BPMN
La mappatura dei processi di business

Ing. Francesco Zammori

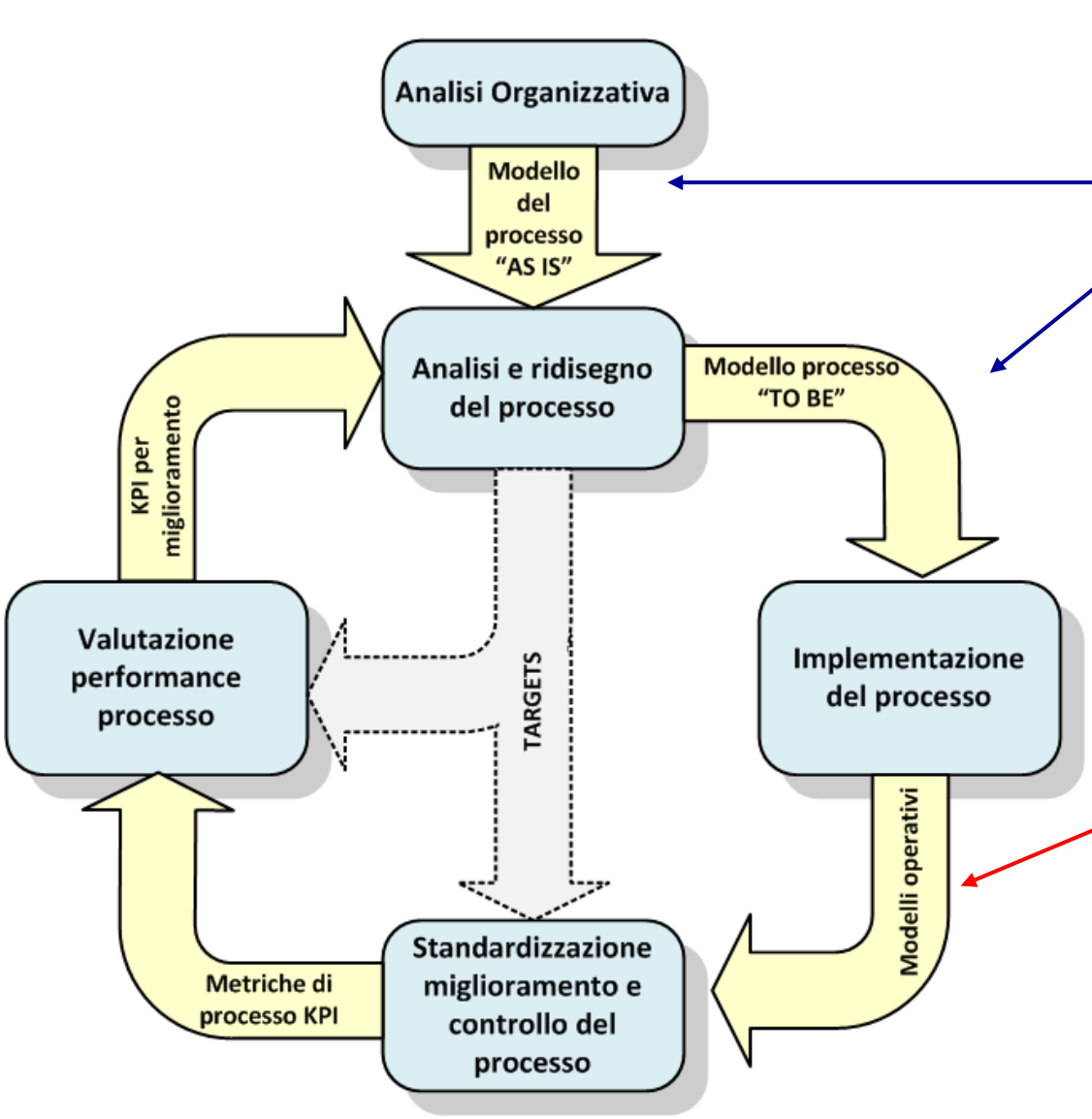
Dipartimento di Ingegneria Industriale

Università di Parma

Siti d'interesse

- Il contenuto di queste slide è, in parte, tratto e rielaborato da materiale a cura di:
  - *Prof. Cimino - Università di Pisa – Modeling and Simulation of Business Processes using BPMN 2.0*
  - *Prof. Marrella - Università di Roma La Sapienza – Modelling Business Processes with BPMN*
- Altro materiale è accessibile dai seguenti siti pubblici:
  - *http://bpmb.de*
  - *https://camunda.org/bpmn/examples*
  - *www.Bizagi .com/bpkn/Work flow Patterns*

Perché modellizzare/mappare i processi?



Comunicazione

Simulazione

Activity Based Costing

…

Modello di dettaglio

Tipo dati, condizioni, mappatura, mancanza dati, …

Integrazione, testing, sviluppo…

Obiettivi della mappatura

- Fornire una rappresentazione sintetica dello stato As-Is
- Individuare le diverse attività svolte e le modalità di esecuzione Rendere esplicite le differenze tra attività
- Spiegare i flussi
- Definire univocamente le responsabilità
- Far comprendere l'impego delle risorse
- Identificare i controlli «chiave»
- Valutare l'efficienza e l'efficacia
- Fornire ausilio per la semplificazione/eliminazione attività a non valore
- Fornire ausilio per l'aggregazione delle attività

The main process mapping techniques

- **Business Processes**
  - **IDEFØ (Icam DEFinition for Function Modeling)**
  - **Cross Functional Flow Diagrams**
  - **BPMN (Business Modelling Notation)**

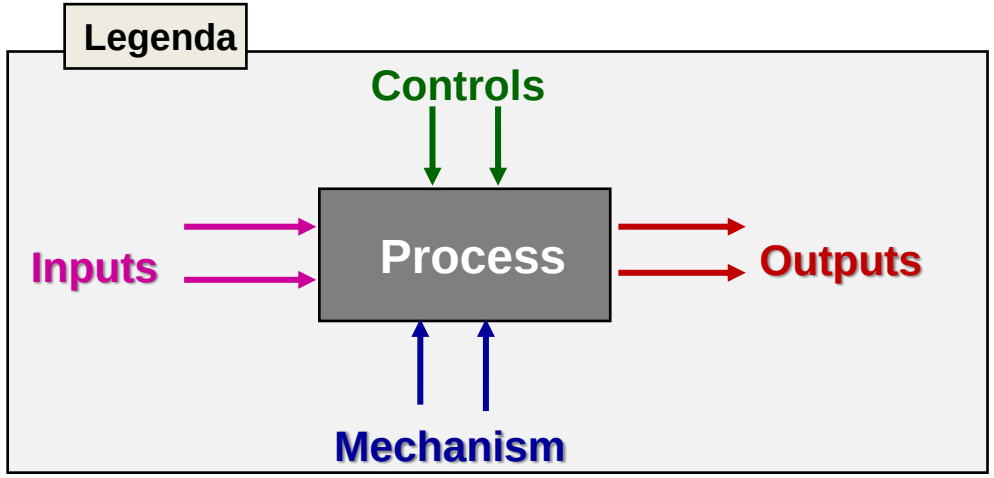
Software & IT Applications Processes

- **DFD (Data Flow Diagram)**
- **UML (Unified Modelling Language)**
- **Flow Diagrams**

Note – We will focus only on BPMN, however, the other techniques will be briefly pinpointed

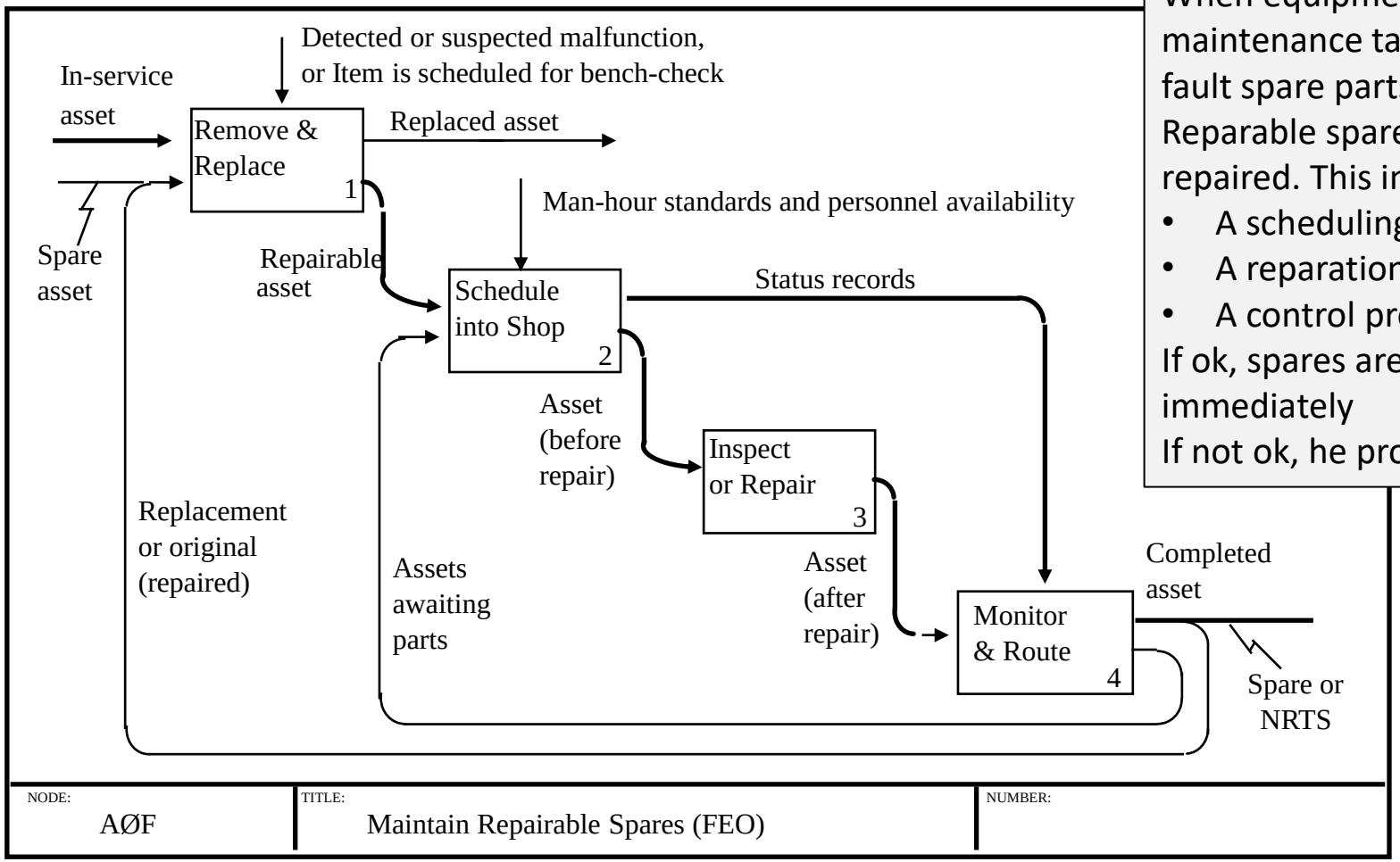
IDEFØ

- It is a **high level process mapping technique** that shows, for each process its inputs, outputs, controls and mechanism
  - *High level means that it gives a generic description, neither single activities nor procedures are described. Also, the logical conditions that regulate a process are not considered.*
- Processes are represented as «Black Boxes»
- Process can be linked, to form a network. Specifically, outputs of a process may become inputs or constraints of another process. This creates a dependency among processes.



- **Inputs** - Things (such as raw materials) that are needed and that will be modified during the process
- **Controls/Constraints** - Elements (such as procedures) needed, but that are not transformed by the process
- **Outputs -** The outcomes of the process
- **Mechanisms** - People, machines and software needed to operate the process

IDEFØ – Management of Spare components



When equipment fails, a maintenance task is opened and fault spare parts are substituted
Reparable spares will be repaired. This implies:

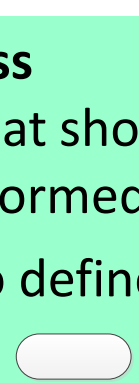
- A scheduling process
- A reparation process
- A control process

If ok, spares are stored or used immediately
If not ok, he process cycles

Source: Integrated Computer-Aided Manufacturing (ICAM): Functional Modeling Manual (IDEF0), June 1981, p. 4-5.

Flow Diagrams

- It is a **low level process mapping technique** that shows how activities are performed.
- Basic icons are used to define
  - *Start/Finish*
  - *Action*
  - *Decision making*
  - *Result*



- We have a bunch of keys, but we do not know which one opens the door.
- What should we do to find the right one?
- To avoid trying the same key more than once a possible solution is shown as a flow diagram

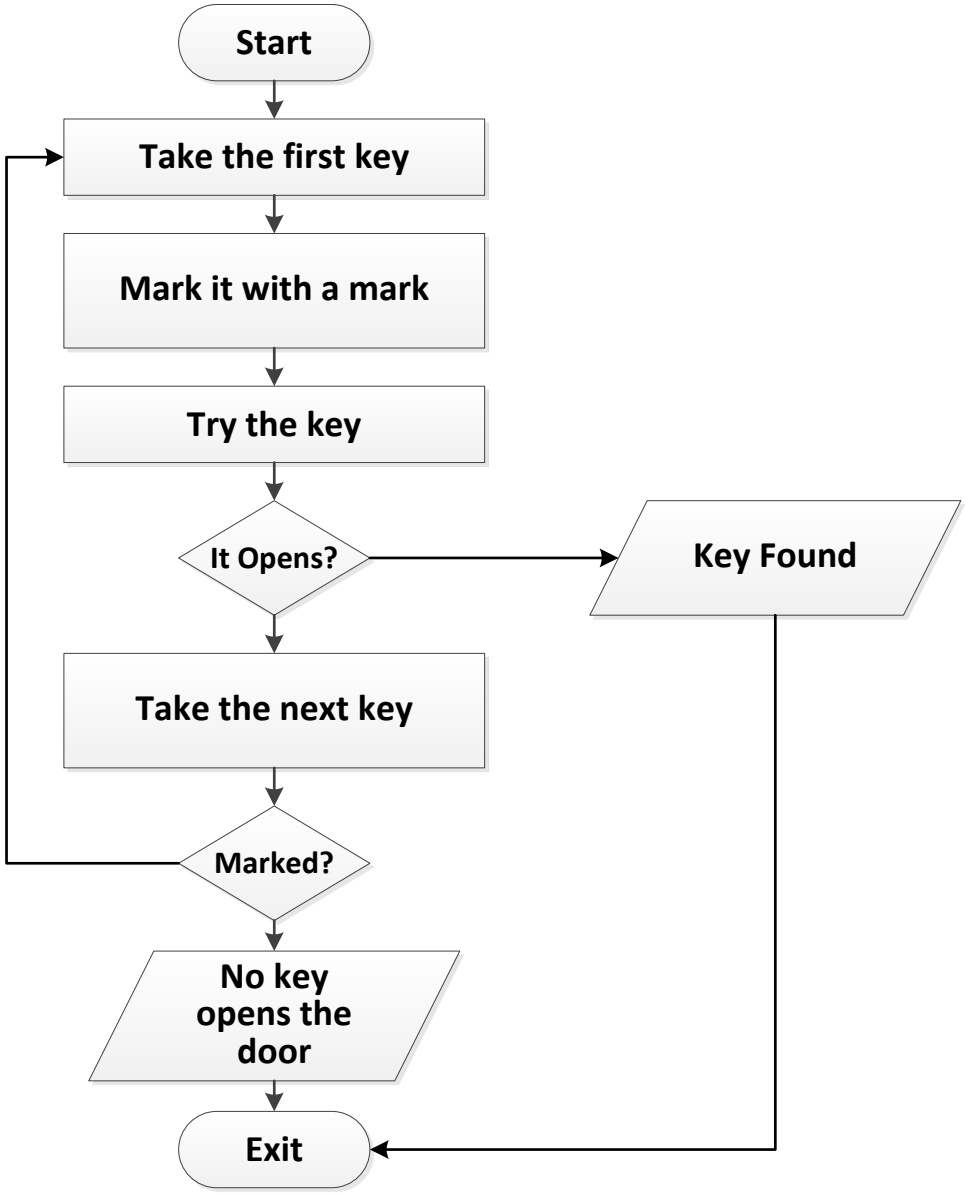
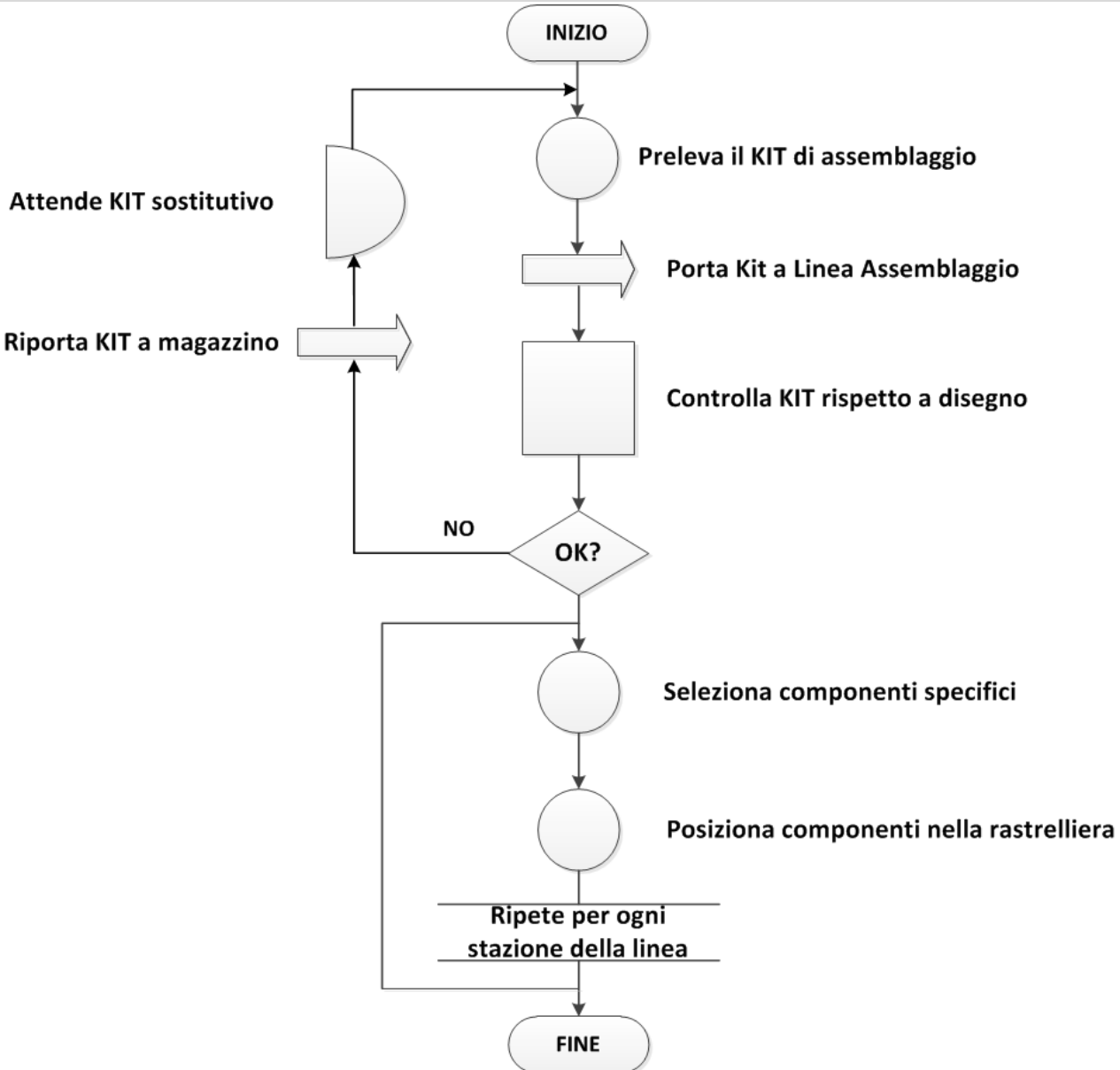


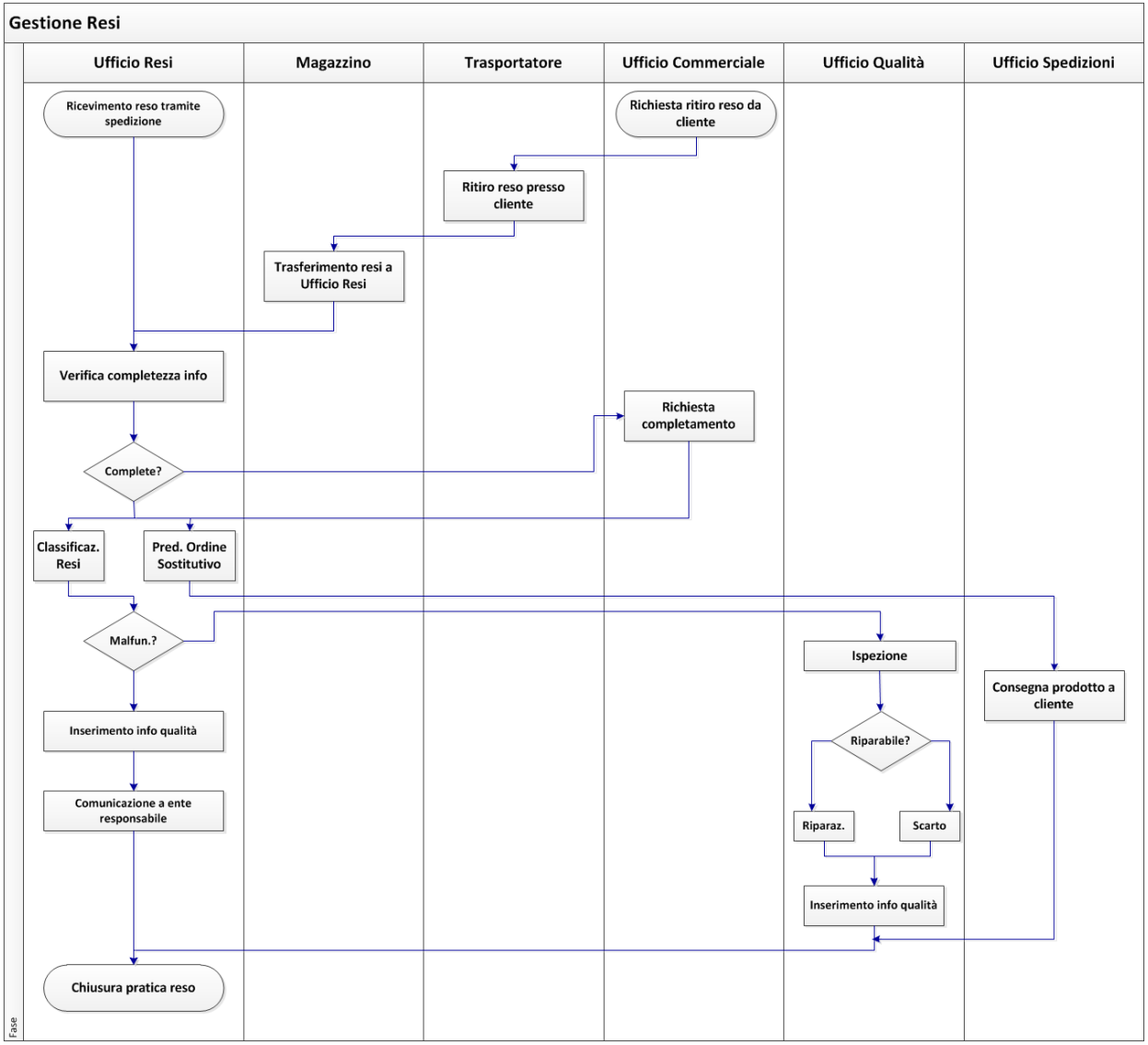
Diagramma di flusso di una procedura «logica»



Alimentazione di una linea di assemblaggio

Diagramma Cross-Funzionale

- Quando ad uno stesso processo lavorano **persone appartenenti a funzioni** (uffici) **diverse** è possibile introdurre delle **swimlane** (corsie) per evidenziare chi fa cosa
- Si parla di **Cross Functional Diagrams**
- Nell'esempio è mostrato un processo di gestione dei resi che coinvolge:
  - *Ufficio Resi*
  - *Magazziniere,*
  - *Trasportatore*
  - *Ufficio Commerciale*
  - *Ufficio Qualità*
  - *Spedizioni*



Data Flow Diagram (DFD)

Il DFD è uno strumento di mappatura particolarmente orientato alla visualizzazione del flusso/elaborazione di dati di un sistema

Particolarmente indicato per processi informativi e/o supportati da SI, il DFD è diventato molto popolare come supporto al BPR

Mostra il flusso di dati in

- *Organizzazioni*
- *Imprese*
- *Sistema Hardware*
- *Sistema Software*

Usa 4 icone di base

- *Dati in movimento – frecce*
- *Trasformazione di dati – circoletti*
- *Origine e Destinazione dati– rettangoli (o terminatori)*
- *Dati contenuti in archivi statici – linee parallele*

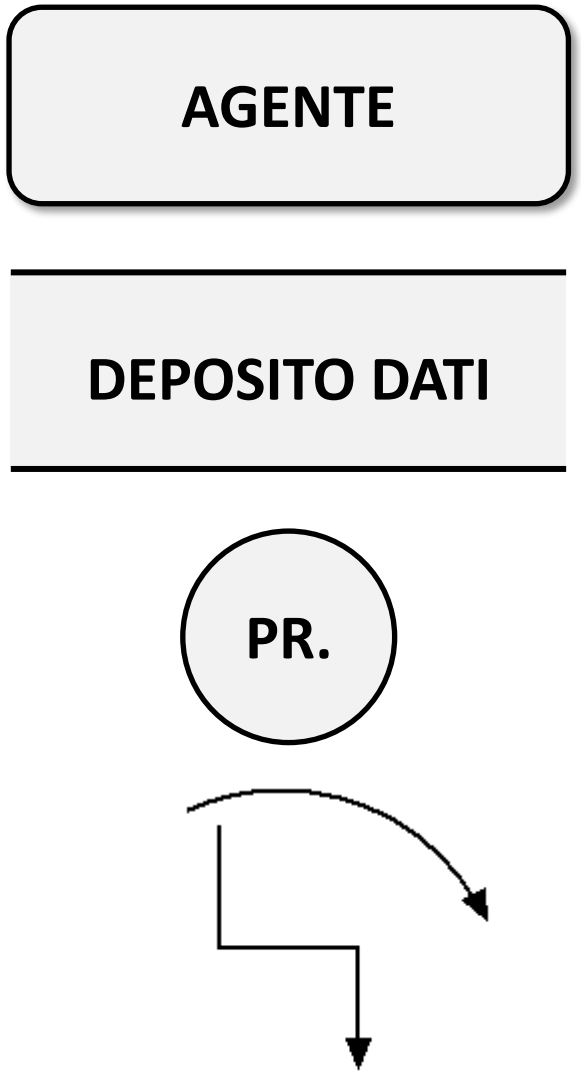
Data Flow Diagram (DFD)

Agente (elemento esterno)
elemento che produce o consuma dati nel sistema

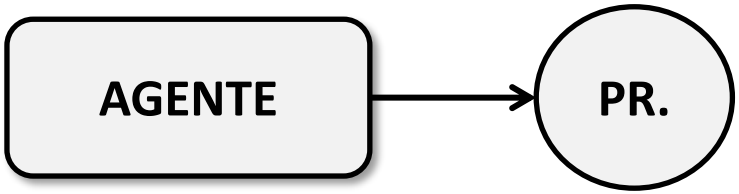
Deposito dati
Informazioni archiviate con accesso in lettura e scrittura

Processo
Porzione del sistema che trasforma i dati

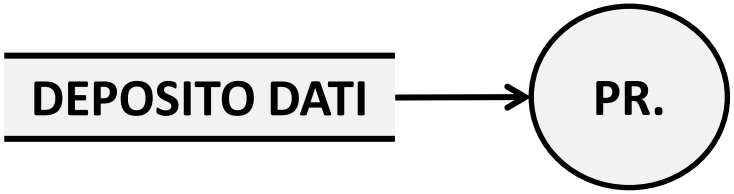
Flusso
Linea attraverso cui si propaga l'informazione



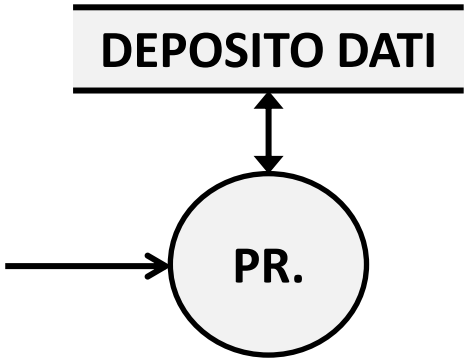
Data Flow Diagram (DFD)



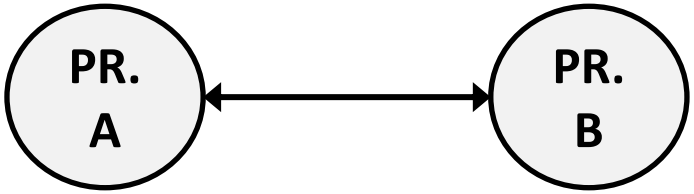
Agente che funge da sorgente



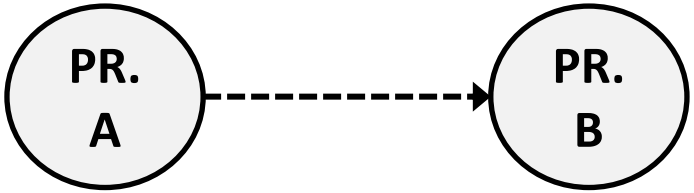
Accesso a dati in lettura



Accesso a dati in lettura/scrittura



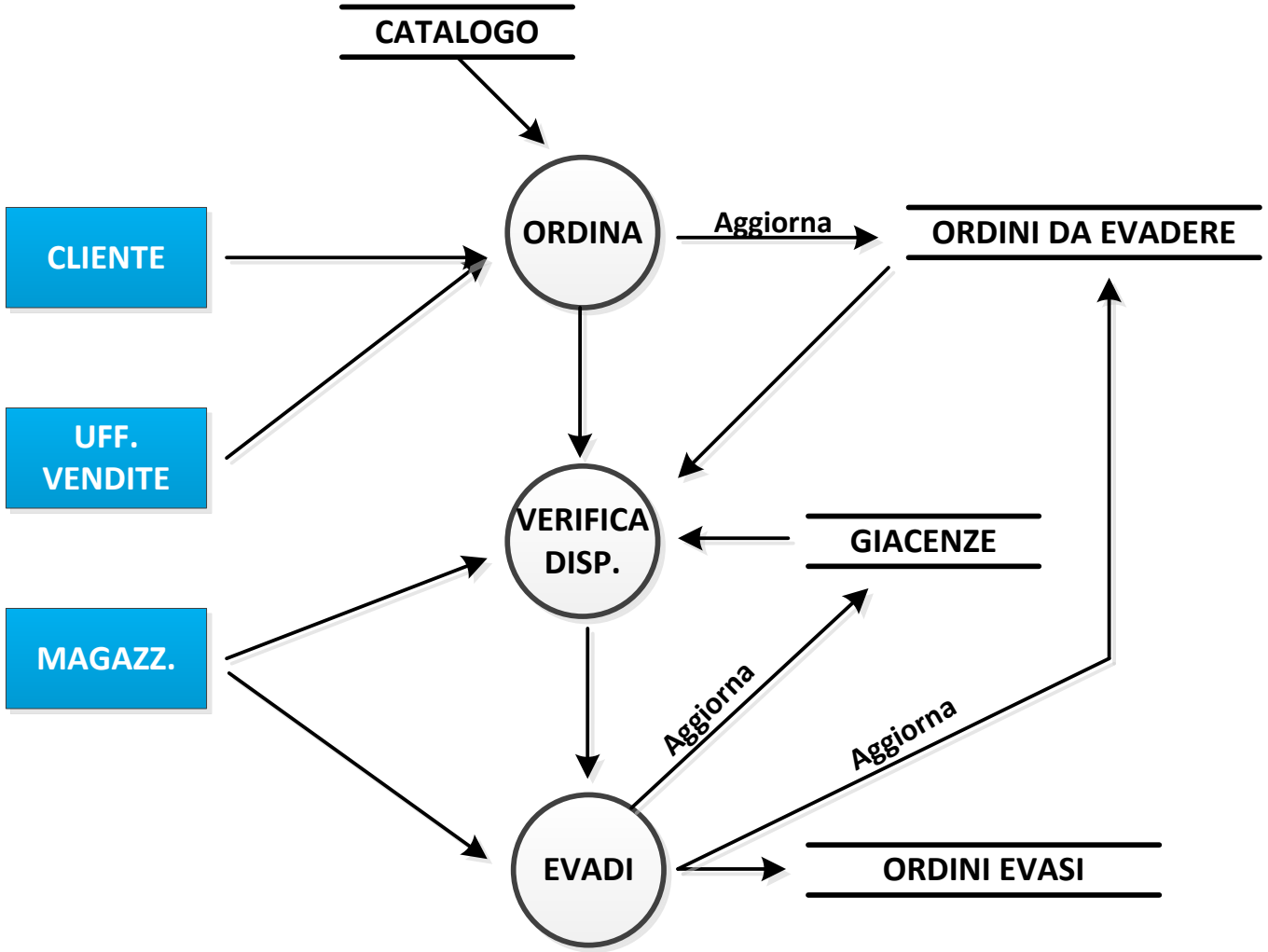
Flusso dati tra processi



Operazione Booleana

Data Flow Diagrams

Esempio DFD - Ordini



Unified Modelling Language

- Il cuore della modellizzazione ad oggetti sta nella corretta costruzione del **modello concettuale**.
- Il **modello deve astrarre**, a partire da un caso pratico generalmente complesso, **gli elementi essenziali e ricorrenti** che lo contraddistinguono e lo generalizzano.

