



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ  
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

# Εισαγωγή στα Δίκτυα Υπηρεσιών

**Διάλεξη 6η: Assisting Lecture 3 - BPMN v2.0  
standard**

Μύρων Παπαδάκης  
Τμήμα Επιστήμης Υπολογιστών



# Introduction to Service Networks

## CS-592 – Spring 2015

Assisting Lecture 5

Business Process Model and Notation - **BPMN 2.0**

Myron Papadakis



# What is BPMN?

- The Business Process Modeling Notation (BPMN) is a graphical notation that depicts the steps in a business process
- **Why is BPMN important?**
  - The world of business processes has changed dramatically over the past few years.
  - Processes can be coordinated from behind, within and over organizations natural boundaries.
  - A business process now spans multiple participants and coordination can be complex.
  - Until BPMN, there has not been a standard modeling technique developed that addresses these issues.



## BPMN and UML



- The unified modelling language (UML) takes an object-oriented approach to the modeling of applications, while BPMN takes a process-oriented approach to modelling of systems.
  - BPMN has a focus on business processes
  - UML has a focus on software design
- Not competing notations but different views on systems.



# Introduction to BPMN (1/2)



- Developed by Business Process Management Initiative (BPMI), and is currently maintained by the Object Management Group since the two organizations merged in 2005
- Goal: provide a notation that is readily understandable by all business users
  - *Business analysts*: create the initial drafts of the processes
  - *Technical developers*: responsible for implementing the technology that will perform those processes
  - *Business people*: manage and monitor the processes
- **Bridges the gap between the business process design and process implementation**



## Introduction to BPMN (2/2)

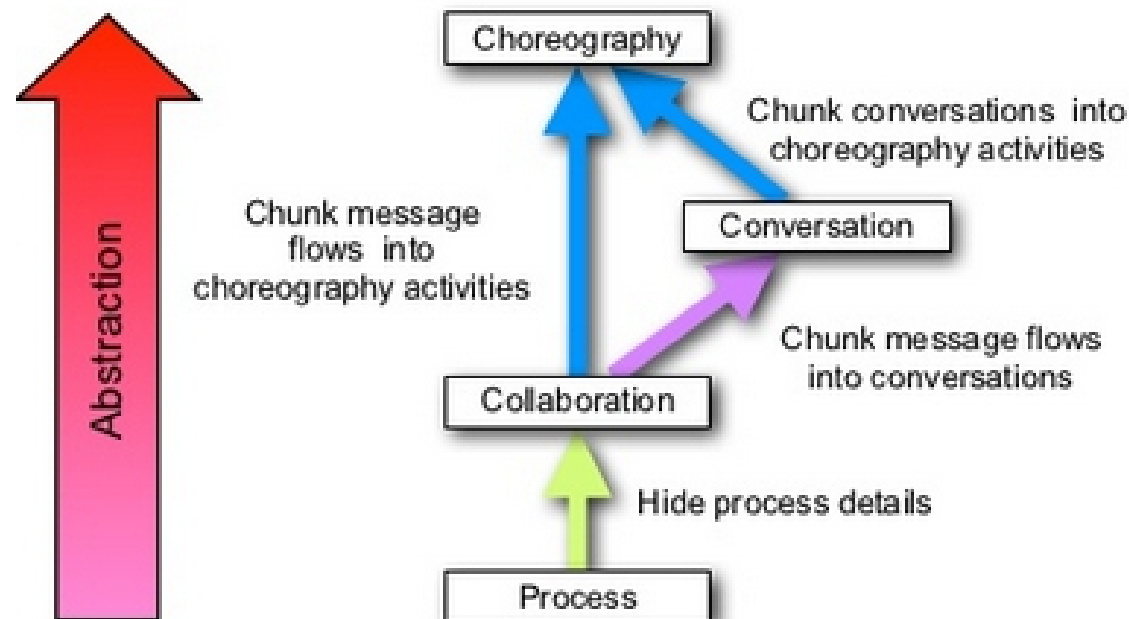


- Similar to flowcharts and UML Activity diagrams
- Flow of activities with various messaging and data
- BPMN 2.0
  - Defines an extensibility mechanism for both Process model extensions and graphical extensions
  - Refines event composition and correlation
  - Extends the definition of human interactions
  - Defines a Choreography Model and a Conversation Model



## BPMN Sub-models (1/2)

- There are three basic types of sub-models within an end-to-end BPMN model
  1. **Processes** (Orchestration)
  2. **Choreographies**
  3. **Collaborations**, which can include Processes and/or Choreographies
- There is a very close link between choreographies, conversations, collaborations and processes





## BPNM Sub-models (2/2)



- **Process** – a sequence of activities that constitutes a business process
  - Focus: the sequence of **activities** and **events**
- **Collaboration** – a process that has two or more participants
  - Focus: the sequence of **activities** and **events** and or the **messages** sent between participants
- **Choreography** – a sequence of interactions between participants
  - Focus: the **participants** in the business process and the **sequence** in which they interact together

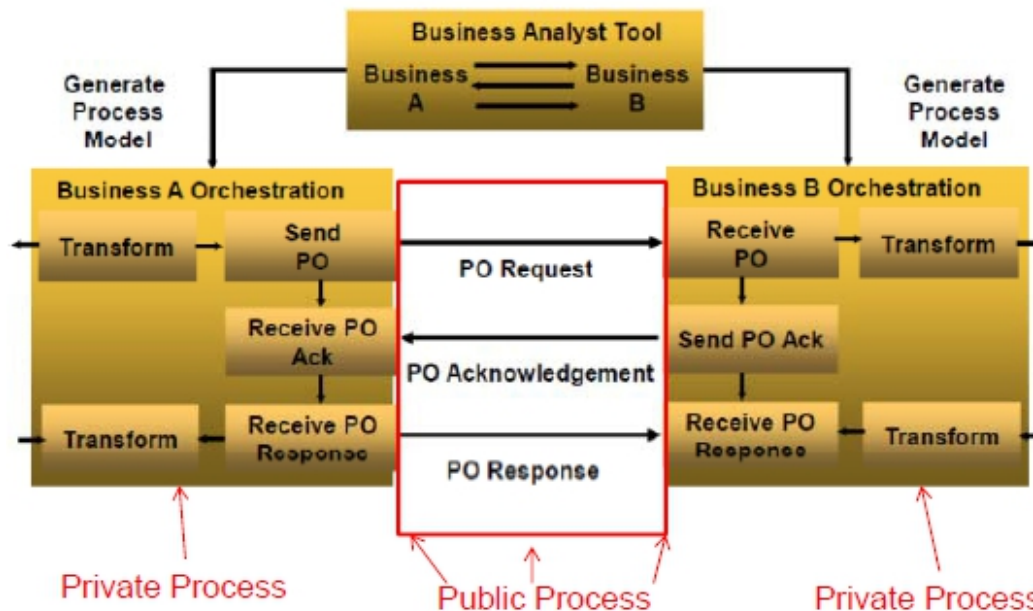




# Orchestration and Choreography



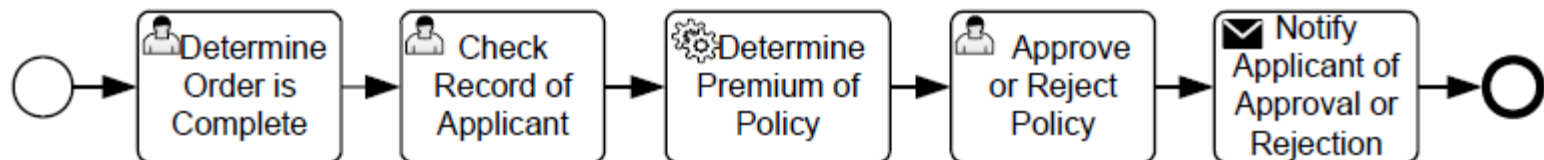
- **Orchestration:** Workflow, internal processes, private processes
  - Contained within one Pool
- **Choreography:** Collaboration, global processes, B2B Processes
  - Defined by the interaction between Pools





## Process Diagrams (1/2)

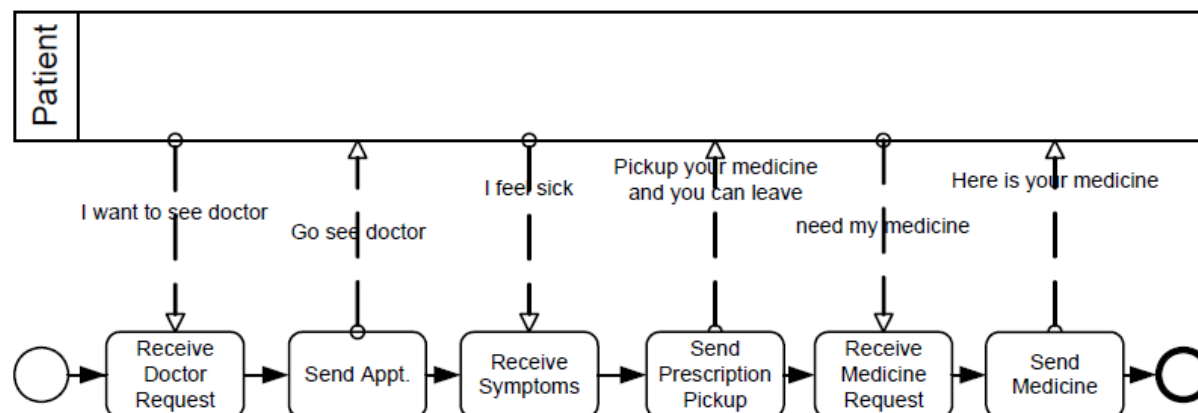
- **Process: A sequence of flow objects**
- **Private Processes:** those internal to a specific organization (also called workflow, BPM processes or orchestration of web services)
  - Contained within a single pool
    1. *Executable*
    2. *Non-executable*: modeled for the purpose of documenting Process behavior at a modeler-defined level of detail





## Process Diagrams (2/2)

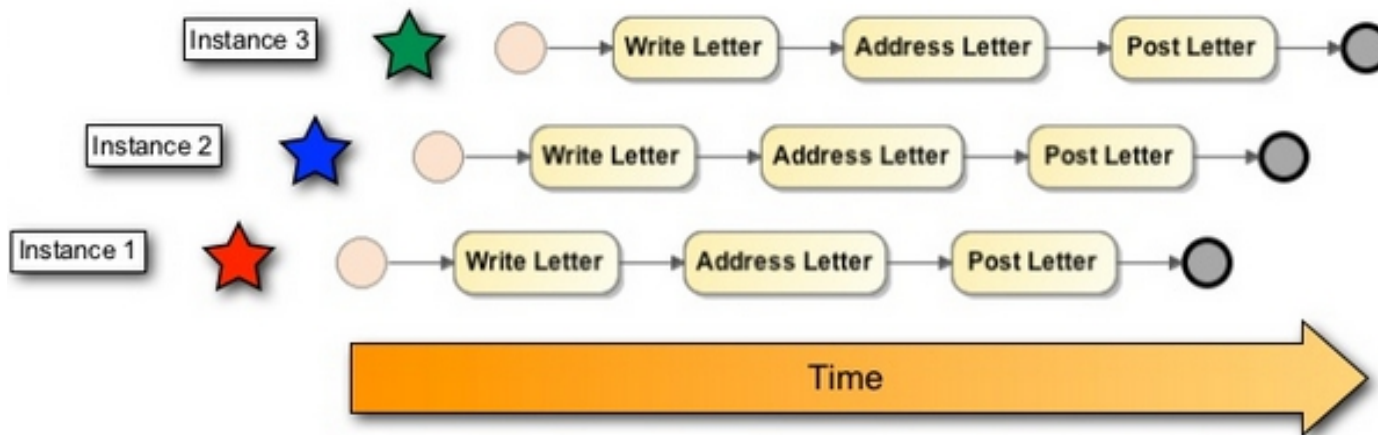
- **Public Process:** represents the interactions between a private Business Process and another Process or Participant.
  - called “abstract” in BPMN 1.2
  - includes only those activities used to communicate with outside world
  - internal activities of the private business process are not shown
  - shows to the outside world the **Message Flows** and the order of those Message Flows that are needed to interact with that **Process**
  - can be modeled separately or within a **Collaboration** to show the flow of **Messages** between the *public Process Activities* and other *Participants*





# Process Instances

- Each time a process receives a new start event, a new instance of that process begins executing
- We say that a **process may have many process instances**

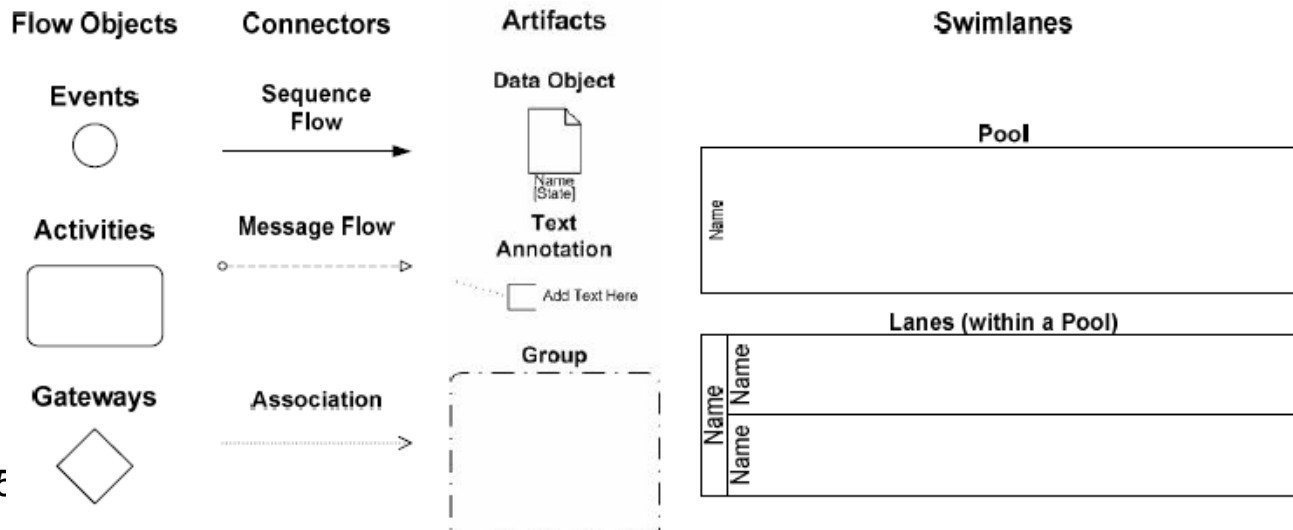




# BPMN Elements for Process Diagrams



- **Flow Objects:** define the behavior of the process
  - Activities, Events, Gateways
- **Connecting objects:** connecting the Flow Objects to each other or other information
  - Sequence flows, Message Flows, Associations, Data Association
- **Data:** Data objects, Data Inputs, Data Outputs, Data Store
- **Swimlanes:** ways of grouping modeling elements
  - Pools, Lanes
- **Artifacts:** add additional information to models
  - Group (grouping related diagram elements), Text Annotation (documenting our models)





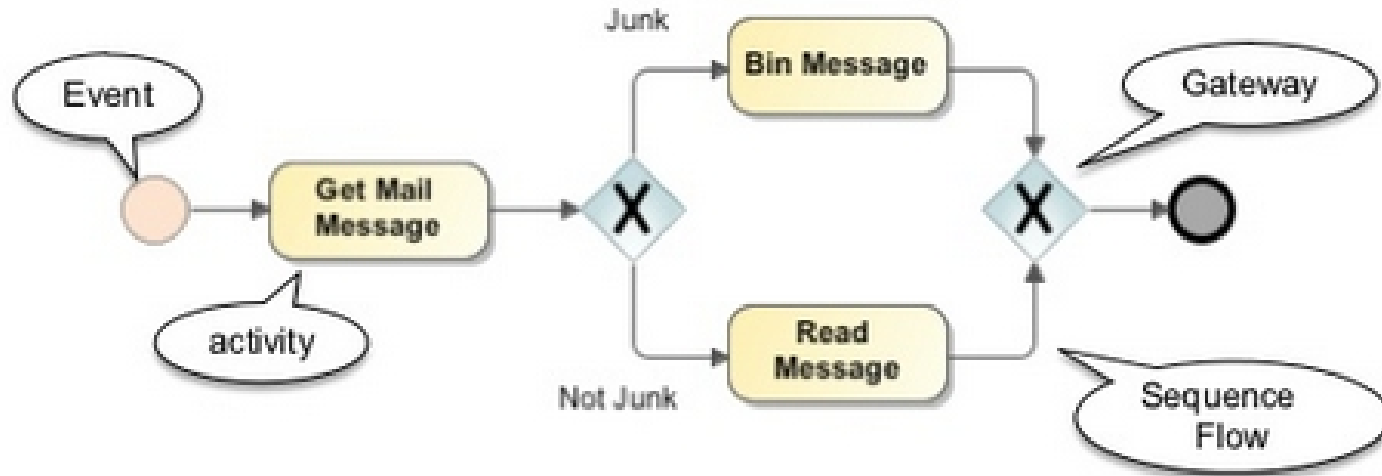
# BPMN Elements for Process Diagrams > Flow Objects



- **Events**
  - something that “happens” during the course of a business process
  - affect the flow of the process, have a cause (trigger) or an impact (result)
  - 3 types of events (when): *Start, Intermediate, End*
- **Activities**
  - work that company performs
  - types of activities: *Task, Sub-Process, Call Activity*
- **Gateways**
  - used to control the divergence and convergence of Sequence Flow
  - determine traditional decisions, forking, merging, joining of paths



# BPMN Elements for Process Diagrams > Flow Objects

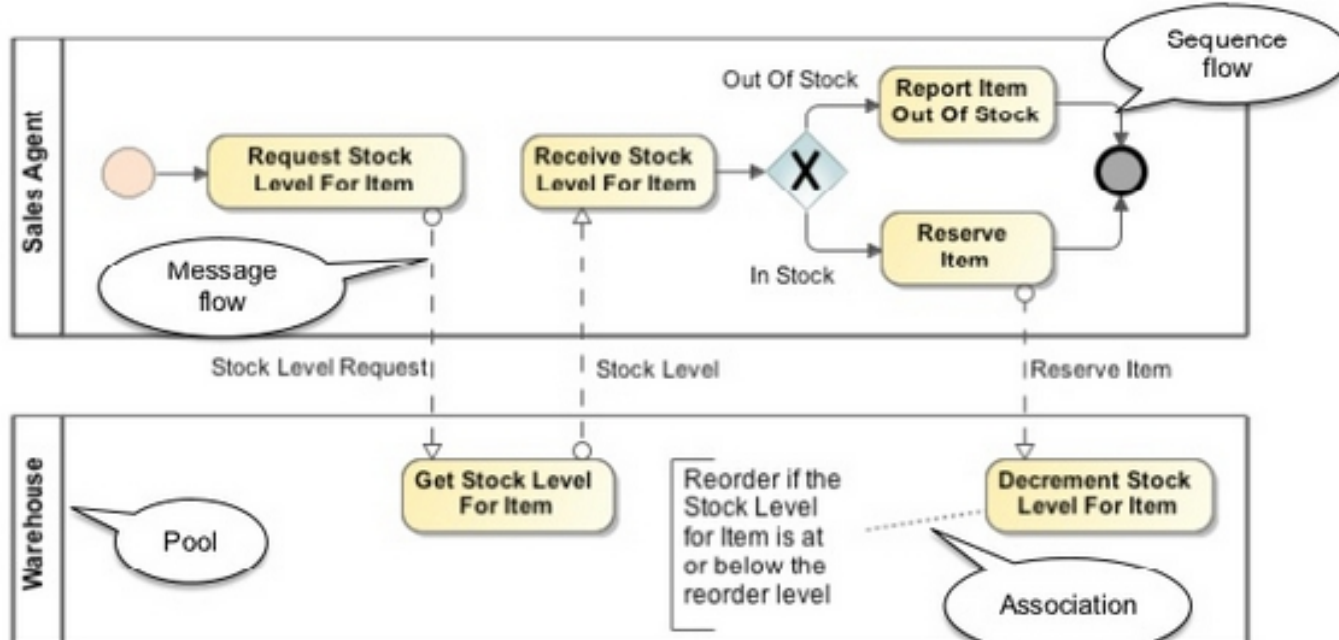




# BPMN Elements for Process Diagrams > Flow Objects > Connecting Flow Nodes



- **Process:** a sequence of flow objects
- **Connect flow nodes** together
  - Sequence Flows: determine the sequence of activities
  - Message flows: messages between process participants
  - Associations: associate text or data with modeling elements



