Using Existing Modelica Models in Modeling with ModelicaML

Wladimir Schamai, EADS Innovation Works

February 6, 2012

Table of Contents

- Introduction
- Motivation
- Typical Usage Scenario
- Live Demo
- OMC API Enhancements
- Conclusion

Introduction / Motivation

- ModelicaML integrates a subset of the UML and the Modelica language
- vVDR (Virtual Verification of Designs against Requirements) is a method that enables a model-based design verification against requirements
- vVDR is supported in ModelicaML

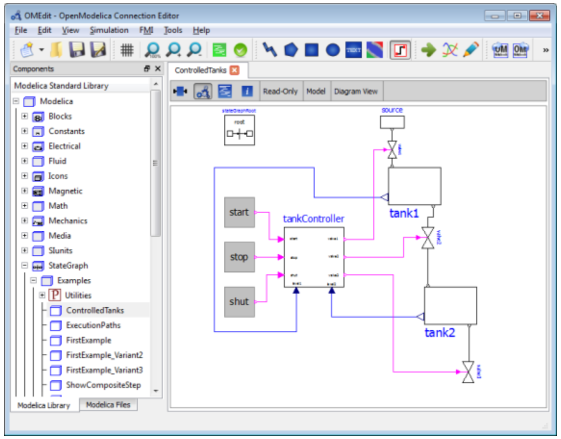
- How to enable the **usage of existing Modelica models in ModelicaML**?
  - E.g. Libraries or models that are created using Modelica tools

Typical Usage Scenario

- Use a **Modelica** tool to:
  - Develop system design models
  - Simulate models

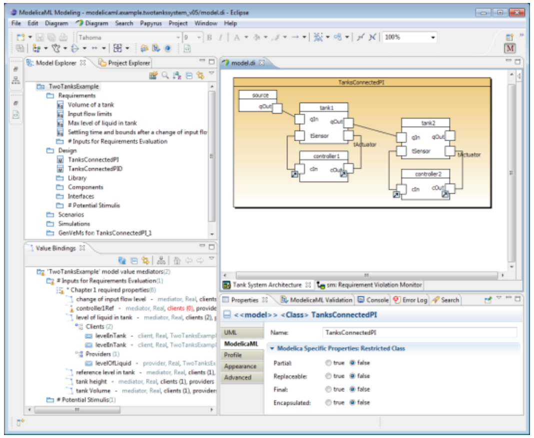
- Use a **ModelicaML** tool to:
  - Import Modelica models
  - Formalize/model requirements, model test / verification scenarios
  - Compose verification models, simulate verification models and generate reports
  - Visualize dependencies using UML graphical notation (e.g. inheritance)

Modelica Tool



Import/sync.

ModelicaML Tool


ModelicaML Modeling - modelicaml.example.twotankssystem_v05/model.di - Eclipse
File
Edit
Diagram
Diagram
Search
Papyrus
Project
Window
Help
Tahoma
100%
Model Explorer
Project Explorer
TwoTanksExample
Requirements
Volume of a tank
Input flow limits
Max level of liquid in tank
Settling time and bounds after a change of input flo
# Inputs for Requirements Evaluation
Design
TanksConnectedPI
TanksConnectedPID
Library
Components
Interfaces
# Potential Stimulis
Scenarios
Simulations
GenVeMs for: TanksConnectedPI_1
Value Bindings
'TwoTanksExample' model value mediators(2)
# Inputs for Requirements Evaluation(1)
* Chapter 1 required properties(6)
change of input flow level - mediator, Real, clients
controller1Ref - mediator, Real, clients (0), provide
level of liquid in tank - mediator, Real, clients (2),
Clients (2)
levelInTank - client, Real, TwoTanksExampl
levelInTank - client, Real, TwoTanksExampl
Providers (1)
levelOfLiquid - provider, Real, TwoTanksEx
reference level in tank - mediator, Real, clients (1),
tank height - mediator, Real, clients (1), providers
tank Volume - mediator, Real, clients (1), provider
# Potential Stimulis(1)
model.di
TanksConnectedPI
source
qOut
tank1
qIn
qOut
tSensor
tActuator
controller1
cIn
cOut
tank2
qIn
qOut
tSensor
tActuator
controller2
cIn
cOut
Tank System Architecture
sm: Requirement Violation Monitor
Properties
ModelicaML Validation
Console
Error Log
Search
<<model>> <Class> TanksConnectedPI
UML
ModelicaML
Profile
Appearance
Advanced
Modelica Specific Properties: Restricted Class

