# 1. Introduction

As organizations face increasing levels of pressure to deliver more efficient and effective operations, business process simulation and analysis is being recognized as an integral part of optimizing performance. Inadequate or poorly designed business processes lead to undesired organizational behavior that may in turn lead to loss of revenues, goodwill or worst. This is why it is important to thoroughly analyse business processes in a safe isolated environment before they are deployed.

Although recognized as desired and relevant within the practice of Business Process management (BPM), simulation and analysis of business processes is still not systematically used in most business process improvement projects. The reasons for this may be many (availability, tooling, training, etc.) but one certain factor is the lack of existence of standards. While mature standards exist for the definition of business process models (e.g. BPMN and XPDL) there is no generally accepted standard for business process simulation and analysis.

In analysing business processes many different possibilities to improve the process are at hand. Structural analysis will concentrate on the structural aspects (e.g. configuration) of a business process model. These will usually consist of statistical analysis often using static methods. Capacity Analysis will on the other hand concentrate on the capacity aspects of a business process model (e.g. limitations) usually based on dynamic analysis often using discreet simulation methods.

To carry out all these analyses, business process models often need to be augmented with process analysis data. Both estimated values and historical execution values are often used as parameterization of the business process model in support of pre-execution or post-execution optimization. Pre-execution optimization will concentrate on “what if” analysis of estimated values as input parameters. While post-execution optimization will concentrate on “what if” analysis of historical values as input parameters.

# 2. Scope

This document introduces the Process Analysis Framework (PAF), a standardized specification that allows business process models captured in either BPMN or XPDL to be augmented with information in support of rigorous methods of analysis.

This specification defines the parameterization and interchange of process analysis data allowing structural and capacity analysis of process models. This specification is meant to support both pre-execution and post-execution optimization of said process models.

This specification consists of an underlying computer-interpretable representation (meta-model) and an accompanying electronic file format to ease the safeguard and transfer of this data between different tools (interchange format).

The meta-model is captured using the Unified Modeling Language (UML) and the interchange format is defined using an XML Schema Definition (XSD). Note that the meta-model and the interchange format represent the core of the normative material of this specification.

In defining the meta-model and the interchange format, priority was given to ensure a resulting XML file that is more human consumable. Conversely, this decision of favoring a more human consumable interchange format has for known side effect that the resulting meta-model does not adhere to all best practices of the object orientation.

In order to support both pre-execution and post-execution optimization, the meta-model and interchange format allow for the capture of both inputs and outputs of the process analysis. Both estimated values and historical execution values are supported as parameterization of the business process model.

One of the goals of this specification is to be complementary to already existing standards related to business process modeling. This first version of this specification is scoped based on the Business Process Model and Notation (BPMN) version 2.0 from the Object Management Group (OMG) and the XML Process Definition Language (XPDL) version 2.2 from the Workflow Management Coalition (WfMC).

In realizing this first version, attention was given as to not duplicate any process model information already provided by these two process modeling standards whenever possible. Great care was also taken to ensure that proper extension mechanism of both BPMN and XPDL were respected as to ensure that interchange could take place within a proper BPMN file, a proper XPDL file or as a standalone XML file.

## 3. Conformance

The meta-model and the interchange format represent the core of the normative material of this specification.

This rest of this specification is organized into sections. All sections of this document are normative except where specifically indicated as none normative.

An individual or organization (vendor or otherwise) cannot claim conformance to this specification unless addressing all normative sections of this specification.

# 4. Elements

## 4.1 Scenario

Process analysis data is used to provide complimentary information to a BPMN or XPDL business process model in the context of process analysis, simulation and optimization. Business process models and their business process elements are external sources. Scenarios within the process analysis data are always in reference to a single business process model (note that many processes can be captured into a single BPMN or XPDL business process model). Thus the business process model is a

separately defined fixed point with variations possible on scenarios.

Scenarios can be used to capture: input parameter specification for analysis, simulation and optimization; or results from analysis, simulation and optimization; or historical data from past real world execution of the business process model.

Scenarios of results will often reference a scenario of input specification (i.e. the results for the referenced input set).

It is possible for a scenario to overload or augment (inherit from) another scenario. In such case, only the changes to the element parameter values, or the added element parameter and their values, need to be specified in the inheriting scenario.

A scenario is composed of a collection of element parameters. Each element parameter of a scenario references a specific element of a process within the business process model.

Each scenario may possess scenario parameters.

An extension capability is provided to the process analysis data. Both scenarios and element parameters can be extended with proprietary vendor extensions.

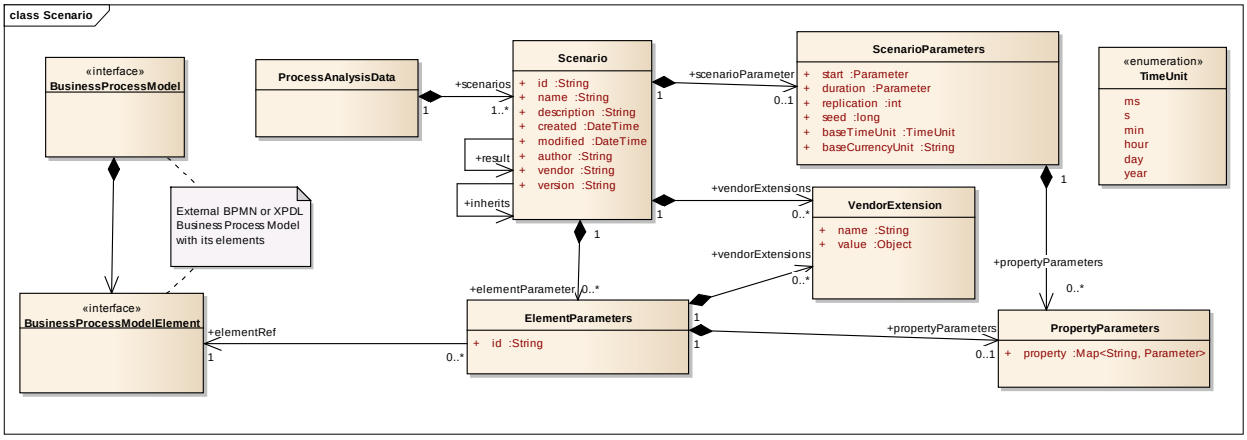

Figure 1

## *ProcessAnalysisData*

The ProcessAnalysis Data class is the root class where all scenarios are defined.