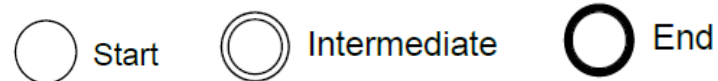




# BPMN Elements for Process Diagrams > Flow Objects > Events



- Description of **event-driven processes**
- Start of an Activity, end of an Activity, message that arrives, etc
- They can start, delay, interrupt, or end the flow of the process.
  - **Start** Events: indicate where a process will start, emit a token (thin line)
  - **Intermediate** events: indicate where something happens somewhere between the start and end of a Process. (double line)
    - **boundary** events (attached on the activity to catch an event)
  - **End** events: where a path of a process will end, consume a token (thick line)



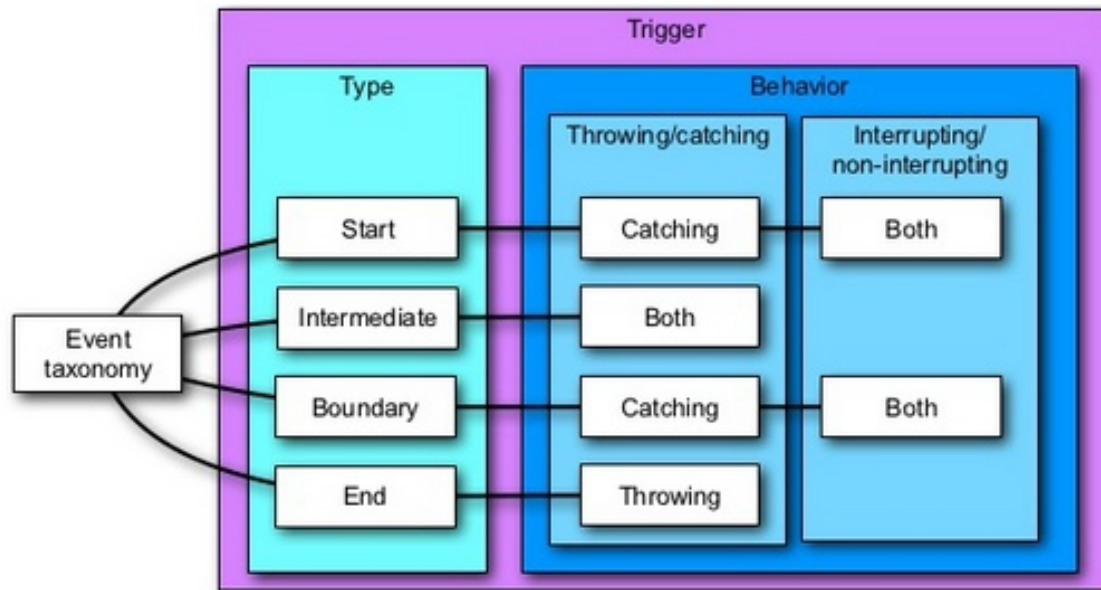
- *Events that catch a trigger*: all Start Events and some Intermediate Events
  - Trigger: determines what causes the event to fire, indicated by a symbol in the event icon
- *Events that throw a Result*: all End Events and some Intermediate Events



# BPMN Elements for Process Diagrams > Flow Objects > Events > Event Semantics



- We can classify the many BPMN2 events according to:
  - **Type**: where they can be used in the process
  - **Trigger**: determines what causes the event to fire
  - **Behavior** – throw or catch – interrupt the containing activity or not

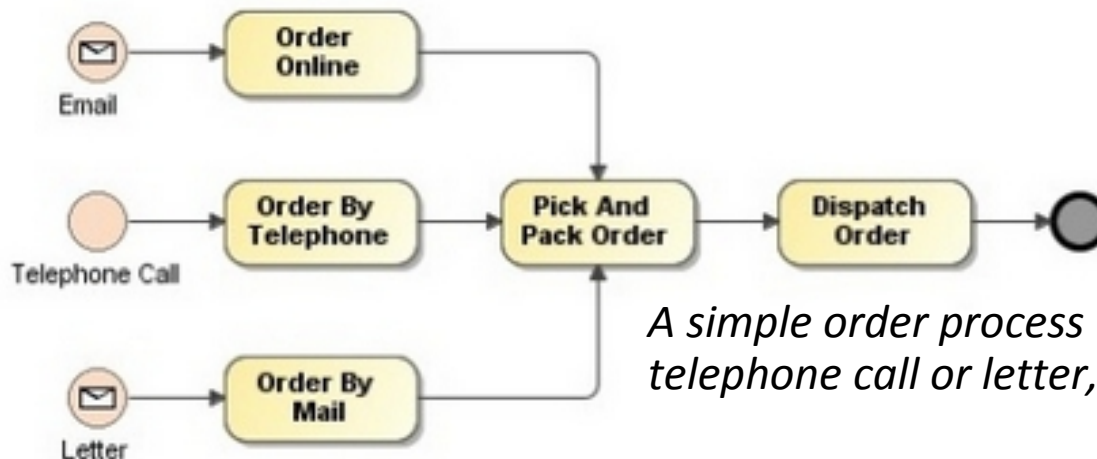
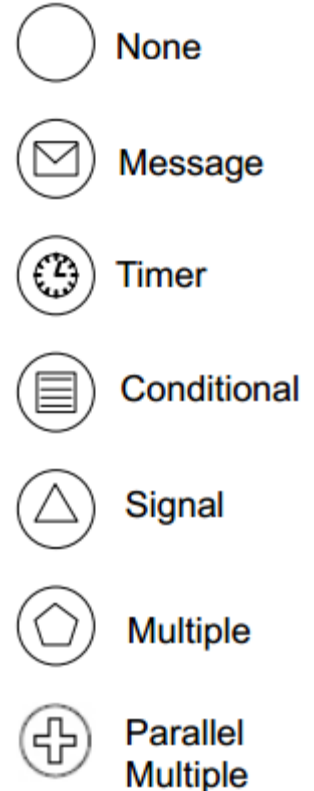




# BPMN Elements for Process Diagrams > Flow Objects > Start Events



- A Start Event shows where a Process can begin.
- **Different types of Start Events** to indicate the varying circumstances that can trigger the start of a Process.
  - These circumstances, such as the arrival of a message or a timer “going-off,” are called **triggers**.
- A Start Event can only have **outgoing Sequence Flows**.
- Trigger based start events are never used in sub-processes
- We can have more than one start events
  - But only one none start event per process!



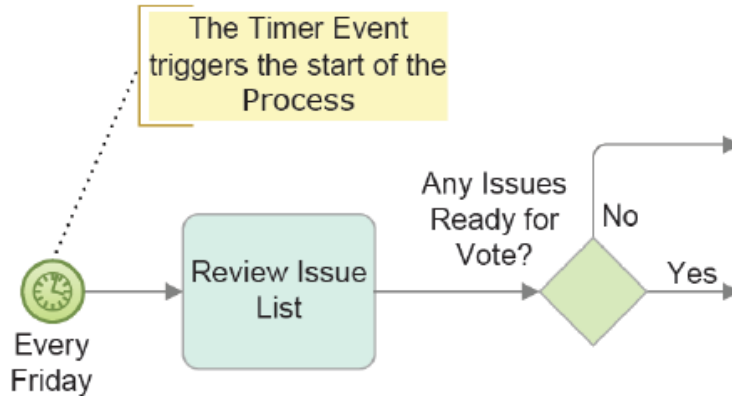
*A simple order process that may be started by email, telephone call or letter, requesting items..*



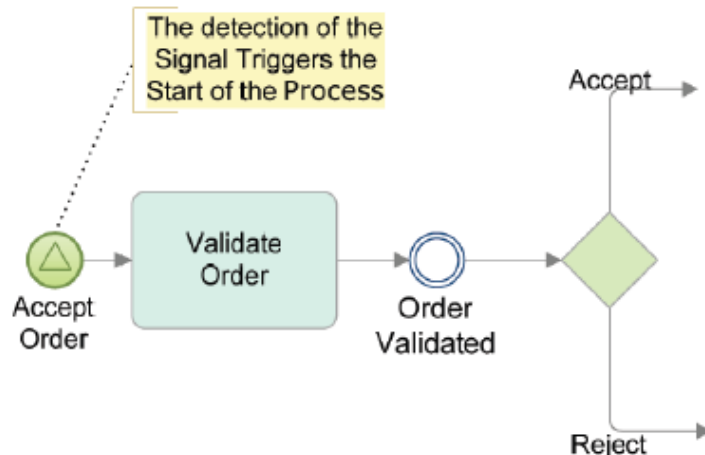
# BPMN Elements for Process Diagrams > Flow Objects > Start Events Examples



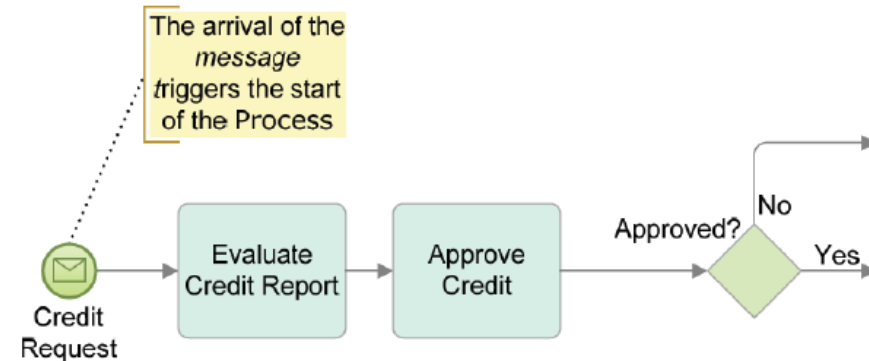
**Timer:** the Process is started (i.e., triggered) when a specific time condition has occurred.



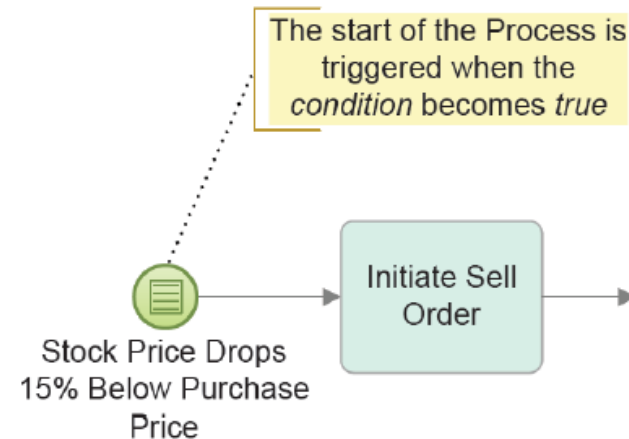
**Signal:** the Process is started (i.e., triggered) when a signal is detected



**Message:** a Process is initiated (i.e. triggered) by the reception of a message, participants must be in separate pools



**Conditional:** a Process is started (i.e., triggered) when a pre-defined condition becomes true





# BPMN Elements for Process Diagrams > Flow Objects > Intermediate Events



- An Intermediate Event indicates where something happens/occurs **after a Process has started and before it has ended.**
- They can be **placed in the normal flow** of the Process or **attached to the boundary** of an activity
- They may **interrupt** the normal processing of an Activity.
- Each type of Intermediate Events can either **throw or catch the event.**
  - A catching Intermediate Event waits for something to happen (i.e., wait for the circumstance defined on the trigger).
  - A throwing Intermediate Event immediately fires (effectively creating the circumstance defined on the trigger).
  - A throwing Intermediate Event can not be attached to the boundary of an Activity.

## Catching Throwing



None



Timer



Message



Signal



Conditional



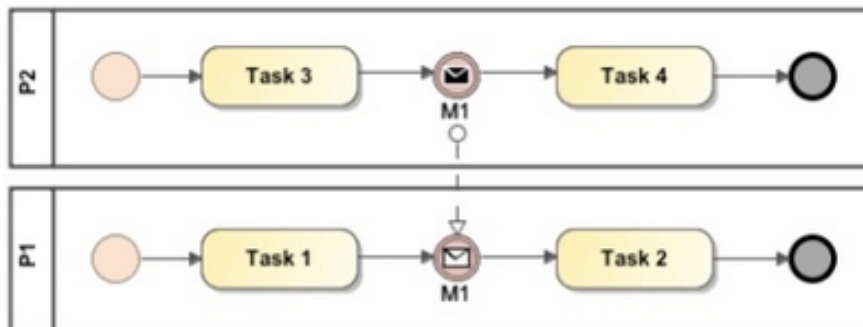
Link



Multiple



Parallel  
Multiple

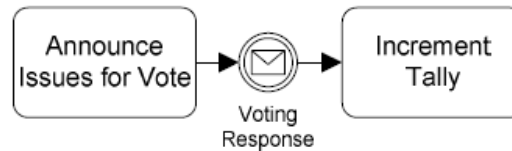




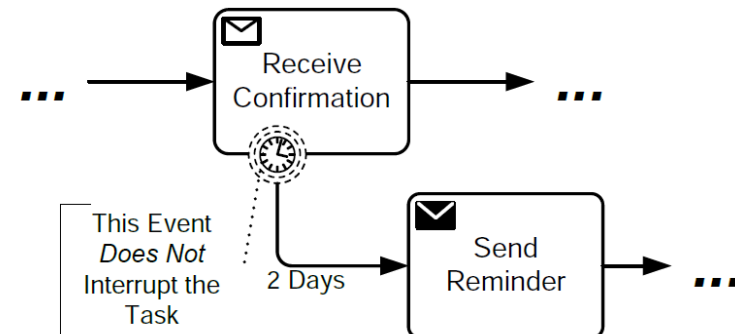
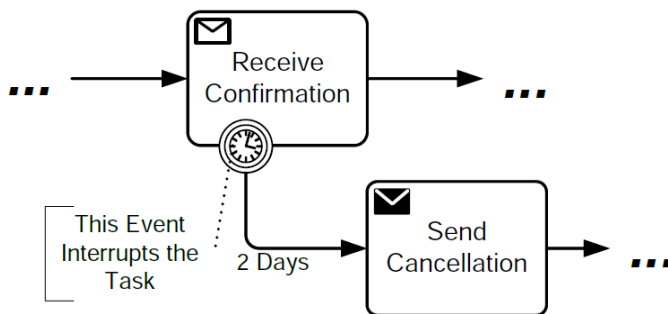
# BPMN Elements for Process Diagrams > Flow Objects > Intermediate Events



- **Normal Flow:** Events that are placed within the process flow represent things that happen during the normal operations of the process, i.e.
  - the response to the Event (i.e. receipt of a message)
  - the creation of the Event (i.e. sending of a message)



- **Boundary:** Events that are attached to the boundary of an activity
  - They can be attached to either Tasks or Sub-Processes







# BPMN Elements for Process Diagrams > Flow Objects > Intermediate Events (Boundary)



Boundary event semantics		
Syntax	Behaviour	Semantics
	Interrupting	If Branch triggered before Task 1 finishes terminate Task 1 execute Task 3 Else execute Task 2
	Non-interrupting	If Fork triggered before Task 1 finishes continue executing Task 1 AND fork to execute Task 3 concurrently Else execute Task 2

*Boundary events: catching intermediate events*

*Interrupting: branch the process flow, non-interrupting: fork the process flow*



# BPMN Elements for Process Diagrams > Flow Objects > Events > Event Types



Types	Start			Intermediate				End
	Top-Level	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	
None								
Message								
Timer								
Error								
Escalation								
Cancel								
Compensation								
Conditional								
Link								
Signal								
Terminate								
Multiple								
Parallel Multiple								



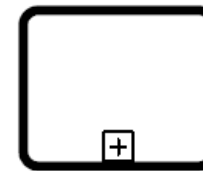
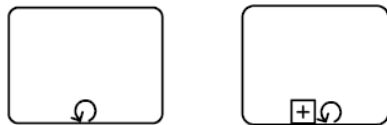


# BPMN Elements for Process Diagrams > Flow Objects > Activities



- **Activity**: work that is performed within a **Business Process**.
- **Performer**: assigned to an Activity to indicate the responsibility to carry out the work
- Types of activities:
  - **Task**: the Process cannot be broken down to a finer level of detail
  - **Sub-Process**: has an internal structure modeled
  - **Call Activity**: 'wrapper' for the invocation of a global Process or Global Task within the execution.

- Standard loops: Activities may be repeated sequentially



- Multi-instance loops: sequential and parallel :



*collapsed*

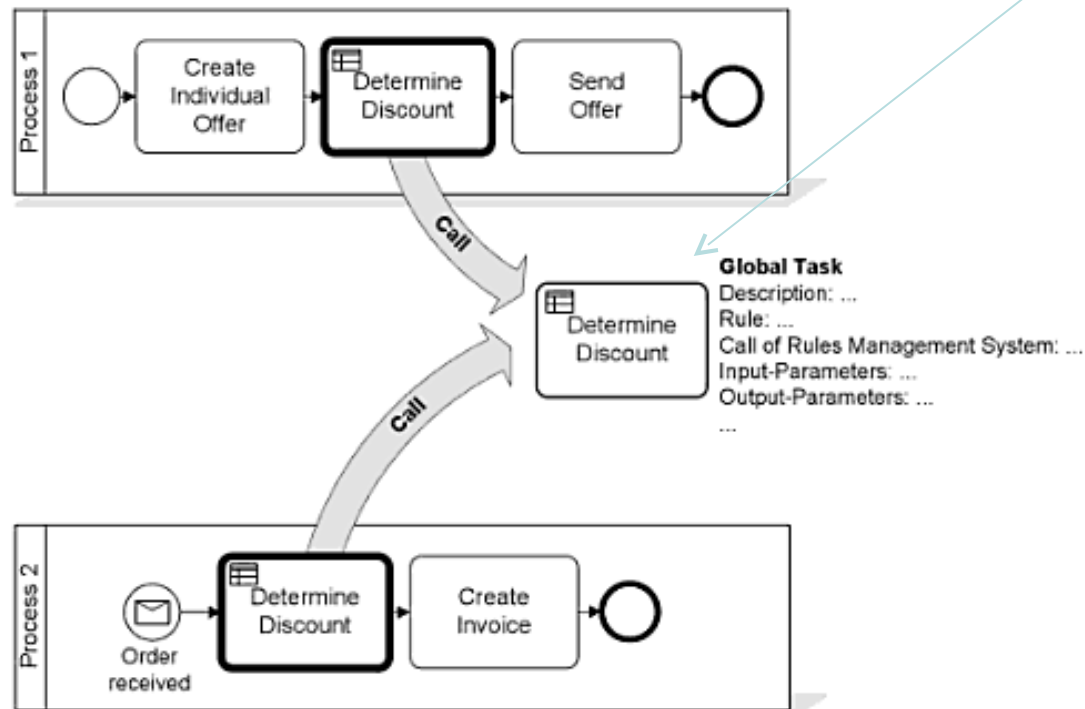




# BPMN Elements for Process Diagrams > Flow Objects > Activities > Call Activity Example

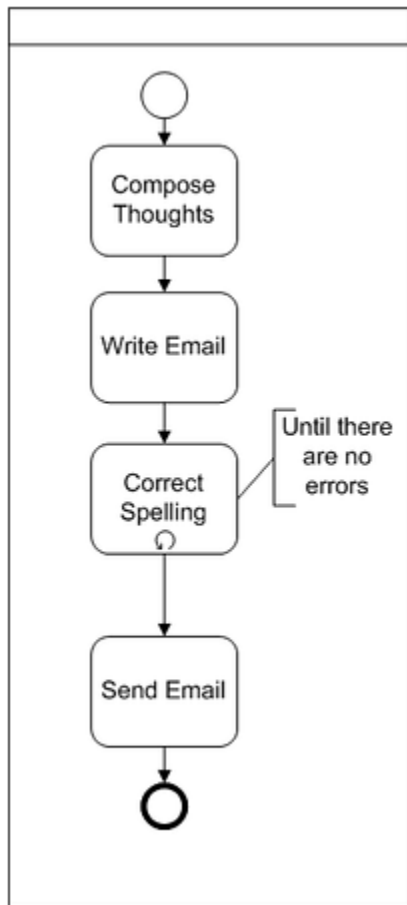


- Call global task from different processes
- Determine discount is a global task, defined only once
- Processes 1 and 2 contain a **call activity** referring to the global task

