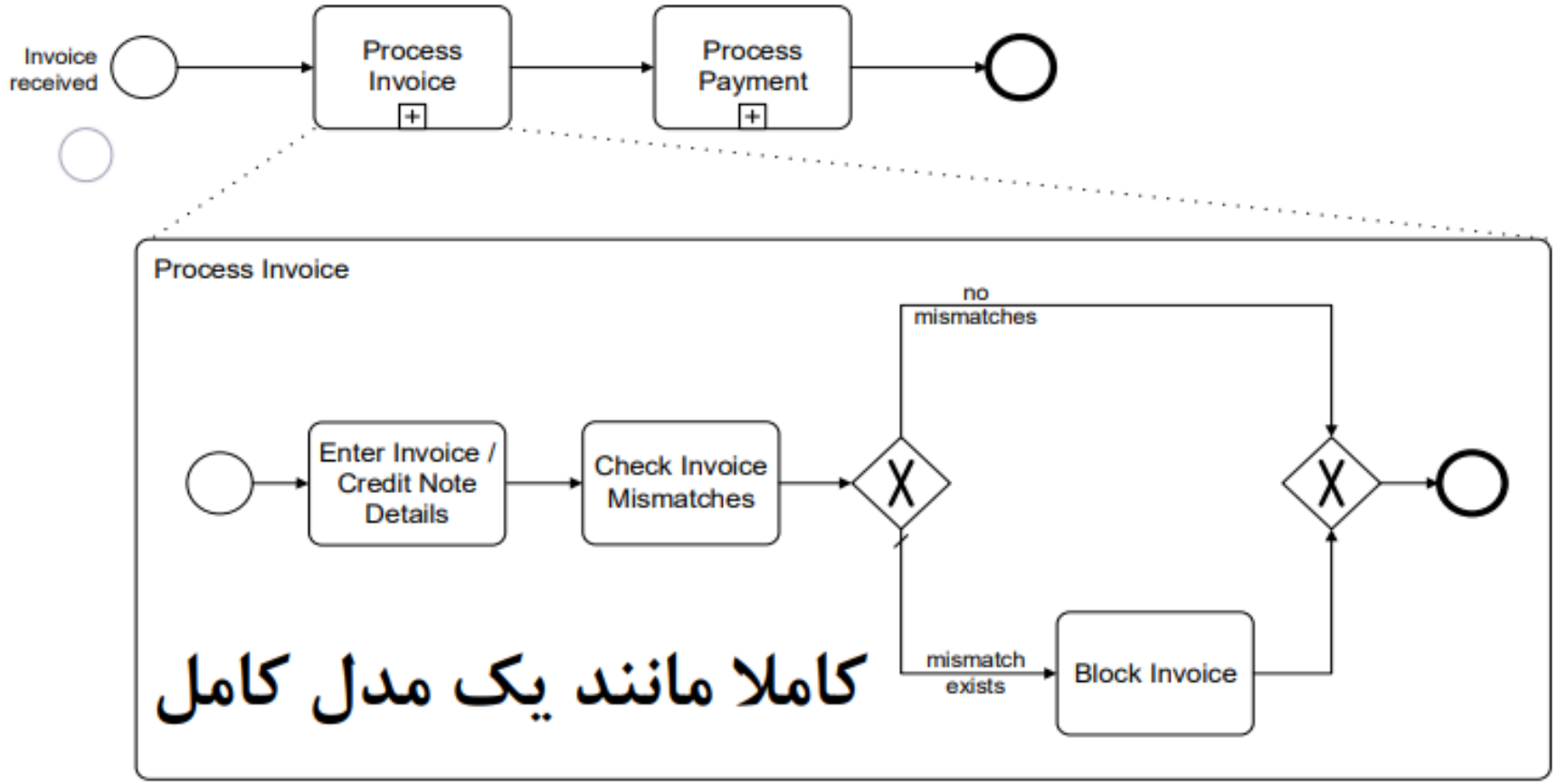
An activity in a process can be decomposed into a “sub-process”

Use this feature to:

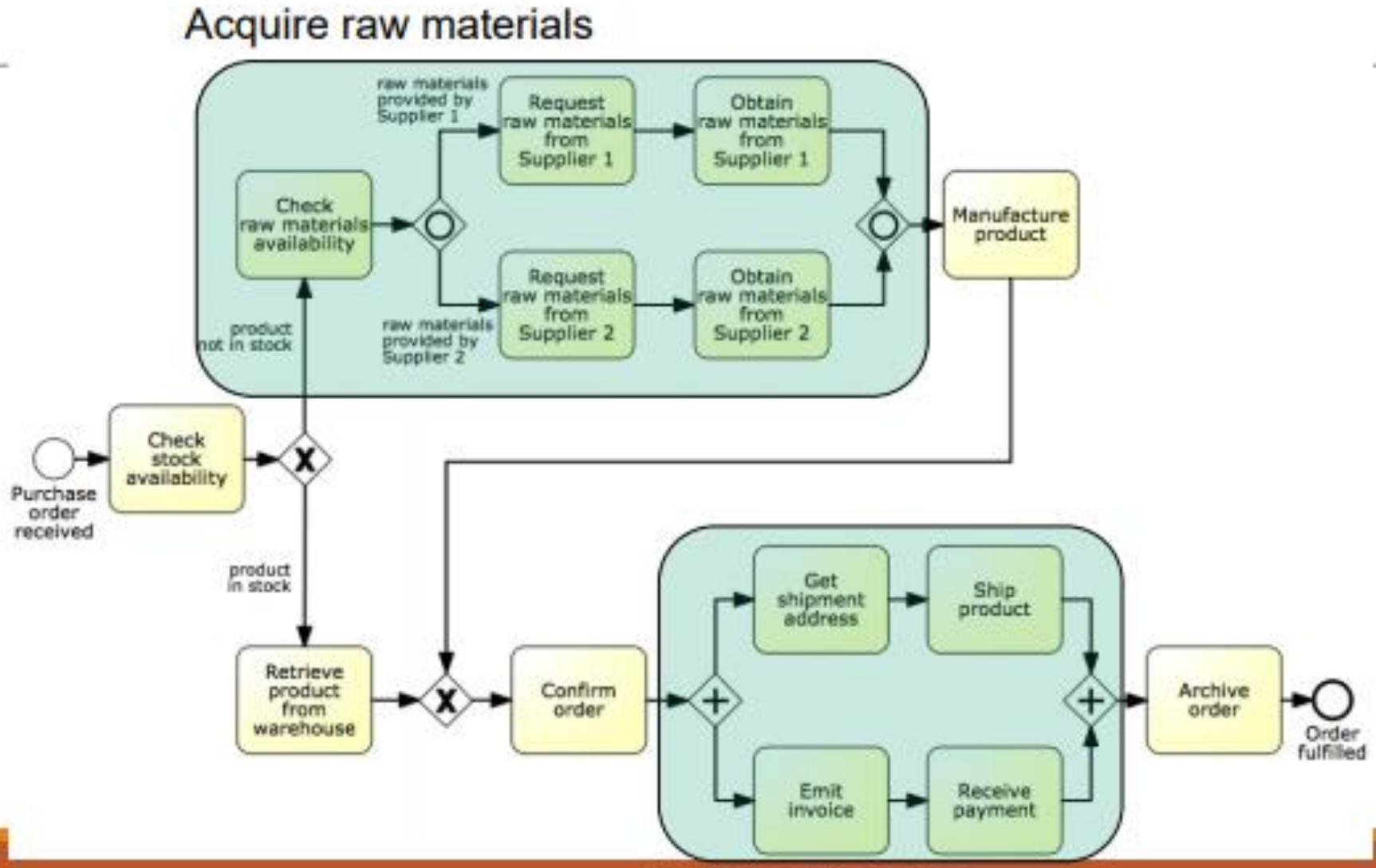
1. Improve understanding by breaking down large models
2. Identify parts that should be:
   - executed multiple times in parallel
   - repeated
   - interrupted, or
   - compensated



