

1 Annex A: Notation

This annex lists the elements with defined notations to be used when the element is shown in diagrams.

A notation for an EAST-ADL element consists of a symbol complemented with textual description of the element's properties and with other possible attached notational elements such as ports.

As a general rule for the textual properties each element is specified with its name, along with possible type name as well as metaclass name as follows: name : type <<metaclass>>. The name is the mandatory short name unless a longer user defined name of the element is given.

The notation uses some common guidelines like all prototypes are rectangles and types are rectangles with cutted corners. Color is the same for each kind of prototype and type: e.g., Actuators are blue, Sensors orange etc. For those elements that are not listed here the general notation is a solid-outline rectangle with the metaclass name at the top right.

The notation described here is a guideline and each tool may adapt and extend it, such as also show datatype for ports, indicate which elements have more detailed submodels, show error information in the symbol etc. By default, the notational elements are scalable enabling engineers to decide how much information will be shown in the notational symbol.

Similarly, tools may avoid repeating the name of the metamodel element in the label of symbol as that takes space from the actual model data. Repeating e.g., <<DesignFunctionPrototype>> for all DesignFunctionPrototypes or <<State>> for states is not required as this takes space from actual specification. It can be omitted or instead show a smaller icon.

1.1 Actuator (from HardwareModeling)

Actuator and HardwareComponentPrototype typed by Actuator share similar notation: solid-outline blue rectangle with double vertical borders. At the perimeter are shown the pins or ports. By default, in ports are at the left pointing towards the function, outports at the right pointing away from the function, and in/out ports at the bottom of the perimeter.

All types, including Actuator, is presented with a rectangle having cutted corners, and prototype is presented without cutted corners.

Actuator shows its name and metaclass. A HardwareComponentPrototype typed by Actuator shows its name, its type (the name of Actuator type), and metaclass.

Reading guide:

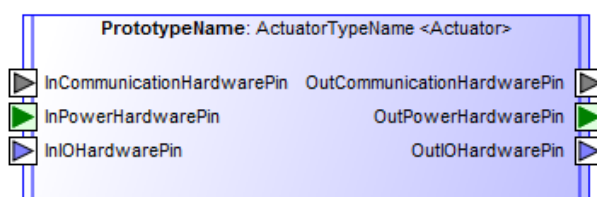
PrototypeName = the name of the prototype

ActuatoryTypeName = name of the type

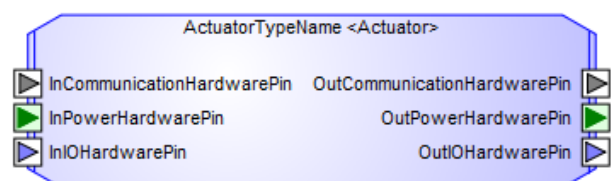
<Actuator> = name of the EAST-ADL concept, metaclass (optional)

Examples:

HardwareComponentPrototype
(typed by an Actuator)




Actuator type

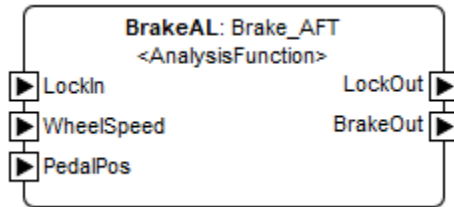


1.2 AnalysisFunctionPrototype (from FunctionModeling)

AnalysisFunctionPrototype is shown as a solid-outline rectangle containing the name (as in Section 1.1.1) and showing the ports at the perimeter of the symbol. By default, in ports are at the left pointing towards the function, outputs at the right pointing away from the function, and in/out ports at the bottom of the perimeter.


The color of the Function may change depending on the kind of its type (FunctionalDevice, AnalysisFunctionType).

Instead of showing the metaclass, an icon can be shown like: .



1.3 AnalysisLevel (from SystemModeling)

The Analysis Architecture is shown using the same symbol as used for AnalysisFunctionPrototype.

and showing the Analysis Level on top of the symbol or related icon, like: .