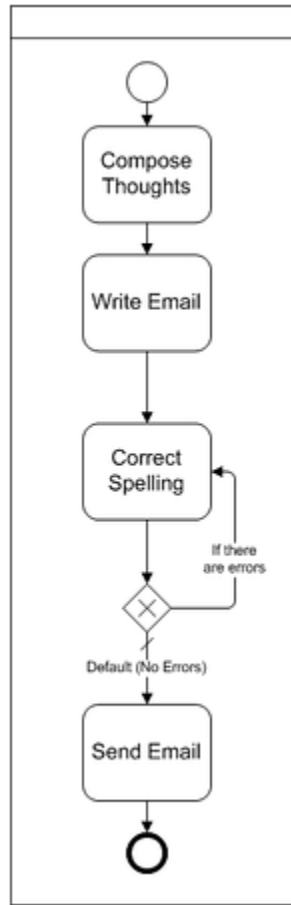




# BPMN Elements for Process Diagrams > Flow Objects > Activities > Loop and Multiple Instance Examples

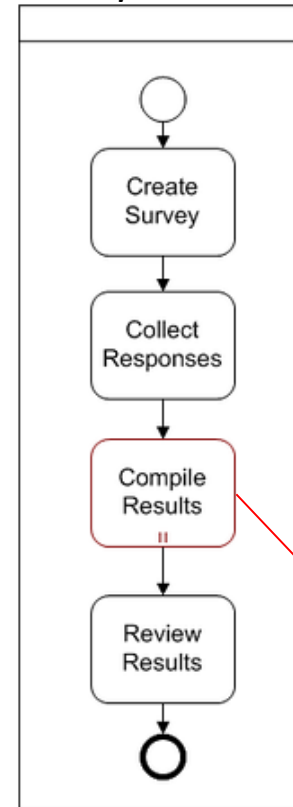


Loop Marker



Using gateways

## Multiple Instance Task Example



*Multiple instance activity:  
carried out for each of the  
compiled results,*

*Number of instances  
determined by the number of  
compiled results*

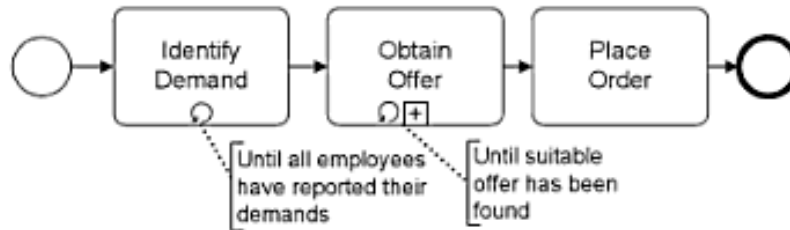
*Since each individual response to the survey can be tabulated independently (parallel), this model shows the task as a multiple instance task.*



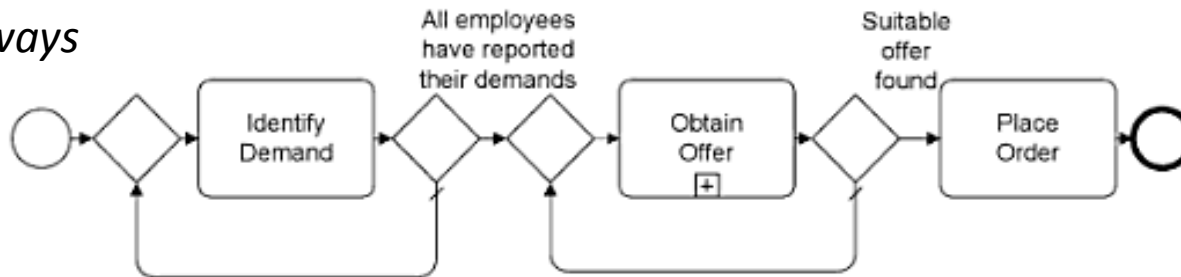
# BPMN Elements for Process Diagrams > Flow Objects > Activities > Ways of Modeling Loops



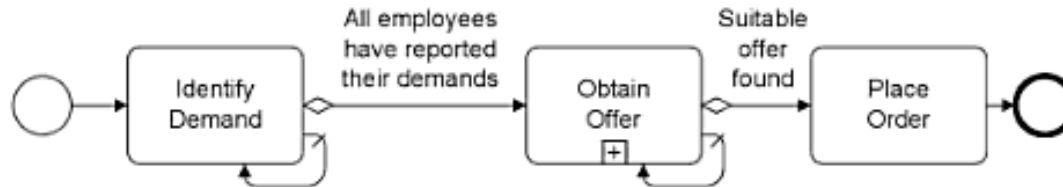
*Using loop markers*



*Using gateways*



*Using conditional flow*



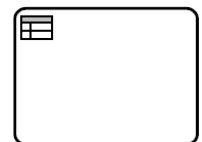
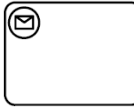
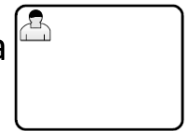
## ***Different ways of modeling loops***



# BPMN Elements for Process Diagrams > Flow Objects > Activities > Task Types



1. **User**: A task where a human performer carries out the task with the assistance of a software application (e.g. user changes his password)
2. **Receive** : Waits for a message to arrive from an external participant relative to the Business Process). Once received, the Task is complete.
3. **Send** : Dispatches a message to an external participant.
4. **Service** : Links to some sort of service, which could be a web service or an automated application (e.g. calculate shipping date)
5. **Script** : Performs a modeler-defined script.
6. **Manual** : A non-automated task that a human performer undertakes outside of the control of the workflow or PMS engine (e.g. telephone technician installing a phone at a customer location), similar to user task but without help of software
7. **Business Rules** : represents an Activity in the Process where a decision engine evaluates Process data and returns the results (also called decision task)

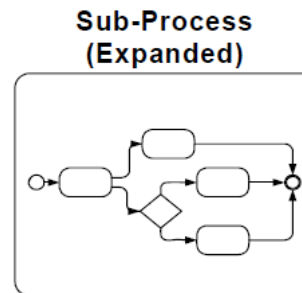
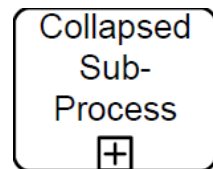




# BPMN Elements for Process Diagrams > Flow Objects > Activities > Sub-processes



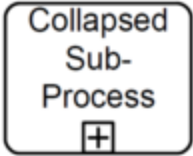



- An Activity whose internal details have been modeled using activities, Gateways, Events, and Sequence Flows
- Can be decomposed hierarchically into other activities
- **Sub process MUST define an internal process with a start and end event.**
- Only reusable within the parent process (i.e. it is not reusable in the overall design).
- The details of the sub-process are not visible in the diagram.
- Internal of a Sub-Process can be modeled with:
  - Activities, Gateways, Events, Sequence Flows





# BPMN Elements for Process Diagrams > Flow Objects > Activities > Sub-processes > Types



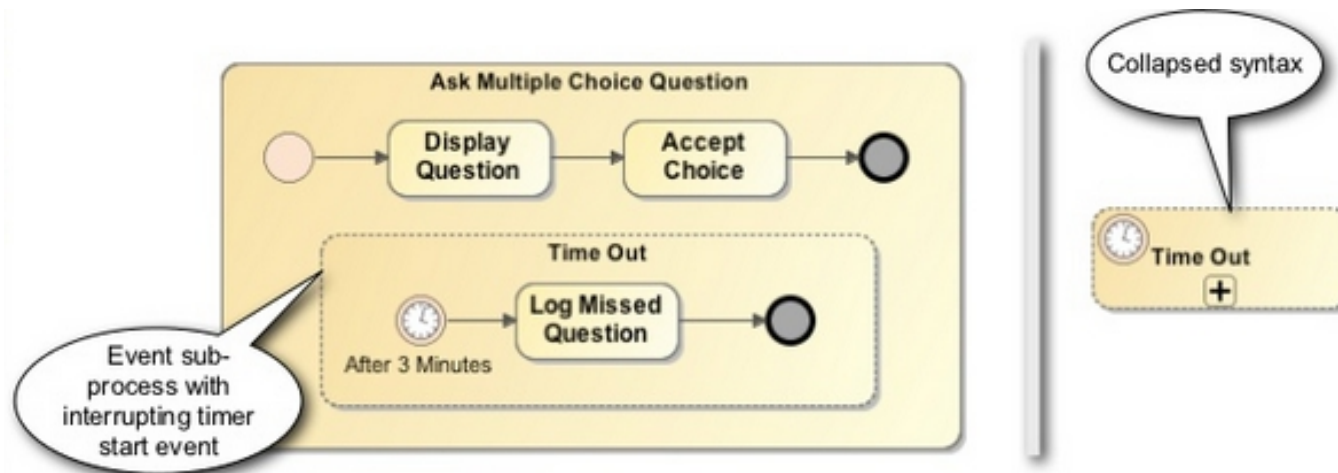
<b>(Embedded) sub-process (collapsed)</b>	The <u>normal type</u> of sub-process. It is embedded in the parent process, activated by it, and can access all of its global data	
<b>Ad-hoc sub-process (collapsed)</b>	The sequence of activities is determined by the process performers. Each activity may be performed zero or more times.	
<b>Event sub-process (collapsed)</b>	A sub-process that has no incoming or outgoing sequence flows. It is triggered by a message, error, escalation, compensation, signal or multiple event.	
<b>Transaction sub-process</b>	The sub-process is all or nothing- it runs to completion or fails completely leaving the system in a consistent state.	



# BPMN Elements for Process Diagrams > Flow Objects > Activities > Sub-processes > Types > Event Sub-process Example



- **Event Sub-processes:** different from normal sub-processes because they are **not part of the normal process flow**:
  - No incoming or outgoing sequence flows, hence not part of the parent process flow!
  - **Always triggered by a start event** whose trigger may be message, error, escalation, compensation, conditional, signal, timer or multiple
  - Must have a dashed boundary



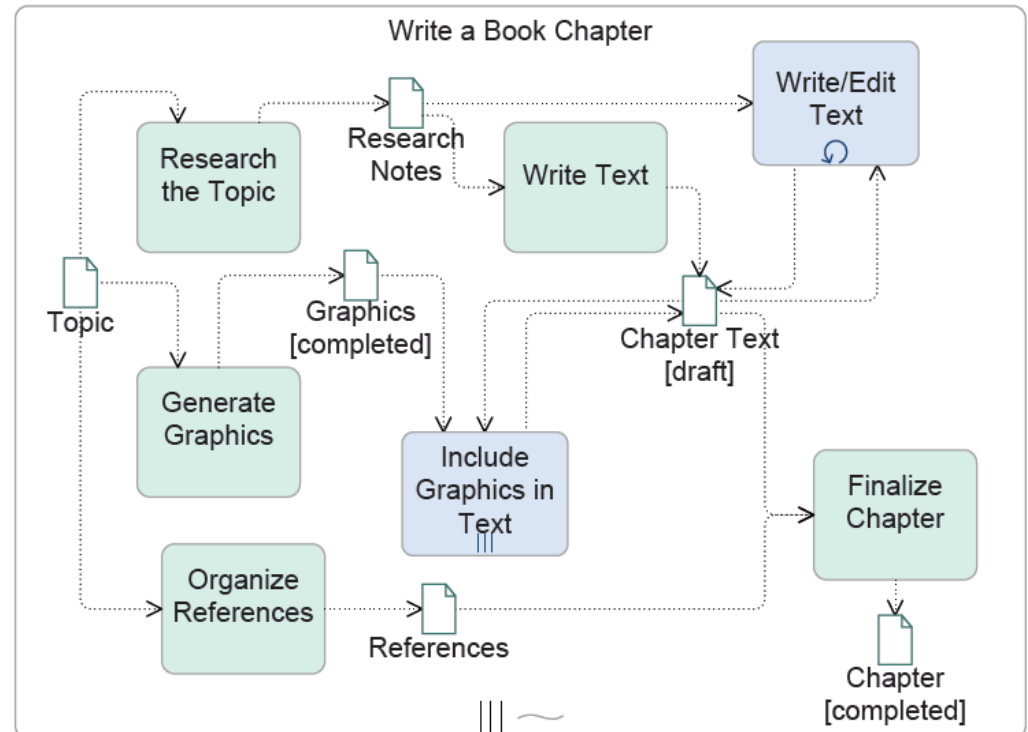
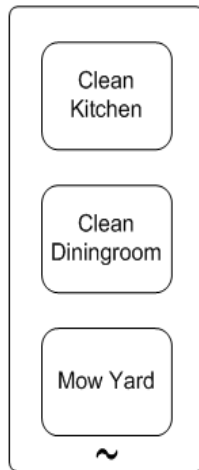
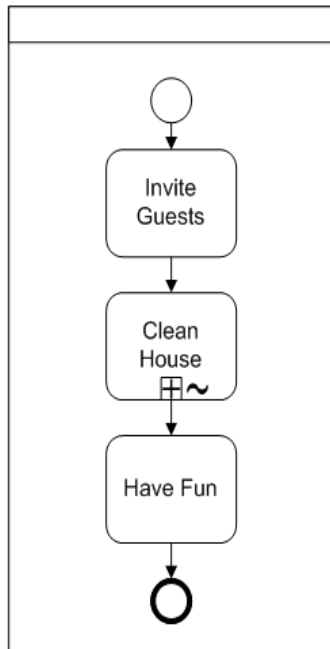




# BPMN Elements for Process Diagrams > Flow Objects > Activities > Sub-processes > Types > Ad Hoc Process Example



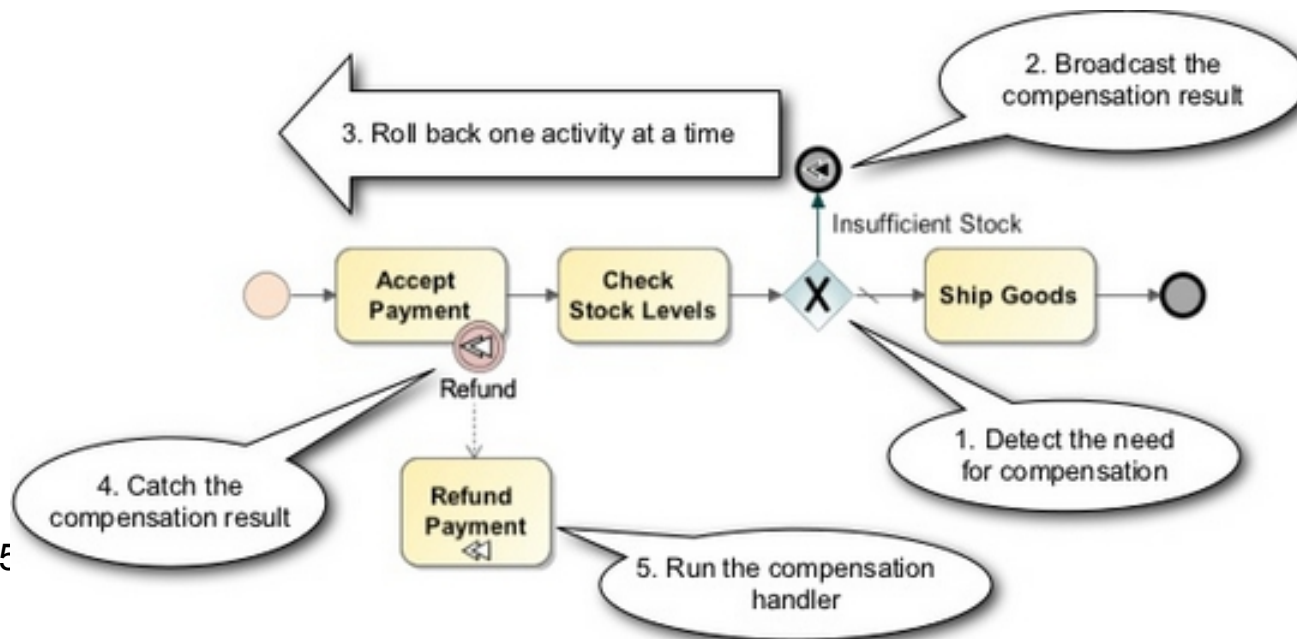
- Activities in an Ad Hoc Process involve human performers who make the decisions as to what Activities to perform, when to perform them, and how many times. The sequence of activities is determined by the performers





# Compensation

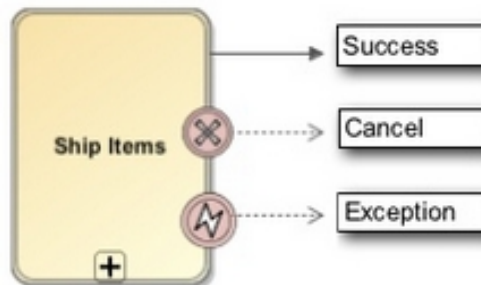
- Involves undoing activities or processes that have already been completed
  - The activity or process is rolled back one activity at a time.
  - Each activity is replaced by its compensation handler (if it has one) which executes.
- **Compensation handler: purpose to undo the action of the activity to which it is associated**





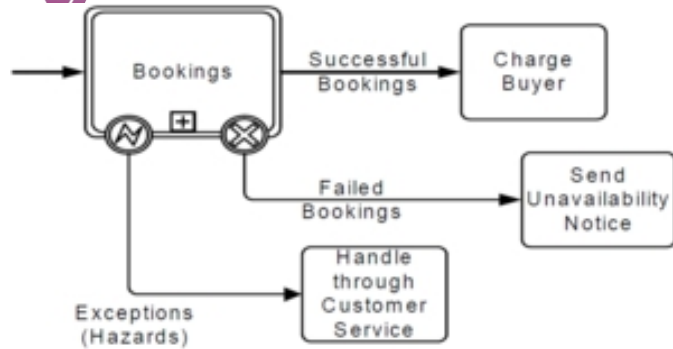
# Transactions

- The transaction is treated as a discrete unit of work.
- **All the activities in the transaction are performed or none of them are.**
- BPMN defines three possible outcomes for transactions:
  - a) **Success:** everything works as planned. Exits via a sequence flow when all paths have succeeded (neither cancelled nor terminated by a hazard.)
  - b) **Cancel:** the transaction is cancelled. Exits via a cancel boundary event. All transaction activities are rolled back and compensated if possible
  - c) **Exception:** something has gone wrong. Exits on a error boundary event. No roll-back possible! Compensation is not performed!

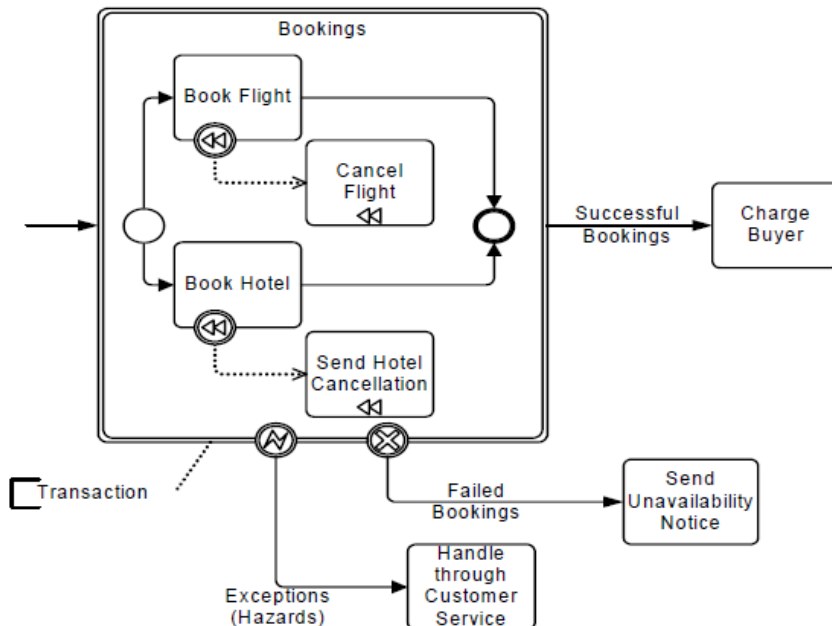




# Compensation and Transactions



- A Cancel Intermediate Event represents the path to follow a cancelled completion
- An Exception Intermediate Event represents the path to follow a transaction hazard (but no compensation is performed)
- Activities used for compensate (with marker) are outside normal flow and are Associated normal activities.
- The link between the normal Activity and the Compensation Activity is done through an Association rather than a Sequence Flow.
- For a Transaction to succeed, all parties involved have to perform their own Activities and reach an agreement point.
- If any one of them withdraws or fails to complete, then the Transaction cancels and all parties need to undo all the work that has completed.
- The Compensation Intermediate Event is never triggered during the normal flow of the Process. It only can be triggered during the roll-back of the Transaction Sub-Process.





