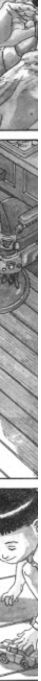
Modellazione di Processi Amministrativi e Compliance Normativa

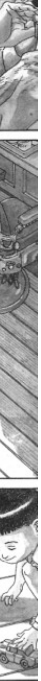
8a.BPMN - descriptive modeling

Matteo Baldoni

A.Y. 2022/2023

Università degli Studi di Torino - Dipartimento di Informatica

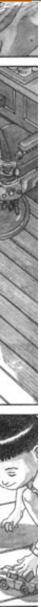




Si noti che

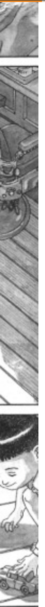
qesti lucidi sono basati su quelli di Marco Montali, KRDB Research Centre for Knowledge and Data Faculty of Computer Science, Free University of Bozen-Bolzano e su quelli di Andrea Marrella.

Resources



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

Business Process Model and Notation

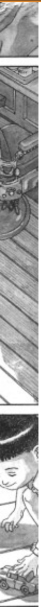


OMG standardization initiative.

Charter (`http://www.bpmn.org`)

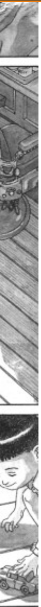
A standard Business Process Modeling Notation (BPMN) will provide businesses with the capability of understanding their internal business procedures in a graphical notation and will give organizations the ability to communicate these procedures in a standard manner. Furthermore, the graphical notation will facilitate the understanding of the performance collaborations and business transactions between the organizations. This will ensure that businesses will understand themselves and participants in their business and will enable organizations to adjust to new internal and B2B business circumstances quickly.

What BPMN 2.0 is About



- BPMN 2.0 as a single specification for notation, metamodel and interchange format.
- Enabling the exchange of BPs and their diagram layouts among process modeling tools to preserve semantic integrity.
- Support for model orchestrations and choreographies as stand-alone or integrated models.
- Support to the display and interchange of different perspectives on a model that allow a user to focus on specific concerns (internal, public, conversation, choreography).
- Provide an execution semantics via translation to executable WS-BPEL processes.
    - Achieved with strong assumptions on the shape of the acceptable BPMN models.

The Paradox of BPMN



Outward familiarity

BPMN sounds very familiary to business people, due to its similarity with flowcharts.

However. . .

Difference #1: BPMN is a formal specification

Has a metamodel and rules of usage. BPMN models can be validated.

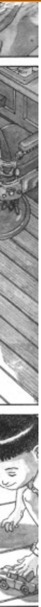
Difference #2: BPMN describes event-triggered behavior

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

Difference #3: BPMN captures business collaborations

Tackles how different process communicate with each other.

What BPMN is



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

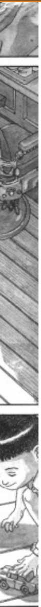
Process Logic

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

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

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

What BPMN is

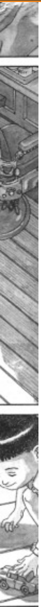


Main points/issues:

- Well-identified notion of case.
- Many-to-many relations are not supported and require to split the overall process into multiple processes.
- Repetitive behaviors that do not require ad-hoc changes.
- Focus on making the process control-flow explicit: BPMN reveals the allowed *order* of activities and *when* they happen. Nothing is said about *where* or *why*.
  - Atomic tasks are not internally specified: BPMN tackles the *process logic*, not the *task logic*.
  - Connection with data/resources quite weak.

Important

This applies to virtually all contemporary process modeling languages.



Uses of BPMN

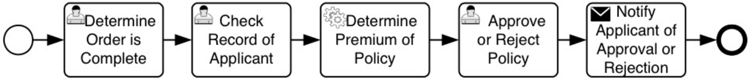
Several coexisting modeling paradigms.

- Process (or orchestration): intra-organizational perspective.
  - Private, non-executable: intra-organizational, for documentation purposes (abstract).
  - Private, executable: intra-organizational, with fully specified information to enable executability (concrete languages for conditions, loops, choices, . . .).
  - Public: interaction between a private BP and an external one. Only the internal activities involved in the interaction are shown.
- Collaboration: interaction between two or more business entities.
  - Multiple private processes with message exchange.
  - Choreography: contract (expected behavior) between interacting participants.
    - No central orchestrator.
    - Similar to a process, but each activity represents a message exchange.
  - Conversation: logical relation implied by message exchange.
    - Focus on business artifacts.
    - Elicitation of participants.
    - Message exchange used by participants to manipulate artifacts.

We will focus on private, public, collaborative abstract processes.

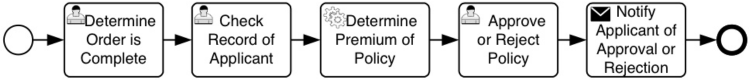
Examples - Control-Flow

Private process.

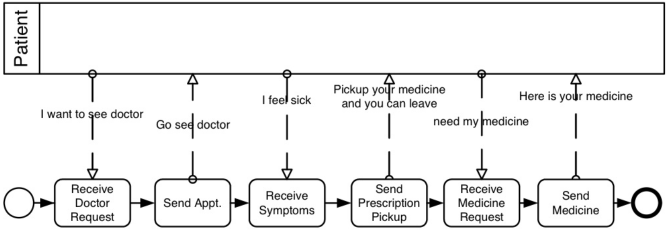

Determine Order is Complete
Check Record of Applicant
Determine Premium of Policy
Approve or Reject Policy
Notify Applicant of Approval or Rejection

Examples - Control-Flow

Private process.

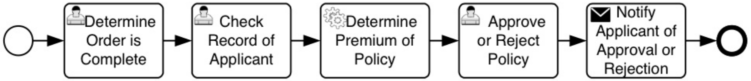

Determine Order is Complete
Check Record of Applicant
Determine Premium of Policy
Approve or Reject Policy
Notify Applicant of Approval or Rejection

Public process.



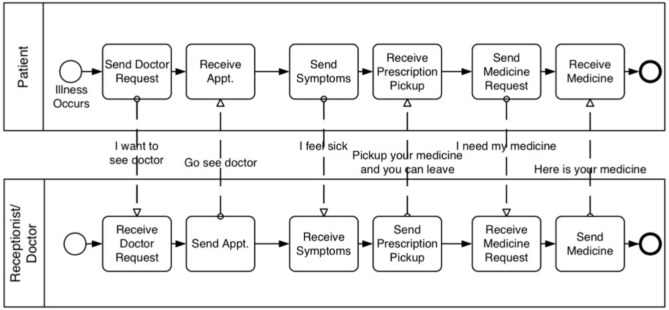
Examples - Control-Flow

Private process.

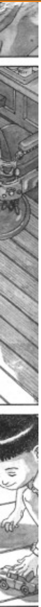


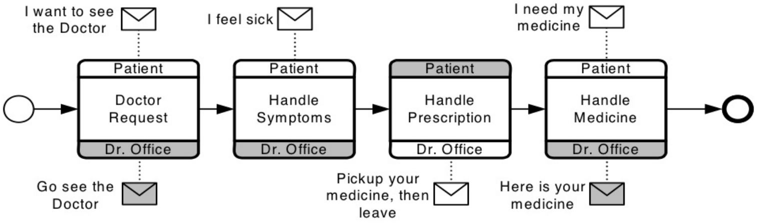
Determine Order is Complete
Check Record of Applicant
Determine Premium of Policy
Approve or Reject Policy
Notify Applicant of Approval or Rejection

Collaborative process.

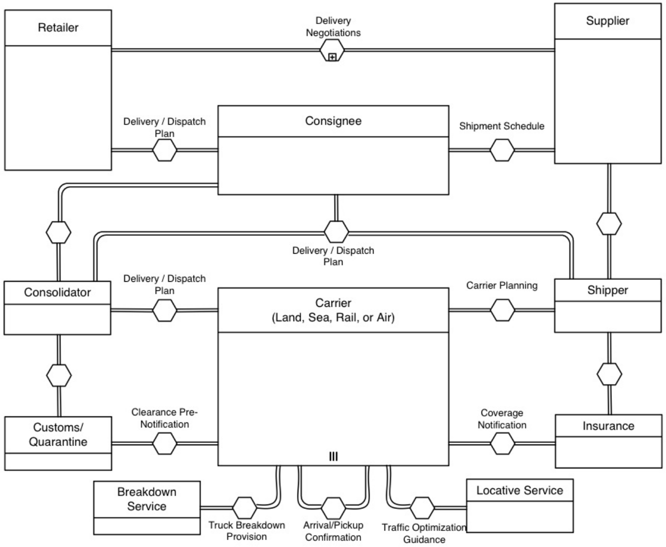


Example - Choreography

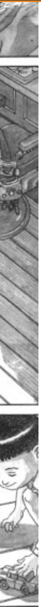


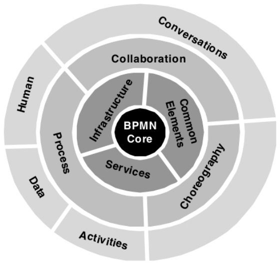


Example - Conversation



Core Structure



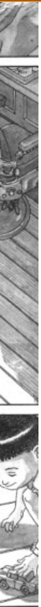


Every component of the structure:

- Is associated to a graphical notation.
- Is associated to a well-defined metamodel, capturing also the relationships with other components.
- Is associated to an XSD that corresponds to the metamodel and is used for validation, storage and interchange.

Metamodel and XSD: `http://www.bpmn.org`.

What BPMN 2.0 is Not



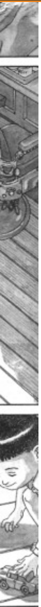
Method

Set of modeling guidelines that can be followed to produce BPMN diagrams that are not only syntactically correct, but are also effective and of high quality.

Style

Basic principles and best practices of usage and composition of BPMN elements to create “good” processes.
Can be validated as additional syntactic rules!

Why Method and Syle Are Important



BPMN: only ensures syntactic correctness, i.e., that the modeling rules written in the official documentation are respected.

With method and syle, new conventions have to be considered, towards:

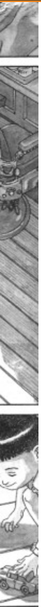
- Clarity $\leadsto$ the BPMN process is unambiguous and self-explanatory.
- Completeness $\leadsto$ all details have been properly considered.
- Structural consistency $\leadsto$ different modelers come up with similar solutions to the same problem, facilitating understanding and communication.
- Shareability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

This impacts the overall conception of the process.

Example

BPMN is a "model" and "notation". Many semantic aspects are embedded into the underlying XML. But humans just see the graphical front-end. Thus, e.g., labels have a fundamental importance, and conventions on how to put labels have a profound impact on modeling.

BPMN Graphical Elements



Strategy:

- elements grouped into 5 families;
- each family organized in two strata - basic and advanced elements.

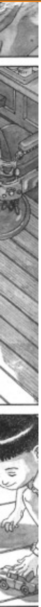
Families:

- Flow objects: behavior of the BP.
- Data: manipulated information.
- Connecting objects: connection between flow objects and other elements.
- Swimlanes: organizational grouping of modeling elements.
- Artifacts: additional infos.

Warning

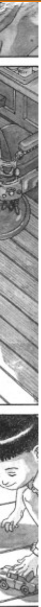
The second strata is very rich!

The Main Basic Elements



▢	Activity.
○	Event.
→	Flow.
◇	Gateway.

Modeling Levels



Top-down, through 3 different levels:

1. Descriptive process modeling. Traditional flowcharting for main processes, focusing on the internal orchestration of activities.
   Uses a minimal subset of the BPMN notation.
2. Analytic process modeling. Expansion of level 1 with reactive behaviors and exception handling.
   Uses an extensive palette of constructs and decorators to enrich the level 1 modeling.
3. Executable BPMN. Enrichment of the process model with all necessary details to enable process orchestration. Typically done at the XML level.

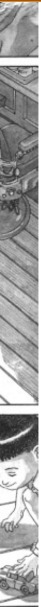
Levels 1-2: non-executable processes.

