Model-Based Product Line Engineering

Before you start working with Model-Based Product Line Engineering (MBPLE), use the MBPLE Profile.mdzip into your project. The profile brings the necessary stereotypes.

Then there are 4 large functionality groups:

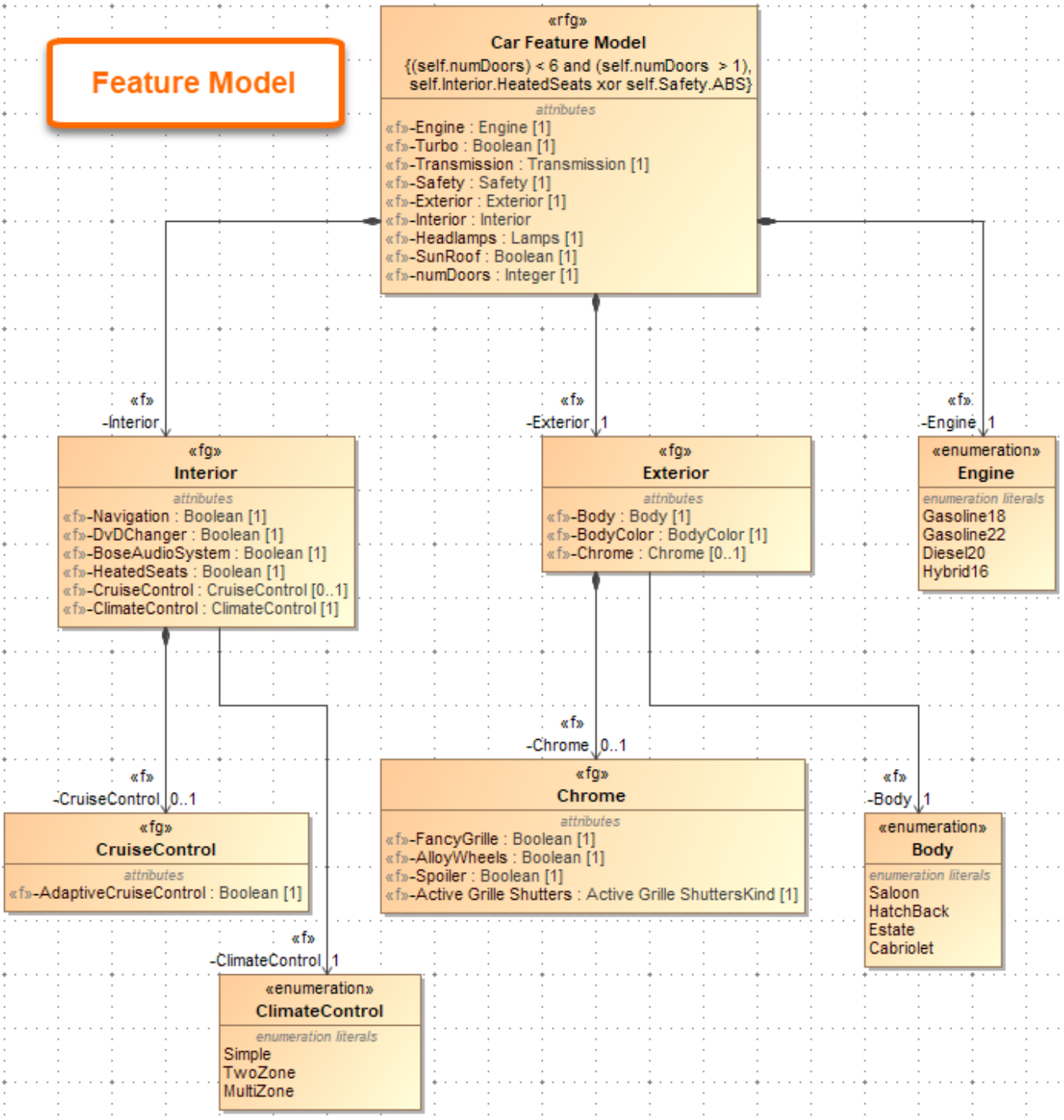
Defining Feature Model

The feature model is a simple UML class model. The feature model root is a class with the stereotype «RootFeatureGroup» applied.

Each feature that can be chosen or not (yes/no features) is modeled as UML property with the type *Boolean*. Each feature that can have multiple alternative choices is modeled as UML property with enumerated type. The alternatives are modeled as enumeration literals. These UML properties must have Feature stereotype applied.