1.4 ArrayDatatype (from Datatypes)

The datatype ArrayDatatype is denoted using a rectangle symbol with optional keyword ArrayDatatype at the top of the symbol or array icon on the top right corner. Symbol may also show attributes of the type, like its MinLength etc.

1.5 ClampConnector (from Environment)

ClampConnector is shown as a solid line between the ports/pins of the system and its environment.

1.6 CommunicationHardwarePin (from HardwareModeling)

CommunicationHardwarePin is shown as a gray solid square pointing to its direction and in case of In&Out port showing two squares. Its name may outside the square.

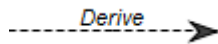


1.7 CompositeDatatype (from Datatypes)

The datatype CompositeDatatype is denoted using the white rectangle symbol with optional keyword CompositeDatatype or icon on the top right corner. Symbol shows the name of the element and may also show attributes of the element.

1.8 DeriveRequirement (from Requirements)

A DeriveRequirement relationship is shown as a dashed arrow between two Requirements. The Requirement at the tail of the arrow (the derived Requirement) depends on the Requirement at the arrowhead (the Requirement derived from).



1.9 DesignFunctionPrototype (from FuntionModeling)

DesignFuntionPrototype is shown as a solid-outline rectangle containing the name (as in Section 1.1.1) and showing the ports at the perimeter of the symbol. By default, in ports are at the left pointing towards the prototype, outports at the right pointing away from the prototype, and in/out ports at the bottom of the perimeter.

The color of the Function may change depending on the kind of its type (LocalDeviceManager, BasicSoftwareFunctionType, HardwareFunctionType, DesignFunctionType).

1.10 DesignLevel (from SystemModeling)

The DesignLevel is shown as a solid-outline rectangle containing the name, with its pins or ports on the perimeter. Contained entities may be shown with their connectors and allocations (White-box view).

1.11 EABoolean (from Datatypes)

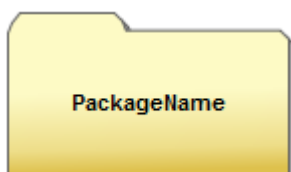
The datatype EABoolean is denoted using the rectangle symbol with keyword Boolean and icon on the top right corner.

1.12 EANumerical (from Datatypes)

The datatype EANumerical is denoted using the blue rectangle symbol with keyword Numerical or icon on the top right corner. Symbol may also show attributes of the element, like its min and max values.

1.13 EAPackage (from Elements)

EAPackage is shown as a folder symbol with a name of the package inside the symbol. The symbol may include an icon or text referring to the content of the package (e.g. requirements, dependability etc.).



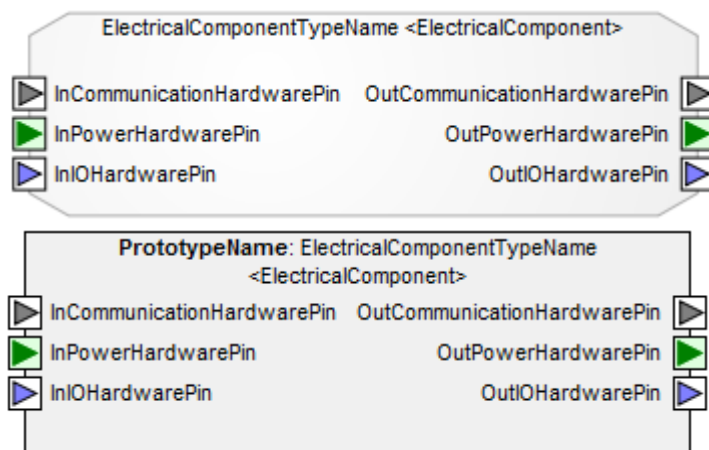
1.14 EAString (from Datatypes)

The datatype EAString is denoted using a green rectangle symbol with keyword string or icon on the top right corner.

1.15 ElectricalComponent (from HardwareModeling)

ElectricalComponentType and ElectricalComponentPrototype shares similar notation, but type is presented as a grey rectangle with cutted corners, and prototype is presented as a grey rectangle. At the perimeter are shown the pins or port. By default, in ports are at the left pointing towards the function, outputs at the right pointing away from the function, and in/out ports at the bottom of the perimeter.

