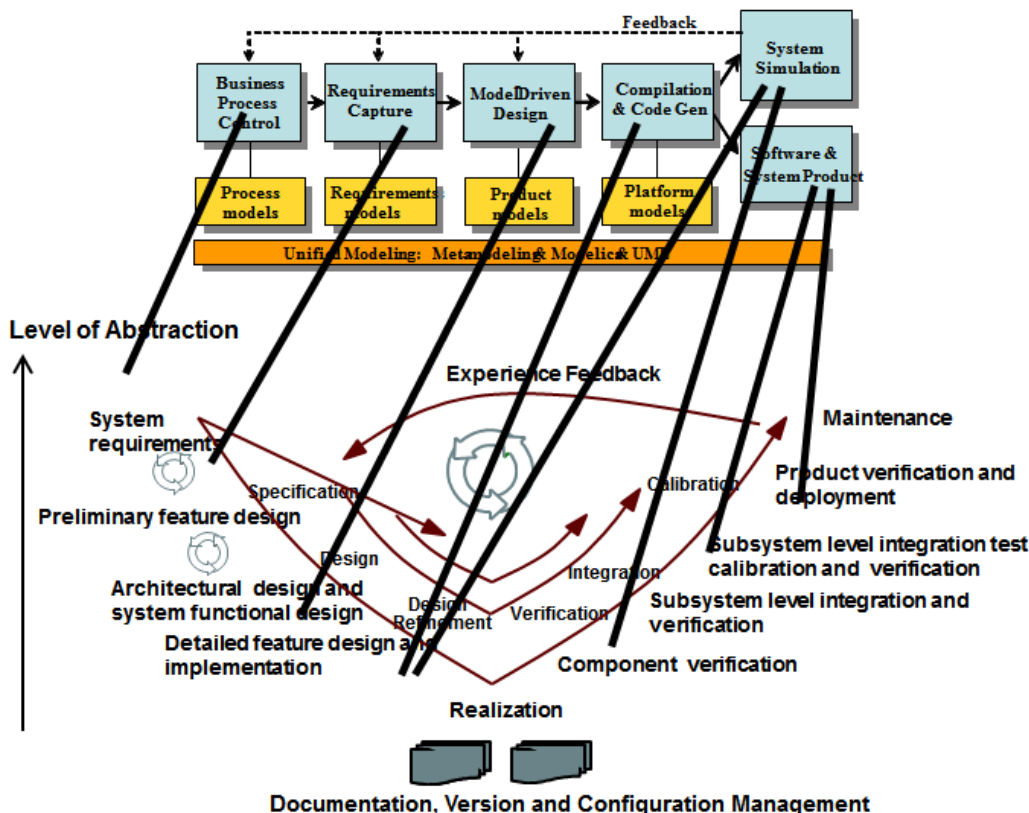


Integrated Modeling of CPS including Requirements: Open Source MBSE Tools Based on Modelica and UML



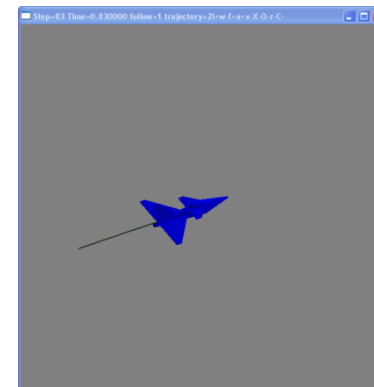
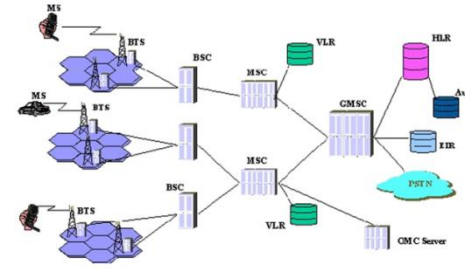
**LCCC MBSE Workshop, Lund
May 4, 2015**