- ✓ UML è un **linguaggio grafico** che permette di visualizzare, specificare, costruire e documentare tutti i componenti di un sistema SW complesso
- ✓ UML offre uno standard per mappare i progetti di sistemi software includendo nell'analisi

Aspetti concettuali

- *processi coinvolti*
- *funzionalità del sistema*

Elementi concreti

- *dichiarazioni di linguaggio*
- *schemi di database,*
- *classi*
- *alte parti riutilizzabili*

Unified Modeling Language

UML è importante perché:

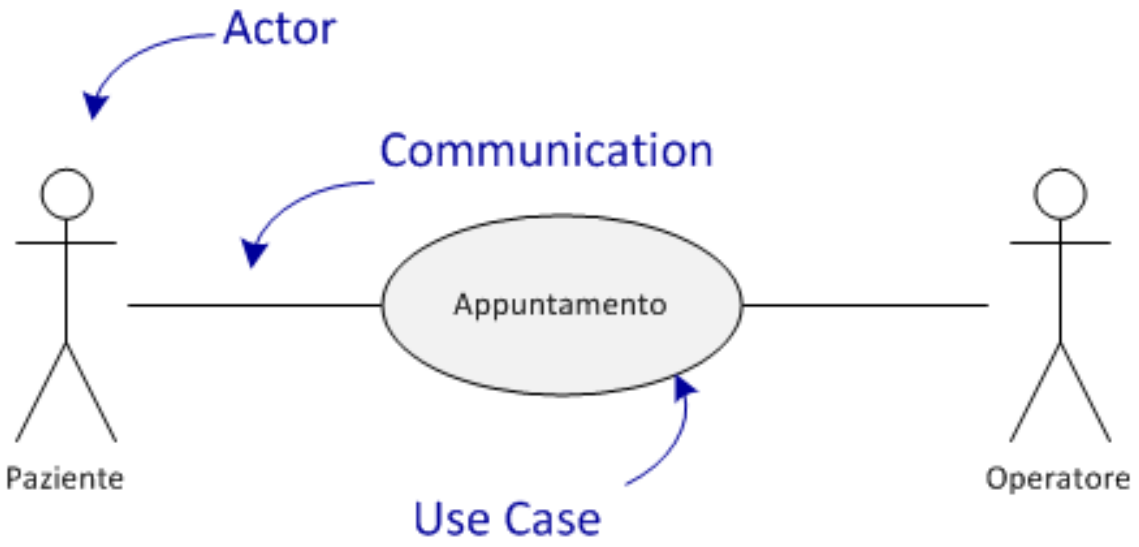
- Descrive l'architettura «Object Oriented» del sistema
- Descrive il comportamento del sistema
- Descrive l'interazione fra l'utente ed il sistema
- Descrive gli stati del sistema
- Descrive le attività svolte dal sistema
- Descrive i componenti dei sistema
- Descrive il modo in cui opera il sistema

Diagrammi usati

- Use case diagrams
- Class diagrams
- Object diagrams
- Sequence diagrams
- Collaboration diagrams
- State chart diagrams
- Activity diagrams
- Component diagrams

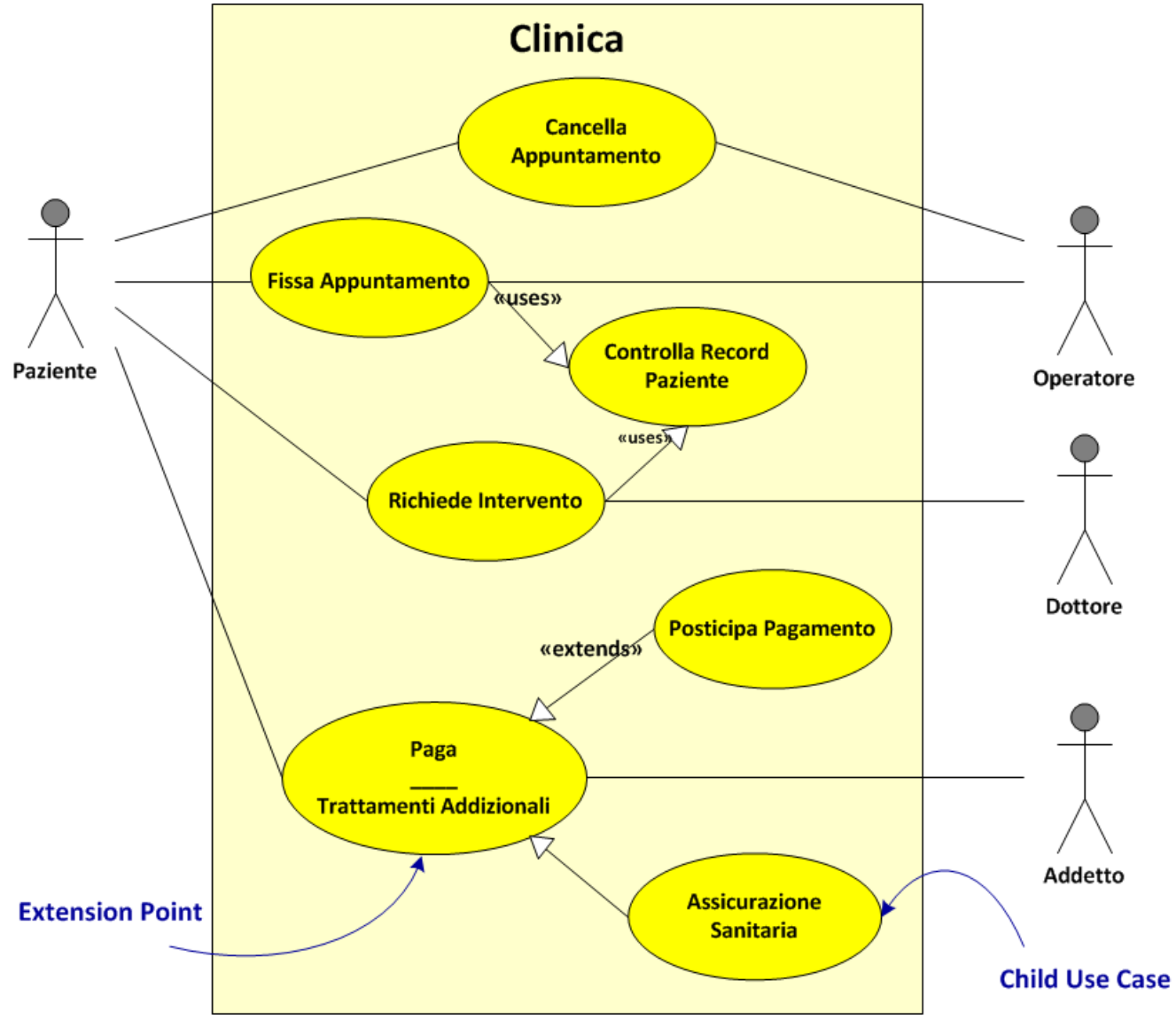
UML: Use Case Diagrams

- Use Case Diagram descrivono ciò che fa il sistema, secondo il punto di vista di un osservatore esterno
- L'enfasi è sul cosa, piuttosto che sul come
- Sono fortemente connessi al concetto di scenario
- Uno scenario è un esempio di ciò che può accadere quando un utente interagisce con il sistema



Un paziente chiama una clinica per fissare un appuntamento. La receptionist trova il primo slot libero e fissa l'appuntamento.

UML: Use Case Diagrams

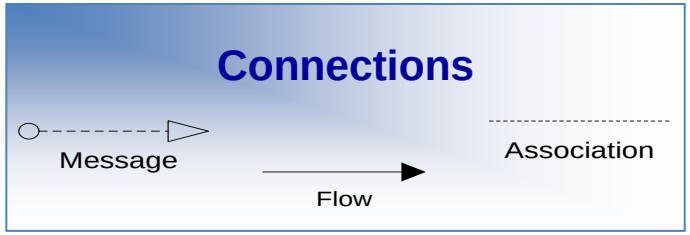


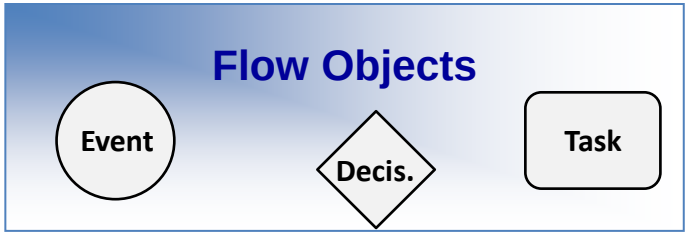
- Confini Sistema
- “Uses”
- “Extend”
- “Generalize”
- “Ext. Point”

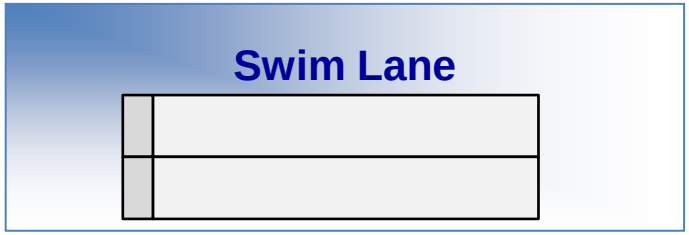
BPMN

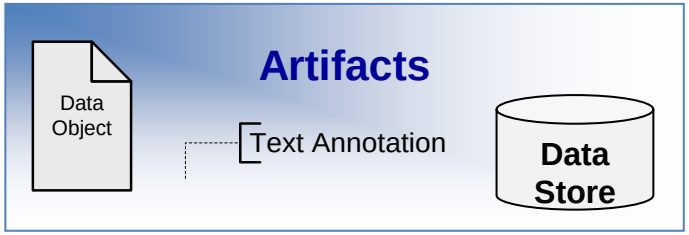
- La tecnica BPMN è un'evoluzione dei diagrammi di flusso Cross-Funzionali
- Rispetto a questi BPMN introduce:
  - ***Gateway Logici complessi***
  - ***Gli Eventi***

- Tale metodo cerca di rappresentare in maniera formale, ma intuitiva, tutti i processi aziendali anche se complessi
- Usa 4 categorie fondamentali di oggetti:



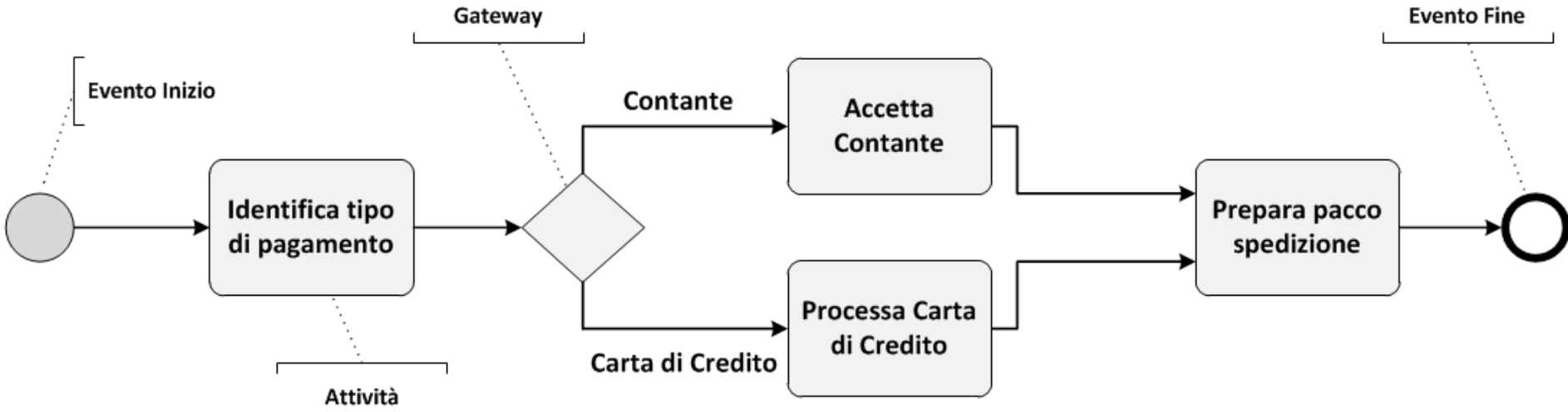






BPMN – Esempio introduttivo

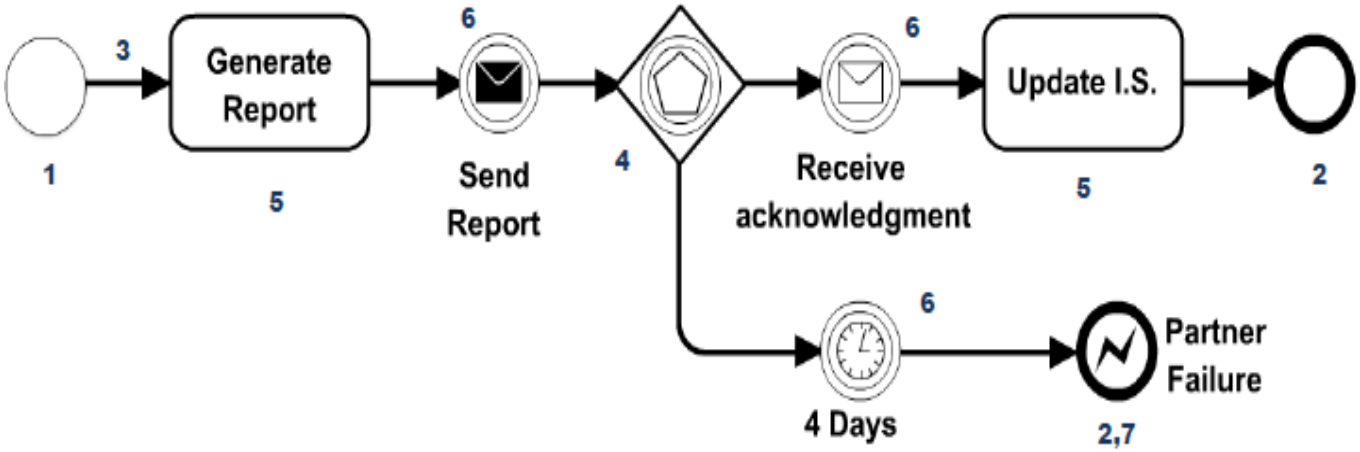
- Pagamento e preparazione di un collo di spedizione
  - *S'indentifica il tipo di pagamento*
  - *A questo punto avviene una decisione logica di tipo esclusiva (il pagamento avviene in contanti o con carta di credito).*
  - *Effettuato il pagamento si prepara il collo di spedizione e il processo termina*



Pagamento e preparazione collo di spedizione

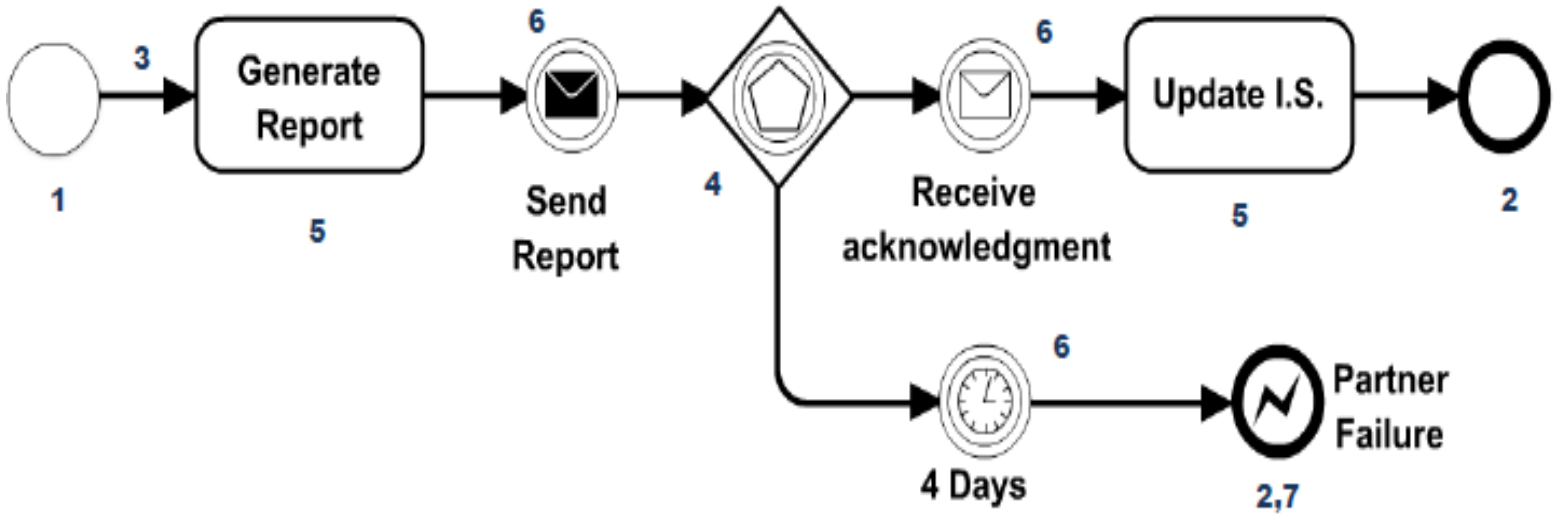
BPMN – Un secondo esempio più complesso

- Consideriamo il seguente processo:
  - Viene generato un report
  - Alla sua conclusione lo si invia ad un supervisore per approvazione
  - Si aspetta il segnale di ricezione/lettura/approvazione
    - Se arriva si aggiorna il SI e il processo termina
    - Altrimenti si genera un evento di errore
- Il processo è mappato di seguito. Pur essendo semplice la mappatura si complica. Perché?
- Perché il flusso logico di funzionamento è regolato da eventi



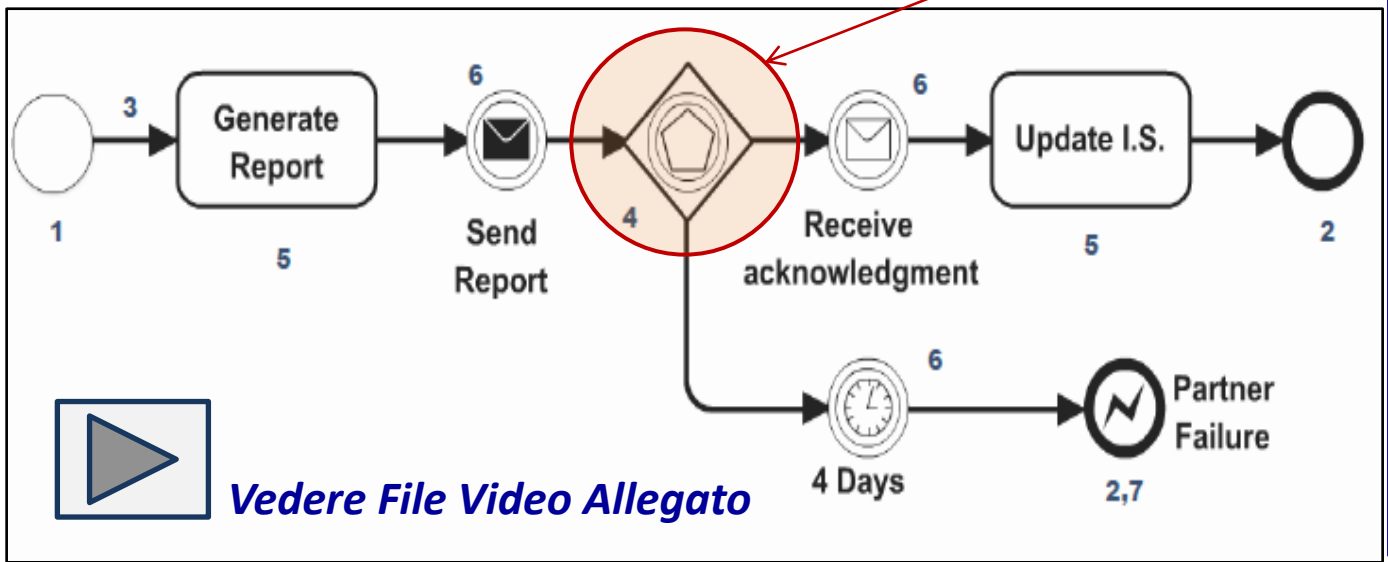
BPMN – Un secondo esempio più complesso

- Bisogna immaginare che l'evento iniziale generi un gettone che fluisce lungo le frecce a tratto continuo
- Quando il gettone incontra un evento «throwing» (6) si genera un evento che verrà intercettato da un altro processo
- Generato l'evento il gettone procede e raggiunge un gateway (4). Si tratta di un gateway basato su eventi. Il gettone aspetta sino a che non si verifica una dei due eventi indicati dagli eventi di tipo «catching» posti subito dopo il gateway basato su evento.
- Il primo evento che si verifica apre il gateway basato su eventi e permette al gettone di proseguire lungo le linee di flusso logico corrispondenti all'evento che si è verificato



BPMN

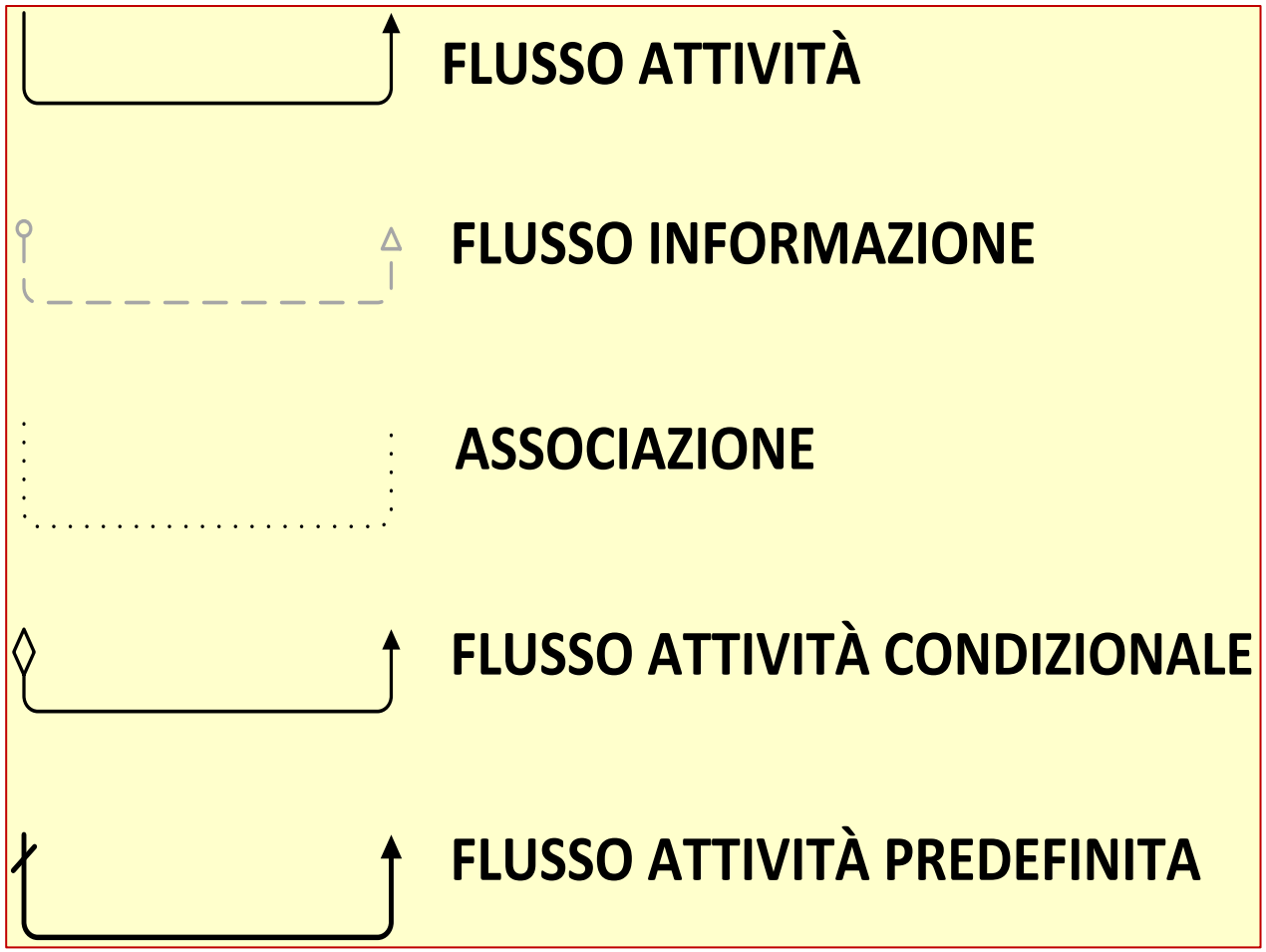
- BPMN si focalizza sul flusso di **controllo** del processo visualizzato mediante il concetto di **«Token»** che fluisce attraverso i vari step del processo
- Si parte da uno *Start Event (1)* che genera il token che dà avvio al processo e che, alla fine, raggiungerà un *End Event (2).*
- Il percorso seguito dal token è definito:
  - dalle frecce (Sequence Flow (3))
  - dai punti decisionali (Gateway (4))
  - attività (5)
  - eventi intermedi (6)



Importante il concetto di **«Race Pattern»** al punto decisionale (4). Solo uno dei due eventi intermedi (6) potrà aver luogo: se si riceve per tempo il report si aggiorna il Sistema Informativo, viceversa si genera un evento errore (con semantica throw (7))

BPMN: Flow Objects

1. Connessioni: rappresentano flussi fisici/informativi e associazioni logiche

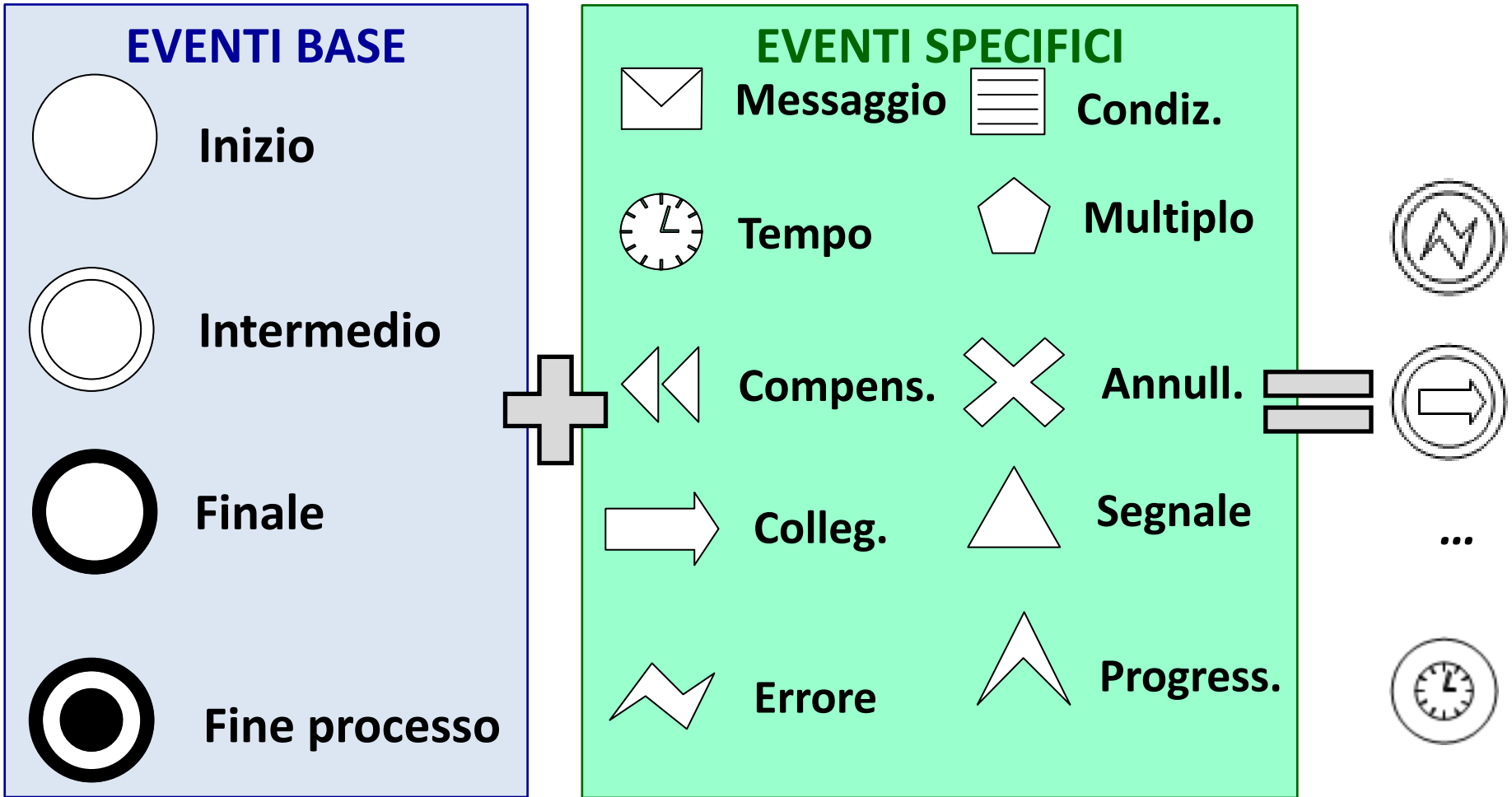


Gli artefatti non possono essere collegati né a flussi di attività né a flussi di informazione

L'associazione serve per legare le attività agli artefatti

BPMN: Flow Objects

2.1. Eventi: rappresentano qualcosa che accade durante il processo e scatena un'azione e/o influenza i risultati



BPMN: Flow Objects

2.1. Eventi: rappresentano qualcosa che accade durante il processo e scatena un'azione e/o influenza i risultati

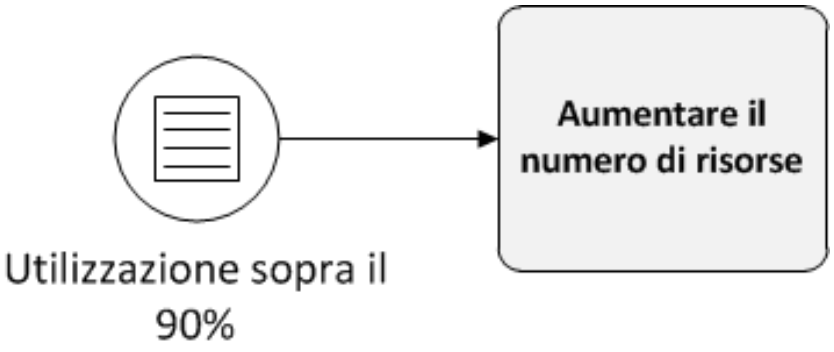
- **Gli eventi specifici indicano direttamente che evento si sta verificando**
- **Possono essere iniziali, finali e intermedi**
- **I principali sono:**
  - **Messaggio –** Si lancia un messaggio specifico
  - **Segnale –** Si lancia un segnale generico, intercettato da tutti i processi che hanno un evento intermedi di tale tipo. Spetterà ai singoli processi capire se il segnale è di loro interesse.
  - **Tempo** – Si verifica dopo un certo lasso temporale
  - **Condizione** – Si verifica allorché accade una specifica condizione
  - **Multiplo** – Specifica una combinazione logica di eventi
  - **Errore** – Specifica l'avvenuto errore Specifica una combinazione logica di eventi

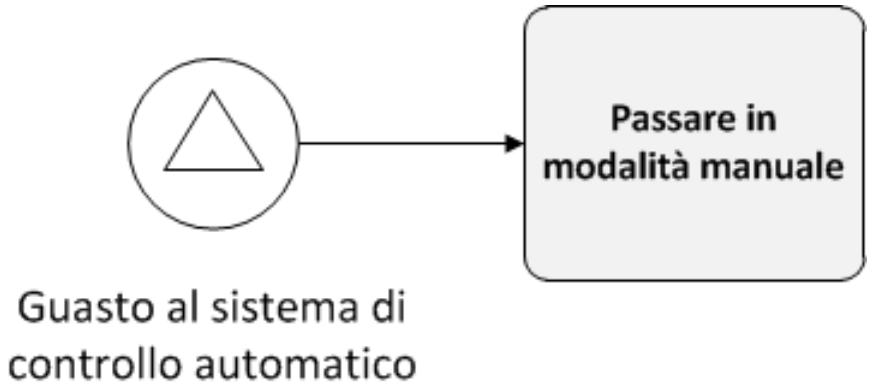
BPMN: Flow Objects

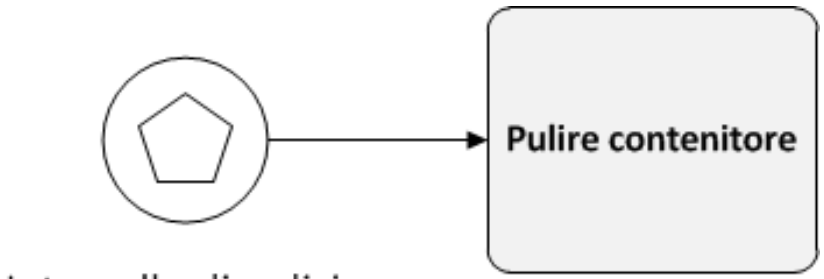
2.1. Eventi: rappresentano qualcosa che accade durante il processo e scatena un'azione e/o influenza i risultati

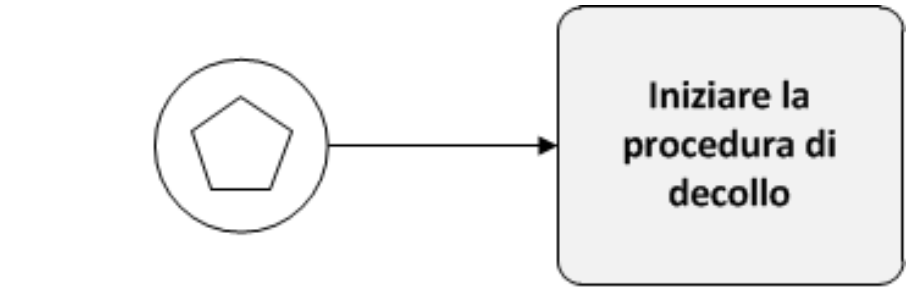
- **Gli eventi base indicano l'istante in cui si verifica un certo evento**
  - **Iniziale –** Genera il gettone che dà origine al processo
  - **Intermedio Catching** – Si verifica all'interno del processo. Quando un gettone raggiunge tale evento vi stazione in attea che si verifichi il processo indicato
  - **Intermedio Throwing** -– Si verifica all'interno del processo. Quando un gettone raggiunge tale evento, scatena l'evento corrispondente (che sarà intercettato da altri processi) e prosegue senza interruzioni
  - **Finale** – Alla fine del processo il gettone viene «distrutto». Se l'evento finale non è di tipo semplice, ma è associato ad un evento di tipo throwing alla distruzione del gettone corrisponde, anche, la generazione di un nuovo evento. In pratica la fine di un processo ne attiva di nuovi.