| Attribute | Description |
|---|---|
| Scenario : scenarios | A collection of scenarios |

## *Scenario*

The Scenario class regroups all ElementParameter for a given scenario.

| Attribute | Description |
|---|---|
| String : id | Unique scenario identifier |
| String : name | A name for this scenario |
| String : description | A description of this scenario |
| DateTime : created | When this scenario was created |
| DateTime : modified | When this scenario was last modified |
| String : author | The scenario author name |
| String : vendor | The name of the software tool that was used to create this scenario |
| String : version | The version of this scenario |
| Scenario : result | In the case that this scenario is the output of an analysis and that the source of the analysis is also provided in the model, this field references the source scenario. |
| Scenario : inherits | Reference to the scenario that this scenario inherits from. When inheriting from scenario, only overload values and added ElementParameter with values are provided. |
| ElementParameter : elementParameter | Collection that compose this scenario |
| ScenarioParameter : scenarioParameter | Parameters about this scenario |
| VendorExtension : vendorExtension | Proprietary vendor extensions for this Scenario |

### *ScenarioParameters*

The ScenarioParameter class defines the parameters about the scenario.

| Attribute | Description |
|---|---|
| Parameter : start | Start time of the scenario |
| Parameter : duration | Duration of the scenario |
| int : replication | Number of replication of that scenario that needs to be executed. Defaults to 1. |
| long : seed | A random seed to be used to initialize a pseudo random number generator.<br><br>Given the exact same model (Business Process Model and Process Analysis Framework Model) and a given seed, the results should be the same across executions.<br><br>Using replication, giving a seed does not mean that each replication will return the same result but that for a given seed and a given number of replication the exact same results are generated. |
| TimeUnit : baseTimeUnit | Base time unit of this scenario. All integer values representing time should be considered as being expressed in that unit.<br><br>Variable costs are represented as the number of baseCurrencyUnit/baseTimeUnit. The default value |

| | |
|---|---|
| | if unspecified is seconds. |
| String : baseCurrencyUnit | Base currency international currency code for this schema expressed using the ISO 4217 (three letter codes). |

### *TimeUnit*

The TimeUnit enumeration represents all the possible time unit.

| Attribute | Description |
|---|---|
| : ms | milliseconds |
| : s | seconds |
| : min | minutes |
| : hour | hours |
| : day | days |
| : year | year |

### *VendorExtension*

The VendorExtension class is a proprietary vendor extension holder. Vendors can add non normative extensions.

| Attribute | Description |
|---|---|
| String : name | The name of the vendor extension. Use an appropriate prefix to your extension names to prevent collision |
| Object : value | The value of the Vendor Extension |

## 4.2 Parameters

Each element parameter of a scenario references a specific element of a process within the business process model.

To address separation of concerns, element parameters are divided into different perspectives. Each perspective regroups a collection of parameters from a common concern.

The values of parameters may be attached to a specific calendar.

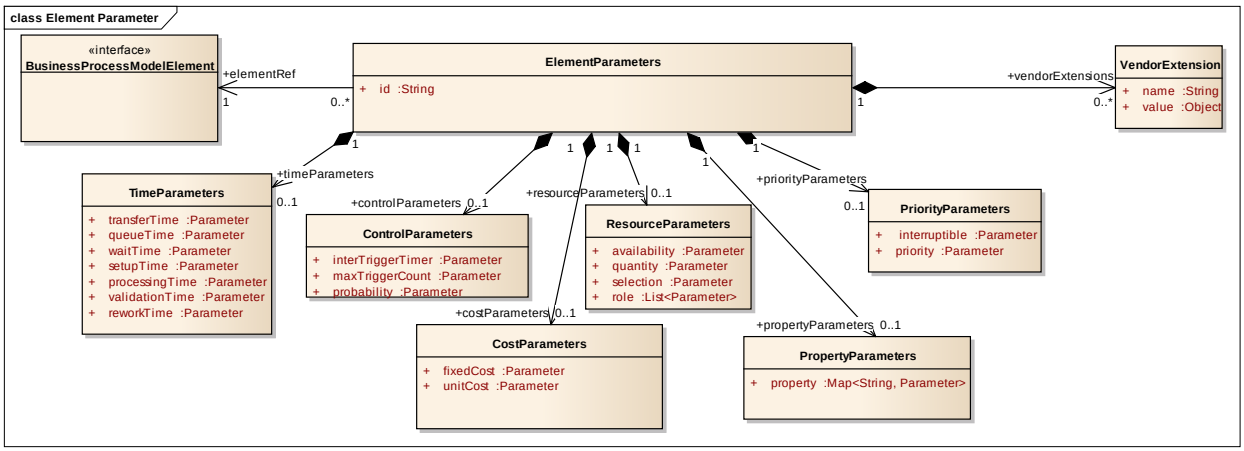


Figure 2

## *ElementParameters*

The ElementParameter class is the concrete class definition of all parameter perspectives.

| Attribute | Description |
|---|---|
| String : id | A unique identifier for this parameters |
| BusinessProcessModelElement : elementRef | A reference to the business process element for which we are defining a parameter |
| VendorExtension : vendorExtensions | Proprietary vendor extensions for this ElementParameter |

## *TimeParameters*

The TimeParameter class regroups all parameters that specify time related parameters for a business process element.

All TimeParemeters capture time intervals and are defined from an external observer point of view.

The two main time interval of interest are duration and lagTime. The duration time interval captures the elapsed time from the start of a work effort to its completion, while the lagTime interval captures the elapsed time from the completion of a predecessor work effort until the start of the successor work effort.

A duration value can be obtained from the sum of setupTime, processingTime, validationTime and reworkTime.

