MTAT.03.231
Business Process Management

Lecture 4 – Advanced Process Modeling

Marlon Dumas

marlon.dumas @t ut . ee
Business Process Lifecycle

- Process identification
- Process architecture
- Process discovery
- As-is process model
- Conformance and performance insights
- Process monitoring and controlling
- Executable process model
- Process implementation
- To-be process model
- Process analysis
- Insights on weaknesses and their impact
- Process redesign
1. Introduction
2. Process Identification
3. Essential Process Modeling
4. Advanced Process Modeling
5. Process Discovery
6. Qualitative Process Analysis
7. Quantitative Process Analysis
8. Process Redesign
9. Process Automation
10. Process Intelligence
BPMN Main Elements - Recap

Flow Objects
- Gateway
- Event
- Activity

Connections
- Message
- Association
- Flow

Pools & lanes

Artifacts
- Data Object
- Data Store
### BPMN Gateways

<table>
<thead>
<tr>
<th>Exclusive (XOR)</th>
<th>Parallel (AND)</th>
<th>Inclusive (OR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Exclusive decision&lt;br&gt;take one branch&lt;br&gt;• Exclusive merge&lt;br&gt;Proceed when one branch has completed</td>
<td>• Parallel split&lt;br&gt;take all branches&lt;br&gt;• Parallel join&lt;br&gt;proceed when all incoming branches have completed</td>
<td>• Inclusive decision&lt;br&gt;take one or several branches depending on conditions&lt;br&gt;• Inclusive merge&lt;br&gt;proceed when all active incoming branches have completed</td>
</tr>
</tbody>
</table>
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
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
Guideline: Multi-level modeling (cont.)

• At each level, decompose according to:
  • Logical milestones towards achieving the outcome of the process
  • Major objects used in the process
• Decompose until processes are of “reasonable” size
  • e.g. up to 20 nodes (tasks+events+gateways) per model
Value chain with sub-processes

1. Purchase Request process
   - Purchase Request
   - Check purchase request for 1st approval
     - approved
       - Send approved request to requestor
     - rejected
       - Consider re-submission
   - Make copy of purchase request
   - Forward to purchase department
   - Approved Purchase Request

2. Approval +
   - Approval
   - Check purchase request for 2nd approval
     - approved
       - Send approved request to requestor
     - rejected

3. Purchase Order +
   - Purchase Order
   - forwarded to purchase department
   - Rejected

4. Goods Receipt +
   - Goods Receipt
   - Forward to purchase department
   - Approved Purchase Request

5. Invoice Verification +
   - Invoice Verification

[Flowchart diagram with process steps and decision points]
Fragment of SCOR’s process hierarchy

Level 3

Process Inquiry and Quote

Receive and Validate Order

…

Level 4

Receive Order

Enter Order

Check Credit

…

Level 5

Access Credit Record

Credit Available?

Clear Order

Contact customer account rep.

…

APICS SCOR framework: http://www.apics.org/
Side Note: Bizagi Milestones (non-standard BPMN)

For details see: [http://tinyurl.com/pp5reuo](http://tinyurl.com/pp5reuo)
Side Note: non-BPMN value chain notation (e.g. Signavio, ARIS)
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. Share common fragments across multiple processes
Shared sub-process

Home loan application received → Register home loan application → Check home loan application → low liability → Reject home loan → high liability → Approve home loan → Sign loan → Home loan application completed

Student loan application received → Register student loan application → Check debts → No debts → Approve student loan → Sign loan → Student loan application completed → No debts → Student loan application completed → Sign loan → Student loan application completed
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. Share common fragments across multiple processes
  3. Delimit parts of a process that can be:
     • Repeated
     • Interrupted
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.
More on rework and repetition

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

Finalise Ministerial Response

- Enquiry investigated
- Prepare Ministerial Response
- Review Ministerial Response
- Response reviewed

Completion condition

- Until Response is approved
- Must have a decision activity
Exercise

After a claim is registered, it is examined by a claims officer. The claims officer writes a “settlement recommendation”. This recommendation is checked by a senior claims officer who may mark the claim as “OK” or “Not OK”. If the claim is marked as “Not OK”, it is sent back to the claims officer and the examination is repeated. If the claim is marked as “OK”, the claims officer notifies the settlement to the customer.
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

1. Obtain Quote from Supplier A
2. Obtain Quote from Supplier B
3. Obtain Quote from Supplier C
4. Select best quote
5. Place PO
6. Obtain Quote from Supplier D
7. Obtain Quote from Supplier E
Solution: with multi-instance activity

Procurement

For each supplier

cardinality

... → Obtain Quote [+1] → Select best quote → Place PO → ...

29
Exercise

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.
Events
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>
</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

Invoice sent

Invoice received

Receive invoice

Send invoice

Send invoice

Receive invoice
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)
Example

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.
Recap: Message and Timer events

Start
Catching
process starts upon message received

Intermediate
Catching
message received during the process

Throwing
message sent during the process

End
Throwing
process ends upon message sent

Start
Catching
process starts when time event occurs

Intermediate
Clock
time event occurred (to model delay)

End
Throwing
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
Choices outside our control...

Stock replenishment
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.
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:

- **Data-driven XOR-split**
- **Event-driven XOR split**
Solution: event-driven XOR split

Stock replenishment
Exercise

In the context of a claim handling process, it is sometimes necessary to send a questionnaire to the claimant to gather additional information. The claimant is expected to return the questionnaire within five days. If no response is received after five days, a reminder is sent to the claimant. If after another five days there is still no response, another reminder is sent and so on until the completed questionnaire is received.
Exception handling
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.

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 cancelation request after the PO registration, the PO processing must be stopped and the cancelation request must be handled.
Abortion (terminate event)

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).
Example 1: terminate event

Signal the negative outcome...

- Every Thursday: Submit replenishment order
  - Order response received
  - Error message received
  - Friday afternoon

- Handle order response
- Order fulfilled
- Notifying purchasing officer
- Order not fulfilled
Example 2: 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

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 cancelation request after the PO registration, the PO processing must be stopped and the cancelation request must be handled.
Solution: exception handling

PO handling

- **PO Received**
  - **Register PO**
  - **Check Availability**
  - **Send PO Response**
  - **Response sent**
  - **PO fulfilled**

- **Handle PO**
  - **Next working day**
  - **weekend/holiday**

- **PO Change received**
  - **Register PO Change**

- **PO Cancel received**
  - **Handle PO Cancellation**
  - **PO canceled**
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.
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

Throwing and catching error events must have the **same** label

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

Solution: internal exception

PO handling

Throwing and catching error events must have the **same** label

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

Order-to-transportation quote
Exercise

When a claim is received, it is registered. After registration, the claim is classified leading to two possible outcomes: simple or complex. If the claim is simple, the policy is checked. For complex claims, both the policy and the damage are checked independently.

A possible outcome of the policy check is that the insurance is invalid. In this case, any processing is cancelled and a letter is sent to the customer. In the case of a complex claim, this implies that the damage checking is cancelled if it has not yet been completed.
The customer may send a request for address change after the PO registration. When such a request is received, it is just 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.

Must be attached to the activity’s boundary
Solution: non-interrupting boundary events

PO handling

- **Register PO**
  - PO Received
  - PO Registered
  - Handle PO
    - Next working day
    - weekend/holiday
  - Check Availability
    - Items not available
    - Items available
    - Response sent
    - PO fulfilled
  - Send PO Response
    - Response sent
    - PO fulfilled
  - Handle PO Change
  - PO Change received
  - Register PO Change
  - Address change received
  - Update customer address
  - Customer address updated
  - Items not available
  - Items available
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