Flow Objects: Events









BPMN: Flow Objects

Gli eventi intermedi possono essere usati per:

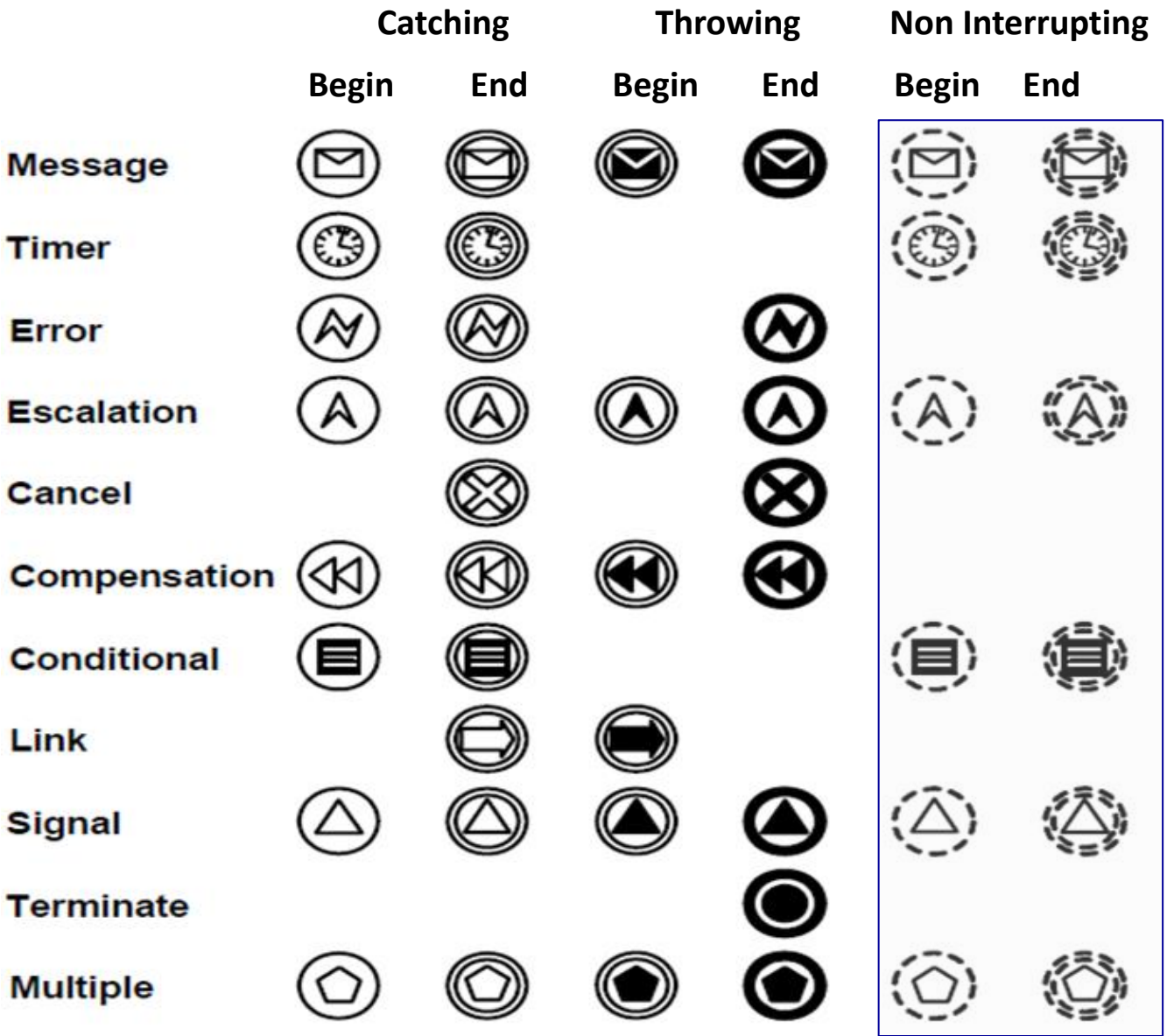
- Mostrare il punto in cui si attendono o si spediscono dei messaggi
- Mostrare punti in cui si possono verificare dei ritardi
- Modificare il flusso standard in relazione a talune eccezioni
- Mostrare casi in cui è necessario «lavoro extra»

Gli eventi intermedi possono essere di due tipi:

- **Tipo Catch**: *rispondono ad eventi lanciati da altri processi*
  - In questo caso quando un token arriva all' evento catch, il token si ferma (la transazione si arresta) fintanto che un evento throw non lancia un omologo segnale che, una volta intercettato, sblocca la transazione
- **Tipo Throw**: *inviano un segnale ad altri processi*
  - In questo caso quando un token arriva all'evento throw, il token prosegue lungo il flusso principale, mentre l'evento genera un segnale che può essere intercettato da un analogo evento di tipo catch

Gli eventi boundary possono essere solo di tipo catch

BPMN: Flow Objects

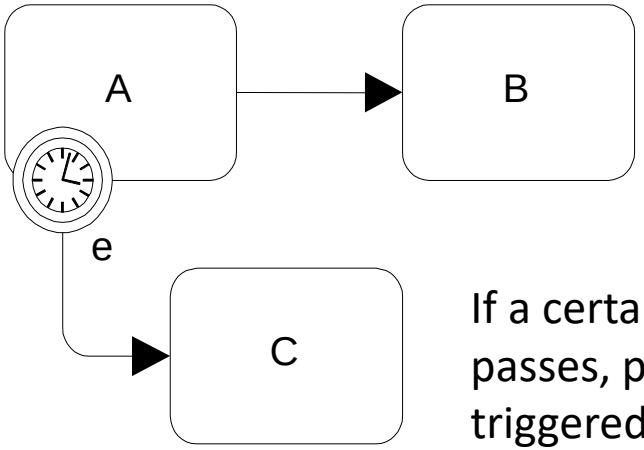


Nuovi nella versione 2.0

In caso di evento il processo non viene sospeso, ma prosegue in parallelo

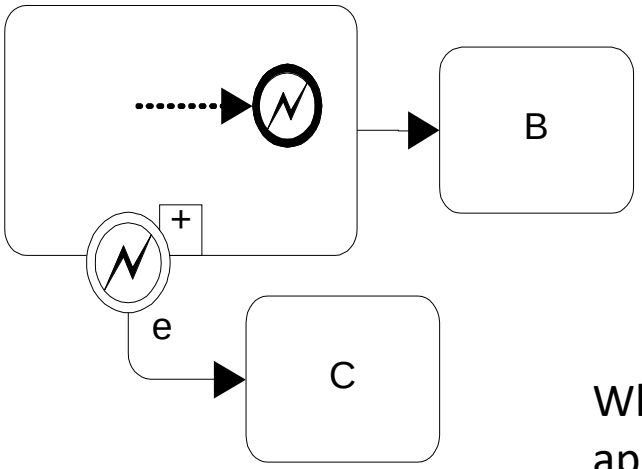
Flow Objects: Events

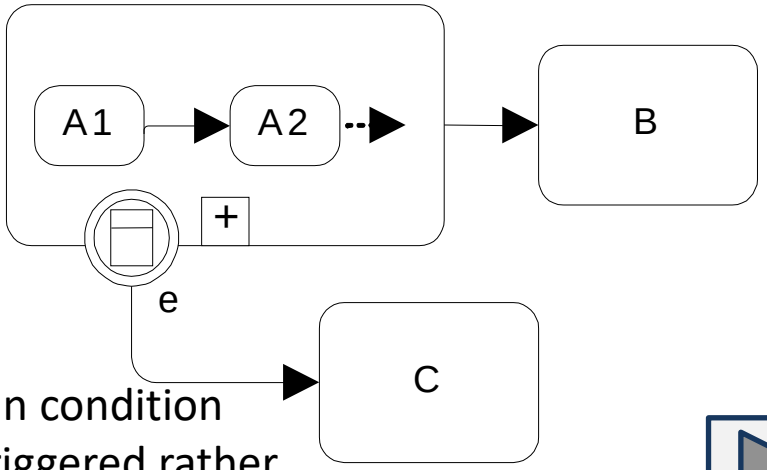
- **Time events**
- **Throw events**
- **Condition events**



If a certain time passes, process C is triggered

In case of error (inside A) C is triggered instead of C

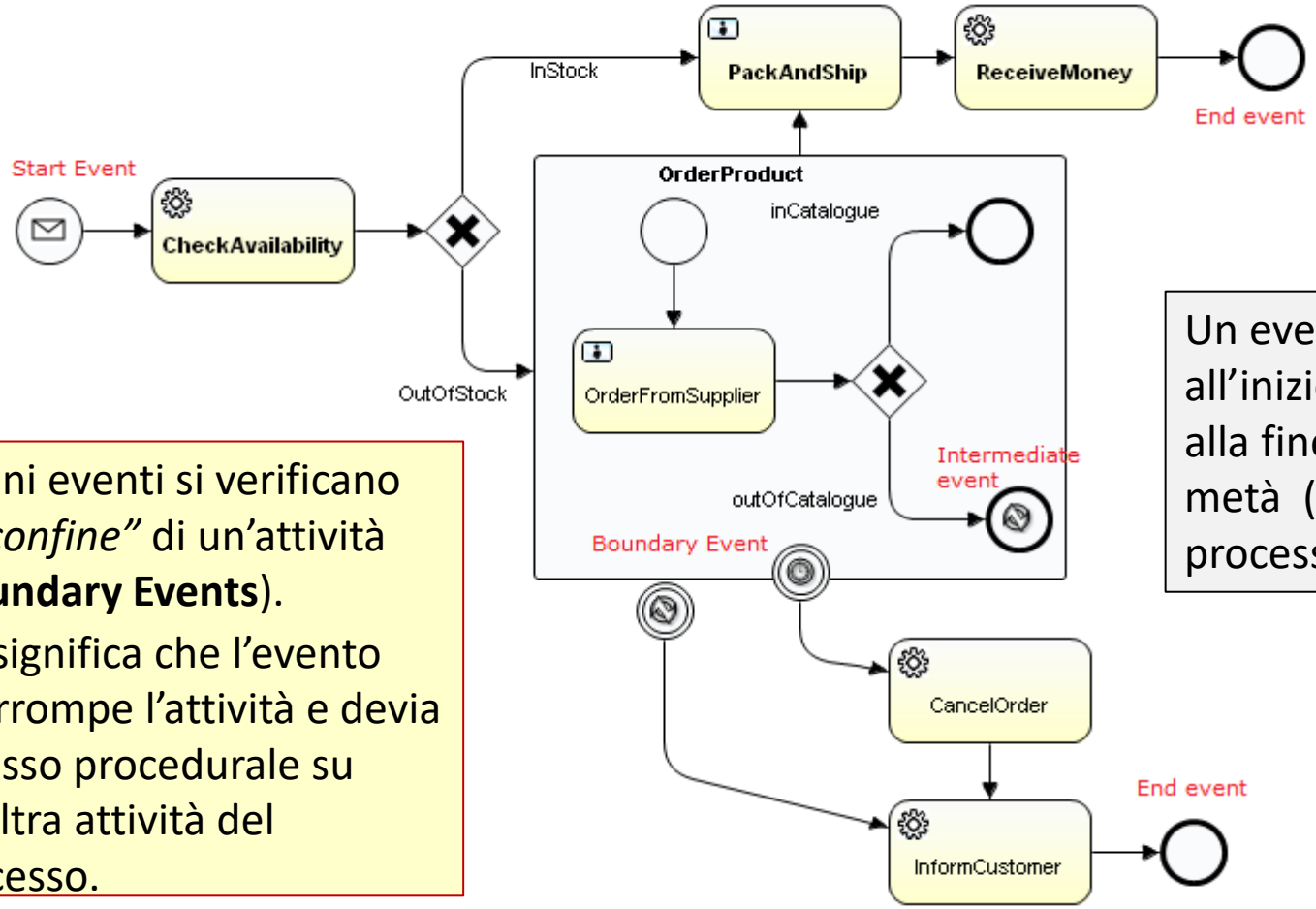




When a certain condition applies, C is triggered rather than B

BPMN: Flow Objects

- Il vantaggio di BPMN sta nell'introduzione degli eventi.
- Un evento segnala l'accadimento di un certo fatto; BPMN permette di modellizzare il modo in cui il sistema reagisce all'evento.



Un evento può verificarsi all'inizio (**Start Events**) o alla fine (**End Events**) o a metà (**Intermediate**) del processo

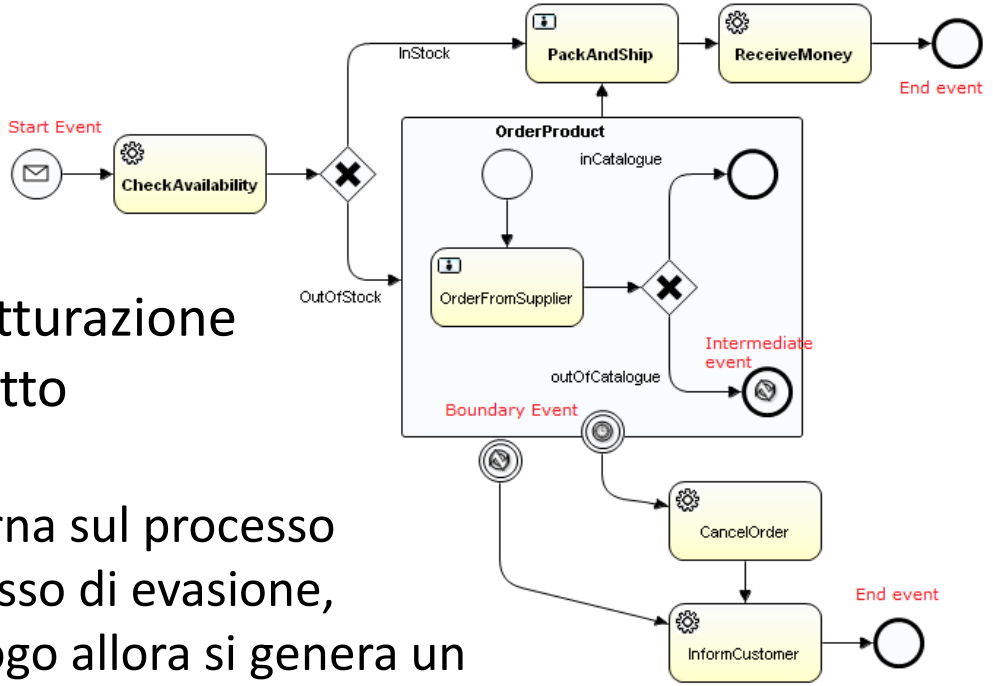
Alcuni eventi si verificano *"al confine"* di un'attività (**Boundary Events**).

Ciò significa che l'evento interrompe l'attività e devia il flusso procedurale su un'altra attività del processo.

BPMN: Flow Objects

Processo di evasione di un ordine

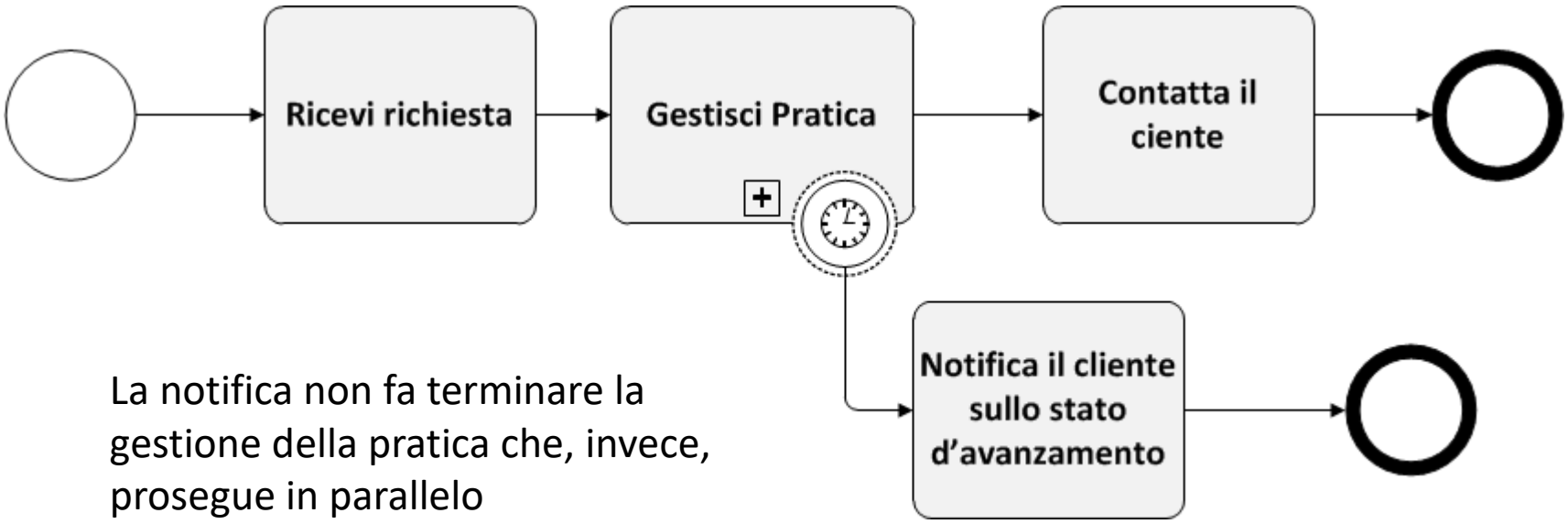
- Ricevuto l'ordine si controlla la disponibilità
- Se disponibile a magazzino si procede con la spedizione e la fatturazione
- Se non disponibile si entra nel sotto processo di ordinazione
  - Se il prodotto è a catalogo si torna sul processo principale e si conclude il processo di evasione,
  - Se il prodotto non è più a catalogo allora si genera un errore
  - L'errore attiva l'evento "boundary" che genera la cancellazione dell'ordine e, successivamente, fa avvisare il cliente.
  - Si noti che, anche in caso di ritardi nella risposta del fornitore si avvisa il cliente senza cancellare l'ordine



Flow Objects: Events

Non Interrupting Events

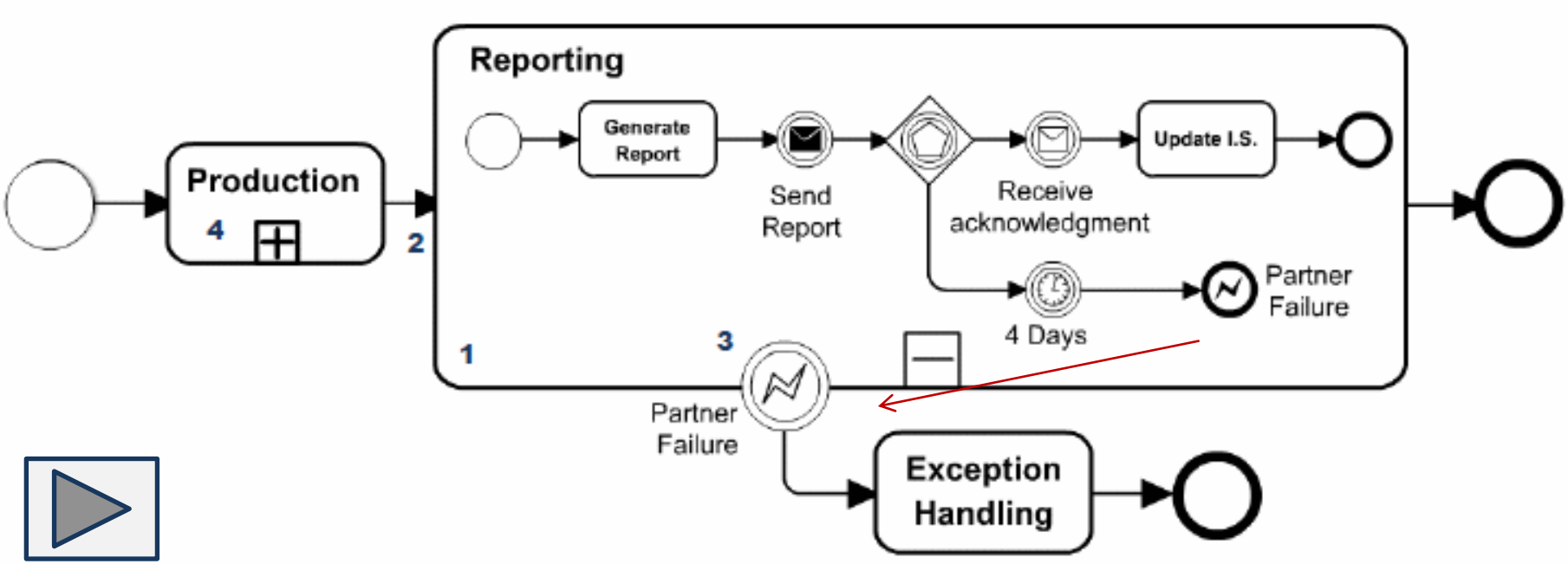
- A volte è auspicabile che il processo reagisca ad un certo evento senza che venga interrotta l'attività che era attiva (nel momento in cui si è manifestato l'evento)
- Questo può essere realizzato utilizzando gli eventi intermedi di tipo non-interrupting
- Tali eventi sono caratterizzati da contorni tratteggiati



La notifica non fa terminare la gestione della pratica che, invece, prosegue in parallelo

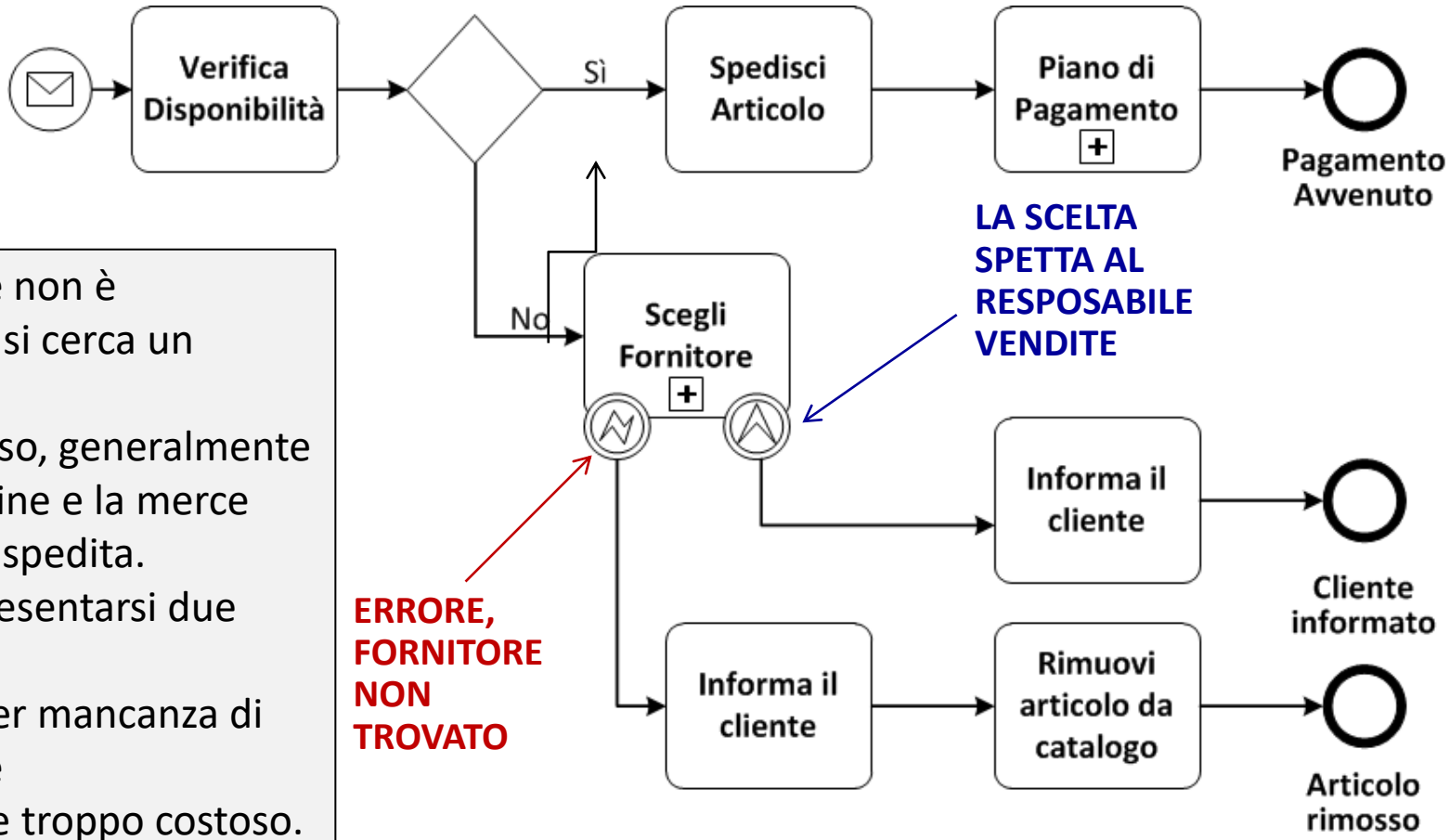
BPMN: Flow Objects

- Nello schema sono presenti due sotto-processi:
  - uno in forma espansa (1)
  - l'altro in forma compatta (4)
- Se si raggiunge l'evento finale «partner failure» (finale relativamente a (2)), l'esecuzione riparte automaticamente dal punto definito dall' omonimo evento di tipo catch (3)



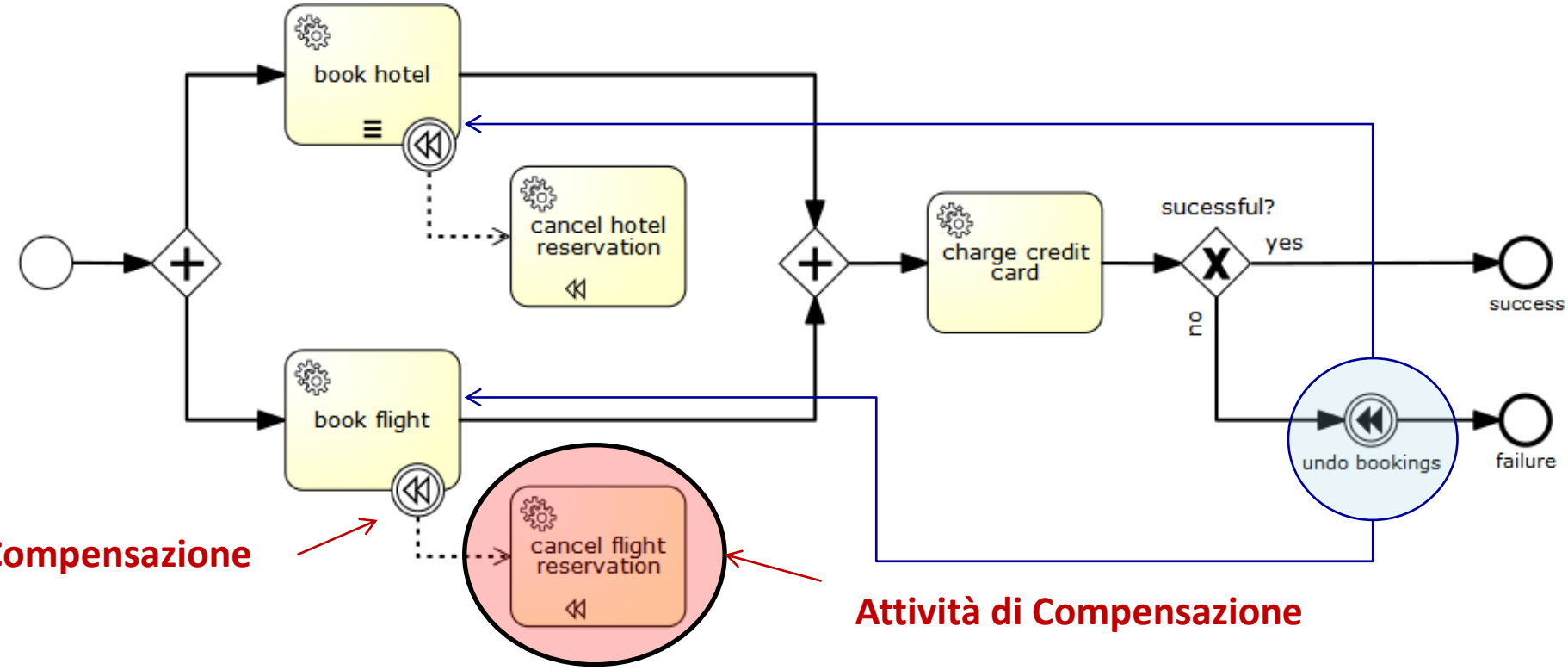
BPMN: Flow Objects

- Se la merce non è disponibile si cerca un fornitore.
- Tale processo, generalmente va a buon fine e la merce può essere spedita.
- Possono presentarsi due problemi:
  - Errore per mancanza di fornitore
  - Fornitore troppo costoso. In questo caso serve il parere del responsabile (***escalation***)



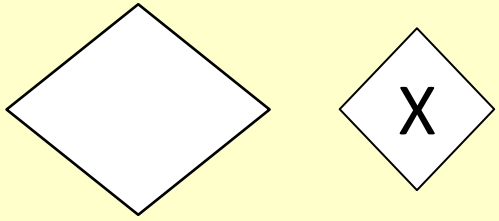
BPMN: Flow Objects

Si deve prenotare un volo e una camera d'hotel. Le due attività avvengono in parallelo. Tuttavia, se una di esse non va a buon fine è necessario cancellare anche l'altra.
Se entrambe vanno a buon fine si paga con carta di credito. Se la transazione ha successo il processo finisce, altrimenti è necessario cancellare entrambe le prenotazioni.
Tale comportamento è gestito **con due eventi di compensazione di tipo boundary** e un **evento intermedio, sempre di compensazione**

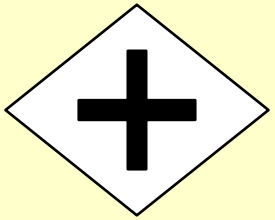


BPMN: Flow Objects

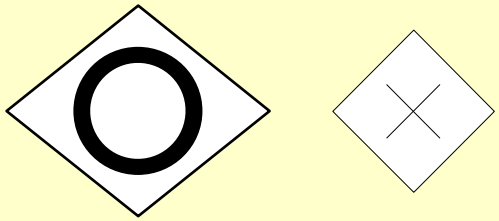
2.2. Gateway: rappresentano i punti che condizionano il flusso del processo



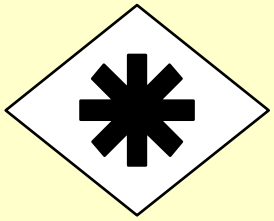
ESCLUSIVO (XOR)



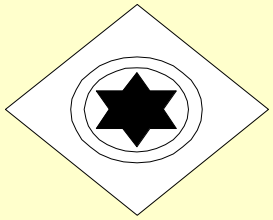
PARALLELO (AND)



INCLUSIVO (OR)



COMPLESSO

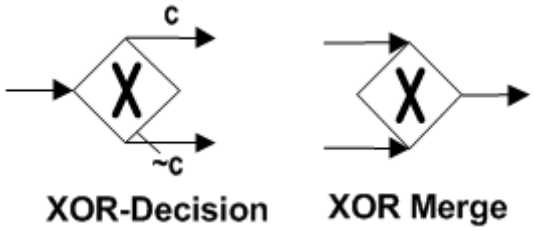


CONDIZIONALE

BPMN: Flow Objects

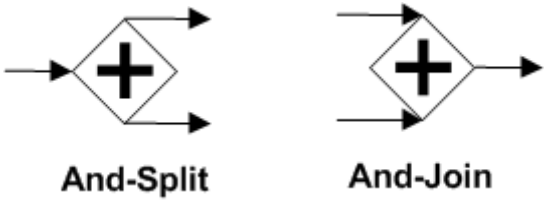
Exclusive (XOR)

- **Exclusive Decision**
  si prosegue lungo un solo ramo uscente
- **Exclusive merge**
  Si procede quando almeno un ramo entrate è stato completato



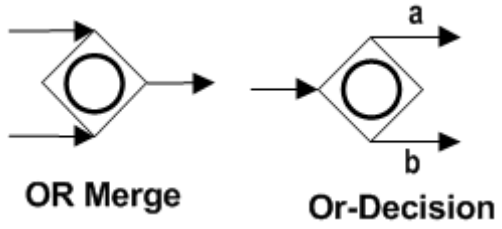
Parallel (AND)

- **Parallel split**
  si procede lungo tutti i rami uscenti
- **Parallel join**
  si procede quando tutti i rami entranti sono stati completati

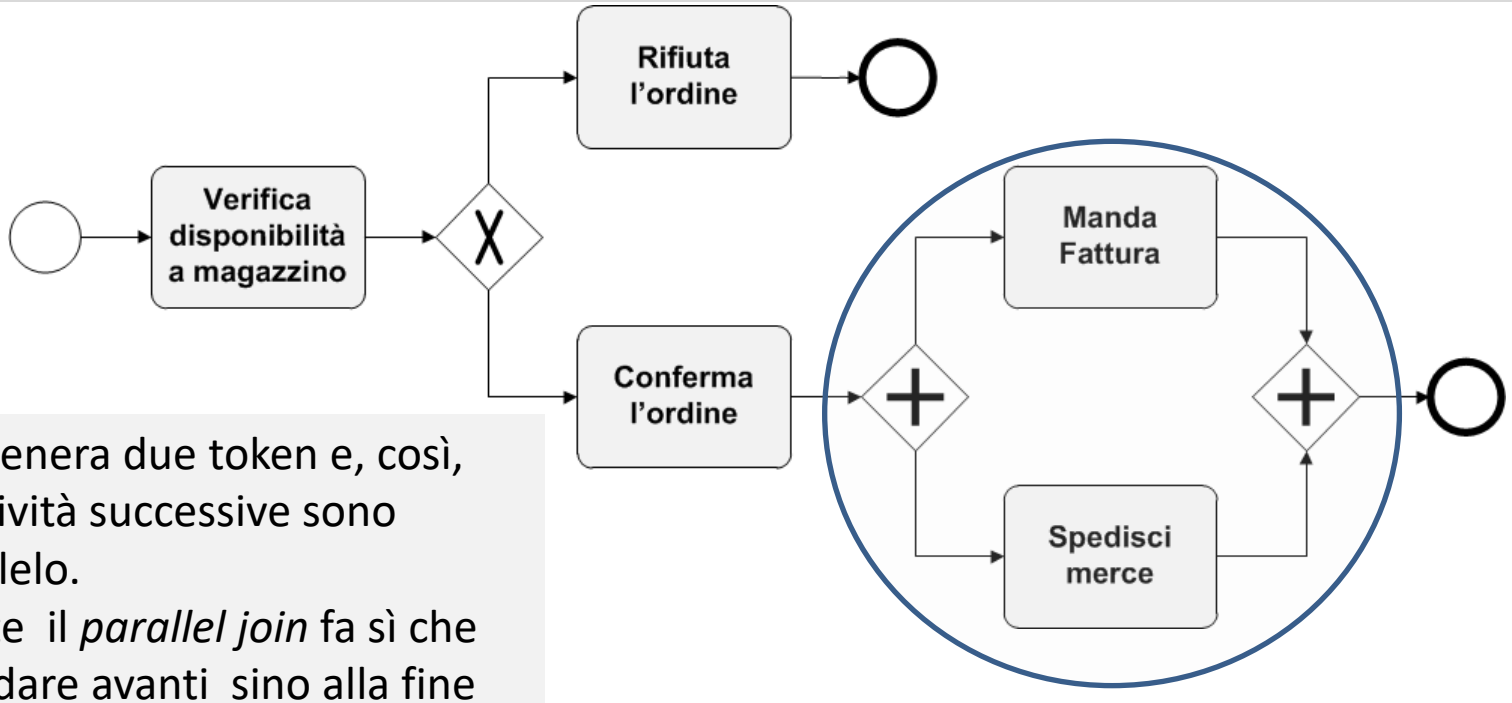


Inclusive (OR)