Peter Fritzson
peter.fritzson@liu.se

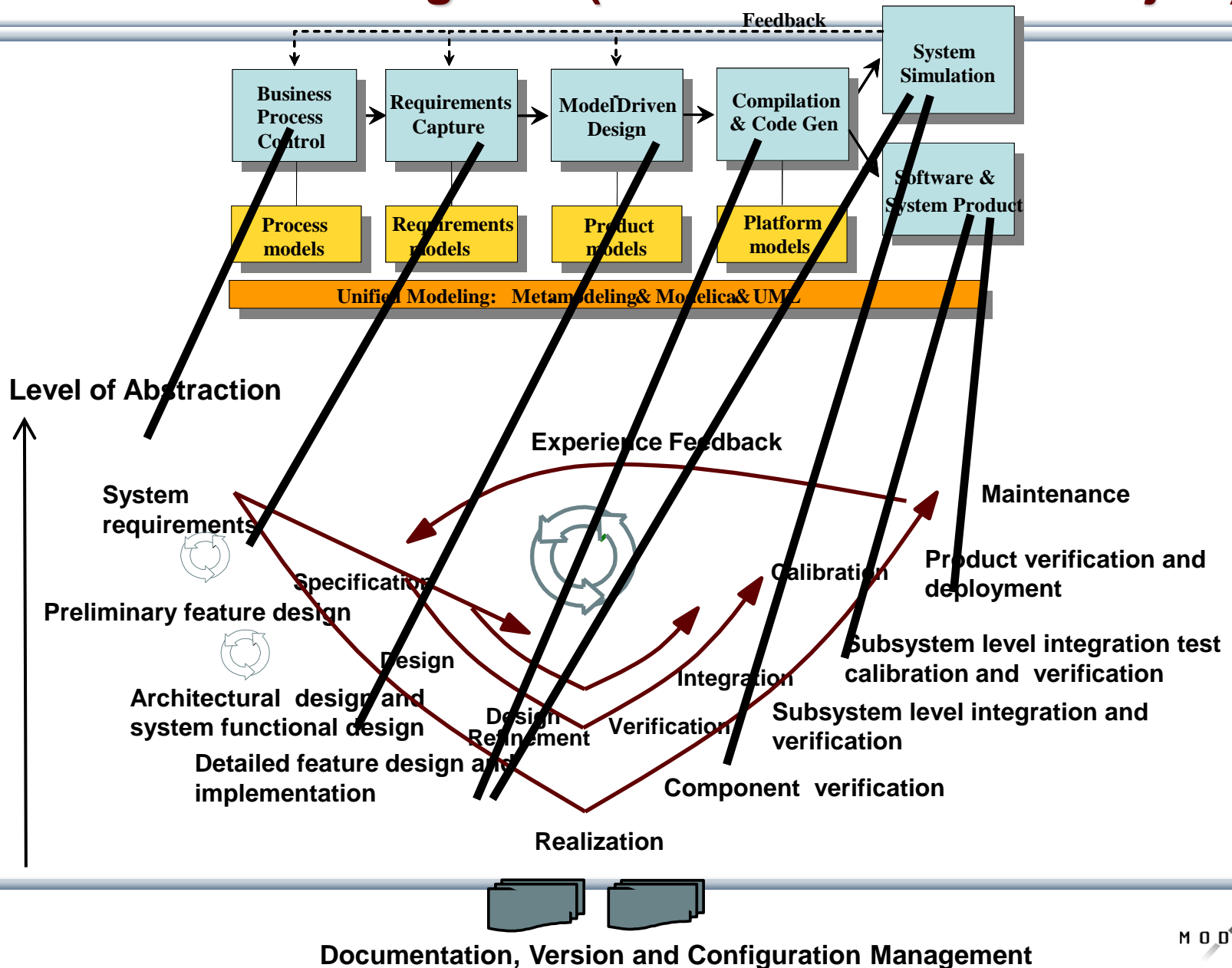
Vice Chairman of Modelica Association
 Director of Open Source Modelica Consortium
 Professor at Linköping University

Industrial Challenges for Complex Cyber-Physical System Products of both Software and Hardware

