

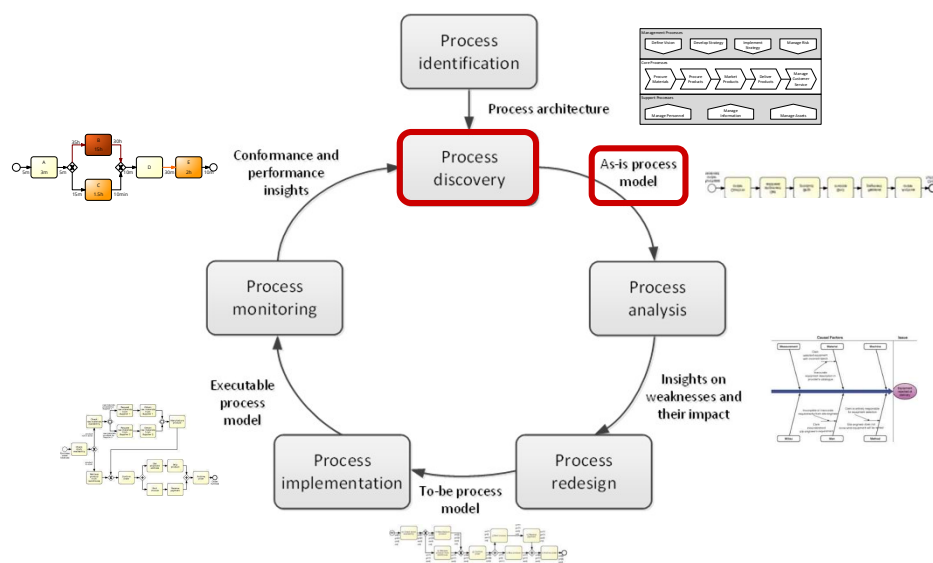
Advanced Process Modeling

Spring 2021 - MAJU

Nauman H. Ansari

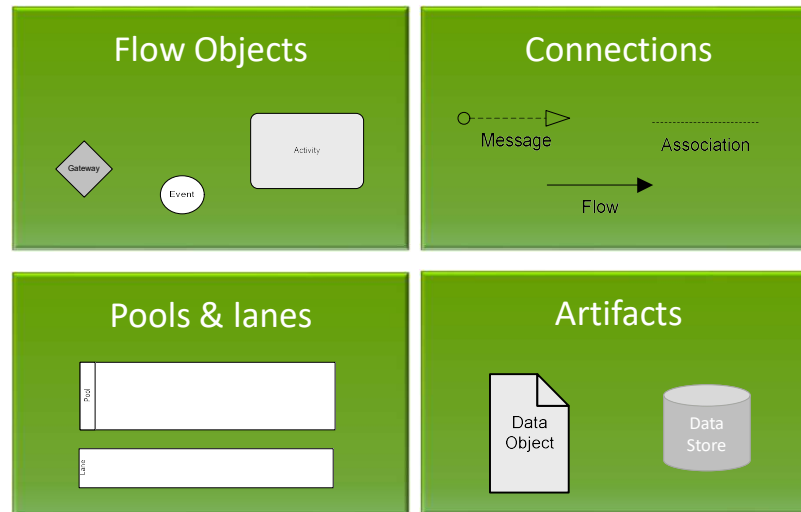
1

Process Modeling in the BPM Lifecycle



2

BPMN Main Elements - Recap



3

BPMN Gateways

Exclusive (XOR)	Parallel (AND)	Inclusive (OR)
<ul style="list-style-type: none"> • <u>Exclusive decision</u> take one branch • <u>Exclusive merge</u> Proceed when one branch has completed 	<ul style="list-style-type: none"> • <u>Parallel split</u> take all branches • <u>Parallel join</u> proceed when all incoming branches have completed 	<ul style="list-style-type: none"> • <u>Inclusive decision</u> take one or several branches depending on conditions • <u>Inclusive merge</u> proceed when all <u>active</u> incoming branches have completed

4

Process Decomposition

- As we tackle more complex business processes, we will undoubtedly produce larger models, i.e. models with many elements, and this will hamper the overall model understandability.
- To improve understandability, we can simplify the model by hiding certain parts within a sub-process.
- A sub-process represents a self-contained, composite activity that can be broken down into smaller units of work.
- Conversely, an atomic activity, also called task, is an activity capturing a unit of work that cannot be further broken down.
- To use a sub-process, we first need to identify groups of related activities, i.e. those activities which together achieve a particular goal or generate a particular outcome.