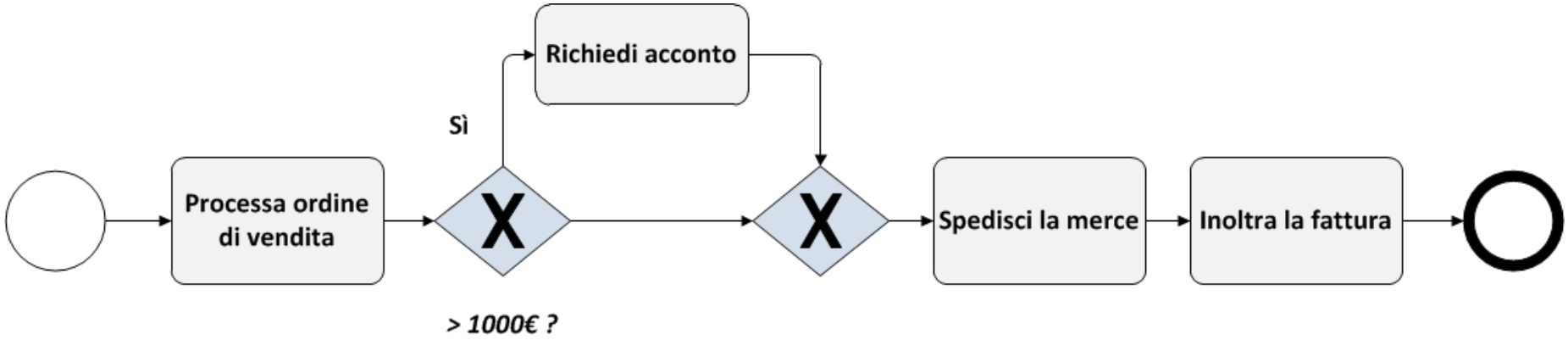
- **Inclusive decision**
  si procede su uno più rami a seconda delle condizioni
- **Inclusive merge**
  si procede quando almeno un ramo entrante (attivo) è stato completati



Exclusive & Parallel Gateways

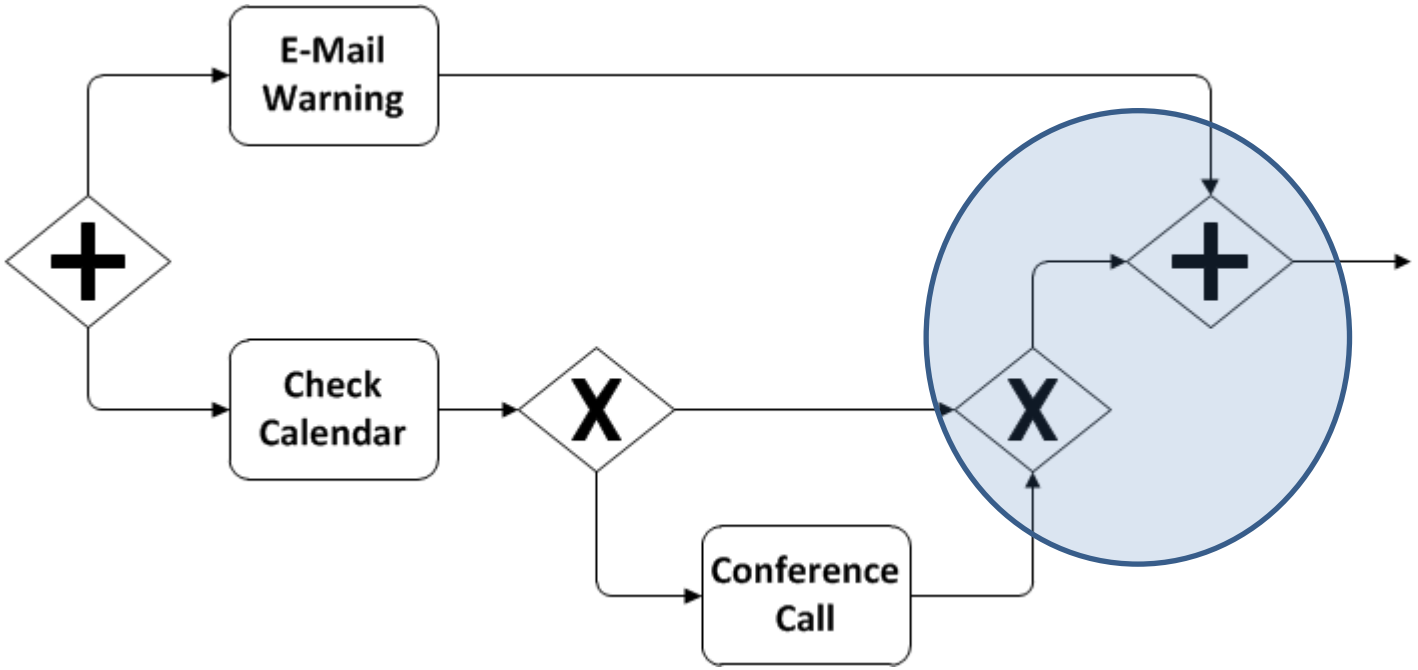


Il *parallel split* genera due token e, così, entrambe le attività successive sono attivate in parallelo.
Successivamente il *parallel join* fa sì che non si possa andare avanti sino alla fine delle due attività parallelo



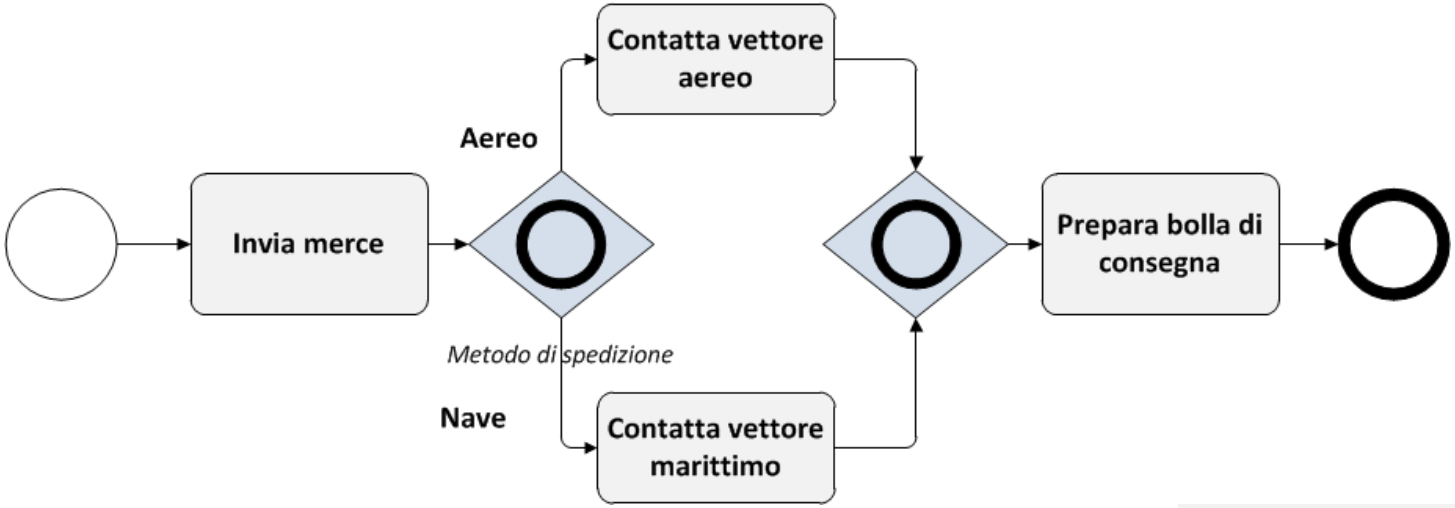
Exclusive & Parallel Gateways

How does this process work?!?

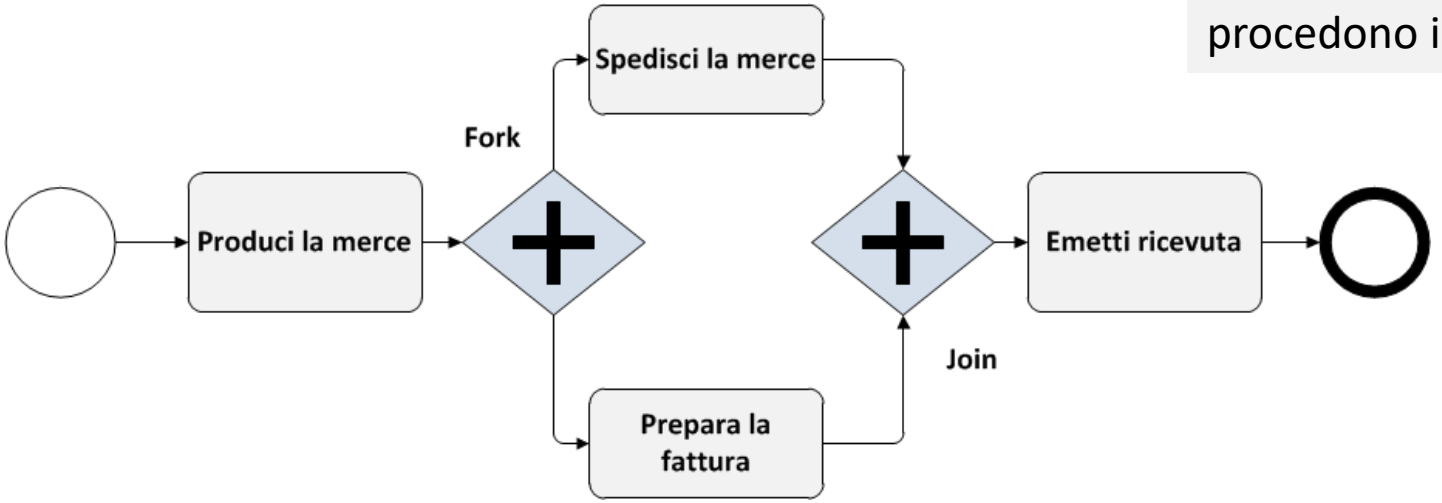


Two tokens are generate by the parallel split. Next, the intervening Exclusive gateway reduces the number of incoming path to the parallel gateway. Everything works fine

Inclusive and Parallel Decision

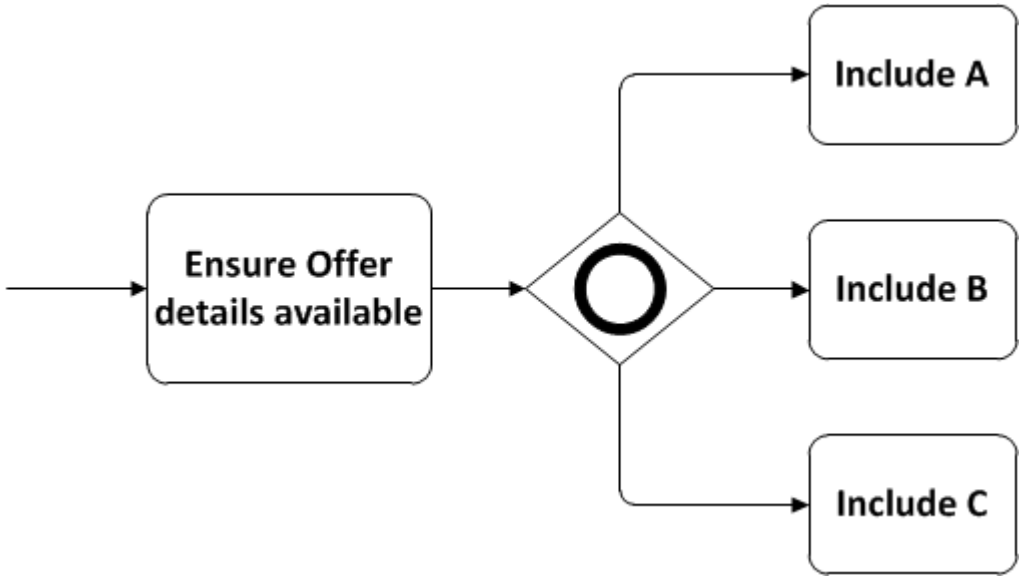


Nel primo caso si attiva solo un'attività.
Nel secondo le due attività procedono in parallelo



Inclusive Gateway

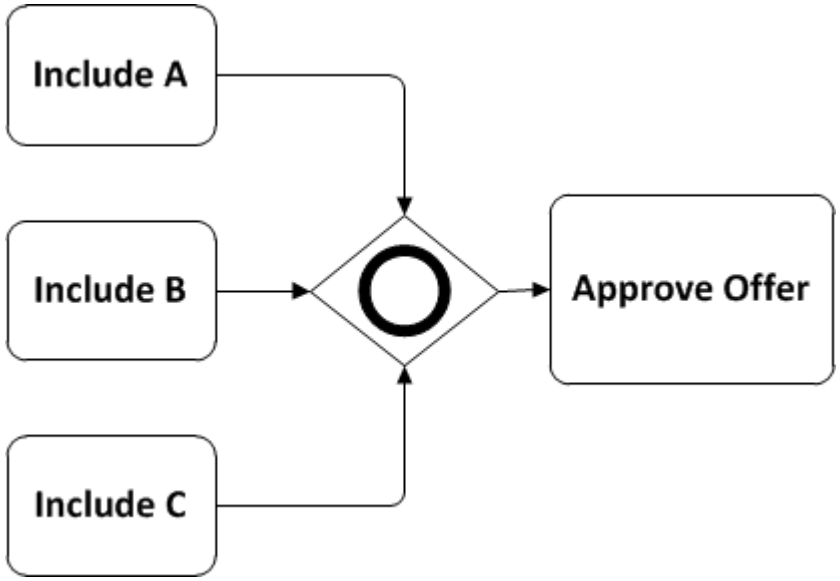
- Inclusive Gateways support decisions where **more than one outcome is possible** at the decision point.
- Inclusive Gateway with multiple outgoing Sequence Flow **creates one or more paths based on the conditions on those Sequence Flow**
- When a token arrives at an Inclusive Gateway, there is an immediate evaluation of all the conditions that are on the Gateway's outgoing Sequence Flow.
- **Every condition that evaluates to true will result in a token moving down** that Sequence Flow.
- At least one of those conditions *must evaluate to true.*



- How many tokes will come out from the inclusive gateway?
- It depends. At least one, but even three if all the conditions are satisfied!!!

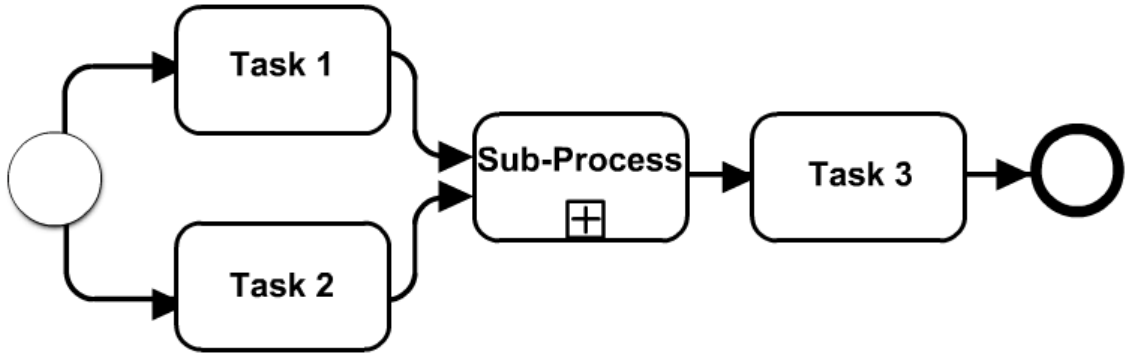
Inclusive Gateway

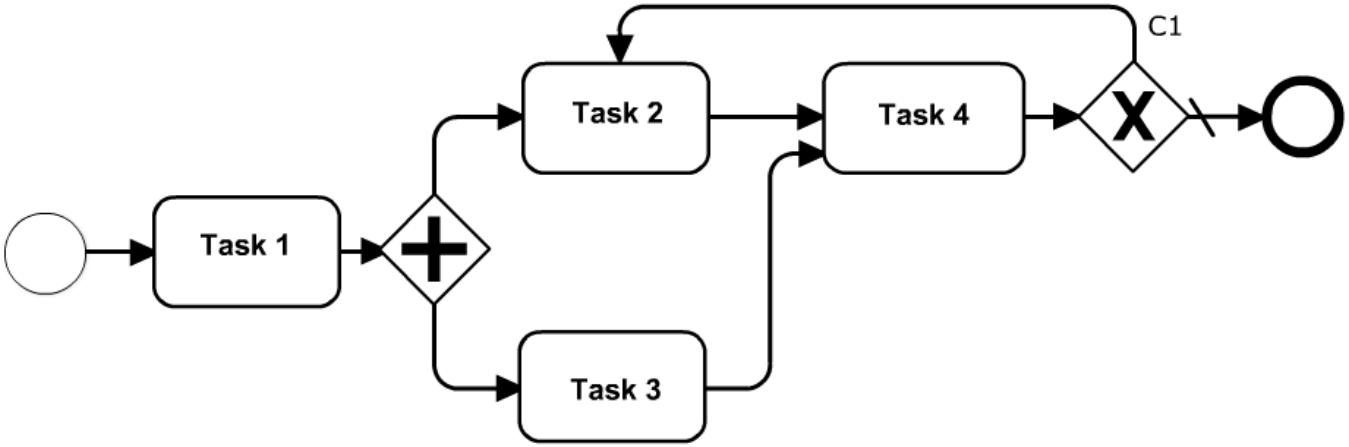
- In order to synchronize flows (the same also apply to the other gateways), we need to place, somewhere downstream the process, another inclusive gateway.
- Only one token will flow downstream that gateway.



- Specifically, when the first token arrives at the Gateway, the Gateway will “look” upstream for each of the other incoming Sequence Flow to see if there is a token that might arrive at a later time
- The Gateway will hold the first token that arrived in the upper path until the other token from the lower path arrives
- When all the expected tokens have arrived at the Gateway, the Process flow is synchronized (the incoming tokens are merged) and then a token moves down the Gateway’s outgoing Sequence Flow

Uncontrolled Flow

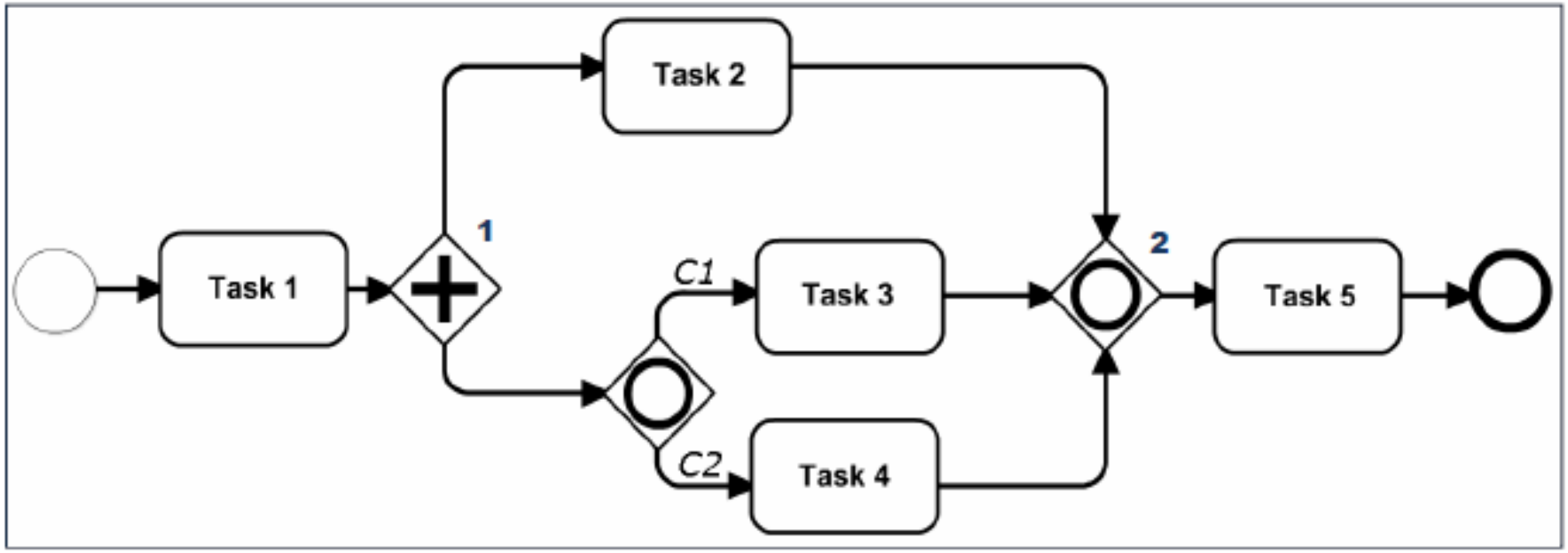




Si tratta di esempi di flussi «incontrollati»

- Nel primo si generano due task che proseguono indipendentemente lungo il loro percorso. Entrambi attraversano il Sub-Process ; chi arriva prima è processato per primo
- Nel secondo è il parallel split a creare due token. Non essendoci un parallel join (di chiusura) entrambi i token fluiscono lungo il loro percorso. Ciascuno di essi incontrerà il gateway esclusivo e potrà procedere verso l'uscita o tornare al Task 2

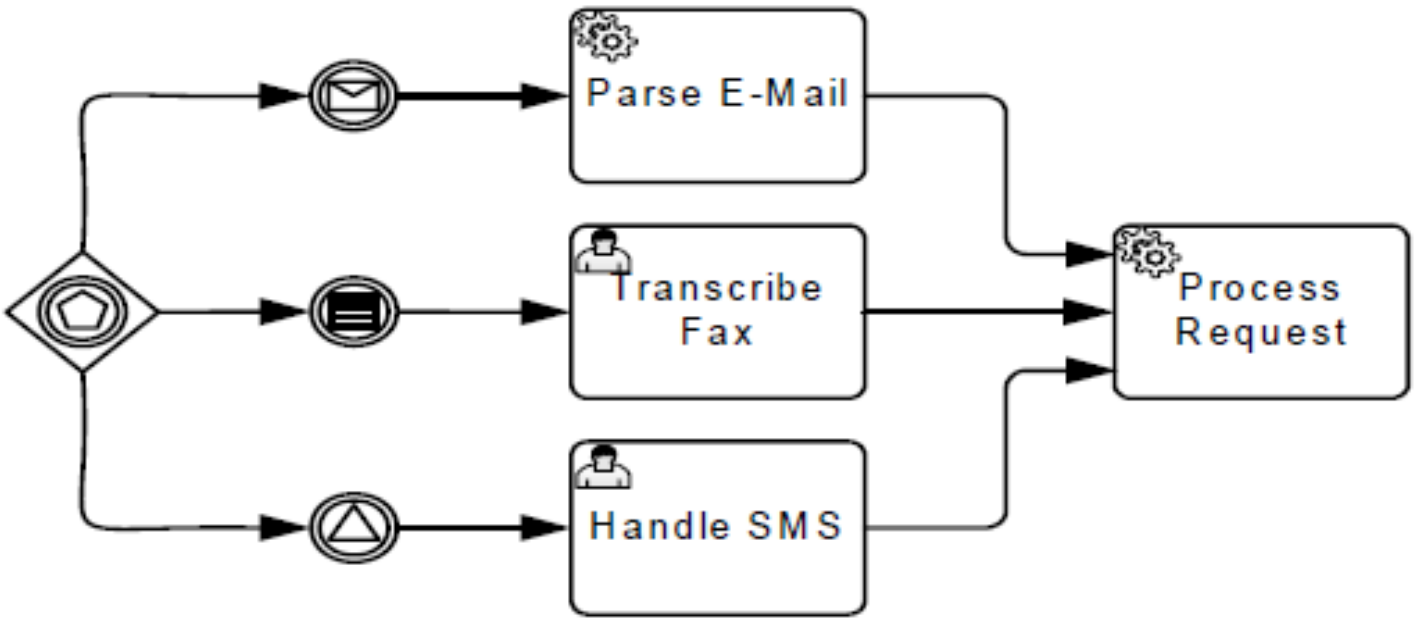
A more complex model



How does this process work?

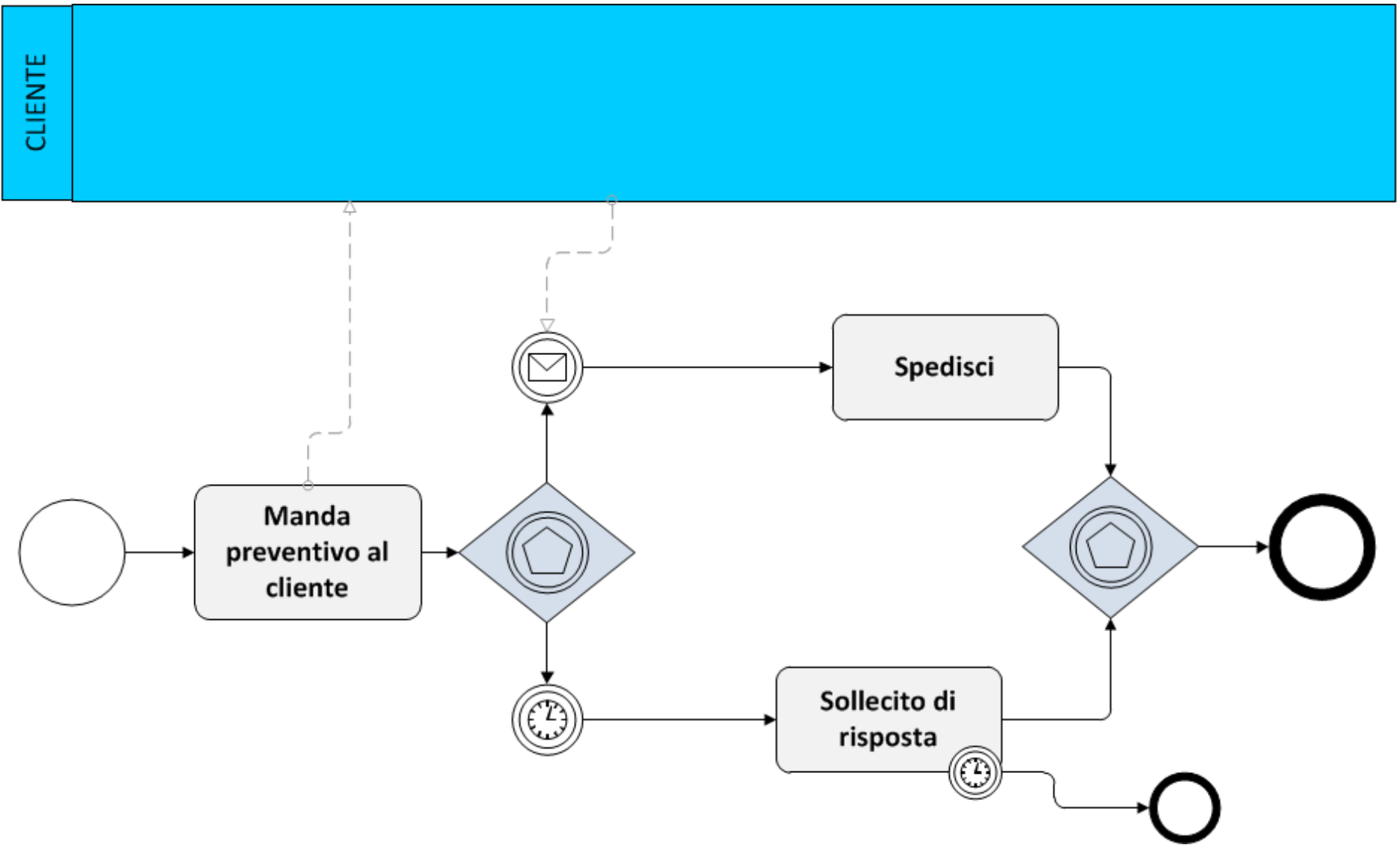
What would happen if the inclusive gateway was substituted with a parallel one?

Event Based Decision



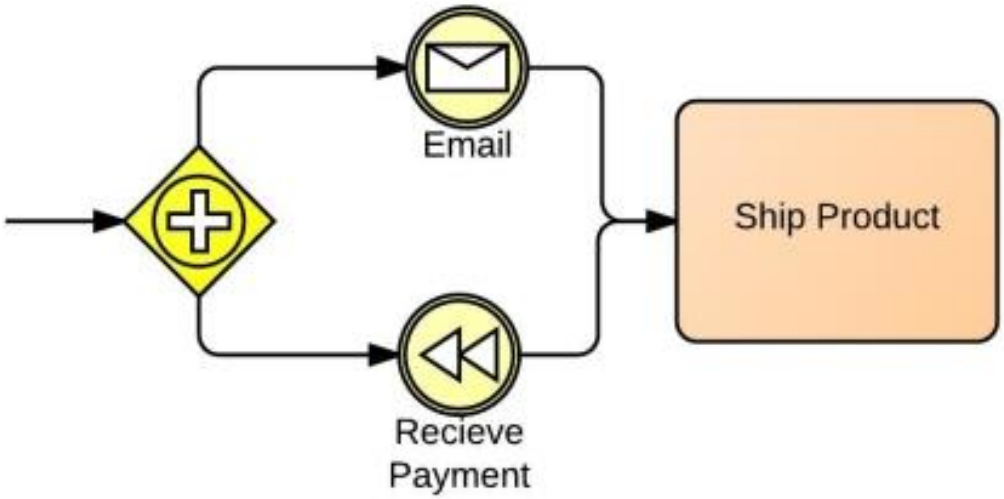
- When the token arrives at the “event based gateway” it ***waits there until one of the events*** (at the immediate left of the gateway) ***is triggered***.
- A ***race pattern logic*** is followed, i.e., the first occurring event defines the path that will be followed by the token.
- For instance, in the above diagram, if a message arrive, then the email is parsed. If a fax arrive, the fax is transcribed. In case of a non standard signal the signal is handled.

Event Based Decision



Event Based Decision – Parallel Event Based

- As the name suggest, this gateway is similar to a parallel gateway
- It allows for multiple processes to happen at the same time, but unlike the parallel gateway, the processes are event dependent.
- You can think of a parallel event-based gateway as a non-exclusive, event-based gateway where multiple events can trigger multiple processes, but the processes are still event dependent.