- Motto: *"if it is not in the diagram, it does not count"*.

Our strategy

We focus on level 1-2. For each level, we introduce the modeling constructs and the BPMN rules. Then we give methodological and stylistic guidelines.

Initial Questions



Contribute to the definition of the process logic.

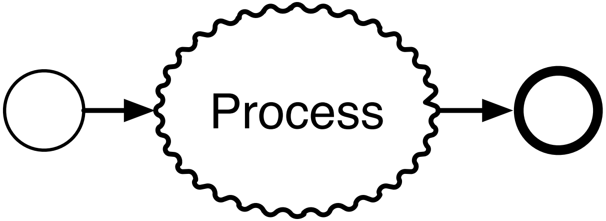
- What is the goal of the process? What is the main “object” evolved by the process? What do process instances represent? case
- How does the process actually start? In response to which event? Are there multiple ways of starting the process? start events
- When does the process end? Are there multiple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

Remember: the process logic is usually implicit. Our goal is to make it explicit!

Start, End, Move

General guidelines:

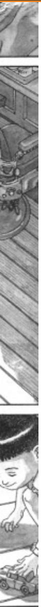
- A process starts because a certain event happens (**thin circle**).
- A process ends signalling this with a certain event (**thick circle**).
- A process flows through activities to move from the start event to the end event. Each move corresponds to a *sequence pattern* (**solid directed edge**).



Warning

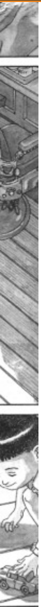
These general rules will be overridden.

Connecting Objects



———▶	Sequence flow.
o— — — —>	Message flow.
- - - - - - - - - ->	Directional association: link from/to a data item (data flow).
- - - - - - - - - - -	Connection with other artifacts (e.g., notes, textual annotations).

Activity



Basic building block of a BPMN process.

Activity

A **unit of work** performed in the process by a *performer*.

Each activity is either a task or a subprocess.

Task

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

Subprocess

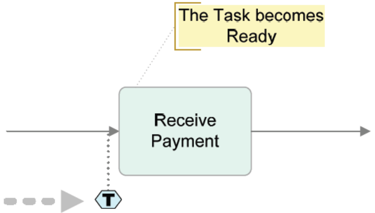
Compound unit of work, whose components can be described as a new, *child BPMN process*.

Token Game[1]

Token game

Provides an intuition of the execution semantics.

- Once a process instance has been created, we use the notion of token to identify the progress (or state) of that instance
- A token is a "theoretical object" used to create a descriptive "simulation" of the behavior associated to each BPMN element (it is not currently a formal part of the BPMN specification)
- A token is created in the start event, traverses the sequence flow and is destroyed in a end event. That is, there is no time associated with the token traveling down a sequence flow

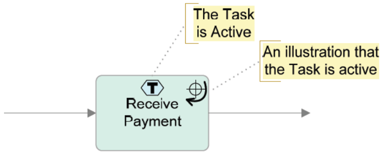


Token Game[1]

Token game

Provides an intuition of the execution semantics.

- Once a process instance has been created, we use the notion of token to identify the progress (or state) of that instance
- A token is a "theoretical object" used to create a descriptive "simulation" of the behavior associated to each BPMN element (it is not currently a formal part of the BPMN specification)
- A token is created in the start event, traverses the sequence flow and is destroyed in a end event. That is, there is no time associated with the token traveling down a sequence flow



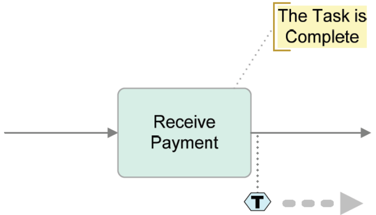
[1]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Token Game[1]

Token game

Provides an intuition of the execution semantics.

- Once a process instance has been created, we use the notion of token to identify the progress (or state) of that instance
- A token is a “theoretical object" used to create a descriptive “simulation" of the behavior associated to each BPMN element (it is not currently a formal part of the BPMN specification)
- A token is created in the start event, traverses the sequence flow and is destroyed in a end event. That is, there is no time associated with the token traveling down a sequence flow

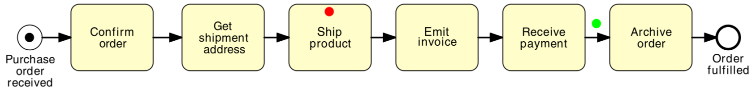


Token Game[1]

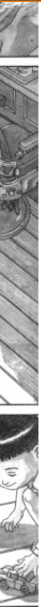
Token game

Provides an intuition of the execution semantics.

- Once a process instance has been created, we use the notion of token to identify the progress (or state) of that instance
- A token is a "theoretical object" used to create a descriptive "simulation" of the behavior associated to each BPMN element (it is not currently a formal part of the BPMN specification)
- A token is created in the start event, traverses the sequence flow and is destroyed in a end event. That is, there is no time associated with the token traveling down a sequence flow



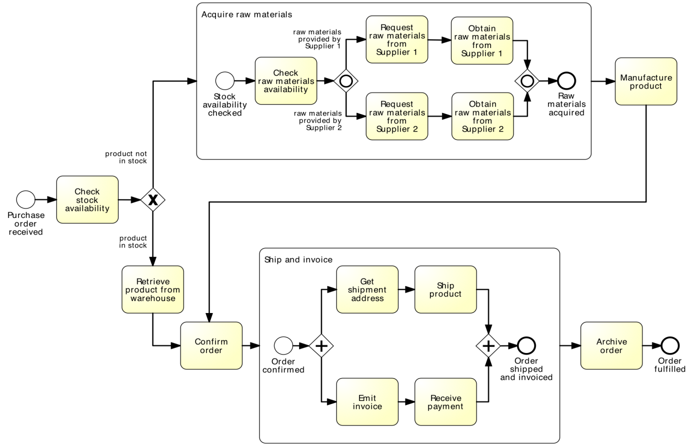
[1]Credits: Andrea Marrella "Modeling Business Processes with BPMN"



Basic Task Types

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

Expanded Subprocess

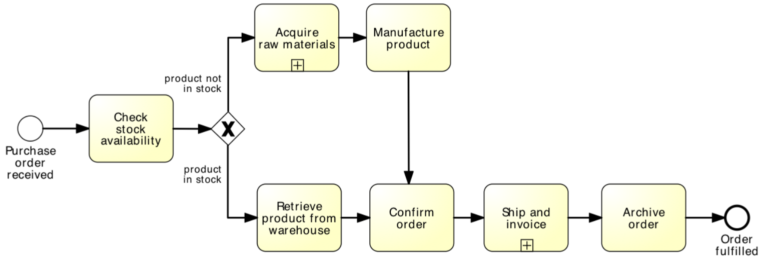


Acquire raw materials
Stock availability checked
Check raw materials availability
raw materials provided by Supplier 1
Request raw materials from Supplier 1
Obtain raw materials from Supplier 1
raw materials provided by Supplier 2
Request raw materials from Supplier 2
Obtain raw materials from Supplier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

Inner specification of a subprocess using a new BPMN diagram.

The subprocess boundary can never be crossed with flow connectors.

Collapsed Subprocess



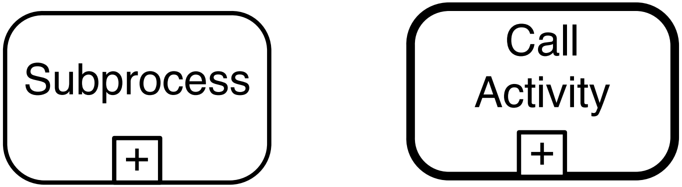
Collapsed view to *hide* the internal subprocess definition.

Subprocess expanded in a *separate diagram*.

Embdedded vs Reusable Subprocess

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

BPMN supports also the notion of call activity: a compound activity whose definition is indepedent.

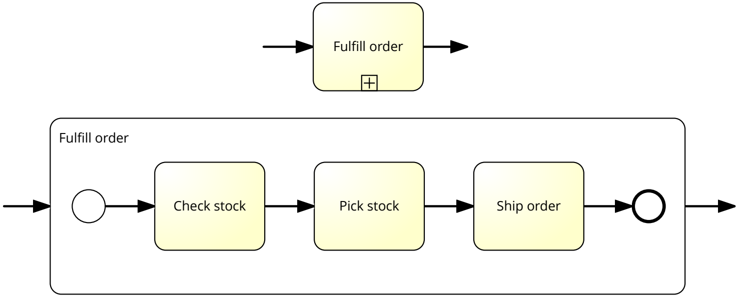


This encourages **reusability**.

Basic Subprocess Types

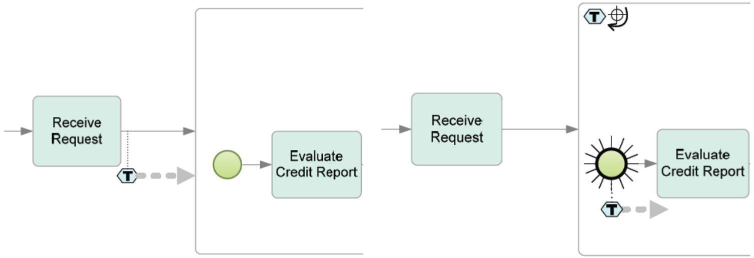
Normal subprocess

Generic subprocess. It always has a **single** *generic start event*.



Token Game[2]

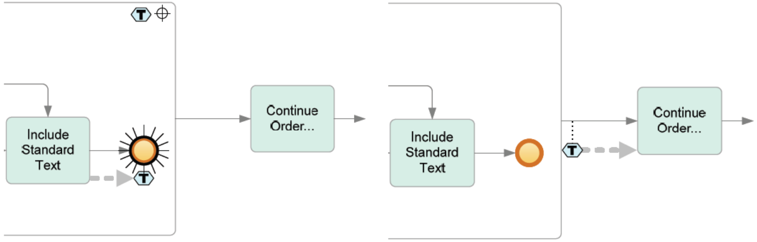
- Behavior across process levels: the subprocess is **active**



[2]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Token Game[2]

- Behavior across process levels: the subprocess is **completed**



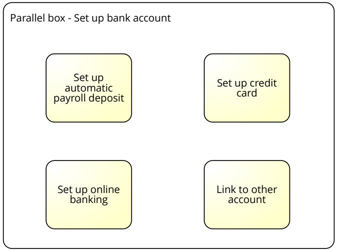
[2]Credits: Andrea Marrella “Modeling Business Processes with BPMN"

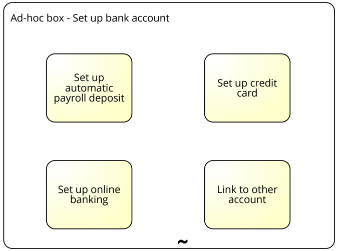
Basic Subprocess Types

Parallel box

Just contains a set of activities to be executed **concurrently**.

- All enabled when the subprocess starts.
- All to be completed to end the subprocess.


Parallel box - Set up bank account
Set up automatic payroll deposit
Set up credit card
Set up online banking
Link to other account

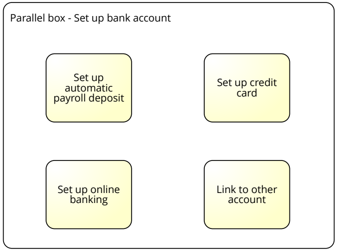

Ad-hoc box - Set up bank account
Set up automatic payroll deposit
Set up credit card
Set up online banking
Link to other account
~

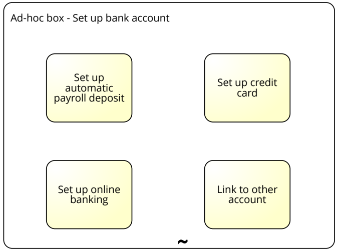
Basic Subprocess Types

Ad-hoc subprocess (marked with a $\sim$)

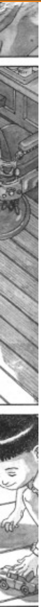
Flexible subprocess with underspecified control-flow (any order, and in any frequency).
Just contains a set of activities.