Example: Sub-Process

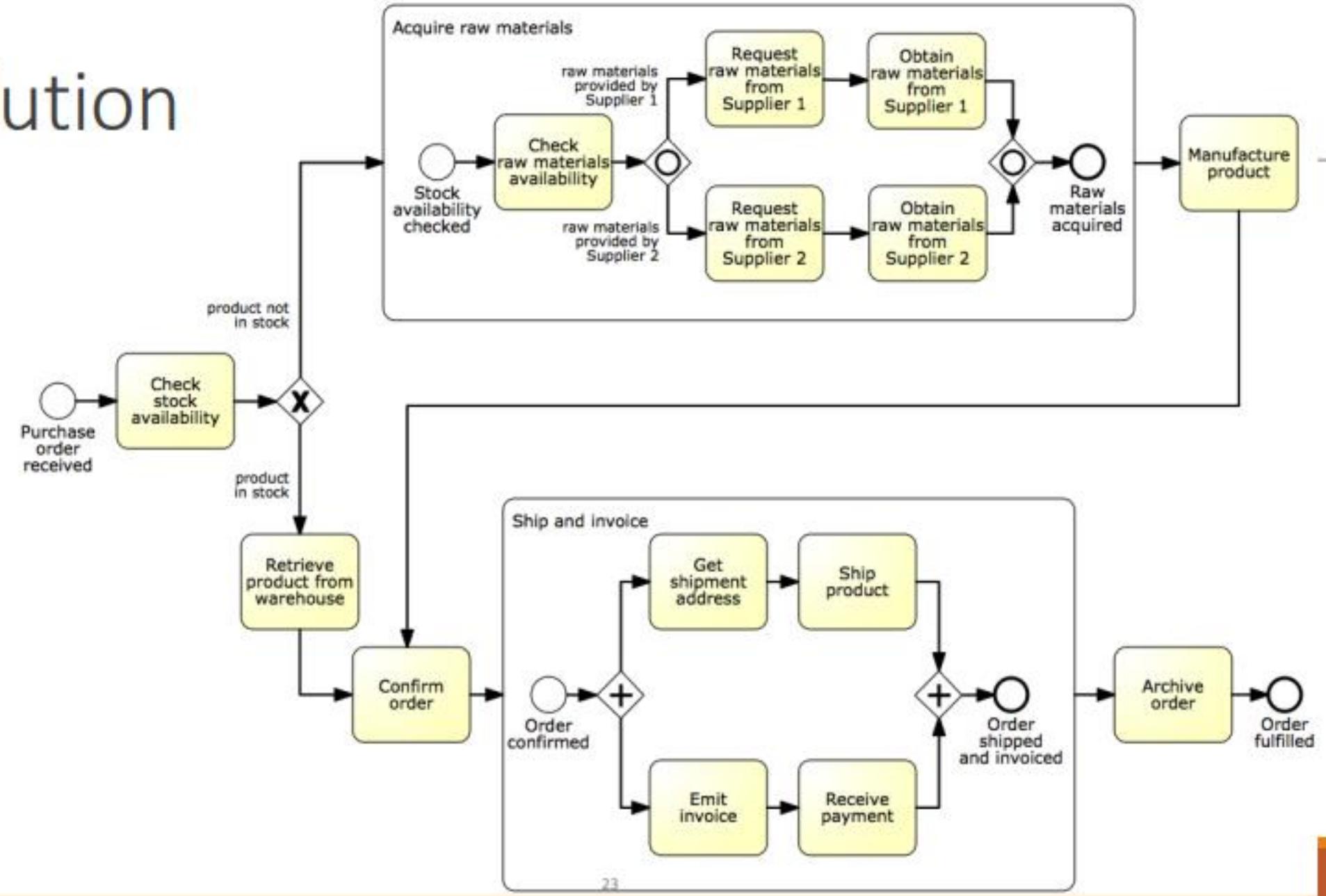


Identify possible sub-processes

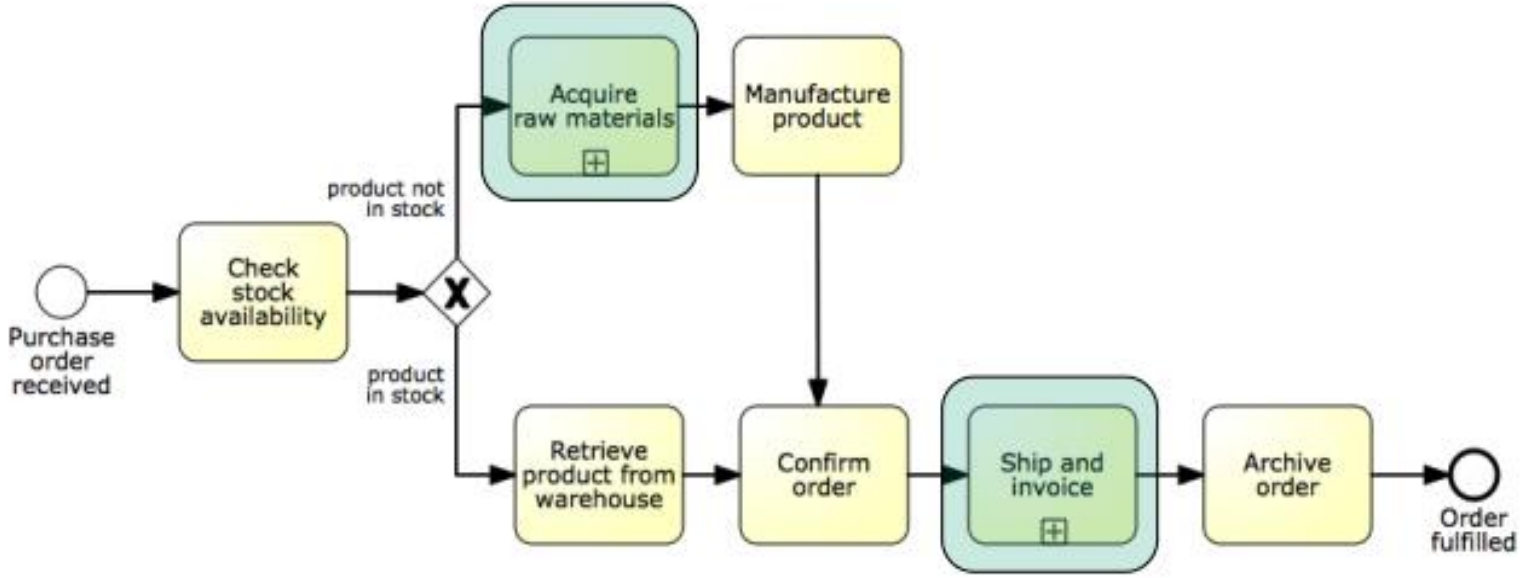


Ship and invoice

Solution

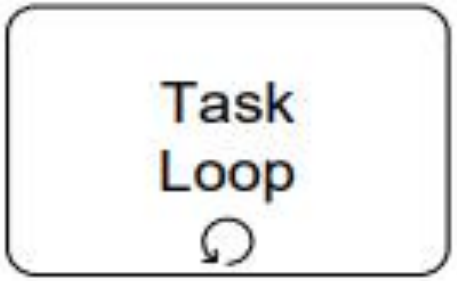

Acquire raw materials
raw materials provided by Supplier 1
Request raw materials from Supplier 1
Obtain raw materials from Supplier 1
Stock availability checked
Check raw materials availability
raw materials provided by Supplier 2
Request raw materials from Supplier 2
Obtain raw materials from Supplier 2
Raw materials acquired
Manufacture product
product not in stock
Purchase order received
Check stock availability
product in stock
Retrieve product from warehouse
Confirm order
Ship and invoice
Get shipment address
Ship product
Order confirmed
Emit invoice
Receive payment
Order shipped and invoiced
Archive order
Order fulfilled

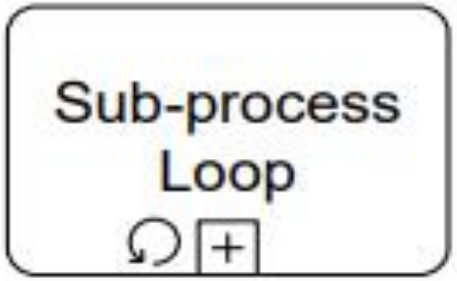
The refactored model



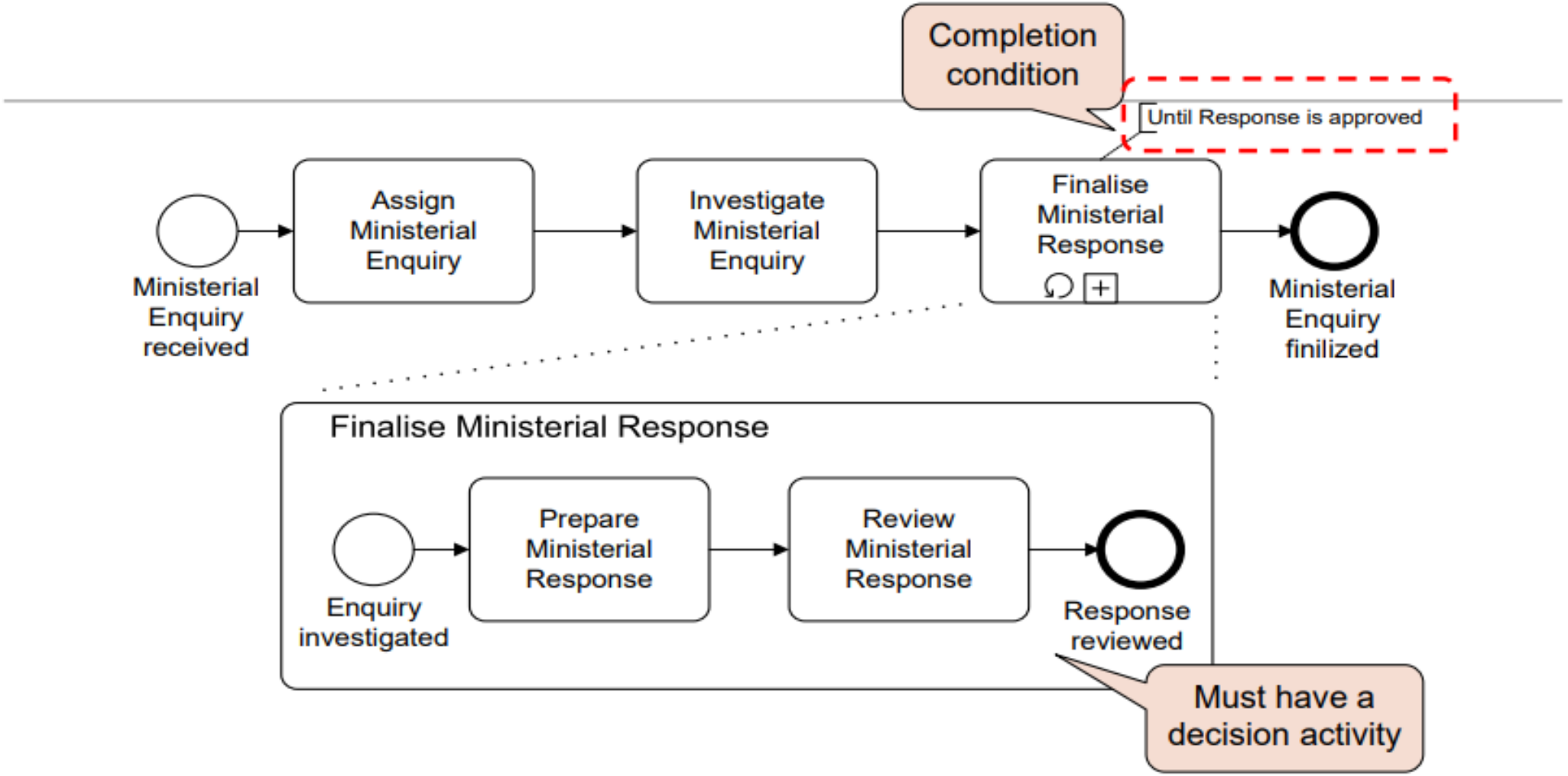
Block-structured repetition: Loop Activity

BPMN also provides the *loop activity* construct to allow the repetition of a task or sub-process





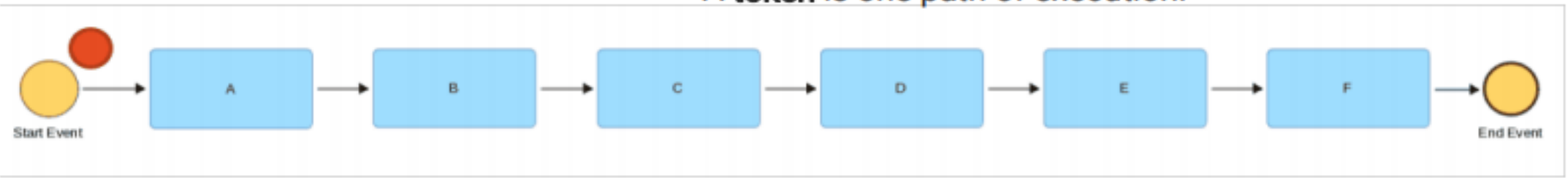
Example: block-structured repetition



Token in BPMN

1. A token is a **theoretical concept** that is used as an aid to define the behavior of a Process models (e.g. Petri Net).
2. The behavior of BPMN elements can be defined by describing how they interact with a token as it "traverses" them.

A **token** is one path of execution.



