Business-Driven Software Engineering
Lecture 5 – Business Process Model and Notation

Jochen Küster jku@zurich.ibm.com
Agenda

- BPMN Introduction
- BPMN Overview
- BPMN Advanced Concepts
- Introduction to Syntax and Semantics of Modeling Languages
- Summary and References
BPMN Introduction
Business Process Model and Notation (BPMN)

- Business Process Model and Notation (BPMN) is an OMG standard for modeling business processes
  - Previously also called Business Process Modeling Notation
- Current version: BPMN 2.0 (final adopted specification)
- Previous versions: BPMN 1.2, BPMN 1.1, BPMN 1.0
- IBM Process Management Technologies Group in Zurich has participated in standardization of BPMN 2.0
BPMN Concepts

- **Flow Objects:**
  - for modeling nodes in a business process
  - activities, gateways and events

- **Connecting Objects:**
  - sequence flow and message flow for connecting nodes, associations for additional information

- **Pools and Lanes**
  - for partitioning sets of activities for business-to-business interaction

- **Artifacts**
  - further elements such as data objects
BPMN Overview
BPMN Flow Objects – Tasks

- **Task**

Different types of tasks:
- General task
- Service task
- User task
- Script task
- Receive task
- Send task
- Manual task
- Business rule task

- **Loop Task**

- Loop condition
- while / repeat until

- **Multi-Instance Tasks**

- Numeric expression for determining repetition times
- sequential/parallel execution of tasks
- used for implementing Multiple Instance Patterns
BPMN Sub-Processes

- A Sub-Process is a compound activity that is included within a Process. It is compound in that it can be broken down into a finer level of detail (a Process) through a set of sub-activities.

[Diagram A: Collapsed Sub-Process]

[Diagram B: Expanded Sub-Process]

1. Receive Credit Report
2. Approval
3. Approve?
4. Include History of Transactions (No)
5. Include Standard Text (Yes)
BPMN Flow Objects - Subprocesses

- **Collapsed Subprocess**
  
  ![Collapsed Subprocess Diagram]

- **Expanded Subprocess**
  
  ![Expanded Subprocess Diagram]

- **Loop**
  
  ![Loop Diagram]

- **Ad-Hoc Subprocess**
  
  `- Sequential/parallel`
  
  ![Ad-Hoc Subprocess Diagram]

- **Multiple Instance Loop**
  
  ![Multiple Instance Loop Diagram]
BPMN Flow Objects - Gateways

- Gateway

- AND, XOR, OR, Complex

- Event-based
Inclusive Gateways are Decisions where there is more than one possible outcome

They are usually followed by a corresponding merging Inclusive Gateway

Inclusive Gateways realize the Or Split and Or Join Control Flow Pattern
BPMN Flow Objects – Event-Based Gateways

- Event-based gateway represents a branching point where alternatives are based on events
- Events can be modeled as receive events
- Event-based gateways realize the Deferred Choice Control Flow pattern
BPMN Flow Objects - Connectors

- **A Sequence Flow** is used to show the order that activities will be performed in a Process.

- **A Message Flow** is used to show the flow of messages between two entities that are prepared to send and receive them.

- **An Association** is used to associate data, information and artifacts with flow objects.
Default Sequence Flow

- A Sequence Flow that exits an Exclusive or Inclusive Gateway may be defined as being the default path
  - A hatch mark at the line beginning shows the default Sequence Flow
- The default path is chosen only if all the other conditions of the Gateway are False
BPMN Flow Objects – Selected Events

- events occurs during the course of the process and affects the flow of the process
- In BPMN 1.1/2.0 distinction between catching and throwing events, important for intermediate events
- Flow dimension: start, intermediate, end
- Type dimension: message, timer, error, …
BPMN Flow Objects – Selected Events

<table>
<thead>
<tr>
<th>start</th>
<th>message</th>
<th>timer</th>
<th>error</th>
<th>link</th>
<th>signal</th>
<th>conditional</th>
<th>compensation</th>
<th>cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mediate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Intermediate events placed into normal flow:
  - Catching or throwing an event
- Intermediate events placed at the boundary of an activity:
  - Exception flow, catching of an event
- Not all combinations exist
- Start events can be thought of catching, end events can be thought of throwing
- New in BPMN 2.0: non-interrupting events
Example: Message and Timer Event

- Message events for sending and receiving messages
- Timer events for timeouts and exception handling
Message and Timer Event - Traces

<start,send(Send Ballot), receive(Receive Votes), IT>

<start,RC,RC,CC,end>, <start,RC,GW, timeout, EH, end>
Example: Error Event

- Error event catches any error that occurs during Grant Claim
- Error event as end generates an error
- Possible trace:
  - \(<RC,GC,event(error),EH>, <RC,GC,CC>\)
Example: Link Event

