Owl axioms from natural language specifications

|  |  |  |
| --- | --- | --- |
| **N°** | **Specification in natural language** | **Formalized Specification** |
| 1 | A **Gateway** MUST have either multiple *incoming* **Sequence Flows** or multiple *outgoing* **Sequence Flows** (i.e., it MUST merge or split the flow). (p.290) | Gateway **SubClassOf** (has\_incoming **min** 2 SequenceFlow) or (has\_outgoing **min** 2 SequenceFlow) |
| 2 | A **Gateway** with a gatewayDirection of mixed MUST have both multiple *incoming* and *outgoing* **Sequence Flows**. (p.290) | MixedGateway **equivalentTo** Gateway  **and** (has\_incoming **min** 2 SequenceFlow)  **and** (has\_outgoing **min** 2 SequenceFlow) |
| 3 | A **Gateway** with a gatewayDirection of converging MUST have multiple *incoming* **Sequence Flows**, but MUST NOT have multiple *outgoing* **Sequence Flows**. (p.290) | ConvergingGateway **equivalentTo** Gateway  **and** (has\_incoming **min** 2 SequenceFlow)  **and** (has\_outgoing **exactly** 1 SequenceFlow) |
| 4 | A **Gateway** with a gatewayDirection of diverging MUST have multiple *outgoing* **Sequence Flows**, but MUST NOT have multiple *incoming* **Sequence Flows**. (p.290) | DivergingGateway **equivalentTo** Gateway  **and** (has\_outgoing **min** 2 SequenceFlow)  **and** (has\_incoming **exactly** 1 SequenceFlow) |
| 5 | An **Event Gateway** MUST have two or more *outgoing* **Sequence Flows**. (p297) | EventBasedGateway **SubClassOf** (has\_outgoing **min** 2 SequenceFlow) |
| 6 | The *outgoing* **Sequence Flows** of the **Event Gateway** MUST NOT have a conditionExpression. (p.297) | EventBasedGateway **SubClassOf not**(has\_outgoing **some** ConditionalSequenceFlow) |
| 7 | The **Start Event** starts the flow of the **Process**, and thus, will not have any *incoming* **Sequence Flows** | StartEvent **SubClassOf** **not**(has\_incoming **some** SequenceFlow) |
| 8 | The **Start Event** should have at least one *outgoing* **Sequence Flow** | StartEvent **SubClassOf** (has\_outgoing **some** SequenceFlow) |
| 9 | The **Start Event** of an **Event Sub-Process** MUST have a defined *trigger*.  The **Start Event** *trigger* (EventDefinition) MUST be from the following types: Message, Error, Escalation, Compensation, Conditional, Signal, and Multiple (p.260). (p.177) | StartEventForEventBasedSubProcess **equivalentTo** StartEvent **and** (has\_eventDefinition **some** (CompensateEventDefinition or ConditionalEventDefinition or ErrorEventDefinition or EscalationEventDefinition or MessageEventDefinition or SignalEventDefinition)) |
| 10 | An **Event Sub-Process** MUST have one and only one **Start Event**. (p.177) | EventBasedSubProcess **SubClassOf** (has\_flowElements **exactly** 1 StartEventForEventBasedSubProcess) |
| 11 | An **Event Sub-Process** MUST NOT have any *incoming* or *outgoing* **Sequence Flows**. (p.177) | EventBasedSubProcess **SubClassOf not (**(has\_incoming **some** SequenceFlow) **or** (has\_outgoing **some** SequenceFlow)) |
| 12 | the **End Event** ends the flow of the **Process**, and thus, will not have any *outgoing* **Sequence Flows.** (p.246) | EndEvent **SubClassOf not** (has\_outgoing **some** SequenceFlow) |
| 13 | An **End Event** MUST be a target for a **Sequence Flow**.  An **End Event** MAY have multiple *incoming* **Sequence Flows**. (p.249) | EndEvent **SubClassOf** (has\_incoming **some** SequenceFlow) |
| 14 | Conditional Sequence Flow is a Sequence Flow that has a ConditionExpression. (p.97) | *ConditionalSequenceFlow* ***equivalentTo*** *SequenceFlow* ***and*** *(has\_conditionExpression* ***some*** *Expression)* |
| 15 | If a *conditional* **Sequence Flow** is used from a source **Activity**, then there MUST be at least one other *outgoing* **Sequence Flow** from that **Activity**. (p. 97) | Activity **SubClassOf not** (has\_outgoing **exactly** 1 ConditionalSequenceFlow) |
| 16 | **Conditional Sequence Flow**: A source **Gateway** MUST NOT be of type Parallel or Event (p.97) | SequenceFlow **SubClassOf** |
| 17 | The *default* **Sequence Flow** should not have a conditionExpression. (p.292) | DefaultSequenceFlow **DisjointWith** ConditionalSequenceFlow |
| 18 | A source **Gateway** MUST NOT be of type **Parallel** or **Event[[1]](#footnote-1)** (p.97) | SequenceFlow **SubClassOf** **not** (  (has\_conditionExpression **some** Expression)  **and** (has\_sourceRef **some**  (ParallelGateway **or** EventBasedGateway))) |
| 19 | Timer attributes (timeCycle, timeDate, and timeDuration) are mutually exclusive and only one attribute may be set at a time[[2]](#footnote-2) (p.274) | TimerEventDefinition **SubClassOf** ((has\_timeDate **some** Expression **and not** ((has\_timeDuration **some** Expression)  **or** ( has\_timeCycle **some** Expression )))  **or**  (has\_timeDuration **some** Expression **and** **not** ((has\_timeDate **some** Expression) **or** (has\_timeCycle some Expression )))  **or**  (has\_timeCycle **some** Expression and not ((has\_timeDate **some** Expression) or (has\_timeDuration **some** Expression )))) |
| 20 | The list of **BPMN** elements that MUST be used in an **Ad-Hoc Sub-Process: Activity**. (p.182) | AdHocSubProcess **SubClassOf** (has\_flowElements **some** Activity) |
| 21 | The list of **BPMN** elements that MUST NOT be used in an **Ad-Hoc Sub-Process: Start Event, End Event[[3]](#footnote-3)** (p.182) | AdHocSubProcess **SubClassOf** not (has\_flowElements **some** (StartEvent **or** EndEvent)) |
| 22 | A Timer Event is an **Event** that has exactly one **TimerEventDefinition**. (p.274) | TimerEvent **EquivalentTo** (Event and (has\_eventDefinition **exactly** 1 TimerEventDefinition)) |
| 23 | An **Intermediate** **Event** MUST be a  source for a **Sequence Flow**. (p. 259) | * IntermediateEvent **EquivalentTo**   (IntermediateCatchEvent **or** IntermediateThrowEvent)[[4]](#footnote-4)   * IntermediateEvent **SubClassOf** (has\_outgoing **some** SequenceFlow) |
| 24 | **None Events** are **Events** that do not have a defined EventDefinition. (p.272) | NoneEvent **EquivalentTo** (Event  **and** (**not** (has\_eventDefinition **some** EventDefinition)) |
| 25 | Default Sequence Flow is a Sequence Flow referenced by (target of) “has\_default” relation. | * has\_defaultElement **InverseOf** has\_default[[5]](#footnote-5) * DefaultSequenceFlow **EquivalentTo** (SequenceFlow **and** (has\_defaultElement **exactly** 1 (Activity **or** ComplexGateway **or** ExclusiveGateway **or** InclusiveGateway))) |

1. We have added this restriction to the SequenceFlow class rather than the ConditionalSequenceFlow class to state that it is inconsistent to have a SequenceFlow which is at the same time Conditional and has parallel or eventbased gateway source. Otherwise, it would be possible to have such a SequenceFlow classified as a NormalSequenceFlow. [↑](#footnote-ref-1)
2. We have rephrased this sentence to include three similar phrases, one for each attribute. [↑](#footnote-ref-2)
3. The original phrase was: “The list of **BPMN** elements that MUST NOT be used in an **Ad-Hoc Sub-Process: Start Event, End Event, Conversations** (graphically), **Conversation Links** (graphically), and **Choreography Activities**.” But we do not take Conversations and Choreography into account in our ontology. [↑](#footnote-ref-3)
4. Formalization has been performed by defining the class of IntermediateEvent, then adding a restriction on this class. [↑](#footnote-ref-4)
5. We have created a new ObjectProperty “has\_defaultElement” inverse of “has\_default” objectProperty. Then, we defined the DefaultSequenceFlow class as the SequenceFlows that have the “has\_defaultElement” relation. [↑](#footnote-ref-5)