Type shows its name and metaclass. Prototype shows its name, its type (aka name of ElectricalComponentType), and metaclass.

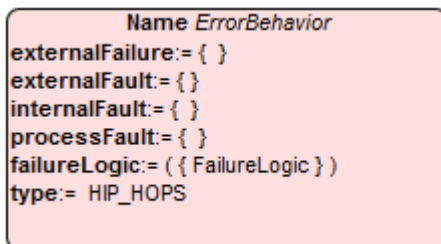


1.16 Enumeration (from Datatypes)

The datatype Enumeration is denoted using the gray rectangle symbol with keyword Enumeration and icon on the top right corner. Symbol shows element name and may also show attributes like its literal values, one per line, in the compartment of enumeration notation.

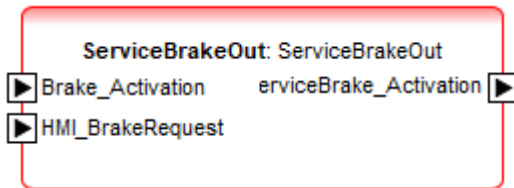
1.17 ErrorBehavior (from ErrorModel)

ErrorBehavior is shown as rounded rectangle with its name and metaclass name in the top of the symbol. Inside the symbol related failures, faults, or failure logic can be shown.



1.18 ErrorModelPrototype (from ErrorModel)

ErrorModelPrototype is shown with a solid red thick lined rounded rectangle with its name inside the symbol (see Section 1.1.1) and its ports or port groups on the perimeter. By default, in ports are at the left pointing towards the function, outports at the right pointing away from the function.



icon:

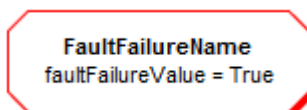
1.19 FailureOutPort (from ErrorModel)

FailureOutPort is shown as a solid square with a red arrow and letter 'O' inside. Its name appears outside the square.



1.20 FaultFailure (from SafetyConstraints)

FaultFailure is shown as a red rectangle with cutted corners. The name of the FaultFailure is shown in the middle of the symbol with the faultFailureValue.



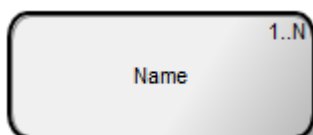
1.21 FaultInPort (from ErrorModel)

FaultInPort is shown as a solid square with a red arrow end and letter 'I' inside. Its name appears outside the square.



1.22 Feature (from FeatureModeling)

Feature is shown as a grey rounded rectangle with a name in the middle of the symbol, and cardinality in the upper right corner.



icon:

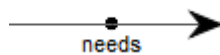
1.23 FeatureFlaw (from Dependability)

FeatureFlaw is shown as a green rounded rectangle with a name of the flaw in the middle of the rectangle.



1.24 FeatureLink (from FeatureModeling)

FeatureLink is described with a solid line using a black filled arrowhead. The kind of link (needs, optionalAlternative, mandatoryAlternative, suggests, impedes, custom) is described in the middle of the link.



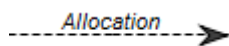
1.25 FeatureTreeNode (from FeatureModeling)

FeatureTreeNode is shown with a solid line between feature and one or more child features. Cardinality of the feature is shown at the child end with black (mandatory) or white (optional) circle.



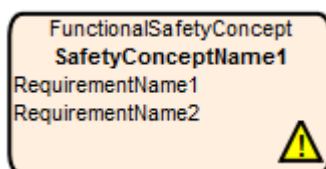
1.26 FunctionAllocation (from FunctionModeling)

A FunctionAllocation is shown as a dependency (dashed line) with an "allocation" keyword attached to it.



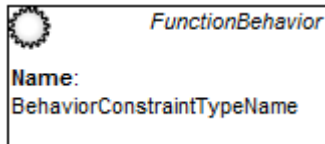
1.27 FunctionalSafetyConcept (from SafetyRequirement)

FunctionalSafetyConcept is described with a solid line rounded rectangle with icon on the bottom right. Contained requirements may be shown in the middle of the symbol along with its name.



