

# Towards a BPMN semantics using UML models

Oana Nicolae<sup>1</sup>   Mirel Cosulschi<sup>2</sup>   Adrian Giurca<sup>1</sup>  
Gerd Wagner<sup>1</sup>

<sup>1</sup>Department of Internet Technology  
Brandenburg Technical University at Cottbus, Germany



<sup>2</sup>Department of Computer Science  
University at Craiova, Romania



## Actual context ...

- Standard initiatives claim to be a consequence of the BPM market maturity.
- Specifications do NOT succeed to provide a well defined semantic of standards, but only an informal description of the languages.
- Open/Proprietary tools do NOT succeed to generate executable code.

### ... and Consequences

- Open/Proprietary tools that implement the Specifications do NOT make a clear distinction between the Specification they implement and their own requirements. This results in different alternatives of code.
- A common understanding of the concepts is then needed.

## Our Solution

- To follow the Language Driven Development (LDD) techniques based on metamodeling - the process of developing platform agnostic models.

### LDD techniques

- Comprise transformations from high-level modelling languages (e.g. BPMN) to DSLs (e.g. WS-BPEL).
- Brought the usage of term PAIS (i.e. execution based on a business process model)
- In the context of OMG's MDA, MOF represents the metamodeling language.
- Actual UML/MOF metamodels focus only on the abstract syntax.
- UML do not provide an executable model but it defines a rigorous and precise semantics to which any executable program language must conform.

## BPMN Generalities

- OMG standardised and released BPMN (now at version 1.1)
- Is recognised as de-facto standard in the area of Business Process Modelling
- Visual but, non-formalised language => lack of a formal behavioural semantics
- Non-executable, is defined only as a "documentation" for business processes
- Lack of a meta-model => no XML - interchange language
- BPMN is used for describing choreography modelling by expressing interconnected interface behaviour models - lack a control flow dependencies between interactions (i.e. ordering constraints)
- Weak support for Service Interaction Patterns
- Ambiguous semantics pertinent to Data Object management
- Related Works: YAWL, Let's Dance, UML AD.

## Briefly about SIP

- Capture the BPM languages suitability and their levels of expressiveness pertinent to B2B collaborations scenarios.
- Literature evaluates BP languages vs. their direct/partial or lack of support for SIP.
- We follow the divisions of SIP introduced by Baros et al. which comprise explanations of these patterns based on WS-BPEL code examples.
- We rely on these patterns as they capture the composition layer inside of a BPMN business process diagram (BPD).
- We discuss only those details pertinent to the communication-related aspects (i.e. reliability), as BPMN is a non-executable language. The patterns related to control-related aspects (i.e. involve a runtime determination) or imply multiple participants can not be directly supported by BPMN.

### References

[http://www.workflowpatterns.com/documentation/documents/serviceinteraction\\_BPM05.pdf](http://www.workflowpatterns.com/documentation/documents/serviceinteraction_BPM05.pdf)  
<http://www.workflowpatterns.com/documentation/documents/ServiceInteractionPatterns.pdf>

## Service Interaction Patterns (SIP) directly supported by BPMN

- Single-transmission Bilateral Interaction Patterns (total/supported - 3/3)
- Single-transmission Multilateral Interaction Patterns (total/supported - 4/1)
- Multi-transmission Interaction Patterns (total/supported - 4/1)

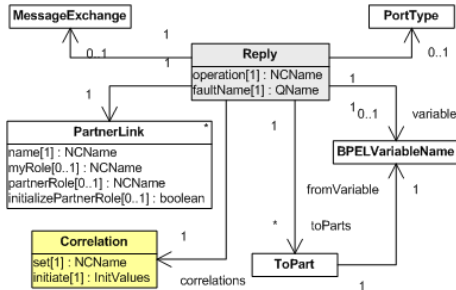
| SIP                      | BPMN 1.1                      | WS-BPEL   |
|--------------------------|-------------------------------|---|
| Send Pattern             | Send Task                     | Reply/Asynchronous Invoke Activity                  |
| Receive Pattern          | Receive Task                  | Receive Activity                                    |
| Send/Receive             | Service Task                  | Synchronous Invoke Activity                         |
| Racing Incoming Messages | Event-based Exclusive Gateway | Pick Activity                                       |
| Multi-responses          | Loop Type                     | Pick Activity inside a while/repeatUntil Activities |



## BPMN 1.1. - Task Types

- BPMN Task = atomic activity => Sent / Receive / Service
- This type of Task is identified by `SendTask` name and its mandatory attribute `MessageRef` of type `Message`, which denotes the message sent to an external partner of the business process. Once the message has been sent, the Task is completed.
- Modelling the behavioural character of BPMN task types using a `TaskType` attribute (BPMN Specification pp.6) is not appropriate as it restricts the extensibility of the model.
- We model `TaskType` attribute as classes specialisations.
- By default, the value of `implementation` property of `Task` class describes a Web Service.

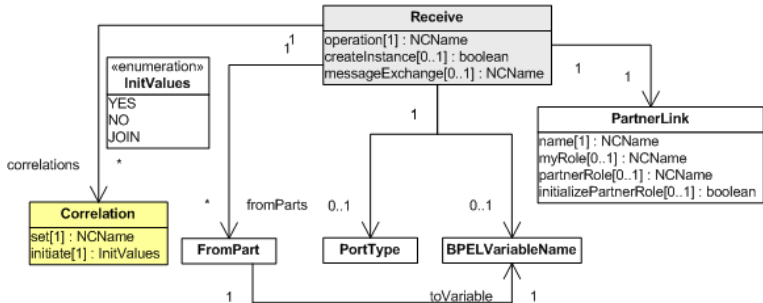
## WS-BPEL 2.0 - Reply Activity



## WS-BPEL 2.0 - Reply Activity

- WS-BPEL uses the `Reply` activity (which is the analogous concept of an asynchronous `Invoke` WS-BPEL activity) to encode this pattern.
- A `Reply` activity always supposes the existence of a previously declared `Receive` activity which receives the message request with the same values for the properties `portType` and `operation`. An activity is performed by a participant to the process i.e. a `partnerLink` which implicitly tight the activity with a performer and indirectly an end-point at runtime.
- The `implementation` property of the Task has the default value `WebService` and this value implies the mapping of its `Participant`, `Interface` and `Operation` properties to WS-BPEL corresponding elements: `partnerLink`, `portType` and `operation`, respectively.

## WS-BPEL 2.0 - Receive Activity



## WS-BPEL 2.0 - Receive Activity

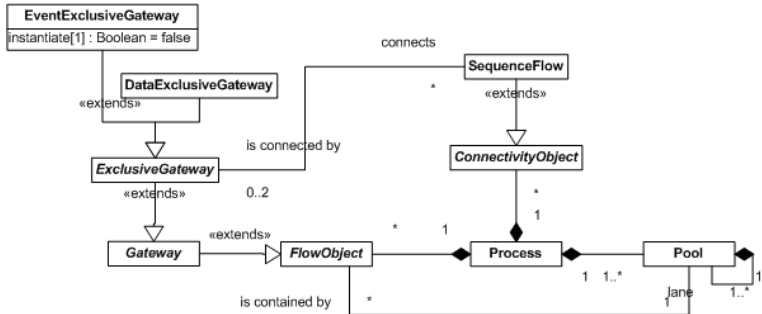
- BPMN captures this pattern using a Task with its `taskType` value `Receive`, which means that the task is waiting for receiving a message.
- Once the message has been received, the Task is completed.
- An incoming flow, bearing the specified `MessageRef` of type `Message` is waited, for a single instance of the process.
- The `ReceiveTask` is again a specialisation class in our model and it can employ an *instantiation mechanism* for the Process using the mandatory `instantiate` property which has the default value `false`. This attribute may be set to `true` value if the Task is the first activity initiated after the Start Event or a starting Task if there is no Start Event in a business process.



## WS-BPEL 2.0 - Invoke Activity

- This pattern is captured by BPMN using the `Service` value for the `typeTask` property.
- A BPMN Service type of task is used to embed the functionality of a Web Service. Its functionality can be easily mapped to a WS-BPEL synchronous invoke element. Both, an incoming message (i.e. `inMessageRef`) and an outgoing message (i.e. `outMessageRef`) need to be specified.

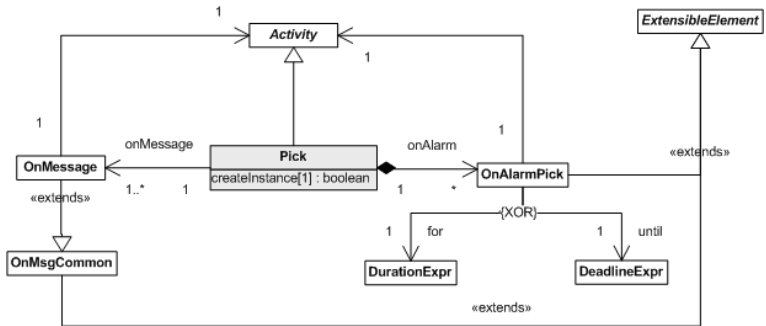
## BPMN 1.1 - Event-based Exclusive Gateway



## BPMN 1.1 - Event-based Exclusive Gateway

- This pattern implies the existence of multiple interactions (i.e. send/receive messages) with the same party. It is directly supported by BPMN using the Exclusive Event-based Gateway.
- Exclusive Event-based Gateway enables a participant to receive, at a point in the activity flow, different, concurrent types of messages from its business partners.
- The receipt of messages can be modeled with a `ReceiveTask` or an Intermediate Event with a Message Trigger. In addition to Messages, other Triggers for Intermediate Events can be used, such as Timers. The arriving of the first message choose a further path and cancel the others messages.
- The use of this type of gateway imposes two conditions: (1) it should act always as a *decision* and (2) it implies the mandatory presence of a second gateway (at least) that acts as a *merge*.
- The `instantiation` property of `EventExclusiveGateway` class is defined to employ the instantiation mechanism for the BPMN Process.

# WS-BPEL 2.0 - Pick Activity





## BPMN 1.1 - Loop Task

- BPMN supports this type of pattern using the loop type of task, so it can create a *repeat until* procedure that stops its mechanism when a trigger event happens. As usual, this structure implies a boolean condition.
- A `loop` activity is modelled also as a specialisation class of `Task` class i.e. `LoopActivity`.
- The `loopCondition` property captures the boolean condition (i.e. `Expression type`) that must apply to the loop task.
- The `loopCounter` property is used only at runtime to count the number of loops and is automatically updated by the process engine.
- The `loopCounter` property is incremented at the start of a loop activity and the boolean condition can depend on it.
- The conditions on loop activities can be evaluated at the start or at the end of the activity i.e. `testTime` property of `LoopTime` enumeration.

## WS-BPEL 2.0 - Pick Activity

- This type of pattern is captured in WS-BPEL using again the `pick` activity and its `onMessage` handler for each type of message.
- More than one message can be accepted, so a `while` or a `repeatUntil` activity must enclose the `pick` activity, and must specify appropriate boolean condition.

## Description

- BPMN 1.1 Standard is well accepted by industry, as it hides its complexity behind a simple and intuitive visual language, enhanced with an extensibility methodology.
- It is an evolving language - two BPMN 2.0 Specification proposals are competing for standardisation. They both add a metamodel for BPMN, capable to capture the language abstract syntax, concrete syntax and its semantics.
- Extending BPMN - the reuse of a popular notation, as many process modelers already use this notation.

## Description

- iBPMN - provides BPMN extensions in order to support the modelling of choreography behaviour.
- BPMN 1.1 provides support for choreography modelling by means of interconnected interface behaviour models, which are susceptible of redundancies (i.e. deadlock situations) and of incompatible behaviour.
- As an alternative, the authors introduce interaction modeling using BPMN i.e. a set of extensions from the main standard: `Participant Sets`, `References` and `Reference Passing`. The suitability of iBPMN for choreography modelling was also tested.
- The expression language and the query language is XPath.

## Description

- The pools are empty.
- One-to-many Send: An auctioning service sends out notifications to all unsuccessful bidders. (the messages all have the same type - but the content may differ).
- One-from-many Receive: The auctioning service does not know in advance how many bidders are going to take part in the auction. The arrival of the requests need to be timely.
- Contingent Request: involves a list with recipients for requests. If the first recipient does not respond within a given time, the request is sent to the next one. (partial support)
- Request with referral: link passing mobility.
- Relayed request: a third participant observes the conversation between two other participants.

## Description

- Simple BPMN - simplification of visual representation.
- Literature also investigates the necessity of such a complex and multiple control flow situations, where some of BPMN concepts are overlapping in their provided functionality.
- The need for a simplified language was sustained by F. Havey that tried to provide a simplified visual notation for BPMN.
- Another aspect of the BPMN workflow concepts simplification can be provided through the use of just a core part of the entire BPMN set of concepts.

## Description

- rBPMN - provides means for rule-based modelling of business processes
- The purpose of this work is to obtain an executable business process by encode it in a specific rule-based language (i.e. R2ML) and further to be capable to deploy it in a particular business rule-engine (i.e. Drools).