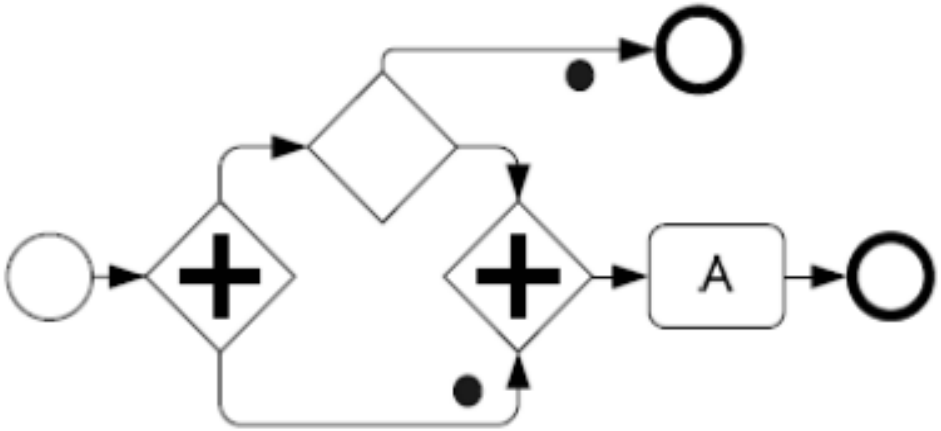
A token is a **pointer** that is associated with an activity.

Token in BPMN

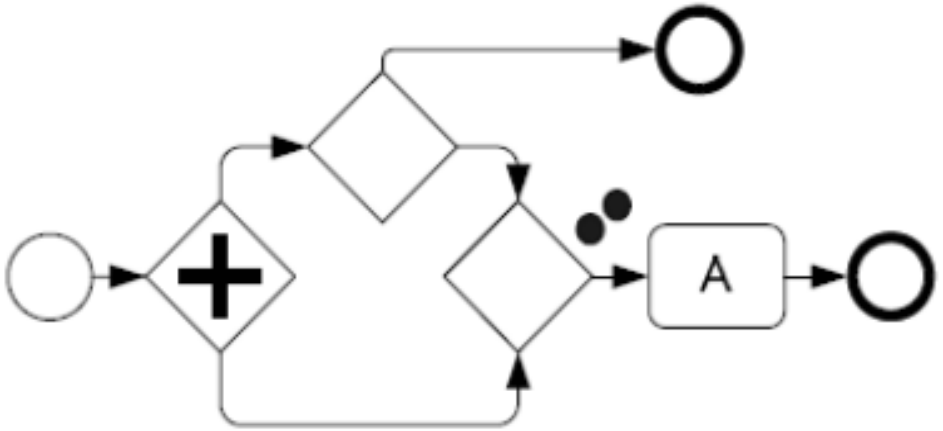
Token is born when a **Start Event** is triggered (e.g. by receiving a Message, when some condition becomes true or for a plain Start Event simply when some employee starts the process).

Token then follows the **Sequence Flow** (**it cannot use Message Flows or other arrows**) and flows through a process (passing through elements such as Gateways or Tasks/Sub-processes) right till the End Event where it is consumed.

Detecting structural Anomalies using Tokens



Deadlock

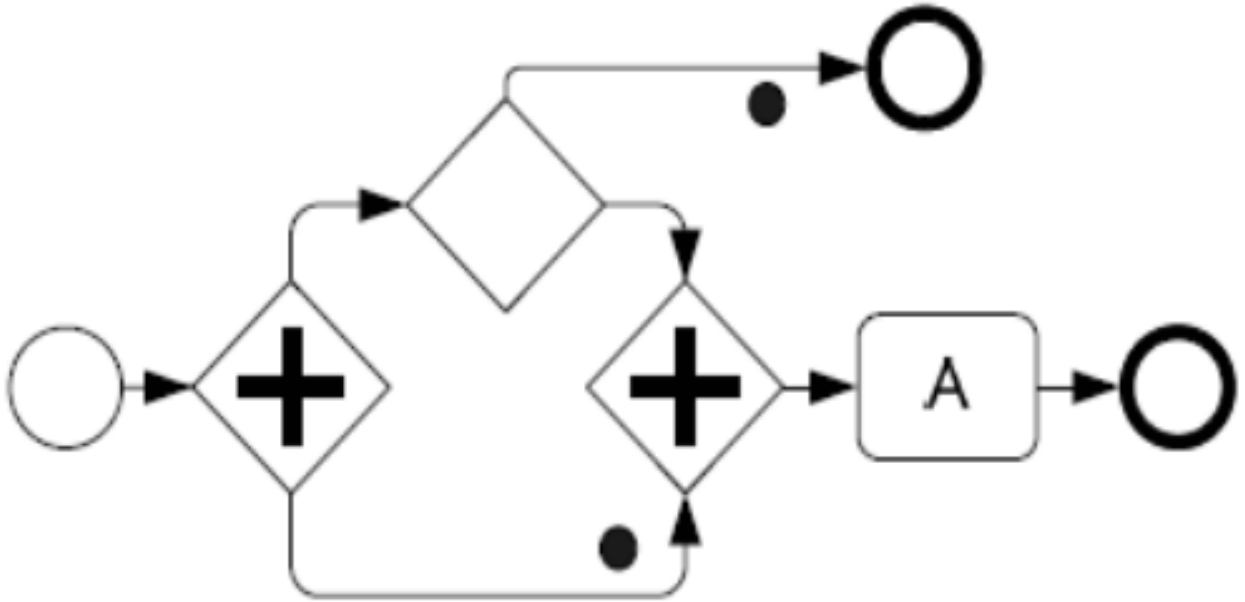


Lack of synchronization

Detecting structural Anomalies using Tokens

Deadlock = some tokens are **blocked**; not enough tokens for execution

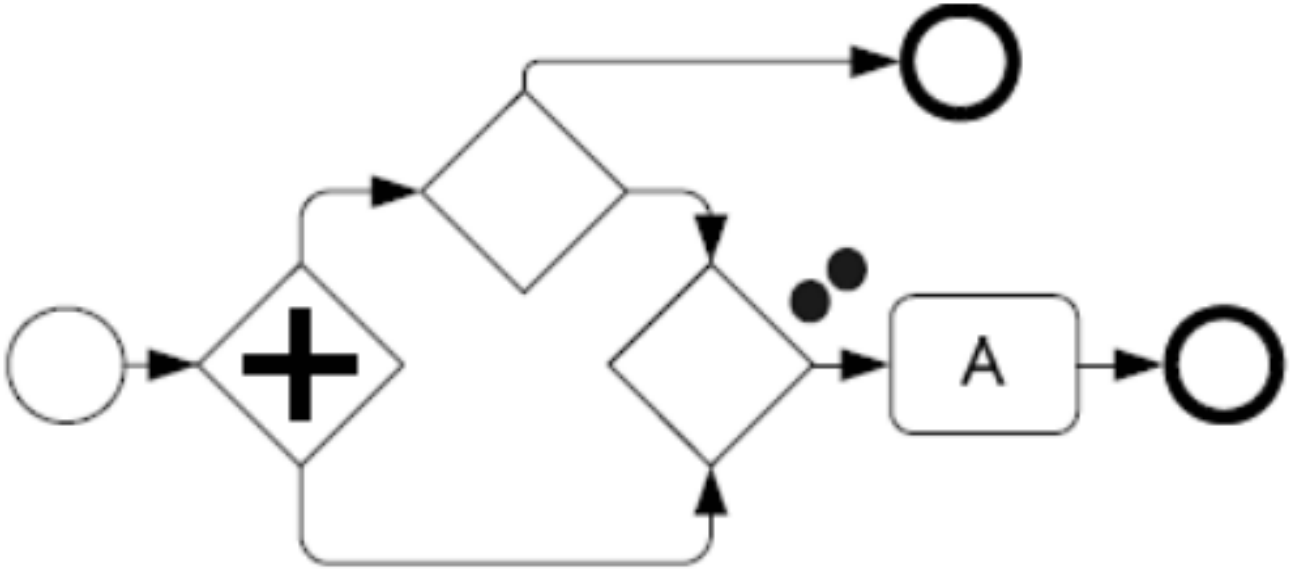
– Subsequent task does not get executed



Detecting structural Anomalies using Tokens

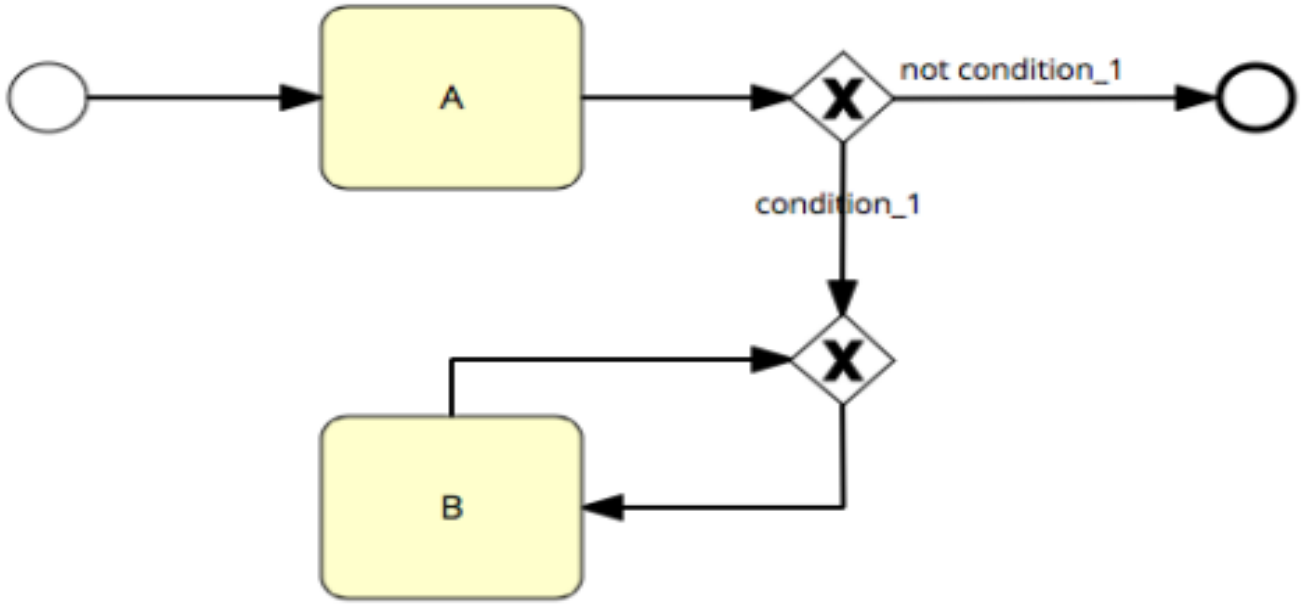
Lack of synchronization = concurrent tokens don't get synchronized, multiple tokens on a single edge