- Which activities are executed is decided case-by-case in an ad-hoc way.
- Sequence flows can be inserted to model precedence constraints.
- The subprocess can always be ended (if no activity is in execution).


Parallel box - Set up bank account
Set up automatic payroll deposit
Set up credit card
Set up online banking
Link to other account


Ad-hoc box - Set up bank account
Set up automatic payroll deposit
Set up credit card
Set up online banking
Link to other account
~

Why Subprocesses?



Modularity and end-to-end visualization.

- Importance of visualizing the whole process in a single page.
- Whole process modeled as a top-level process, whose components are modularly detailed.
- Hierarchical modeling facilitates selective expansion of details.

Top-down modeling.

- Iterative modeling by increasing level of details.
- Partial modeling while maintaing coherence of the end-to-end process.

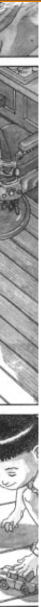
Governance boundaries.

- End-to-end process typically spans across different governance and/or organizational boundaries.
- Subprocesses can be used to unambiguously demarcate such boundaries.

Scope event handling.

- Subprocesses provide boundaries for definining specific reactions to events.
- Basis for exception handling.

Gateways



Gateway

Manages the

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

In BPMN: split gateways have a single input sequence flow, multiple output sequence flows (vice-versa for join gateways).

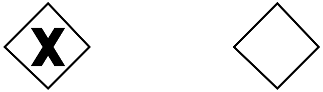
- Hybrid join+split gateways are also possible.

Execution Semantics of BPMN

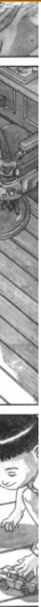
Can be found in the official documentation, Chapter 13.

Exclusive Gateway (XOR Gateway)

Empty diamond or diamond with an inner **X**.



Exclusive Gateway (XOR Gateway)



XOR Split

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

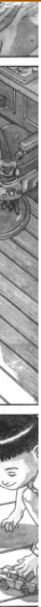
- *Data-based XOR split*: decision taken by checking data-based conditions and evaluating which one is true.
  - Conditions are expressed at this stage using natural language.
  - Default flow: represented with a bar on the sequence flow.
- *Deferred choice*: external decision under the responsibility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

Typical guideline:

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

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

Token Game: Gateway[3]

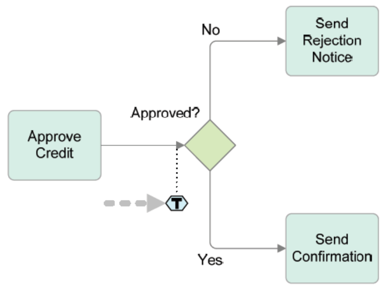


- A gateway implies that there is a gating mechanism that either allows or disallows passage of tokens through the gateway
- As tokens arrive at a gateway, they can be merged together on input, or split apart on output depending on the gateway type
- A split gateway represents a point where the process flow diverges, while a join gateway represents a point where the process flow converges

[3]Credits: Andrea Marrella “Modeling Business Processes with BPMN"

Token Game: XOR split[4]

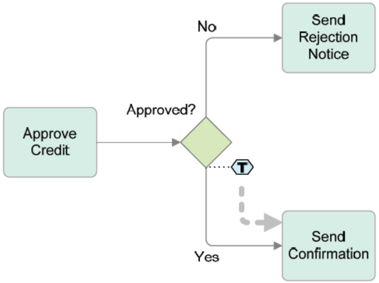
- Exclusive gateways are locations within a process where there are two or more alternative paths
- The criteria for the decision, which the exclusive gateway represents, exist as conditions on each of the outgoing sequence flow
- When a token arrives at an exclusive gateway, there is an immediate evaluation of the conditions that are on the gateway's outgoing sequence flow. One of those conditions must always evaluate to true



[4]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Token Game: XOR split[4]

- Exclusive gateways are locations within a process where there are two or more alternative paths
- The criteria for the decision, which the exclusive gateway represents, exist as conditions on each of the outgoing sequence flow
- When a token arrives at an exclusive gateway, there is an immediate evaluation of the conditions that are on the gateway's outgoing sequence flow. One of those conditions must always evaluate to true

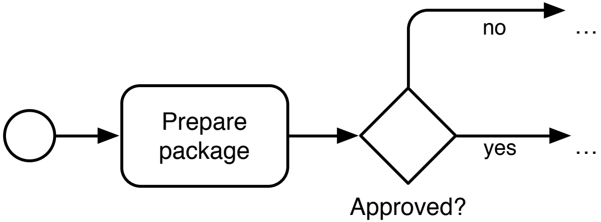


[4]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

XOR Split

Warning

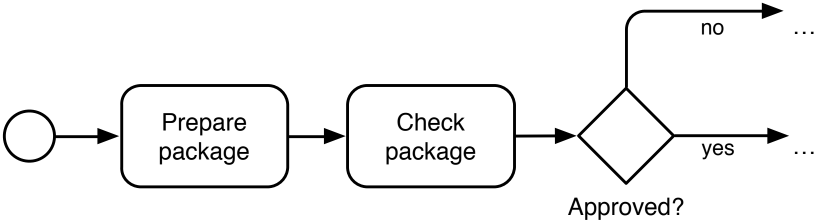
An XOR split does not make a decision. It just tests conditions.



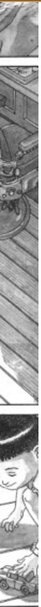
XOR Split

Warning

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



Exclusive Gateway (XOR Gateway)



XOR join

Merges common process continuations into a unique BPMN fragment.

- Whenever an input is enabled, the output is enabled.

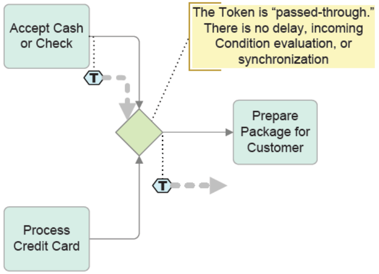
Corresponds to the notion of simple/multi-merge in the workflow patterns literature.

Attention

Is the same as connecting multiple sequence flows to the same activity.

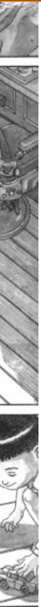
Token Game: XOR join[5]

- When a token arrives at the exclusive gateway, there is no evaluation of conditions (on the incoming sequence flow), and immediately moves down the outgoing sequence flow


Accept Cash or Check
The Token is "passed-through." There is no delay, incoming Condition evaluation, or synchronization
T
Prepare Package for Customer
T
Process Credit Card

[5]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Claim Management



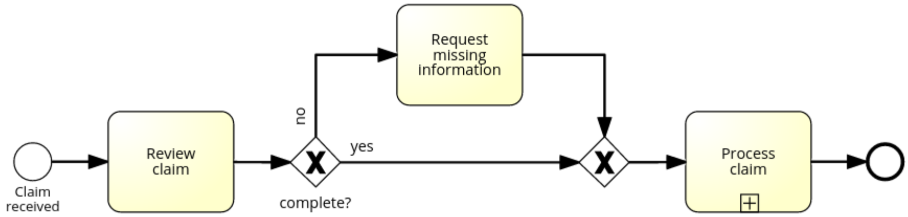
Example

The management of claims starts when a claim is received. The claim is reviewed, so as to check whether the containing info are complete or not. If not, then a dedicated task for obtaining missing info is performed. In both cases, the claim is then processed (this is a subprocess).

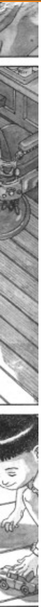
Claim Management

Example

The management of claims starts when a claim is received. The claim is reviewed, so as to check whether the containing info are complete or not. If not, then a dedicated task for obtaining missing info is performed. In both cases, the claim is then processed (this is a subprocess).

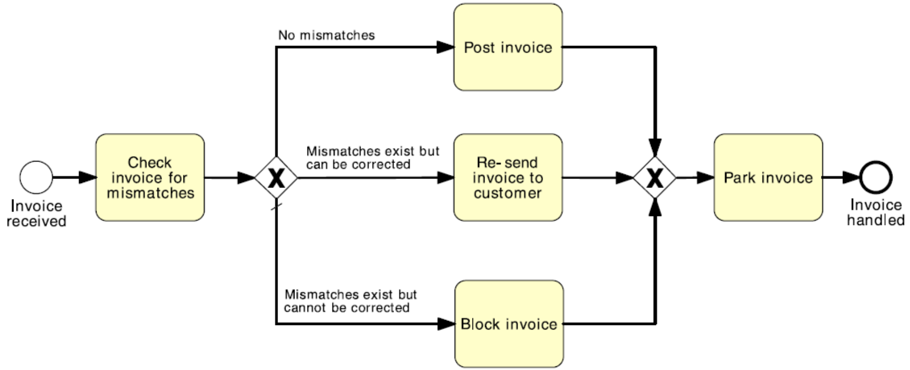


Exercise



- As soon as an invoice is received from a customer, it needs to be checked for mismatches
- The check may result in either of these three options:
  - there are no mismatches, in which case the invoice is posted
  - there are mismatches but these can be corrected, in which case the invoice is re-sent to the customer
  - there are mismatches but these cannot be corrected, in which case the invoice is blocked
- Once one of these three activities is performed the invoice is parked and the process completes

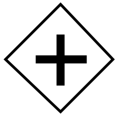
Exercise



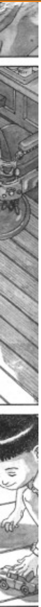
No mismatches
Post invoice
Mismatches exist but can be corrected
Re- send invoice to customer
Invoice received
Check invoice for mismatches
Park invoice
Invoice handled
Mismatches exist but cannot be corrected
Block invoice

Parallel Gateway (AND Gateway)

Diamond with an inner $+$.



Parallel Gateway (AND Gateway)



AND split

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

Such parallel threads:

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

Attention

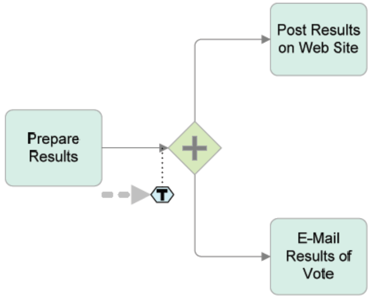
Is the same as connecting multiple sequence flows from the same activity.