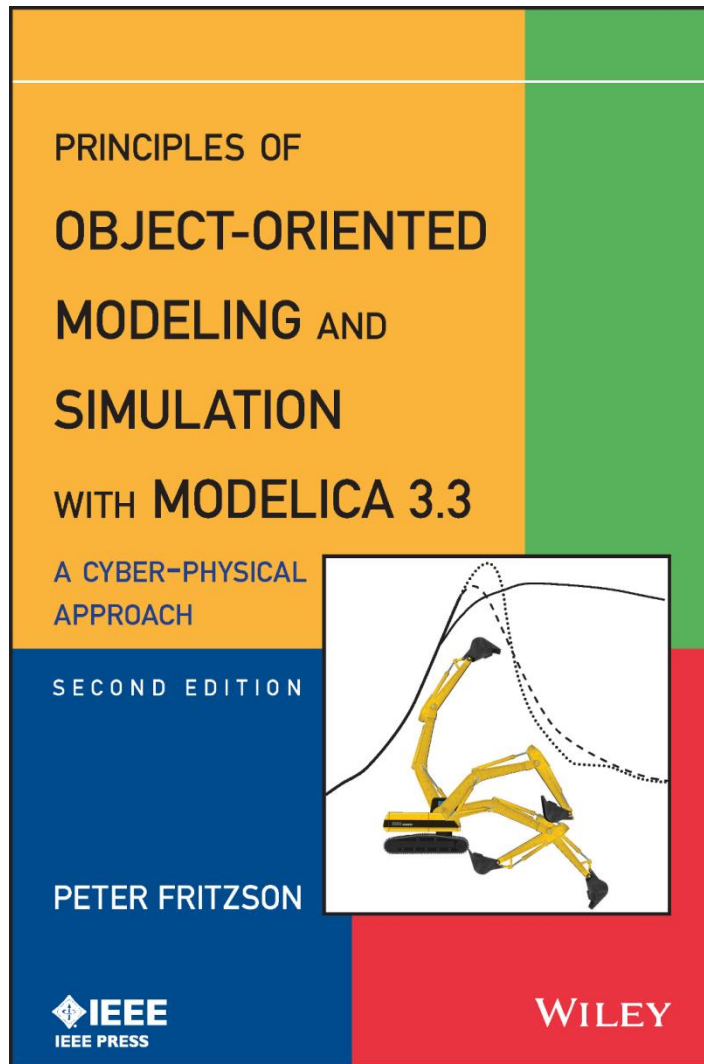
- Increased **Software** Fraction
- **Shorter** Time-to-Market
- Higher demands on effective strategic **decision** making
- **Cyber-Physical** (CPS) – Cyber (software)
Physical (hardware) products



Open Source Model-Based Development Environment Covers Product-Design V – (OPENPROD ITEA2 Project)



New Big Modelica Book, 2014 (Warning! Commercial)



Peter Fritzson Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach

Can be ordered from Wiley or Amazon
Wiley-IEEE Press, 2014, 1250 pages

- OpenModelica
 - www.openmodelica.org
- Modelica Association
 - www.modelica.org

Overview of this Talk

- Part I – Introduction to the OpenModelica Open Source MBSE Environment
- Part II – Dynamic debugging of equation-based models
- Part III – Dynamic verification/testing of formalized requirements vs Models in MBSE

Part I

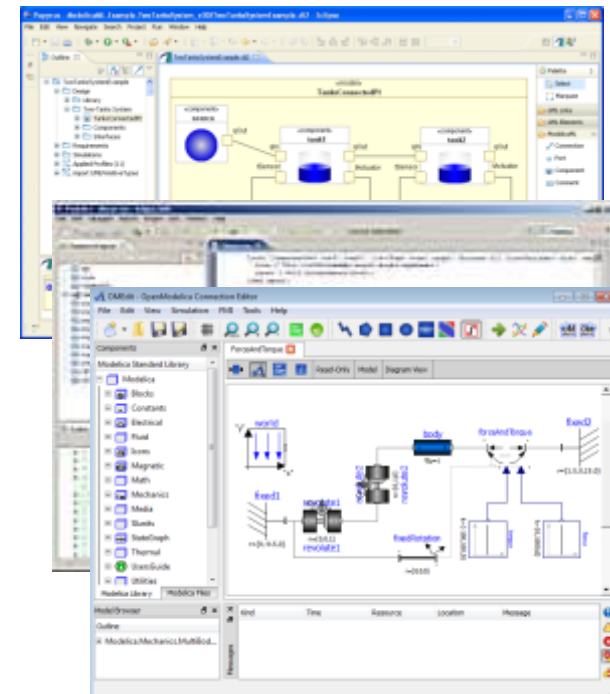
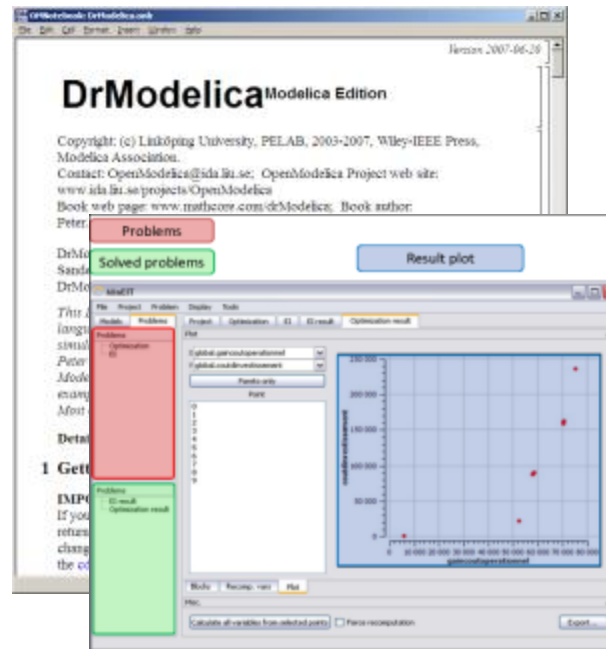
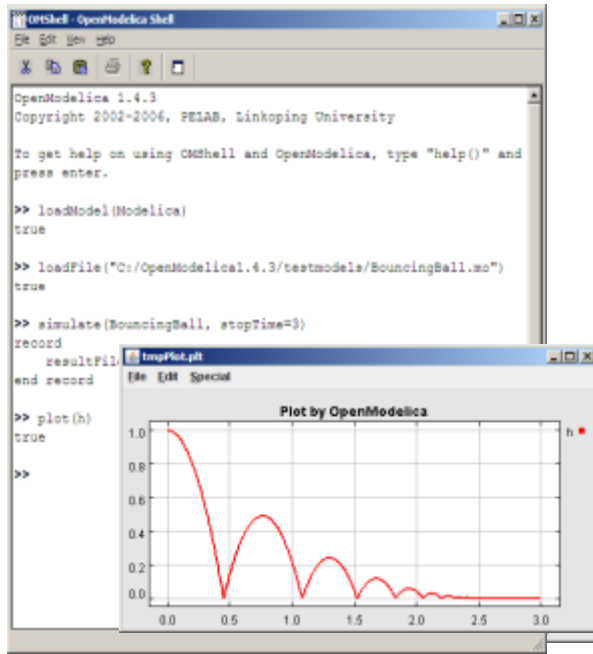
Introduction to the OpenModelica Environment