A lagTime value can be obtained from the sum of transferTime, queueTime and

waitTime.

TimeParemeters values must resolve to either a NumericParameter, FloatingParameter or DurationParameter.

The default value of all unspecified TimeParameters is considered to be 0 seconds.

Parameters from the temporal perspective only apply to types of business process elements that take place over an interval of time.

| Attribute | Description |
|---|---|
| Parameter : transferTime | The time spent traveling from the previous processing step |
| Parameter : queueTime | The delay between the Successor being offered and the successor being allocated |
| Parameter : waitTime | The time between the Successor being allocated and it being actually started. |
| Parameter : setupTime | The time expended prior to performing the actual work . |
| Parameter : processingTime | The time actually spent doing the work at hand. |
| Parameter : validationTime | The time spent reviewing or inspecting the work done. |
| Parameter : reworkTime | The time spent correcting or redoing the work done |

### *ControlParameters*

The ControlParameter class regroups all parameters that specify the control flow of a business process element

Parameters from the control perspective only apply to certain types of business process elements.

| Attribute | Description |
|---|---|
| Parameter : interTriggerTimer | The time interval between occurrences. After the specified time interval, the event occurs and will keep occuring every time interval until the maxTriggerCount is reached.<br><br>Modeler should be careful when putting an inter trigger time to a boundary event that can be triggered otherwise (ex signal or timer).<br><br>This parameter must resolve to either a NumericParameter, FloatingParameter or DurationParameter.<br><br>The default value of this parameter is that the event never occurs. Setting a duration of 0 seconds would mean that the event occurs instantly. |

| | |
|---|---|
| Parameter : maxTriggerCount | The maximum number of times to trigger of this event.<br><br>This parameter must resolve to a NumericParameter.<br><br>The default value of this parameter is considered to be the infinity so not defining it means that the event will not stop occurring after a maximum number of times. |
| Parameter : probability | The probability of the control being passed to this element.<br><br>This parameter must resolve to either a NumericParameter or a FloatingParameter.<br><br>The default value of this parameter varies depending on the element being referenced.<br><br>For sequence flow, the default probability is distributed evenly between outgoing sequence flow that does not have a probability defined.<br><br>ex: 4 outgoing sequence flow, none with defined probability, they each have a probability of 0.25<br><br>ex: 3 outgoing sequence flow, one with a probability of 0.4 defined, the other two that don't have a probability defined will receive the probability of 0.3.<br><br>For events, the default probability is 0. |

### *ResourceParameters*

The ResourceParameter class regroups all parameters that specify the resources of a business process element

Parameters from the resources perspective only apply to certain types of business process elements.

| Attribute | Description |
|---|---|
| Parameter : availability | Determine wether a resource is available or not. This will often be varied according to a calendar to represent when a resource is available.<br><br>This parameter must resolve to a BooleanParameter.<br><br>The default value of this parameter is true (the resource is available) |
| Parameter : quantity | The quantity of resources. |

| | |
|---|---|
| | This parameter must resolve to a NumericParameter.<br><br>The default value of this parameter is 1. |
| Parameter : selection | Criteria for selecting the desired resource.<br><br>This is an override of the BPMN ResourceRole element behavior to assign roles to resources.<br><br>This parameter can reference roles defined using the Role ResourceParameters to easily reference the resources associated with the activity.<br><br>This parameter must resolve to either a StringParameter or NumericParameter.<br><br>The default value of this parameter is to conserve the BPMN ResourceRole behavior and not override it. |
| List<Parameter> : role | The roles (may be more than one) of a resource.<br><br>Theses roles can be reused in the Selection ResourceParameter.<br><br>Each role must resolve to a StringParameter<br><br>The default value of this parameter is to not define roles for the resource. |

### *CostParameters*

The CostParameter class regroups all parameters that specify the cost of a business process element

Parameters from the cost perspective only apply to certain types of business process elements.

The variable cost of the activity is the sum of the variable human, utility and material cost.

The cost of an activity is the sum of the variable and fixed cost.

| Attribute | Description |
|---|---|
| Parameter : fixedCost | The fixed cost that has to be paid each time.<br><br>The cost is expressed as the number of baseCurrencyUnit defined in the Scenario Parameters.<br><br>This parameter must resolve to either a NumericParameter or a FloatingParameter. |

| | |
|---|---|
| | The default value of this parameter is 0. |
| Parameter : unitCost | The cost per unit of time that has to be paid.<br><br>The cost is expressed as the number of baseCurrencyUnit per baseTimeUnit defined in the Scenario Parameters.<br><br>This parameter must resolve to either a NumericParameter or a FloatingParameter.<br><br>The default value of this parameter is 0. |

### *PropertyParameters*

The PropertyParameter class regroups all parameters that specify the property parameters of a business process element

Parameters from the property perspective only apply to certain types of business process elements.

| Attribute | Description |
|---|---|
| Map<String, Parameter> : property | Specify additional properties that are assigned to BPMN Elements.<br><br>Those properties can be accessed in ExpressionParameters.<br><br>Property parameters are evaluated as soon as a token enter the element and before everything else.<br><br>This parameter can resolve to any type of parameter and does not have a default value. |

### *PriorityParameters*

The PriorityParameters class regroups all parameters that specify the property parameters of a business process element

Parameters from the instance perspective only apply to certain types of business process elements.

| Attribute | Description |
|---|---|
| Parameter : interruptible | Determine wether the execution of this element is interruptible.<br><br>This parameter must resolve to a BooleanParameter. |

| | |
|---|---|
| | The default value of this parameter is false (the element is uninterruptible). |
| Parameter : priority | Determine the priority of a business process element.<br><br>This parameter must resolve to either a FloatingParameter or a NumericParameter.<br><br>The default value of this parameter is 0. |

## 4.3 Parameter Types

A parameter must have a default value. The default value of a parameter can be modified for various intervals of time by enumerating additional values for the parameter. Each additional value must specify a calendar of applicability.

Each value provided for a parameter must be of a specific type.