# BPMN Elements for Process Diagrams > Flow Objects > Gateways (1/3)

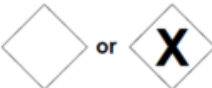
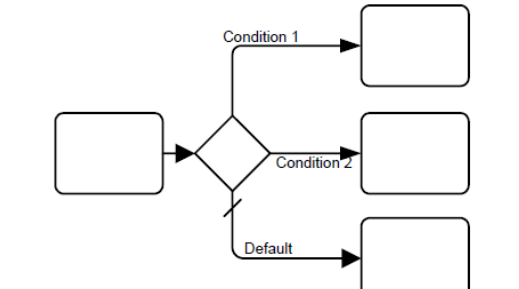

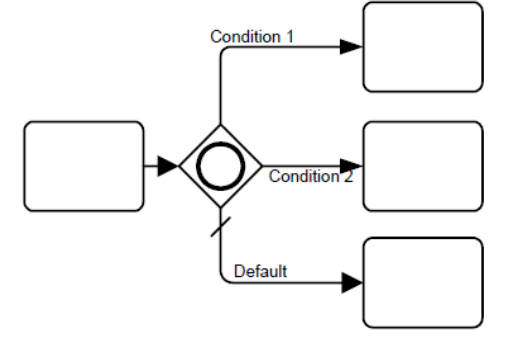

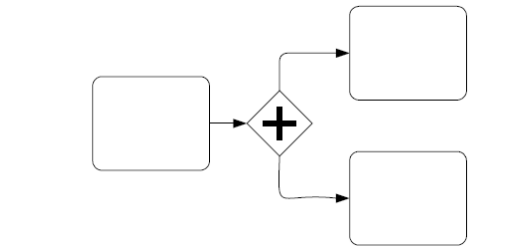


- Used to **control the flow** through the process
- Gateways **converge (merge)** and **diverge (split) the flow** through a process
- There are five types of gateways
  - **Exclusive** and **parallel** gateway are the basic gateways
- Each type has specific converging and diverging behavior controlled by:
  - Conditions on the incoming and outgoing flows
  - Events on the outgoing flows
  - A single condition on the gateway itself



# BPMN Elements for Process Diagrams > Flow Objects > Gateways (2/3)


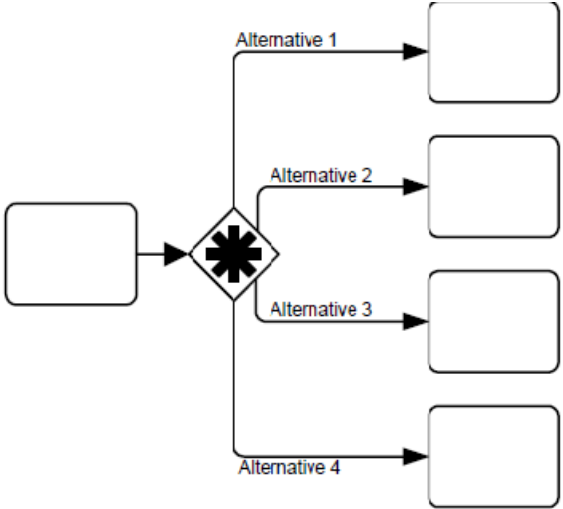

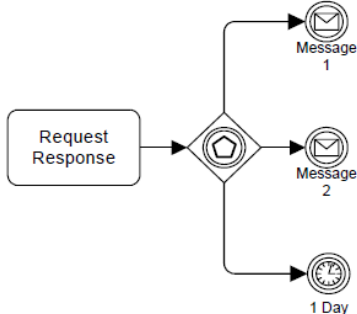


<p><b>Exclusive</b></p> 	<p>Create Alternative paths and <b>only one</b> can be taken (diverging), consumes incoming token.</p> <p>Diverging (output): emits single token to outgoing flow whose condition is True</p>	
<p><b>Inclusive</b></p> 	<p>Converging (input): merges tokens from upstream diverging inclusive gateway</p> <p>Diverging (output): Create alternative paths but also parallel. <b>All</b> combinations of paths can be taken</p>	
<p><b>Parallel</b></p> 	<p>Waits for token on all of its input flows (fork) and emits token to all outgoing flows (fork)</p>	



# BPMN Elements for Process Diagrams > Flow Objects > Gateways (3/3)

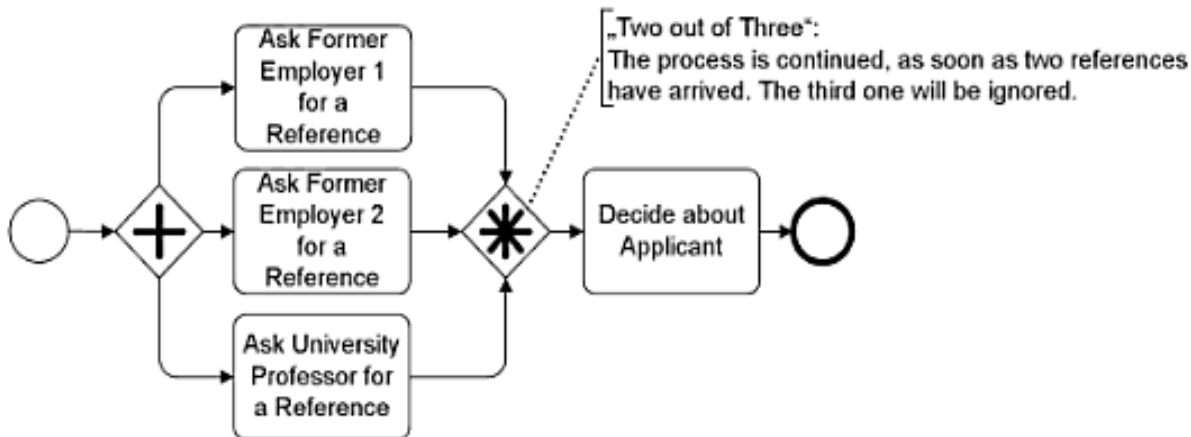
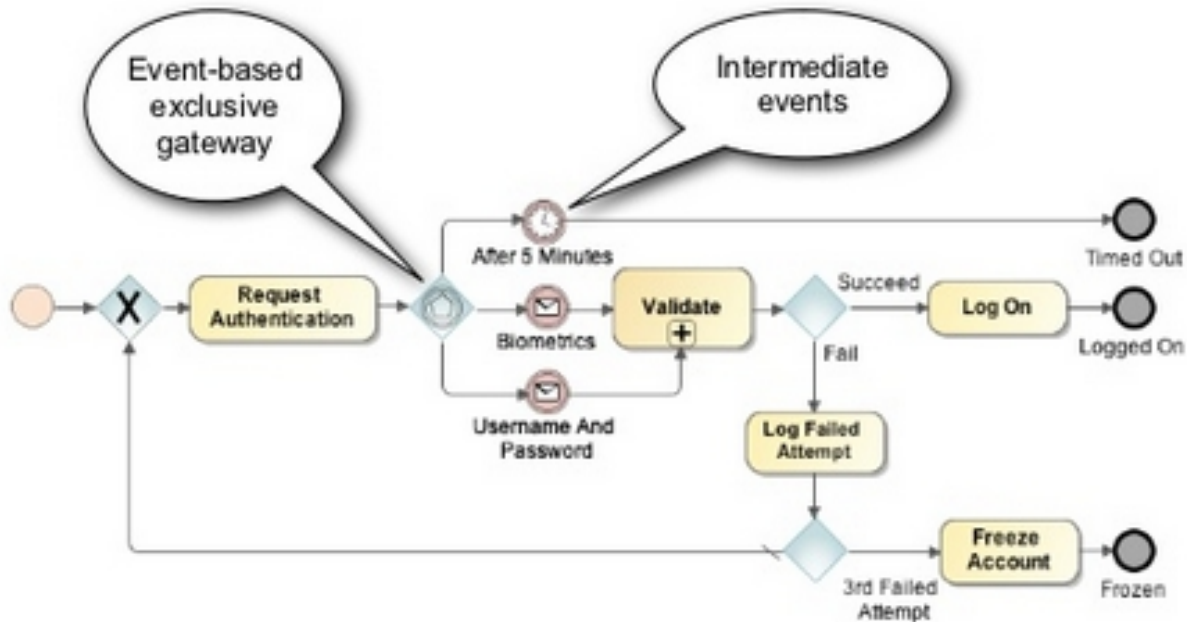


<p><b>Complex</b></p> 	<p>Model complex synchronization behavior Has an internal <b>state</b>.</p> <p><b>Similar to Inclusive, but the activation condition determines if the instance can continue if not all of the tokens have arrived at the gateway merge</b></p>	
<p><b>Event-based</b></p> 	<p>Alternative paths based on events. Type of exclusive gateway (instead of decisions, events)</p> <p>Converging: consumes incoming token Diverging: emits token on all outgoing flows which are connected to intermediate events. <u>1<sup>st</sup> to fire passes on token</u></p>	





# BPMN Elements for Process Diagrams > Flow Objects > Gateways > Examples

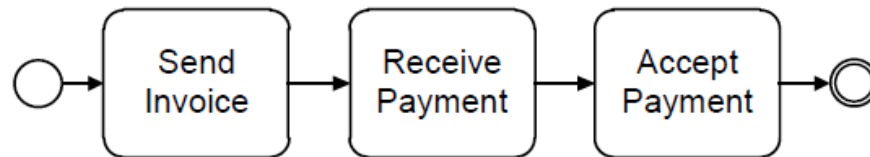






# Activities and Flows

- The **Sequence Flow** defines the order of flow objects in a Process (Activities, Events, and Gateways).
- Each Activity can have one or more incoming Sequence Flow and one or more outgoing Sequence Flow.




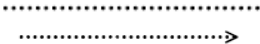



- Typically, an Activity will tend to have a single incoming and a single outgoing Sequence Flow.



# Sequence Flow



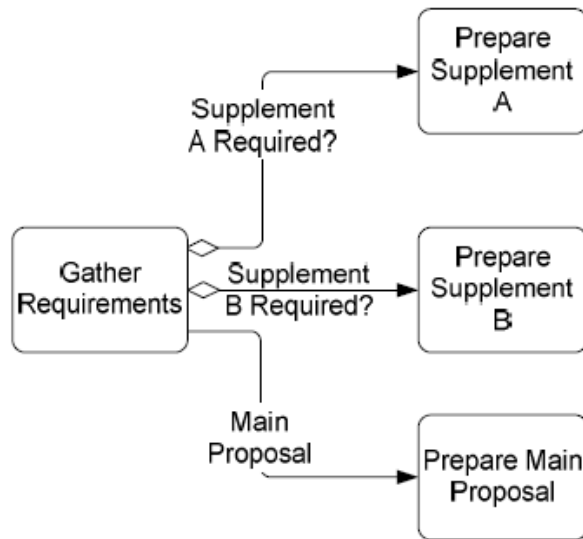
<b>Sequence Flow</b>	Show the order of flow elements in a Process or a Choreography Source & target: <i>events</i> (all types), <i>activities</i> (task and sub-process), <i>choreography activities</i> (for choreography diagrams) and <i>Gateways</i>	
	<b>Conditional (Sequence) Flow:</b> define a condition expression indicating that the token will be passed down if the condition evaluates to true. After an Activity, a diamond symbol is used From Gateways (not parallel or event), no diamond symbol needed	
	<b>Default (Sequence) Flow:</b> Source: Activity, Gateway (Exclusive, Inclusive, Complex) Default flow is taken if all outgoing sequence flows from the above elements are not valid	
<b>Association</b>	Associates Artifacts with Flow Objects	
<b>Data Association</b>	Shows inputs and outputs or movement of data between Data Objects. They do not affect the flow.	



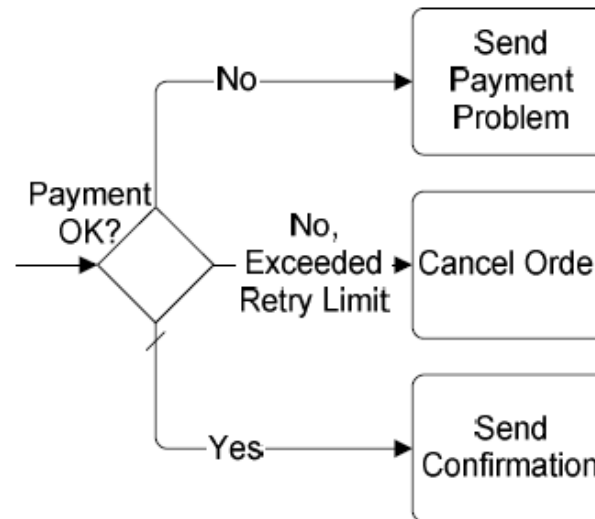
# Sequence Flow > Examples



## Conditional Flow



## Default Flow



A **hatch** mark at the line beginning shows the **default** Sequence Flow

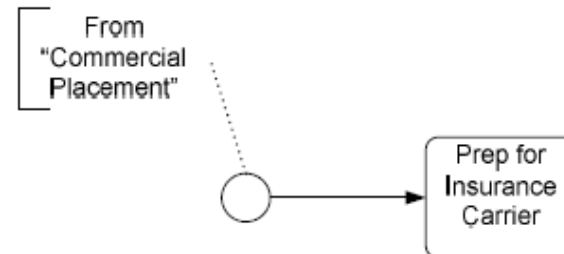
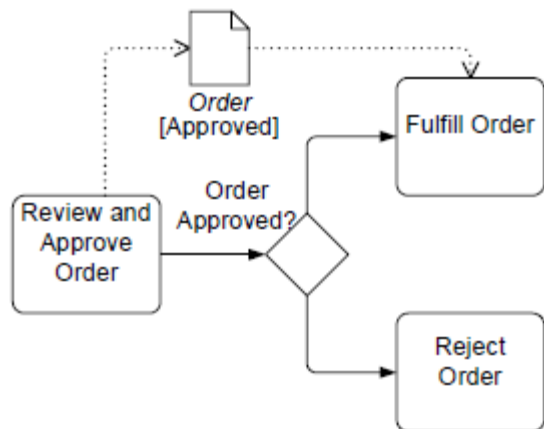
The default path is chosen only if all the other conditions of the Gateway are False



## Sequence Flow > Associations > Examples

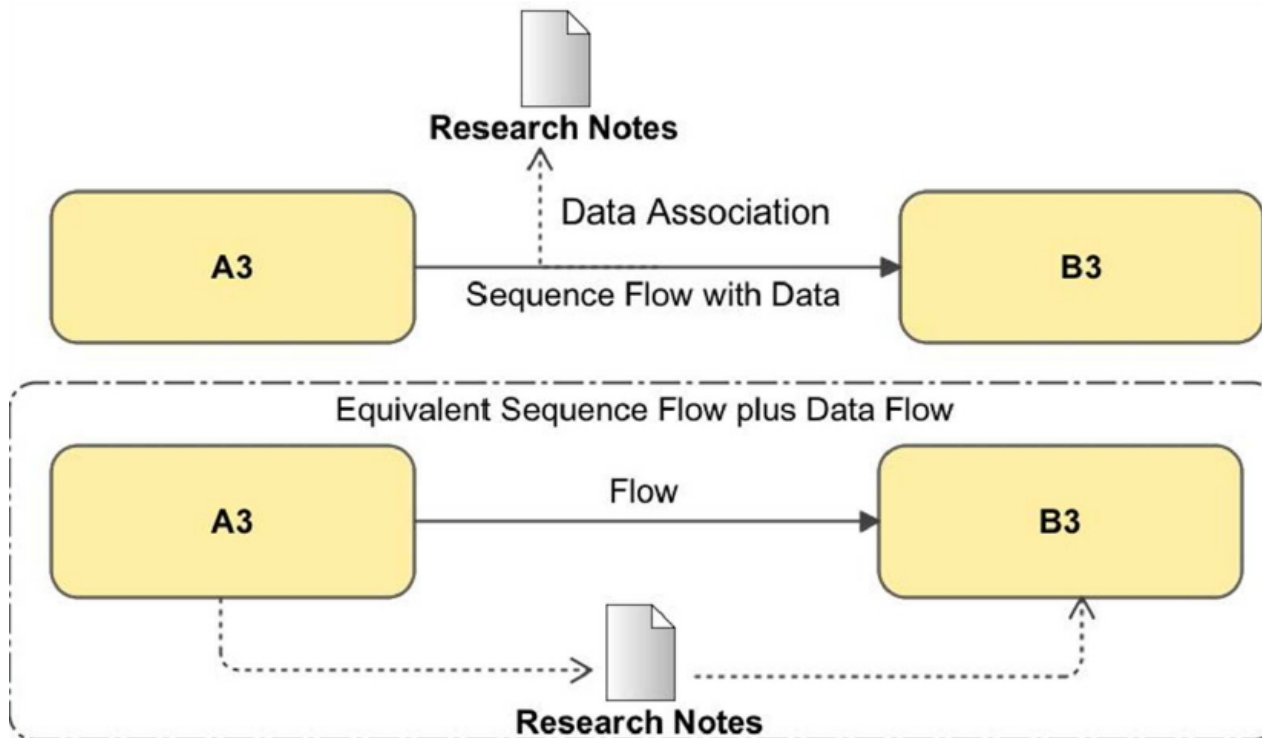


- An Association is used to associate objects to one another (such as Artifacts and Activities)
- Associations are used to show how data is input to and output from Activities
- Text Annotations can be associated with objects





# Sequence Flow > Data Association > Examples





# Data Elements

<b>Data objects</b>	Represents information flowing through the Process, e.g. business documents, emails ,letters). Lifecycle is tightly tied with lifecycle of its parent Process or Sub-process. Data in a data object can be accessed by its immediate parent and siblings. Can also represent a <i>collection</i> of elements. Have state.	
<b>Data Store</b>	A Data Store is a place where the process can read or write data, e.g. a database. <b>It persists beyond the lifetime of the process instance.</b>	
<b>Data inputs</b>	Specifies data requirements needed for the execution of tasks, global tasks and top-level processes and reusable processes Represent inputs to the top-level Process or to show inputs of a called Process	
<b>Data outputs</b>	Data result of the entire process	