Name: TanksConnectedPI

Partial: ○ true ● false

Replaceable: ○ true ● false

Final: ○ true ● false

Encapsulated: ○ true ● false

Concept

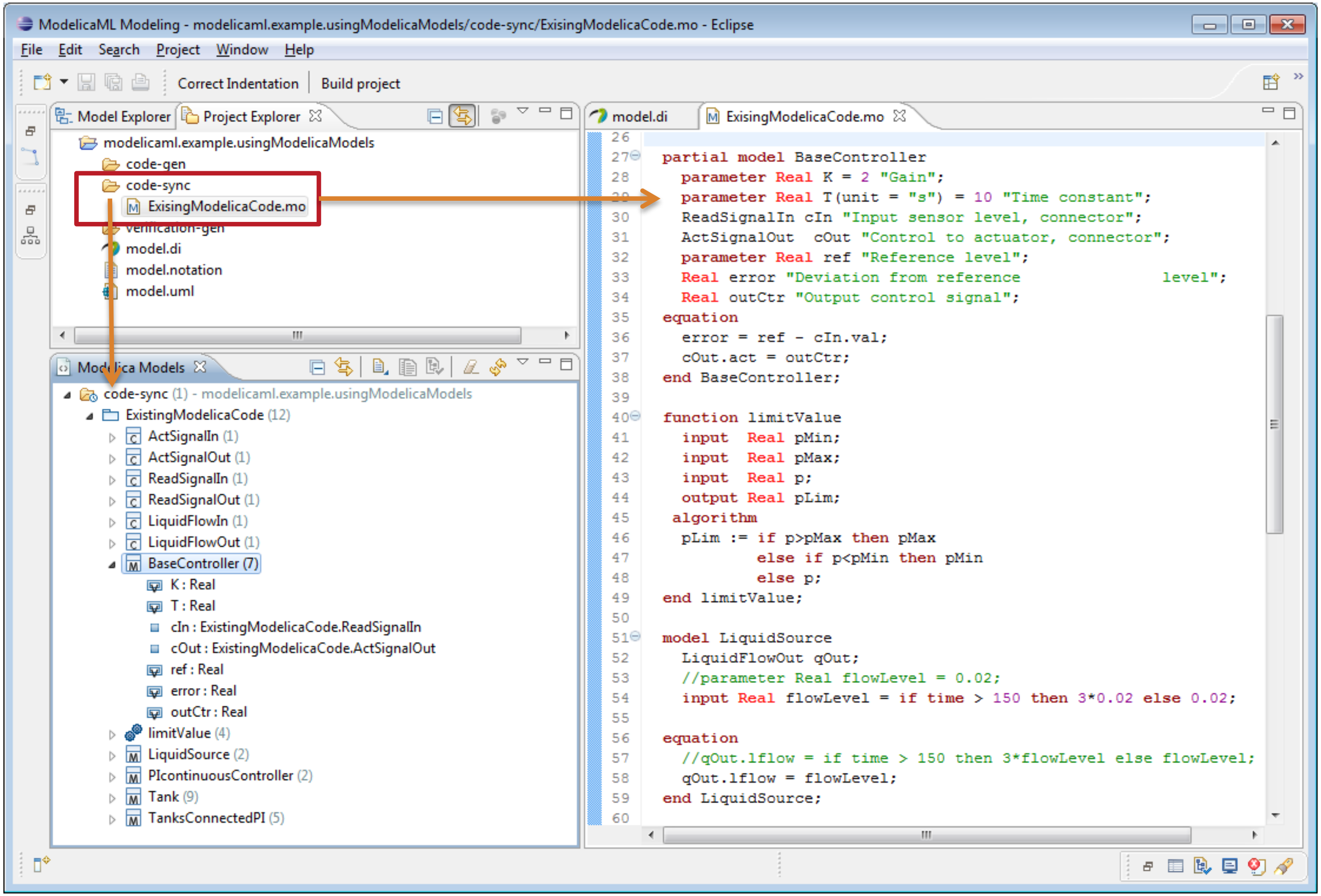
- Modelica models can be stored in the “**code-sync**” folder in ModelicaML Eclipse projects