The OpenModelica Open Source Environment

www.openmodelica.org

- Advanced Interactive Modelica compiler (OMC)
 - Supports most of the Modelica Language
 - **Modelica and Python scripting**
- Basic environment for creating models
 - **OMShell** – an interactive command handler
 - **OMNotebook** – a literate programming notebook
 - **MDT** – an advanced textual environment in Eclipse
- **OMEdit** graphic Editor
- **OMDebugger** for equations
- **OMOptim** optimization tool
- **OM Dynamic optimizer** collocation
- **ModelicaML** UML Profile
- **MetaModelica** extension
- **ParModelica** extension



OSMC – International Consortium for Open Source Model-based Development Tools, 43 Members

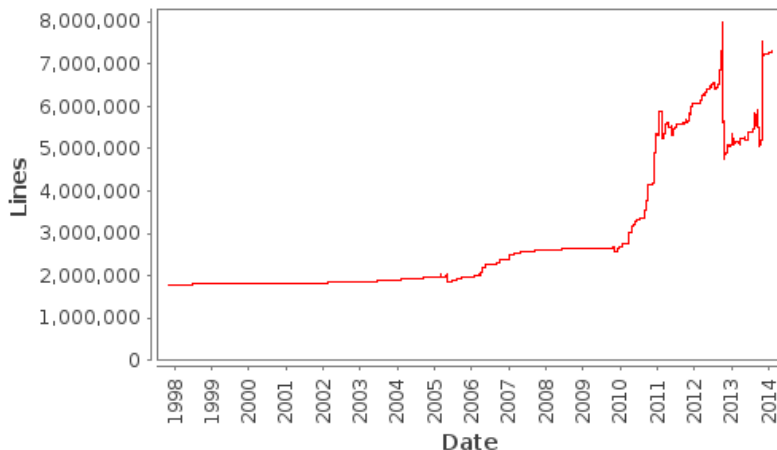
Founded Dec 4, 2007

Open-source community services

- Website and Support Forum
- Version-controlled source base
- Bug database
- Development courses
- www.openmodelica.org