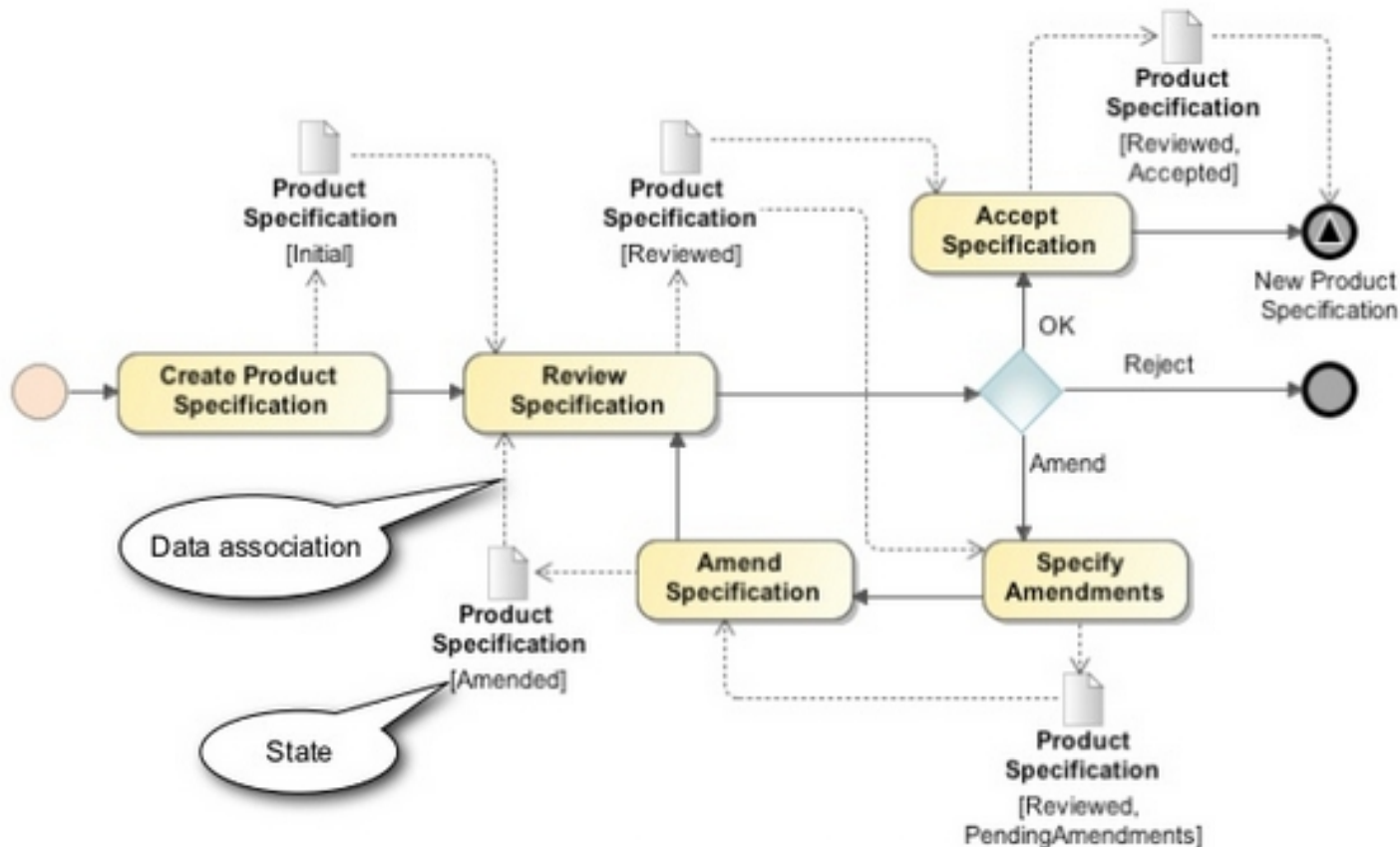
*Data inputs and data outputs are hardly ever used*



# Data Elements > Data Objects Example

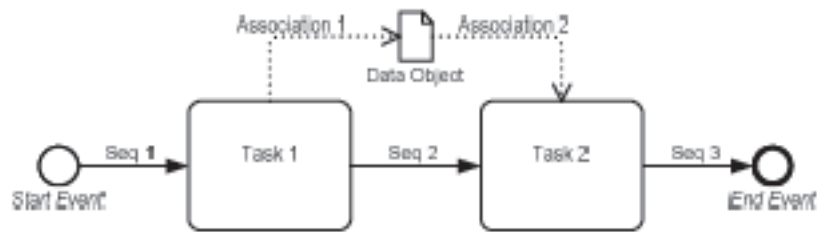


- Use **states** to show how data objects are transformed by the activities





# Data Elements > Data Objects and Data Flow Example



Sequence Flow and Data Flow are decoupled



They can be bound together

*Data objects connected to other elements by data associations, not by sequence flows!*

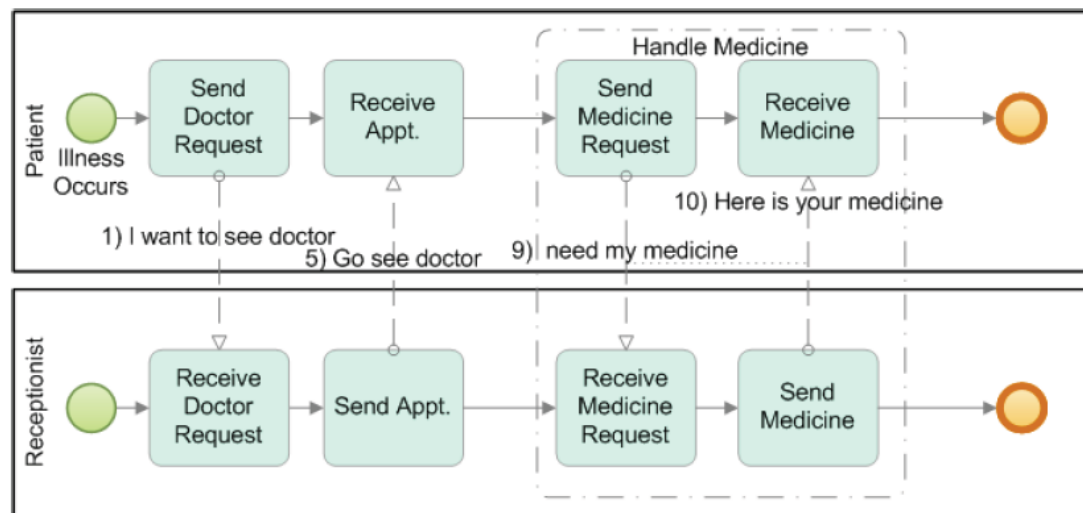
*Indicate data flowing through the process*





# Artifacts

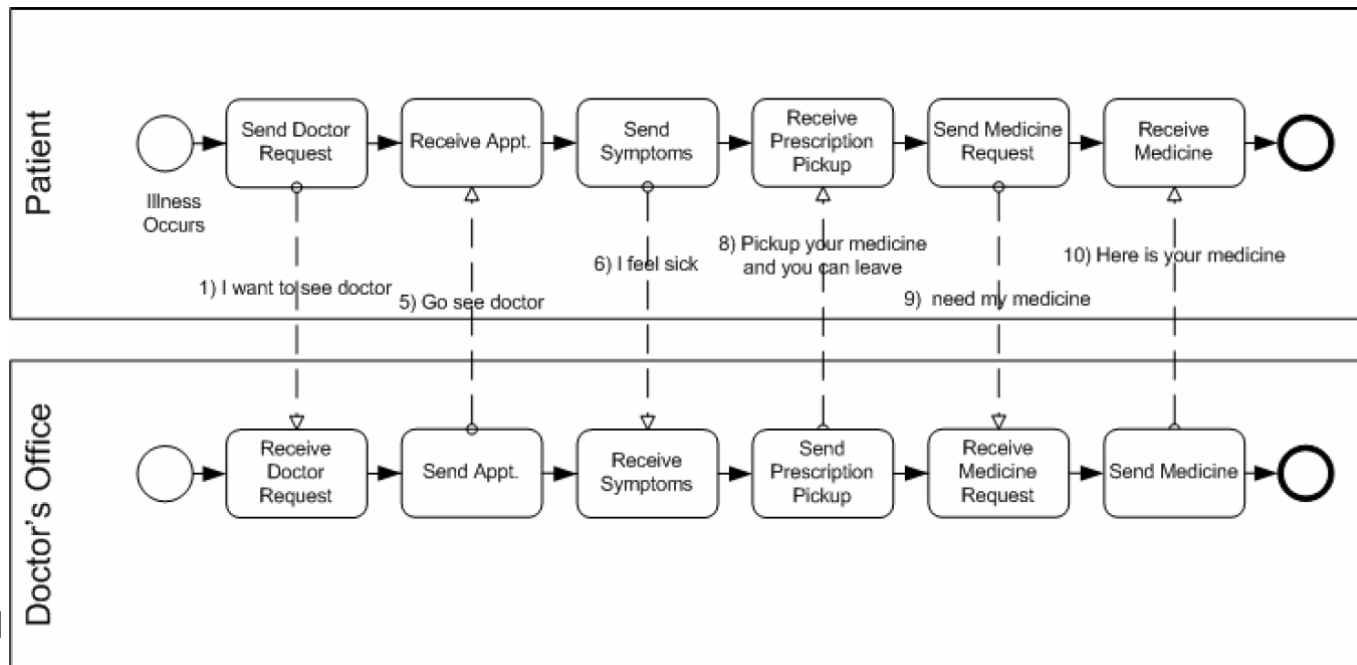
- Additional Information to a Process, the diagram becomes more readable
- An Artifact cannot be source or target of a sequence flow or a message flow
  - **Group:** used for documentation or analysis purposes, but does not affect the sequence flow.
  - **Annotation:** mechanism for a modeler to provide additional text information for the reader of a BPMN diagram





# Pools and Lanes

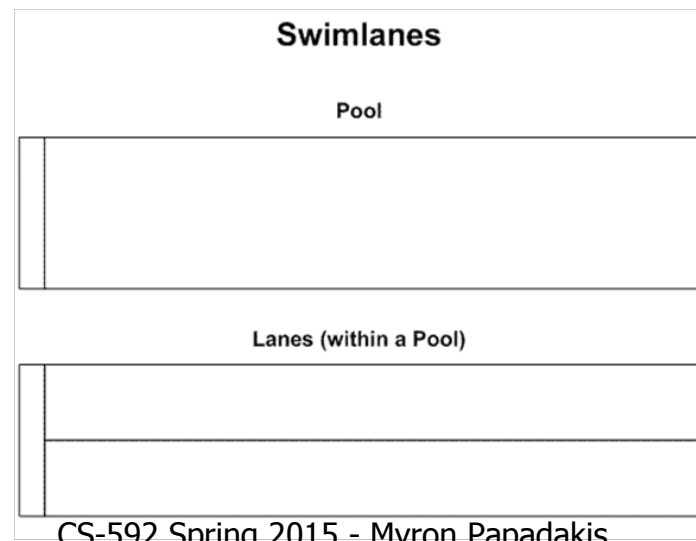
- Mechanism to organize activities into separate visual categories in order to illustrate different functional capabilities or responsibilities.
- Two types of BPD swimlanes: **pools** and **lanes**
- Pools
  - used when the diagram involves two separate business entities or participants
  - The activities within separate Pools are considered self-contained processes





# Pools and Lanes

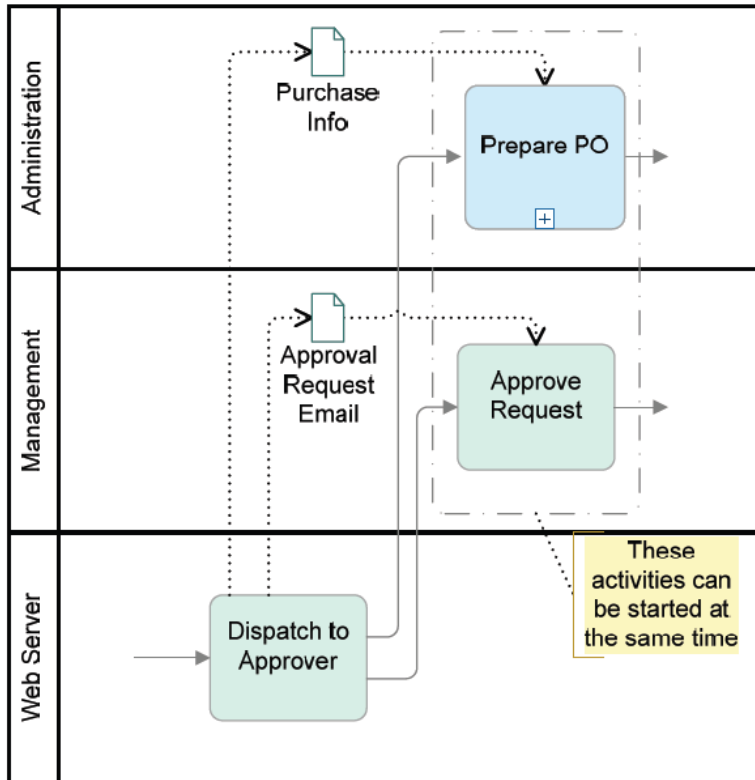
- Lanes are often used to separate the activities associated with a specific company function or role.
  - **Sequence flow may cross the boundaries of Lanes within a Pool but Message Flow may not be used between Flow Objects in Lanes of the same Pool.**
- Sub-partition within a Process (often within Pools)
  - Organize and categorize Activities.
  - Represent **internal** roles, systems, internal department
  - Nesting is allowed





# Pools and Lanes

- Lanes create sub-partitions for the objects within a Pool.
- These partitions are used to group Process elements (showing how they are related), or which **roles** have responsibility for carrying out the Activities.
- Lanes often represent organization roles (e.g., Manager, Administration, Associate, etc), but can represent any desired classification (e.g., underlying technology, organizational departments, company products, etc).
- Sequence Flow can cross Lane boundaries. Message Flow is not used within or across Lanes of a Pool.
- Lanes can be nested.





# Collaborations



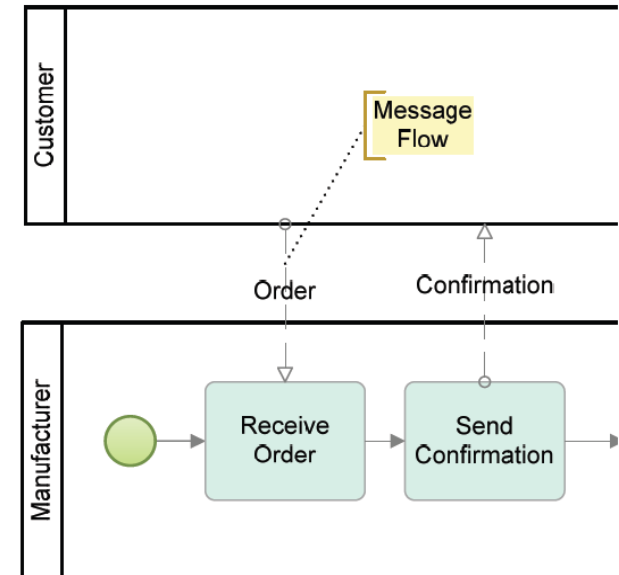
- Collection of Participants shown as **Pools**, their interactions (message flows)
- **May include processes within pools and/ or choreographies between pools**
  - Each pool contains a separate process
  - A pool is the graphical representation of a Participant in a Collaboration
  - A pool does not need to represent a company/organization (can be role or system)
  - **A pool may be empty – a “black box”**
- Collaboration diagrams: **any combination of Pools, Processes and Choreographies**
- **A collaboration diagram is like a process diagram but with more than one participant**
  - If a diagram has a single pool and no lanes it is a process diagram and the pool boundary rectangle can be omitted



# Collaboration Diagrams > Message Flow

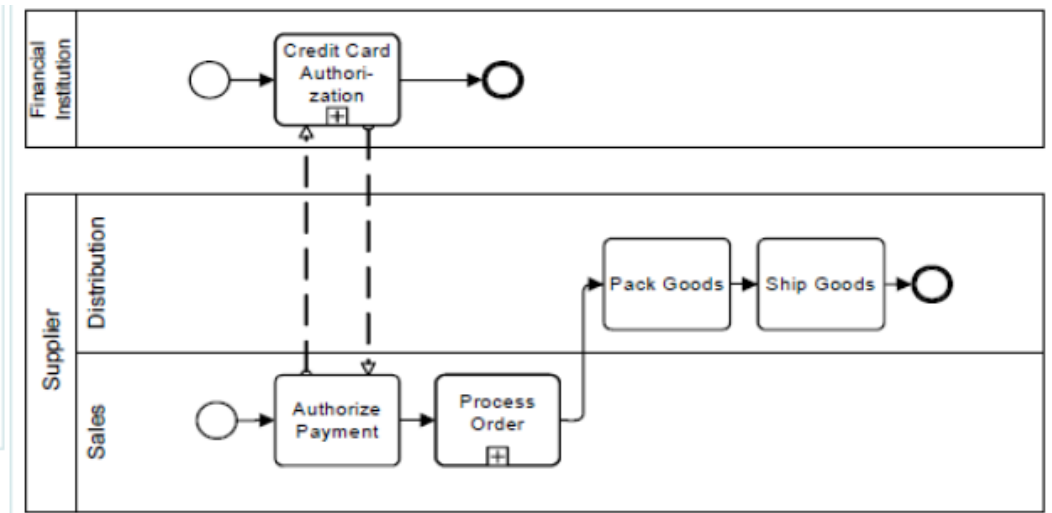
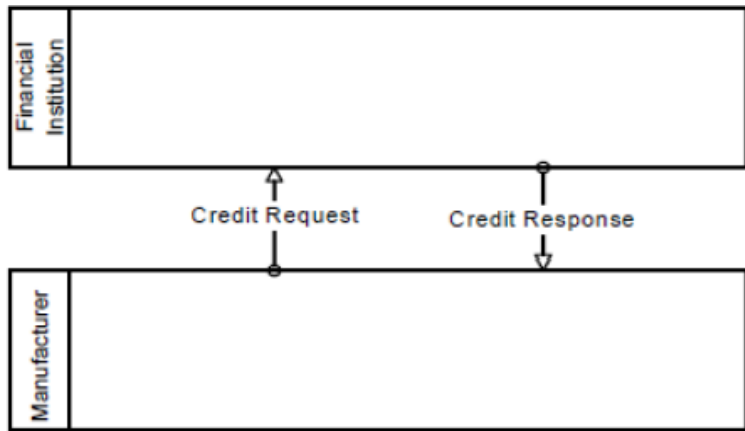
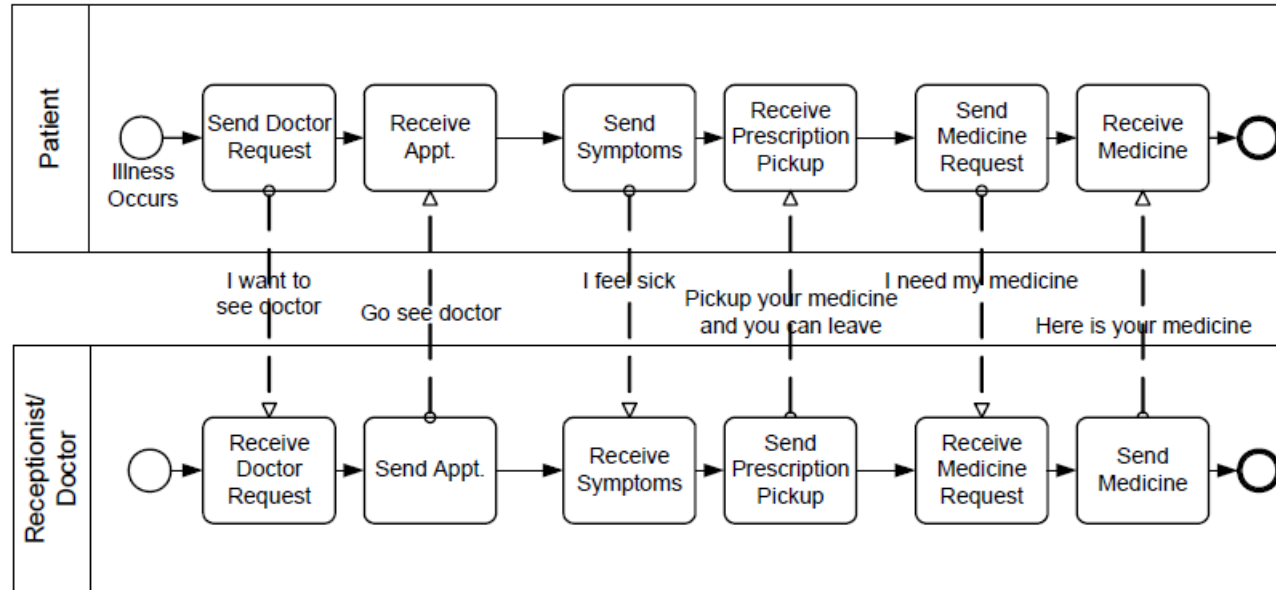


- Message Flow defines the messages/communications between two separate participants (shown as Pools) of the diagram.
- **Message Flow must always occur between two separate Pools and cannot connect two objects within a single Pool.**
- Message Flow is only used in collaborations (diagrams with two or more Pools).
- Where a Pool has Process elements, the Message Flow connects to those elements
- **Sequence Flow cannot cross a Pool boundary** - i.e., a Process is fully contained within a Pool.