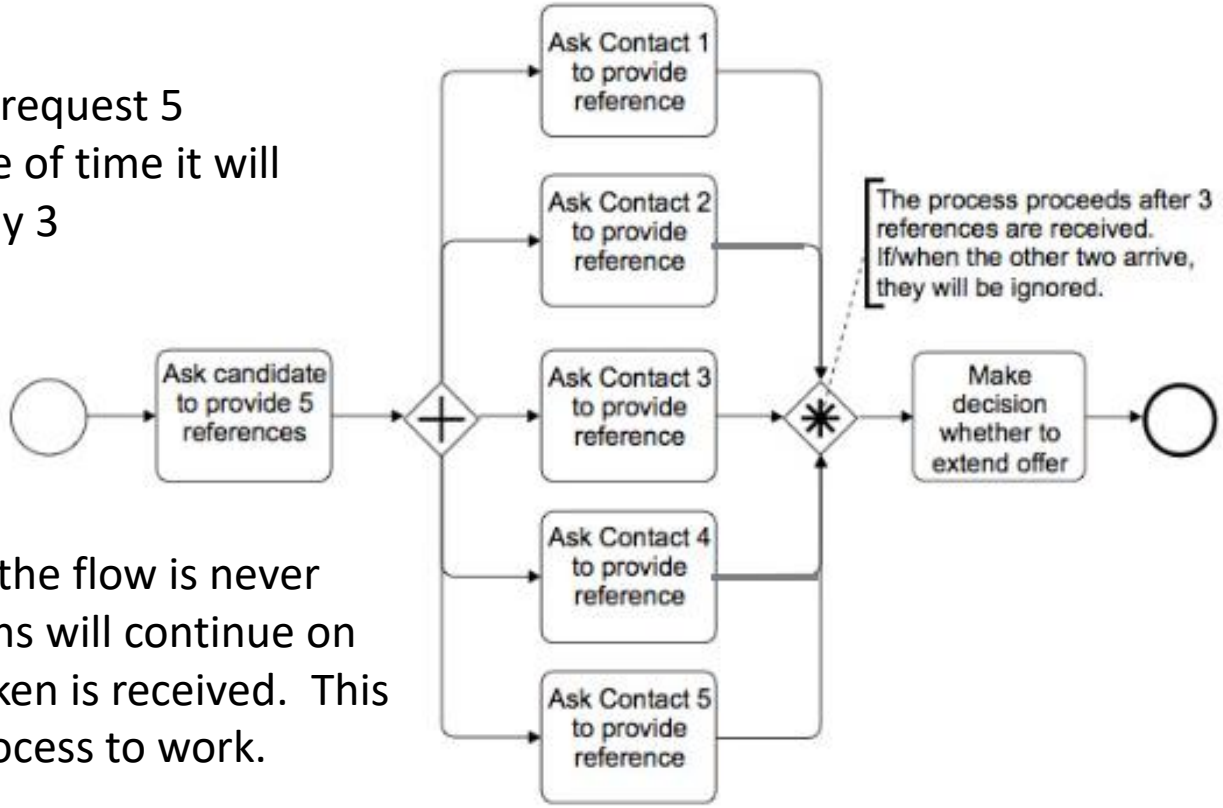
In this case, a soon as a confirming e-mail and a payment acknowledgment are received, the product is shipped

Complex Decision – Complex Gateways

- As the name signifies, complex gateways are only used for the most complex flows in the business process. Indeed, complex Gateways allow modelers to create their own rules to *the merging and/or splitting behavior* of the Gateway
- So, if you need multiple gateways to describe the business flow, then that's an ideal case for the complex gateway. the complex gateway may send a token down one or more of the Gateway's outgoing Sequence Flows
- For example, an outgoing assignment may evaluate Process data and then select different sets of outgoing Sequence Flow, based on the results of the evaluation
- However, the outgoing assignment should ensure that at least one of the outgoing Sequence Flow will always be chosen
- There are many patterns that can be performed with the Complex Gateway, such as typical Inclusive Gateway behavior, batching of multiple tokens, accepting tokens from some paths but ignoring the tokens from others, etc.
- The Gateway looks the same for each of these patterns, so Complex gateways need more descriptive text because the functioning (i.e., the underlying logic) must be explained using words in place of symbols

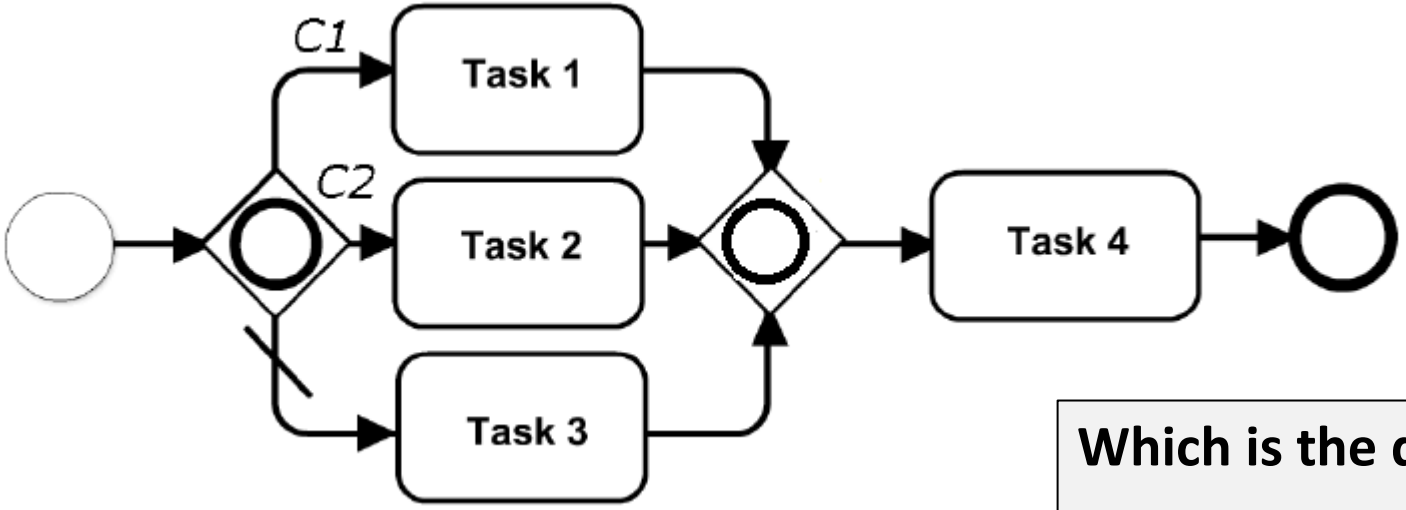
Complex Decision – Complex Gateways

- The process is intended to request 5 references, but for the sake of time it will proceed after receiving only 3



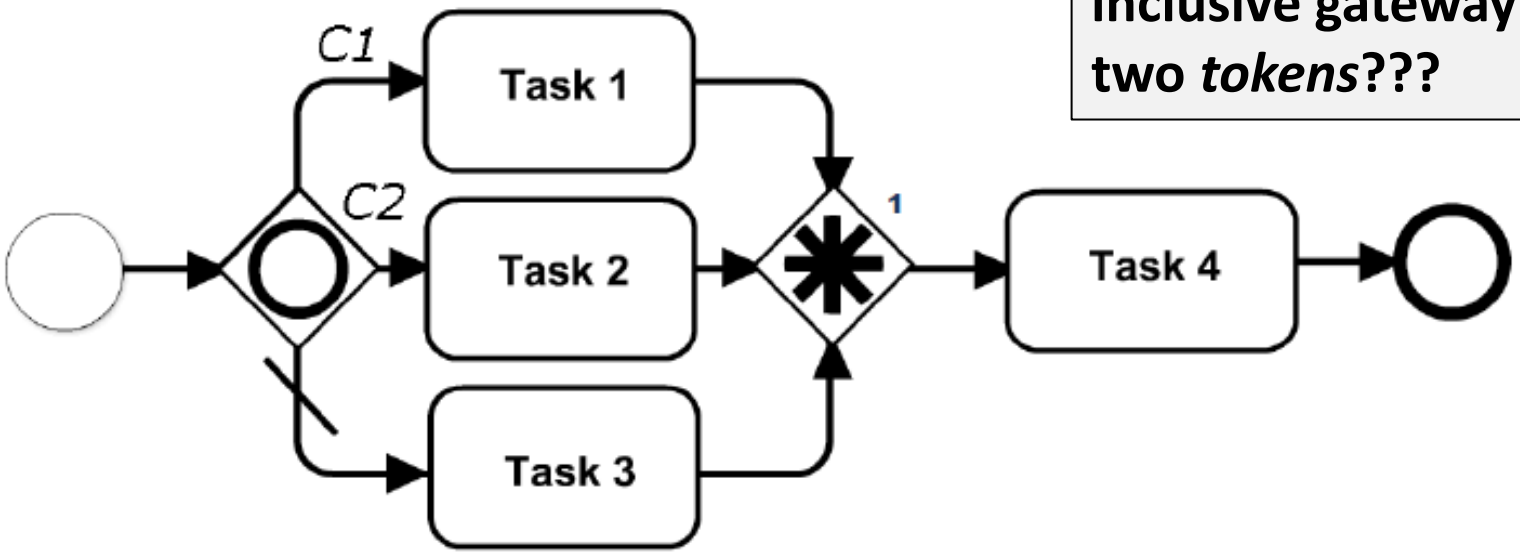
- If no gateway is used then the flow is never synchronized and the tokens will continue on to the next task as each token is received. This is not how we want the process to work.
- An exclusive gateway would behave the same way when used to merge flows. If a parallel gateway is used then the gateway would wait for all 5 tokens to arrive before proceeding. This also is not how we intend for the process to work.
- Similarly, an inclusive gateway waits for all tokens that were emitted from the corresponding splitting gateway.
- Therefore, in this instance the best solution is the Complex Gateway.

Complex Decision



Which is the difference?

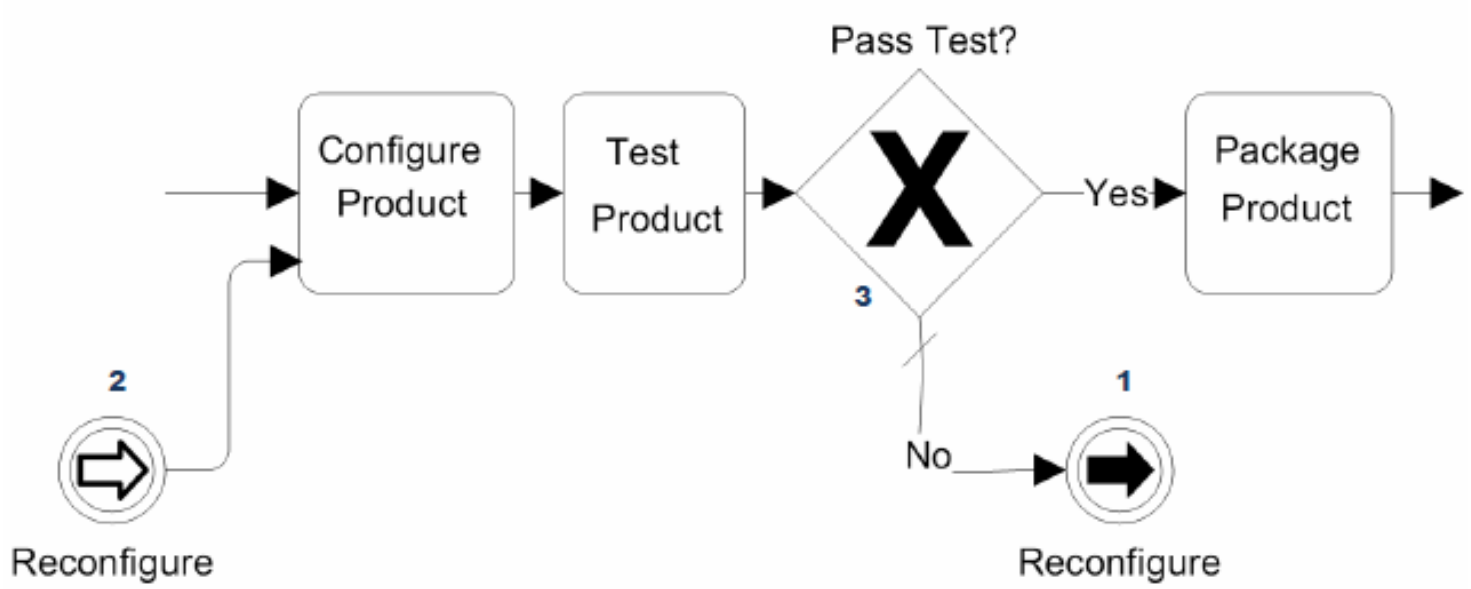
What could happen if the inclusive gateway produces two *tokens*???



Collegamento

BPMN consente anche la segmentazione dei processi, ad un dato livello, per creare diverse sezioni modulari

Nel processo qui riportato, se si attiva in ramo «no» (3) del gateway, il flusso torna all'evento Reconfigure (2) e, in pratica, si ottiene un ciclo

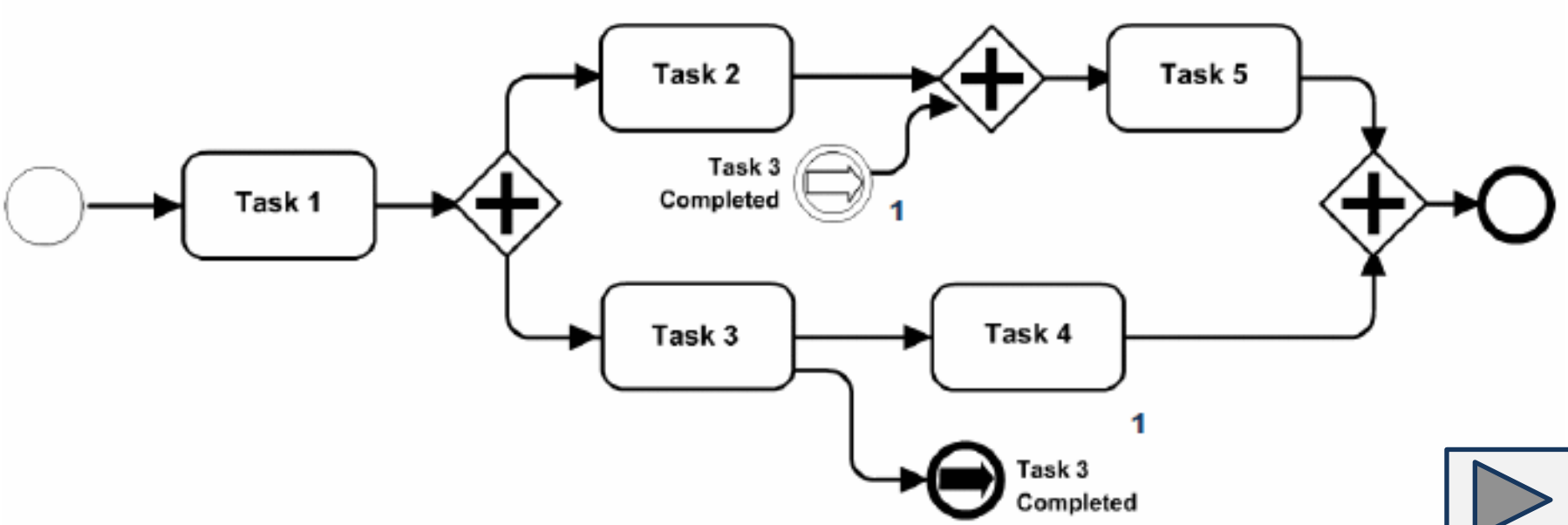


Collegamento

Il collegamento permette anche di realizzare il concetto di «milestone»

In pratica un certo evento è abilitato solo quando il processo ha raggiunto uno specifico stato.

Ad esempio uno studente può iscriversi solo dall'apertura alla chiusura del periodo di iscrizione

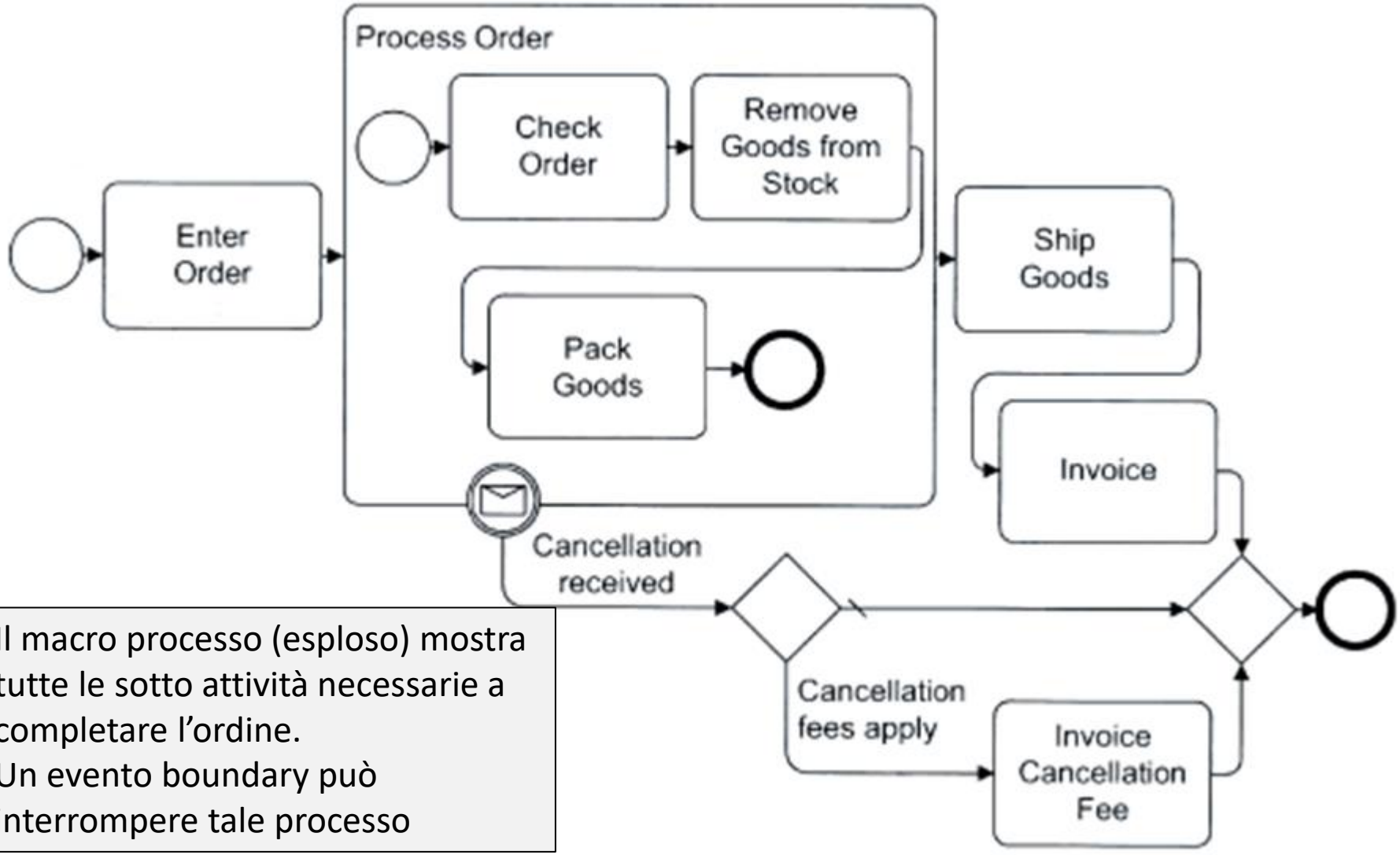


BPMN: Flow Objects

- **2.3. Task: rappresentano i processi che devono essere mappati (ciò che l'azienda vuole informatizzare)**

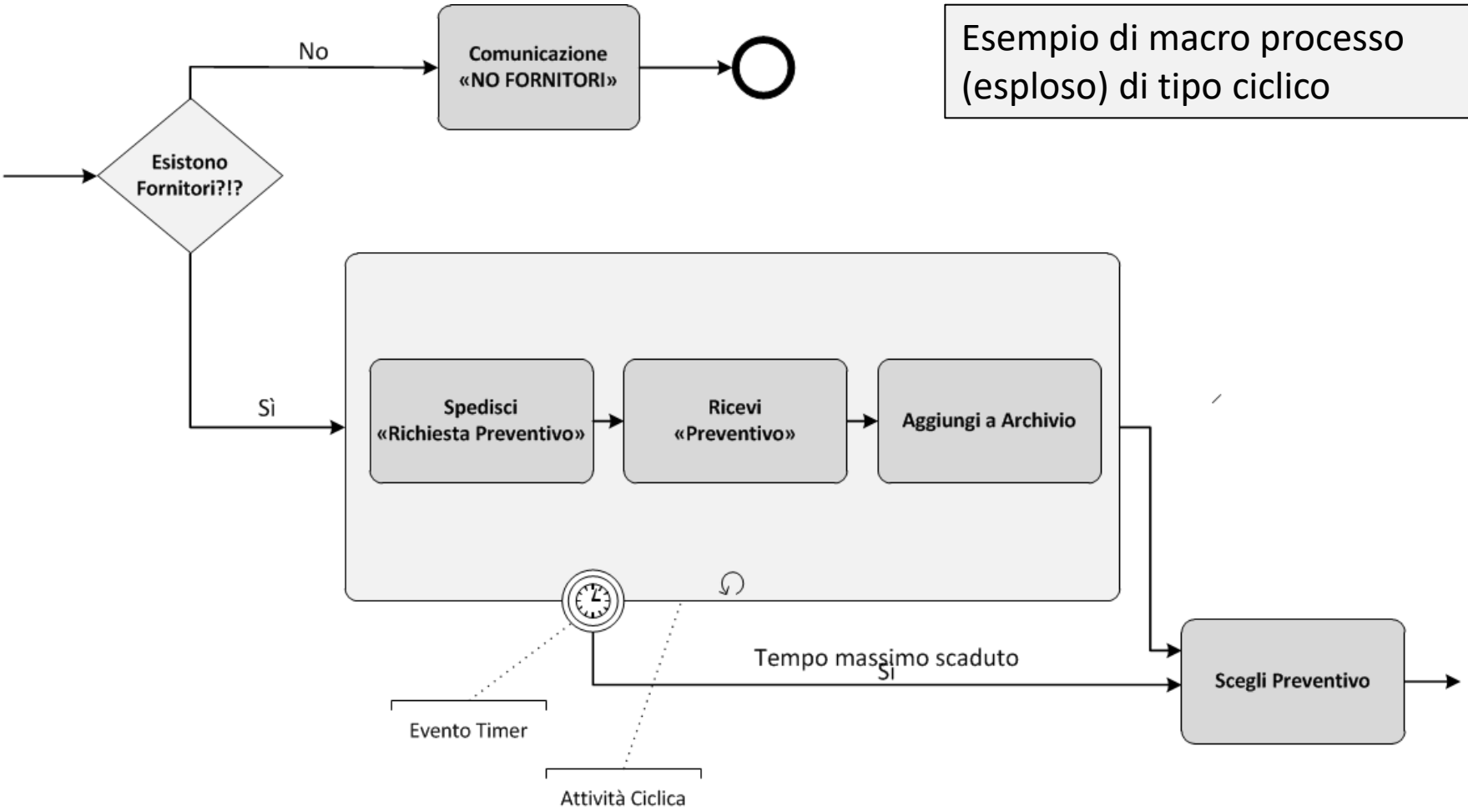
task	Processo Semplice	Multi instance task |||	Processo multiplo composto da attività seriali
loop ↻	Processo ciclico che si ripete un certo numero di volte	Multi instance task ≡	Processo multiplo composto da attività parallele
Collapsed Subprocess +	Macro Processo. Non visualizza le attività che lo costituiscono		

BPMN: Flow Objects



Il macro processo (esploso) mostra tutte le sotto attività necessarie a completare l'ordine.
Un evento boundary può interrompere tale processo

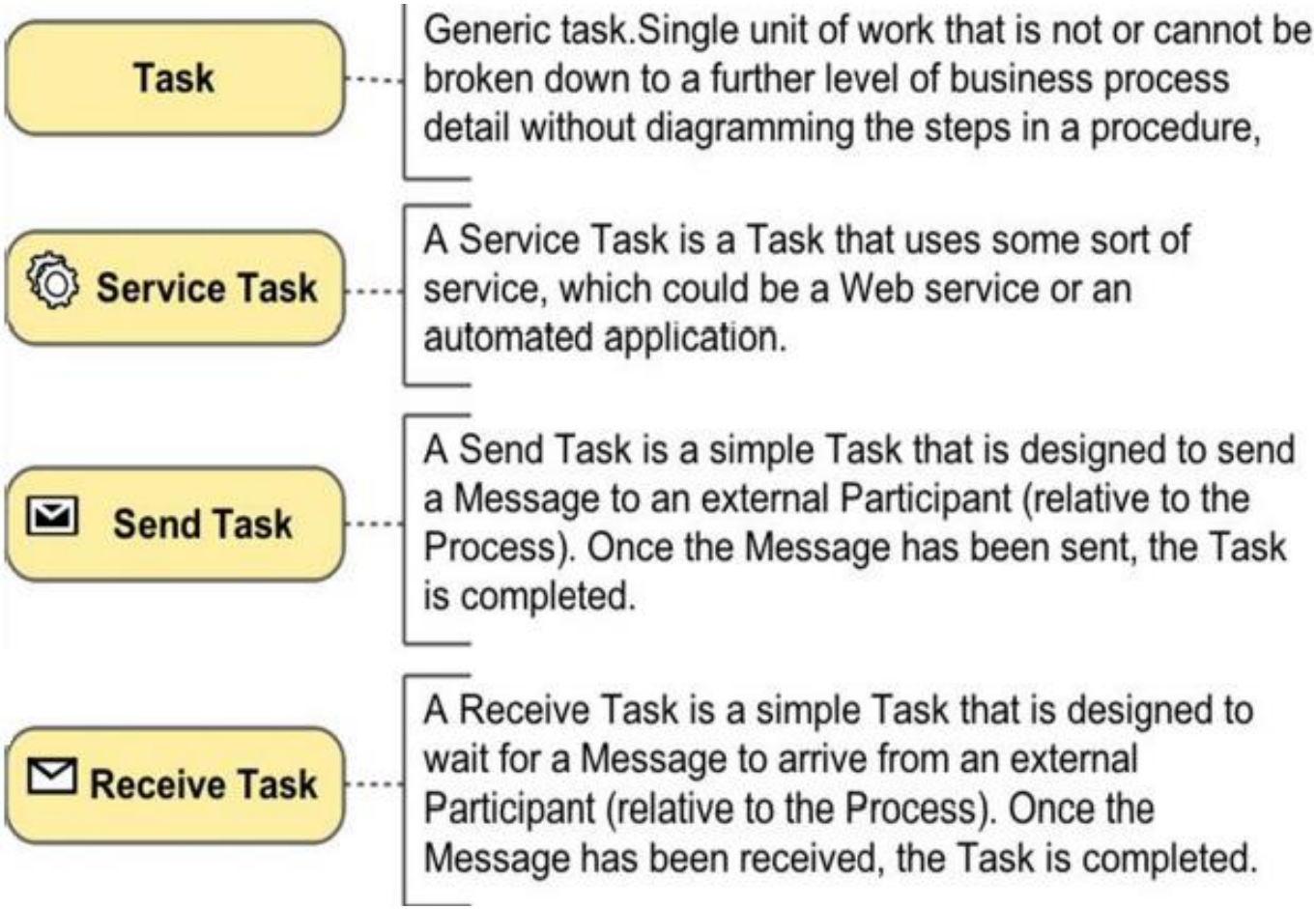
BPMN: Flow Objects



Esempio di macro processo (esploso) di tipo ciclico

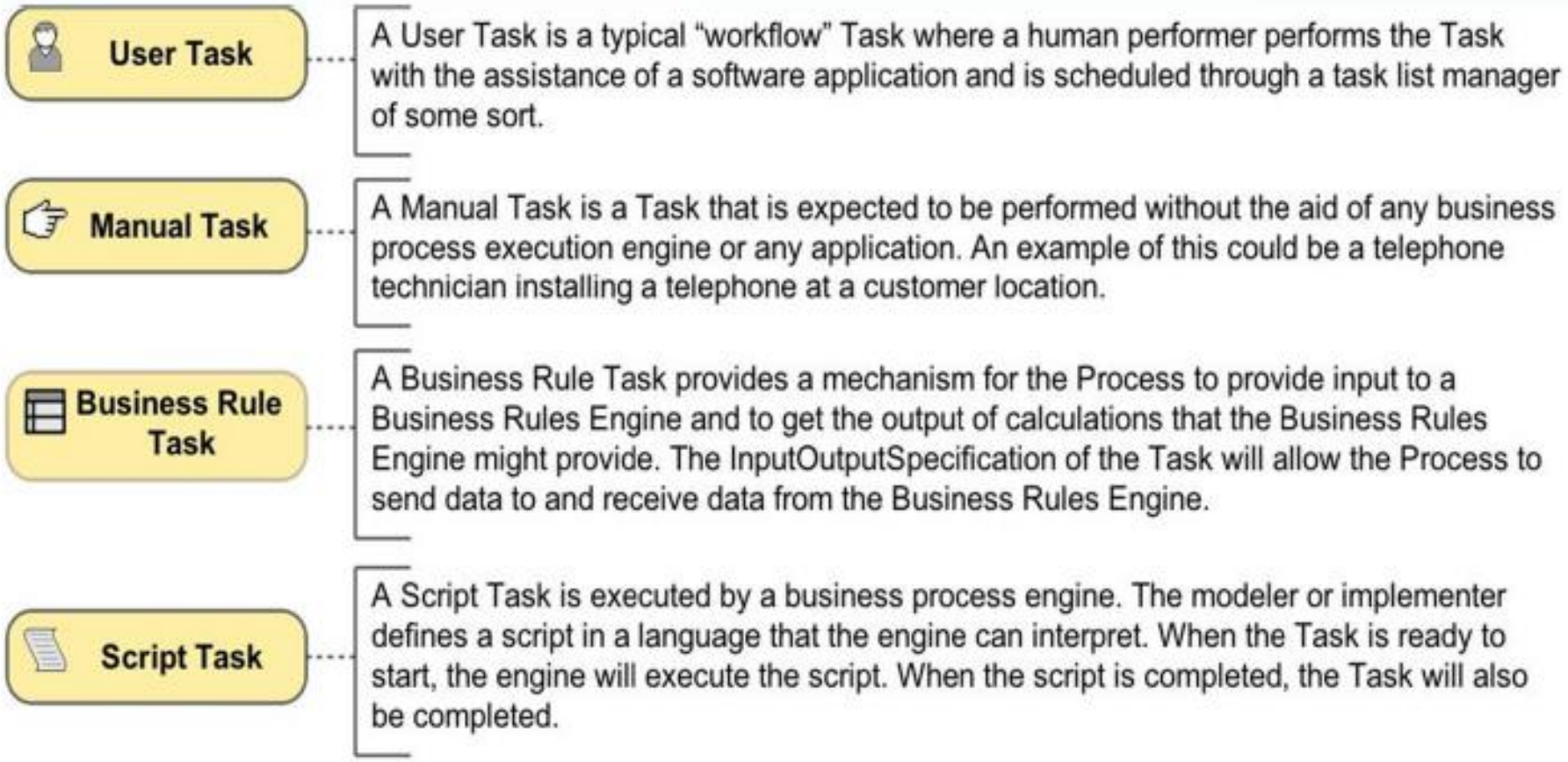
BPMN: Flow Objects

- **2.3. Task - The processes/activities that need to be mapped, probably for a future digitalization and computerization.**
- **Some additional icons can be used to specify the inner nature of the task**



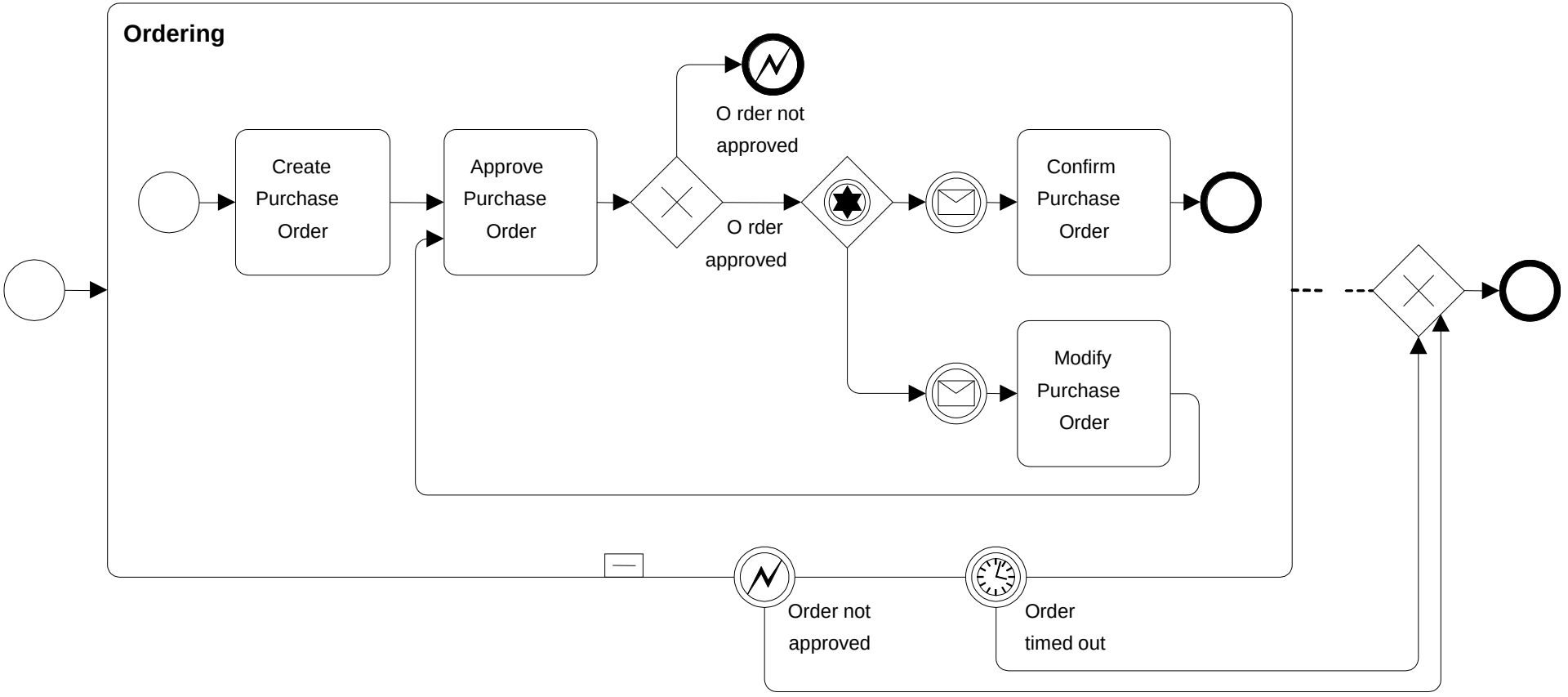
BPMN: Flow Objects

- **2.3. Task - The processes/activities that need to be mapped, probably for a future digitalization and computerization.**
- **Some additional icons can be used to specify the inner nature of the task**



Summing Up – A comprehensive example

An Ordering Process

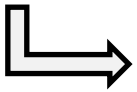


BPMN: Swim Lane

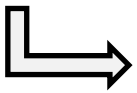
Sono necessari due tipi di astrazione

Resource
Risorse umane o attrezzature (ad esempio una stampante) che sono richieste per svolgere un'attività e/o un compito

Resource Class
Insieme di risorse omogenee che condividono caratteristiche e/o competenze comuni (ad esempio: impiegati, manager, progettisti, contabili, ecc.)