AND join

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

Token Game: AND split[6]

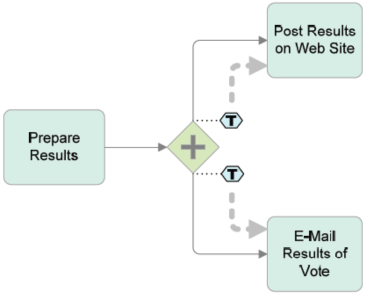
- When a token arrives at a parallel gateway, there is no evaluation of any conditions on the outgoing sequence flow.
- The parallel gateway will create parallel paths
- This means that the gateway will create a number of tokens that are equal to the number of outgoing sequence flow. One token moves down each of those outgoing sequence flow



[6]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Token Game: AND split[6]

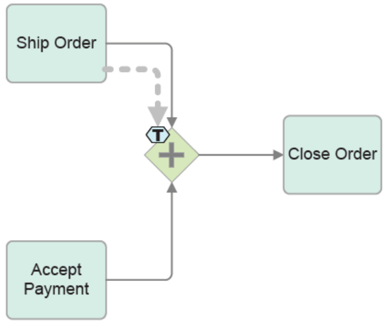
- When a token arrives at a parallel gateway, there is no evaluation of any conditions on the outgoing sequence flow.
- The parallel gateway will create parallel paths
- This means that the gateway will create a number of tokens that are equal to the number of outgoing sequence flow. One token moves down each of those outgoing sequence flow



[6] Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Token Game: AND join[7]

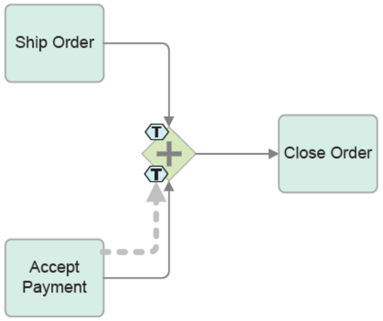
- To synchronize the flow, the parallel gateway will wait for a token to arrive from each incoming sequence flow
- When the first token arrives, there is no evaluation of a condition for the incoming sequence flow, but the token is “held” at the gateway and does not continue
- When all the tokens are arrived, then they are merged and one token moves down the outgoing sequence flow



[7]Credits: Andrea Marrella “Modeling Business Processes with BPMN"

Token Game: AND join[7]

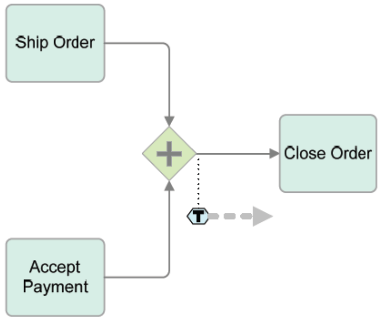
- To synchronize the flow, the parallel gateway will wait for a token to arrive from each incoming sequence flow
- When the first token arrives, there is no evaluation of a condition for the incoming sequence flow, but the token is “held” at the gateway and does not continue
- When all the tokens are arrived, then they are merged and one token moves down the outgoing sequence flow



[7]Credits: Andrea Marrella “Modeling Business Processes with BPMN"

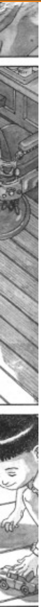
Token Game: AND join[7]

- To synchronize the flow, the parallel gateway will wait for a token to arrive from each incoming sequence flow
- When the first token arrives, there is no evaluation of a condition for the incoming sequence flow, but the token is "held" at the gateway and does not continue
- When all the tokens are arrived, then they are merged and one token moves down the outgoing sequence flow



[7] Credits: Andrea Marrella "Modeling Business Processes with BPMN"

Car Renewal Fee



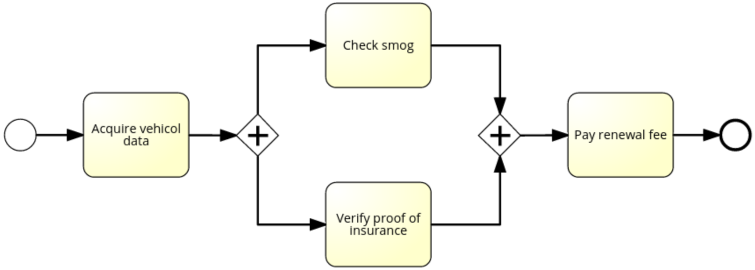
Example

The process starts by entering the vehicle data. Two activities need then to be carried out: on the one hand, a smog check needs to be passed; on the other hand, a proof of insurance must be provided. Once both activities have been executed, the renewal fee is finally paid.

Car Renewal Fee

Example

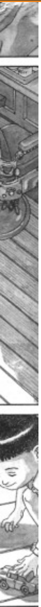
The process starts by entering the vehicle data. Two activities need then to be carried out: on the one hand, a smog check needs to be passed; on the other hand, a proof of insurance must be provided. Once both activities have been executed, the renewal fee is finally paid.



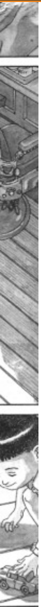
Order Management

Example

- The order management process starts by checking whether the goods requested in the order are available.
- If not, the order is rejected.
- If so, the order is confirmed.
- After the confirmation, an invoice is sent, and at the same time the requested goods are shipped. Then the process terminates.

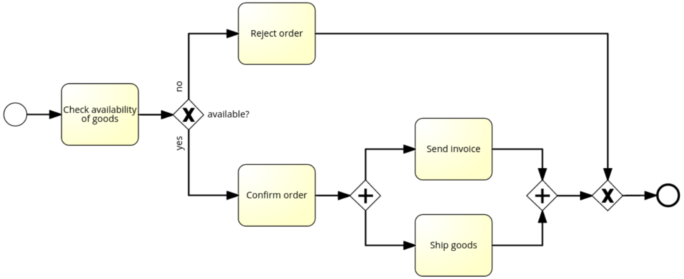


Order Management

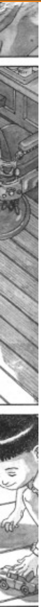


Example

- The order management process starts by checking whether the goods requested in the order are available.
- If not, the order is rejected.
- If so, the order is confirmed.
- After the confirmation, an invoice is sent, and at the same time the requested goods are shipped. Then the process terminates.

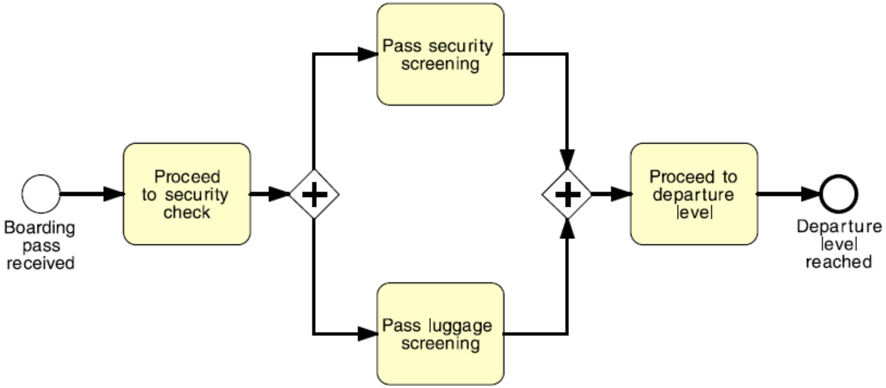


Exercise



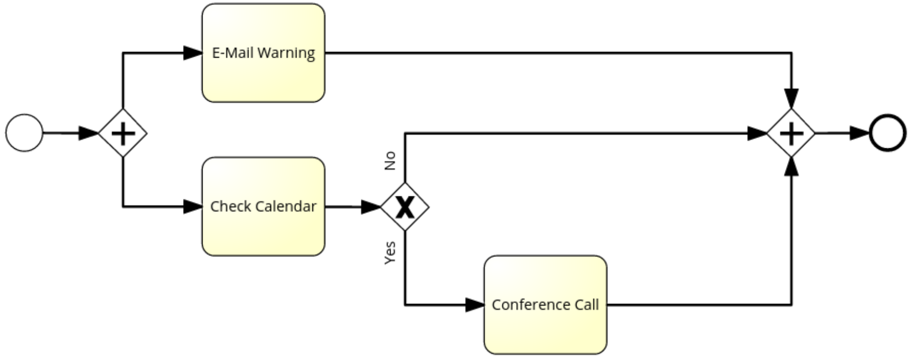
- Once the boarding pass has been received, passengers proceed to the security check. Here they need to pass the personal security screening and the luggage screening. Afterwards, they can proceed to the departure level

Exercise


Boarding pass received
Proceed to security check
Pass security screening
Pass luggage screening
Proceed to departure level
Departure level reached

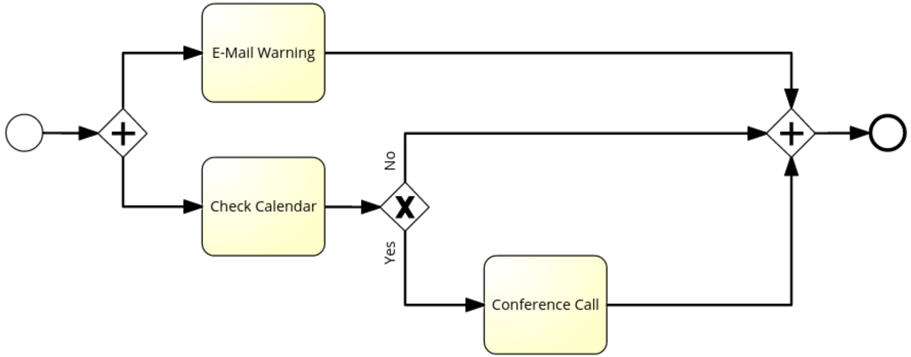
Exercise

- Describe the behavior of this process



Exercise

- Describe the behavior of this process

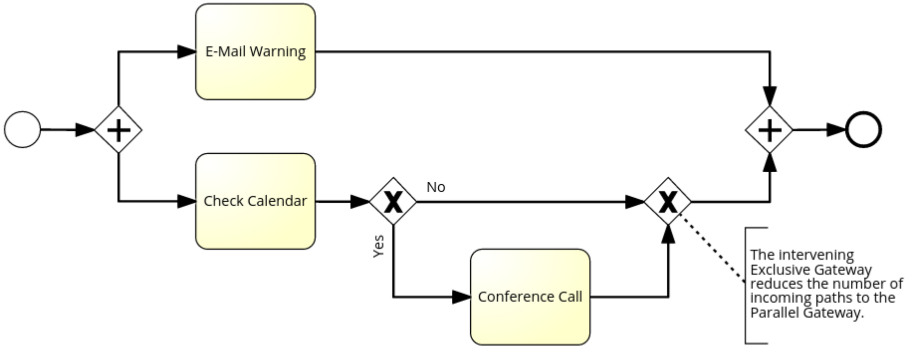


Note:

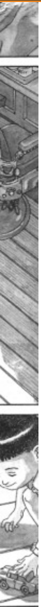
Only two of three paths will be used at any one time. Thus, the Process will be stuck waiting for the third path.

Exercise

Solution:

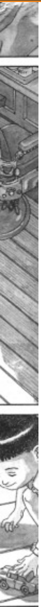

E-Mail Warning
Check Calendar
No
Yes
Conference Call
The intervening Exclusive Gateway reduces the number of incoming paths to the Parallel Gateway.

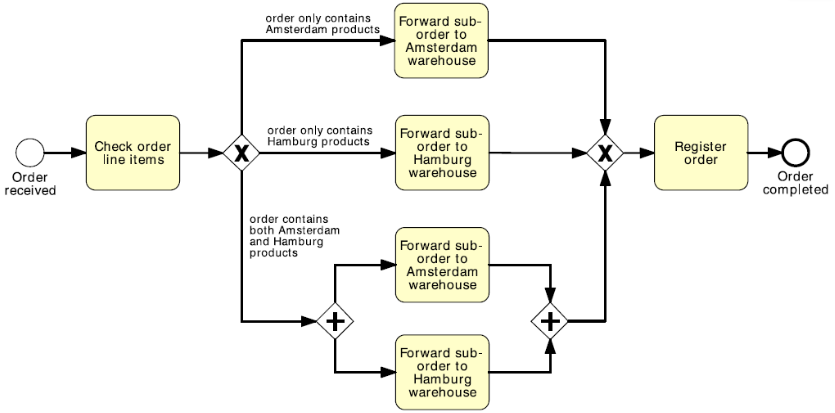
Exercise



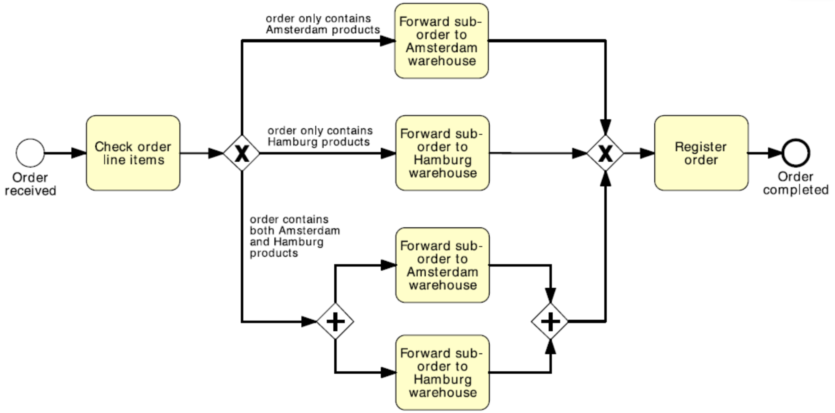
- A company has two warehouses that store different products: Amsterdam and Hamburg
- When an order is received, it is distributed across these warehouses: if some of the relevant products are maintained in Amsterdam, a sub-order is sent there; likewise, if some relevant products are maintained in Hamburg,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