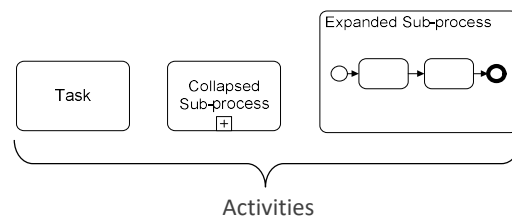
© Marcello La Rosa

5

5

Process Decomposition

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

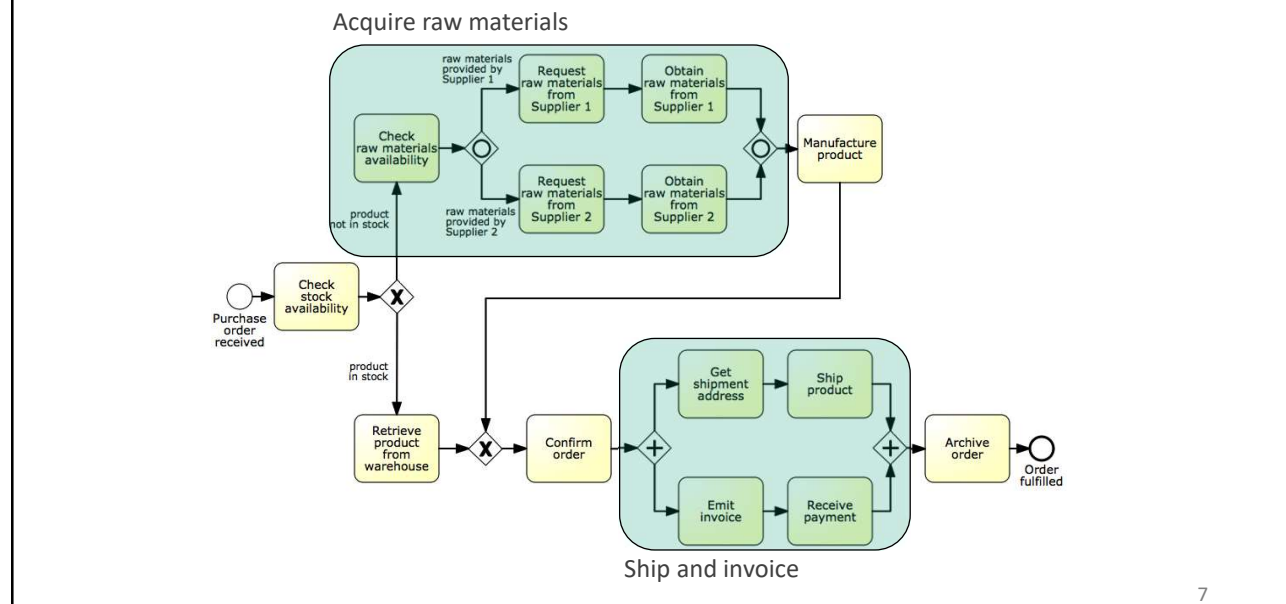


Use this feature to:

1. Decompose large models into smaller ones, making them easier to understand and maintain
2. Share common fragments across multiple processes
3. Delimit parts of a process that can be:
 - Repeated
 - Interrupted