– Subsequent task get executed twice (or more)



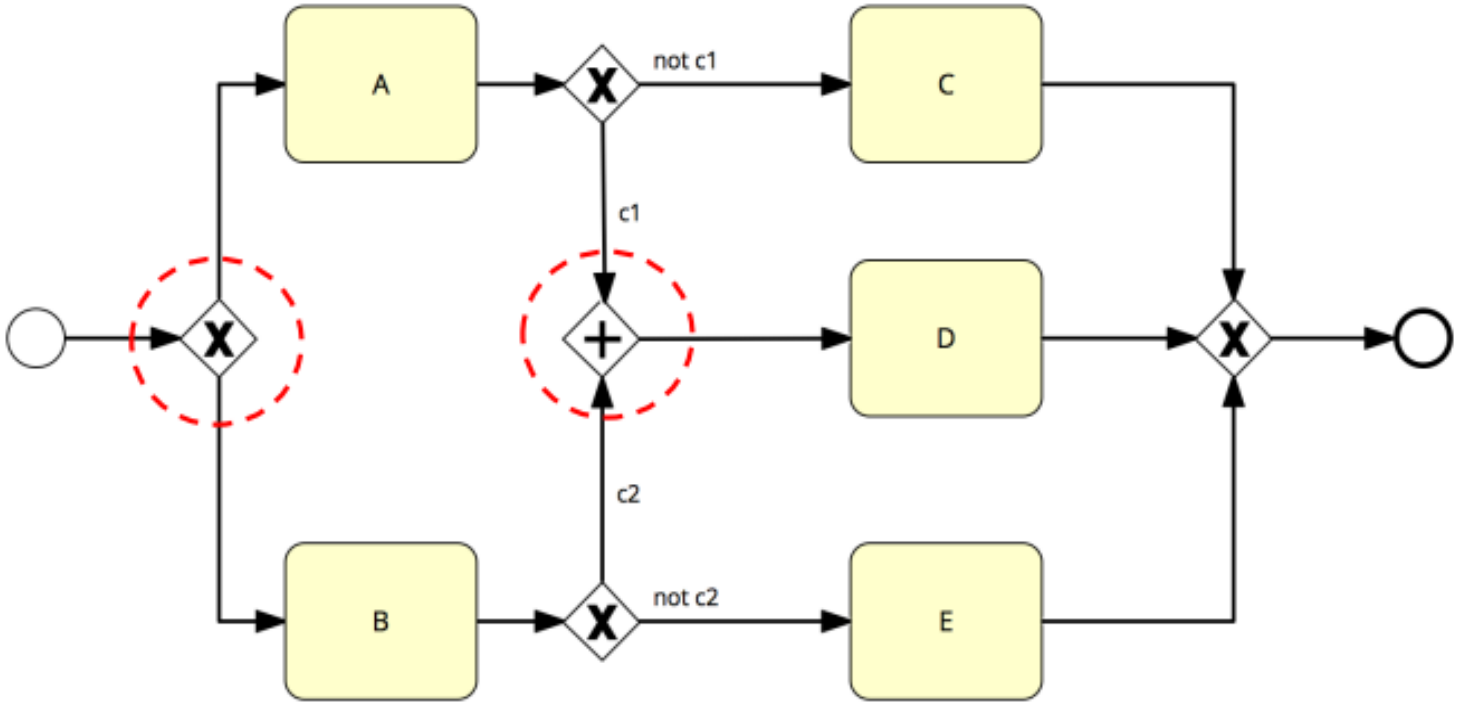
No option to complete

If condition_1 is true, the instance cannot complete and activity B will be repeated forever (**livelock**)



No option to complete

If c1 is true after executing A, or c2 is true after executing B, the instance cannot complete (**non-deterministic deadlock**)

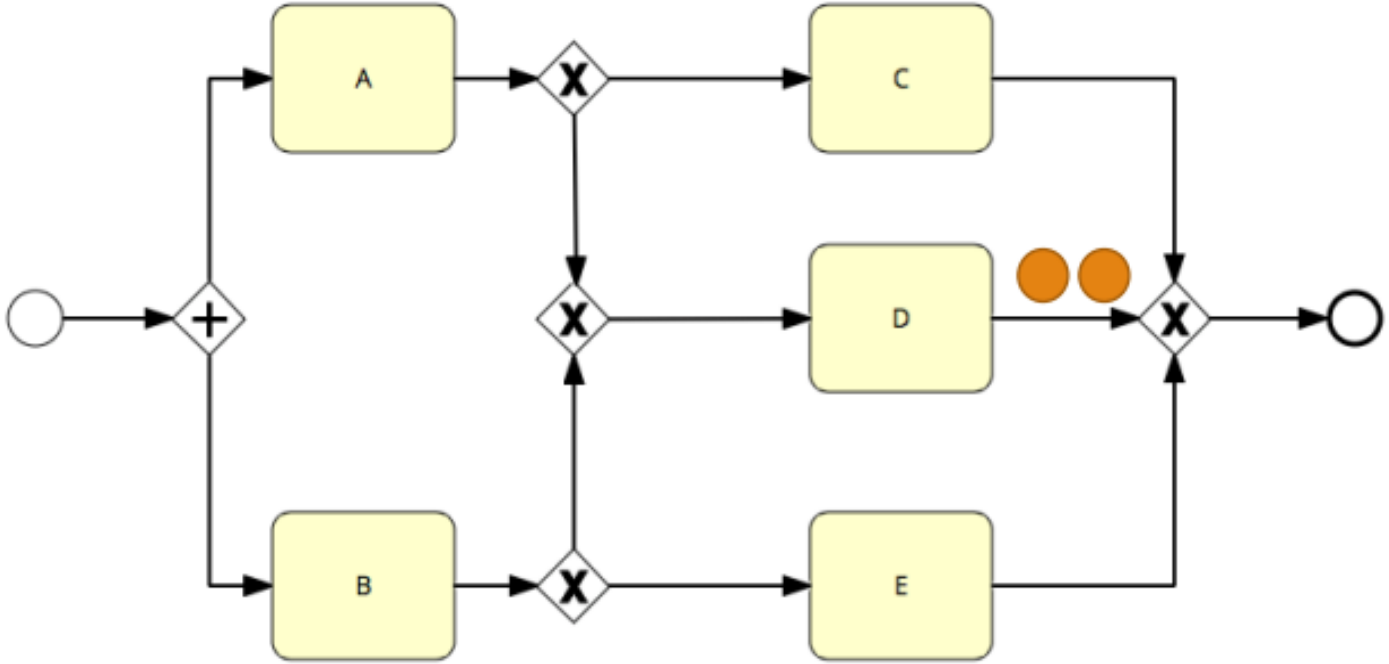


dead activity?

Note: this model also suffers from a **dead activity** (D)

no proper completion

At the moment of completion, there will be two tokens in the end event (**lack of synchronization**)



A situation where there is **more than one token** on some sequence flow that it should be.

Soundness Checking

1. A **process model** that is free of deadlock and lack of synchronization problems is called **sound**.

2. Model Checking (Soundness Checking)

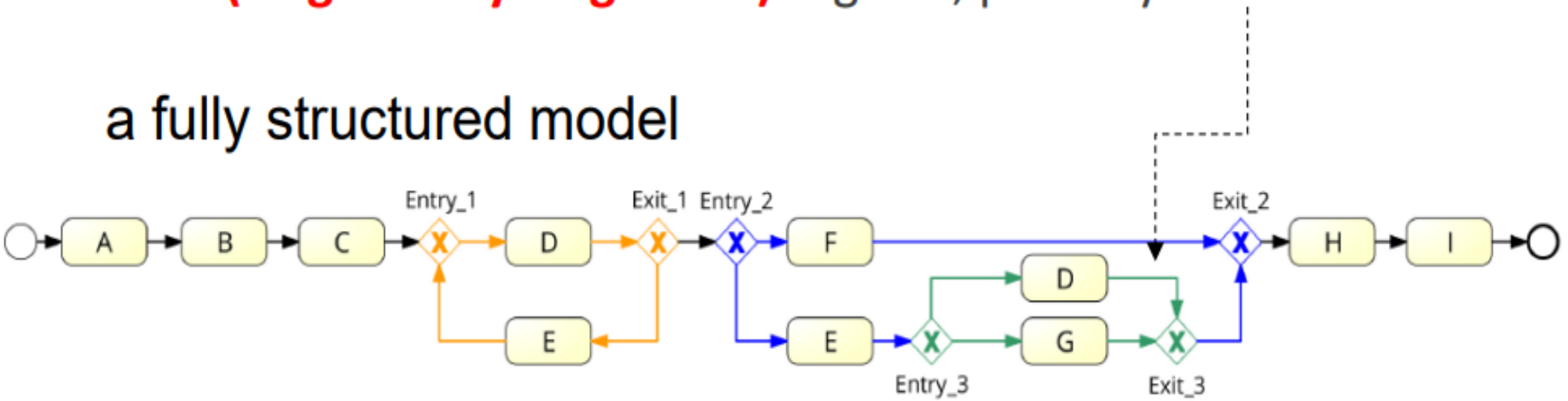
Lack of Synchronization

1. Lack of synchronization is a situation where **two or more parallel branches are not correctly synchronized** in the process model.
2. While this behavior is **not an error** by default, it is introduced into the model unconsciously in most cases.