- A dedicated viewer allows the browsing of the contained Modelica models
- A dedicated helper translates Modelica models from the “code-sync” into ModelicaML and mark them as “**proxies**”
  - Restriction: the top level Modelica models must be packages and not have any import or extends relations
- The translated models can be **synchronized** with the ModelicaML proxies whenever the Modelica models have been modified
- When synchronizing any identifiable element is updated, other are re-created (references will get lost)

- The created “proxies” can be used in ModelicaML models (i.e. referenced, instantiated)

- No code is generated from “proxies” classes
- For the simulation the code from both folders must be loaded
  - the generated ModelicaML model code from “**code-gen**” folder
  - and the code from the “**code-sync**” folder

Implementation

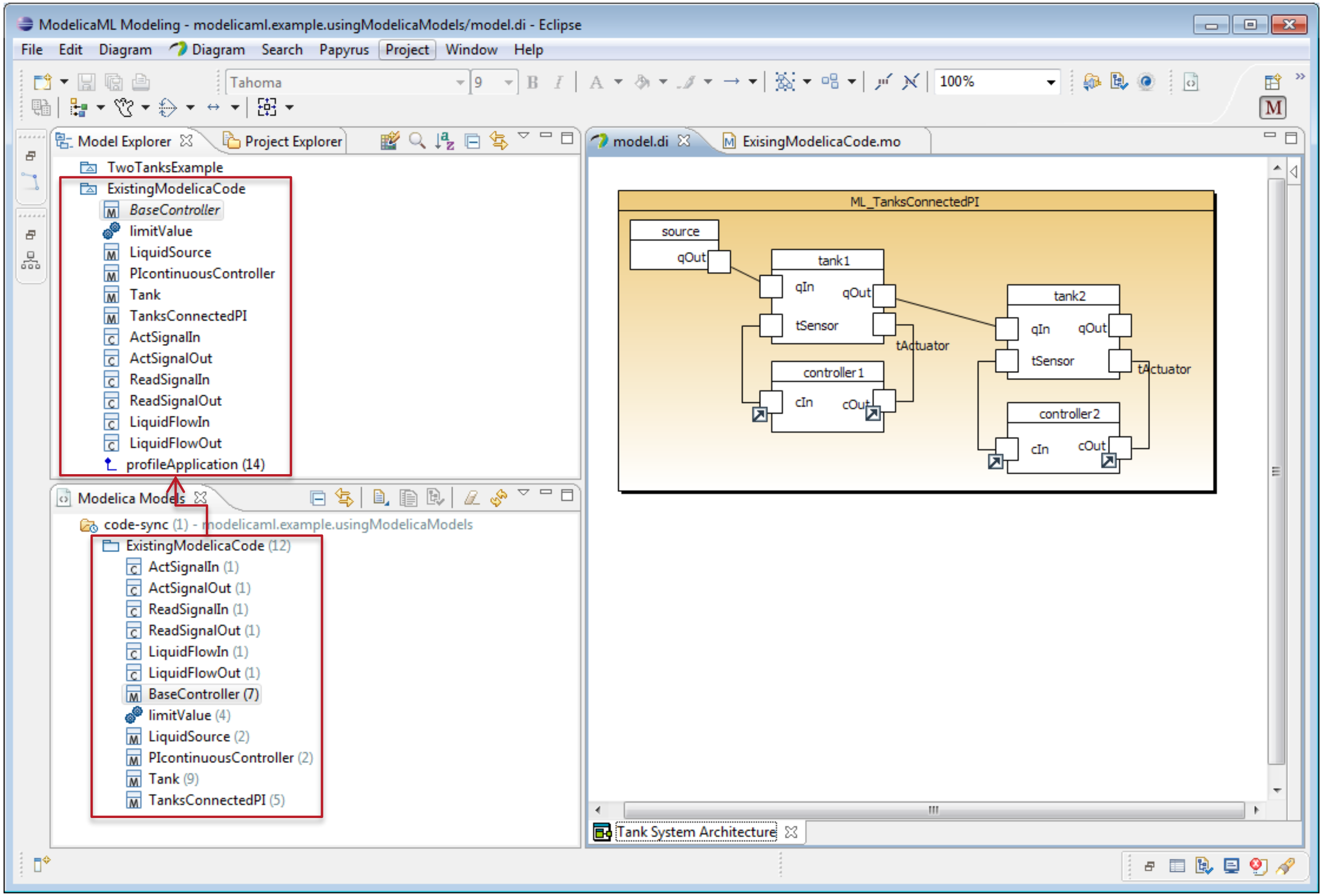


```
modelicaml.example.usingModelicaModels
  code-gen
  code-sync
    ExisingModelicaCode.mo
  verification-gen
  model.di
  model.notation
  model.uml
```

```
code-sync (1) - modelicaml.example.usingModelicaModels
  ExistingModelicaCode (12)
    ActSignalIn (1)
    ActSignalOut (1)
    ReadSignalIn (1)
    ReadSignalOut (1)
    LiquidFlowIn (1)
    LiquidFlowOut (1)
    BaseController (7)
      K : Real
      T : Real
      cIn : ExistingModelicaCode.ReadSignalIn
      cOut : ExistingModelicaCode.ActSignalOut
      ref : Real
      error : Real
      outCtr : Real
    limitValue (4)
    LiquidSource (2)
    PIcontinuousController (2)
    Tank (9)
    TanksConnectedPI (5)
```

```

partial model BaseController
  parameter Real K = 2 "Gain";
  parameter Real T(unit = "s") = 10 "Time constant";
  ReadSignalIn cIn "Input sensor level, connector";
  ActSignalOut  cOut "Control to actuator, connector";
  parameter Real ref "Reference level";
  Real error "Deviation from reference               level";
  Real outCtr "Output control signal";
equation
  error = ref - cIn.val;
  cOut.act = outCtr;
end BaseController;

function limitValue
  input  Real pMin;
  input  Real pMax;
  input  Real p;
  output Real pLim;
 algorithm
  pLim := if p>pMax then pMax
          else if p<pMin then pMin
          else p;
end limitValue;

model LiquidSource
  LiquidFlowOut qOut;
  //parameter Real flowLevel = 0.02;
  input Real flowLevel = if time > 150 then 3*0.02 else 0.02;

equation
  //qOut.lflow = if time > 150 then 3*flowLevel else flowLevel;
  qOut.lflow = flowLevel;
end LiquidSource;