1.28 FunctionBehavior (from Behavior)

FunctionBehavior appears as a solid-outline rectangle with "FunctionBehavior" at the top right or icon on the top left corner. The rectangle contains the name.



1.29 FunctionClientServerPort (from FunctionModeling)

A FunctionClientServerPort of kind server is shown as a solid square with a circle inside. Its name appears outside the square.

A FunctionClientServerPort of kind client is shown as a solid square with a half circle inside. Its name appears outside the square.






1.30 FunctionConnector (from FunctionModeling)

FunctionConnector is shown as a solid line.

1.31 FunctionFlowPort (from FunctionModeling)

FunctionFlowPort is shown as a solid square pointing to its direction and in case of In&Out port showing two squares. Its name may appear outside the square.



 (in),  (out),  (inout)

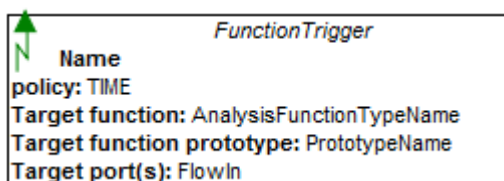
1.32 FunctionPowerPort (from FunctionModeling)

FunctionPowerPort is shown as a solid-outline rectangle with a lighting symbol inside and name of the port below the symbol.

(e.g. Papyrus:  , MetaEdit+:  Power)

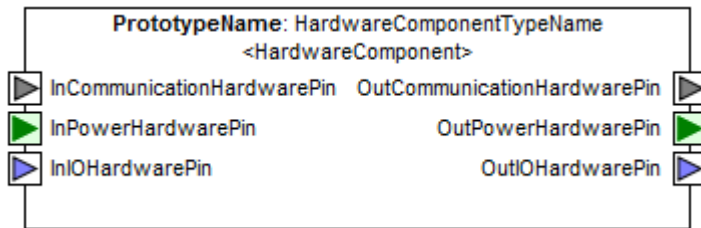
1.33 FunctionTrigger (from Behavior)

FunctionTrigger is shown as a rectangle with its name in the middle of the symbol and icon or metaclass name (FunctionTrigger) in the top of the symbol. Inside a symbol, properties of FunctionTrigger, like its policy or related functions, prototypes and ports can be listed.



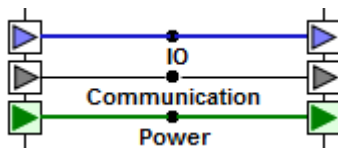
1.34 HardwareComponentPrototype (from HardwareModeling)

HardwareComponentPrototype is shown as a solid-outline rectangle. The rectangle contains the name, its type, metaclass (see Section 1.1.1) and its pins or ports on the perimeter. By default, in ports are at the left pointing towards the function, outports at the right pointing away from the function, and in/out ports at the bottom of the perimeter.



1.35 HardwareConnector (from HardwareModeling)

HardwareConnector is shown as a solid line between the pins of HardwareComponents. The line may show the name of the HardwareConnector and different Connector kinds may be illustrated with different style or color (like blue for IO, grey for Communication, and green for Power).



1.36 HardwarePort (from HardwareModeling)

HardwarePort is shown as a solid thick-line square with its name outside the square. Unlike other ports (and pins) it does not show any direction.

1.37 HardwarePortConnector (from HardwareModeling)

HardwarePortConnector is shown as a double-headed arrow showing all the HardwareConnectors included in the HardwarePortConnector. The name of HardwarePortConnector is shown middle in the symbol.

1.38 Hazard (from Dependability)

The Hazard is shown as a yellow solid-outline triangle with "Haz" at the top and its name.

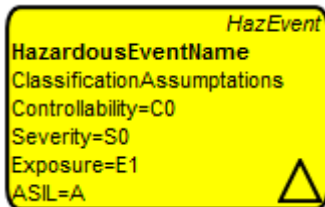


1.39 Hazard (from Dependability)

HazardousEvent is connected to one or more Hazards. This connection can be shown as a relationship line presented as a solid line with an open arrowhead.

1.40 HazardousEvent (from Dependability)

The HazardousEvent is shown as a yellow solid-outline rounded rectangle with "HazEvent" at the top right or icon. It shows the name of the HazardousEvent and optionally other related properties like Severity, ASIL level etc.



1.41 HazardousEvent (from Dependability)

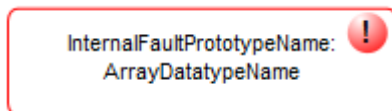
Relationships from HazardousEvent are shown as solid lines with open arrowheads. The kind of event, like Environment or Traffic, is shown at the end of the line.

1.42 ImplementationLevel (from SystemModeling)

The ImplementationLevel is shown as a solid-outline rectangle containing the name.

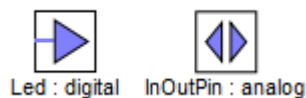
1.43 InternalFaultPrototype (from ErrorModel)

InternalFaultPrototype is shown with a solid red thin rounded rectangle with its name and type in the middle of the rectangle. Icon of exclamation mark in the upper right corner the InternalFaultPrototype.



1.44 IOHardwarePin (from HardwareModeling)

IOHardwarePin is shown as a blue solid-outline square pointing to its direction and in case of In&Out port showing two squares. Its name appears outside the square.



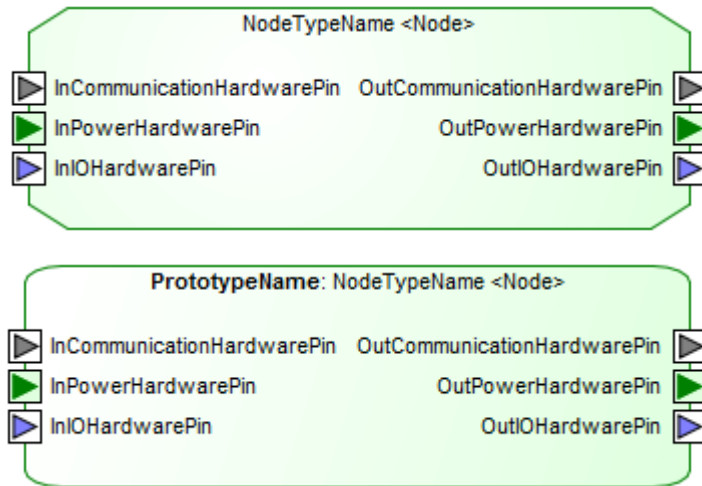
1.45 LogicalPortConnector (from HardwareModeling)

LogicalPortConnector is shown as a thick solid line between one or more HardwarePorts. The name, kind and speed of LogicalPortConnector can be shown in the middle of the line.

1.46 Node (from HardwareModeling)

NodeType and NodePrototype share a similar notation (green rectangle), but NodeType is presented as a rectangle with cutted corners, and NodePrototype is presented with a rounded rectangle. The ports or pins are shown at the perimeter. By default, in pins/ports are at the left pointing towards the function, outports at the right pointing away from the function, and in/out ports at the bottom of the perimeter.

NodeType shows its name and metaclass. NodePrototype shows its name, its type (aka name of NodeType), and metaclass.



1.47 PortGroup (from FunctionModeling)

FunctionConnectors connected to FunctionPorts of a PortGroup are graphically collapsed into a single line.

The PortGroup is rendered as its contained ports, but with a double outline.

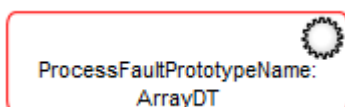
1.48 PowerHardwarePin (from HardwareModeling)

PowerHardwarePin is shown as a green solid square pointing to its direction and in case of In&Out port showing two squares. Its name appears outside the square.



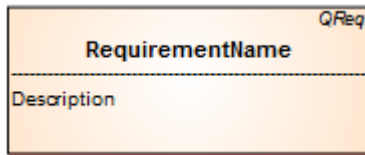
1.49 ProcessFaultPrototype (from ErrorModel)

ProcessFaultPrototype is shown with a solid red line rounded rectangle with its name and type in the middle of the rectangle. Icon of Wheel in the upper right corner mark the ProcessFaultPrototype.