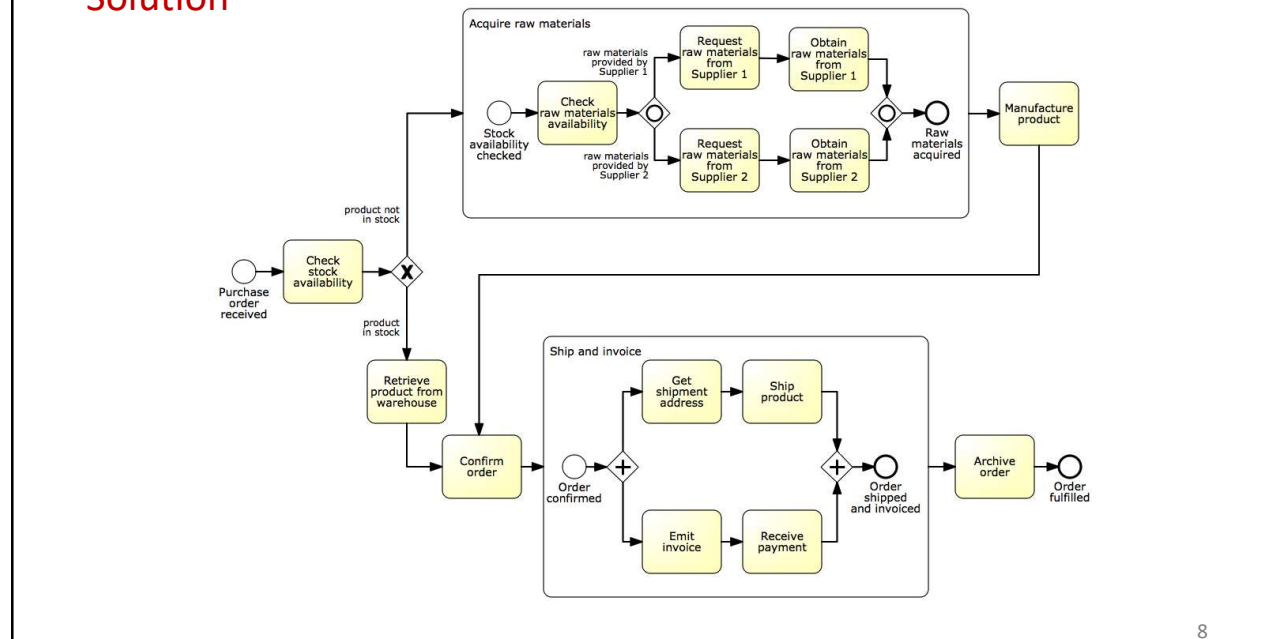
6

Identify possible sub-processes



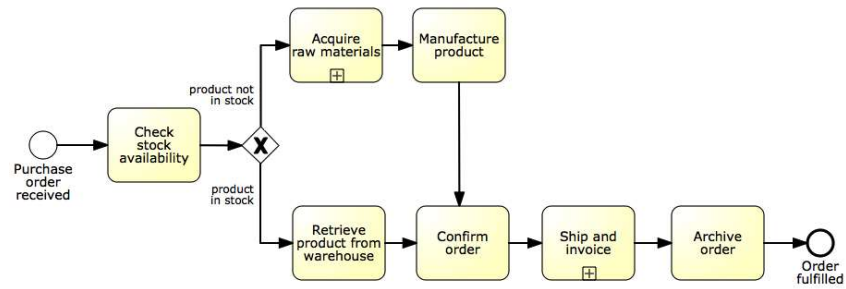
7

Solution



8

The refactored model

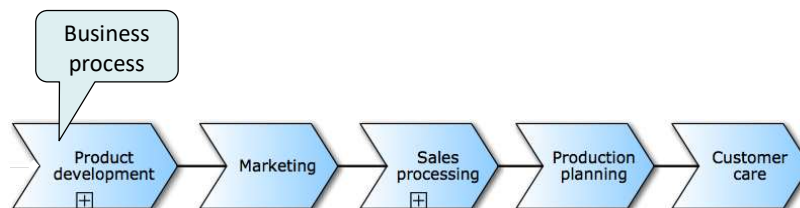


9

9

Value chain modelling

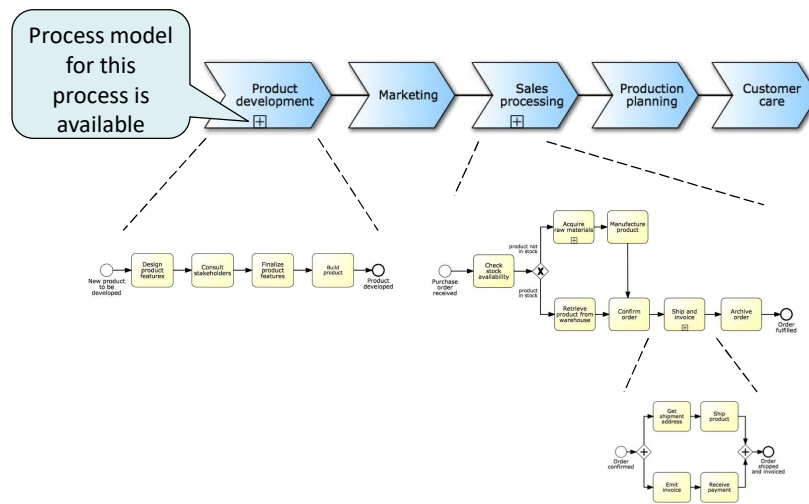
Chain of (high-level) processes an organisation performs in order to achieve a business goal, e.g. deliver a product or service to the market.



10

10

Linking value chains with process models



11

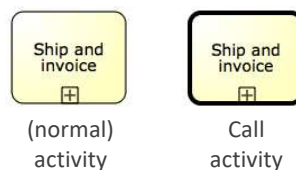
11

Process Reuse

By default, a sub-process is “embedded” into its parent process (i.e. it is stored within the same file)

In order to maximize reuse, it is possible to “extract” the sub-process and store it as a separate file in the process model repository

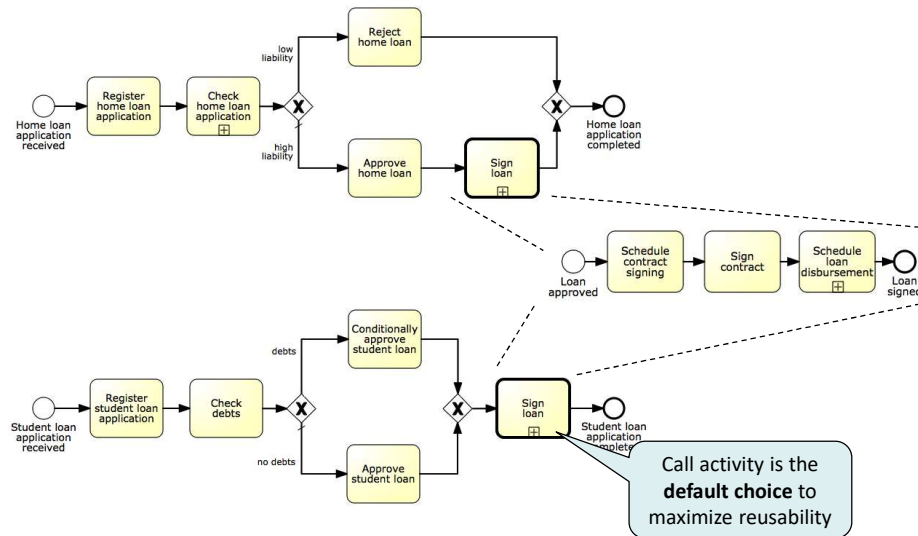
Such a sub-process is called “global” model, and is invoked via a “call” activity