```

Live Demo



ModelicaML Modeling - modelicaml.example.usingModelicaModels/model.di - Eclipse

Model Explorer:

- TwoTanksExample
- ExistingModelicaCode
  - BaseController
  - limitValue
  - LiquidSource
  - PIcontinuousController
  - Tank
  - TanksConnectedPI
  - ActSignalIn
  - ActSignalOut
  - ReadSignalIn
  - ReadSignalOut
  - LiquidFlowIn
  - LiquidFlowOut
  - profileApplication (14)

Modelica Models:

- code-sync (1) - modelicaml.example.usingModelicaModels
  - ExistingModelicaCode (12)
    - ActSignalIn (1)
    - ActSignalOut (1)
    - ReadSignalIn (1)
    - ReadSignalOut (1)
    - LiquidFlowIn (1)
    - LiquidFlowOut (1)
    - BaseController (7)
    - limitValue (4)
    - LiquidSource (2)
    - PIcontinuousController (2)
    - Tank (9)
    - TanksConnectedPI (5)

OMC API Enhancements

- Queering of Modelica models using OMC CORBA API

- getImportCount(M1), getNthImport(M1, 1)
- getInitialAlgorithmCount(M1) , getNthInitialAlgorithm(M1, 1)
- getAlgorithmCount(M1), getNthAlgorithm(M1, 1)
- getInitialEquationCount(M1), getNthInitialEquation(M1, 1)
- getEquationCount(M1) , getNthEquation(M1, 1)
- getNthComponentCondition(M1, 1)
- isEnumeration(M1)
- getEnumerationLiterals(M1)
- isReplaceable(M1, “C1”)
- getAnnotationCount(M1), getNthAnnotationString(M1, 1)

- In progress: consrainedBy and partial derivative function relations

Thank you for your attention!

Wladimir Schamai

EADS Innovation Works (Germany)

wladimir.schamai@eads.net