Exercise




Order received
Check order line items
order only contains Amsterdam products
order only contains Hamburg products
order contains both Amsterdam and Hamburg products
Forward sub-order to Amsterdam warehouse
Forward sub-order to Hamburg warehouse
Forward sub-order to Amsterdam warehouse
Forward sub-order to Hamburg warehouse
Register order
Order completed

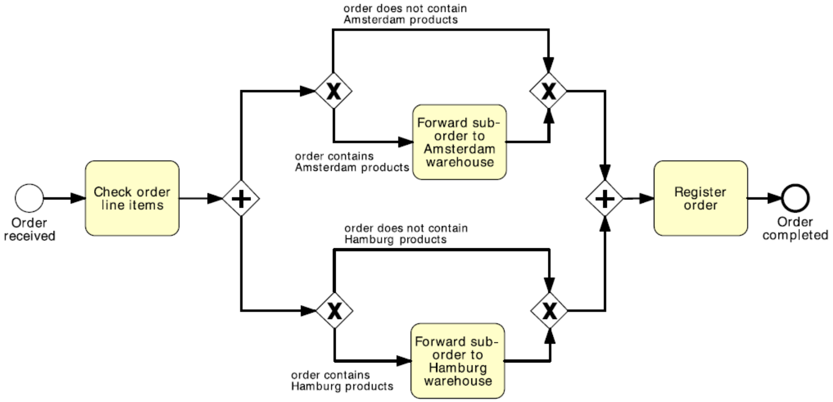
Exercise


Order received
Check order line items
order only contains Amsterdam products
Forward sub-order to Amsterdam warehouse
order only contains Hamburg products
Forward sub-order to Hamburg warehouse
order contains both Amsterdam and Hamburg products
Forward sub-order to Amsterdam warehouse
Forward sub-order to Hamburg warehouse
Register order
Order completed

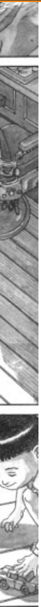
Note:

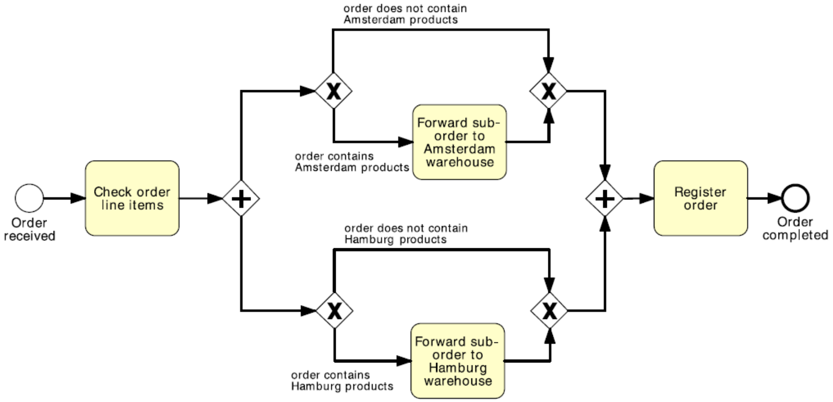
Some activities represented in the process model have to be duplicated!

Exercise


order does not contain Amsterdam products
Forward sub-order to Amsterdam warehouse
order contains Amsterdam products
Order received
Check order line items
Register order
Order completed
order does not contain Hamburg products
Forward sub-order to Hamburg warehouse
order contains Hamburg products

Exercise



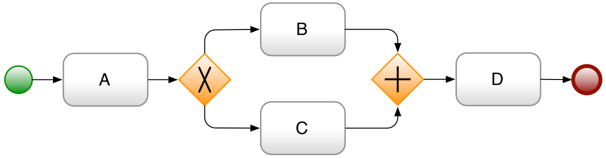

order does not contain Amsterdam products
Forward sub-order to Amsterdam warehouse
order contains Amsterdam products
Order received
Check order line items
Register order
Order completed
order does not contain Hamburg products
Forward sub-order to Hamburg warehouse
order contains Hamburg products

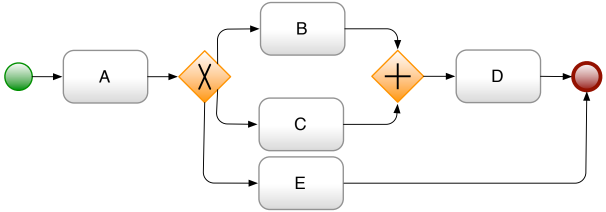
Note:

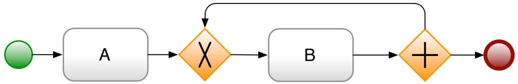
This process works also for empty orders (i.e., for orders that do not contain neither Amsterdam nor Hamburg products).

Playing with Gateways

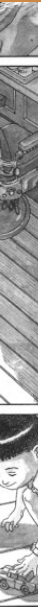
Are these models correct?







Start Event



Circle with a thin border.

Start event

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

- Every subprocess has a single, generic start event.
- Top-level processes may have multiple start events.
- Top-level processes may associate a **trigger** to the event.

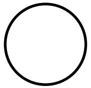
The trigger indicates the semantics of the event, and identifies the meaning of the process as the handling of such an event.

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

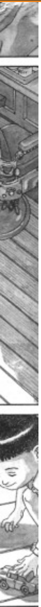
Basic triggers: none, message, timer, multiple.

None Start Event

Just a thin circle.



None Start Event



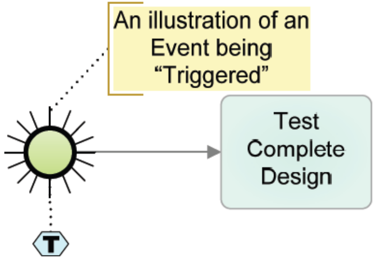
None start

A start event without trigger. It is the only one usable to start a subprocess.

- In a top-level process, indicates that the trigger is unspecified or that the process is autonomously started with manual intervention.
- In a subprocess it signifies that the subprocess starts as soon as the parent process enables it.

Token Game: Start Events[8]

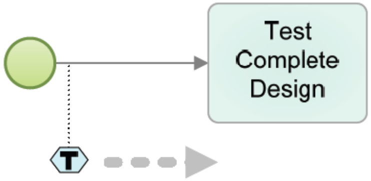
- Start Events are where the flow of a Process starts, and, thus, are where tokens are created. When a Start Event is triggered, the token is generated
- Immediately after the Start Event triggers and the token generated, the token will then exit the Start Event and travel down the outgoing Sequence Flow



[8]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

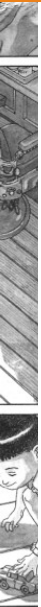
Token Game: Start Events[8]

- Start Events are where the flow of a Process starts, and, thus, are where tokens are created. When a Start Event is triggered, the token is generated
- Immediately after the Start Event triggers and the token generated, the token will then exit the Start Event and travel down the outgoing Sequence Flow

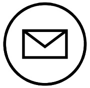


[8]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

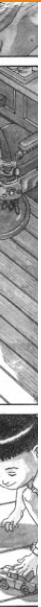
Message Start Event



Thin circle with a white envelope inside.



Message Start Event



Message start

Means that the process is triggered upon receipt of a message.

- Message exchange understood as a pair of events: emission and receipt.
- A message is just a signal coming from outside the process.
- Denotes a request delivered to the process.

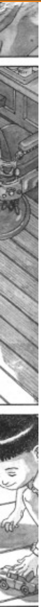
To maximize clarity, a message start event has to be coupled with a label of the form *“Receive X”*.

Message flow

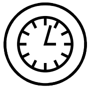
An explicit message flow can be drawn to represent who is sending the message.

- The “who” is modeled using a rectangle titled with the name of the participant (cf. pools later on).

Timer Start Event



Thin circle with a white clock inside.



Timer Start Event

Timer start

Indicates a **scheduled** process. The schedule may be:

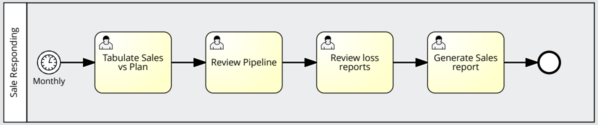
- a punctual schedule precisely located in time.
- a recurring schedule.

This is specified in the event label.

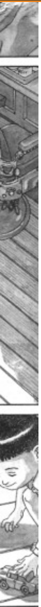
A new case is created each time the temporal condition for the schedule becomes true.

Example

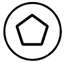
The “sales reporting” process is executed monthly.

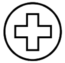


Multiple Start Event



Thin circle with a pentagon or a plus inside.





Multiple start

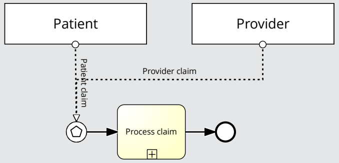
A start event that internally groups a set of triggers. Two cases:

- Multiple start (pentagon): the process is initiated every time one of the multiple triggers happens.
- Multiple-parallel start (plus): the process is initiated every time all the triggers occur (in any order).

Multiple Start Event

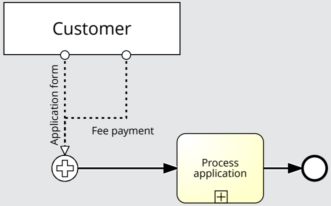
Example

A claim is processed every time a claim is received by a customer or a provider.

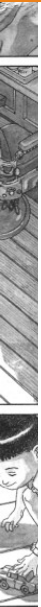


Example

An application is processed every time the customer fills an application form and pays a corresponding fee.



Alternative Start Events



With a multiple start, the same flow is executed for any of the modeled trigger.

In some cases, the initial activity of a top-level process varies depending on the trigger.

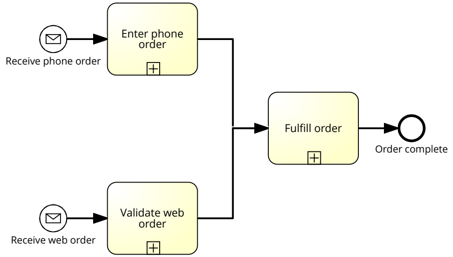
This is simply modeled by inserting alternative start events (and their corresponding flows) in the diagram. Semantics:

- Each start event is an alternative trigger.
- When one of such events is triggered, a case is created.
- The case ignores further triggers: these will be interpreted as the creation of new, separate cases.

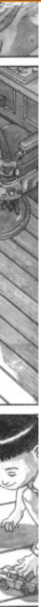
Channel-dependent start

Example

The processing of an order starts when the customer submits an order. This can be done in two ways: via a phone call, or via a web form. A phone order must be entered by an operator into the system, whereas a web order must be validated by an operator. Regardless the channel used to submit the order, it is then processed.



End Event



Circle with a thick border.

End event

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

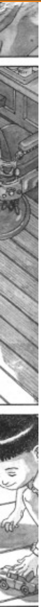
- Unlike start events, multiple end events are common.
- Typically, each end event indicates a different end state for the (sub)process.

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

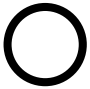
Basic results: none, message, termination, multiple.

- Represented using black, filled icons.

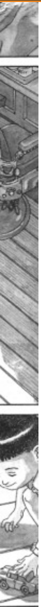
None End Event



Just a thick circle.



None End Event



None end

An end event that does not produce any result.

- For parallel flows, if they all end into a none end event, it is much better to collapse them into a single none end event.

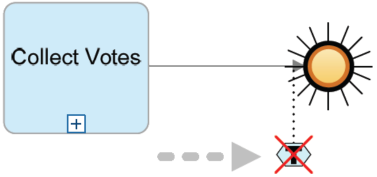
Process termination

A (sub)process instance terminates when:

1. An execution thread associated to that instance reaches an end event.
2. There are no other execution threads associated to the same instance that are still flowing inside the (sub)process.