Features can be placed in the root feature group (class) directly or can be placed into a subgroup. In the latter case, a subgroup is modeled as a separate UML class with the stereotype «FeatureGroup» applied. The subgroup is connected to the root feature group with composition. The composing property must have the stereotype «Feature» applied.

There can be multiple grouping levels. Hereby, features are organized into a feature tree (starting from root feature group).



An example of the feature model

Defining Configurations

The feature model is a class model. To configure the feature model, you need to create the instance model of that class model. You can use any modeling tool means for Instance Specification creation for the configurations.

If the feature model is flat (no nesting, all features are created directly under the root feature group) then the easiest way to create configurations is through the Instance table.

To create the configuration using Instance table

---

1. Create an instance table.

2. Set your root feature group as the classifier
3. Fill the table out.

Each row of the table is a separate configuration.

If the feature model is more complicated, has a deep hierarchy, the easiest way to create configurations is through the **Automatic Instantiation Wizard**.

To create the configuration using Automatic Instantiation Wizard

---

1. Right-click your root feature group class.
2. From the shortcut menu, select **Tools** > **Create Instance**
3. Follow the steps of the wizard.

Defining System Model with Variation Points

This step does not need configurations. It can be done before/in parallel with defining configurations.

Model your system with the normal SysML means. Use requirements, BDD, IBD, activity diagrams – whatever is necessary.

Whenever some part of your system model is variable, you need to demarcate that part with **Variation Points**.

If the model element (requirement, block, part, action, etc.) does not exist in some variants of your system, then demarcate it with the **Existence** variation point.

To demarcate an element with the Existence variation point

- Right-click the element and from the shortcut menu, select **Variants Modeling** > **Add Variation Point** > **Existence**.

This action will demarcate the system model element with a variation point. A little (v) icon will appeal to it.

Physically, in the model, the variation point is stored as a special constraint with «**ExistenceVariationPoint**» applied to it. Constraint's constrainedElement = your system model element.

Next, you need to specify feature impacts – i.e. specify which features from the feature model influence this variation point.

To specify the features from the feature model

---

1. In the variation point specification window, go to the **Feature Impacts** panel of the variation point.
2. Click the **Add** button to specify the features.

Then you need to construct a condition (*Boolean* expression) for the variation point. This is specified in the **Expression** field of the variation point specification window. Use any of the scripting languages executable by the modeling tool (JavaScript, Python, Ruby, etc.). Oftentimes, if your expression is simple/trivial you can skip the last step. If you keep the Expression field empty, the modeling tool **will infer** the expression for you. If your variation point depends on one or more *Boolean* features, say, X, Y, and Z, and you keep the expression field empty, the modeling tool will infer the expression: "X and Y and Z".

The Existence variation point kind is the most common. There are other kinds of variation points – the ones that can modify field values.

Putting PLE to use

Once you specify the variation points, you have functions that make use of variant specifications.

There are two functions:

- Variant highlight
- Variant realization

Variant highlight

You can highlight a particular variant in the diagrams. Use the drop-down menu in the **Variants** toolbar (if it is not visible, switch it on by right-clicking the empty space on the toolbar and checking Variants). The drop-down menu contains all the configurations that you have created (see section Defining Configurations). Once you select one configuration, variable elements (the ones with variation points) will be colored in red/green/yellow on the diagram depending on whether element exists (green), does not exist (red), or is changed (yellow) in the selected configuration.

Variant Realization

You can change the model that you have to the particular variant. This is done through the Variant Realization transformation.

To change the model to the particular variant

---

1. In the main toolbar, click **Tools** > **Transformations** > **Variant Realization**
2. Select the scope of the transformation – entire model or some packages,
3. Click **Next** (skip step 3 of the wizard – it is not relevant for this transformation type).
4. Select the configuration that you want to be realized.
5. Click **Finish**.

The model will be changed (elements modified/removed) to a particular variant of the system according to what variation points stipulate. After you have realized variant you can do anything that you do with ordinary system models - analysis, simulation, reporting, publishing, etc.

Be careful to NOT overwrite your initial project (which contains 150% model).

- If you are working with local files, use **Save As** action to save the system model variant into a different project file.
- If you are working on the server project, you can use **Commit to Branch** (menu **Collaborate** > **Commit to Branch**) action to store your variant model in a branch (while the 150% model is kept in the trunk).

Related pages

- Decomposing model
- Instance table
- Automatic Instantiation Wizard
- Specifying criteria for querying model