Constraints

Related Pages:

- Simple Mapping of Attributes
- Simple Mapping of Classes
- More Complex Mappings
- Mapping with a Mapping Handler
- Constraints

Constraints are independent model elements in UML. They can be created using the **New Element** dialog (see below). Typically, constraints are put into the classes they are constraining. However, one constraint can also be used for more than one class. In this case, put the constraint in an appropriate package.
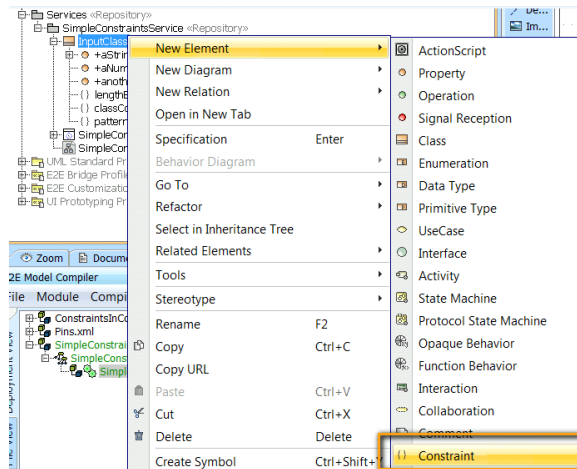

Each constraint is specified by a name, a specification and a list of constrained elements:
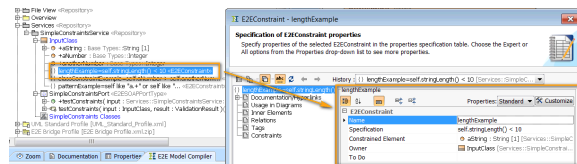

The constraint specification contains Action Script expressions that refer to the current context using the keyword `self`. The current context depends on the constrained element. If this element is an attribute - as in the example above - `self` refers to this attribute. If the constrained element is a class, `self` refers to an instance of this class at runtime. Constraints can be displayed in class diagrams if the presentation option **Show Constraint** is set:
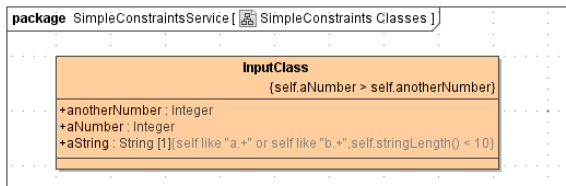

After having specified the constraints on a class and its attributes we can validate these constraints by using the <<ValidateConstraints>> action:
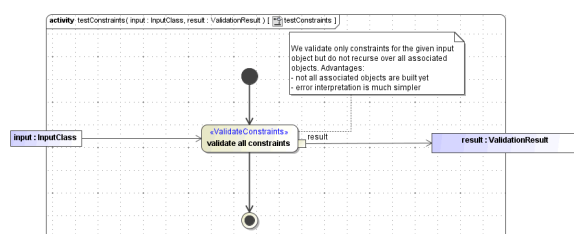
The validate constraints action takes as input any class that has constraints applied to it. Then all constraints are evaluated for this class. The result of the validation is an object of type **ValidationResult**t hat contains a list of all violated constraints (**ConstraintViolation[]**), for example:
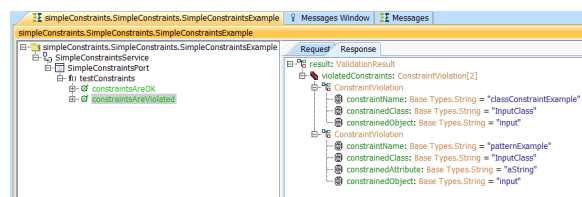
In the examples shown so far, the validation context has been either a class or an attribute being referred to by `self`. Sometimes it might be handy to give additional information to the constraint. For example, the following constraints also refers to a **constraintX** object:
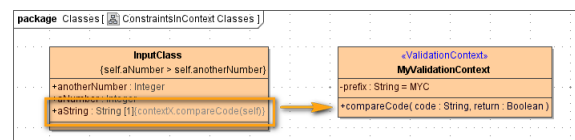
The constraint applied to **aString** uses the **objectX** . This object must be an instance of a <<ValidationCo ntext>> class that is provided as input to the <<ValidateConstraints>> action as shown in the figure below: