BPMN 2: Formal Semantics of Selected Elements
And Lessons Learned

Felix Kossak
Overview

- A (somewhat) alternative modelling approach
- Event-based gateways
  - Examples for analysing a specification / standard for ambiguity and contradictions
- “Deontic BPMN”
  - Example for adaptation of initial ground model
- [Possible “Bonus”: A general communication concept
  - Example for “non-conservative refinement”]
- Summary
An alternative modelling approach


Considerations:
- Preventing errors vs empowering users
- Independent processes vs centralised control
- Staying close to an accepted standard vs the perfect alternative
Preventing errors vs empowering users

- The BPMN standard is designed to make “everything” possible
- Enable intuitive diagrams
- Designers are responsible for correctness of diagrams
Independent processes vs centralised control

- Closer to the BPMN standard
- Simulation of different, concurrent instances
- Resolution of event targets within the model

- However, centralisation might not always be feasible

Technical solution:
- Use of “coloured” tokens (also helpful with sub-processes)
Analysing BPMN 2.0: Event-based gateways

Lessons for analysing specifications and standards

The following is largely taken from

“Event-based gateways: Open questions and inconsistencies”
(BPMN’2012)
Examples for analysing BPMN 2.0: Event-based gateways
Semantic issues with event-based gateways

Discussed at BPMN’2012

Questions:

- Activation of events or receive tasks in the configuration of the event-based gateway
- Exactly when is a process instantiated by an instantiating event-based gateway?
- Starting sub-processes using event-based gateways
- On the meaning of ‘waiting’ for process instances started by parallel event-based gateways
Q I: Activation of Events or Receive Tasks

Q: When should an event-based gateway send tokens to its event (or receive task) nodes?

- Repercussions for the timing of process instantiation (for instantiating gateways)

We start with non-instantiating gateways.

The following quotations refer to the BPMN 2.0 standard.
Activation of Events

“The choice of the branch to be taken is deferred until one of the subsequent Tasks or Events completes” [13.3.4]

- Do not send a token until event trigger has been caught (?)

“Waiting starts when the Intermediate Event is reached” [13.4.2]

- No trigger can be caught until a token has reached the event node (?)

© Software Competence Center Hagenberg GmbH
Activation of Events: Alternative 1

if controlCondition(gate) then
  foreach seqFlow in outgoingSeqFlows(gate) do
    ProduceToken(seqFlow, instance(enablingToken(gate)))

Disadvantages:

- For every branch not chosen, the token must be removed again (suggested by “Withdrawn” state [Fig. 13.2])
  - Against intuitions (exceptional behaviour)
- Gateway loses control:
  What if different triggers are caught simultaneously?
Activation of Events: Alternative 2

if controlCondition(gate) and eventCondition(gate) then
    let seqFlow = selectSF(outgoingSequenceFlows(gate), caughtTrigger(gate)) do
        ProduceToken(seqFlow, instance(caughtTrigger(gate)))
        PassTrigger(caughtTrigger(gate), targetNode(seqFlow))

Disadvantages:
- Gateway needs to know everything about its event nodes
- “PassTrigger” is intuitively redundant
- Catching trigger by event node a pure formality
if controlCondition(gate) and eventCondition(gate) then
  let eventNode = selectNode(eventNodes(gate), caughtTrigger(gate)) do
    ProduceToken(outgoingSequenceFlow(eventNode), instance(caughtTrigger(gate))
    ConsumeTrigger(caughtTrigger(gate))

Disadvantages:

- Gateway has mixed, complex semantics
- Event nodes reduced to mere symbols without semantics
Activation of Events: Summary

Alternative 3 ("single node") selected by Stephen White [BPMN’2012]

Crucially:
- No alternative is fully compatible with the standard!
Q II: When Is a Process Instantiated?

Now suppose that the e-b. gateway is instantiating:
When Is a Process Instantiated? – Alternative 1

```
if controlCondition(gate) then
    foreach seqFlow in outgoingSeqFlows(gate) do
        ProduceToken(seqFlow, instance(caughtTrigger(gate)))
```

This variant can only work, if

“controlCondition” is true by default!

This basically means that the process always has an instance waiting to catch a trigger

- no matter if there ever will be a trigger!
If controlCondition(gate) and eventCondition(gate) then
let seqFlow = selectSF(outgoingSequenceFlows(gate), caughtTrigger(gate)) do
  ProduceToken(seqFlow, instance(caughtTrigger(gate)))
  PassTrigger(caughtTrigger(gate), targetNode(seqFlow))

In this case, controlCondition may be true by default (as for a start event) and we can create an instance once the eventCondition has become true.

Difference to multiple start events? (Single point of entry?)
When Is a Process Instantiated? – Alternative 3

```plaintext
if controlCondition(gate) and eventCondition(gate) then
  let eventNode = selectNode(eventNodes(gate), caughtTrigger(gate)) do
    ProduceToken(outgoingSequenceFlow(eventNode), instance(caughtTrigger(gate))
    ConsumeTrigger(caughtTrigger(gate))
```

Similar than before –
the gateway is now basically a complex start event node.

- Intuitive?
- And again: Difference to multiple start events?
Starting Sub-Processes (2)

But:

An instantiating event-based gateway is not supposed to get a token!

Moreover [10.4.2]:

“... the implicit Start Event ...
SHALL NOT have a trigger.”
Q III: Starting Sub-Processes with Event-Based Gateways

May an instantiating e-b. gateway instantiate a sub-process?

[13.2.4] Sub-Processes have

- either “a unique empty Start Event” OR
- “Activities and Gateways without incoming Sequence Flows”
  - These will “get a token”.