Token Game: End Events[9]

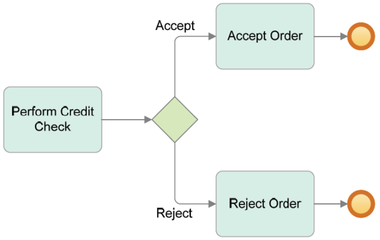
- When a token arrives at an End Event, the result of the event, if any, occurs and the token is consumed
- it is possible to have one or more paths (threads) that continue even after the token in one path has reached an End Event and has been consumed
- If the Process still contains an unconsumed token, then the Process is still "active." After all active paths have reached an End Event, the Process is then complete



[9]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

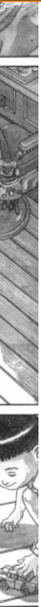
Token Game: End Events[9]

- When a token arrives at an End Event, the result of the event, if any, occurs and the token is consumed
- it is possible to have one or more paths (threads) that continue even after the token in one path has reached an End Event and has been consumed
- If the Process still contains an unconsumed token, then the Process is still "active." After all active paths have reached an End Event, the Process is then complete

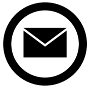


[9]Credits: Andrea Marrella "Modeling Business Processes with BPMN"

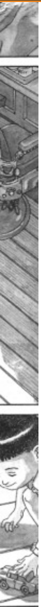
Message End Event



A thick circle with a black envelope inside.



Message End Event



Message end

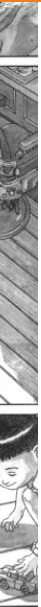
Indicates that when the end event is reached, a corresponding message is sent.

- A message flow from the end event to a separate participant should be drawn.

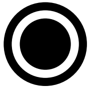
Warning

Do not directly attach to the same message end event multiple parallel flows.

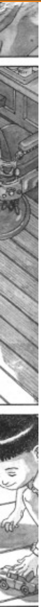
Terminate End Event



A thick circle with a bulls-eye (black-filled circle) inside.



Terminate End Event



Terminate end

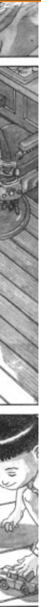
Indicates that the process must be completed **immediately**.

- If there are still active execution threads, they are immediately removed.
- Typical usage: an exception in one parallel path of a process.

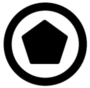
Example

The delivery of an order starts when the customer submits the order. In parallel, two branches are activated. The first branch is about preparing a package for the order delivery (subprocess). The second branch is about checking the order and preparing a receipt. Once the order is checked, two possible outcomes may arise. If there is a problem, the process immediately terminates. If everything is ok, the receipt is prepared. Once the package and the receipt have been prepared, the process ends by sending the receipt back to the customer.

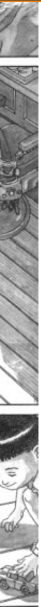
Multiple End Event



A thick circle with a pentagon inside.



Multiple End Event



Multiple end

Denotes a single ending point where multiple results are triggered.

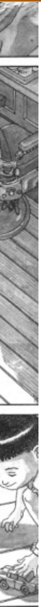
Attention

There is a mismatch in the graphical notation of multiple start and end events. An end event with a pentagon denotes that **all** results are produced.

Example

Modify the previous example by modeling that when the process is correctly terminated, then not only the receipt, but also a package code, are sent back to the customer.

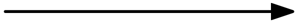
Sequence and Message Flows



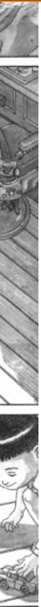
Sequence flow

Denotes orchestration: the flow of control within a process.

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



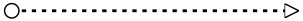
Sequence and Message Flows



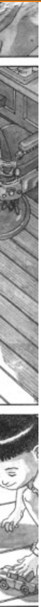
Message flow

Denotes communication between the process and an external entity.

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



Organizational Modeling



Two basic abstractions:

- **Resource**: Anything or anyone involved in the performance of a process activity. It can be a human actor, an equipment (e.g. a printer) or a software system
  - The resource perspective of a process is interested in active resources
- **Resource class**: Set of resources with shared characteristics, e.g. Clerks, Managers, Insurance Officers
  - **Role** (skill, competence, qualification) Classification based on what a resource can do or is expected to do (e.g., a clerk is a role)
  - **Group** (department, team, office, organizational unit) Classification based on the organization's structure (e.g., the administration department in an organization)

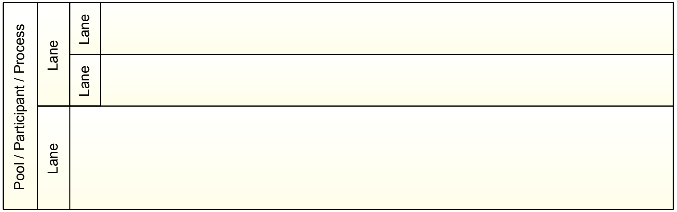
Organizational Modeling

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

- Example: Customer, Supplier, Lab, Warehouse, . . .

Lane: resource class in a given organizational space (pool), which shares the same process as other internal resource classes.

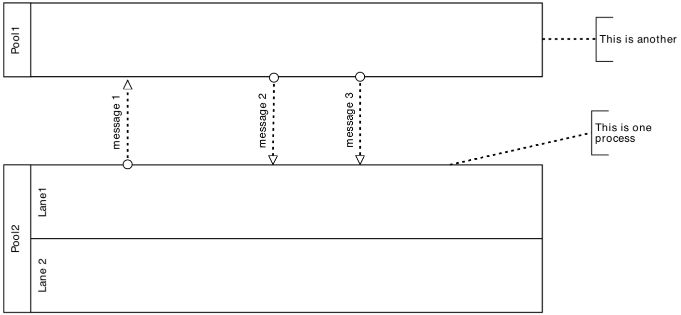
- Example: Manager, Sales Department, Engineer, . . .



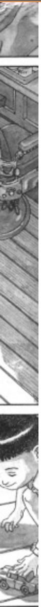
An “external” pool can also be treated as a black-box (no elements inside).

Organizational Modeling

- Each Pool represents a distinct Process and each set of resource classes has its own Pool
- Pools act as containers for the flow objects of a specific set of resource classes and communicate by means of messages.(dotted lines)
- In each pool we may have multiple lanes, often assumed to represent internal business roles within a Process. Lanes actually provide a generic mechanism for partitioning the objects within a Pool



Naming Conventions for Pools

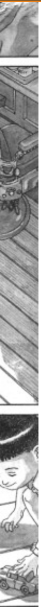


When the pool is a white-box, it is best to use as name the name of the process.

- Facilitates readability (the name of the process does not appear elsewhere).
- Helps disambiguation (the same participant is typically involved in many processes, but BPMN requires each participant to be bound to a single process within a diagram).
- Helps singling out collaboration (which may very well happen within the same organizational setting).
- Avoids the wrong creation of multiple independent processes within the same organization (for which we should use lanes instead).

When the pool is a black-box, it is best to name it with the name of the business entity or role representing that participant.

More on Pools



In a collaboration, at most one pool can be omitted.

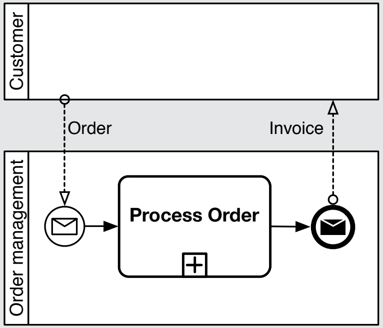
- Typically, a subprocess should omit the pool when it is expanded.
- Otherwise, make sure that the name exactly matches that of the pool used in the parent process.
    - They might be considered different participants in the XML, though.

More on Pools

In a collaboration, at most one pool can be omitted.

- Typically, a subprocess should omit the pool when it is expanded.
- Otherwise, make sure that the name exactly matches that of the pool used in the parent process.
  - They might be considered different participants in the XML, though.

Example

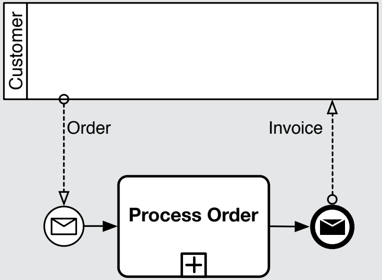


More on Pools

In a collaboration, at most one pool can be omitted.

- Typically, a subprocess should omit the pool when it is expanded.
- Otherwise, make sure that the name exactly matches that of the pool used in the parent process.
  - They might be considered different participants in the XML, though.

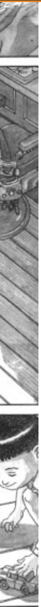
Example



More on Lanes

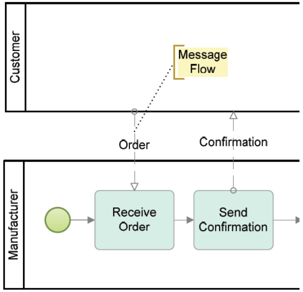
Lanes denote optional subdivisions of a process level.

- BPMN 2.0 uses lanes to mix any type of categorization for activities, not just actors/roles/divisions.
    - E.g.: a laneset for “Employee” (role), one for “Purchasing” (department)
- Multiple categorizations can be provided using lanesets.
- If lanes are modeled, each node in the process (activity, gateway, event) must be attached to exactly one lane/sublane.
- Each lane categorization stands at a specific process level. It is not inherited by subprocesses.



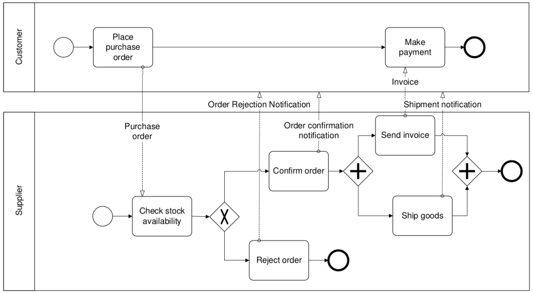
Message Flows

- 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
- Sequence flow cannot cross a pool boundary - i.e., a Process is fully contained within a pool

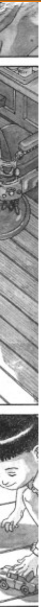


Message Flows

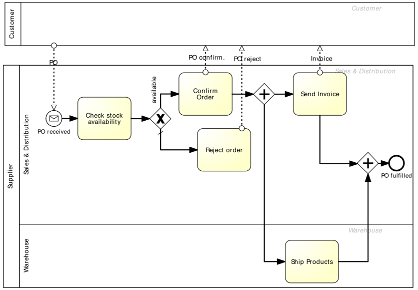
- Message flow defines the messages/communications between two separate participants (shown as pools) of the diagram
- Message flow must always occur between two separate pools and cannot connect two objects within a single pool
- Message flow is only used in collaborations diagrams with two or more pools
- Sequence flow cannot cross a pool boundary - i.e., a Process is fully contained within a pool



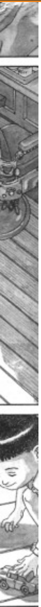
Message Flows



- 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
- Sequence flow cannot cross a pool boundary - i.e., a Process is fully contained within a pool

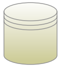


Data

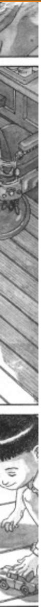


Warning

BPMN is domain agnostic: it does not provide a data model nor a pre-defined language to query and manipulate data.

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

Data Object Associations



Dotted arrows connecting a data object with an activity/event.

I/O Data

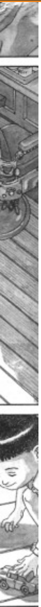
- Input data object association: maps the data object to an input variable of the activity/event.
- Output data object association: maps an output variable produced by the activity/event to the data object.

The mapping can be detailed in many ways (e.g., simple copy vs transformation vs . . .).

Data flow

Data output association followed by a data input association.