- Link Events can be used for Off-Page connectors
- Link Events can be used as “Go-To” objects
- Note: Shorthand for merge for flow incoming to “Close Order”
- Note: Shorthand for AND-Split after “Fill Order”
Example: Signal Event

- Sending of a signal can be caught by another process or within the process
- Other than a message that has a specific source and target
- Catching of a signal can only take place as a boundary event
Example: Conditional Event

- Conditional event is triggered when a condition becomes true
- Settle Claim is started once the Customer state is in “Ready”
- Conditional event implements data-based task trigger pattern
Example: Compensation Event

- Compensation Event in normal flow throws a compensation event
- Compensation Event at boundary will be triggered by a compensation event
- Compensation activities compensate the work done in the source activity
- Complex logic for broadcasting compensation events
- Only completed activities can be compensated
Compensation Event - Traces

- Trace without compensation:
  - <start,RC,start(SC),N,M,end(SC),CC,end>

- Trace with compensation:
  - <start,RC,start(SC),N,M,end(SC),throw(compensate),CS,end>
BPMN Flow Objects – Pools

- Pools represent participants in the process (business entity or business role)
- Boundaries can be suppressed if there is only one pool
- Sequence Flow cannot cross the boundaries of a Pool
- Lanes subdivide pools
- Sequence flow can cross boundaries of lanes
BPMN Flow Objects – Sequence Flow and Message Flow

- Sequence flow shows the order of activities in a process model
- Message flow shows the messages exchanged between activities or pools
BPMN Flow Objects – White Box Pool

- Sequence flow shows the order of activities in a process model
- Message flow shows the messages exchanged between activities or pools
BPMN Advanced Concepts
BPMN Data Objects

- Data Objects are Artifacts that are used to show how data and documents are used within a Process
- Data Objects are used to define inputs and outputs of activities
- Data Objects can be given a “state” that shows how a document may be changed or updated within the Process
Intermediate Events attached to the boundary of an activity represent triggers that can interrupt the activity.

All work within the activity will be stopped and flow will proceed from the Event.

Timer, Errors, Messages, etc. can be Triggers.
Transactions

- A Transaction is a Sub-Process that has a double border.
- Transactions are supported by a transaction protocol (e.g., WS-Transaction).
- Normal Outgoing Sequence Flow represents the path to follow a successful completion.
- A Cancel Intermediate Event represents the path to follow a cancelled completion (compensation is performed).
- An Error Intermediate Event represents the path to follow a transaction hazard (but no compensation is performed).
Cancel Events and Transactions

- Cancel Intermediate Events must be attached to a transaction subprocess boundary
- Cancel Intermediate Event is triggered by a Cancel End Event inside the Transaction or by a Transaction Protocol Cancel message
Transaction Example with Compensation

- Book Flight
- Book Hotel
- Cancel Flight
- Send Hotel Cancellation
- Failed Bookings (Hazards)
- Successful Bookings

Transaction
BPMN 2.0 improvements (over previous versions)

- Standardized execution semantics
  - in English text

- Standardized serialization for diagram interchange
  - based on a common metamodel

- Mapping of a BPMN subset to BPEL
  - demonstrates alignment with existing technologies and standard

- Some new modeling elements
  - Non-interrupting event, choreography support, referenced call activity
Introduction to Syntax and Semantics of Modeling Languages
Syntax and Semantics of (Modeling) Languages

- Syntax: Form of the "words" in a language
- Semantics: Meaning of the "words" in a language, e.g. how to execute program statement

- Programming languages:
  - Syntax: Context-free grammar
  - Semantics: Denotational semantics, attribute grammars, ..

- Visual Modeling Language:
  - Syntax: Metamodel + OCL constraints
  - Semantics: often informally described in the language specification
Abbreviated Meta Model for Process Models (BPMN 2.0)

- Meta model defines the abstract syntax of process models
Instantiation of the Meta Model

[Diagram showing process elements and exceptions]

- Process
- FlowElementsContainer
- FlowElement
- FlowNode
- SequenceFlow
- Exception Handling
- Grant Claim
- Close Claim
- Reject Claim
- Register Claim
Informally described in the standard:

- A **Process** is instantiated when one of its **Start Events** occurs. Each occurrence of a **Start Event** creates a new **Process Instance** unless the **Start Event** participates in a **Conversation** that includes other **Start Events**. In that case, a new **Process instance** is only created if none already exists for the specific **Conversation** (identified through its associated correlation information) of the **Event** occurrence. Subsequent **Start Events** that share the same correlation information as a **Start Event** that created a **Process instance** are routed to that **Process instance**. Note that a **global Process** must neither have any empty **Start Event** nor any **Gateway** or **Activity** without **incoming Sequence Flow**. An exception is the **Event Gateway**.