12

12

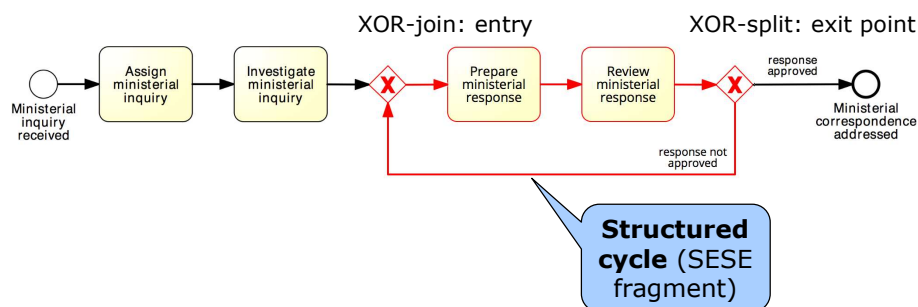
Example: process reuse



13

13

More on rework and repetition

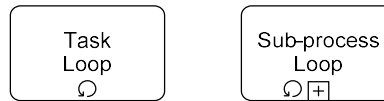


SESE = Single Entry Single Exit fragment, i.e. a fragment delimited by a single entry node and a single exit node (there are no other incoming arcs into the fragment or outgoing arcs from the fragment)

14

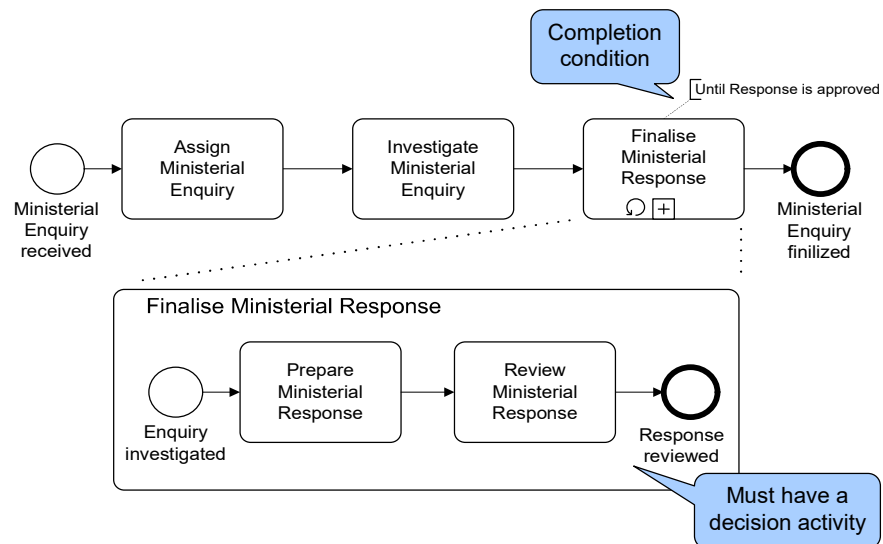
Block-structured repetition: Loop Activity

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



15

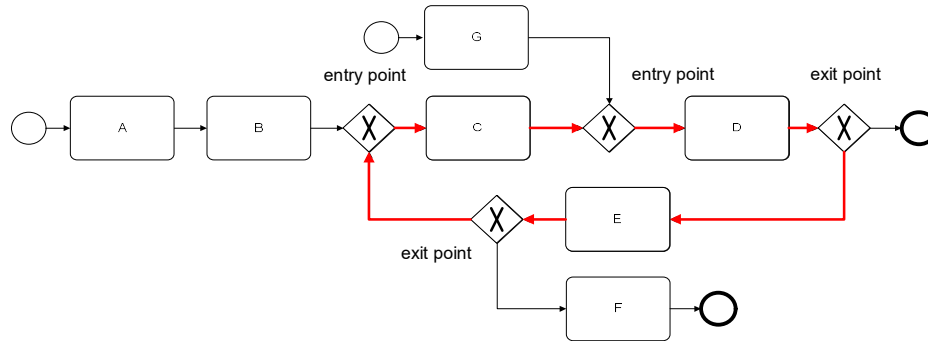
Example: block-structured repetition



16

Arbitrary Cycles

Arbitrary = unstructured, i.e. it can have multiple entry and exit nodes (non-SESE).