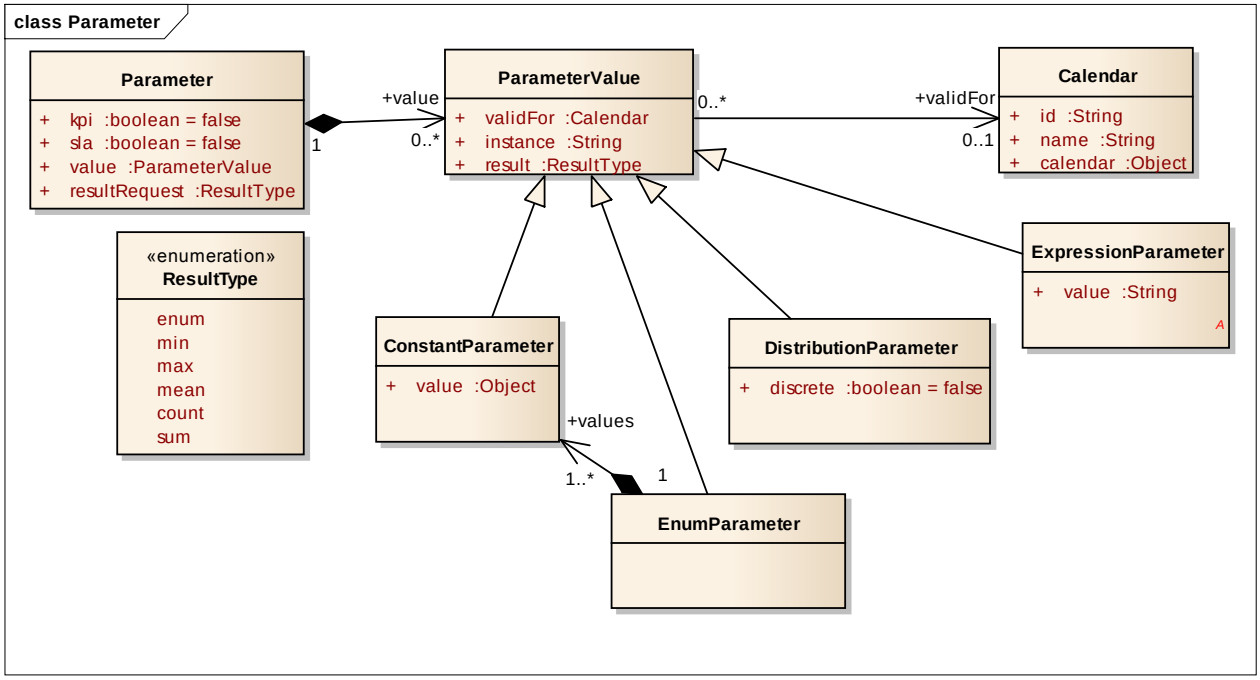


Figure 3

### *Parameter*

The Parameter class regroups the ParameterValues for a parameter.

| Attribute | Description |
|---|---|
| boolean : kpi | Determine if this Parameter is a Key Performance Indicator |
| boolean : sla | Determine if this Parameter is a Service Level |

| | Agreement |
|---|---|
| ParameterValue : value | The value of the parameter |
| ResultType : resultRequest | Used for input only, this indicates the result types that we expect to see in the result scenario for this parameter. |

### *ParameterValue*

The ParameterValue class is the abstract class definition of all ParameterValue types.

| Attribute | Description |
|---|---|
| Calendar : validFor | References a calendar of applicability for this value. If unspecified, the ParameterValue is the default value of this parameter. |
| String : instance | The unique identifier of the process instance that this value was obtained from. Used for representing an output value only. |
| ResultType : result | Used for result values only, this indicates the type of result being requested. |

### *ResultType*

The ResultType enumeration represents all the possible output that can be generated from the analysis.

| Attribute | Description |
|---|---|
| : enum | Output all the data points as a result |
| : min | Output the minimum value as a result |
| : max | Output the max value as a result |
| : mean | Output the mean value as a result |
| : count | Output the number of occurence as a result |
| : sum | Output the sum of all results |

## 4.3.1 Constant Parameters

Constant parameters are parameters that will always resolve to the same value over time.

