Modeling Hints for Modelica External Interfaces

19th Modelisax Meeting

Thomas Beutlich

2018-08-15

get it right®

www.esi-group.com

Configuration of String parameters for external files

String parameters for external files

Introduction

- String parameters refer to files to be used as
  - parameter source
  - look-up table
  - data base
  - external fluid
  - log-file
  - applications to run (python etc.)
  - FMU
  - …
- Examples:
  - *Modelica.Blocks.Sources.CombiTimeTable.fileName*
  - *Modelica.Utilities.Streams.readFile.fileName*
  - *Modelica.Utilities.Streams.writeRealMatrix.fileName*

String parameters for external files

Parametrization

- Absolute path
  - Example: *"C:\myFile.ext"* or *"/home/user/myFile.ext"*
  - Quick and convenient way for debugging
  - Non-portable models between different computers or OSs
- Relative path to Modelica package by modelica URI
  - Example: *Modelica.Utilities.Files.loadResource("modelica://Package/Resources/myFile.ext")*
  - Needs to be called as argument of *Modelica.Utilities.Files.loadResource* for conversion to absolute path
  - Portable Modelica packages, but non-portable translated simulation models
- Relative path to Modelica model file by *classDirectory*
  - Example: *classDirectory() + "myFile.ext"*
  - Non-standard Modelica extension
  - Portable Modelica files, but non-portable translated simulation models

String parameters for external files

Parametrization ctd.

- Relative path to working directory
  - Example: *"../Data/myFile.ext"*
  - Dependency on working directory
    - Insecure if working directory is unknown or changed (by purpose or side-effect)
  - Portable Modelica file and portable translated simulation model, if external files are copied
- Relative path to Modelica model file and simulation model
  - Solution: *if Modelica.Utilities.Files.exist("myFile.ext") then "myFile.ext" else classDirectory() + "myFile.ext"*
  - Requires MSL v3.2.2, where *Modelica.Utilities.Files.exist* is marked as impure function (by annotation *Impure*)
  - Portable Modelica file and portable translated simulation model, if external files are copied

Disguise implementation details in member functions

Record member function taking external object argument

Problem

```
/* Library */
record R
  parameter String fileName = "" "External file";
  final parameter ExtObj obj = ExtObj(fileName) "Implementation detail";
  function f
    input ExtObj obj;
    input Real x;
    output Real out;
    external "C" out = ExtFun(obj, x);
  end f;
end R;

/* Model */
parameter R r(fileName="C:\myFile.ext");
Real y = r.f(r.obj, time);
```

- Problem: Want to disguise the implementation as external object to user

Record member function taking external object argument

Solution

```
/* Library */
package Interfaces "Interfaces"
  partial record RBase
    replaceable function f = Interfaces.fBase;
  end RBase;

  partial function fBase
    input ExtObj obj;
    input Real x;
    output Real out;
  end fBase;
end Interfaces;
```

Record member function taking external object argument

Solution ctd.

```
/* Library */
record R
  parameter String fileName = "" "External file";
  final parameter ExtObj obj = ExtObj(fileName) "Implementation detail";
  extends Interfaces.RBase(
    redeclare final function f = f(obj=obj));
end R;

function f
  extends Interface.fBase;
  external "C" out = ExtFun(obj, x);
end f;

/* Model */
parameter R r(fileName="C:\myFile.ext");
Real y = r.f(time);
```

Record member function taking external object argument

Summary

- Definition of clean Interfaces for records and function enables inheritance, and thus possibility of function redeclaration
- Legal Modelica
- Tested successfully in Dymola and SimulationX
- Fails in OpenModelica with „cyclic dependency“ error

Thank you!