Data Store Associations



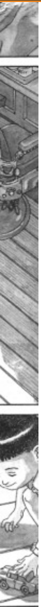
Like data object associations, but:

- An association pointing to a data store reference denotes an update.
- An association starting from a data store reference denotes a query.

Remember:

- A data store reference points to a specific unit of persistent information (not an entire storage/db/container).
- The corresponding data store is external to the process (since it is persistent).

Data Store Associations



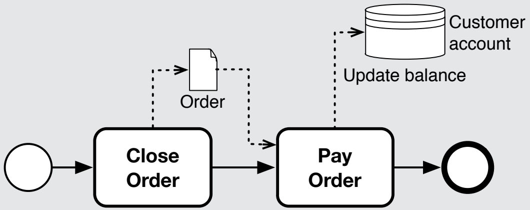
Like data object associations, but:

- An association pointing to a data store reference denotes an update.
- An association starting from a data store reference denotes a query.

Remember:

- A data store reference points to a specific unit of persistent information (not an entire storage/db/container).
- The corresponding data store is external to the process (since it is persistent).

Example



Data Store Associations

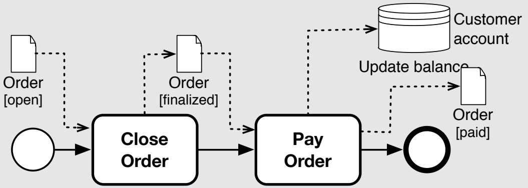
Like data object associations, but:

- An association pointing to a data store reference denotes an update.
- An association starting from a data store reference denotes a query.

Remember:

- A data store reference points to a specific unit of persistent information (not an entire storage/db/container).
- The corresponding data store is external to the process (since it is persistent).

Example



Documentation and Annotations

Documentation

The descriptive BPMN level (level 1), associates a **documentation** element to each modeling construct.

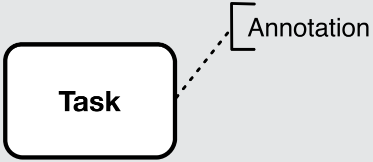
- Used to *link* additional documentation to the graphical diagram.

However, documentation has no graphical counterpart.

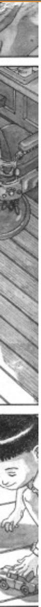
Short documentation can be shown inside annotations.

Text annotation

Textual element used to decorate any modeling construct with a textual note.



Order Management Process with Data Objects



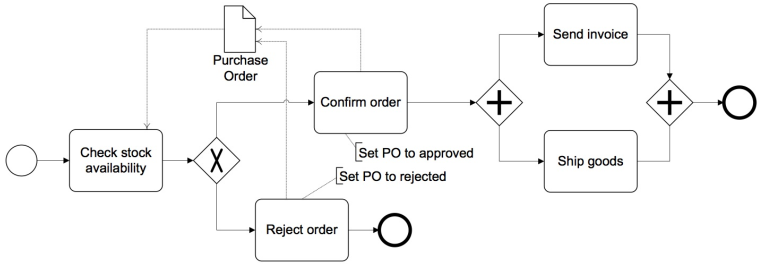
Example

Extend the Order Management Process by including the *purchase order* data object, used by the check stock activity and manipulated (“state” update) when the order is rejected/confirmed.

Order Management Process with Data Objects

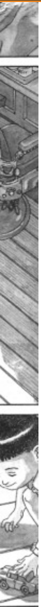
Example

Extend the Order Management Process by including the *purchase order* data object, used by the check stock activity and manipulated (“state” update) when the order is rejected/confirmed.



It is also possible to use explicit state information.

Level 1 Method



Typical level 1 goal

Create a non-executable process diagram that communicates the process logic in a meaningful way.

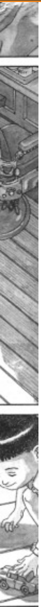
How?

- BPMN does not come with any specific modeling methodology.
- Hence, we need to complement the official BPMN specification with a method that:
  - standardizes the structure of BPMN level 1 models;
  - gives modeling principles shared by business and IT;
  - facilitates understanding of such models.

Key aspects of the method

1. Hierarchical, top-down modeling.
2. Label-matching to link processes and their subprocesses.
3. Prescriptive method, but meant to be adapted.

Modeling Approach

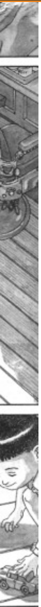


Single semantic model

Hierarchical modeling

Top-down decomposition

Modeling Approach



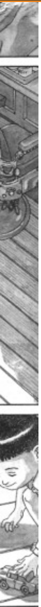
Single semantic model

A single process definition is used to describe the process **end-to-end**.

Hierarchical modeling

Top-down decomposition

Modeling Approach



Single semantic model

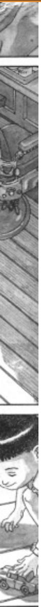
Hierarchical modeling

The semantic model is layered into **process levels**.

- Set of interlinked process diagrams (“logical pages”).
- Usage of collapsed subprocesses only.
- Single top-level diagram, unbounded nesting.

Top-down decomposition

Modeling Approach



Single semantic model

Hierarchical modeling

Top-down decomposition

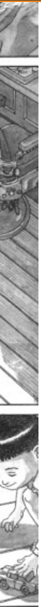
First: understand the end-to-end process as a whole.

- Top-level diagram = **high-level map** of the end-to-end process.
- Must fit into a single page.

Then: recursive drill-down.

- **Child-level diagrams** reveal as much details as required to understand the steps in the high-level map.

Modeling Approach



Single semantic model

Hierarchical modeling

Top-down decomposition

First: understand the end-to-end process as a whole.

- Top-level diagram = **high-level map** of the end-to-end process.
- Must fit into a single page.

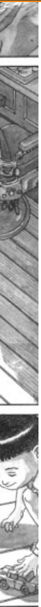
Then: recursive drill-down.

- **Child-level diagrams** reveal as much details as required to understand the steps in the high-level map.

Bottom-up modeling favours *flat diagrams*.

- No difference for the semantic model, but a big difference in terms of understandability!

End State



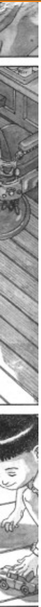
End State

Termination point of a given activity.

Questions

- How did the activity end?
- Did it complete successfully?
- Did it complete in some exception condition?
- How many end states does it have?
- How do these relate to alternative flows inside the activity?
- Are these different end states **important for the parent level**?
  - If not, then they should actually collapse!

End States - Parent Level



Each activity with multiple end states must be immediately followed by a gateway whose purpose is to **discriminate behaviors depending on which end state has been reached for the activity**.

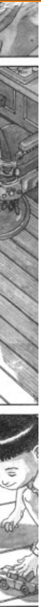
Activity with 2 end states

1. Gateway labeled with the same name of one such end states, plus a question mark.
2. The following two gates labeled with “yes” and “no”.

Activity with 3+ end states

1. Gateway labeled with a generic question.
2. Each gate labeled with the label of its corresponding end state.

End States - Child Level



If the activity is a subprocess, the different end states can be made explicit in the model:

1. Distinguish explicitly all the distinct end states.
2. Define a separate end event for each distinct end state.
3. Label each end event with an informative text.

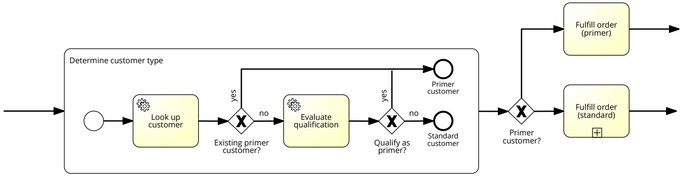
Top-down traceability

The combination of multiple end states (child level) with gateway (parent level) with matching labels provides a persistent visual link between the parent diagram and the child diagram.

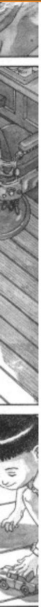
Example

A claim management process is triggered whenever a customer signals a problem. The process reacts by determining the customer type. If the customer is a premium customer, a high-priority claim reaction subprocess is invoked. If the customer is instead a standard one, a standard claim reaction subprocess is invoked. Then . . .

To determine the customer type, the company looks up the details of the customer. If the customer is already known to be premium, the subprocess immediately terminates. Contrariwise, the qualifications of the customer are evaluated, leading to two possible outcomes: the customer qualifies as a premium customer, or as a standard customer.



Step 1: Process Scope



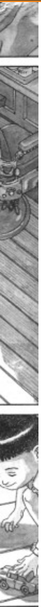
Step 1: find consensus concerning the process scope

Before entering into the details of the process, there must be consensus about *where it starts and ends*.

Questions for step 1

- How does the process start? Does it on request or is it regularly scheduled?
- What determines the process completion?
  - There is nothing after the completion that can be part of this process model.
- Which notion of "case" applies?
  - Guideline: the case typically corresponds to the start event. If the start event is a request, the corresponding case is about fulfilling the request.
- Are there different ways to end the process? Are they conceptually relevant?
  - Granularity is important.

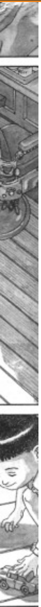
Running Example: Car Dealer Order-To-Cash



Focus: process for purchasing a new car, from the car dealer perspective.

- When does the process start?
- What information are needed to start a process?
- What is the notion of a case?
- What happens if I want to buy two cars?
- When does the process terminate?

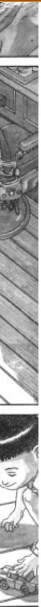
Running Example: Car Dealer Order-To-Cash



Focus: process for purchasing a new car, from the car dealer perspective.

- When does the process start?
  Well-identified moment in time: when the customer creates a purchase order.
- What information are needed to start a process?
  Detail specification of the car; customer info.
- What is the notion of a case?
  A purchase order.
- What happens if I want to buy two cars?
  Are they part of a single *contract* and *financial transaction*?
- When does the process terminate?
  - Successful completion: transaction done.
  - (Relevant) exceptions:
    - financing unavailable;
    - delivery date unacceptable;
  - Note: they both represent that the transaction has *not* been completed, but for different reasons.
    - Different involved stakeholders, different mitigation strategies!

Step 2: High-Level Map



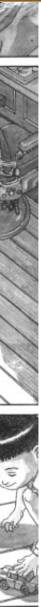
Step 2: definition of the high-level map

1. Elicit a list of ≤ 10 major activities in the process;
2. Elicit the end states of each activity, only if they have an impact on the high-level map.

Guidelines

- They must be proper BPMN activities.
  - Recurrent units of work with a clearly marked start/end.
- They act as stable *containers*, to be detailed (not changed).
- They must match with the governance boundaries of the company.
- Each activity instance must correspond one-to-one to the case.
- Each activity instance must correspond one-to-one to an instance of each one of the other activities.
- Each activity instance will start when others of the list complete.

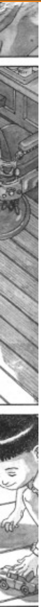
Step 2: Car Dealer



Interaction among the owner of the company, the sales manager, the service manager, the finance manager may lead to this map.

ACTIVITY	RESP.	COND.	REQUIREMENTS	END STATES
Finalize order	Sales dept.	N		Reserved from stock Dealer trade Order from factory
Get car from dealer	Sales dept.	Y	Order finalized	Car received
Get car from factory	Sales dept.	Y	Order finalized	Car received Order cancelled (dates mismatch)
Prepare car	Service dept.	N	Car obtained	Car ready
Arrange financing	Finance dept.	N	Order finalized	Financing confirmed Financing unavailable
Close and deliver	Finance dept.	N	Financing confirmed Car ready	Transaction complete
Handle order cancellation	Finance dept.	E	Order cancelled *or* financing unavailable	Delivery date unacceptable Financing unavailable

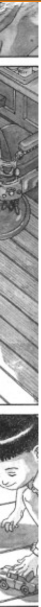
Step 3: Top-Level Process Diagram



Step 3: organize the high-level map into a top-level process diagram

1. Determine how the process start.
2. Model each activity in the high-level map as a collapsed subprocess.
3. Suitably connect the activities using sequences and basic gateways.