Code Statistics

/trunk: Lines of Code



Industrial members

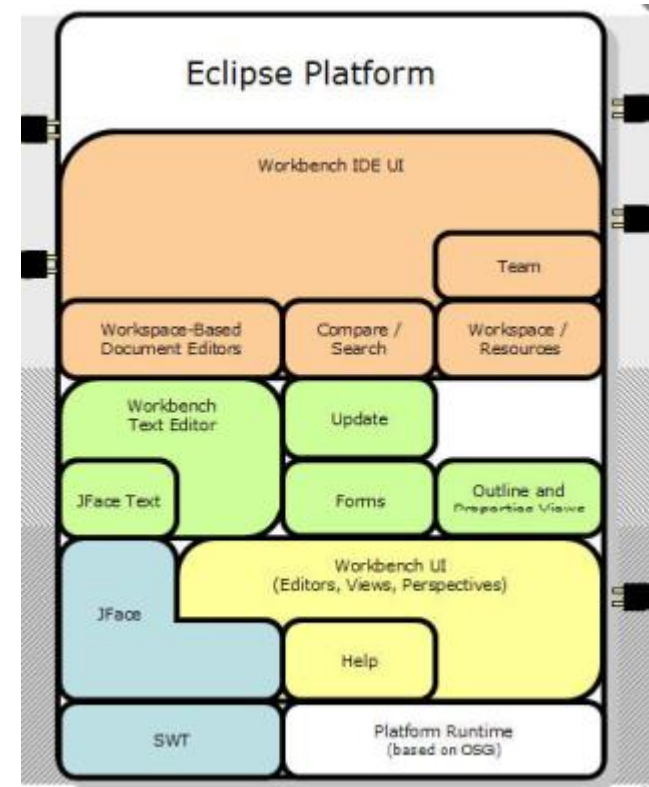
- ABB AB, Sweden
- Bosch Rexroth AG, Germany
- Siemens Turbo, Sweden
- CDAC Centre, Kerala, India
- Creative Connections, Prague
- DHI, Aarhus, Denmark
- EDF, Paris, France
- Equa Simulation AB, Sweden
- Fraunhofer IWES, Bremerhaven
- IFP, Paris, France
- ISID Dentsu, Tokyo, Japan
- ITI, Dresden, Germany
- Maplesoft, Canada
- Ricardo Inc., USA
- RTE, France
- TLK Thermo, Germany
- Sozhou Tongyuan, China
- VTI, Linköping, Sweden
- VTT, Finland
- Wolfram MathCore, Sweden

University members

- Austrian Inst. of Tech, Austria
- UC Berkeley, USA
- TU Berlin, Insti UEBB, Germany
- FH Bielefeld, Bielefeld, Germany
- TU Braunschweig, Germany
- Univ Calabria, Italy
- TU Dortmund, Germany
- TU Dresden, Germany
- Université Laval, Canada
- Ghent University, Belgium
- Halmstad University, Sweden
- Heidelberg University, Germany
- TU Hamburg/Harburg Germany
- Linköping University, Sweden
- KTH, Stockholm, Sweden
- Univ of Maryland, Syst Eng USA
- Univ of Maryland, CEEE, USA
- Politecnico di Milano, Italy
- Ecoles des Mines, CEP, France
- Mälardalen University, Sweden
- Univ Pisa, Italy
- Univ Stellenbosch, South Africa
- Telemark Univ College, Norway

OpenModelica MDT – Eclipse Plugin

- Browsing of packages, classes, functions
- Automatic building of executables; separate compilation
- Syntax highlighting
- Code completion, Code query support for developers
- Automatic Indentation
- Debugger



OpenModelica Eclipse MDT: Code Outline and Hovering Info

The screenshot displays the Eclipse IDE with the OpenModelica MDT. The main editor shows the following code snippet from 'Absyn.mo':

```
case (MATRIX(matrix = exp1))
  local list<list<list<ComponentRef>>> res1;
  equation
    res1 = Util.listListMap(exp1, getCrefFromExp);
    res2 = Util.listFlatten(res1);
    res = Util.listFlatten(res2);
  then
    res;
case (RANGE(start = e1, step = SOME(e3), stop = e2))
  equation
    l1 = getCrefFromExp(e1);
    l2 =
      function getCrefFromExp "function: getCrefFromExp
        Returns a flattened list of the
        component references in an expression"
        input Exp inExp;
        then
          output list<ComponentRef> outComponentRefList;
          algorithm
            outComponentRefList:=matchcontinue inExp
            local
              l1 =
                ComponentRef cr;
```

The tooltip for the `getCrefFromExp` function provides the following information:

- Function name: `function: getCrefFromExp`
- Description: Returns a flattened list of the component references in an expression
- Input: `input Exp inExp;`
- Output: `output list<ComponentRef> outComponentRefList;`
- Algorithm: `algorithm outComponentRefList:=matchcontinue inExp`
- Local variables: `local l1 = ComponentRef cr;`

The left sidebar shows the project tree and the code outline for 'Absyn', listing various algorithm items such as `ADD`, `ALG_ASSIGN`, `ALG_BREAK`, etc.

The bottom panel shows a list of errors, including:

- The identifier at start and end are different
- The identifier at start and end are different
- The identifier at start and end are different, pa...