# Collaboration Diagrams > Examples





# Conversation Diagrams








- Conversation diagrams are **simplified versions and informal descriptions of Collaboration diagrams** (but do maintain all of their features)
- **Model conversations between participants (pools)**
  - Conversations allow a modeler to group collaboration interactions between two or more participants, which together achieve a common goal
- A Conversation is a **logical grouping of Message exchanges** (Message Flows) that can **share a Correlation** (topic).
  - the logical relation often concerns a **business object**, e.g., “order”, “shipment and delivery”, etc.
  - **correlation**: a method to identify the Process instance which a message should be routed.
  - Processes can appear within the Pools, to show how Conversation and Activities are related
- **A Conversation is associated with a set of name-value pairs, or a Correlation Key** (e.g., “Order Identifier,” “Delivery Identifier”), which is recorded in the Messages that are exchanged.





# Conversation Diagrams > Elements

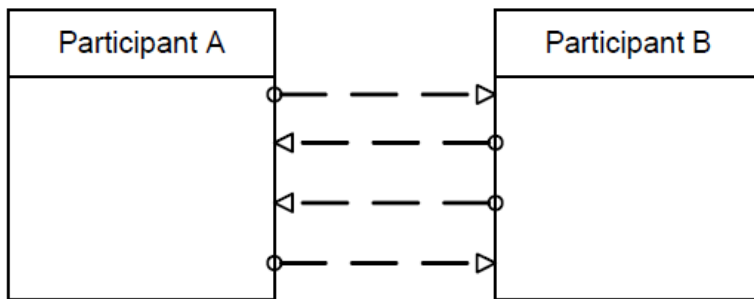


<b>Communication element/Conversation element</b>	Represents a set of Message Flows grouped together based on a concept and/or a CorrelationKey. Involves two or more Participants.	
<b>Sub-conversation (compound conversation element)</b>	Hierarchical view of a diagram. Consists of Message flows, Conversations and/or Sub-Conversations. Shares the Participants of its parent diagram	
<b>Call Conversation (thick line)</b>	A Call Conversation calling a GlobalConversation. A GlobalConversation is a reusable, atomic Conversation definition that can be called from within any Collaboration by a Call Conversation (for re-use).	
	A Call Conversation calling a Collaboration (similar to sub-conversation but has a thick border)	
<b>Conversation Link</b>	Used to connect conversation nodes to and from Participants. Conversation links to Activities, Pools and Events Conversation links for Call Conversation shows the names of the Participants	  

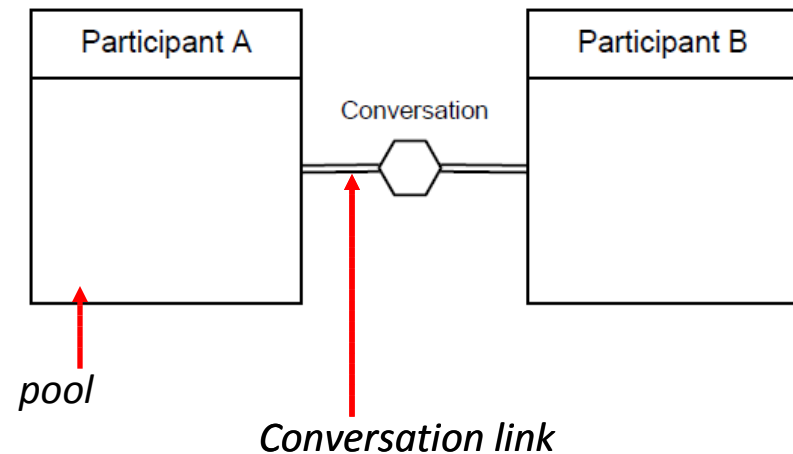


# Conversation Diagrams

- Additional Elements:
  - conversation node elements (conversation, sub-conversation and call Conversation)
  - conversation Link

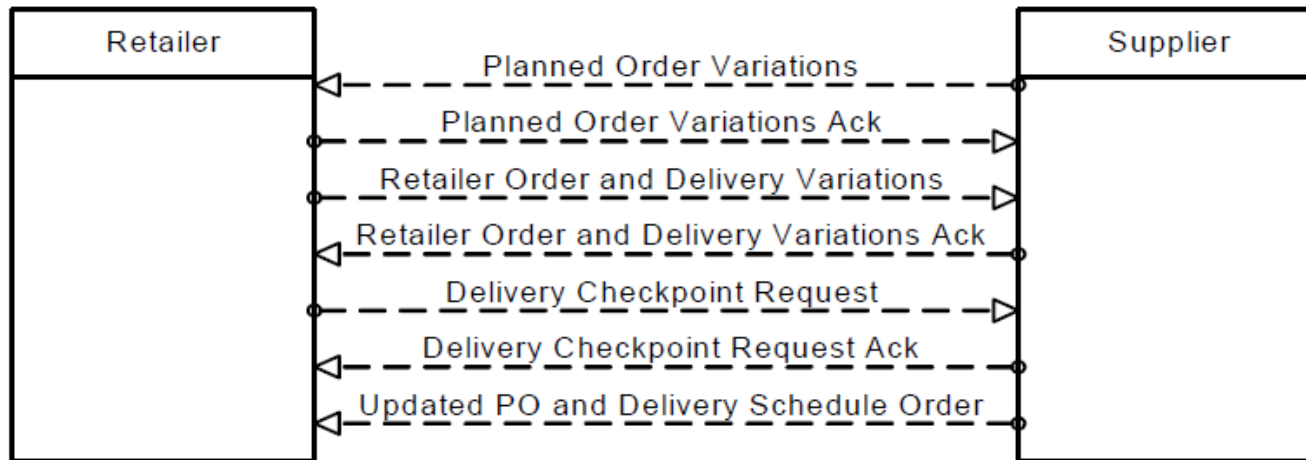
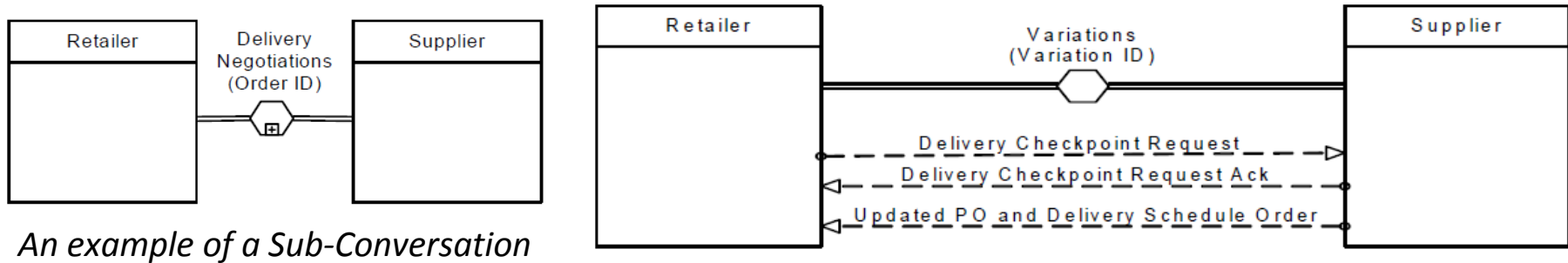


A Conversation diagram where the Conversation is expanded into Message Flows





# Conversation Diagrams > Sub-Conversation

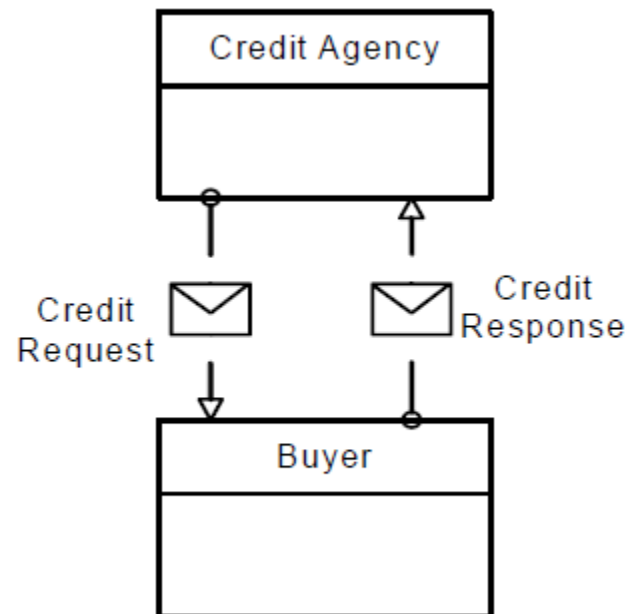
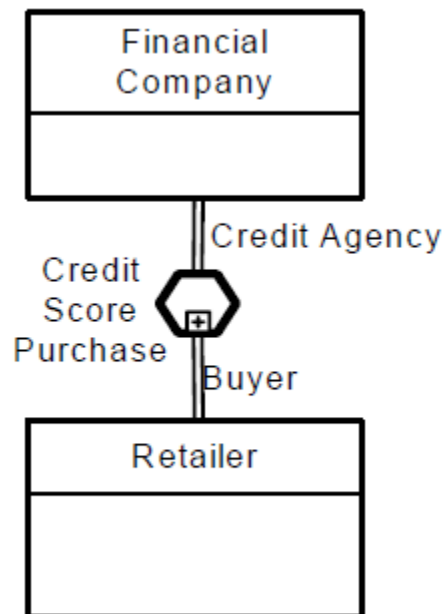




## Conversation Diagrams > Call Conversation



- Calling a **collaboration**





# Choreography

- A choreography is a compact representation of a Collaboration that **focuses (defines) on the sequence of interactions** between participants and generally hides process details.
- **Helps to show who initiates the activity and the first message.**
- A choreography comprises choreography activities, events and gateways connected by sequence flows.
- Each choreography activity represents an interaction between two or more participants.
  - Pools are the graphical representation of participants
- A choreography diagram can be used **to expand and analyze in detail the exchange of messages associated with a *conversation* node in a *conversation* diagram.**
- All of the gateways can be used in choreographies
  - Some **constraints** as to how the gateways are used in conjunction with the choreography activities that precede and follow the gateways.



# Choreography

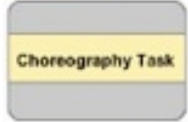

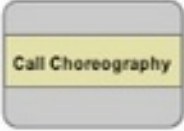
- The **Choreography** looks similar to a *private Business Process* since it consists of a network of **Activities, Events, and Gateways**
  - However, a **Choreography** is different in that the **Activities** are interactions that represent a set (1 or more) of **Message** exchanges, which involves two or more *Participants*.
  - In addition, unlike a normal **Process**, **there is no central controller**, responsible entity or observer of the **Process**.
- Choreography Data
  - **Neither Data Objects nor Repositories are used in Choreographies.**
    - Both of these elements are used exclusively in Processes and require the concept of a central locus of control.



# Choreography Activity Types

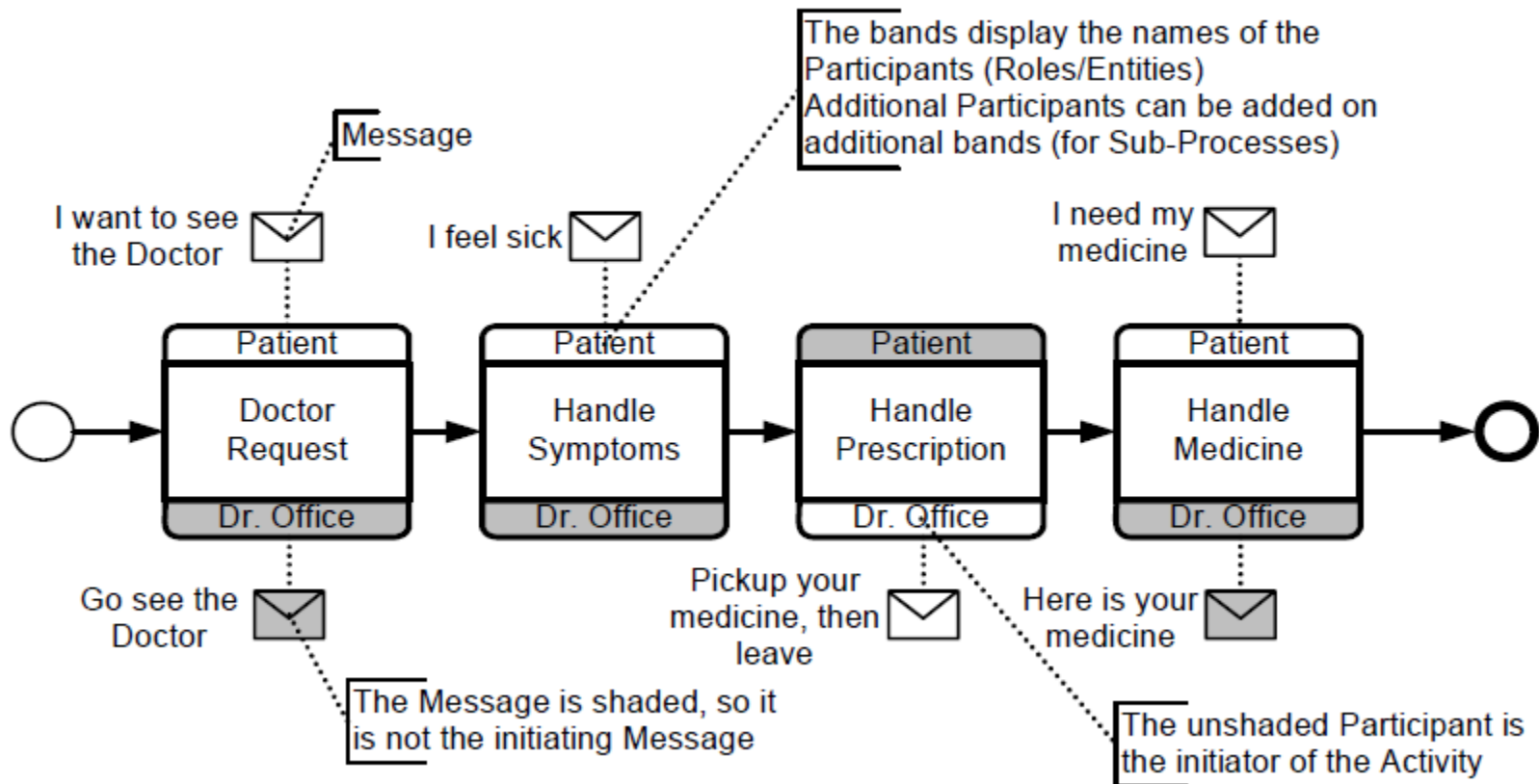


- A choreography activity represents a cohesive grouping of messages between 2 or more participants
- Choreography Activity Types

<b>Choreography Task</b> 	A cohesive grouping of messages sent between two or more participants that is atomic. The participant <b>bands</b> above and below the name are where participants are referenced.
<b>Sub-choreography</b> 	A cohesive grouping of messages sent between two or more participants that may be <b>hierarchically decomposed</b> into other choreography activities.
<b>Call choreography</b> 	A call to an already defined choreography task or sub-choreography. Uses task syntax with thick border to call tasks, and sub-choreography syntax with thick border to call sub-choreographies.



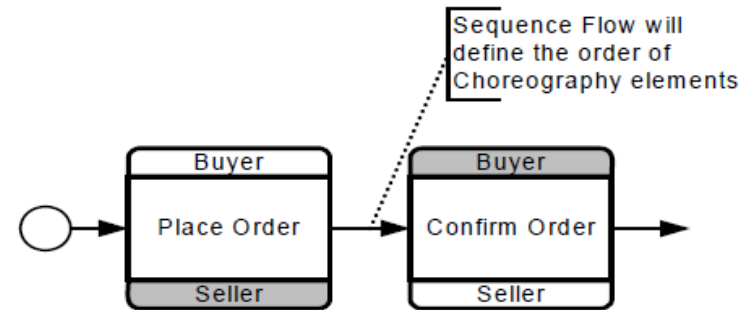
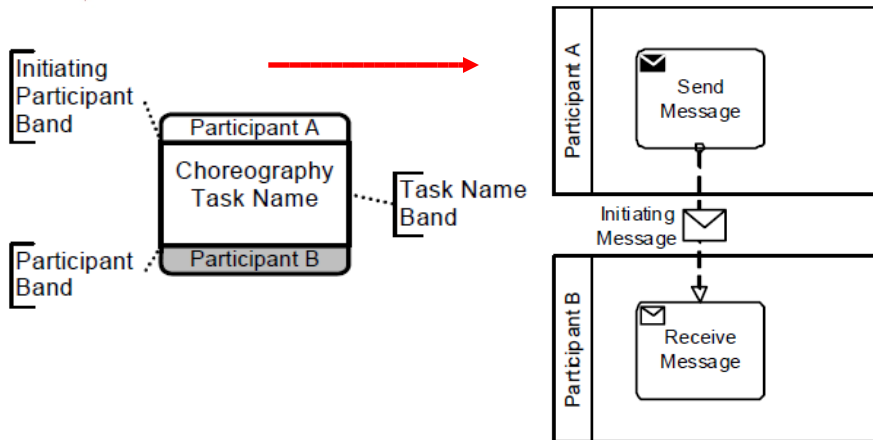
# Choreography Example



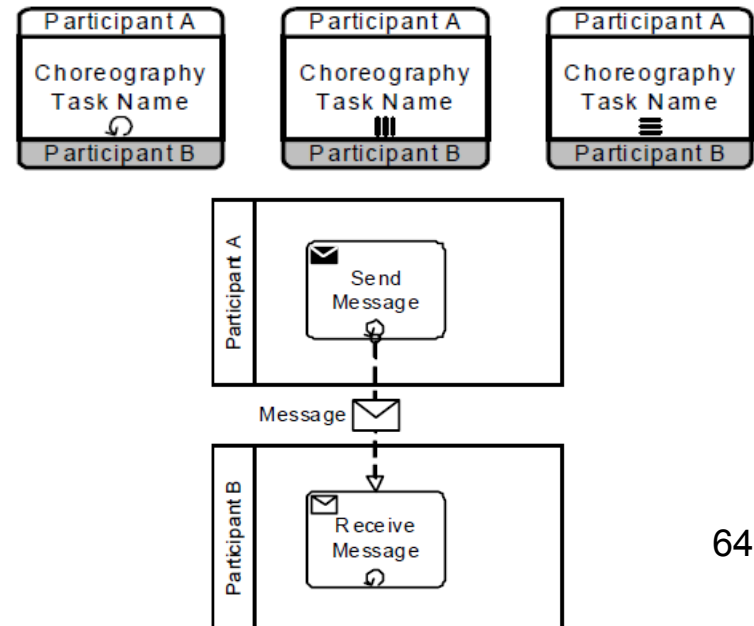
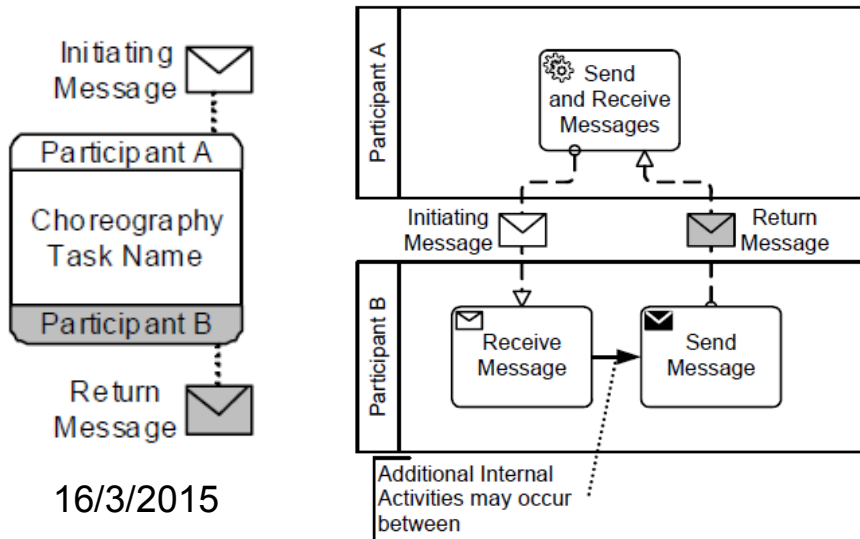