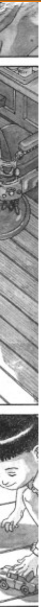
Step 3: Top-Level Process Diagram

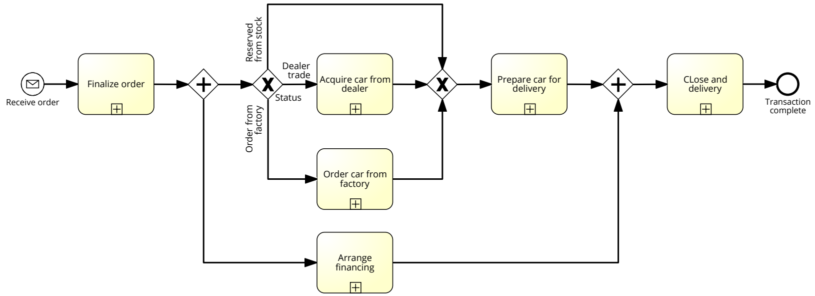


Guidelines

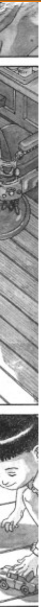
- Focus first on happy paths, then incorporate exceptional paths.
- Requirements determine control-flow dependencies by matching inputs/outputs and preconditions/effects.
- Which predecessors?
  - One: single incoming path.
  - Multiple: synchronization.
  - Alternative predecessors: multiple incoming paths (merge).
- Conditional activity: suitable coupling with a gate of the gateway testing the end state of the previous activity.
- Absence of dependency implicitly indicates concurrency.

Step 3: Top-Level Process Diagram





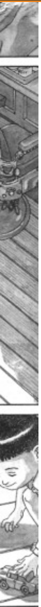
Step 3: Car Dealer

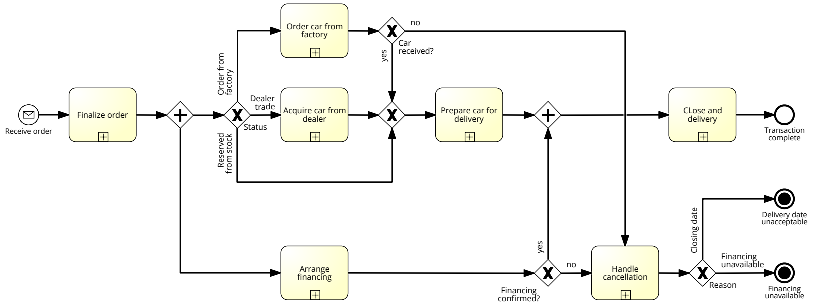


Key observations

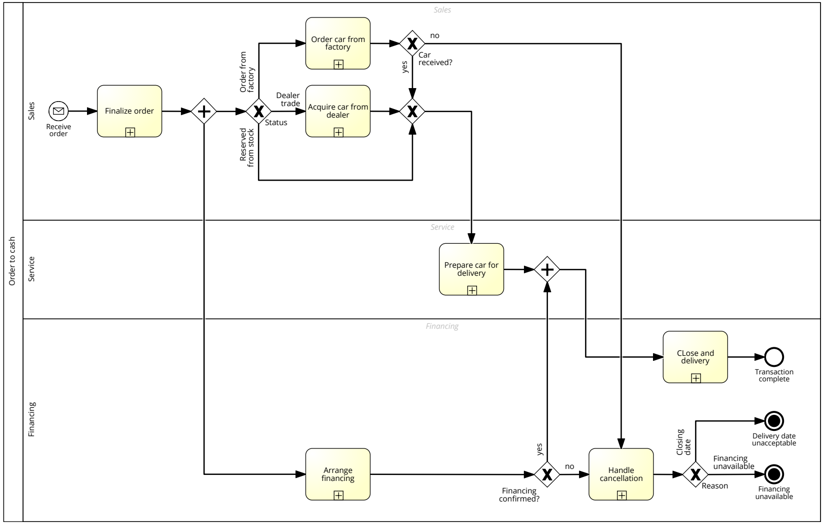
- A car is obtained if it is received or already present in the stock of the dealer.
- Arrange financing does not interact with the part of the process in which the car is obtained and prepared.
- Exceptional flows implicitly make optional all the normal activities that come after the exception trigger.

Step 3: Car Dealer

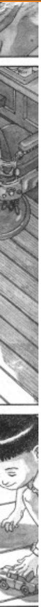




Step 3: Car Dealer



Step 4: Child-Level Expansion



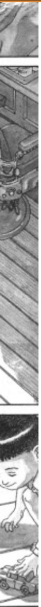
Step 4: reveal the internal details of top-level activities

1. Create a separate diagram for each top-level activity.
2. Link it to the corresponding collapsed subprocess.
3. Detail the content of the child-level process, possibly refining the classification of end states (this impacts on the high-level map).
4. Continue recursively.

Guidelines

- A child-level process must have a unique none start event.
- There must be agreement of the end states and the corresponding gateway in the parent process.
- If the child-level process has a pool, its name must match that of the parent process (i.e., the name of the process).
- Lanes can be added - they are “local” to each process.

Step 4: Car Dealer



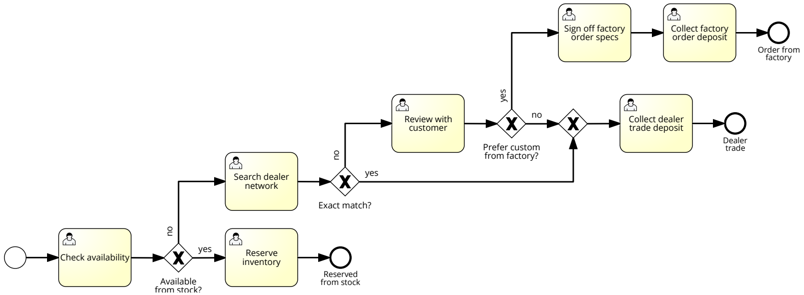
Finalize order

The order starts to be finalized when a receipt for the order is generated. The receipt contains info about the customer (car buyer), the car (or specification of a car), the agreed price. There are three possible ways to finalize an order: by reserving the car from stock, by trading with another dealer, or by placing an order to a factory.

First of all, the car dealer checks the availability of the car in its internal stock. If it is available, then the car is reserved and the activity ends. If not, then the car dealer searches its dealer network, trying to find some other dealer offering exactly the car desired by the customer. If a perfect match is found, then the dealer trade deposit is collected. Contrariwise, a joint review with the customer is performed over the best approximate matching. The customer then takes a decision, concerning whether she prefers to obtain a custom configuration of the car directly from the factory, or whether she prefers the approximate match. In the latter case, the process continues like in the case of a perfect match. In the former case, the order specs of the factory are signed off, and then the factory order deposit is collected.

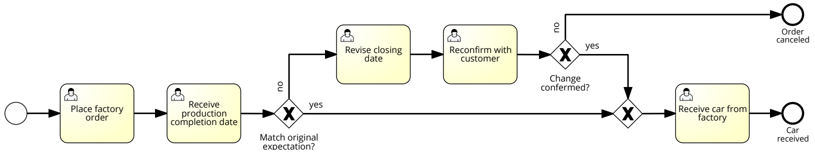
Step 4: Car Dealer

Finalize order

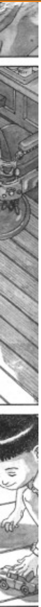


Step 4: Car Dealer

Order car from factory



Step 5: Message Flows



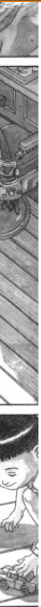
Step 5: add message flows clarifying the process business context

1. Insert all relevant external participants as black-box pools.
2. Draw message flows between such black-box pools and activities/events in the process model of interest.
3. Ensure consistency across levels: input/output messages attached to an activity must exactly be reflected in the child process, considering numbers and labels.
    - Top-down traceability.
    - The inner process clarifies the orderings among events.

Observations

- Top-down traceability of messages requires the top-level process to list all of them.
- Whenever we have user tasks, message flows become optional.
    - Output message flow: not guaranteed to be generated.
    - Input message flow: not necessarily required to execute the task.

Step 5: Car Dealer



Order car from factory

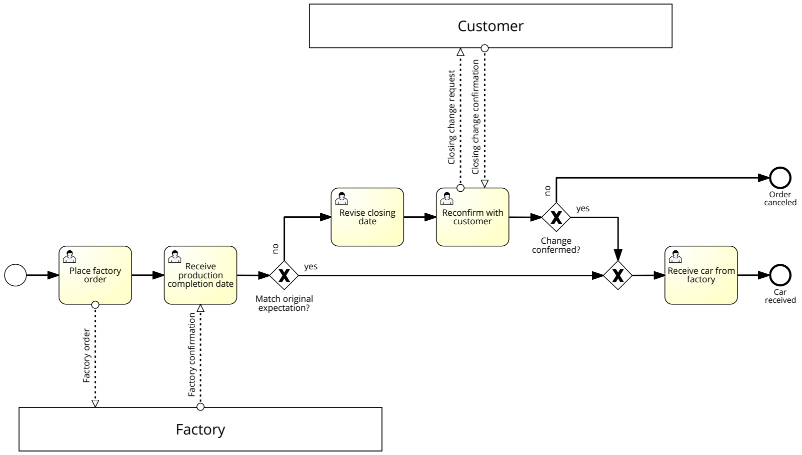
It involves the interaction with the customer and the factory. This interaction foresees four possible messages:

- Factory order - sent to the factory to place an order.
- Factory confirmation - confirmation sent back to the process; it contains the expected completion date.
- Closing change request - sent to the customer to check whether she is willing to change the closing date.
- Closing change confirmation - the confirmation by the customer.

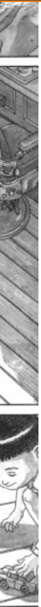
The subprocess starts by placing the order to the factory, then receiving a confirmation together with the expected production date. This date is then confronted with the original, planned date. If they match, then the car is received from the factory and the subprocess terminates. Otherwise, the closing date is revised, consequently reconfirming with the customer whether the new date is ok or not. If so, the car is received from factory, otherwise, the process terminates in the exceptional state where it is cancelled.

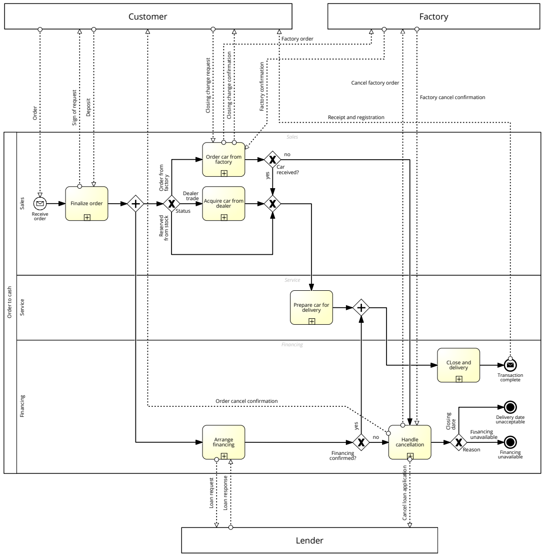
Step 5: Car Dealer

Order car from factory



Step 5: Car Dealer




Customer
Factory
Order
Sign of request
Deposit
Closing change request
Closing change confirmation
Factory confirmation
Factory order
Cancel factory order
Factory cancel confirmation
Receipt and registration
Order to cash
Sales
Service
Financing
Receive order
Finalize order
Order from factory
Dealer trade
Status
Reserved from stock
Order car from factory
Acquire car from dealer
no
Car received?
yes
Prepare car for delivery
CLose and delivery
Transaction complete
Order cancel confirmation
Arrange financing
Financing confirmed?
yes
no
Handle cancellation
Closing date
Delivery date unacceptable
Financing unavailable
Reason
Financing unavailable
Loan request
Loan response
Cancel loan application
Lender