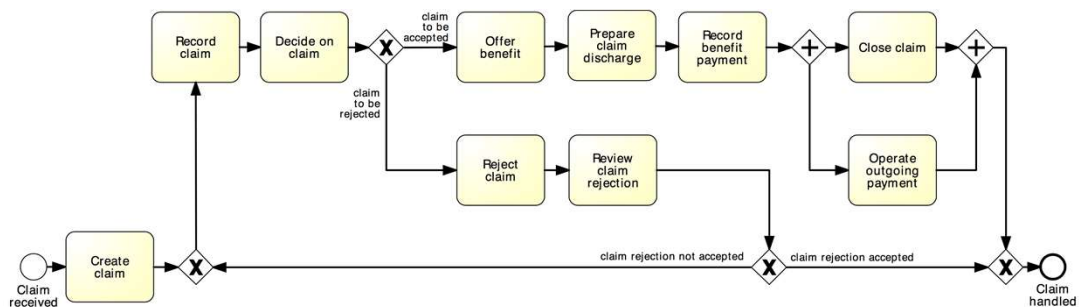


17

17

Exercise 4.1 (a)

Identify the entry and exit points that delimit the unstructured cycles in the process model shown below. What are the repetition blocks?



18

Solution 4.1 (a)

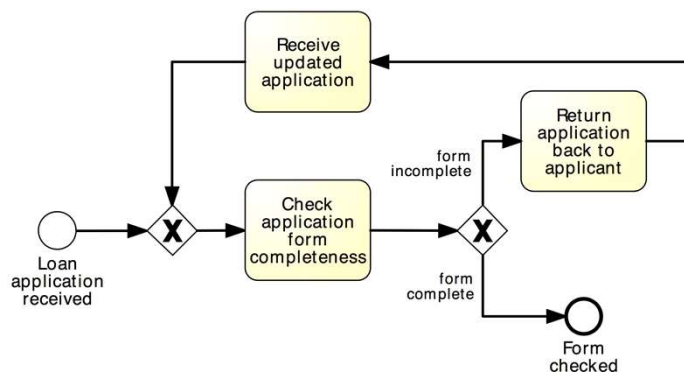
- The repetition block goes from activity “Record claim” to activity “Review claim rejection”. The entry point to the cycle is the arc from activity “Create claim” to the subsequent XOR-join. The exit points are the arcs “claim to be accepted” and “claim rejection accepted”, the former emanating from within the repetition block.

19

19

Exercise 4.1

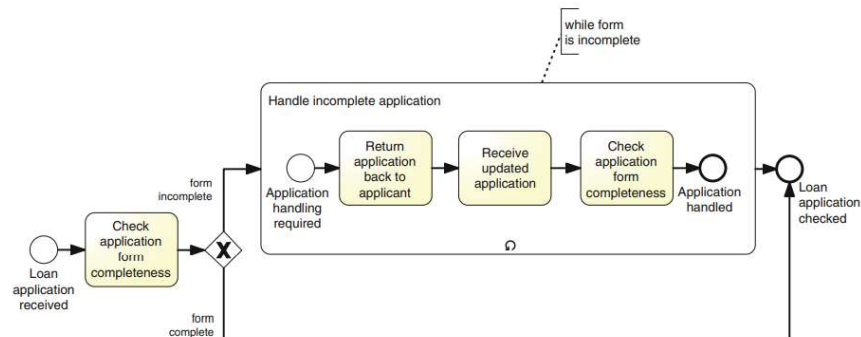
Model the business process shown below using a loop activity.



20

Solution 4.1 (b)

- The repetition block is made up of the activities “Check application form completeness”, “Return application back to applicant”, and “Receive updated application”. The entry point to the cycle is the outgoing arc of the XOR-join, while the exit point is the arc “form complete” which emanates from within the repetition block. To model this cycle with a loop activity, we need to repeat activity “Check application form completeness” outside the loop activity, as shown below.



21

21

Parallel Repetition: multi-instance activity

- The loop activity allows us to capture sequential repetition, meaning that instances of the loop activity are executed one after the other.
- Sometimes though, we may need to execute multiple instances of the same activity at the same time
- Parallel repetition is useful when the same activity needs to be executed for multiple entities or data items, such as:
 - Request quotes from multiple suppliers
 - Check the availability for each line item in an order separately
 - Send and gather questionnaires from multiple witnesses in the context of an insurance claim



22

Example: multi-instance activity

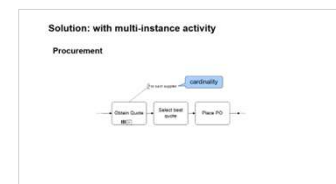
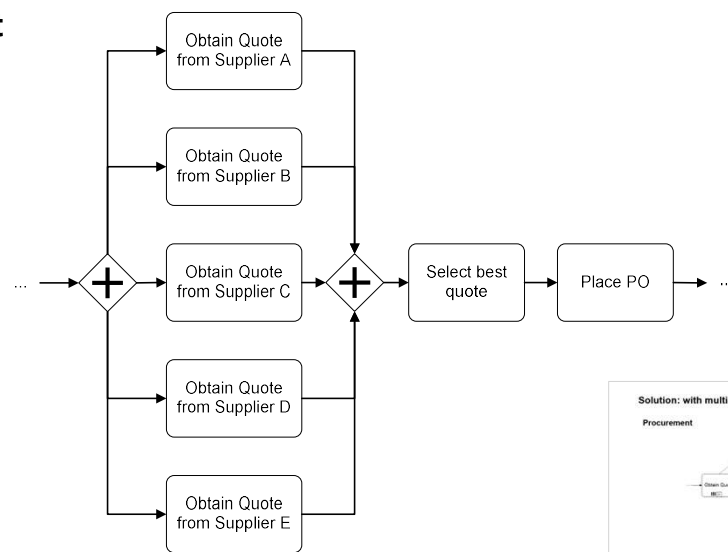
Procurement

In procurement, typically a quote is to be obtained from all preferred suppliers (assumption: five preferred suppliers exist). After all quotes are received, they are evaluated and the best quote is selected. A corresponding purchase order is then placed.