# Choreography Tasks

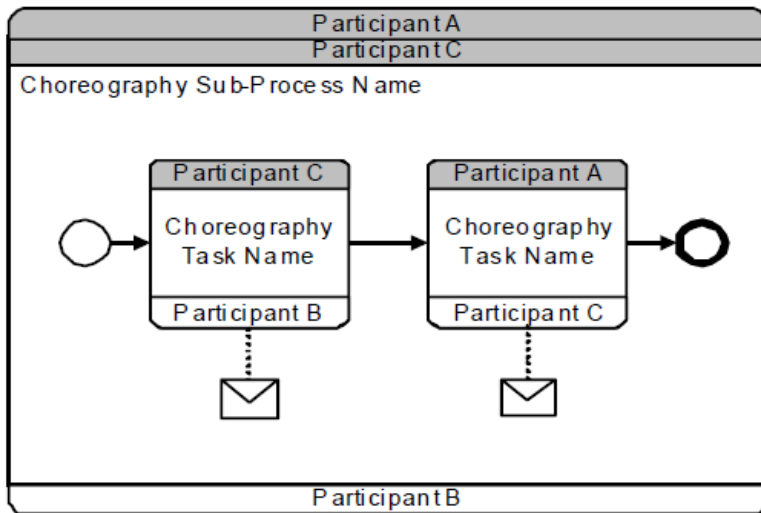


## Two-way choreography task

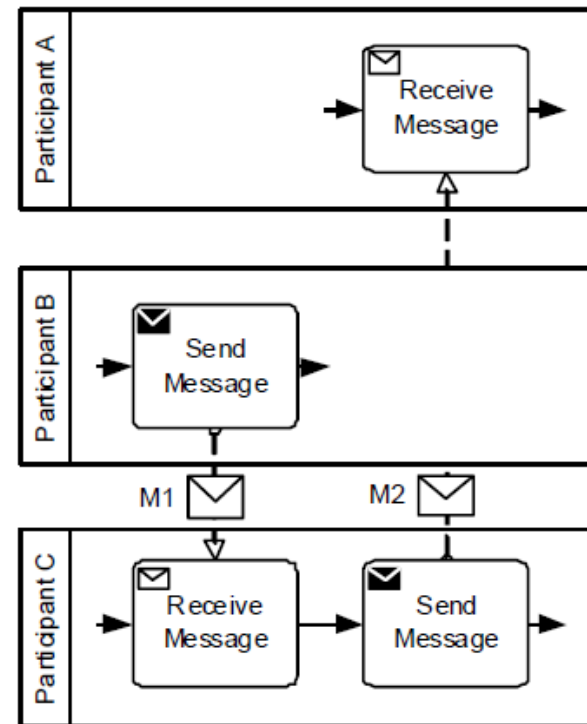




# Choreography > Sub-choreography



*Expanded sub-choreography*



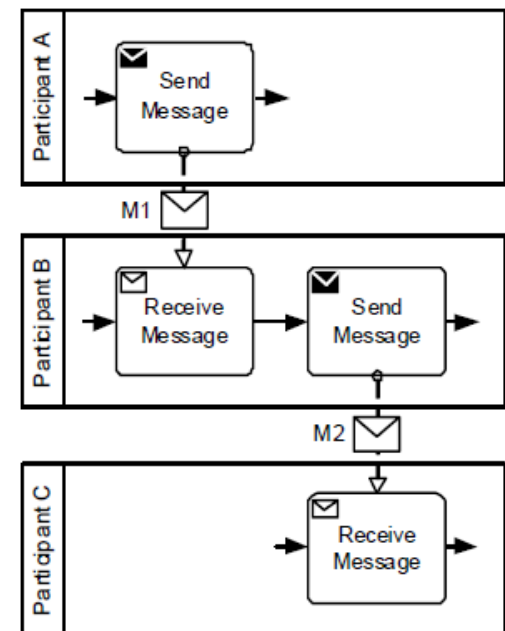
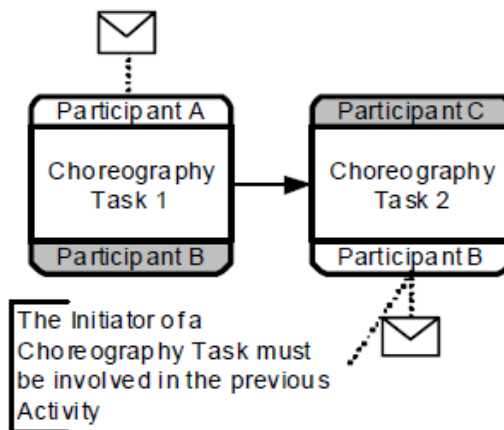
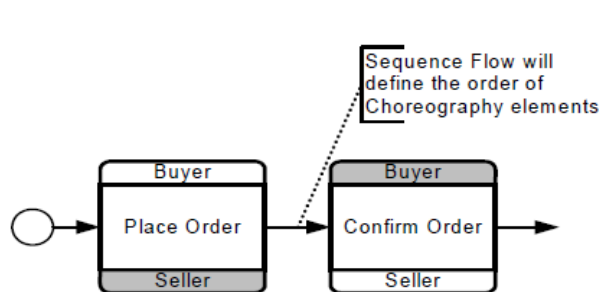
*A Collaboration view of an expanded Sub-Choreography*



# Sequencing of Choreography Activities



- Participants only know about the progress of the choreography by the Messages they occur. Thus, ordering of Choreography Activities need to take into account when the Participants should send/receive Messages
- **Basic rule of Choreography Activity sequencing :**
  - The Initiator of a Choreography Activity MUST have been involved (as Initiator or Receiver) in the previous Choreography Activity (except the 1<sup>st</sup> one).

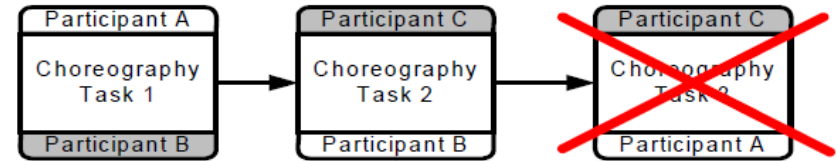
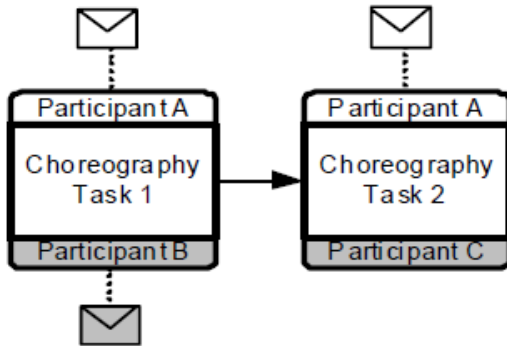




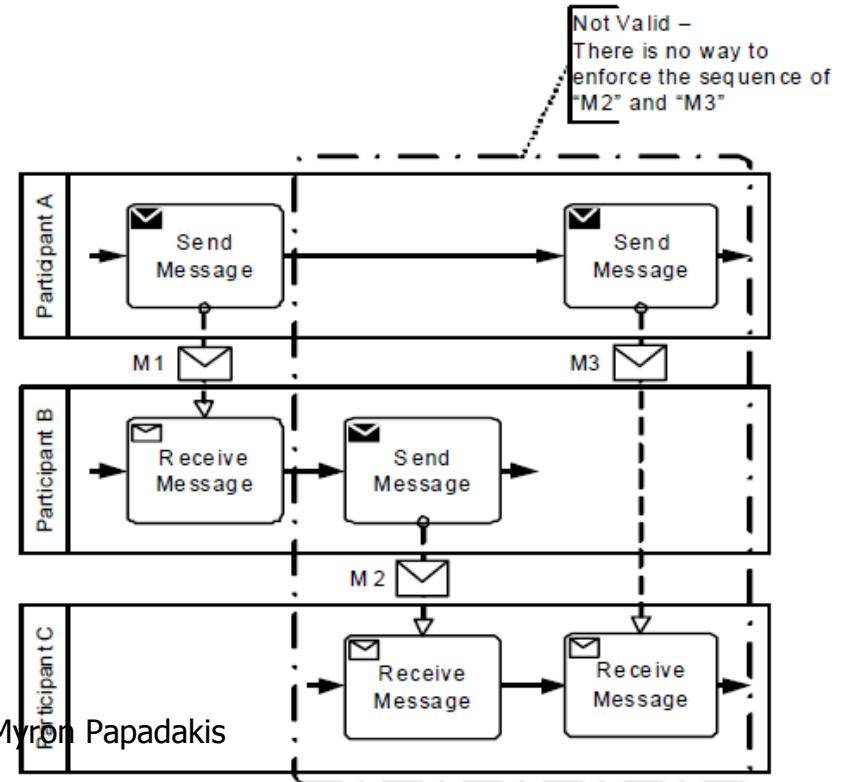
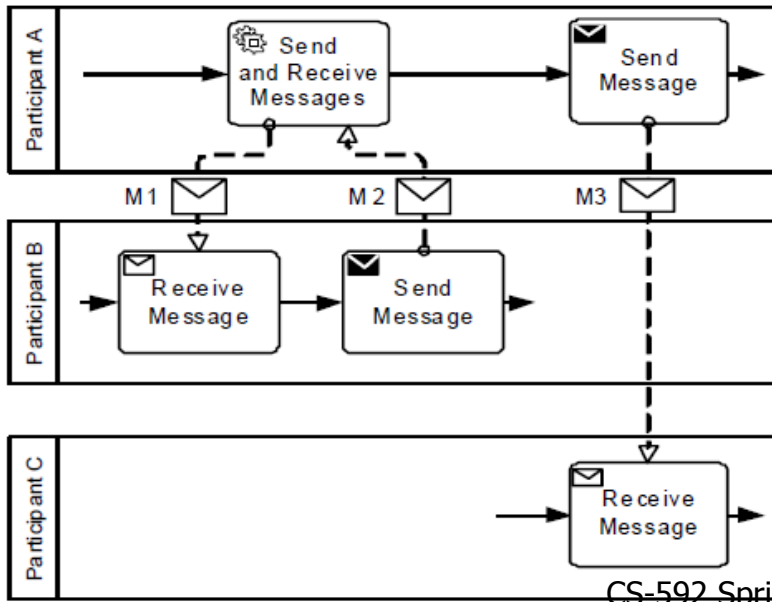
# Sequencing of Choreography Activities



*Valid*



The Initiator of a Choreography Task must be involved in the previous Activity





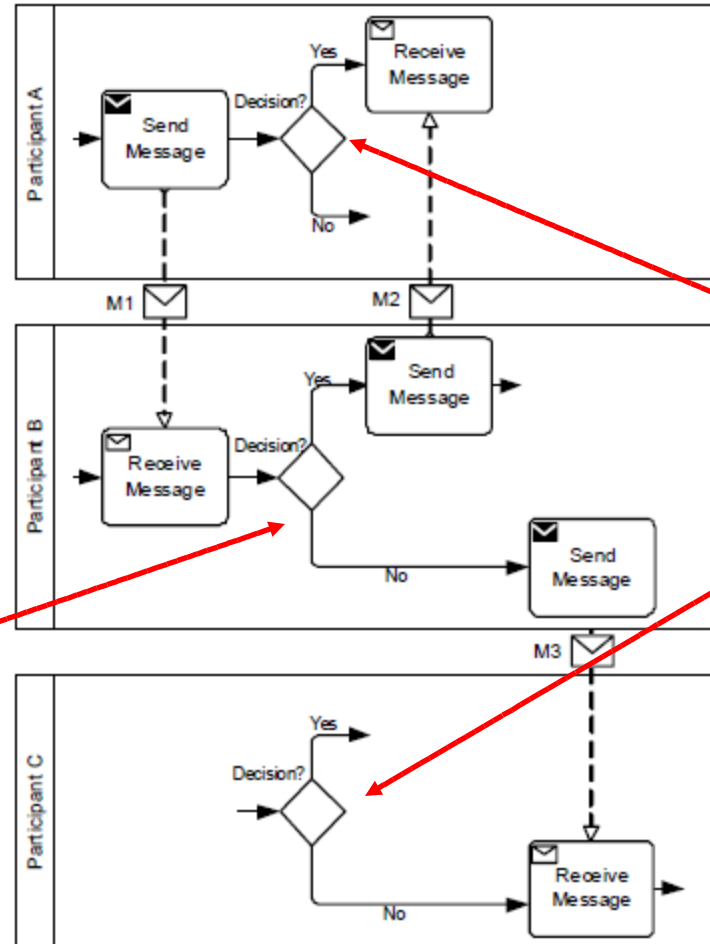
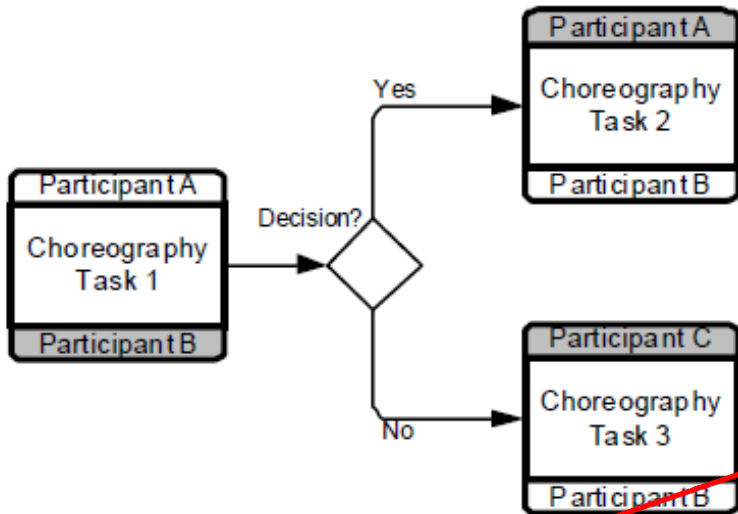
## Choreographies > Exclusive Gateway



- **Each choreography activity and the sequence flow is reflected in each participant process.**
- The gateway is reflected in the process of each participant process that is an initiator of choreography activities that follow the gateway.
- For the **receivers** in Choreography Activities that follow the Gateway, an **Event-Based Gateway** is used to consume the associated Message (sent as an outcome of the Gateway).
  - When a Participant is the receiver of more than one of the alternative Messages, the corresponding receives follow the Event-Based Gateway.
  - If the Participant is the **receiver** of only one such Message, that is also consumed through a receive following the **Event-Based Gateway**. This is because the Participant Process does not know whether it will receive a Message (since the Gateway entails a choice of outcomes).



# Choreographies > Exclusive Gateway Example



*Initiator – decision based*

*Event-Based (receivers)*



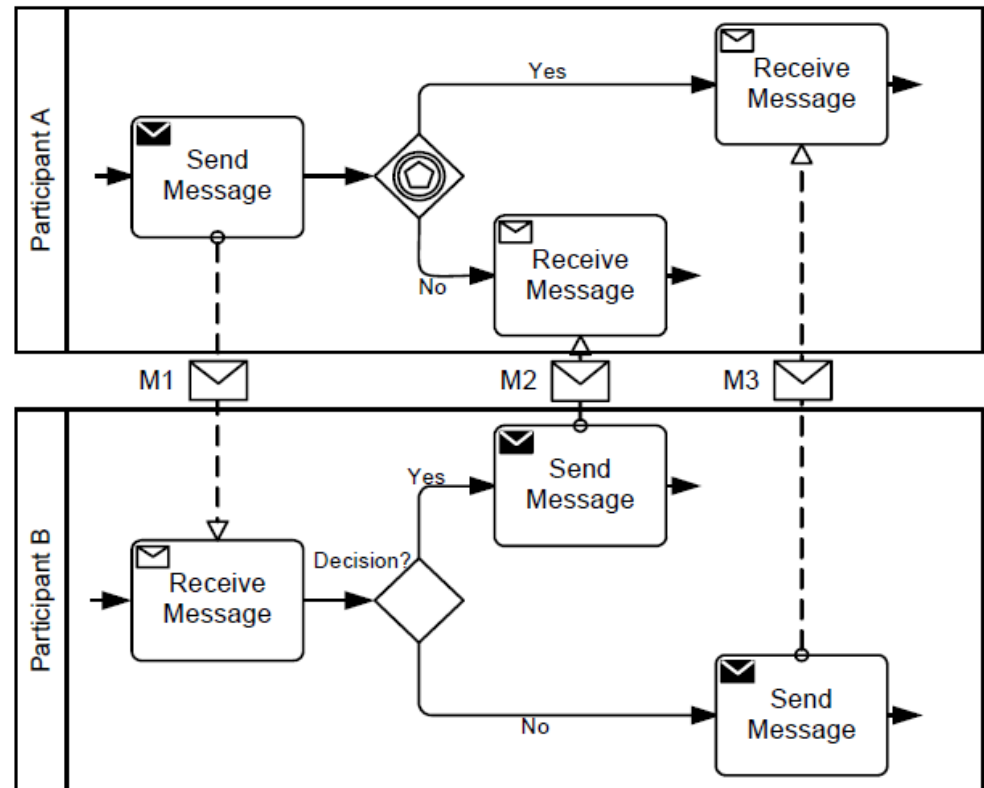
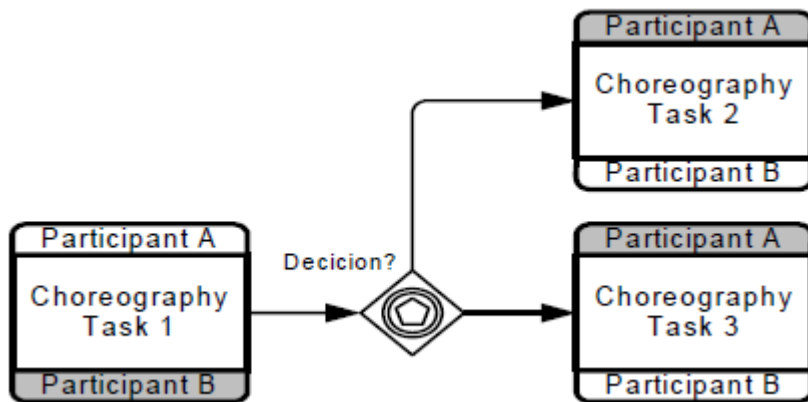
# Choreographies > Event-Based Gateway



- No message intermediate events, but signals may be used
- Timers allowed: participants on the right-side of gateway must be involved in the Choreography Activity that immediately precedes the gateway
- Constraints
  - Senders the same or receivers the same on the right-hand side
  - Each choreography activity and sequence flow connections are depicted in each Participant process
  - If senders following the gateway are the same => exclusive gateway in the participant's process (choice is determined by the participant only)
  - If senders are different=> sending occurs through different Processes
  - If receivers are the same=> event-based gateway to receiver's pool with different receives, (senders can be different or same)
  - If receivers are different=> the senders must be the same
  - The event-based gateway is reflected for different receiver Processes such that the respective receive follows the Gateway. Timeout for not waiting forever