Structured models

A model is considered structured if it is made up **only** of **SESE (single entry single exit)** regions, possibly nested.

a fully structured model



Structured is “better”

– Easier to understand

– Easier to analyze

– Easier to automatically layout

– Easier to abstract (zoom-out)

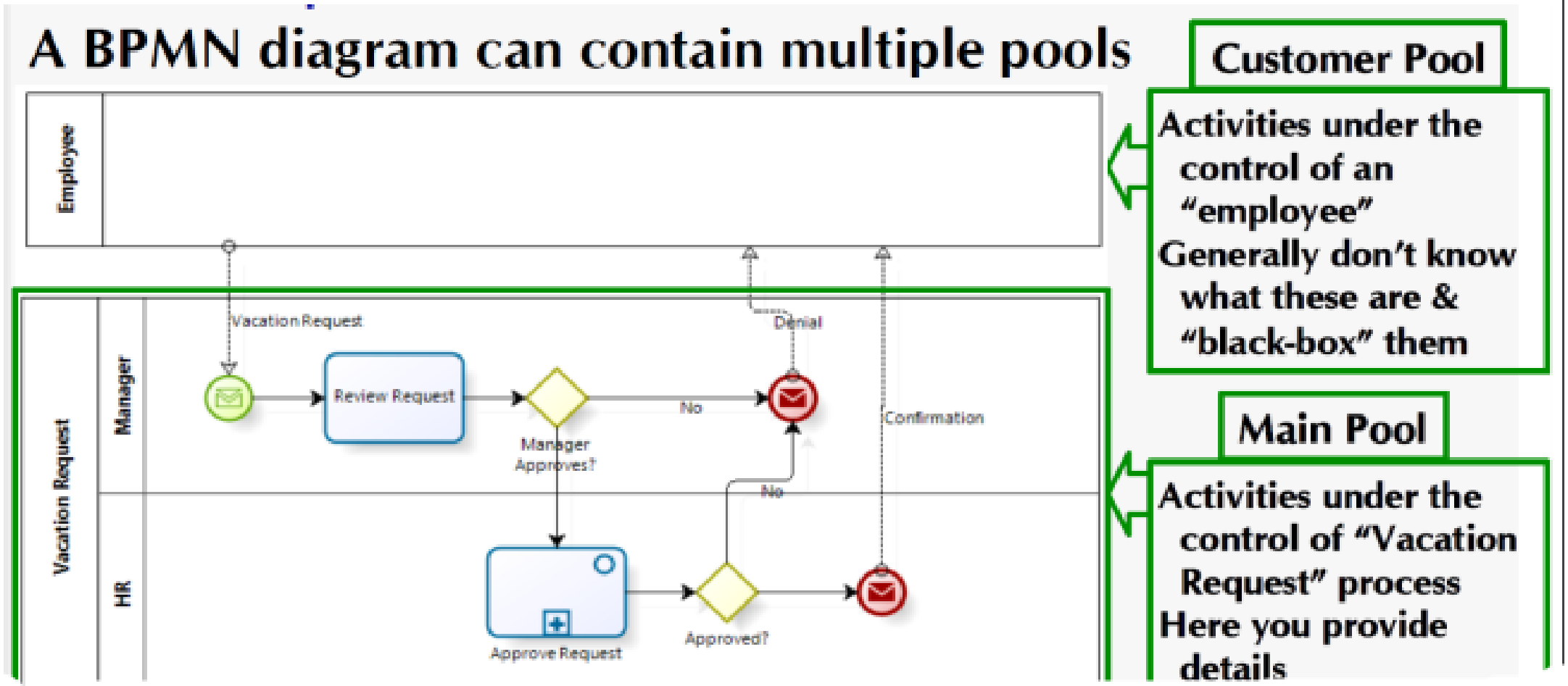
Pools and lanes

Each “pool” is associated with a **single process** that is:

! Separately owned, managed & **independently** running

The **main pool** is the process you wish to execute & manage

Example: Vacation request process

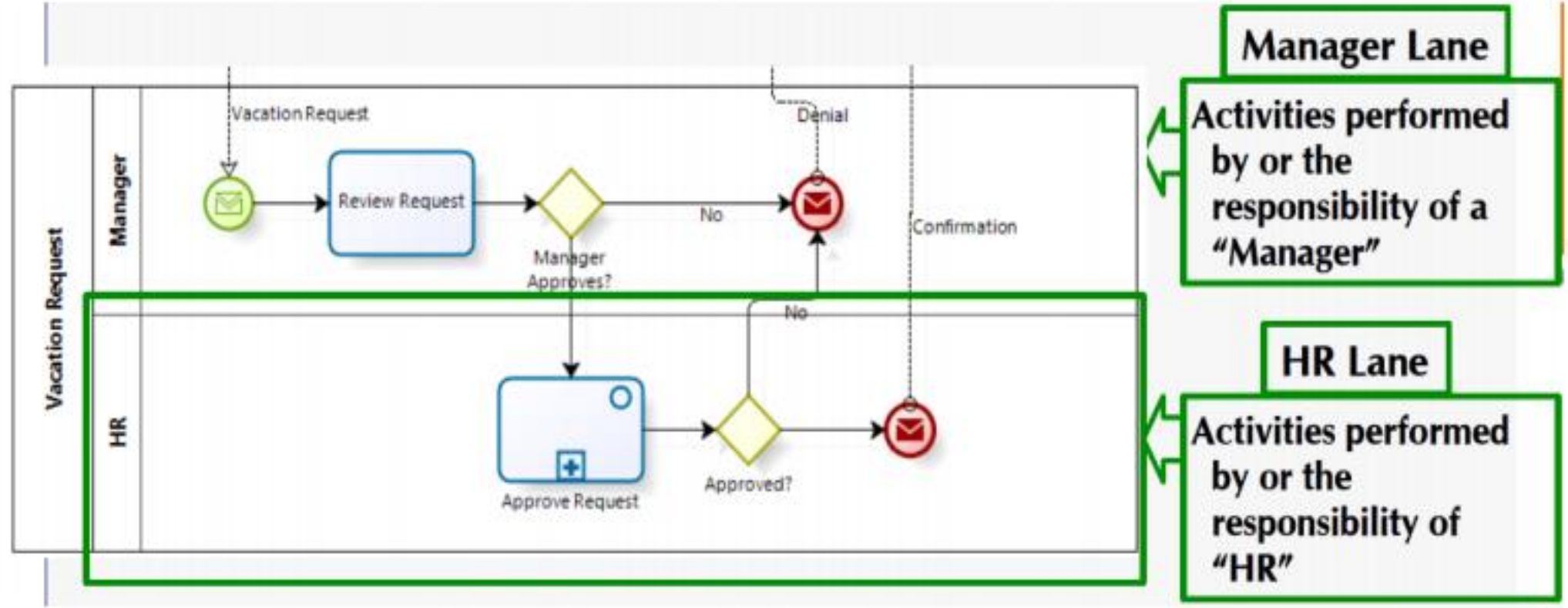


Swim-lane

A swimlane represents a **role** at some level of abstraction:

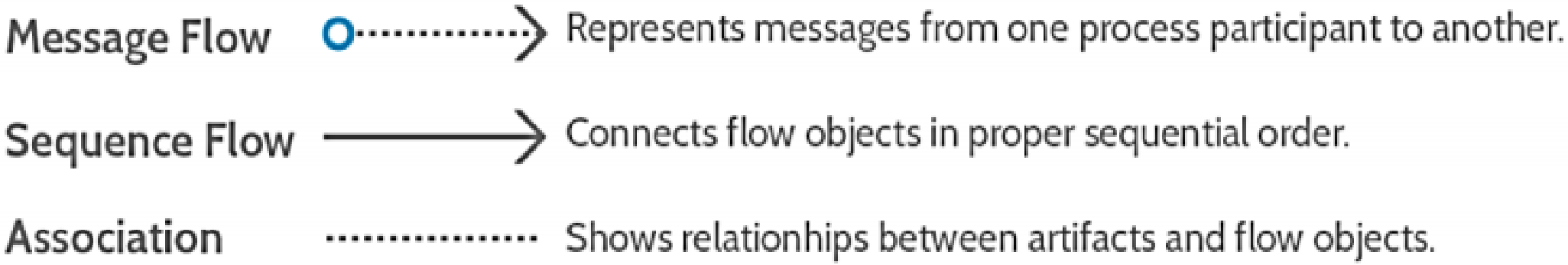
1. Organizational unit
2. Position or job title (role)
3. More generally, common grouping of functional capabilities

Swim-lane


Vacation Request
Manager
HR
Vacation Request
Review Request
Manager Approves?
No
Denial
Confirmation
No
Approve Request
Approved?
Manager Lane
Activities performed by or the responsibility of a "Manager"
HR Lane
Activities performed by or the responsibility of "HR"

Connecting objects

Connecting objects show associations and the flow of work/information



Connecting objects

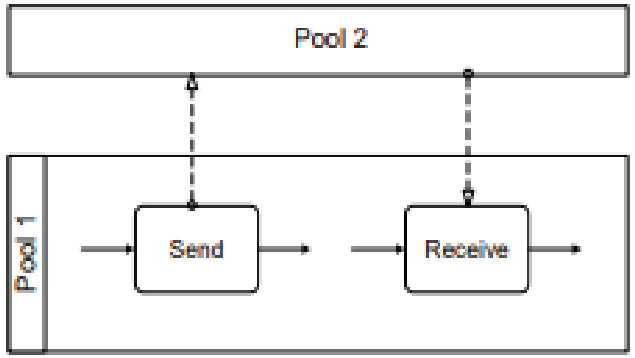
1. **Sequence flow** shows the flow of work, and is necessary for connecting together activities. Without a sequence flow, your map is invalid.
2. **Message flow** is used when different departments or organizations send information between each other. **It's the only kind of flow that can take place between pools or lanes (organizations or departments).**
3. **Association** is used to tie documents, databases, and other artifacts together with activities. For example, if you had an activity to sign off on a purchase order, you'd use a document symbol and an association line to link the two together.

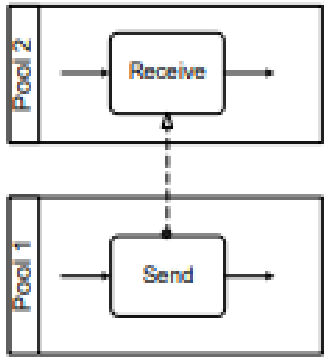
Message Flow

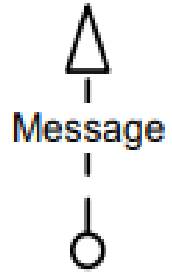
A *Message Flow* represents a flow of information between two process parties (**Pools**)

A Message Flow can connect:

- directly to the boundary of a Pool ➔ captures an *informative* message to/from that party
- to a specific activity or **event** within that Pool ➔ captures a message that triggers a specific activity/event within that party







Events

Definition:

an event is "**a noticed change in state**"

- It is an outcome or result
- **not** an activity or task

Two notions

- An occasion (something that happens)
- A result (important incident)

BPMN 2.0: more than 60 different types of events

	Start	Start: Event sub-pr. Inter.	Start: Event sub-pr. Non-inter.	Intermediate: Catching	Intermediate: Boundary Inter.	Intermediate: Boundary Non. Inter.	Intermediate: Throwing	End
None								
Message								
Timer								
Error								
Escalation								
Cancel								
Compen-sation								

	Start	Start: Event sub-pr. Inter.	Start: Event sub-pr. Non-inter.	Intermediate: Catching	Intermediate: Boundary Inter.	Intermediate: Boundary Non. Inter.	Intermediate: Throwing	End
Conditional								
Link								
Signal								
Terminate								
Multiple								
Multiple paralel								

Event Type	Start	Intermediate *catch*	Intermediate *throw*	End	Description
None					No specific trigger for the event (for example to start the process as a sub-process).
Message					Message to do one of the following: start the process, wait to resume the process, resume the process, signify the end of the process.
Timer					Event is triggered at a specific date/date or regular time interval (time cycle).
Error					Ends a sub-process with an error. On a task boundary, either catches the specified error or any error if no error code is specified.
Cancel					Ends a transactional sub-process. On a task boundary, catches the cancel event thrown from within the sub-process.
Compensation					Either throws or catches a call for compensation. Used to process compensating activities for previously executed tasks.
Conditional					Triggered based on the evaluation of conditions.
Link					Creates a "go to" or "off page connector" to break up a process for better legibility.
Signal					Broadcasts or catches signals.
Multiple					Indicates that one of several possible triggers are to be thrown or caught.
Terminate					Ends the process and all activities within without compensation or error handling.