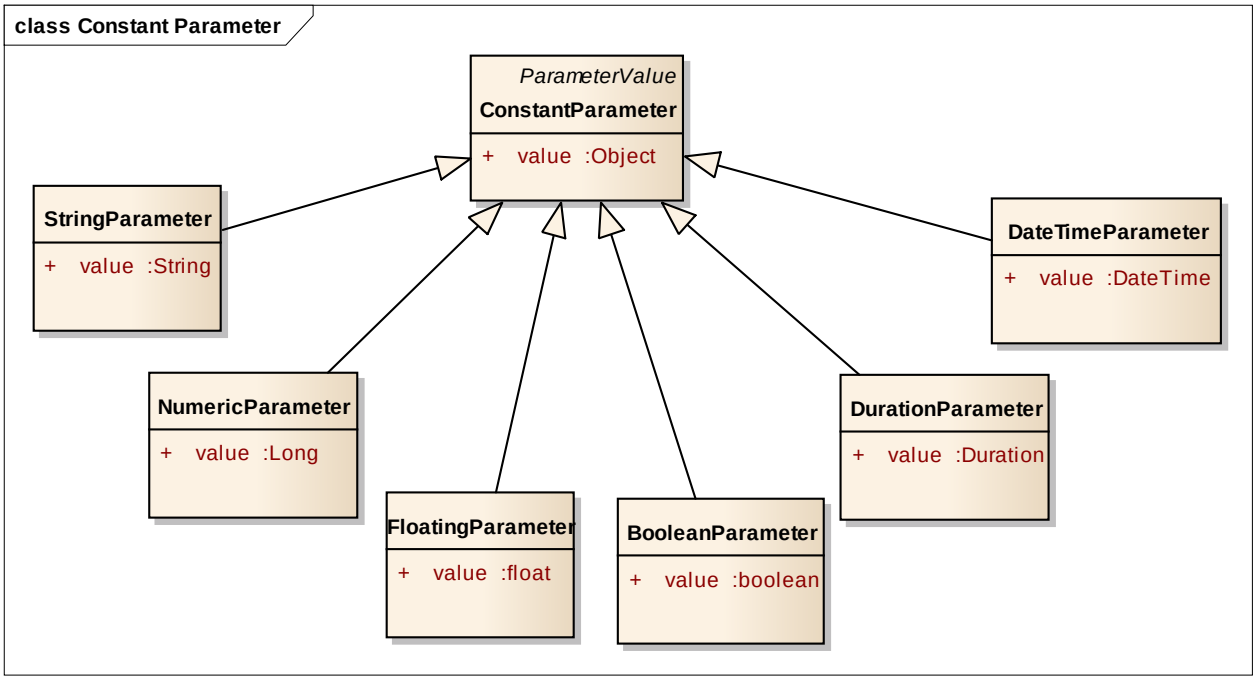


Figure 4

## BooleanParameter

| Attribute | Description |
|---|---|
| boolean : value | true or false |

## ConstantParameter

| Attribute | Description |
|---|---|
| Object : value | |

## DateTimeParameter

An instant in time defined using the ISO 8601 format

| Attribute | Description |
|---|---|
| DateTime : value | Using the YYYY-MM-DDThh:mm:ssZ format. All DateTime are assumed to be in Zulu time. |

### DurationParameter

A duration defined using the ISO 8601 format

| Attribute | Description |
|---|---|
| Duration : value | Duration is specified using either the long format (PnnYnnMnnDTnnHnnMnnS) or the short format (e.g. PnnW) |

### FloatingParameter

| Attribute | Description |
|---|---|
| float : value | a floating point value |

### NumericParameter

| Attribute | Description |
|---|---|
| Long : value | Integer value |

### StringParameter

| Attribute | Description |
|---|---|
| String : value | String value |

## 4.3.2 Distribution Parameters

Distribution parameters are parameters that have different values over time but will statistically be distributed according to a given distribution.

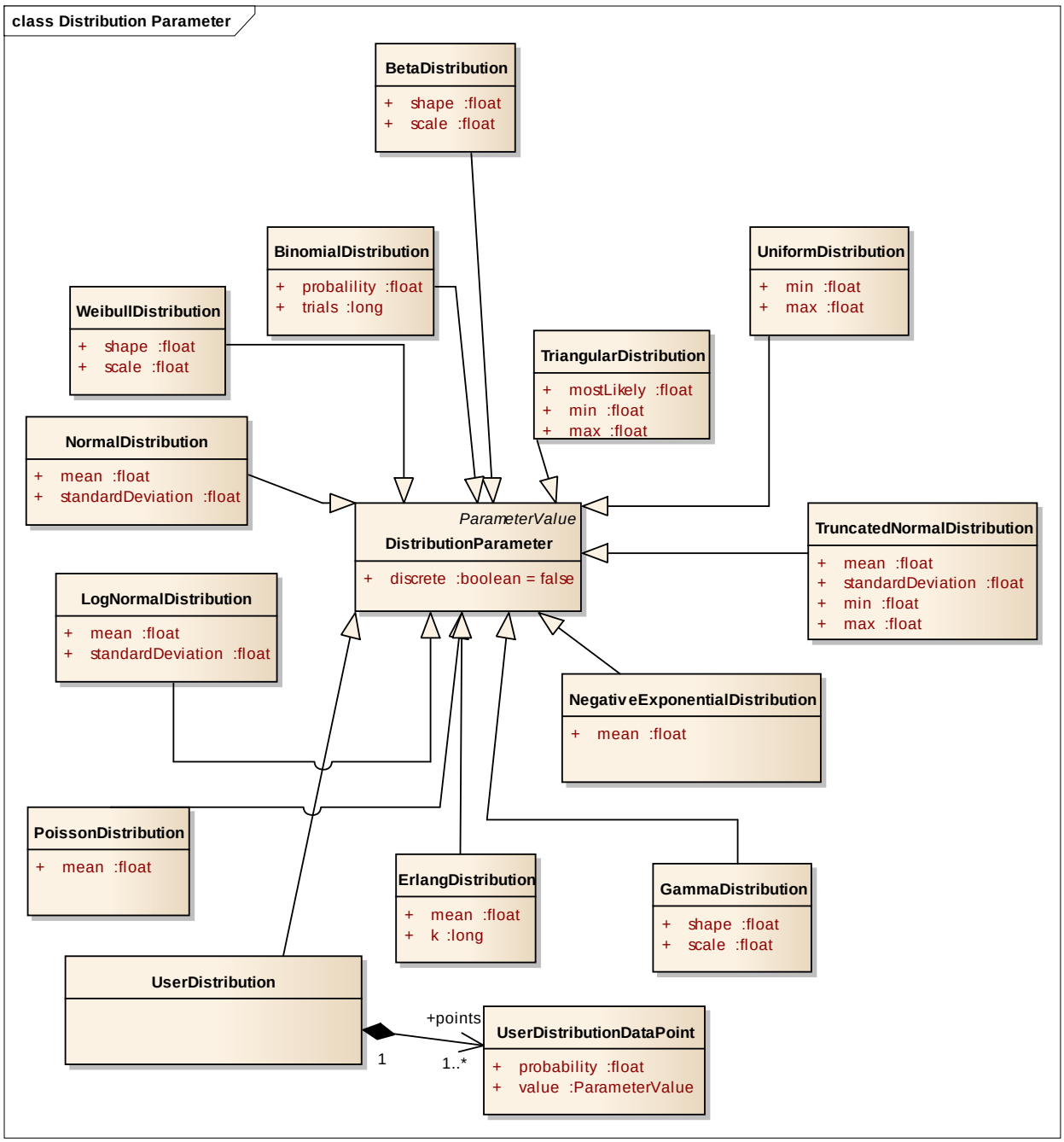


Figure 5

## DistributionParameter

| Attribute | Description |
|---|---|
| boolean : discrete | If set to true than the distribution is discrete, if set to false than the distribution is continuous. The |

| | |
|---|---|
| | default value is set to false. |

### BetaDistribution

The BetaDistribution class provides a sample from the beta distribution, which is a real distribution. The beta distribution can assume a wide variety of shapes and is often used as a rough model where real-life data is limited

| Attribute | Description |
|---|---|
| float : shape | The shape value |
| float : scale | The scale value |