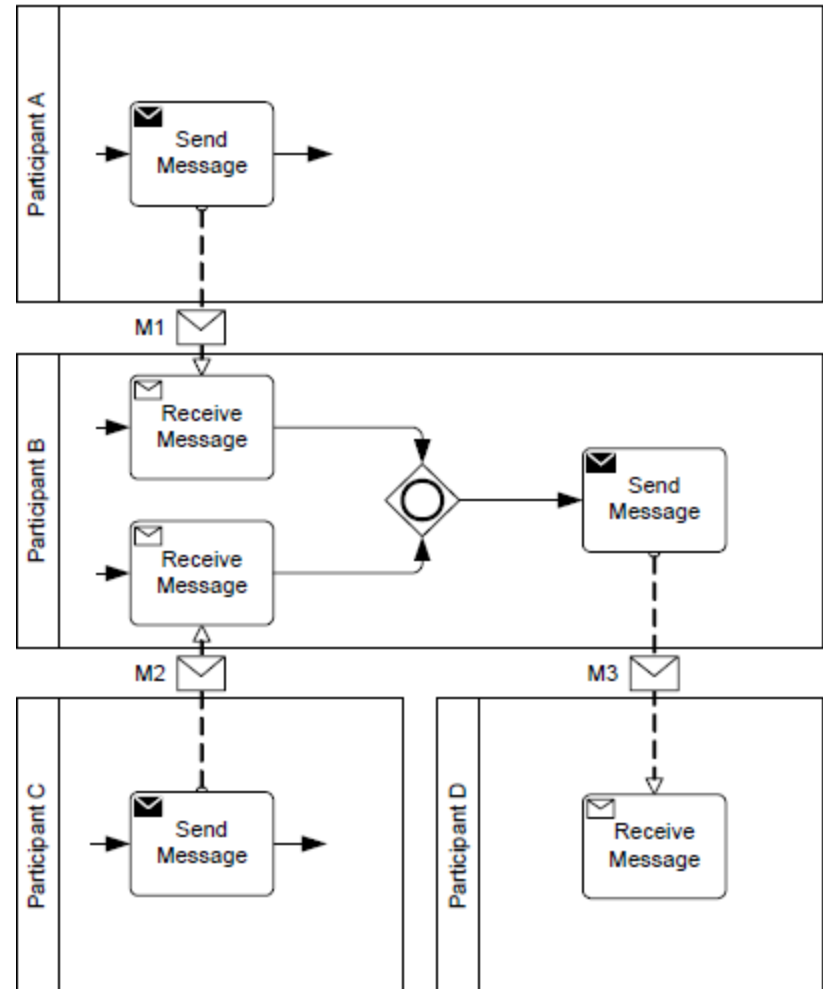
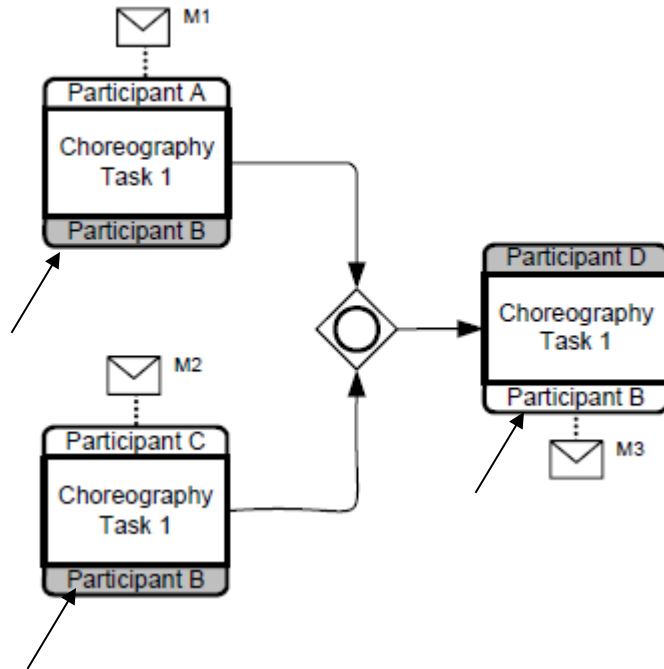
# Choreographies > Event-Based Gateway > Example







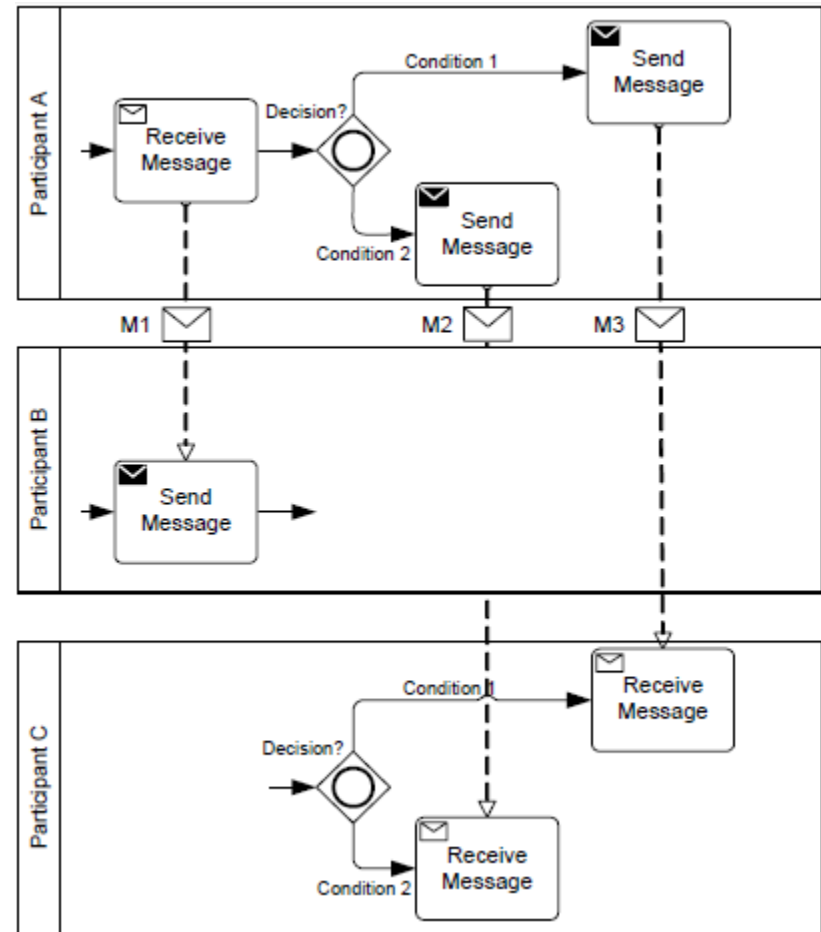
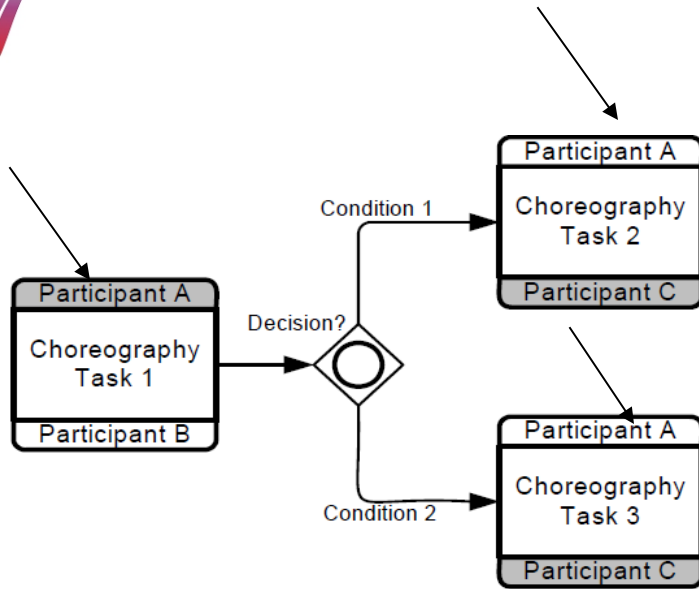
# Choreographies > Inclusive Gateway Example



**Enforcing Merge Behaviour:**  
*initiator of activities after gateway and receiver of the activity preceding gateway must be the same (Participant B)*



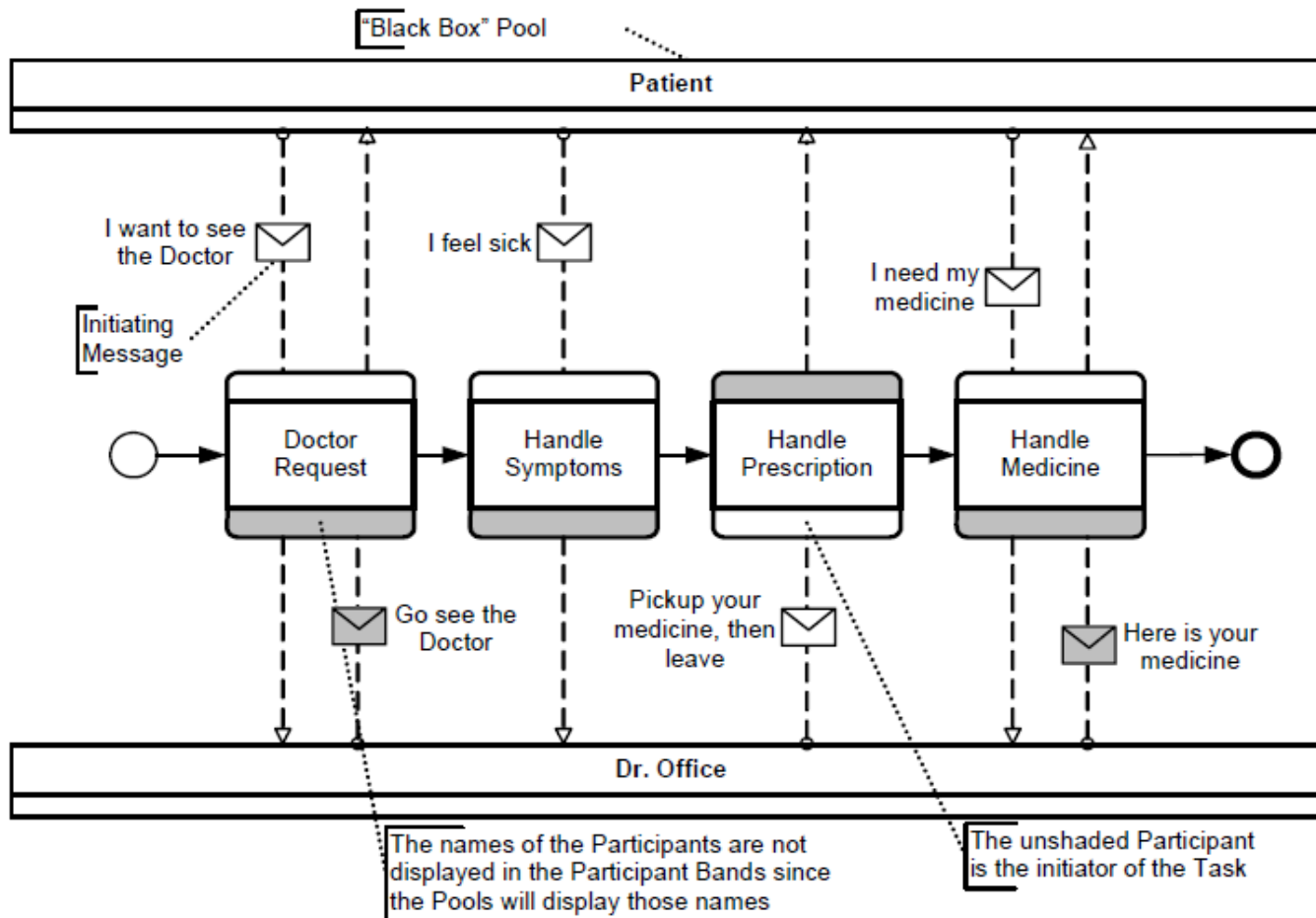
# Choreographies > Inclusive Gateway Example



***Enforcing split behaviour:*** Initiators of the activities after gateway must be the same as the receiver of activities preceding the Gateway. (Participant A)

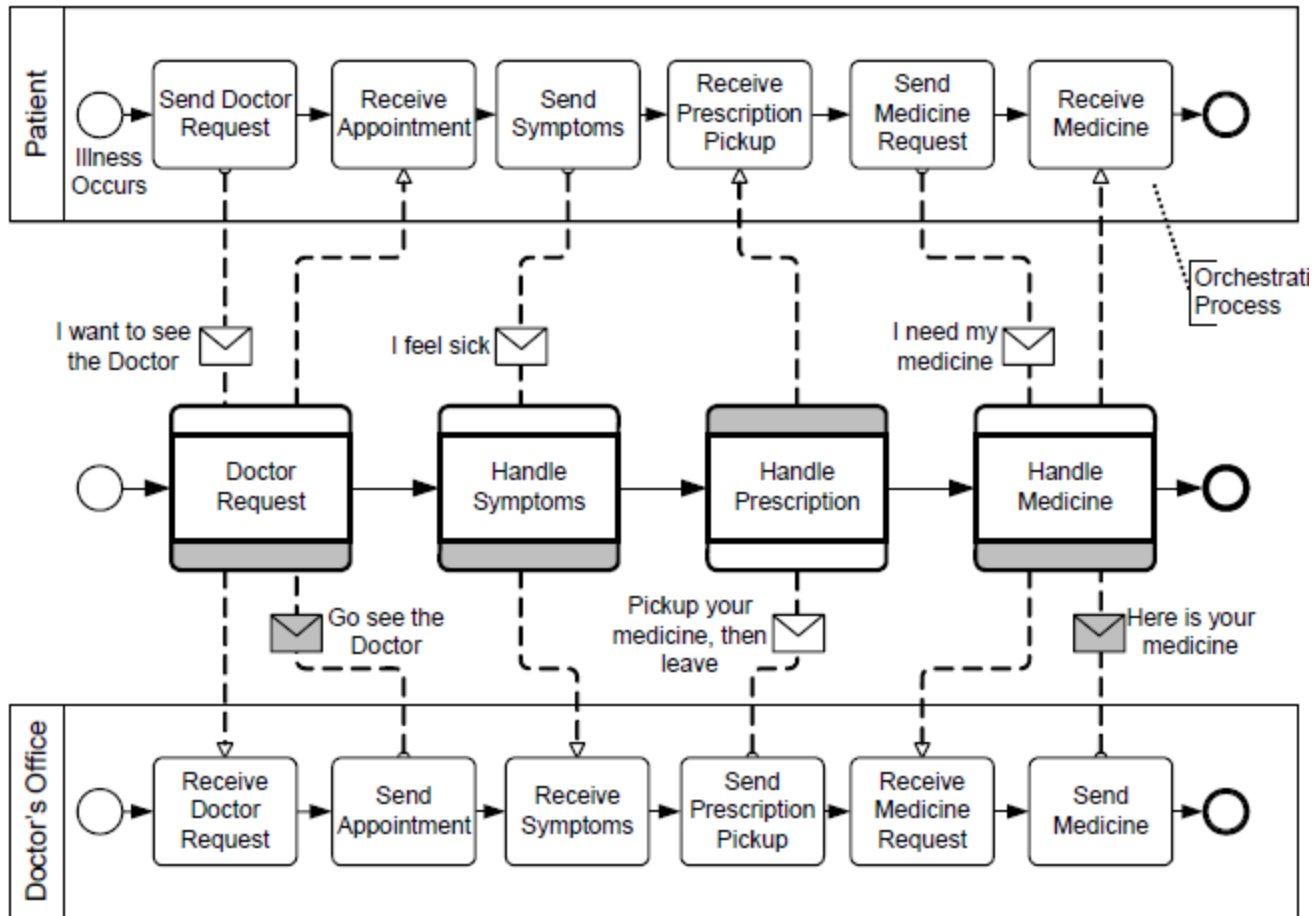


# Choreography within Collaboration





# Choreography within Collaboration



*An example of a Choreography Process combined with Pools that contain Processes*



# BPMN2 Tools

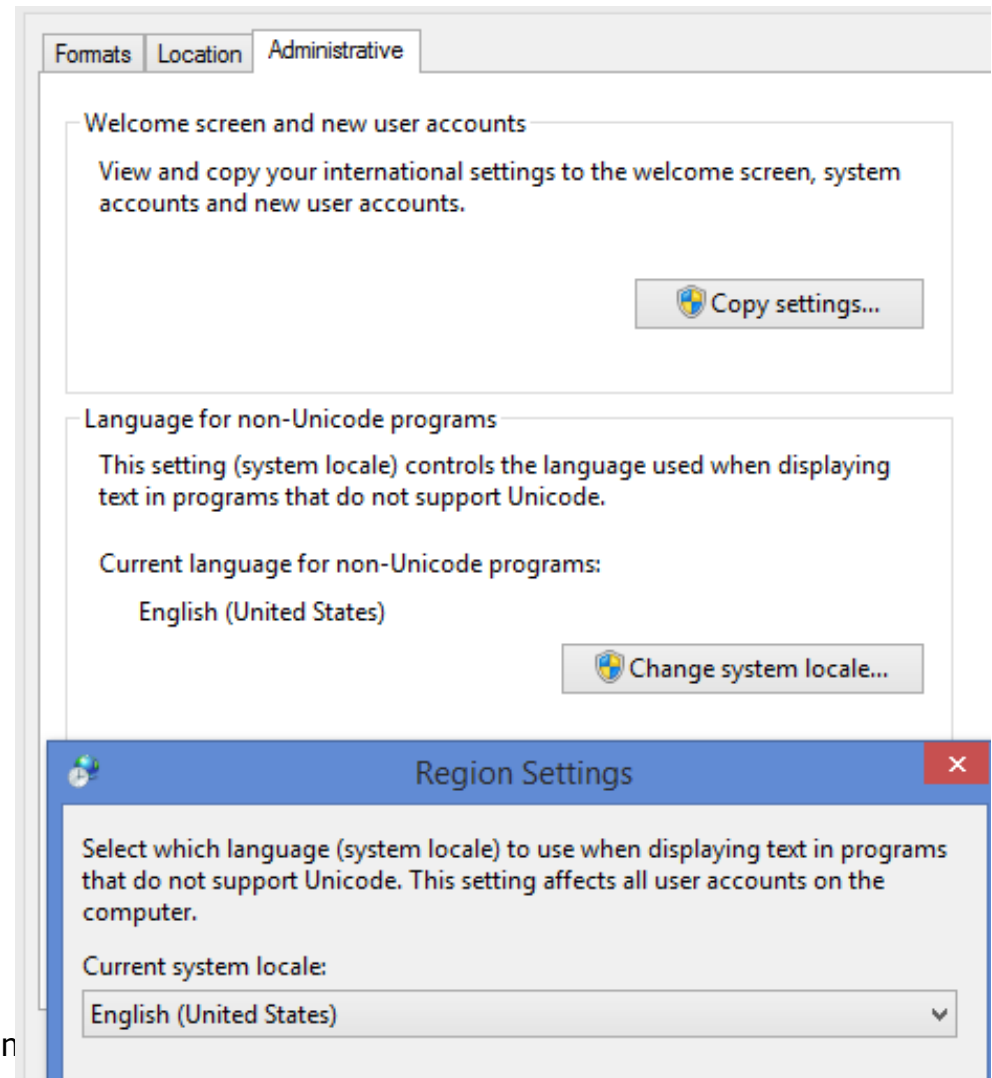
- ADONIS Community Edition: <http://www.adonis-community.com/>
  - You have to register in order to download it.
  - After installing the tool, and when you create a new model, you have to check that "BPMN 2.0" is selected in the "Model Type".  
Each time, check from the toolbar "View->Mode->BPMN 2.0 All modeling objects" is selected.
- Yaoqiang Editor: <http://sourceforge.net/projects/bpmn/>
  - A simple BPMN editor for modeling business processes. It is compatible with BPMN 2.0.
  - This tool is sufficient for the purposes of the project because we are only focusing on modeling.



# Adonis Installation (Possible Issue)



- If Adonis cannot start...
- For Adonis Installation (mostly greek users)
- Control Panel > Region > Administrative > Language for non-Unicode Programs > Change System Locale to English
- Restart





# References

- BPMN 2.0 Tutorial, Daniel Brookshier
- <http://en.bpmn-community.org>
- [http://training-course-material.com/training/BPMN\\_2.0\\_Introduction](http://training-course-material.com/training/BPMN_2.0_Introduction)
- Several Slides from
  - <http://www.slideshare.net/jimarlow/introductiontobpmn005> (Jim Arlow)
  - [http://www.dis.uniroma1.it/~marrella/slides/Sem\\_PM\\_11-12\\_BPMN.pdf](http://www.dis.uniroma1.it/~marrella/slides/Sem_PM_11-12_BPMN.pdf)
  - <http://www.bpmn.org/>
  - <http://www.processmodeling.info/posts/highlights-from-bpmn-2-0-activity-types/>

# Τέλος Ενότητας



Ευρωπαϊκή Ένωση  
Ευρωπαϊκό Κοινωνικό Ταμείο



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης





# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «**Εκπαίδευση και Δια Βίου Μάθηση**» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



**Σημειώματα**

# Σημείωμα αδειοδότησης

- Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά Δημιουργού - Μη Εμπορική Χρήση - Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-nd/4.0/>

## •Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

•Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

# Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Κρήτης, Μύρων Παπαδάκης. «**Εισαγωγή στα Δίκτυα Υπηρεσιών. Διάλεξη 6η: Assisting Lecture 3 - BPMN v2.0 standard**».  
Έκδοση: 1.0. Ηράκλειο/Ρέθυμνο 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:  
<https://elearn.uoc.gr/course/view.php?id=416/>