Behavior

During syntax tree manipulation, common operations are often extracted to utility methods in order to simplify the task and reuse functionality. It is possible to extract such utilities into static methods or create node wrappers holding the utility code in virtual methods. However, in MPS a better solution is available: the behavior language aspect. It makes it possible to create virtual and non-virtual instance methods, static methods, and concept instance constructors on nodes.

Concept instance methods

A Concept instance method is a method, which can be invoked on any specified concept instance. They can be both virtual and non-virtual. While virtual methods can be overridden in extending concepts, non-virtual ones cannot. Also a virtual concept method can be declared abstract, forcing the inheritors to provide an implementation.

Concept instance methods can be implemented both in concept declarations and in concept interfaces. This may lead to some method resolution issues. When MPS needs to decide, which virtual method to invoke in the inheritance hierarchy, the following algorithm is applied:

- If the current concept implements a matching method, invoke it. Return the computed value.
- Invoke the algorithm recursively for all implemented concept interfaces in the order of their definition in the implements section. The first found interface implementing the method is used. In case of success return the computed value.
- Invoke the algorithm recursively for an extended concept, if there is one. In case of success return the computed value.
- Return failure.

Overriding behavior methods

In order to override a method inherited from a super-concept, use the Control/Cmd + O keyboard shortcut to invoke the Overri de dialog. There you can select the method to override. By typing the name of the desired method to override you narrow down the list of methods.

Concept constructors

When a concept instance is created, it is often useful to initialize some properties/references/children to the default values. This is what concept constructors can be used for. The code inside the concept construction is invoked on each instantiation of a new node of a particular concept.

> The node's constructor is invoked before the node gets attached to the model. Therefore it is pointless to investigate the node's parent, ancestors, children or descendants in the behaviour constructor. These calls will always evaluate to null. You should define NodeFactories (Editor Actions) in order to have your nodes initialized with values depending on their context within the model.

Concept static methods

Some utility methods do not belong to concept instances and so should not be created as instance methods. For concept-wide functionality, MPS provides static concept methods. See also Constraints

Previous Next