Ruolo
Ciò che la risorsa sa fare o che ci si attende sappia fare (in termini di abilità, competenza, qualifica)



Gruppo
Raggruppamento legato alla struttura organizzativa (dipartimento, team, ufficio, unità organizzativa)

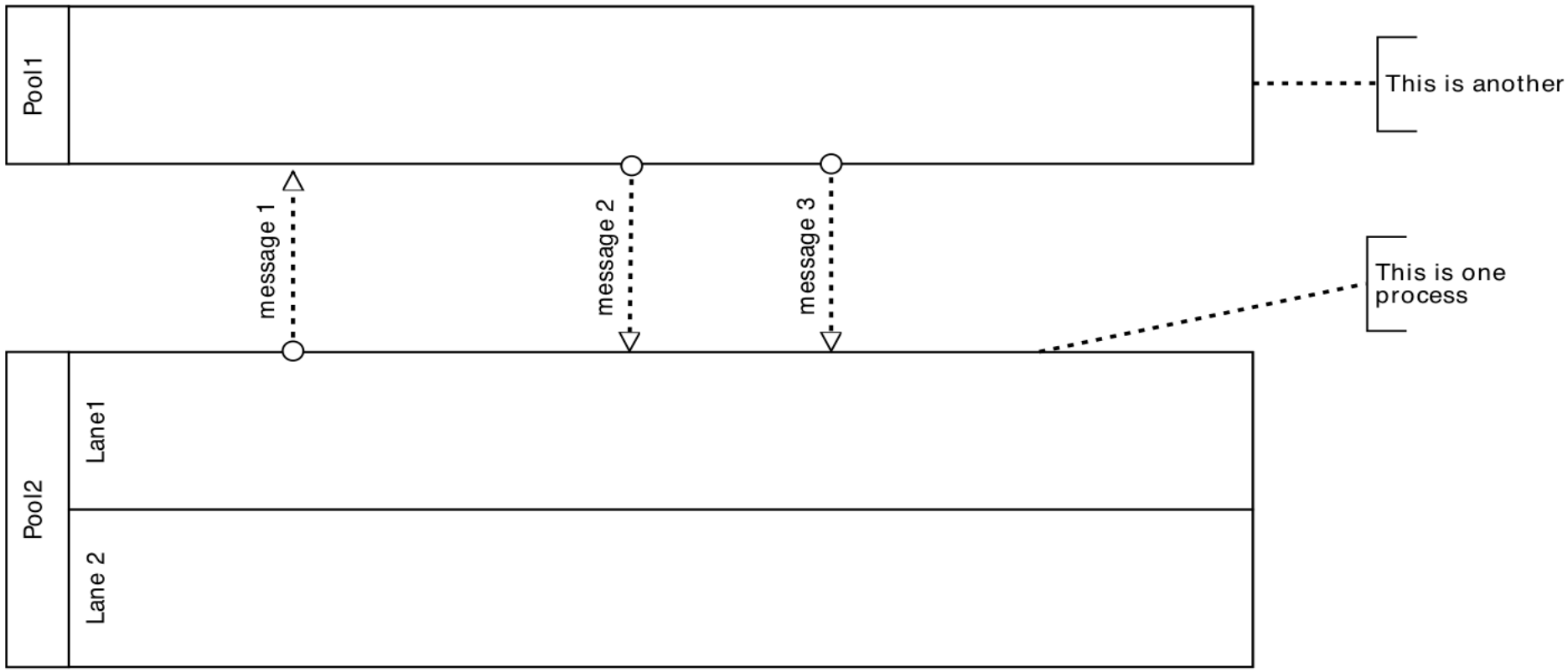
BPMN: Swim Lane

In BPMN, le Resource Class sono gestite usando:

Pools
entità indipendenti dall'organizzazione (ad esempio clienti e fornitori)

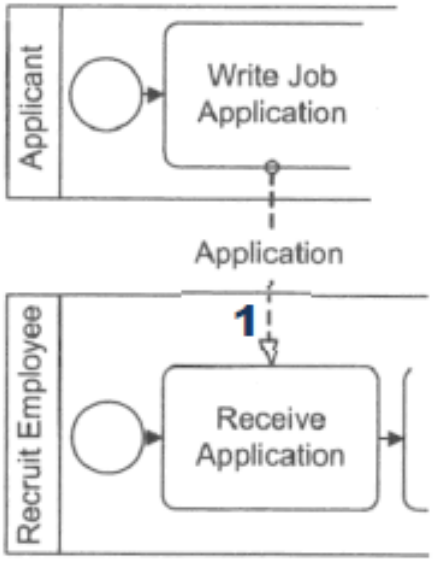
Lane
Resource class appartenenti alla stessa organizzazione che condividono spazi comuni (uffici) e/o le stesse attrezzature e sistemi informativi

BPMN: Swim Lane

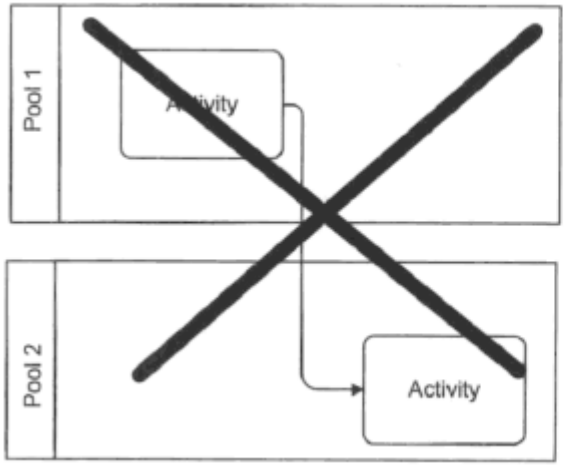


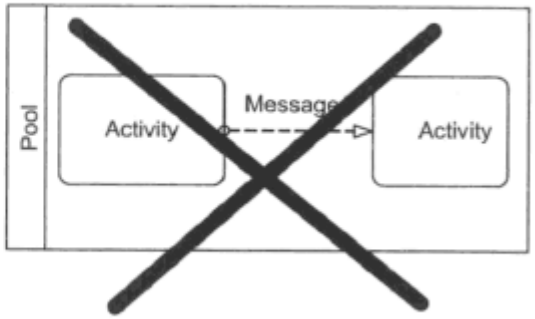
Fra i pool possono esserci solo flussi informativi

BPMN: Swim Lane



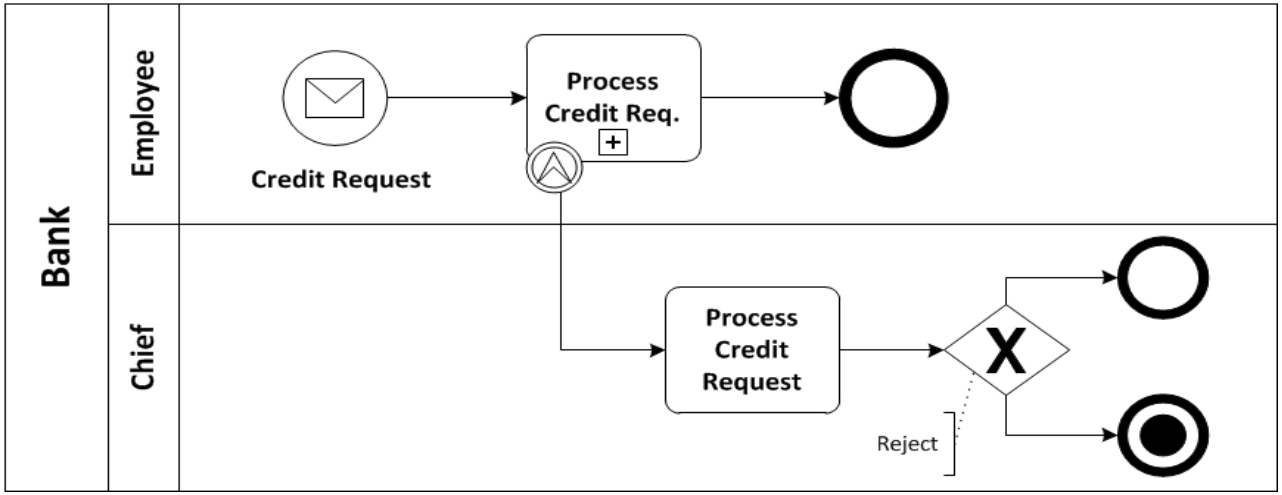
✓*Ammesso*



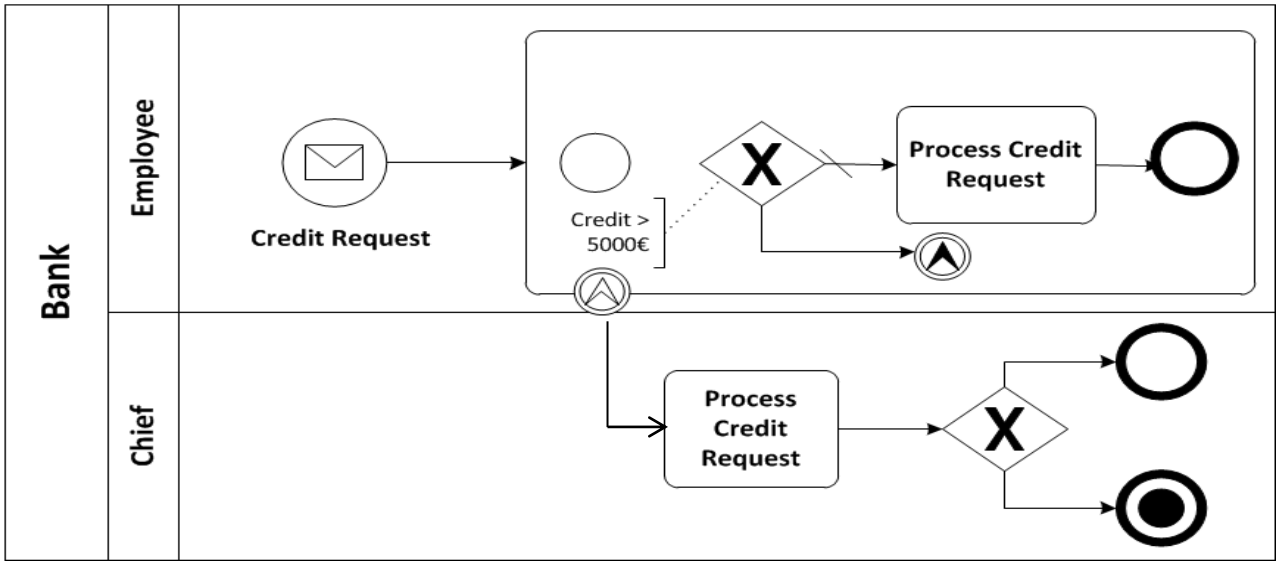


✓*Non Ammesso*

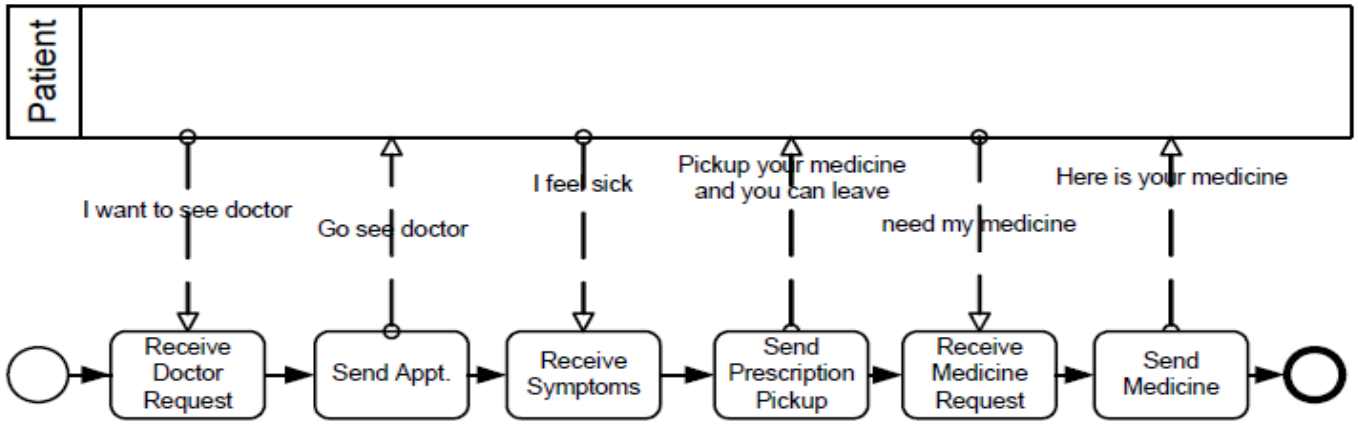
BPMN: Flow Objects



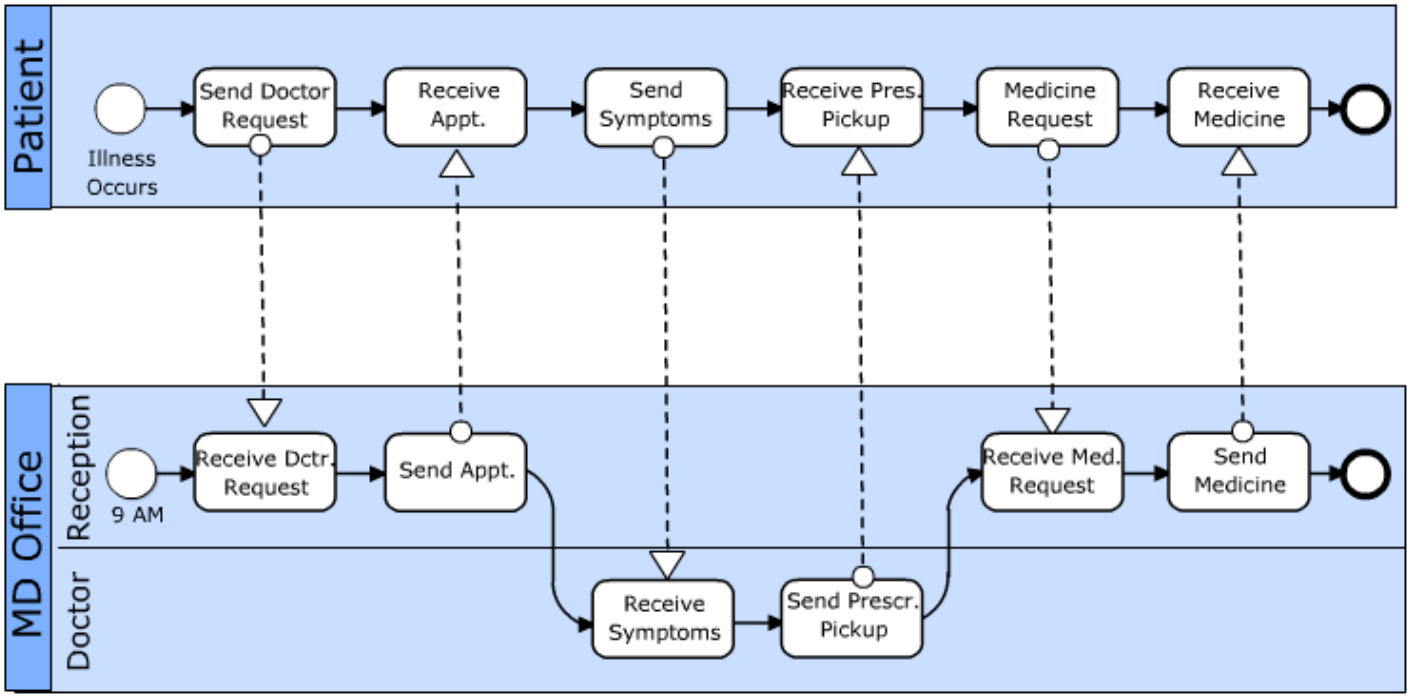
Exploded Process



BPMN: Swim Lane

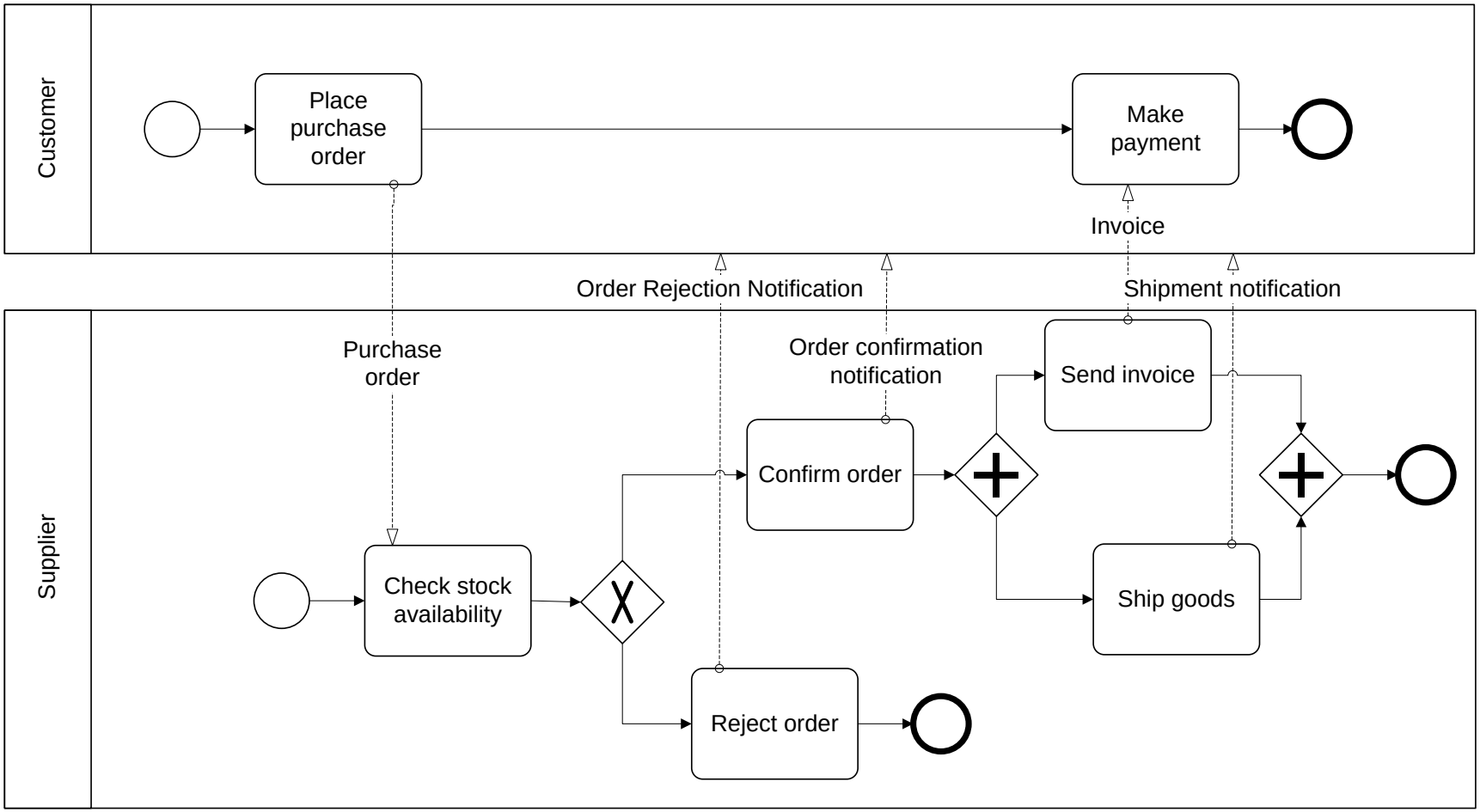


Public Process



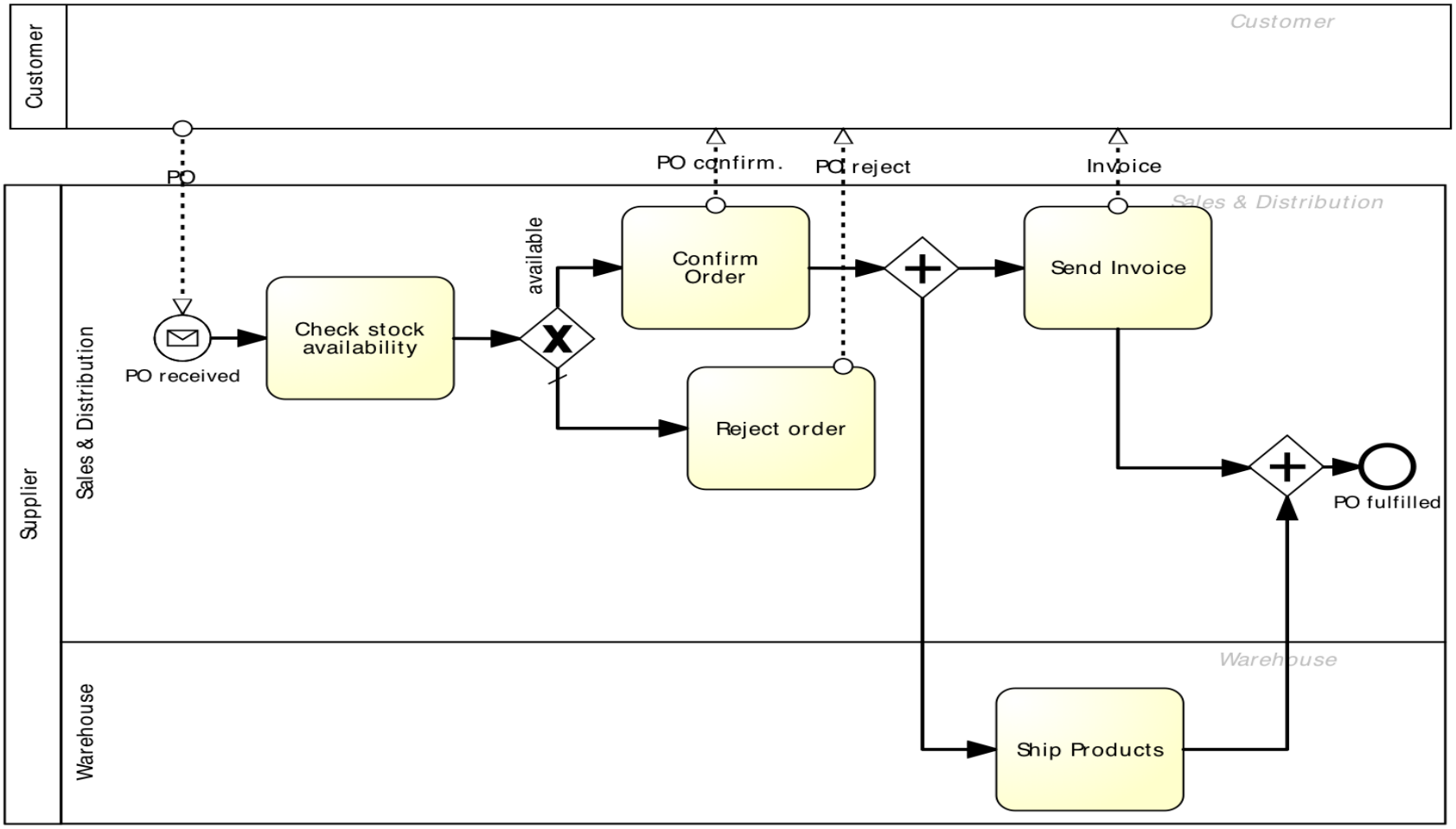
Collaborative Process

Pools



In this case two swim lanes are used to show which actions are performed by the customer and by the supplier, respectively
Note that only informative flows are allowed between swim lanes

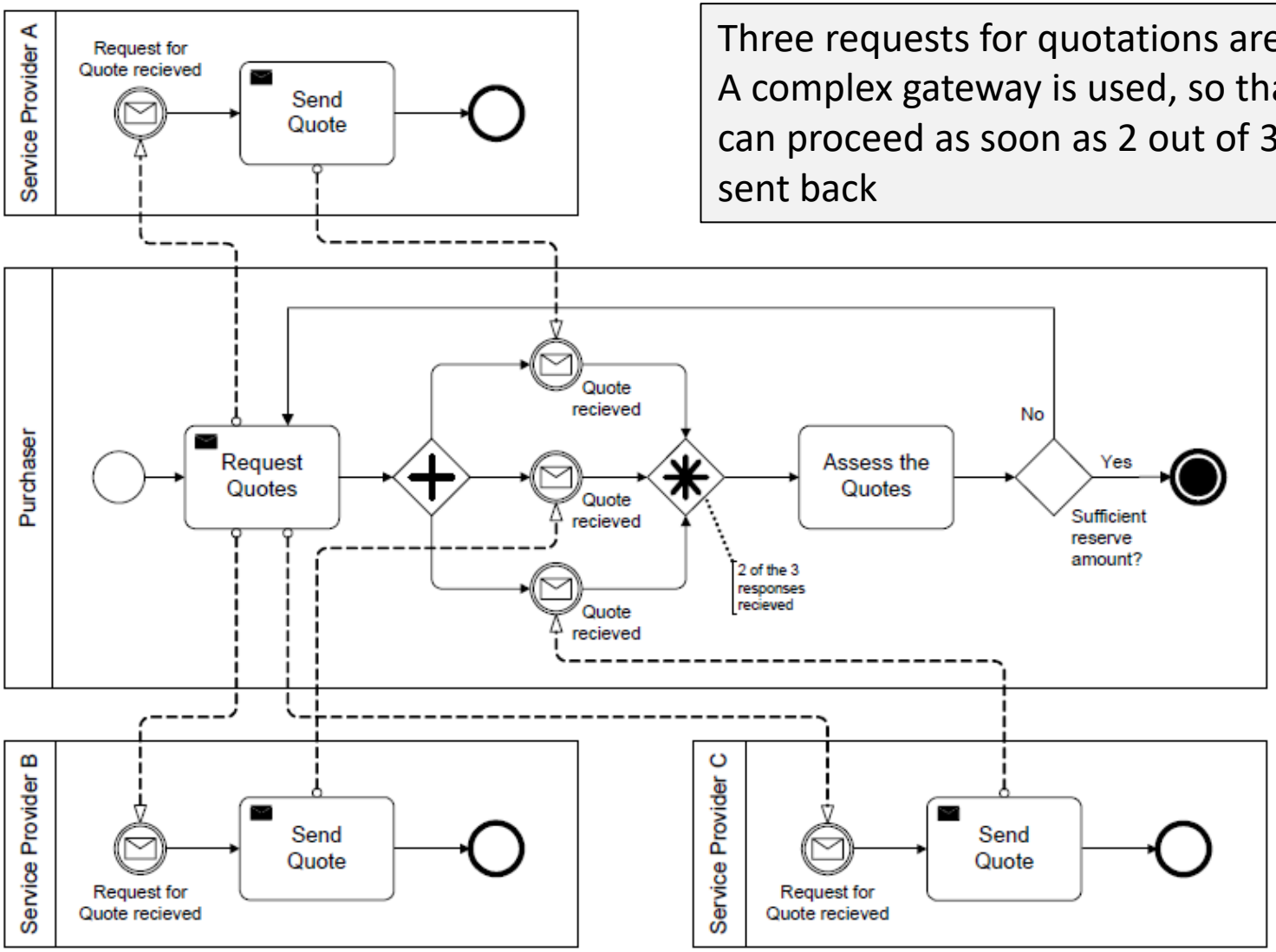
Lanes



Sometimes it is sufficient to indicate an actor of the process, without drawing all the task that he or she performs

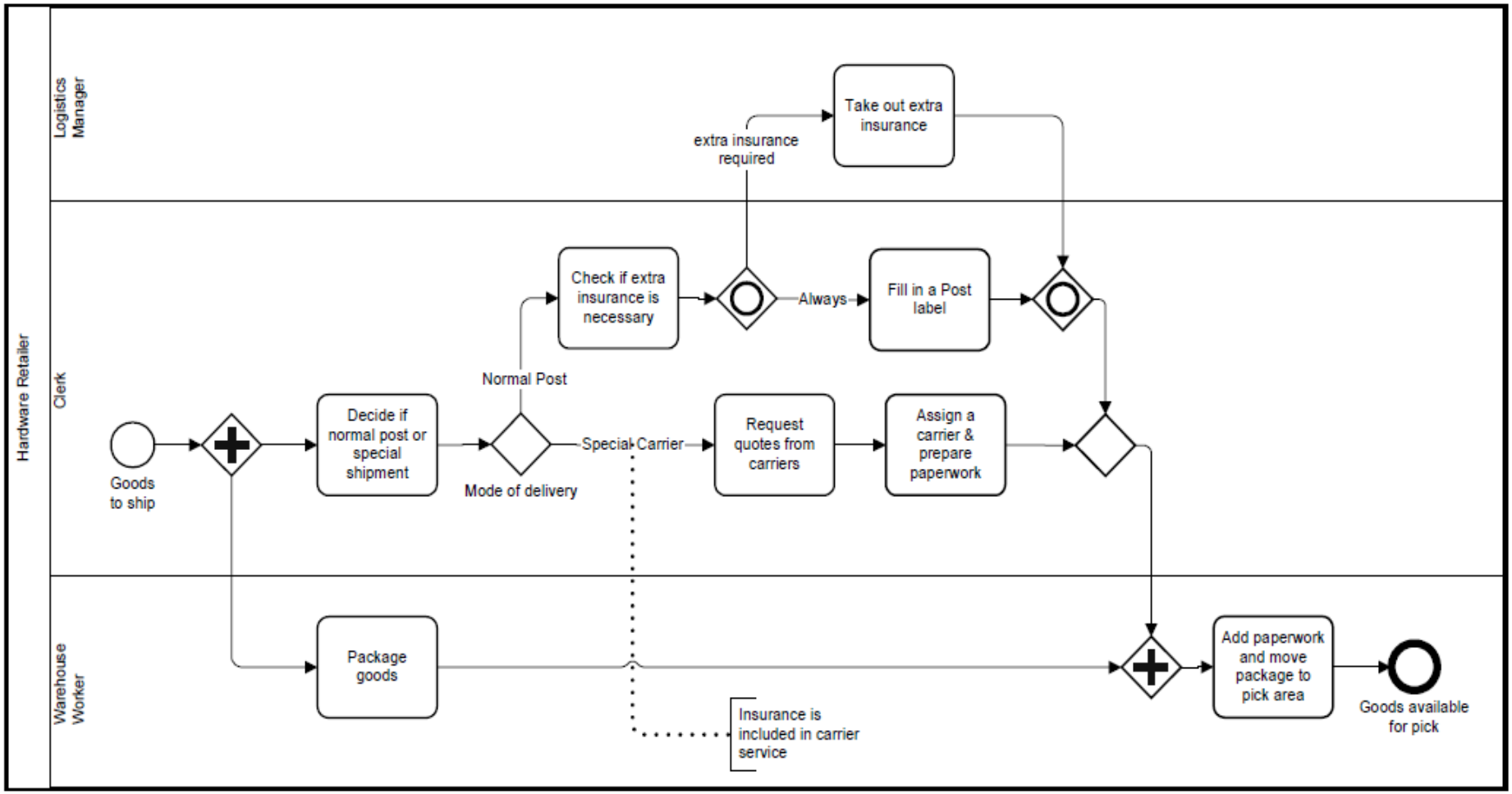
In this case the customer is indicated, but his own process is not mapped

Pools

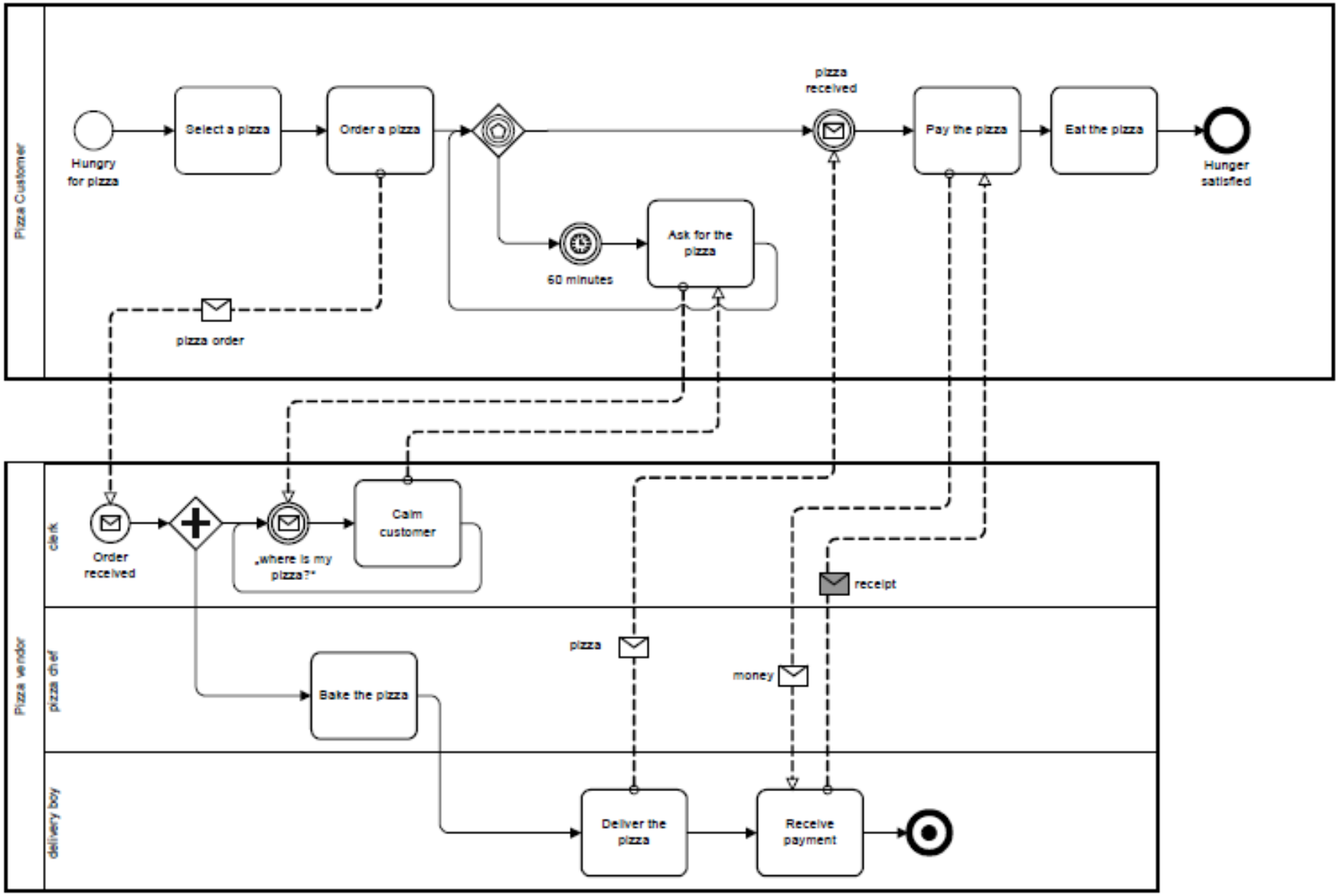


Three requests for quotations are sent.
A complex gateway is used, so that the process can proceed as soon as 2 out of 3 quotations are sent back

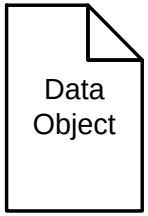
Shipment Process of a Hardware Retailer



«Pizza Collaboration»

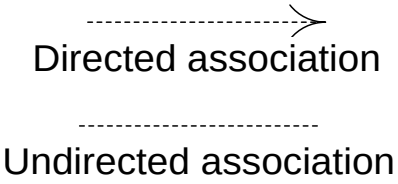


BPMN: Artefatti



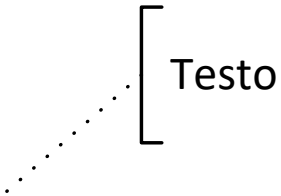
Oggetto Dati – mostra i dati necessari e/o prodotti da un'attività