1.50 QualityRequirement (from Requirements)

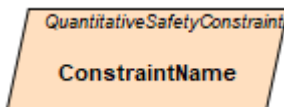
QualityRequirement is shown as a solid rectangle with 'QReq' or icon on the top right or and its name in the middle.



Requirement icon: 

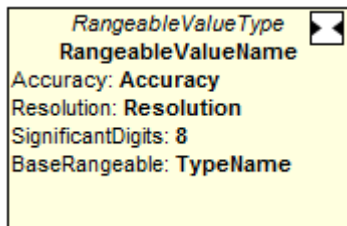
1.51 QuantitativeSafetyConstraint (from SafetyConstraints)

QuantitativeSafetyConstraint is shown as a rhombus with its name in the middle of the symbol. Its metaclass can be shown in the upper right corner.



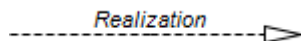
1.52 RangeableValueType (from Datatypes)

The datatype RangeableValueType is denoted using a yellow rectangle symbol with keyword RangeableValueType or icon on the top right corner. Symbol shows element name and may also show attributes like its resolution and accuracy.



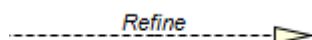
1.53 Realization (from Elements)

A Realization relationship is shown as a dashed line with a triangular arrowhead at the end that corresponds to the realized entity. The entity at the tail of the arrow (the realizing EAElement or the realizing ARElement) depends on the entity at the arrowhead (the realized EAElement).



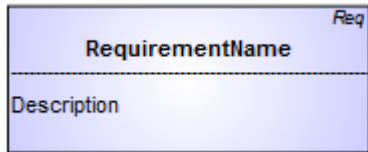
1.54 Refine (from Requirements)

A Refine relationship is shown as a dashed line with hollow arrowhead between the Requirements and EAElement. The entity at the tail of the arrow (the refining EAElement) depends on the Requirement at the arrowhead (the refined Requirement).



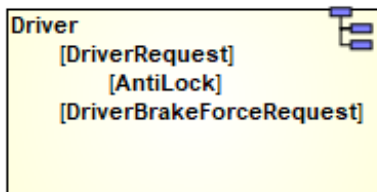
1.55 Requirement (from Requirements)

Requirement is shown as a solid rectangle with Req top right and its name. Inside a symbol, a description of the requirement can be shown.



1.56 RequirementsHierarchy (from Requirements)

RequirementsHierarchy is shown as a yellow solid-outline rectangle with icon on top right corner. Symbols shows the hierarchy name and contained requirement entities may also be shown inside (White-box view) within their hierarchy.



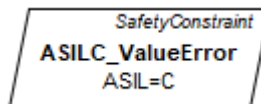
1.57 SafetyCase (from SafetyCase)

SafetyCase is described with a solid line rectangle with its name in the middle of the symbol. The upper border of the element shows a black and yellow line.



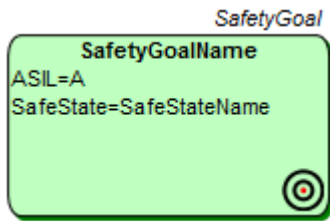
1.58 SafetyConstraint (from SafetyConstraints)

SafetyConstraint is shown as a rhombus with its name in the middle of the symbol. Its metaclass can be shown in the upper right corner and its attributes like ASIL in the middle of the symbol.



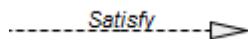
1.59 SafetyGoal (from SafetyRequirement)

SafetyGoal is a green rounded rectangle with icon on the bottom right and optional SafetyGoal text on top right. Inside the symbol is shown its name and its properties like ASIL or SafeState inside the symbol.