23

Solution: without multi-instance activity

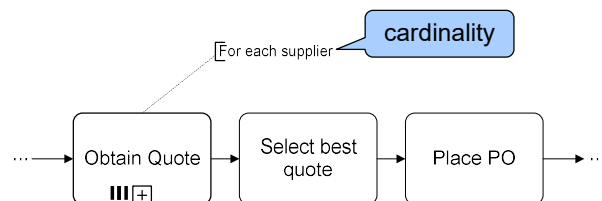
Procurement



24

Solution: with multi-instance activity

Procurement



25

Further example: multi-instance activity

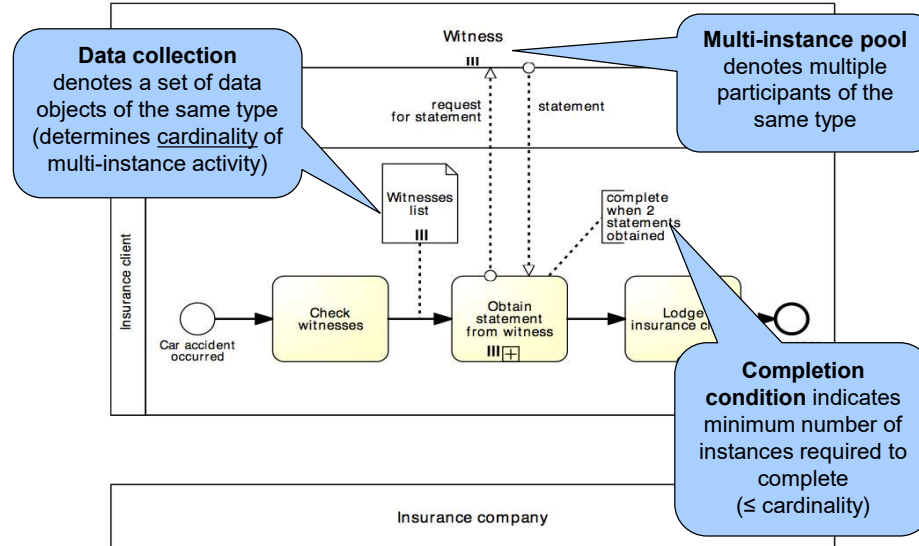
Motor insurance claim lodgement

After a car accident, a statement is sought from the witnesses that were present, in order to lodge the insurance claim. As soon as the first two statements are received, the claim can be lodged to the insurance company without waiting for the other statements.

26

Solution: multi-instance activity

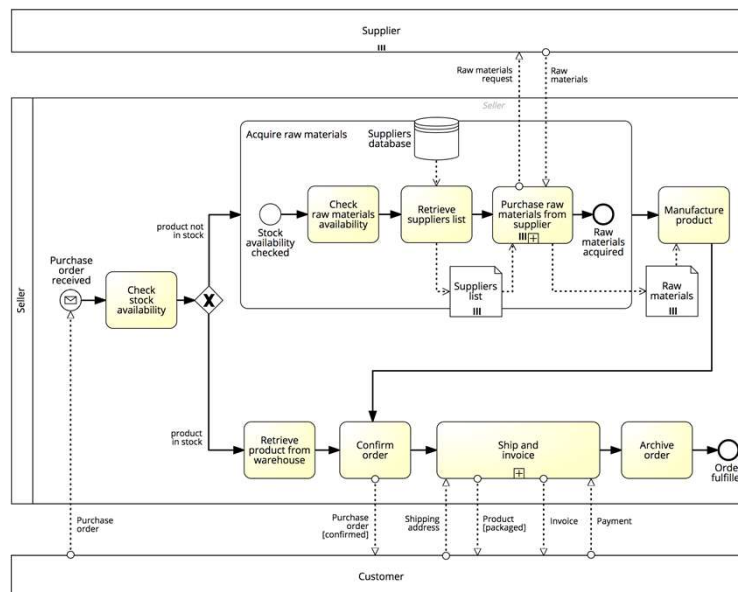
Motor insurance claim lodgement



27

Our order-to-cash example...

now with pools, messages and MI markers



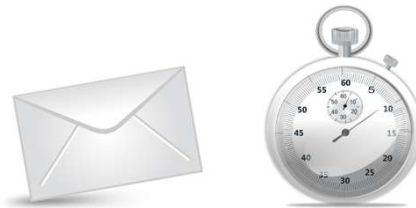
28

Handling Events

- We have learnt that events are used to model something that happens instantaneously in a process.
- We saw start events, which signal how process instances start (tokens are created), and end events, which signal when process instances complete (tokens are destroyed).
- When an event occurs during a process, e.g., an order confirmation is received after sending an order out to the customer and before proceeding with the shipment, the event is called intermediate. A token remains trapped in the incoming sequence flow of an intermediate event until the event occurs.
- Once the event occurs, the token traverses the event instantaneously, i.e., events cannot retain tokens. An intermediate event is represented as a circle with a double border.