Event categorizing

1. An event can appear at the **beginning** of a process, **within** a process (intermediate) or at the **end** of a process
2. An event can "**catch** a trigger", which means that it reacts to something or it can "**throw** a result"
3. An event can be generic or one of several predefined types: time-based, message-based, rule-based, signal-based, exception-based, etc.
4. An event can be positioned **within** sequence flow or attached at the **boundary** of an activity.
5. An event can **interrupt** the current process execution or not.
6. Some types of events can start a **parallel**, event-based sub-process.

BPMN Events

1. **Catching events (which react to a trigger)**
   1. All start events catch information (such as receiving an email)

2. **Throwing events (which the process triggers)**

BPMN Events

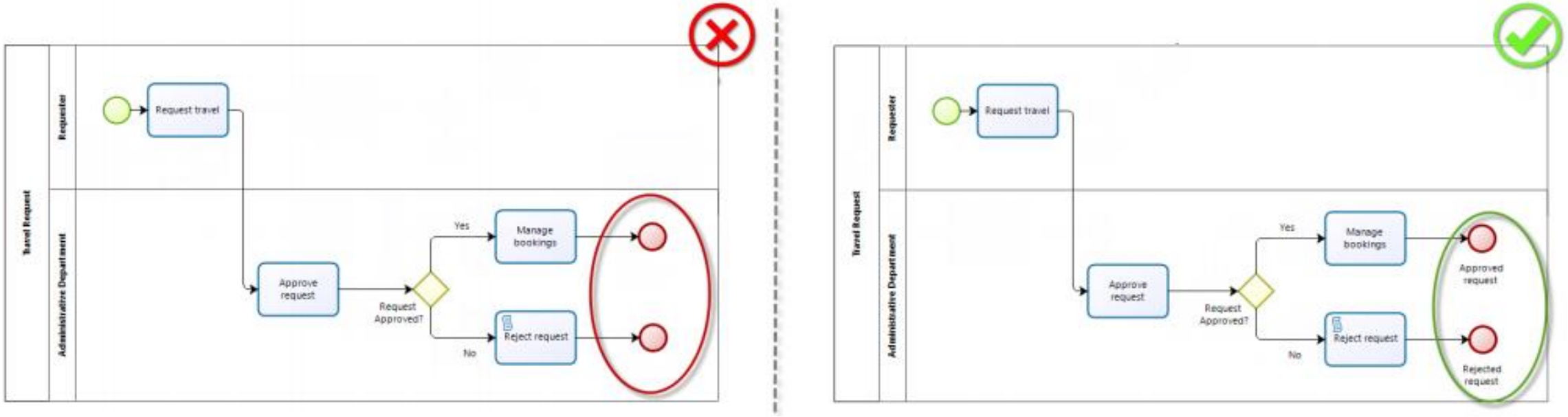
Events that occur in the **middle** of the process (between activities) are called ***Intermediate events***.

Events can also occur **during** an activity. This is known as a ***Border event***.

For example you may be assigned a task with a deadline. The **deadline** would be shown as a time event on a human task activity.

Labeling Events

Use labeling when **multiple** start and end events are used. Name them so the diagram can be self-explanatory and allow users to know how the process ends.



External event

Event originating from outside the process

- Customer pool, backend system, web service

Listened for (“caught) by process

Example message types include:

- SOAP (a messaging protocol for web services), Java Message Service (JMS), email, fax, phone, form, etc.

Internal (to process) events

Timer (countdown or calendar based)

Link (used to connect long sequence flow lines)

Conditional

Transaction failure (compensation)

Signal event (broadcast/listen; T/C)

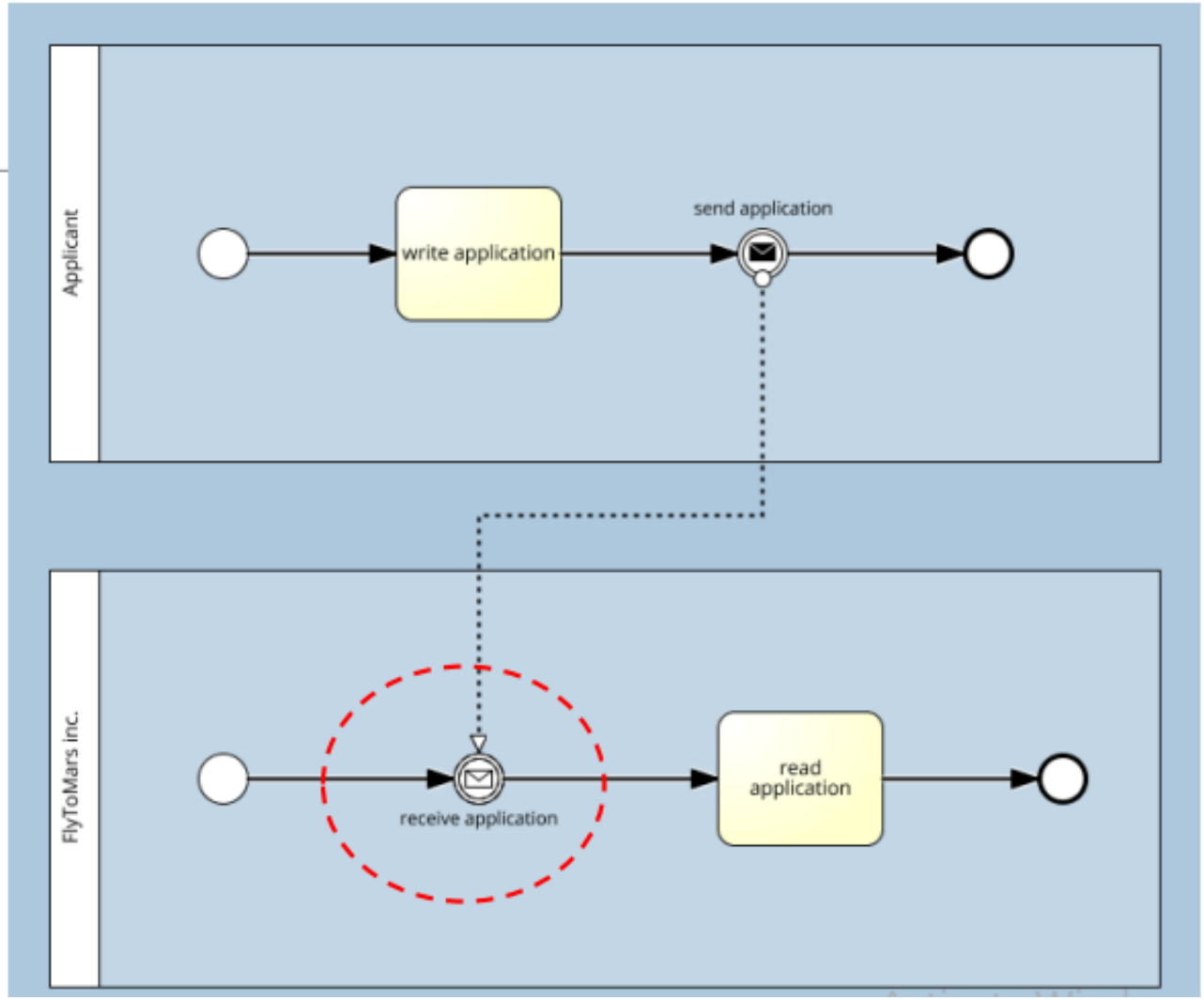
Multiple (fires off multiple event types; T/C)

Message (T/C)

Message event

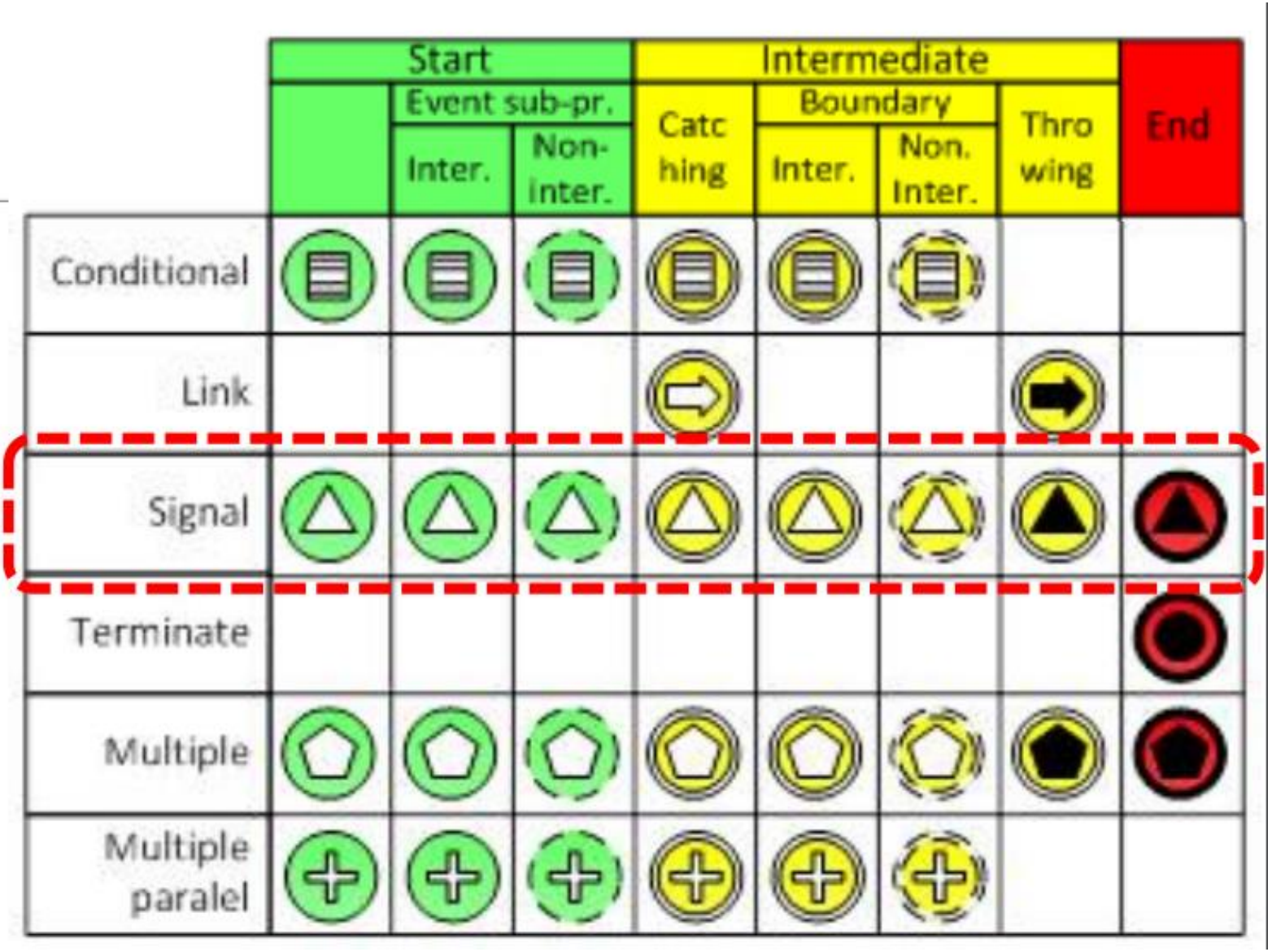
The company has to wait for the application:

The arrival is depicted by a **catching intermediate message event**



Signal event

Broadcasts or catches signals



Difference between Message and signal events

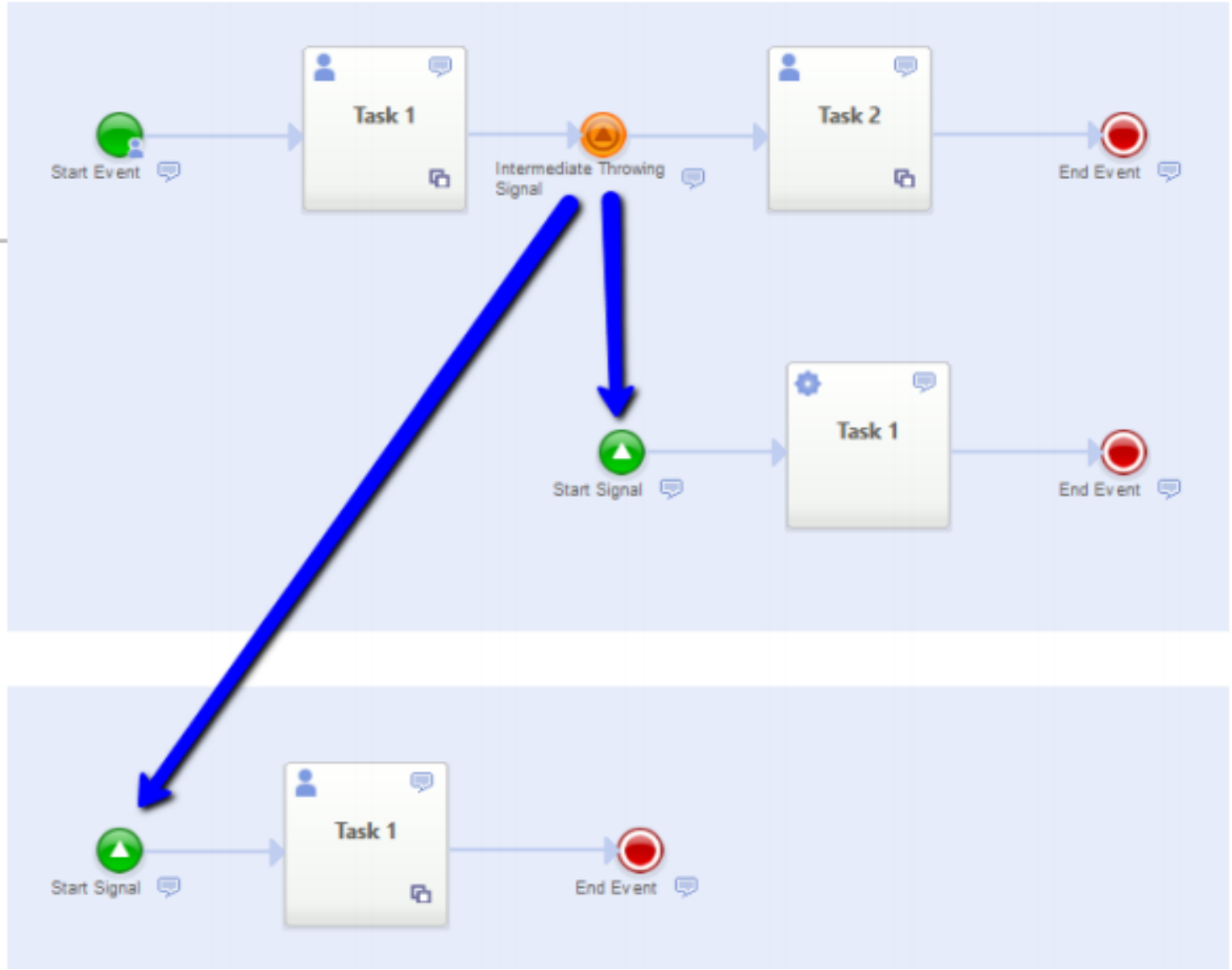
Message is triggered to a **single** target while a Signal is broadcast to **multiple targets**.

1. Message is a **point-to-point** communication
2. Signal is a **broadcast** communication (e.g. Twitter).

As the Signal is broadcast, different processes can have catching events listening for the same signal.