### BinomialDistribution

The BinomailDistribution class provides a sample from the binomial distribution, which is an integer distribution. It returns the expected number of successes, given a number of trials and a probability of success. For example, if light bulbs from a supplier are known to be 10% faulty, you could use the binomial distribution to estimate the number of faulty bulbs in a batch of five.

| Attribute | Description |
|---|---|
| float : probalility | The probability of success |
| long : trials | The number of trials |

### ErlangDistribution

The ErlangDistributioion class provides a sample from an ERLANG K distribution, which is a real distribution. The Erlang is a family of distributions: it has a different curve depending on the value of the K parameter.

When K = 1, the Erlang distribution is identical to the **negative exponential** distribution (this is because it is based on the sum of K samples from a negative exponential distribution with the same mean).

When K = 2, the Erlang distribution is a bell-shaped distribution, strongly skewed to the left (similar in shape to the **Log Normal** distribution).

When K is larger than 2, the Erlang distribution starts to resemble the **normal** distribution.

However, unlike the **Normal** or **Log Normal** distributions, the **Erlang** distribution is characterized by its mean alone.
You can use the Erlang distribution for sensitivity analysis by changing the K parameter (for example, for testing the effect of stoppages). Low K values cause maximum chaos,

while high K values reduce chaos.

| Attribute | Description |
|---|---|
| float : mean | The mean value |
| long : k | The K Value |

### GammaDistribution

The GammaDistribution class provides a sample from the gamma distribution, which is a real distribution. It returns a sample from the distribution with a specified shape and scale.

| Attribute | Description |
|---|---|
| float : shape | The shape value |
| float : scale | The scale value |

### LogNormalDistribution

The LogNormalDistribution class provides a sample from a Log Normal distribution, which is a real distribution. It is a bell-shaped distribution, strongly skewed to the right.

Data is said to come from a log normal distribution if the logarithms of the sample values follow a normal distribution.

| Attribute | Description |
|---|---|
| float : mean | The mean value |
| float : standardDeviation | The standard deviation value |

### NegativeExponentialDistribution

The NegativeExponentialDistribution class provides a sample from the Negative Exponential distribution, which is a real distribution. It may be thought of as the complement of the Poisson distribution.

| Attribute | Description |
|---|---|
| float : mean | The mean value |

### NormalDistribution

The NormalDistribution class provides a sample from the Normal distribution, which is a real distribution. This is one of the most common distributions in nature, and has a symmetrical bell-shaped curve. It is useful for modeling situations where values are evenly distributed around a mean.

| Attribute | Description |
|---|---|
| float : mean | The mean value |
| float : standardDeviation | The standard deviation |

### PoissonDistribution

The PoissonDistribution class provides a sample from the Poisson distribution, which is an integer distribution. Typically, it is used to estimate the number of arrivals within a given period (for example, size of batches for tokens). It may be thought of as the complement of the negative exponential distribution.

| Attribute | Description |
|---|---|
| float : mean | The mean value |

### TriangularDistribution

This provides a sample from the Triangular distribution. As its name suggests, this distribution has a triangular 'curve'.

| Attribute | Description |
|---|---|
| float : mostLikely | The most likely value |
| float : min | The lower bound of the generated numbers |
| float : max | The upper bound of the generated numbers |

### TruncatedNormalDistribution

The TruncatedNormalDistribution class provides a sample from the Truncated Normal distribution, which is a real distribution. This is similar to the normal distribution with the difference being that minimum and maximum values for sampling are specified.

| Attribute | Description |
|---|---|
| float : mean | The mean value |
| float : standardDeviation | The standard deviation value |
| float : min | The lower bound of the generated numbers |
| float : max | The upper bound of the generated numbers |

### UniformDistribution

The UniformDistribution class provides a sample from the Uniform distribution, which is a real distribution. It may be used when there is equal probability of obtaining any real value in the specified range.

| Attribute | Description |
|---|---|
| float : min | The lower bound of the generated numbers |
| float : max | The upper bound of the generated numbers |

#### UserDistribution

The UserDistribution class provides a custom sampling of points with the likeliness of each one to occurs.

| Attribute | Description |
|---|---|
| UserDistributiobnDataPoint : points | A list of data point. |

#### UserDistributionDataPoint

The UserDistributionDataPoint class represents a data point in the User Distribution

| Attribute | Description |
|---|---|
| float : probability | The probability of this data point occurring expressed as a fraction from 0 to 1.The sum of all data point probabilities should add to 1.0 |
| ParameterValue : value | The value of the Data Point |

#### WeibullDistribution

The WeibullDistribution class provides a sample from the weibull distribution. It returns a sample from the distribution with a specified shape and scale.

| Attribute | Description |
|---|---|
| float : shape | The shape value |
| float : scale | The scale value |

### 4.3.3 Enumeration parameters

Enumeration parameters are collections of constant parameters. Enumeration parameters provide a collection of data points resulting from analysis, simulation and optimization or from real world execution of the business process model (historical data).

Every time the parameter is evaluated, the next value in the collection is returned.

#### EnumParameter

| Attribute | Description |
|---|---|
| ConstantParameter : values | A collection of values for this enumeration |

### 4.3.4 Expression parameters

Expression parameters are parameters that have different values over time according to the output of an expression.