- A **Process** can also be started via an **Event-Based Gateway** that has no **incoming Sequence Flow** and its Instantiate flag is **true**. If the **Event-Based Gateway** is **exclusive**, the first matching **Event** will create a new **instance** of the **Process**. The **Process** then does not wait for the other **Events** originating from the same **Event-Based Gateway** (see also semantics of the **Event-Based Exclusive Gateway** on page 437). If the **Event-Based Gateway** is **parallel**, also the first matching **Event** creates a new **Process instance**. However, the **Process** then waits for the other **Events** to arrive. As stated above, those **Events** must have the same correlation information as the **Event** that arrived first. A **Process instance** completes only if all **Events** that succeed a **Parallel Event-Based Gateway** have occurred.
Semantics of BPMN 2.0 (continued)

- An Activity is *Ready* for execution if the required number of *Tokens* is available to activate the Activity. The required number of *Tokens* (one or more) is indicated by the attribute StartQuantity. If the Activity has more than one *Incoming Sequence Flow*, there is an implied *Exclusive Gateway* that defines the behavior.

- When some data InputSet becomes available, the Activity changes from *Ready* to the *Active* state. The availability of a data InputSet is evaluated as follows. The data InputSets are evaluated in order. For each InputSet, the data inputs are filled with data coming from the elements of the context such as *Data Objects* or Properties by triggering the input *Data Associations*. An InputSet is *available* if each of its *required* data inputs is available. A data input is *required* by a data InputSet if it is not optional in that InputSet. If an InputSet is available, it is used to start the Activity. Further InputSets are not evaluated. If an InputSet is not available, the next InputSet is evaluated. The Activity waits until one InputSet becomes available.

- Complicated and subject to interpretation of the reader
- Now consolidated and defined in BPMN spec
Example: Token semantics

- Start Event generates a new token t1
- Token t1 flows to Register Claim
- Register Claim is instantiated and consumes the token
- ...

Diagram:

```
Start Event -> Register Claim
  |          |
  |          |
  v          v
Grant Claim -> Close Claim
  |          |
  |          |
  v          v
Reject Claim
```

Semantics and Scope of Events

Throwing of an event creates a trigger that carries information.

- **Publication**: Trigger can be received by any catching events in the scope where the trigger is published.

- **Direct resolution**: Timer and conditional events are directly resolved where they are activated.

- **Propagation**: Trigger is forwarded to the innermost scope which has an event capable to catch the trigger.

- **Cancellation**: Directed towards a process or activity instance. Terminates all running instances and compensates completed activities.

- **Compensation**: Compensation of a subprocess calls all compensations of enclosed activities.
Compensation is triggered using a throw compensation event

- The activity which needs to be compensated is referenced (optional if clear from context)
- If no activity is referenced the compensation event is broadcast inside the subprocess or process
- Compensation is triggered synchronously, throw compensation event waits for completion (exceptions are possible)
- Default compensation ensures that compensation activities are performed in reverse order of the execution of original activities

- Compare BPMN 2.0 pages 280-282, 405-406, 241
Semantics of Inclusive Gateway

The **Inclusive Gateway** is activated if

- At least one incoming sequence flow has at least one *Token* and

- for each empty incoming sequence flow, there is no *Token* in the graph anywhere upstream of this sequence flow, i.e., there is no directed path (formed by Sequence Flow) from a *Token* to this sequence flow unless
  - the path visits the inclusive gateway or
  - the path visits a node that has a directed path to a non-empty incoming sequence flow of the inclusive gateway.

- Upon execution, a *Token* is consumed from each incoming sequence flow that has a *Token*. A *Token* will be produced on some of the outgoing sequence flows. In order to determine the outgoing sequence flows that receive a *Token*, all conditions are evaluated. The evaluation does not have to respect a certain order.

- For every condition, which evaluates to true, a *Token* must be passed on the respective sequence flow. If and only if none of the conditions evaluates to true, the *Token* is passed on the default sequence flow. In case all conditions evaluate to false and a default flow has not been specified, the inclusive gateway throws an exception.
BPMN semantics: Inclusive Gateway

- In this case, second inclusive gateway waits
- Inclusive gateways still problematic, can lead to deadlocks
Syntactic and Semantic Correctness

- Syntactic correctness of the process model
  - Is the process model syntactically correct?
  - Is the process model an instance of the meta-model of the language
  - Usually checked by the modeling tool

- Semantic correctness of the process model
  - Is the process model correct with regards to the semantics of the modeling language?
  - Is the process model a correct model with regards to reality?
  - Does it contain deadlocks, does it terminate?
  - Can be checked by following best-practices of process modeling
  - Can be validated by using simulation
Summary of Lecture and References

- Presentation of core BPMN concepts along examples
- Discussion of syntax definition for visual modeling languages
- Introduction to defining informal semantics of visual modeling languages

Further Reading:
- BPMN 2.0 specification