They affect the process flow:

- Start
- Intermediate
- End



29

29

How do we model this scenario?

PO handling

A Purchase Order (PO) handling process starts when a **PO is received**. The PO is first registered. If the current date is not **a working day**, the process waits until the **following working day** before proceeding. Otherwise, an availability check is performed and a **PO response is sent** back to the customer.

30

BPMN event types

Start Intermediate End



Untyped Event – Indicates that an instance of the process is created (start) or completed (end), without specifying the cause for creation/completion



Start Message Event – Indicates that an instance of the process is created when a message is **received**



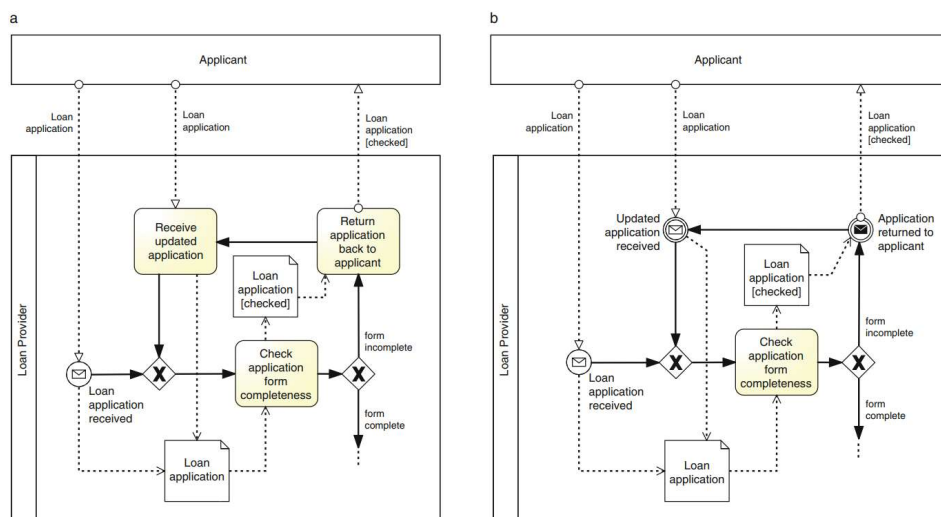
End Message Event – Indicates that an instance of the process is completed when a message is **sent**



Intermediate Message Event – Indicates that an event is expected to occur during the process. The event is triggered when a message is **received** or **sent**

31

Replacing activities that only send or receive messages



32

32

Temporal events

Start Intermediate End



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

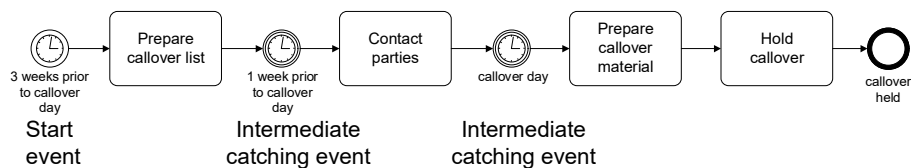


Intermediate Timer Event – Triggered at certain date(s)/time(s), or after a time interval has elapsed since the moment the event is “enabled” (delay)

33

Example: temporal events

In a small claims tribunal, callovers occur once a month to set down the matter for the upcoming trials. The process for setting up a callover starts **three weeks prior** to the callover day, with the preparation of the callover list containing information such as contact details of the involved parties and estimated hearing date. **One week prior** to the callover, the involved parties are notified of the callover date. Finally, **on the callover day**, the callover material is prepared and the callover is held.

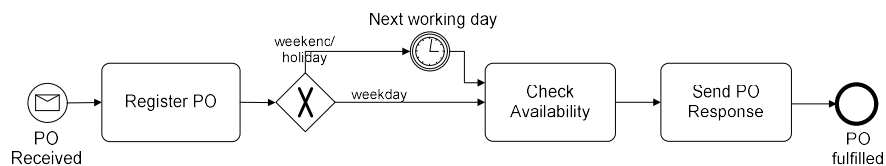


34

Coming back to our scenario...

