Constructs for Meta Properties Modeling in Modelica

Hilding Elmqvist[1], Hans Olsson[1], Martin Otter[2]

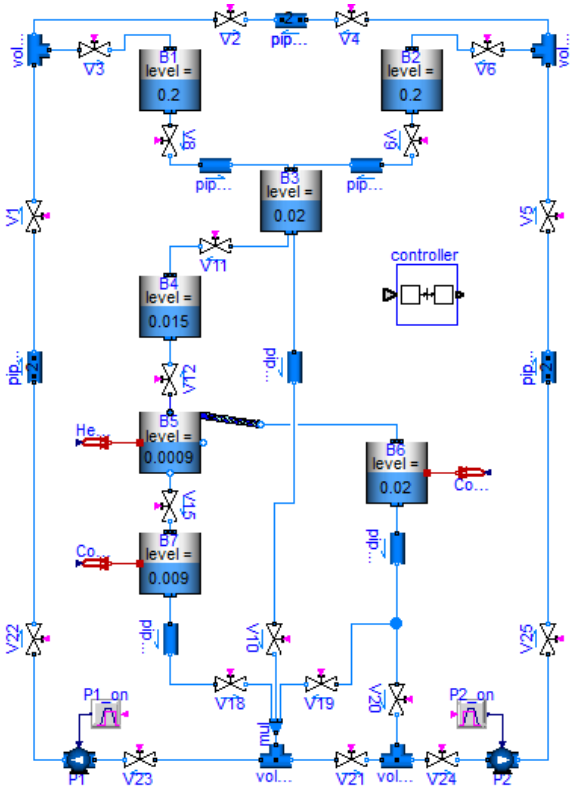
[1]Dassault Systemes, Sweden, {Hilding.Elmqvist, Hans.Olsson}@3ds.com
[2]Institute of System Dynamics and Control, DLR, Germany, Martin.Otter@dlr.de

This article proposes two new language constructs for meta-properties modeling in Modelica:

1. Accessing all instances of a given class in a simulation model with an extended iterator, e.g. Real u[:] = {c.v for c in Class};
2. Extracting in a convenient way the desired information from such instances by allowing to pass type compatible model instances as arguments to functions. For example: Real r=get(sub); where get(..); is a function call with a record input argument and sub is an instance of a model that contains all elements of the record as public variables.

Both extension proposals can be formally described with rewriting rules.

In several applications the usefulness of the proposed features are shown. In particular global properties of a model can be computed, such as total power, total mass, total center of mass, or kinetic and potential energy of a multi-body system. An important application is to bind behavioral models and requirement models in a convenient way, for example checking requirements for all instances of a class in a behavioral model, without changing the behavioral model. An example is given below for a fluid system, where requirements shall be checked for all pumps present in this system:



The following statement declares instance req of class PumpRequirements and passes all instances of class PrescribedPump that are present in the fluid system to the left. The req instance checks whether all pumps fulfill the defined requirements, e.g,
when in operation, the pump should not cavitate:

```
PumpRequirements req(
  obs={fromPrescribedPump(p,
                 getInstanceName())
       for p in  PrescribedPump});
```