Contenitore di Dati – sono archivi permanenti necessari a mantenere dati e informazioni generati dal processo



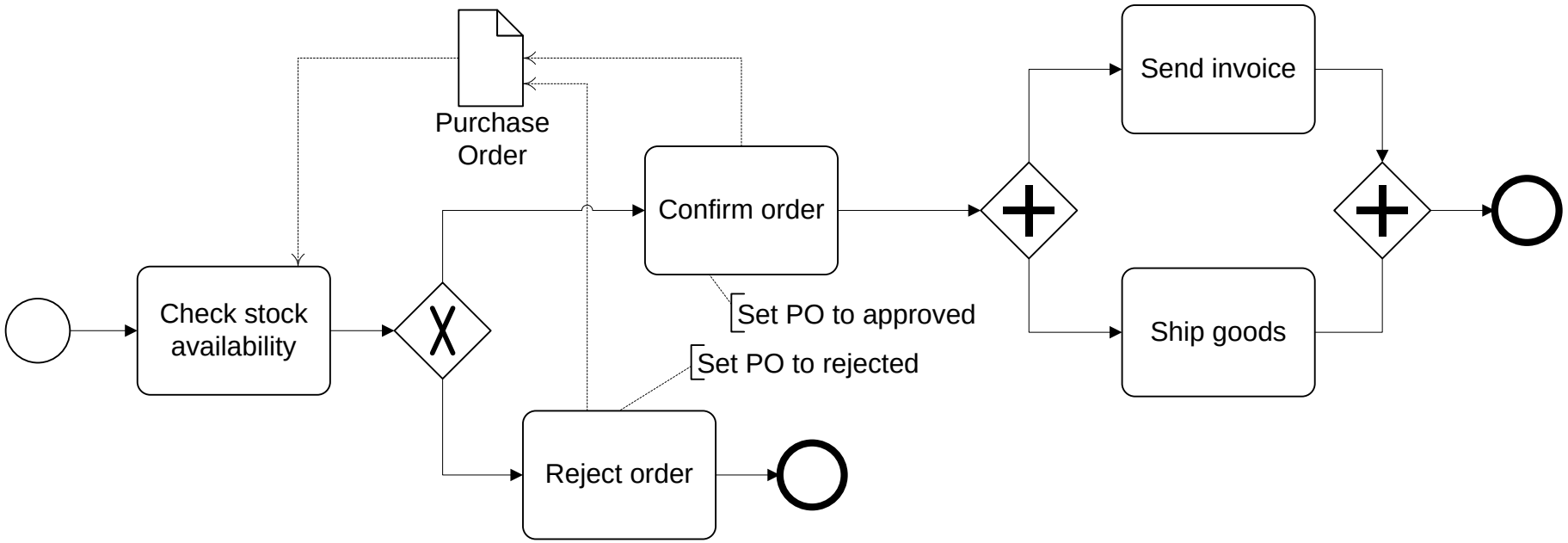
Associazioni – sono usati per collegare gli artefatti (come dati e contenitori) con i *flow object*

Gruppo – elementi che svolgono parti affini e/o temporalmente vicine



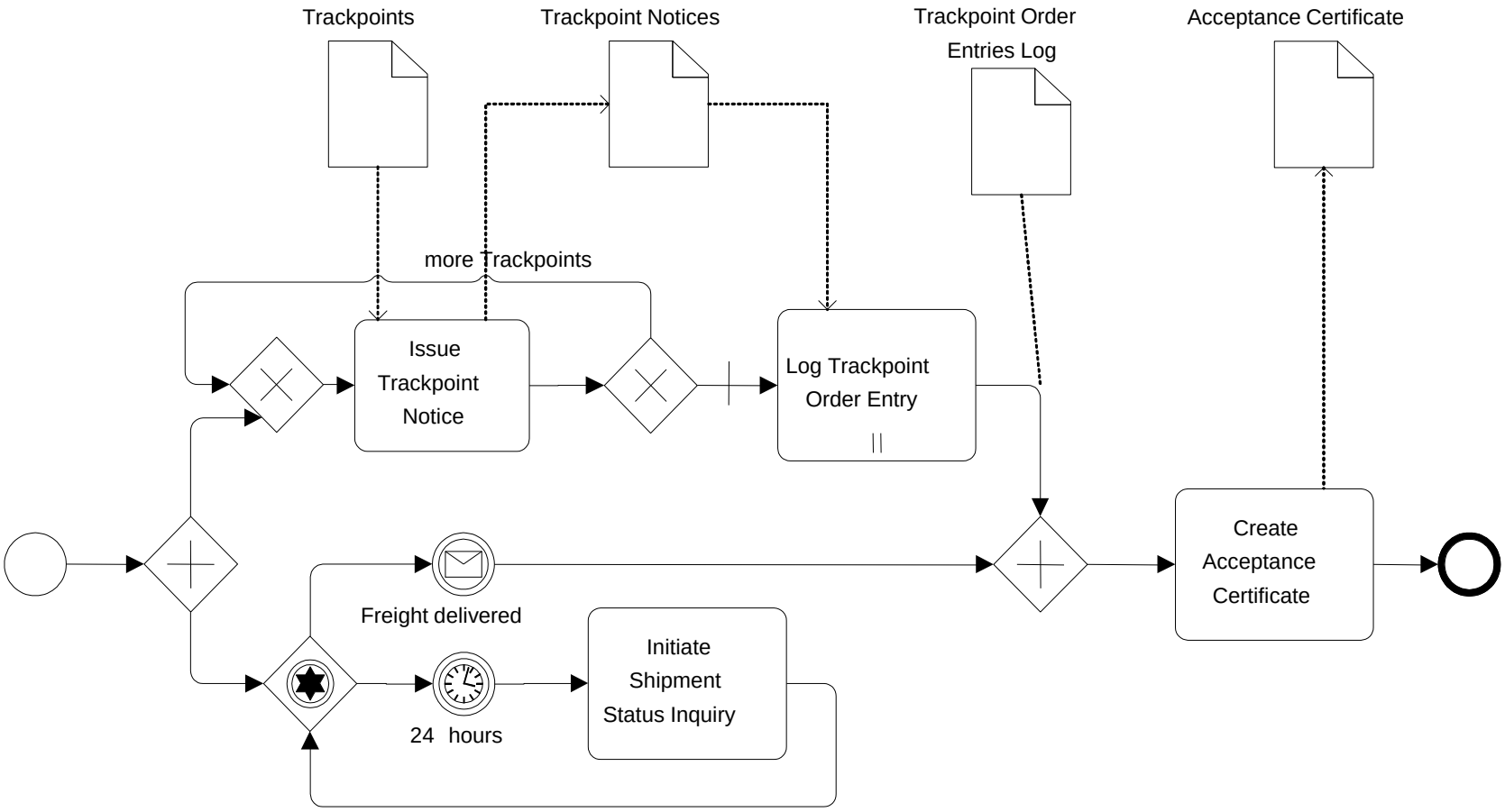
Annotazioni – brevi spiegazioni testuali

BPMN: Artefacts



Artefacts are used to improve the readability of the model

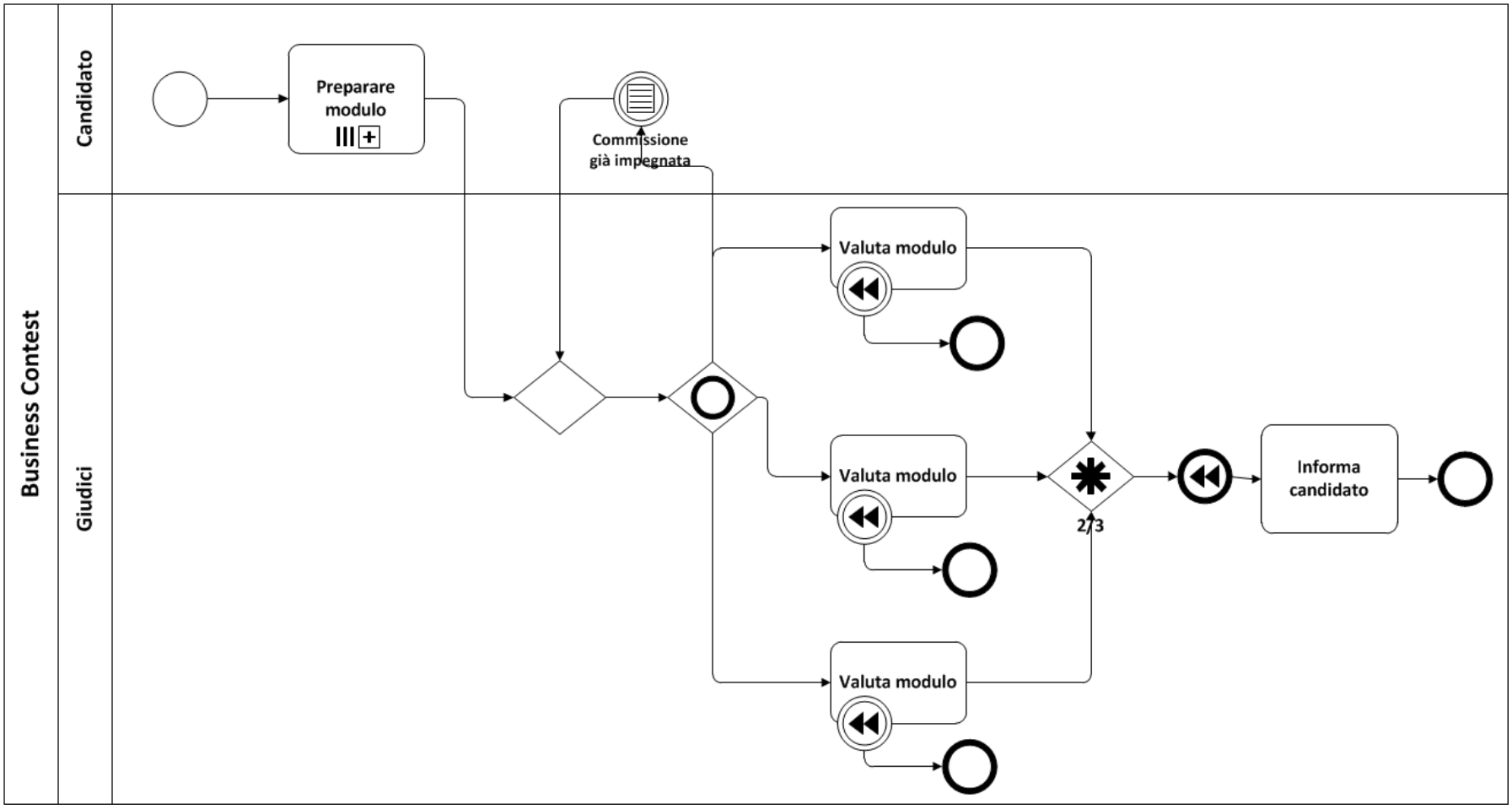
BPMN: Artefacts



Workflow patterns - Advanced Branching & Synchro

Cancelling Partial Join

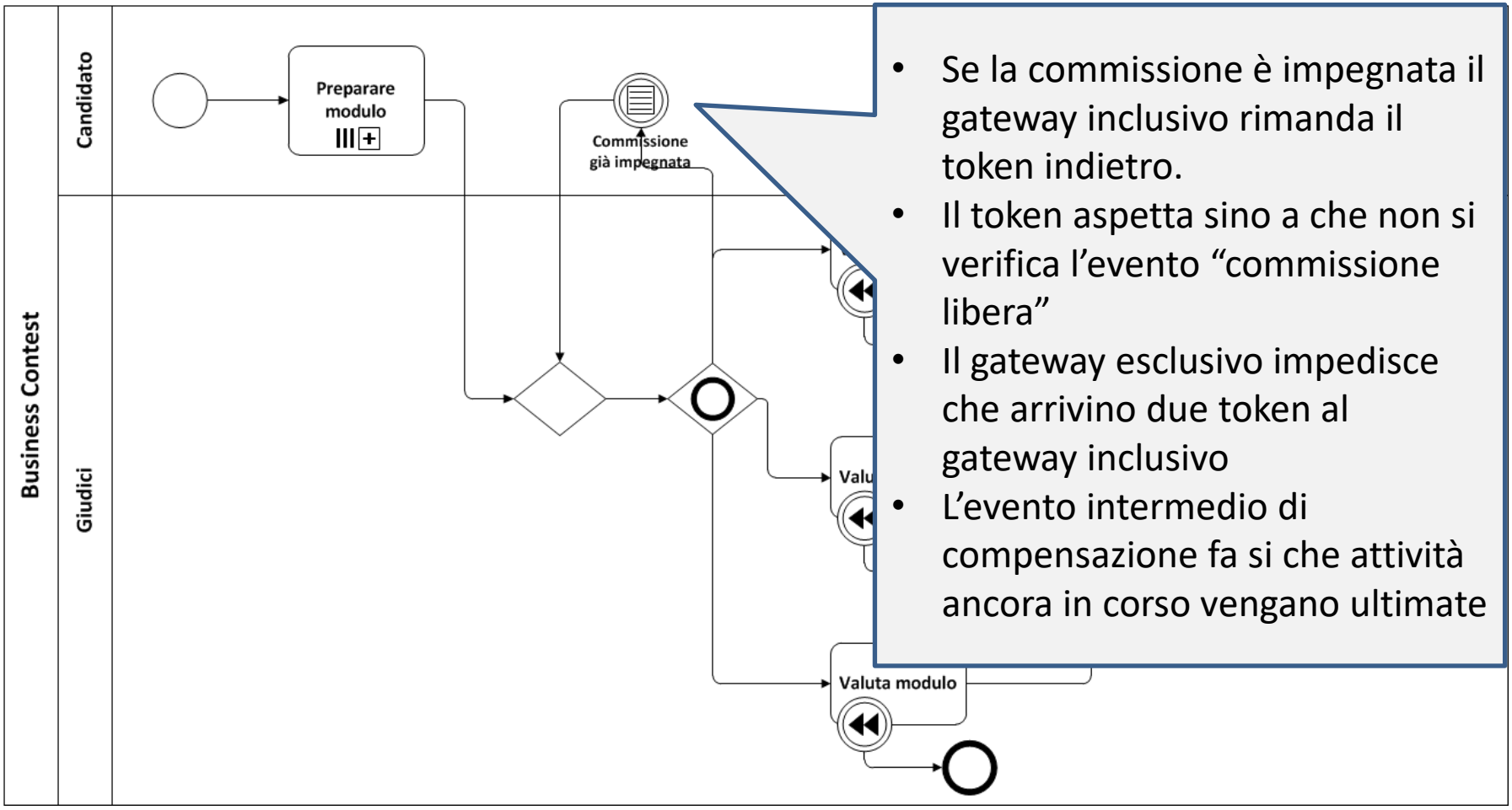
Si procede se almeno *m* di *n* rami sono stati percorsi e si bloccano i restanti *(n – m)* percorsi



Workflow patterns - Advanced Branching & Synchro

Cancelling Partial Join

Si procede se almeno *m* di *n* rami sono stati percorsi e si bloccano i restanti *(n – m)* percorsi

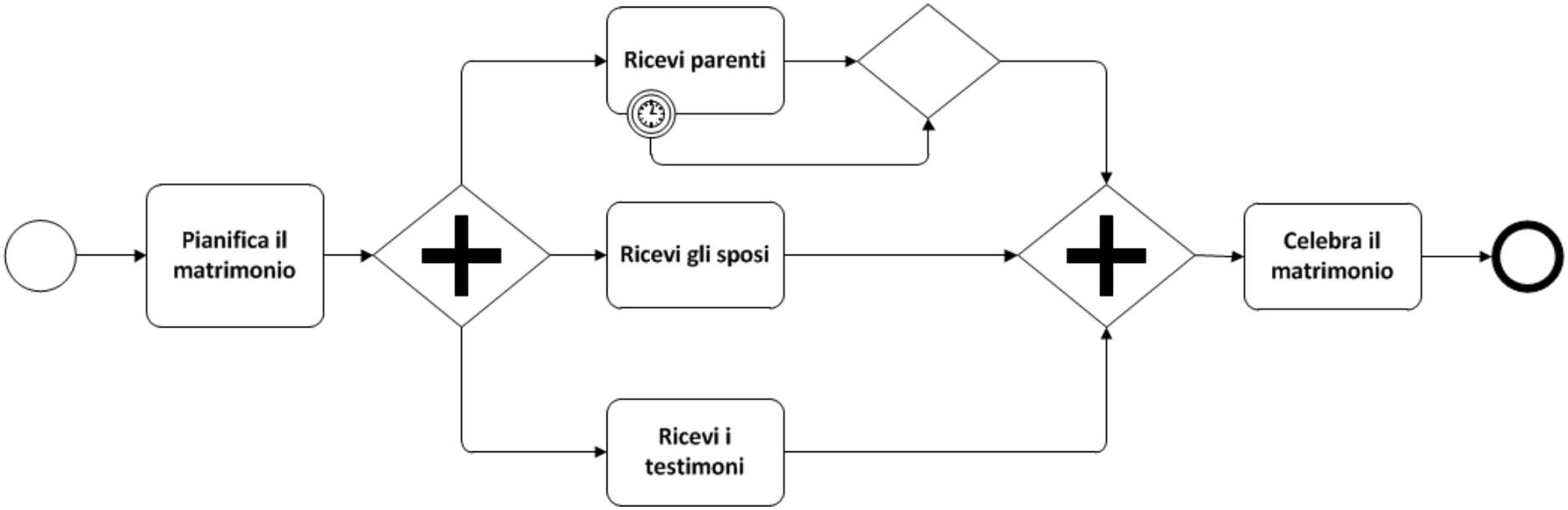


- Se la commissione è impegnata il gateway inclusivo rimanda il token indietro.
- Il token aspetta sino a che non si verifica l'evento "commissione libera"
- Il gateway esclusivo impedisce che arrivino due token al gateway inclusivo
- L'evento intermedio di compensazione fa si che attività ancora in corso vengano ultimate

Workflow patterns - Advanced Branching & Synchro

General Synchronizing Merge

Per procedere è necessario che tutti gli *n* rami siano stati percorsi; alcuni rami possono essere trascurati in seguito al verificarsi di specifiche condizioni

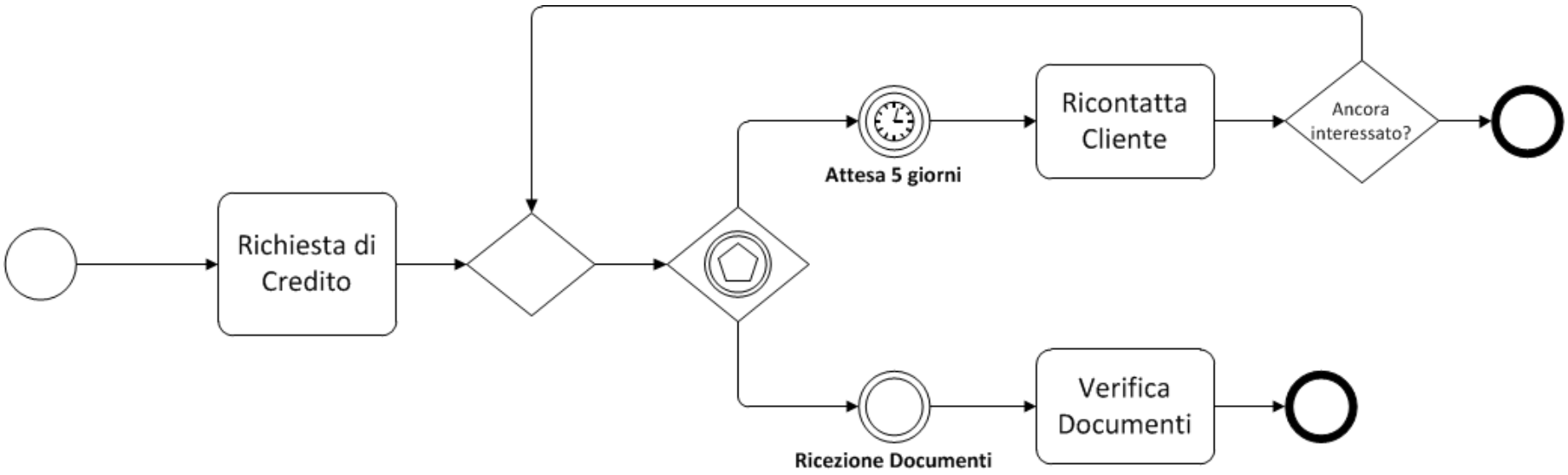


- Se i parenti tardano, si procede comunque con la celebrazione
- Ovviamente non è possibile procedere se mancano gli sposi e I testimoni

Workflow patterns – State Based Patterns

Deferred Choice

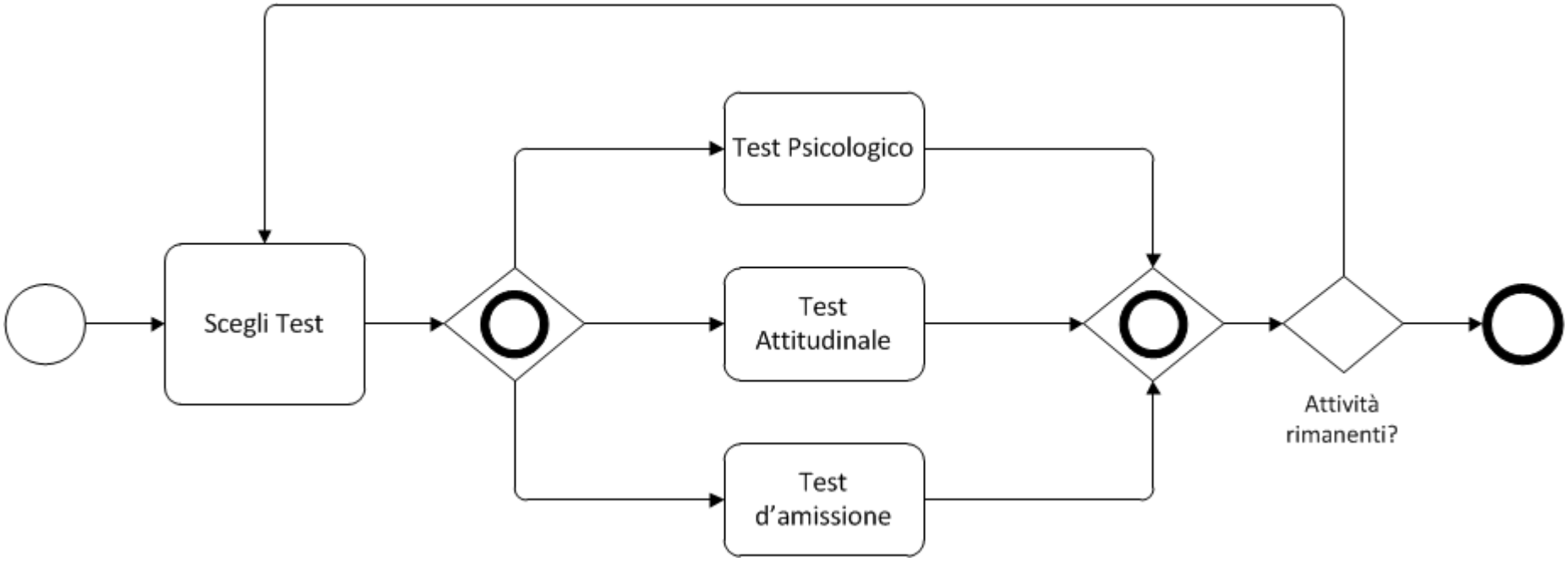
La decisione (circa il ramo da seguire) dipende non dal flusso di dati del processo stesso, ma, piuttosto, dall'interazione del processo con l'ambiente esterno.
Quando un ramo del processo è abilitato, l'altro dovrebbe essere disabilitato



Workflow patterns – State Based Patterns

Interleaved Routing

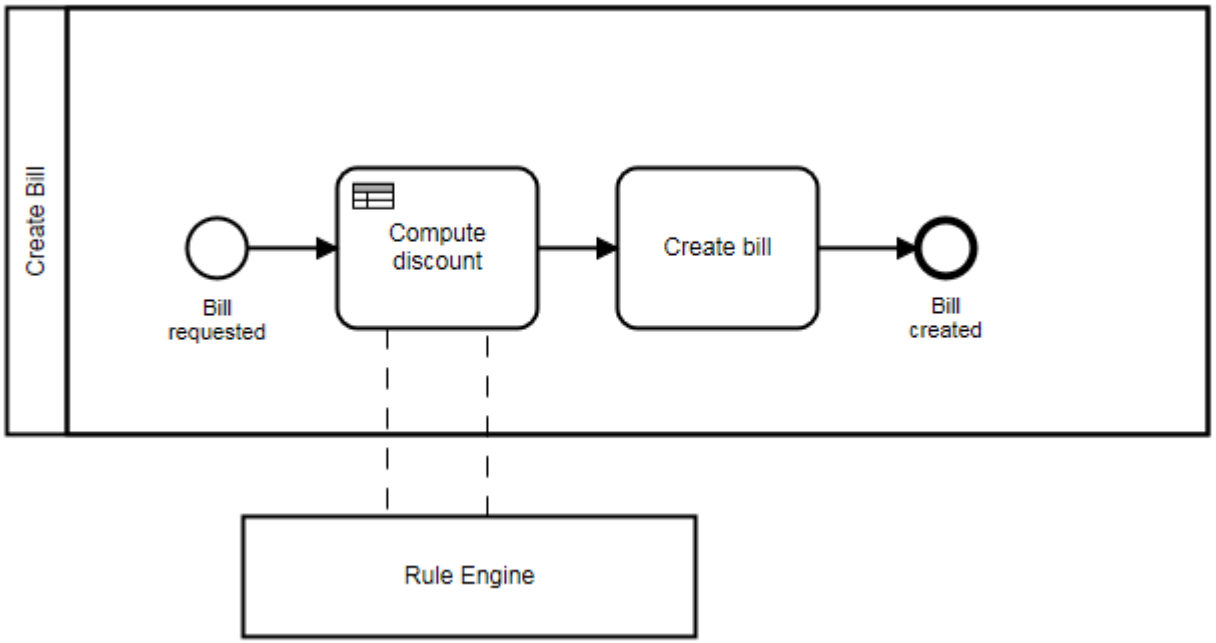
Tutti i rami devono essere percorsi, uno alla volta e una sola volta



Supponiamo ad esempio che alla prima iterazione venga scelto il test attitudinale.
Dato che ci sono ancora due attività rimanenti il token torna indietro
A questo punto si hanno solo due scelte “test psicologico” o “test d’ammissione”
Il processo ricomincia come prima

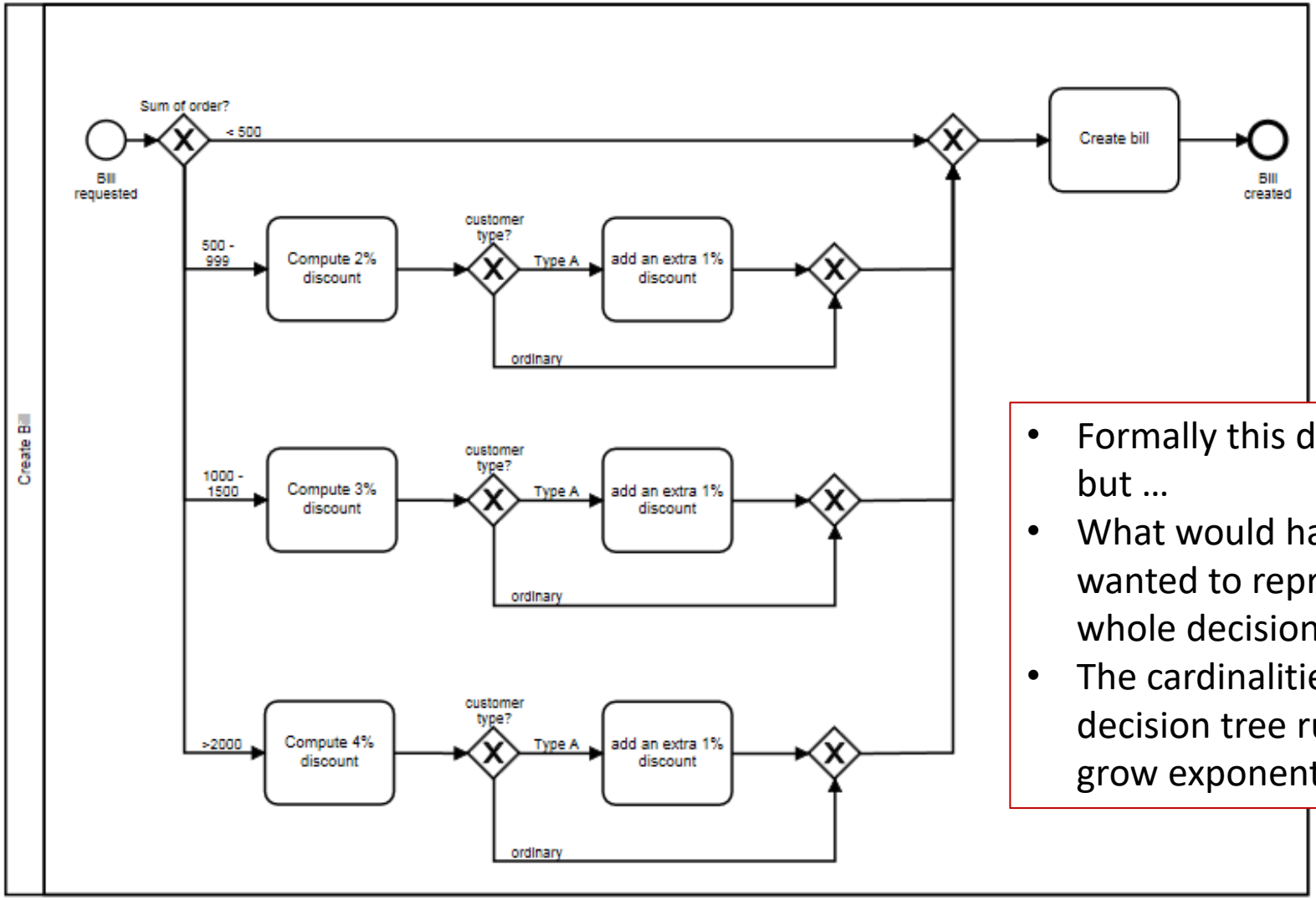
Examples

- **Creating a bill**
  - *A discount needs to be computed.*
  - *The sum of the order and the customer type are the relevant criteria to compute the discount*



- We **focus on the two steps process flow** i.e., a discount computed before the bill is created.
- It **does not make sense to model the calculation** of the discount itself in the BPMN model
- The cardinalities of the decision tree rules will grow exponentially!!!

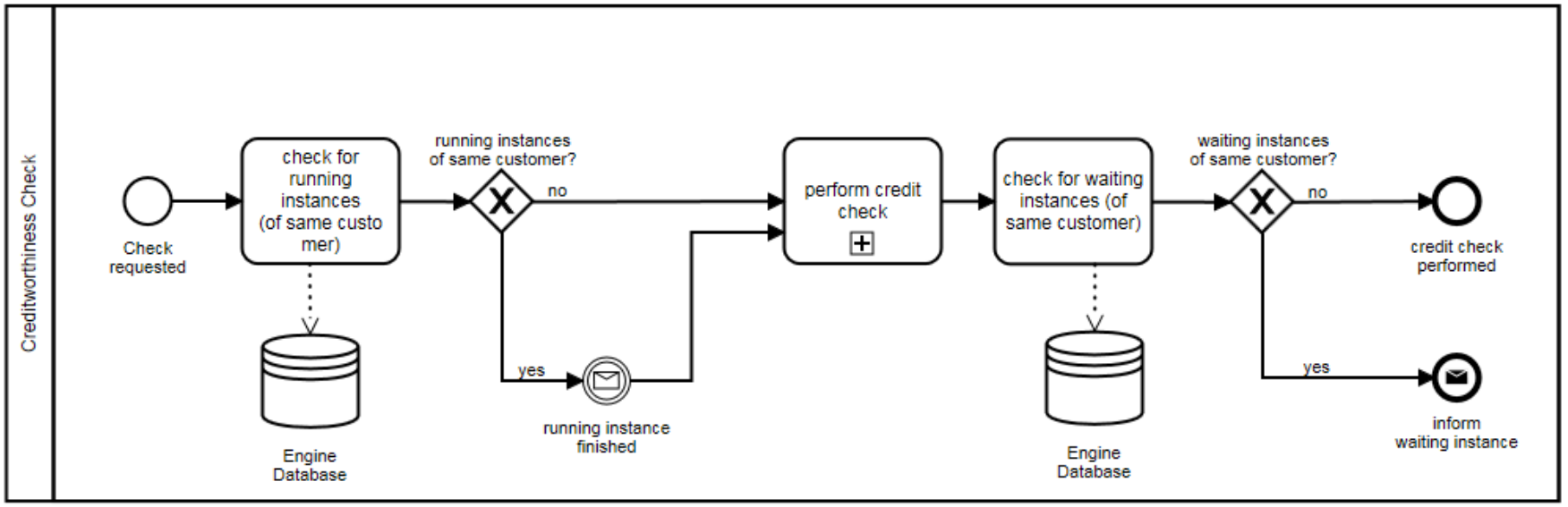
Creating a Bill - The wrong way …



- Formally this diagram is ok but …
- What would happen if one wanted to represent the whole decision tree?
- The cardinalities of the decision tree rules will grow exponentially!!!

Examples

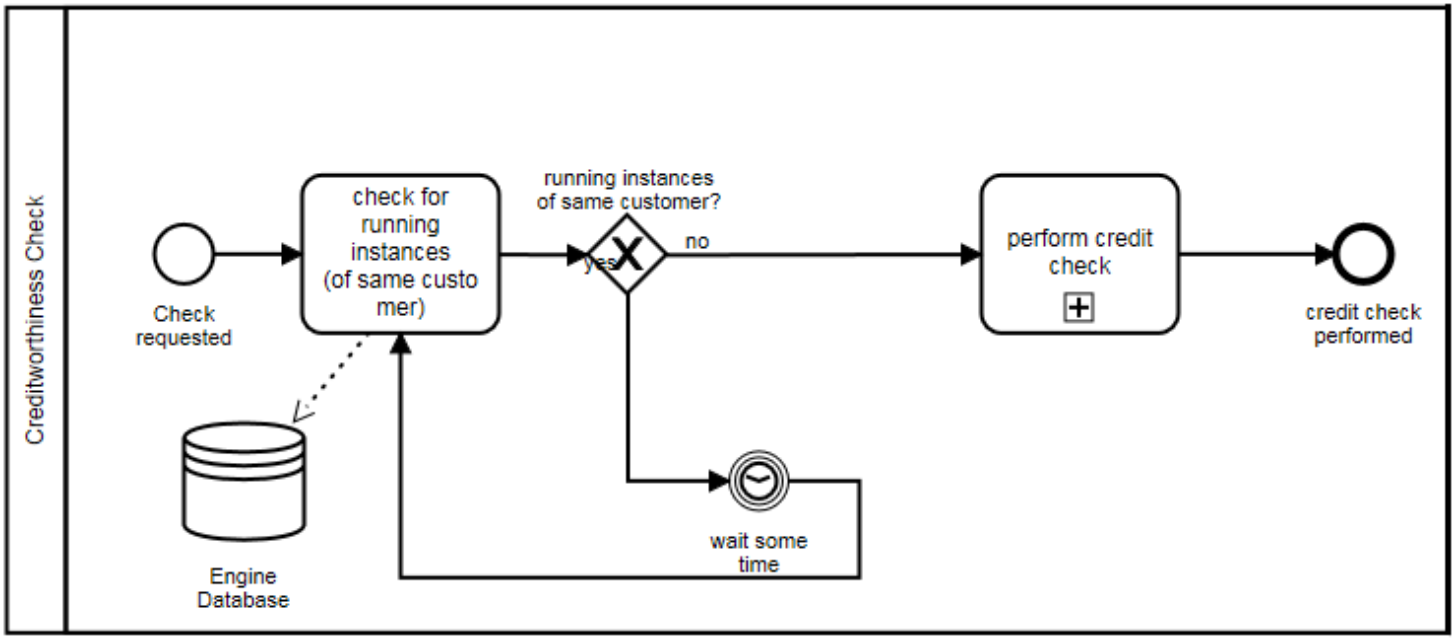
- **Concurrent instances**
  - *Credit Cards checking*
  - *If one check is running, another one cannot be performed at the same time*
  - *We make the hypothesis that all requests come from the same customer*



Please note that, to determine the recipient of the message, a second data request before the end of the instance is needed

Examples

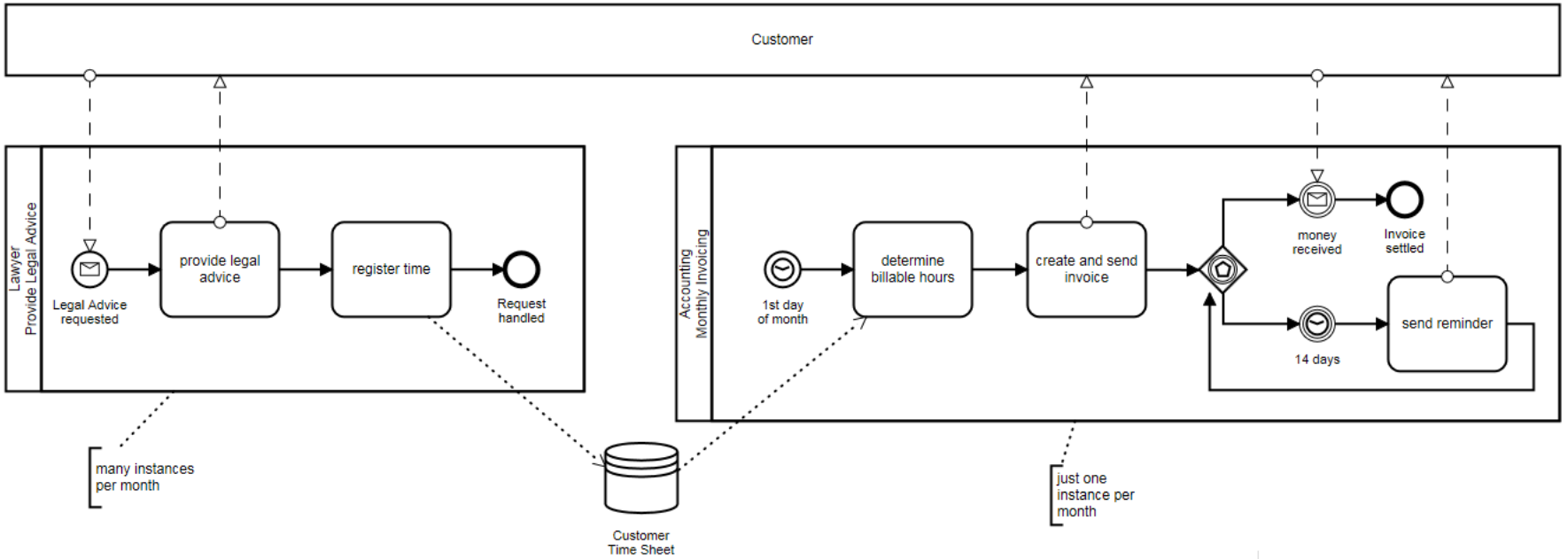
- **Concurrent instances**
  - *Credit Cards checking*
  - *If one check is running, another one cannot be performed at the same time*
  - *We make the hypothesis that all requests come from the same customer*



A different approach based on a delay

Examples

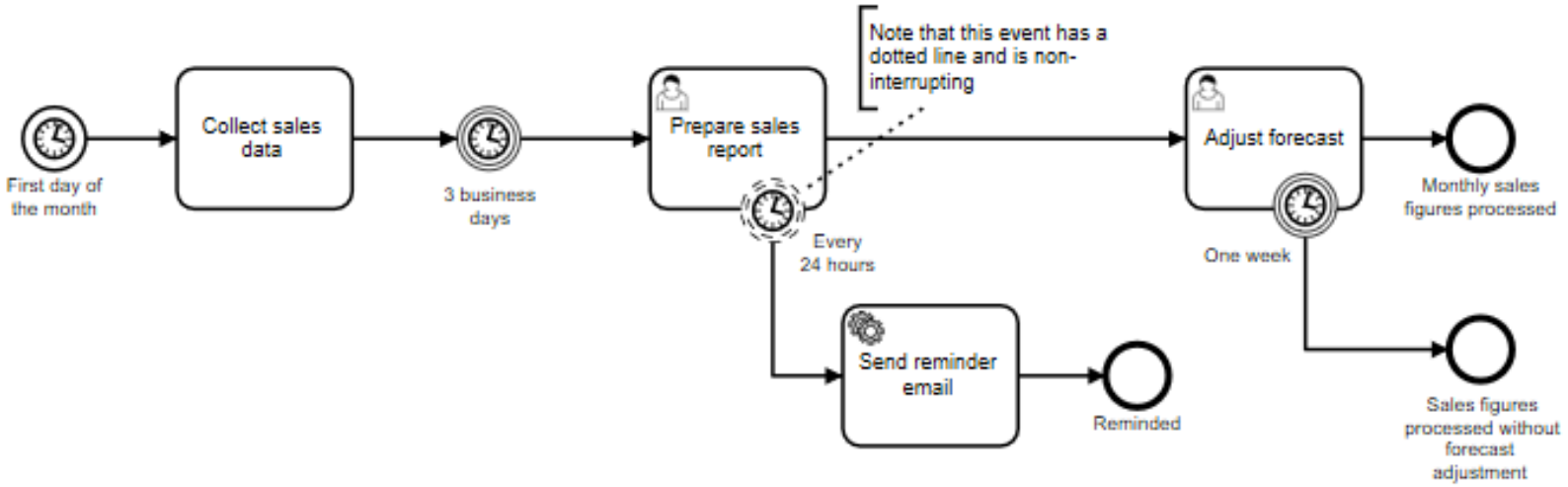
- **Monthly invoicing**
  - A lawyer offers legal advice to his customers
  - Customers can ask for legal advice whenever they need it
  - The lawyer provides the requested advice and puts the billable hours on the customer's time sheet.
  - When the month is over, the lawyer's accountant determines the billable hours based on the time sheet and creates the invoice.



Examples

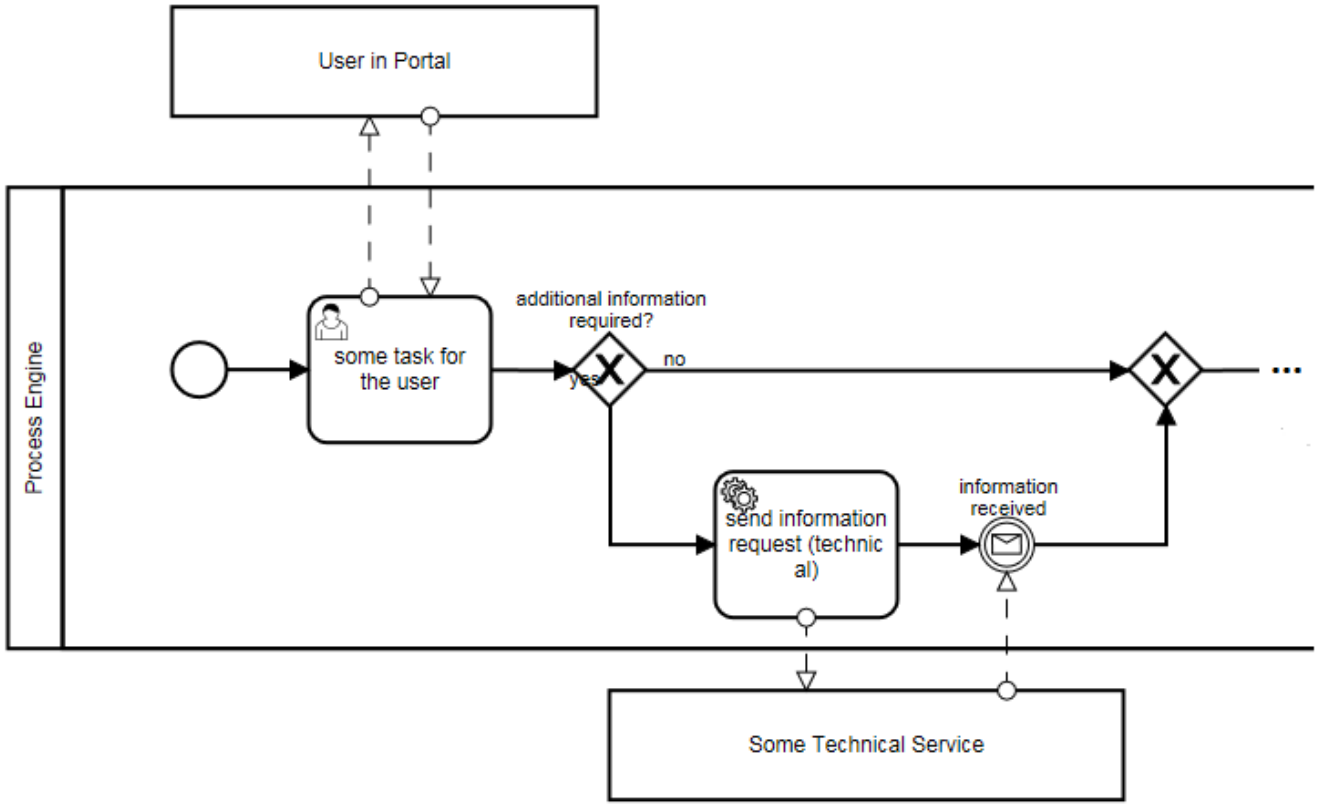
- **Timer Events**

  Timer events are events which are triggered by a defined timer. They can be used as start event, intermediate event or boundary event. Boundary events can be interrupting or not



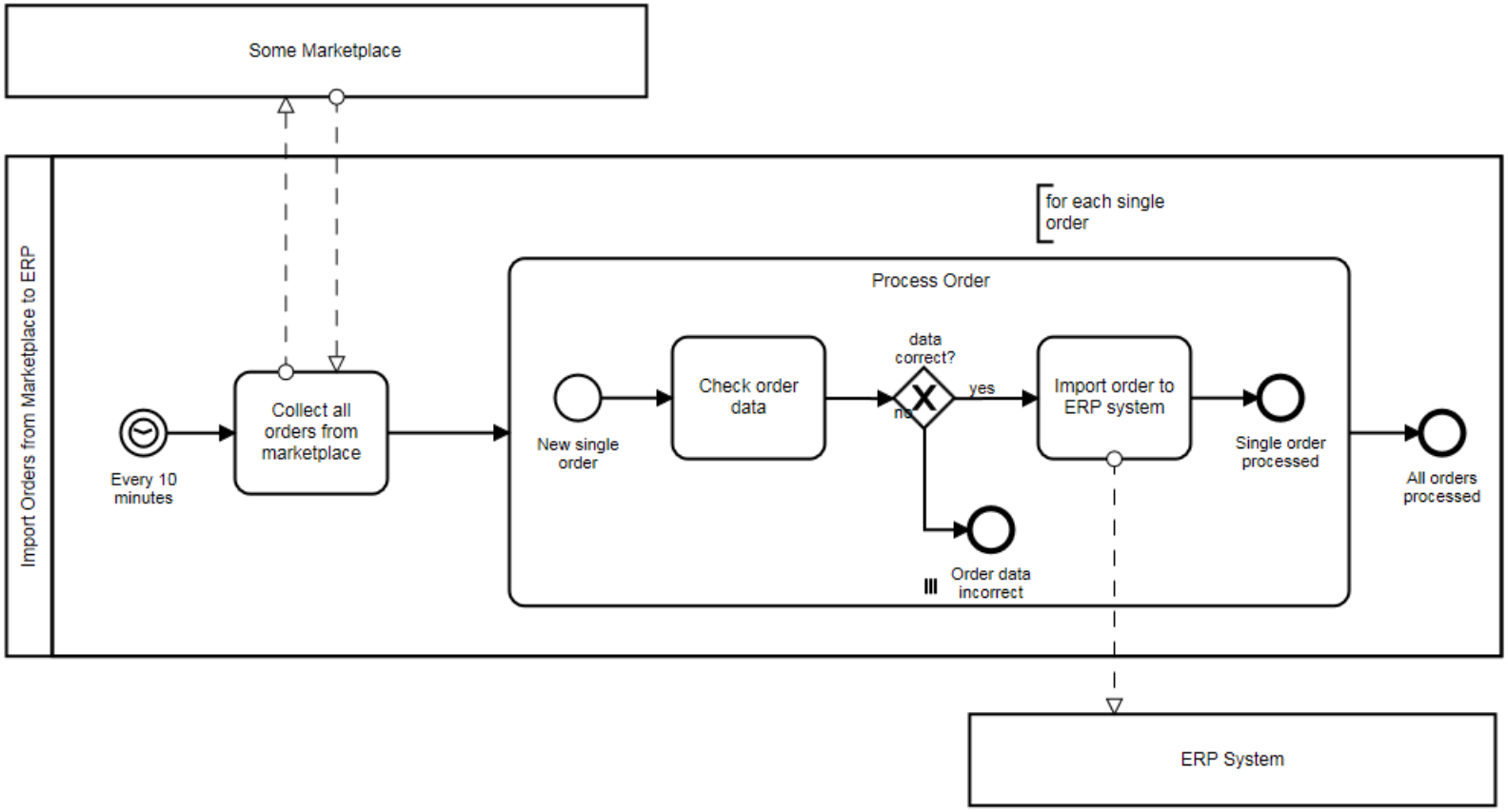
Examples

- **User Task on a Portal**
  - A task is performed by a user in a portal
  - At the end additional information might be required
  - If that is the case, the process engine sends an information request to a technical service



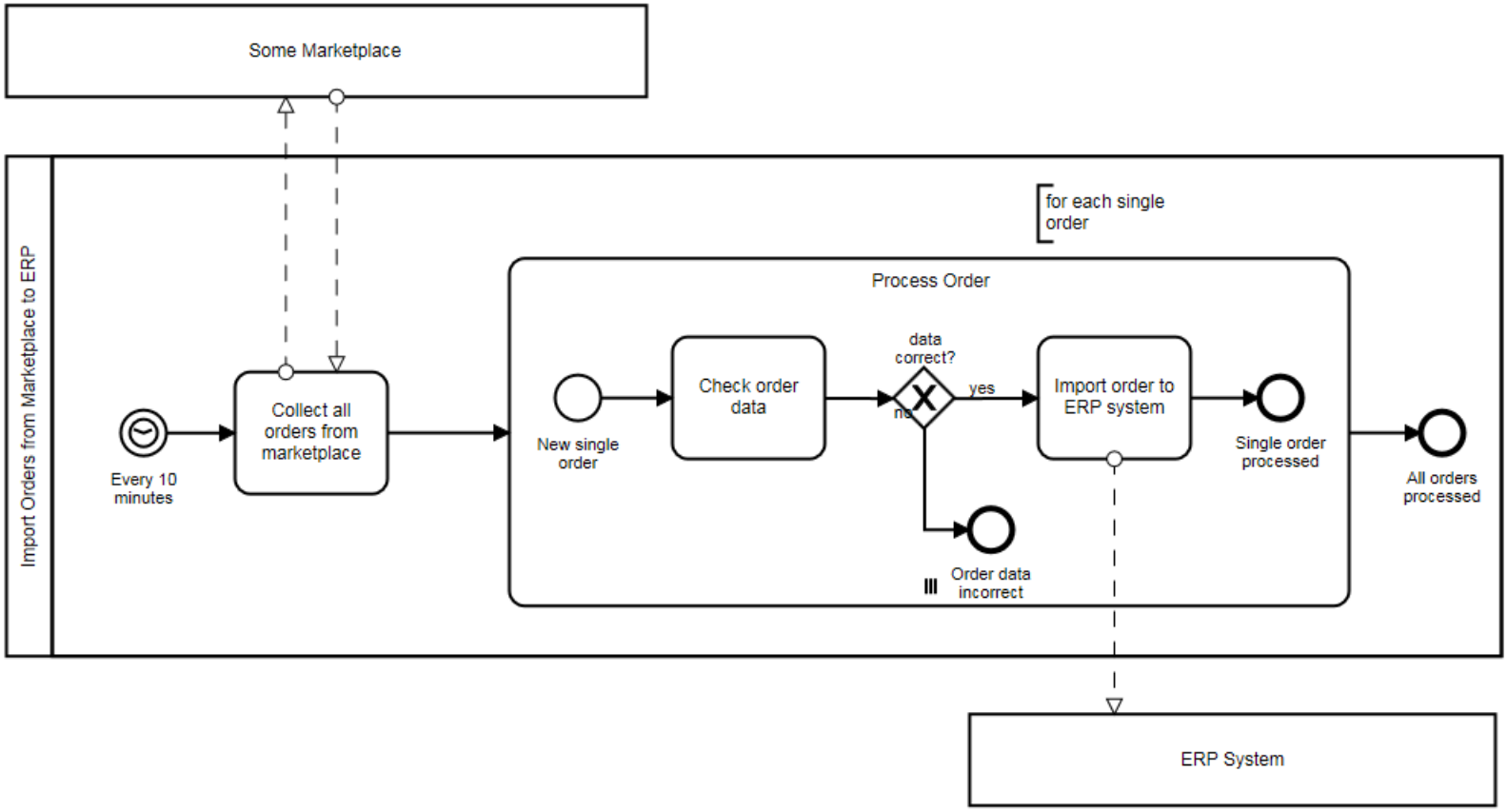
Examples

- **1 To N problem**
  - A company receives orders also from a digital marketplace
  - Orders from the marketplace are periodically fetched as a batch
  - Every order in a batch needs to be validated and, next, loaded on the ERP



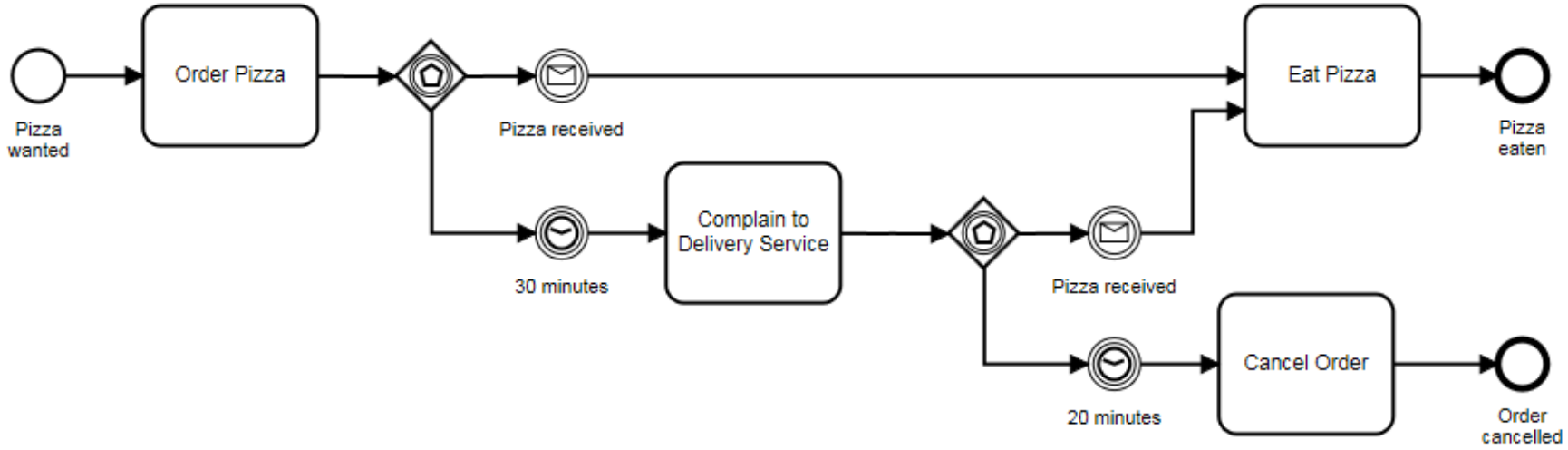
Examples

- **1 To N problem**
  - A company receives orders also from a digital marketplace
  - Orders from the marketplace are periodically fetched as a batch
  - Every order in a batch needs to be validated and, next, loaded on the ERP



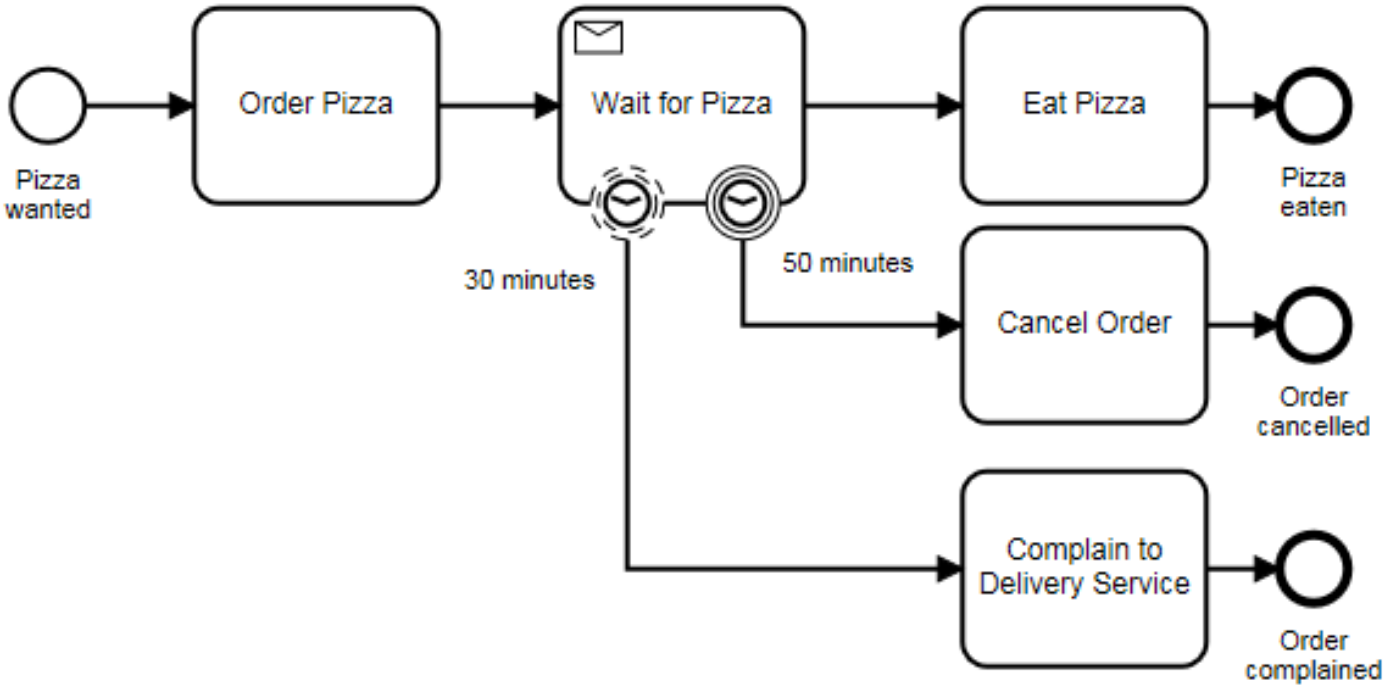
Examples

- **2 Steps Escalation – The Pizza problem**
  - We order a pizza
  - If delivery takes longer than 20 minutes we complain to the delivery service
  - We allow them another 30 minutes
  - If they do not make it in time, we give up and cancel our order



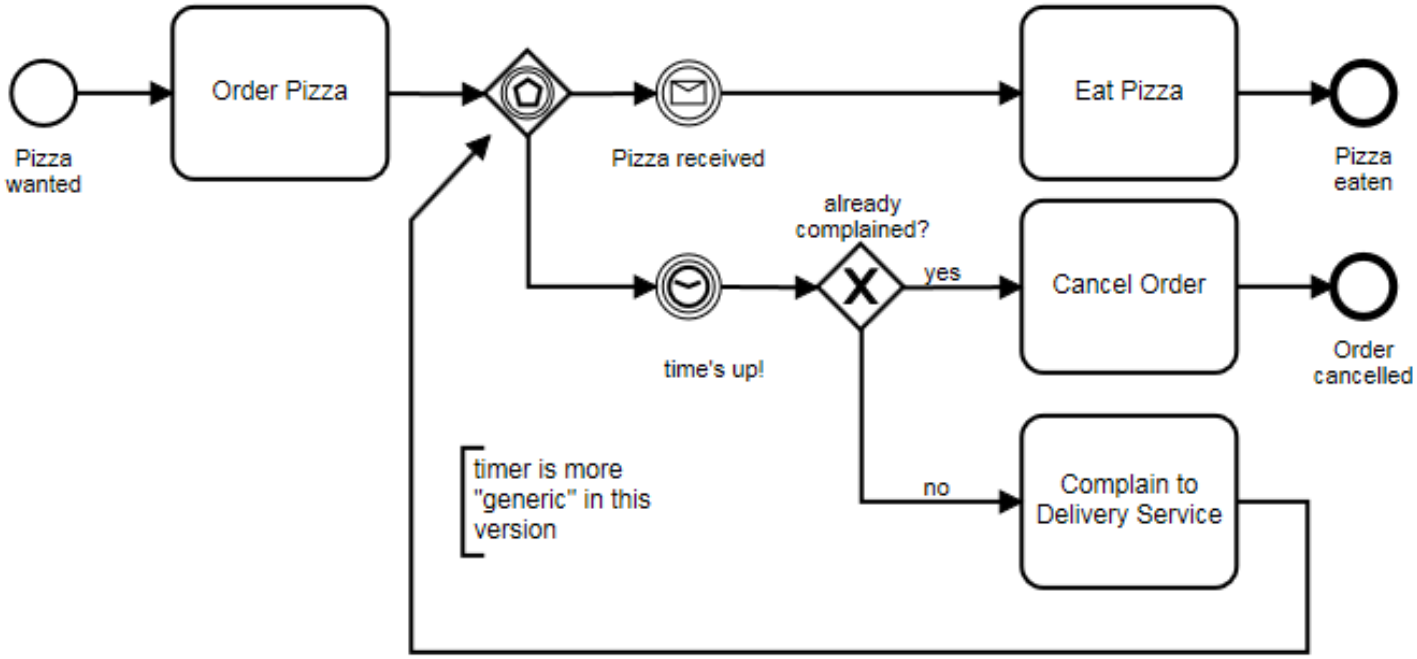
Examples

- **2 Steps Escalation – The Pizza problem**
  - We order a pizza
  - If delivery takes longer than 20 minutes we complain to the delivery service
  - We allow them another 30 minutes
  - If they do not make it in time, we give up and cancel our order



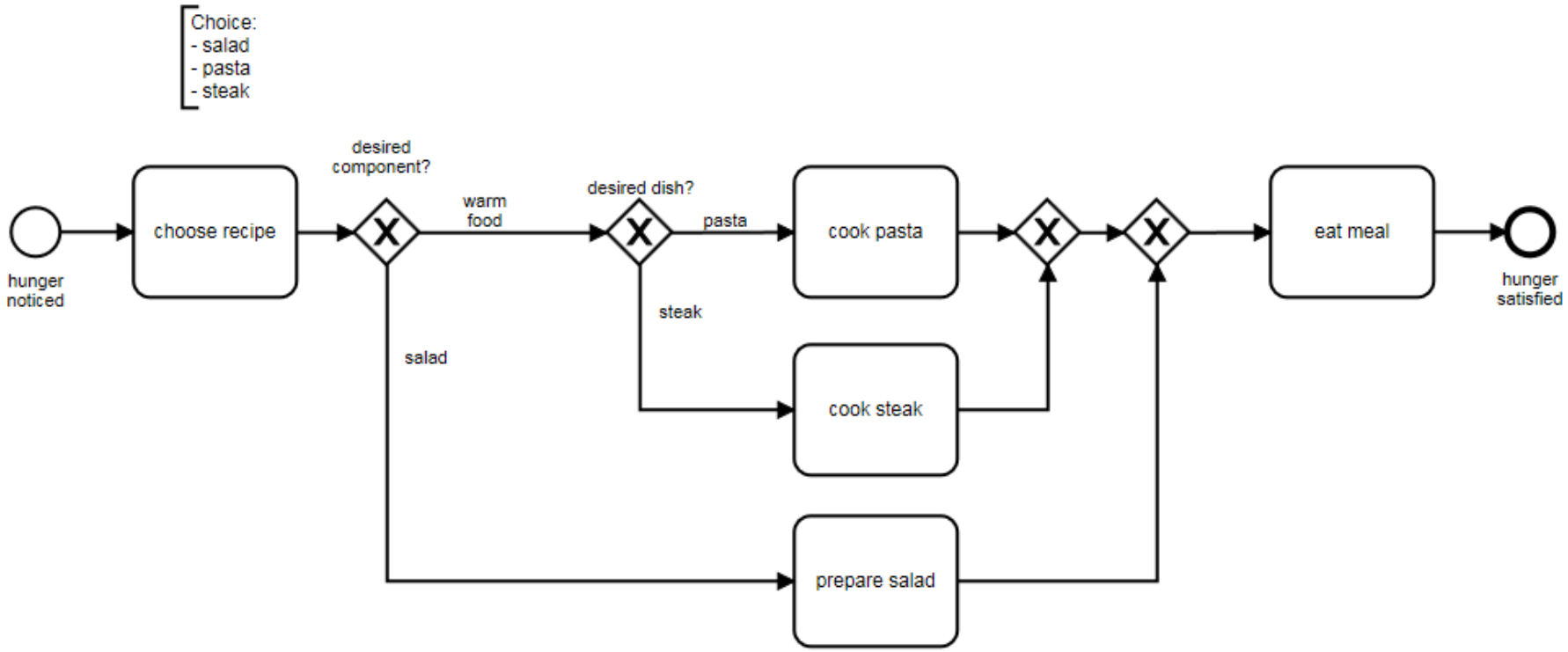
Examples

- **2 Steps Escalation – The Pizza problem**
  - We order a pizza
  - If delivery takes longer than expected we complain to the delivery service
  - We allow them some extra time
  - If we get frustrated for the long waiting , we give up and cancel our order



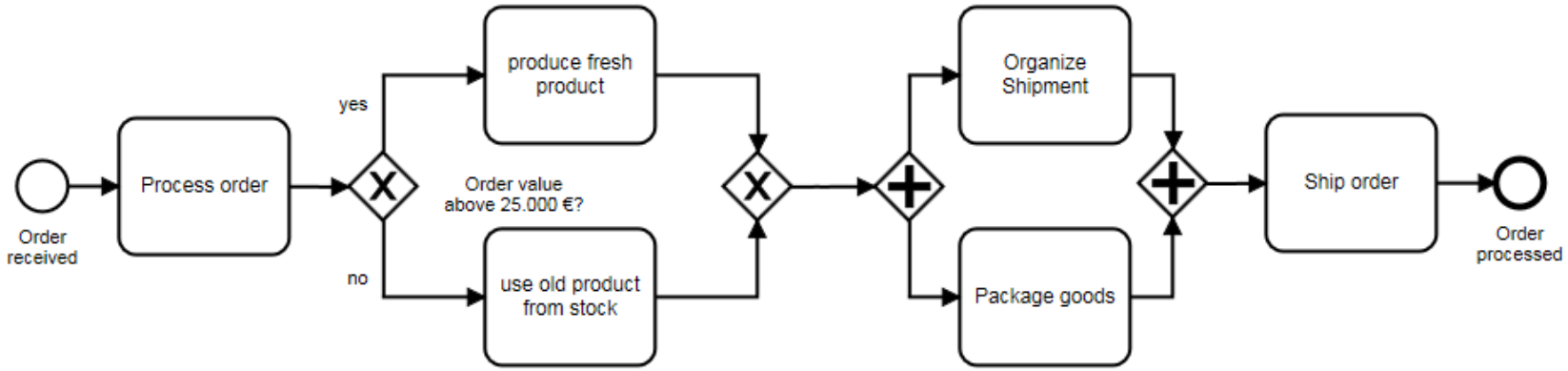
Examples

- **Symmetric modelling in case of complex decisions**
  - The better the layout, the higher the degree of understanding !!!
  - Here's an example concerning a "complex" cooking decision



Examples

- **Symmetric modelling in case of complex decisions**
  - The better the layout, the higher the degree of understanding !!!
  - Here's an example concerning a two step decision with task performed in parallel



Examples

- **Creating a book outline**
  - Two tasks are needed: (*i) Cover creation and (ii) Plot Creation & Text Editing*
  - Cover Creation can starts provided that plot and some concepts have ben defined. Next it can proceed in parallel; also both tasks can be interrupted at any time so that the other one can restart and vice versa
  - Only when both have been completed the process ends