MTAT.03.231 Business Process Management

Lecture 4
Process Modelling (part 2)

Marlon Dumas
Professor of Information Systems @ University of Tartu
Co-Founder of Apromore
Course structure

- **Week 2**: Process identification
- **Weeks 3-4**: Process discovery
- **Weeks 5-7**: Process analysis
- **Weeks 8-9**: Process redesign
- **Weeks 10-11**: Process implementation
- **Weeks 12-14**: Process monitoring and controlling
BPMN Main Elements - Recap

Flow Objects
- Gateway
- Event
- Activity

Connections
- Message
- Association
- Flow

Pools & lanes
- Pool
- Lane

Artifacts
- Data Object
- Data Store
BPMN Gateways

**Exclusive (XOR)**
- **Exclusive decision**
  - take one branch
- **Exclusive merge**
  - Proceed when one branch has completed

**Parallel (AND)**
- **Parallel split**
  - take all branches
- **Parallel join**
  - proceed when all incoming branches have completed

**Inclusive (OR)**
- **Inclusive decision**
  - take one or several branches depending on conditions
- **Inclusive merge**
  - proceed when all active incoming branches have completed
Exercise: critique the following model

http://tinyurl.com/nnnfgd5
Anything wrong with this model?
Is this better?
Identifying sub-processes

Acquire raw materials

Ship and invoice
Using the Expanded Sub-Process Notation
Sub-processes

• An activity in a process can invoke a separate sub-process
• Use this feature to:
  1. **Decompose** large models into smaller ones, making them easier to understand and maintain
Guideline: Multi-level modeling

- **Level 1**: value chain
  - Simple linear description of the phases of the process
  - No gateways
  - Each activity chain is a sub-process

- **Level 2+**: expand each activity in the value chain, add incrementally the following:
  - Decisions, handoffs (lanes, pools)
  - Parallel gateways, different types of events
  - Data objects & data stores
  - And as much detail as you need, and no more
Multi-Level Modeling: Example

Level 1: Value Chain

Level 2: Detailed Processes
Sub-processes

• An activity in a process can invoke a separate sub-process

• Use this feature to:
  1. **Decompose** large models into smaller ones, making them easier to understand and maintain
  2. **Delimit** parts of a process that can be:
     • Repeated
     • Interrupted
Block-Structured Repetition
Block-structured repetition: Activity loop

*Activity loop markers* allow us to state that a task or a sub-process may be repeated multiple times.

![Diagram of activity loop markers](image)
More on rework and repetition

Structured cycle (SESE component)
Example: block-structured repetition

- Assign Ministerial Enquiry
- Investigate Ministerial Enquiry
- Finalise Ministerial Response

**Completion condition**

- Until Response is approved

- Ministerial Enquiry received

- Enquiry investigated

**Finalise Ministerial Response**

- Prepare Ministerial Response
- Review Ministerial Response

- Response reviewed

- Must have a decision activity
Parallel repetition: multi-instance activity

The multi-instance activity provides a mechanism to indicate that an activity is executed *multiple times concurrently*

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
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.
Solution: without multi-instance activity

Procurement

- Obtain Quote from Supplier A
- Obtain Quote from Supplier B
- Obtain Quote from Supplier C
- Obtain Quote from Supplier D
- Obtain Quote from Supplier E

Select best quote

Place PO
Solution: with multi-instance activity

For each supplier

Obtain Quote  
Select best quote  
Place PO
Events

In BPMN, events model something instantaneous happening during the execution of a process

Types of event:
- Start
- Intermediate
- End
### BPMN event types

<table>
<thead>
<tr>
<th>Start</th>
<th>Intermediate</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="symbol.png" alt="Untyped Event" /></td>
<td><img src="symbol.png" alt="Untyped Event" /></td>
<td><img src="symbol.png" alt="Untyped Event" /></td>
</tr>
</tbody>
</table>

**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.
Comparison with sending/receiving tasks

- Invoice received
- Invoice sent

Receive invoice

Send invoice

Invoice received

Invoice sent

≠
So, when to use what?

Use message events only when the corresponding activity would simply send or receive a message and do nothing else.
## Temporal events

<table>
<thead>
<tr>
<th>Start</th>
<th>Intermediate</th>
<th>End</th>
</tr>
</thead>
</table>

**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)
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.
Recap: Message and Timer events

- **Start**
  - Catching
  - Process starts upon message received
  - Process starts when time event occurs

- **Intermediate**
  - Catching
    - Message received during the process
  - Throwing
    - Message sent during the process
    - Time event occurred (to model delay)

- **End**
  - Throwing
    - Process ends upon message sent
Data-based vs. event-based choices

• In an XOR-split gateway, one branch is chosen based on expressions evaluated over available data
  → Choice is made immediately when the gateway is reached
• Sometimes, the choice must be delayed until something happens
  → Choice is based on a “race between events”
• BPMN distinguishes between:
  • Exclusive decision gateway (XOR-split)
  • Event-based decision gateway
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
Choices outside our control...

A restaurant chain submits a purchase order (PO) to replenish its warehouses every Thursday. The restaurant chain’s procurement system expects to receive either a “PO Response” or an error message. However, it may also happen that no response is received at all due to system errors or due to delays in handling the PO on the supplier’s side. If no response is received by Friday afternoon or if an error message is received, a purchasing officer at the restaurant chain’s headquarters should be notified. Otherwise, the PO Response is processed normally.
Solution: event-driven XOR split

Stock replenishment
Exception Handling
Exceptions are events that deviate a process from its “normal” course

The simplest form of exception is to notify that there is an exception (negative outcome)

This can be done via the Terminate end event: it forces the whole process to abort (“wipes off” all tokens left behind, if any)

Abortion (terminate event)
Example: terminate event

Abort the process by removing all tokens...
Exception handling

Handling exceptions often involves stopping a sub-process and performing a special activity.

Types of exceptions for an activity (task/sub-process) in BPMN:

External: something goes wrong outside the process, and the execution of the current activity must be interrupted. Handled with the Message event.

Internal: something goes wrong inside an activity, whose execution must thus be interrupted. Handled with the Error event.

Timeout: an activity takes too long and must be interrupted. Handled with the Timer event.

All these events are catching intermediate events. They stop the enclosing activity and start an exception handling routine.
Let’s extend our PO handling process

**PO handling**

A 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.
Let’s extend our PO handling process

A PO change request may be received anytime after the PO is registered. This request includes a change in quantity or line items. When such a request is received, any processing related to the PO must be stopped. The PO change request is then registered. Thereafter, the process proceeds as it would do after a normal PO is registered. Further, if the customer sends a PO cancellation request after the PO registration, the PO processing must be stopped and the cancellation request must be handled.
Solution: exception handling

PO handling
**Internal exception: error event**

**Error Event** – Indicates an error: the “end” version generates an error event while the “catching intermediate” version consumes it when attached to the boundary of an activity.

Must be attached to the activity’s boundary.
Example: internal exception

PO handling

Consider again our “PO Handling process” example with the following extension: if an item is not available, any processing related to the PO must be stopped. Thereafter, the client needs to be notified that the PO cannot be further processed.
Solution: internal exception

PO handling

Must catch an error event thrown from **within** the same activity

Throwing and catching error events must have the **same** label
Example: activity timeout

Order-to-transportation quote

Once a wholesale order has been confirmed, the supplier transmits this order to the carrier for the preparation of the transportation quote. In order to prepare the quote, the carrier needs to compute the route plan (including all track points that need to be traversed during the travel) and estimate the trailer usage.

By contract, wholesale orders have to be dispatched within four days from the receipt of the order. This implies that transportation quotes have to be prepared within 48 hours from the receipt of the order to remain within the terms of the contract.
Solution: activity timeout
The customer may send a request for address change after the PO registration. When such a request is received, it is registered, without further action.
Non-interrupting boundary events

Sometimes we may need to trigger an activity in parallel to the normal flow, i.e. without interrupting the normal flow.

This can be achieved by using non-interrupting boundary events.
Solution: non-interrupting boundary events

PO handling
Summary

- In this lecture we have learned about:
  - BPMN sub-processes
  - Repetition markers: loop marker and parallel multi-instance marker
  - Events: timer, message and error events
  - Event-based choice gateway
  - Boundary events: interrupting and non-interrupting
  - Error events (throw and catch)
And once I’ve got a model, what’s next?

Process analysis techniques:

- Added-value and waste analysis
- Root-cause analysis
- Flow Analysis
- Queuing Analysis
- Process Simulation