1.60 Satisfy (from Requirements)

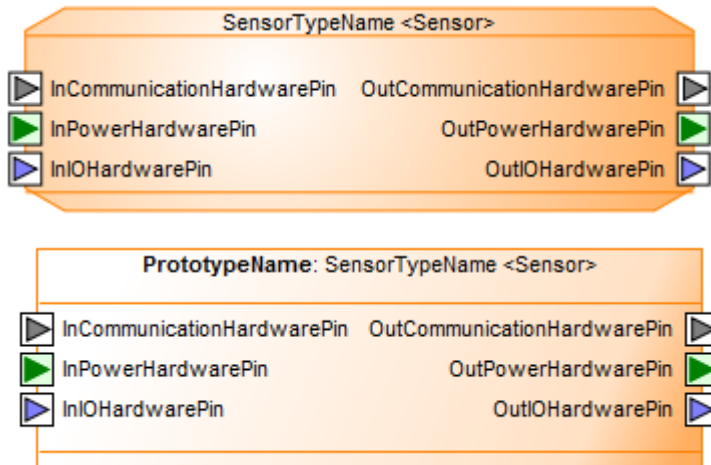
A Satisfy relationship is shown as a dashed line with an arrowhead at the end that corresponds to the satisfied Requirement or UseCaseUseCase. The entity at the tail of the arrow (the satisfying EAElement or the satisfying ARElement) depends on the entity at the arrowhead (the satisfied Requirement or UseCaseUseCase).



1.61 Sensor (from HardwareModeling)

SensorType and SensorPrototype shares a similar notation (double horizontal lines on top and bottom of the rectangle and orange color), but SensorType is presented as a rectangle with cutted corners, and prototype is presented as a rectangle. At the perimeter are shown the pins or ports. By default, in ports are at the left pointing towards the function, outputs at the right pointing away from the function, and in/out ports at the bottom of the perimeter.

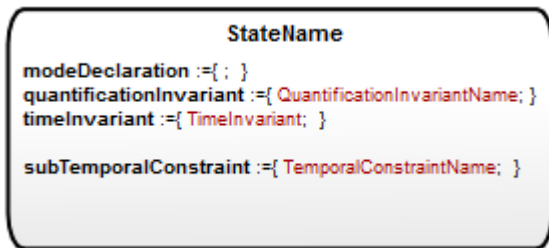
SensorType shows its name and metaclass. SensorPrototype shows its name, its type (aka name of SensorType), and metaclass (see Section 1.1.1).



1.62 State (from TemporalConstraints)

State is shown as a solid lined rounded rectangle with its name centered on top. Init states are shown with a double bordered rectangle. Depending on the state's properties the symbol may show additional information like if the state is error state or if it has invariants or constraints.





1.63 SystemModel (from SystemModeling)

The default notation for a SystemModel is a solid-outline rectangle containing the SystemModel's name, and with compartments separating by horizontal lines containing features or other members of the SystemModel. Contained entities may also be shown with their connectors (White-box view).

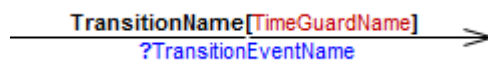
1.64 Timing (from Timing)

Timing is shown as rounded blue rectangle with its name in the middle of the symbol and optionally icon or metaclass name (Timing) in the top of the symbol.



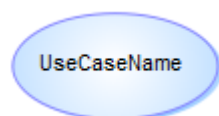
1.65 Transition (from TemporalConstraint)

Transition is shown as a solid line with an open arrowhead. The line shows the name of the transition, possible guard name and transition event name.



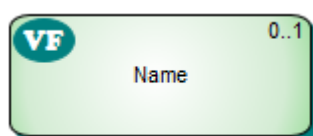
1.66 Use Case (from UseCases)


Use case is shown as a solid lined ellipse with the name of the Use Case in the middle.



1.67 VehicleFeature (from VehicleFeatureModeling)

VehicleFeature is shown as a green rounded rectangle with a name in the middle of the symbol, and cardinality in the upper right corner. 'VF' in the upper left corner may be used to indicate element.



Papyrus icon: 

1.68 VehicleLevel (from SystemModeling)

The VehicleLevel is shown as a solid-outline rectangle containing the name. Contained entities may be shown (White-box view).

1.69 Verify (from VerificationValidation)

A Verify relationship is shown as a dashed arrow between the Requirements and VVCase.