[13.4.1] “... the only scenario where a Gateway can exist without incoming Sequence Flows” is an instantiating Event-Based Gateway.
Starting Sub-Processes (3)

Solution [Stephen White, BPMN’2012]:

The kind of gateway which may start a sub-process is *not* an event-based gateway.

But

- This is not clear from the standard
- and actually contradicting the standard as well.
Q IV: The Meaning of ‘Waiting’

The parallel event-based gateway:
The Meaning of ‘Waiting’ (2)

“... the Process then waits for the other Events to arrive” [13.1]

Now if the upper event was triggered, can task T1 already be started?
If T1 could be started while T2 is still waiting for its event, then we could actually model something new: *asynchronously* started branches.

But we actually cannot [White, BPMN’2012].

But then do we really need this?
Do we really need this?

How about:
What we can learn from this exercise

- Certain inconsistencies or gaps can only be detected if all relevant requirements are “in a single mind”
- Certain inconsistencies or gaps can only be detected if a formal model is built
- Certain inconsistencies or gaps can only be detected if a complete formal model is built (incl. all requirements)
- **Structure** of the *presentation* of a specification / standard has to consider different aspects than modelling
  - Requirements → single model → different presentations
- Later RFCs take much more effort to handle (and are less likely to be actually considered)
But now for something nice.
“Deontic BPMN”

Work by Christine Natschläger

- *Extending BPMN with Deontic Logic* (Dissertation / Logos, 2012)
- “Deontic BPMN: A Powerful Extension of BPMN with a Trusted Model Transformation” (SoSyM, 2013)

**Task:** Adapt the ground model to incorporate a new requirement.
Order Process: Standard BPMN Diagram
Order Process: “Deontic BPMN” Diagram
“Deontic BPMN”

Deontic classification of (user) tasks

- O(Task)
- P(Task)
- F(Task)
- X(Task)

- Obligatory, permissible (optional), forbidden, alternative
- Classification can be *conditional*

In classical diagrams hidden in the structure

- Goal (1): make it visible at first sight
- Goal (2): simplify diagrams
WorkflowTransition (Standard)

WorkflowTransition(node) =
   if eventCondition(node) and controlCondition(node) and dataCondition(node) and resourceCondition(node) then
   EventOperation(node)
   ControlOperation(node)
   DataOperation(node)
   ResourceOperation(node)
WorkflowTransition (simplified)

Set PerformanceOperation :=
    \{ EventOperation, ControlOperation, DataOperation, ResourceOperation \};

controlFlowCondition := controlCondition
performanceCondition := dataCondition and resourceCondition and eventCondition

Then:
WorkflowTransition(node) =
    if controlFlowCondition(node) and performanceCondition(node) then
        PerformanceOperation(node)
WorkflowTransitionDeontic: The problem
WorkflowTransitionDeontic

monitored choiceCondition : FlowNodes → BOOLEAN

WorkflowTransitionDeontic(node) =
  if choiceCondition(node) then
    if controlFlowCondition(node) and
      performanceCondition(node) then
      PerformanceOperation(node)
    else
      if controlFlowCondition(node) then
        ControlFlowOperation(node)
  else
    PerformanceOperation(node)

where ControlFlowOperation simply passes a token on
WorkflowTransitionDeontic / rewritten

\[
\text{WorkflowTransitionDeontic}(node) = \\
\begin{array}{l}
\text{if controlFlowCondition}(node) \text{ then} \\
\text{if choiceCondition}(node) \text{ then} \\
\quad \text{if performanceCondition}(node) \text{ then} \\
\quad \quad \text{PerformanceOperation}(node) \\
\text{else} \\
\quad \text{skip} \\
\text{else} \\
\quad \text{ControlFlowOperation}(node)
\end{array}
\]
Semantics of gateways in deontic diagrams

- Formally unchanged

However, we need the following constraint to be heeded by modellers:

\[
\text{foreach sequenceFlow in }
\text{outgoingSequenceFlows(gateway) holds}
\text{conditionExpression(sequenceFlow) =}
\text{f(choiceCondition(target(sequenceFlow)))}
\]

for some reasonable function \( f \)
(default: identity)
Deontic BPMN example: Lessons

- Specifications are rarely fixed forever
- It is not practically possible to try to foresee all sorts of changes
- In the given example, a conservative refinement sufficed
  - But not always!
An example for refinement with changes: An enhanced communication concept

General idea: use "event pools"
- Every activity has an "inbox"
- An inbox is a view on different event pools
An enhanced communication concept

Literature:

- Fleischmann et al.: *Subject-Oriented Business Process Management* (Springer, 2012)
- “An enhanced communication concept for business processes“ (EMISA 2015)
Enhanced communication concept:
Required changes in model (1)

Original model:

monitored availableTriggers: FlowNodes → Set

New model:

derived availableTriggers: FlowNodes → EventPools

Original model:

if trigger ∈ availableTriggers(flowNode) then ...

New model:

if containsTrigger(
    availableTriggers(flowNode), trigger) then ...
Enhanced communication concept: Required changes in model (2)

Original:

```
rule RemoveTrigger(trigger)
```

New:

```
rule RemoveTrigger(trigger, instance)
```

Example: a signal

- Do not remove – others may still need it!
- But remember which process and instance already caught it
Enhanced communication concept: Required changes in model (3)

Additional problem:

“RemoveTrigger” does not always remove a trigger!
Summary

Example BPMN
Using formal methods (ASMs) in:

- Analysis of
  - Requirements
  - Specifications
  - Standards
- Change management
  For formal specifications