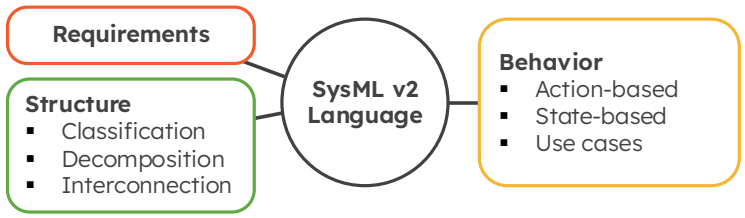
Cheat Sheet | SysML v2 Textual Notation Basics

Explore with SysIDE



Constructs & Expressions

Construct	Expression	Example
Line note	`// <text>`	`// this is a comment`
Block note	`//* <text> */`	`//* also comment */`
Comment	`/* <text> */`	`/* a comment */`
Documentation	`doc /* <text> */`	`doc /* Doc text */`
Definition	`<kind> def <name>`	`part def Vehicle`
Classifier	`classifier <name>`	`classifier Person`
Usage	`<kind> <name> : <type>`	`part bike : Vehicle`
Feature	`Feature <name> : <type>`	`feature age : Integer`
Specialization	`<child> :> <parent>`	`SportsCar :> Vehicle`
Redefinition	`:>> <property>`	`:>> total_wheels = 2`
Import	`<visibility> import <name>`	`private import ISQ::*;`
Multiplicity range	`<name> [<lowerBound>..<upperBound>]`	`Wheel [0..*]`
Multiplicity	`<name> [<count>]`	`wheel [4] : Wheel`
Qualified reference	`<namespace>::<member>`	`ISQ::MassValue`
Assignment	`<target> = <value>`	`mass = 1500 [kg]`
Comparison operators	`<  <=  ==  !=  >=  >  ===  !==`	`tank.fuelLevel == tank.maxFuelLevel`
Metadata	`@<metadata>`	`@ToolMetadata`

Keywords & shorthand notation

Keyword	Short form	Repeatable*
`defined by`	`:`	✓
`specializes`	`:>`	✓
`subsets`	`:>`	✓
`references`	`::>`	✗
`crosses`	`=>`	✗
`redefines`	`:>>`	✓

*a:b,c allowed
a::>b,c not allowed

Packages

Package Definition

```
package VehicleModel {
    public import VehicleParts;
    private import ISQ::MassValue;
    alias MV for ISQ::MassValue;
}
```

Imports

Alias

Parts

Part Definition

```
part def Engine;
part def Vehicle {
    attribute mass : MassValue;
    attribute wheels : Integer;
}
```

Specialization

```
part def SportsCar :> Vehicle {
    :>> wheels = 4;
}
```

Part Usage

```
part vehicle : Vehicle {
    :>> mass = 1500 [kg];
    part engine : Engine;
}
```

Items

Item Definition

```
item def Fuel{
    attribute fuelMass :> ISQ::mass;
}
```

Connections & Ports

Connection Definition

```
connection def DeviceConn {
    end part hub : Hub;
    end part device : Device;
    attribute bandwidth : Real;
}
```

Connection Usage

```
connection connection : DeviceConn {
    end part hub ::> mainSwitch;
    end part device ::> sensorFeed;
}
```

Port Definition

```
port def FuelingPort {
    attribute flowRate : Real;
    out fuelOut : Fuel;
    in fuelIn : Fuel;
}
```

Port Usage

```
port fuelTankPort : FuelingPort;
```

Brought by **Sensmetry**

Developer of SysIDE | Member of OMG Standards Development Organization

Actions

Action Definition

```
action def StartEngine {
    in ignitionSignal : Boolean;
    out status : EngineStatus;
}
```

Composite Action

```
action def Drive {
    action accelerate : Accelerate;
    first start then accelerate;
}
```

Perform Action

```
part vehicle : Vehicle {
    action driveVehicle : Drive;
    action pressGas {
        perform driveVehicle;
    }
}
```

State Definition

```
state def EngineStates {
    state Off;
    state Running;
}
```

Constraints & Requirements

Constraint Definition

```
constraint def IsFull {
    in tank : FuelTank;
    tank.fuelLevel == tank.maxFuelLevel
}
```

Constraint Usage

```
part def Vehicle {
    part fuelTank : FuelTank;
    constraint tankIsFull : IsFull {
        in tank = fuelTank;
    }
}
```

Requirement Definition

```
requirement def MassRequirement {
    subject vehicle : Vehicle;
    attribute massActual  : ISQ::MassValue;
    attribute massLimit   : ISQ::MassValue;
    require constraint {massActual <= massLimit}
}
```

Requirement Usage

```
requirement <R1> vehicleMass : MassRequirement {
    attribute :>> massActual = vehicle.mass;
    attribute :>> massLimit  = 1800 [kg];
}
```

Satisfy Requirement

```
satisfy R1 by vehicle;
```