Reconfiguration

In the FuelTank example (see "Progressive Reconfiguration"), the capacity of a FuelTank in a Vehicle context is reconfigured to 46 litres. In the WheelHubAssembly example, (see "Deep Reconfiguration"), the diameter of the Tire, Tire Bead and Rim, the inflationPressure of the WheelAssembly, etc., in a Truck context will be reconfigured to suit the truck.

When modeling you can use two methods of reconfiguration:

- Progressive Reconfiguration
- Deep Reconfiguration

Sample model

You can see an example of a Deep Reconfiguration project by opening context specific values.mdzip in the <md.install.dir>/samples/SysML directory.

Progressive Reconfiguration

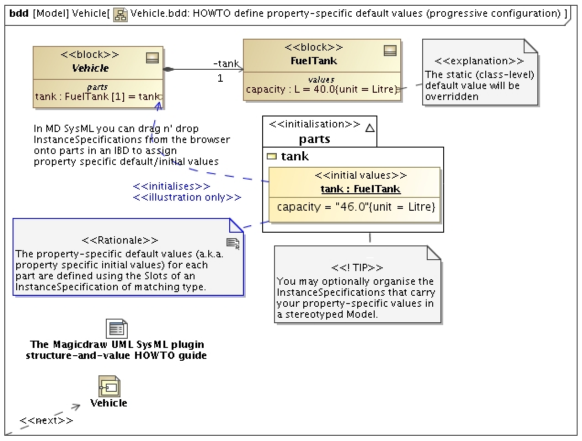
Progressive Reconfiguration enables SysML to handle a wide range of systems engineering configuration tasks. Progressive Reconfiguration continuously applies the following values:

- Static class-level default values.
- Inherited Property-specific initial values.
- Redefined Property-specific initial values.
- Property-specific initial values.

Property-specific initial values are specific to the usage of a Block as a Part Property in a higher context (i.e. another structured block or “assembly”). If there are many Part Properties of the same type, these Part Properties may have different property-specific default values and will then be initialized differently.

For more information about working with the Report Wizard, see the MagicDraw ReportWizard UserGuide.pdf Property-specific initial values are managed by the higher-context structured block, which owns the Part Properties that initialize or configure their (possibly different) values on instantiation. For example, the generic capacity of a FuelTank (not any particular one) is 40 liters (class-level default value). For a vehicle, however, the generic capacity of its FuelTank is 46 liters. An abstract Vehicle block will thus configure its tank:FuelTank part property by initializing it with a new capacity value. This can be done with Progressive Reconfiguration that will assign the instance specification tank:FuelTank to the property tank:FuelTank of the Vehicle block.

An example of Progressive Reconfiguration is when the values of y and z of a Location are reconfigured to 1 in the Thing context. Thus, the “values (Thing)” compartment in the l:Location part (in the Thing package) will display 1 as the values of y and z.


bdd [Model] Vehicle[Vehicle.bdd: HOWTO define property-specific default values (progressive configuration)]
<<block>>
Vehicle
parts
tank : FuelTank [1] = tank
-tank
1
<<block>>
FuelTank
values
capacity : L = 40.0{unit = Litre}
<<explanation>>
The static (class-level) default value will be overridden
In MD SysML you can drag n' drop InstanceSpecifications from the browser onto parts in an IBD to assign property specific default/initial values
<<initialisation>>
parts
tank
<<initial values>>
tank : FuelTank
capacity = "46.0"{unit = Litre}
<<initialises>>
<<illustration only>>
<<Rationale>>
The property-specific default values (a.k.a. property specific initial values) for each part are defined using the Slots of an InstanceSpecification of matching type.
<<! TIP>>
You may optionally organise the InstanceSpecifications that carry your property-specific values in a stereotyped Model.
The Magicdraw UML SysML plugin structure-and-value HOWTO guide
Vehicle
<<next>>

Deep Reconfiguration

Deep Reconfiguration enables you to configure deep-nested part(s) with context-specific value(s). Consider, for example, the case of a truck reusing a complex WheelHubAssembly for three pairs of wheels, each with different characteristics. Although the basic WheelHubAssembly might be suitable for a range of vehicles (a car, touring car, and minivan), it is not nearly suitable for a large truck. Some of the WheelHubAssembly parts and subparts required for a truck are larger and must be stronger to handle heavy loads. They include:

- the diameter of the Tire, TireBead, and Rim will be larger.
- the inflationPressure value of the WheelAssembly will be higher.
- the LugBoltJoint will be subject to greater torque and boltTension.
- the LugBoltThreadedHole will have larger lugBoltSize and threadSize.

In this case, Progressive Reconfiguration will fail because the new configuration requirements “cascade” throughout the entire complex WheelHubAssembly from the outermost context to the deepest part. Since no Progressive Reconfiguration approach can handle this deep reconfiguration of complex assemblies, you need to use Deep Reconfiguration.

You can start with a completely new TruckWheelHubAssembly that configures a completely new TruckWheelAssembly, right down to a TruckLugBoltJoint. However, you could use, instead, SysML PropertySpecificType strategy, which is a set of “on-the-fly” extensions (subtypes) of each Block used in a complex assembly hierarchy, to afford a point of redefinition of the Part Properties and their Value Properties as required. See the ‘PropertySpecificType‘ section in OMG SysML specifications.

An example of Deep Reconfiguration is when the value of x of a Location in the UniverseContext package is reconfigured to 3 in the UniverseContext context. Thus, the “values (UniverseContext)” compartment in the l:Location part (in the t1:Thing part in the UniverseContext package) will display 3 as the value of x. If UniverseContext is selected, the value of z, instead of x, will be reconfigured to 2.

For more information about Deep Reconfiguration, see training.nomagic.com.