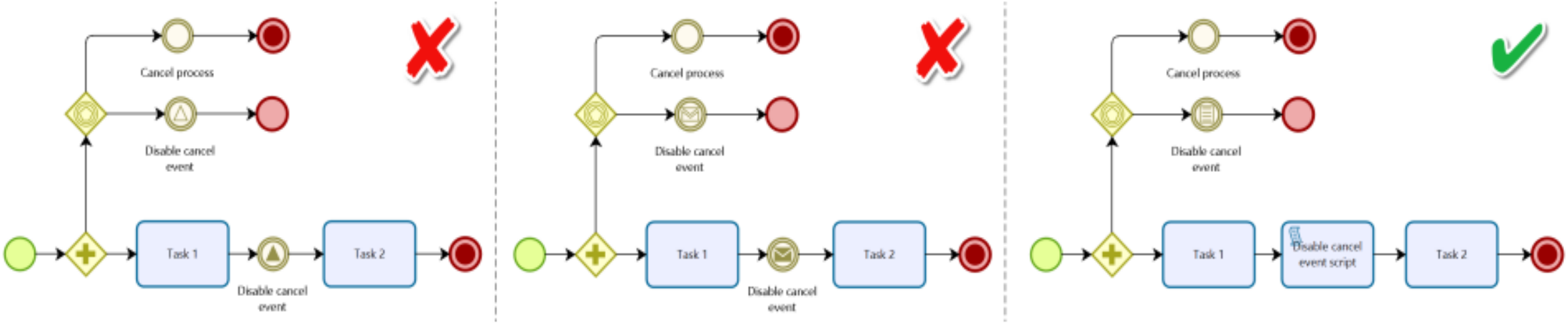
Signal event

broadcast communication



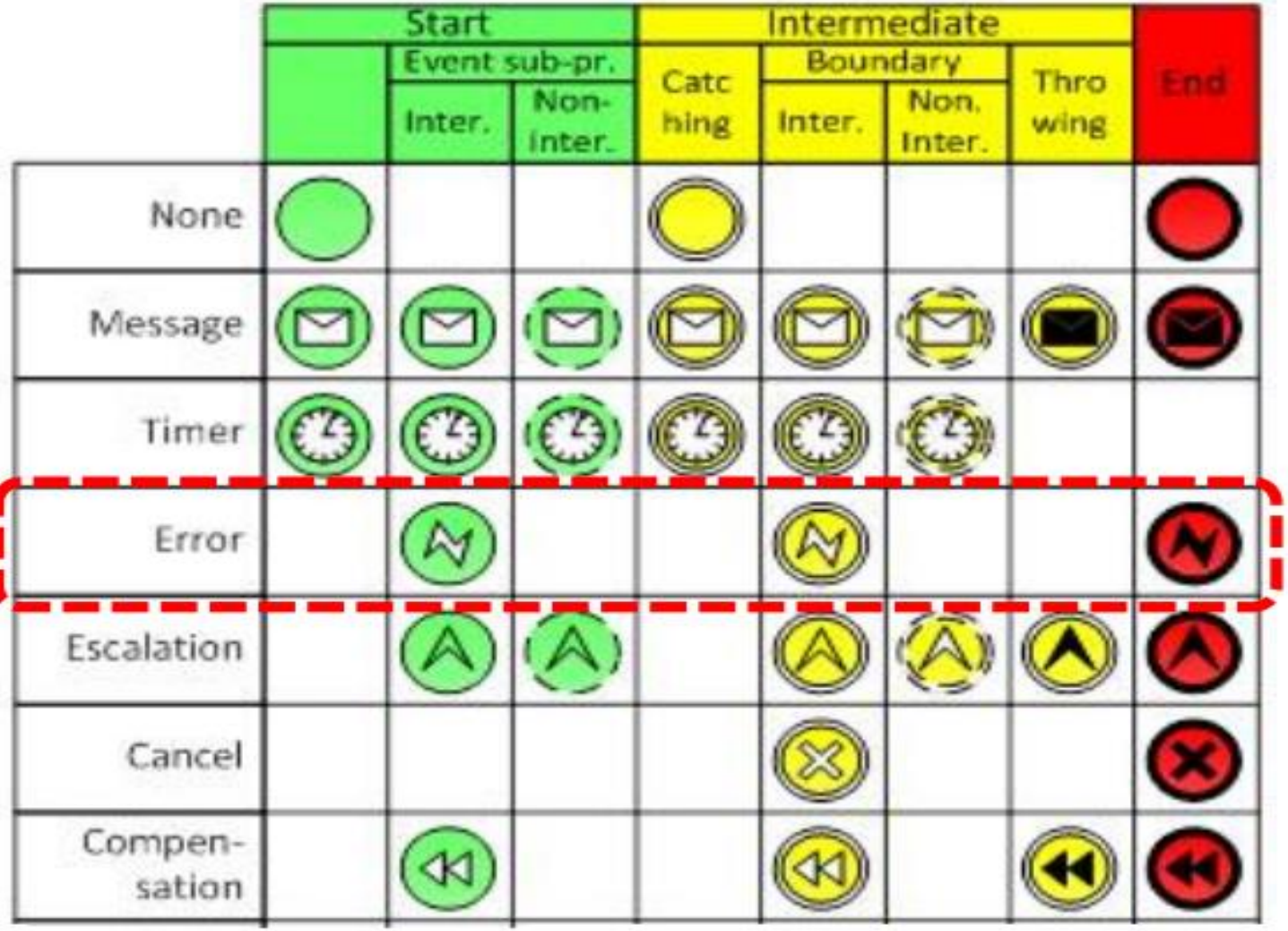
Anti patterns

Don't use signals nor messages to activate paths inside the process



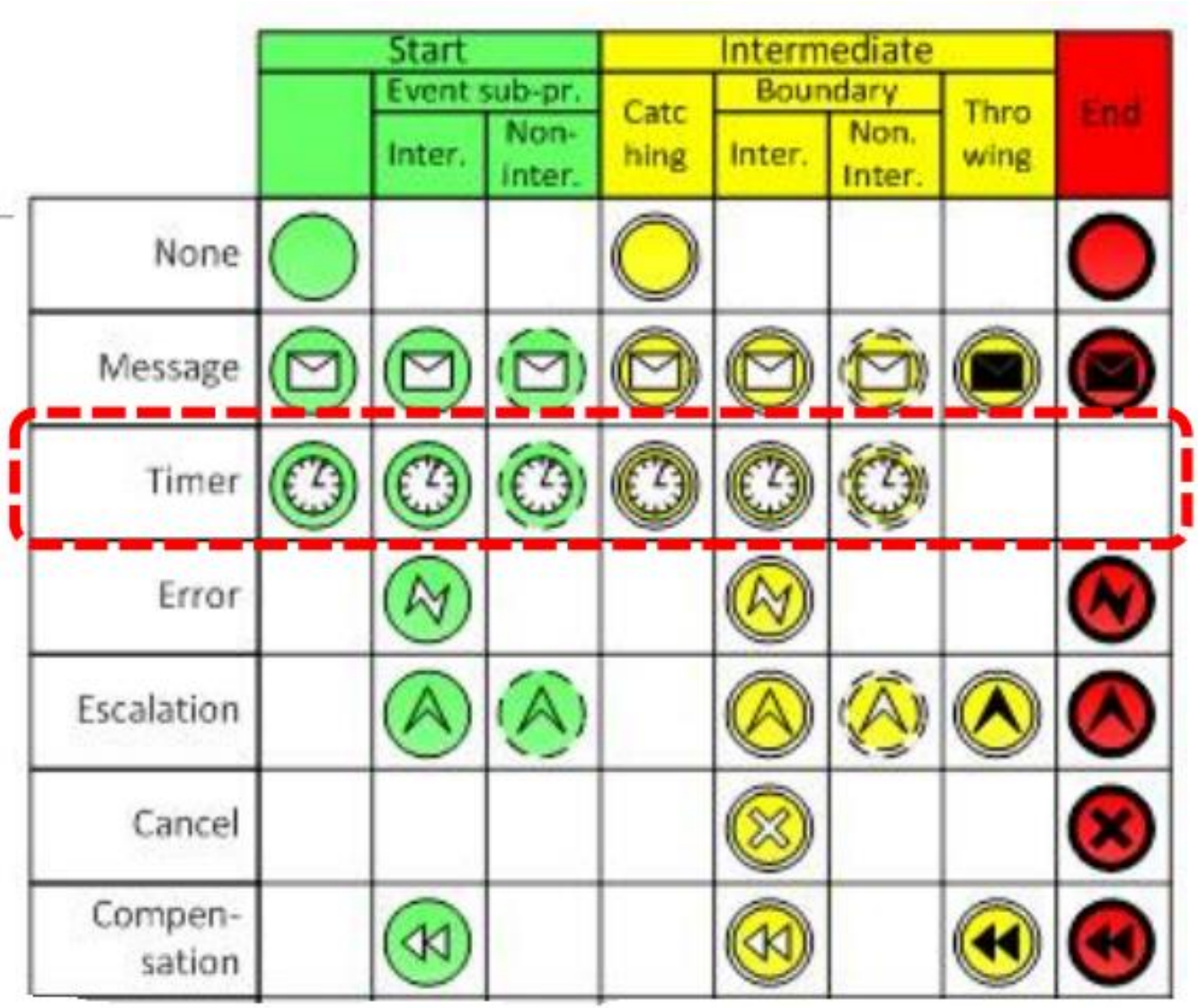
Error events

1. Error events are events which are triggered by a defined error.
2. **Interrupting**



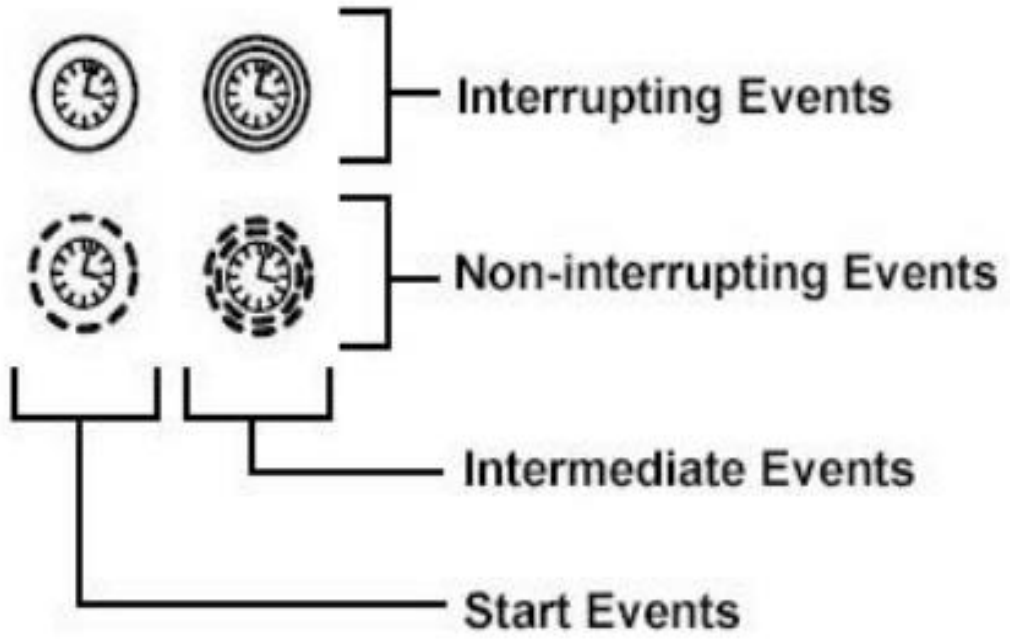
Timer events

Timer events



Timer events

All 4 of these Timer Events are "**Catching Events**" meaning they are **waiting** for a trigger before emitting a token and allowing flow to proceed down a particular path.



An interrupting event is more common that a non-interrupting event.

Temporal events

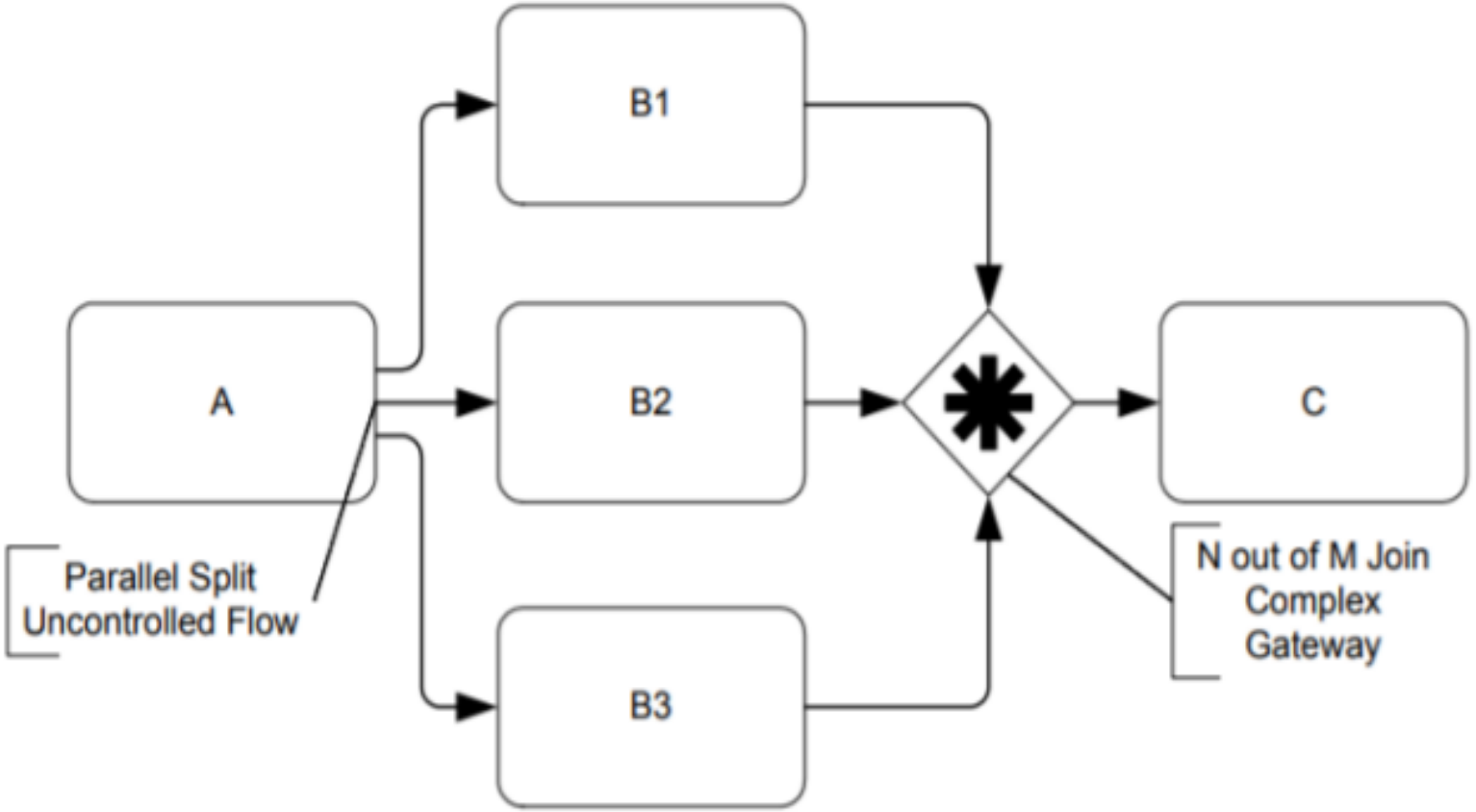
Start Intermediate End

Start Timer Event – Indicates that an instance of the process is created at certain date(s)/time(s), e.g. start process at 6pm every Friday

Intermediate Timer Event – Triggered at certain date(s)/ time(s), or after a time interval has elapsed since the moment the event is "enabled" (delay)

Complex Decision Gateway

N out of M Join



Thank you for your attention