PO handling

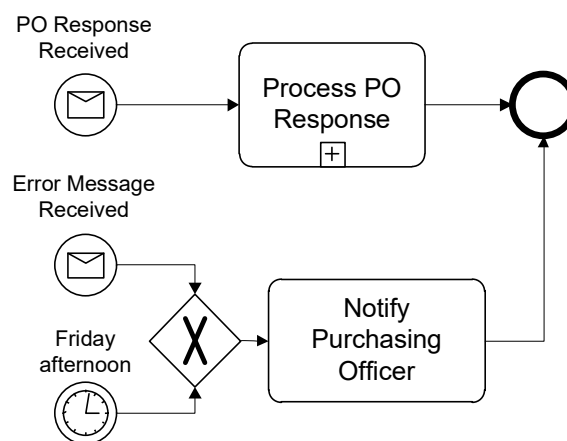
A Purchase Order (PO) handling process starts when a **PO is received**. The PO is first registered. If the current date is not a **working day**, the process waits until the **following working day** before proceeding. Otherwise, an availability check is performed and a **PO response is sent** back to the customer.



35

Multiple start events

The first start event that occurs will trigger an instance of the process

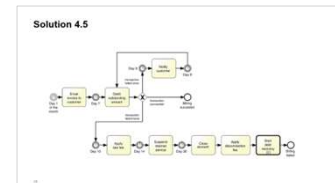


36

Exercise 4.5: Internet Service Provider

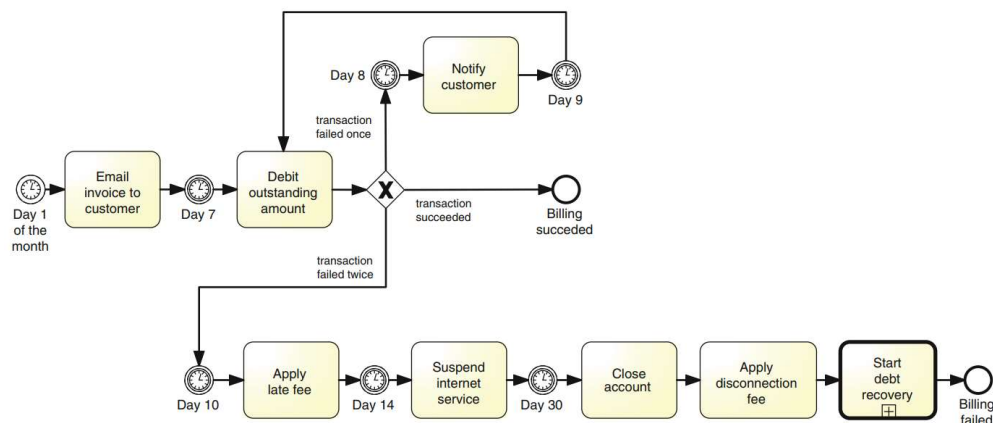
Model the billing process of an Internet Service Provider (ISP).

The ISP sends an invoice by email to the customer on the first working day of each month (Day 1). On Day 7, the customer has the full outstanding amount automatically debited from its bank account. If an automatic transaction fails for any reason, the customer is notified on Day 8. On Day 9, the transaction that failed on Day 7 is re-attempted. If it fails again, on Day 10 a late fee is charged to the customer's bank account. At this stage, the automatic payment is no longer attempted. On Day 14, the Internet service is suspended until payment is received. If the payment is still outstanding on Day 30, the account is closed and a disconnection fee is applied. A debt-recovery procedure is then started.



37

Solution 4.5



38

38

Racing events: Event-based decision

With the XOR-split gateway, a branch is chosen based on conditions that evaluate over available data

→ The choice can be made immediately after the token arrives from the incoming flow

Sometimes, the choice must be delayed until an event happens

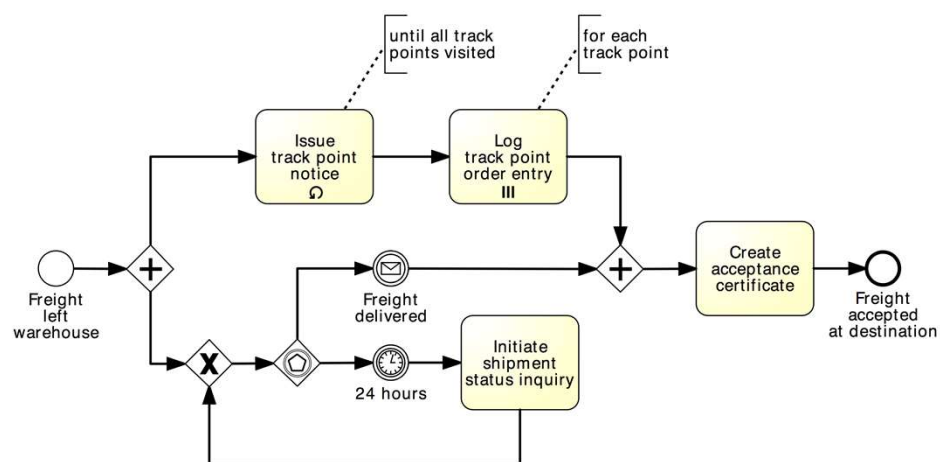
→ The choice is based on a “race” among events

Two types of XOR split:



39







Example: Event-based decision



40

40

Recap: message and timer events

	Start	Intermediate		End
	Catching		Throwing	
Untyped: indicate start point, state changes or final states.				
Message: Receiving and sending messages.				
Timer: Cyclic timer events, points in time, time spans or timeouts.	