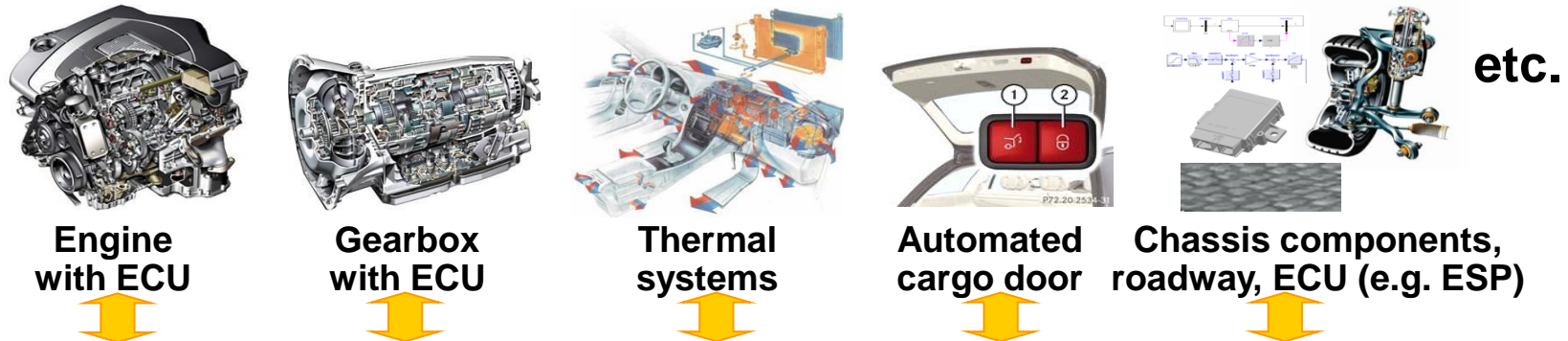
64M of 254M

Ctrl Contrib (Bottom)

Identifier Info on Hovering

Code Outline for easy navigation within Modelica files

General Tool Interoperability & Model Exchange Functional Mock-up Interface (FMI)



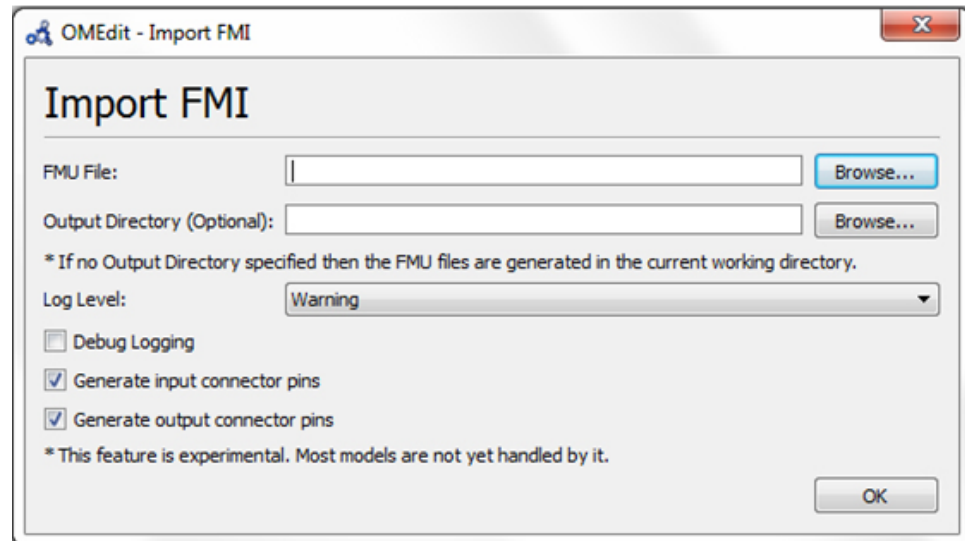
functional mockup interface for model exchange and tool coupling