#### ExpressionParameter

| Attribute | Description |
|---|---|
| String : value | The XPATH expression |

The expression has to be expressed using the XPATH 1.0 language defined at http://www.w3.org/TR/1999/REC-xpath-19991116/. For the purpose of the process analysis framework, XPATH is extended to provide these additional functions under the "paf" namespace:

| XPath Extension Function | Description / Usage |
|---|---|
| paf:getProperty(*name*) | Returns the value of a property parameter.<br>**Arguments**<br>*name*: the name of the property parameter.<br>**Return**<br>Returns the value |
| paf:getResource(*name*, *qty*) | Selects a collection of available resource(s) required for an Activity.<br>**Arguments**<br>*name*: the name of the resource required by the Activity. In the case of BPMN this is the attribute used to uniquely identify BPMN resource element.<br>*qty*: the quantity of the resource required by the Activity, expressed as an integer.<br>**Return**<br>Collection of resource(s) or an empty collection if the resource requirements were not satisfied.<br>**Remarks** |

| | |
|---|---|
| | Resources are defined in the BPMN interchange using the <resource> element. |
| paf:getResourceByRoles(*[role*, …], *qty*) | Selects a collection of available resource(s) that can satisfy the role(s) required for an Activity. Selected resource(s) will play all roles specified by the list of roles required.<br>**Arguments**<br>*[role …]:* the variable list of required role(s).<br>*qty:* the quantity of a resource that satisfies the specified role(s), required by the Activity, expressed as an integer.<br>**Return**<br>Collection of resource(s) or an empty collection if the resource requirements were not satisfied<br>**Remarks**<br>A role can be applied to a resource using the PAF *role* parameter from the *ResourceParameters* perspective. |
| paf:orResource(*[resources*, …]) | Select the first collection of available resource(s) from the list of alternative resource(s) used for an Activity.<br>**Arguments**<br>A variable list of resources returned by the getResource() or getResourceByRoles() functions.<br>**Return**<br>Collection of resource(s).<br>**Remarks**<br>This allows alternative behaviour for resource selection. The evaluation order of resources is from left to right. |

## 4.4 Calendar

Calendars are defined at the scenario level and Parameter Values references them.

A calendar is serialized using the iCalendar (RFC 5545) format and it provides the time interval for which a parameter value should be used.

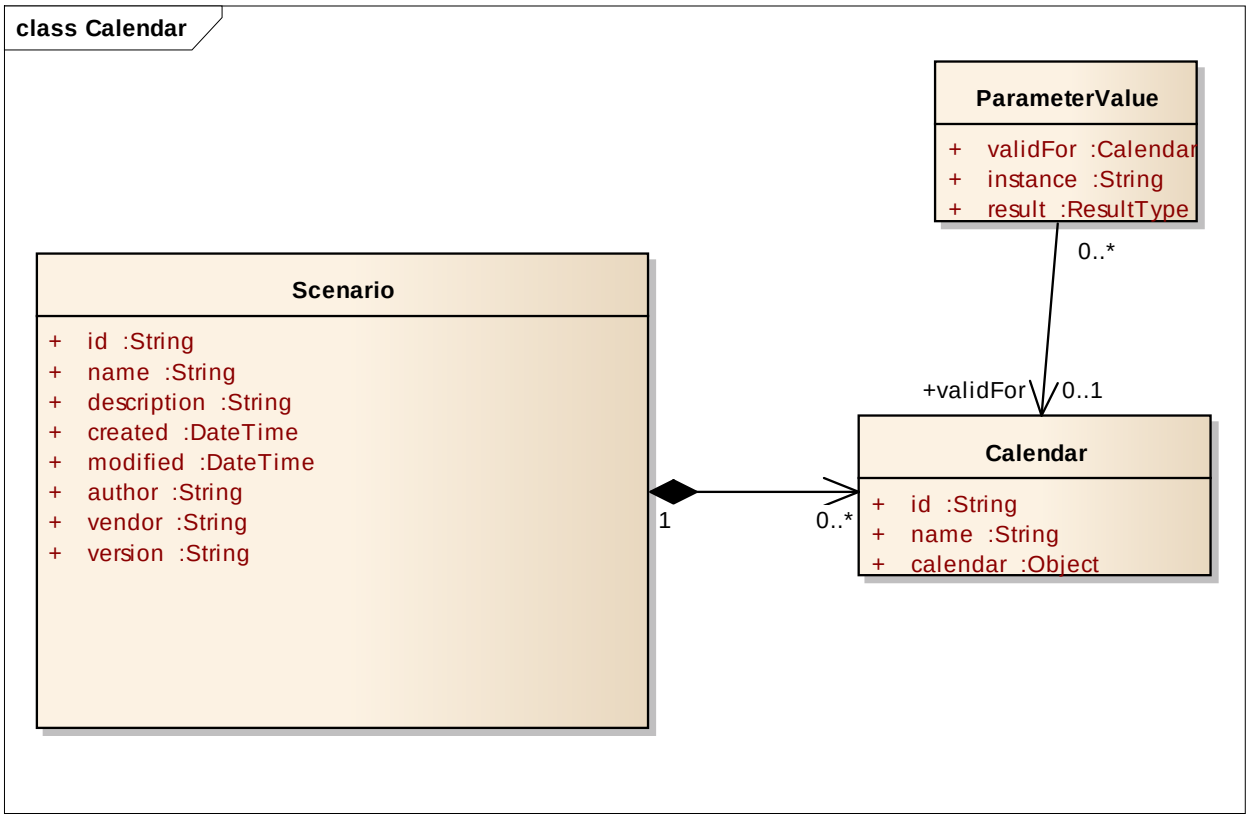


Figure 6

### *Calendar*

The calendar class serializes the iCalendar format.

| Attribute | Description |
|---|---|
| String : id | Calendar unique identifier |
| String : name | Descriptive name for this calendar |
| Object : calendar | iCalendar serialization of events (VEVENT) describing this calendar |

# 5. Parameters usage for BPMN 2.0

## 5.1 Time Parameters

| | | Time Parameters | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | transferTime | queueTime | waitTime | setupTime | processingTime | validationTime | reworkTime |
| Events | Start Event | No - BPMN Events map to time point and thus cannot have Time Parameters which are time intervals | | | | | | |
| | Intermediate Event | | | | | | | |
| | End Event | | | | | | | |
| Activities | Task | **Yes** | | | | | | |
| | Sub Process | **Yes - but only for activities without decomposition** | | | | | | |
| | Transasction | | | | | | | |
| | Call Activity | | | | | | | |
| | Event Sub Process | | | | | | | |
| Gateways | Gateway | No - BPMN Gateways do not map to time intervals as they are only visualizations of branching logic | | | | | | |
| Connecting Objects | Sequence Flow | No - BPMN Connecting Objects do not have time interval associated to them | | | | | | |
| | Message Flow | | | | | | | |
| | Data Association | | | | | | | |
| | Association | | | | | | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | | | | | | |
| | Data Store | | | | | | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | | | | | | |
| | Pool | | | | | | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | | | | | | |
| Attributes | ResourceRole | No | | | | | | |
| | Resource | | | | | | | |

Time Parameters are applicable only to Activities that does have a decomposition in the model. If an activity have a decomposition in the BPMN model, the Time Parameters of the decomposition should be used and the abstraction can't have Time Parameters defined.

## 5.2 Control Parameters

| | | Control Parameters | | |
|---|---|---|---|---|
| | | interTriggerTimer | maxTriggerCount | probability |
| Events | Start Event | **Yes** | **Yes** | **Yes - but inside Event Sub Process only** |
| | Intermediate Event | **Yes - but for Catch Event only** | No | **Yes - but for Boundary Event only** |
| | End Event | No | | |
| Activities | Task | **Yes - but only for activities without decomposition without incoming sequence flow** | | No |
| | Sub Process | | | |
| | Transasction | | | |
| | Call Activity | | | |
| | Event Sub Process | **Yes - but only for Event Sub Process without decomposition** | | |
| Gateways | Gateway | **Yes - but only for Event Based Gateway starting a process** | | No |
| Connecting Objects | Sequence Flow | No | | |
| | Message Flow | | | |
| | Data Association | | | |
| | Association | | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | | |
| | Data Store | | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | | |
| | Pool | | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | | |
| Attributes | ResourceRole | No | | |
| | Resource | | | |

InterTriggerTime is applicable to all type of Start Events, Intermediate Catching Events and Event Sub Process without decomposition. It can also be applied to Activities that are initiating the process because they don't have an incoming sequence flow (Task, Sub Process, Transaction and Call Activity).

MaxTriggerCount can be applied to the al type of Start Events, Event Sub Process without decomposition and Activities that are initiating the process because they don't have an incoming sequence flow (Task, Sub Process, Transaction and Call Activity).

Probability can be applied to Boundary Intermediate Events. It can also be applied on Event Sub Process without decomposition or on the Start Event inside the decomposition of the Event Sub Process. It can also be applied on an Event Based Gateway that starts a process.

## 5.3 Resource Parameters

| | | Resource Parameters | | | |
|---|---|---|---|---|---|
| | | availability | quantity | role | selection |
| Events | Start Event | No | | | |
| | Intermediate Event | | | | |
| | End Event | | | | |
| Activities | Task | No | | | |
| | Sub Process | | | | |
| | Transasction | | | | |
| | Call Activity | | | | |
| | Event Sub Process | | | | |
| Gateways | Gateway | No | | | |
| Connecting Objects | Sequence Flow | No | | | |
| | Message Flow | | | | |
| | Data Association | | | | |
| | Association | | | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | | | |
| | Data Store | | | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | | | |
| | Pool | | | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | | | |
| Attributes | ResourceRole | No | | | **Yes** |
| | Resource | **Yes** | | | No |

Availability, Quantity and Role are applied to a BPMN Resource while the Selection Resource Parameter is applied to a BPMN ResourceRole.

## 5.4 Cost Parameters

| | | Cost Parameters | |
|---|---|---|---|
| | | fixedCost | unitCost |
| Events | Start Event | No | |
| | Intermediate Event | | |
| | End Event | | |
| Activities | Task | **Yes** | |
| | Sub Process | | |
| | Transasction | | |
| | Call Activity | | |
| | Event Sub Process | | |
| Gateways | Gateway | No | |
| Connecting Objects | Sequence Flow | No | |
| | Message Flow | | |
| | Data Association | | |
| | Association | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | |
| | Data Store | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | |
| | Pool | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | |
| Attributes | ResourceRole | No | |
| | Resource | **Yes** | |

The FixedCost Cost Parameter can be applied to Activities and the both FixedCost and UnitCost can be applied to BPMN Resource.

## 5.5 Property Parameters

| | | Property Parameters |
|---|---|---|
| | | property |
| Events | Start Event | **Yes** |
| | Intermediate Event | |
| | End Event | |
| Activities | Task | **Yes** |
| | Sub Process | |
| | Transasction | |
| | Call Activity | |
| | Event Sub Process | |
| Gateways | Gateway | No |
| Connecting Objects | Sequence Flow | **Yes** |
| | Message Flow | |
| | Data Association | No |
| | Association | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process |
| | Data Store | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process |
| | Pool | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process |
| Attributes | ResourceRole | No |
| | Resource | |

Property Parameters can be applied to Events, Activities, Sequence Flow and Message Flow.

## 5.6 Priority Parameters

| | | Priority Parameters | |
|---|---|---|---|
| | | interuptible | priority |
| Events | Start Event | No | |
| | Intermediate Event | | |
| | End Event | | |
| Activities | Task | **Yes** | |
| | Sub Process | **Yes - but only for activities without decomposition** | |
| | Transasction | | |
| Gateways | Gateway | No | |
| Connecting Objects | Sequence Flow | No | |
| | Message Flow | | |
| | Data Association | | |
| | Association | | |
| Data | Data Object | No - BPMN Data Elements have no impact on the Simulation or the Analysis of the Process | |
| | Data Store | | |
| Swimlanes | Lane | No - BPMN Swimlanes have no impact on the Simulation or the Analysis of the Process | |
| | Pool | | |
| Artifacts | Artifact | No - BPMN Artifacts have no impact on the Simulation or the Analysis of the Process | |
| Attributes | ResourceRole | No | |
| | Resource | | |

Priority Parameters can be applied only to Activities without decomposition.