courtesy Daimler

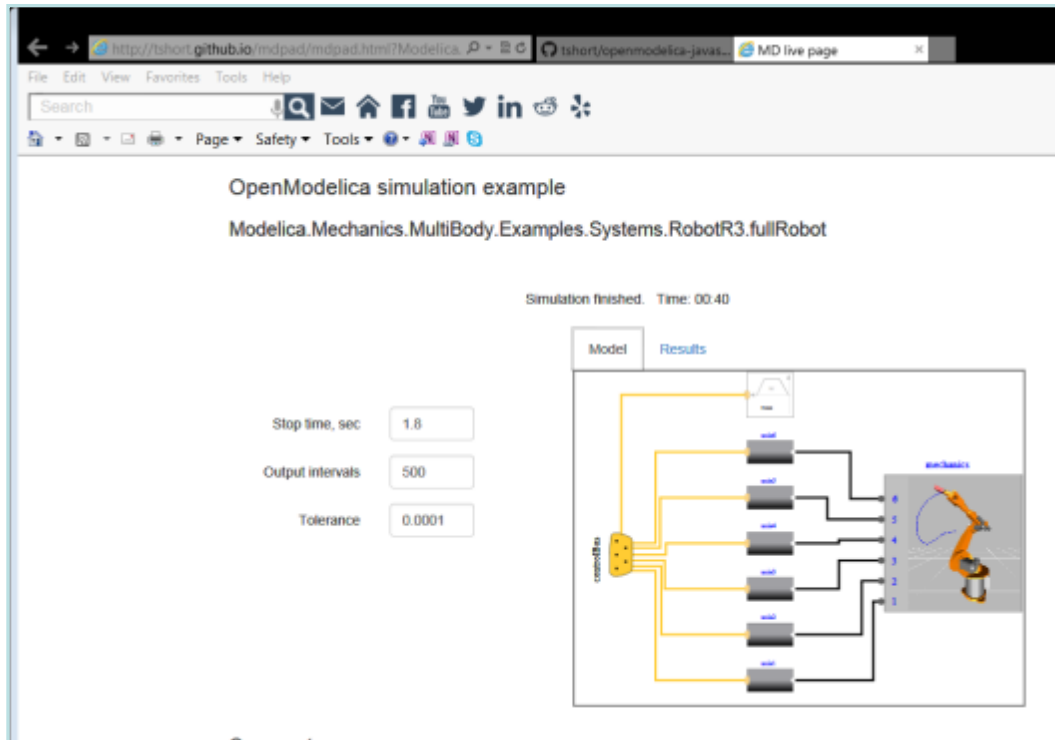
- FMI development was started by ITEA2 MODELISAR project. FMI is a Modelica Association Project now
- **Version 1.0**
- FMI for Model Exchange (released Jan 26,2010)
- FMI for Co-Simulation (released Oct 12,2010)
- **Version 2.0**
- FMI for Model Exchange and Co-Simulation (released July 25,2014)
- **> 50 tools** supporting it (<https://www.fmi-standard.org/tools>)

FMI in OpenModelica

- FMI Model Exchange implemented (FMI 1.0 and FMI 2.0)
- A prototype of FMI 2.0 co-simulation is available
- Ongoing work to support full FMI 2.0 co-simulation
- The FMI interface is accessible via the **OpenModelica scripting environment** and the **OpenModelica connection editor**



OpenModelica Simulation in Web Browser Client



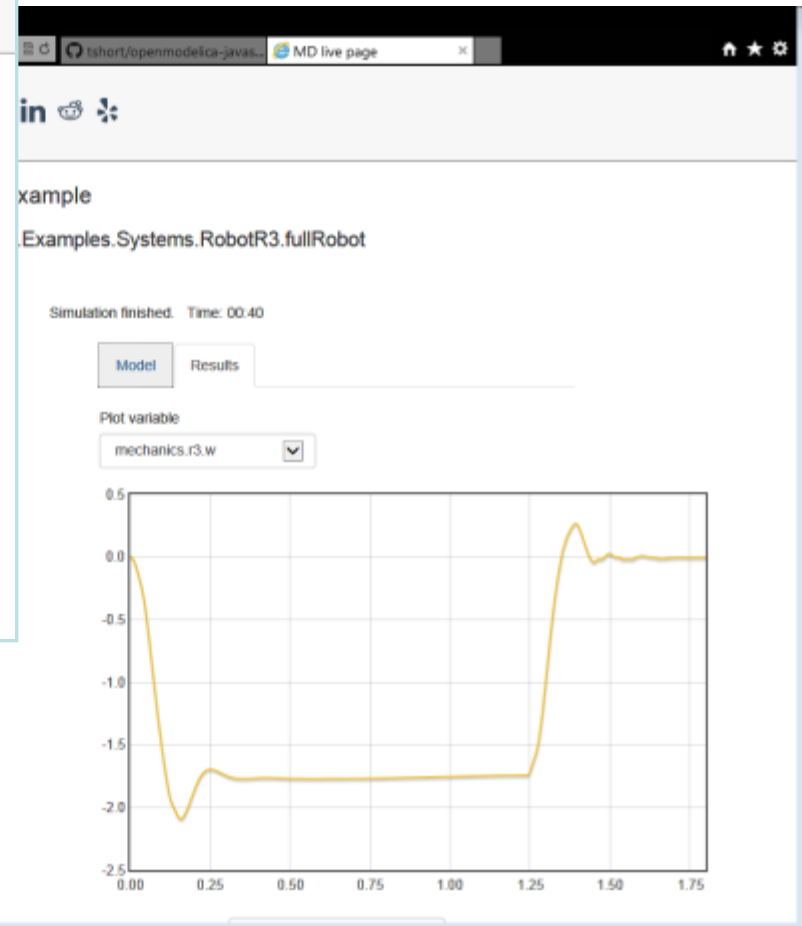
OpenModelica simulation example
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot

Simulation finished. Time: 00:40

Stop time, sec: 1.8
Output intervals: 500
Tolerance: 0.0001

Model Results

MultiBody RobotR3.FullRobot



OpenModelica simulation example
Modelica.Mechanics.MultiBody.Examples.Systems.RobotR3.fullRobot

Simulation finished. Time: 00:40

Model Results

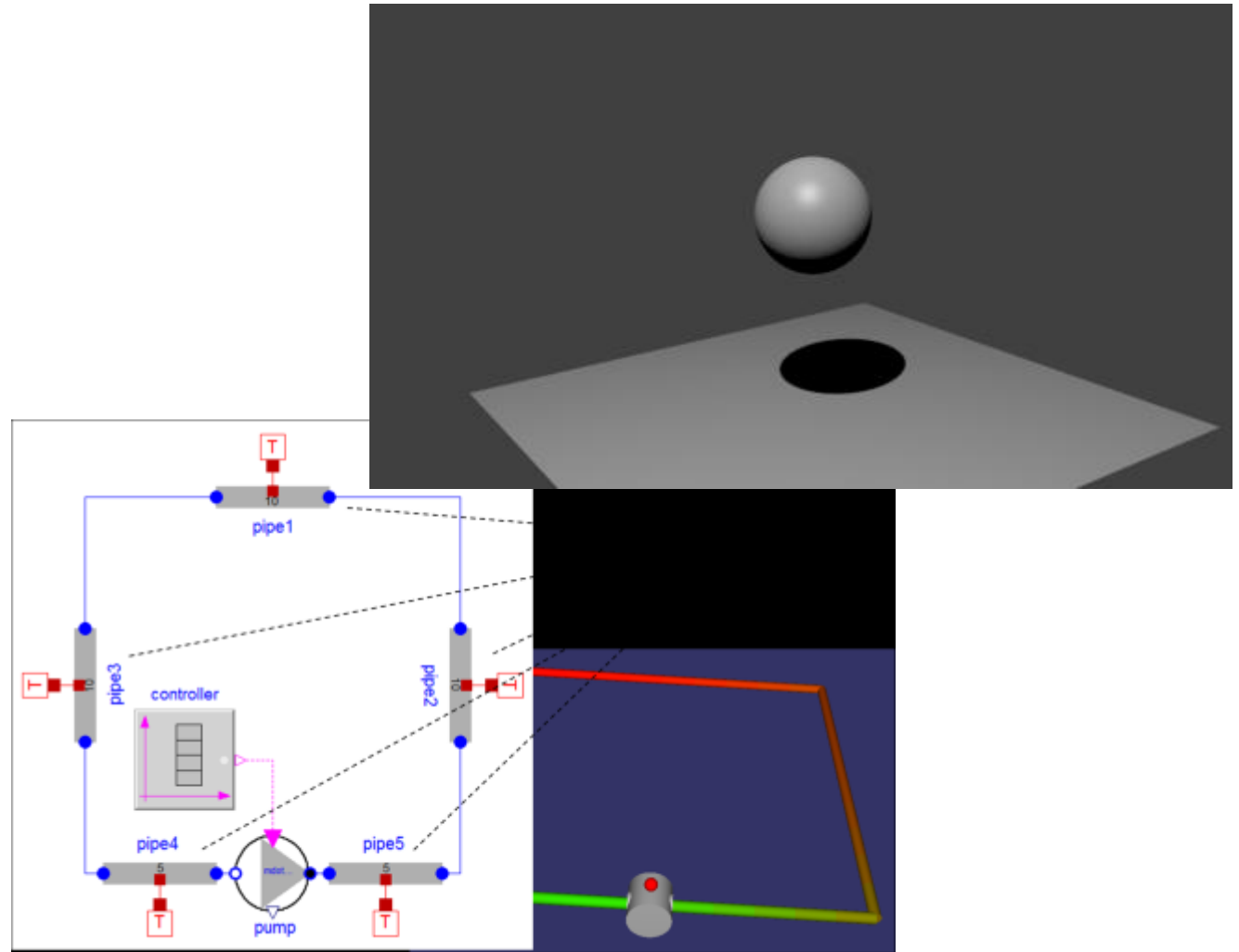
Plot variable: mechanics.r3.w

Time (s)	Value
0.00	0.00
0.25	-2.10
0.50	-1.80
1.00	-1.80
1.25	-1.80
1.40	0.30
1.50	0.00
1.75	0.00

OpenModelica compiles to efficient Java Script code which is executed in web browser

Modelica3D Library with OpenModelica

- Modelica 3D Graphics Library by Fraunhofer FIRST, Berlin
- Part of OpenModelica distribution
- Can be used for 3D graphics in OpenModelica



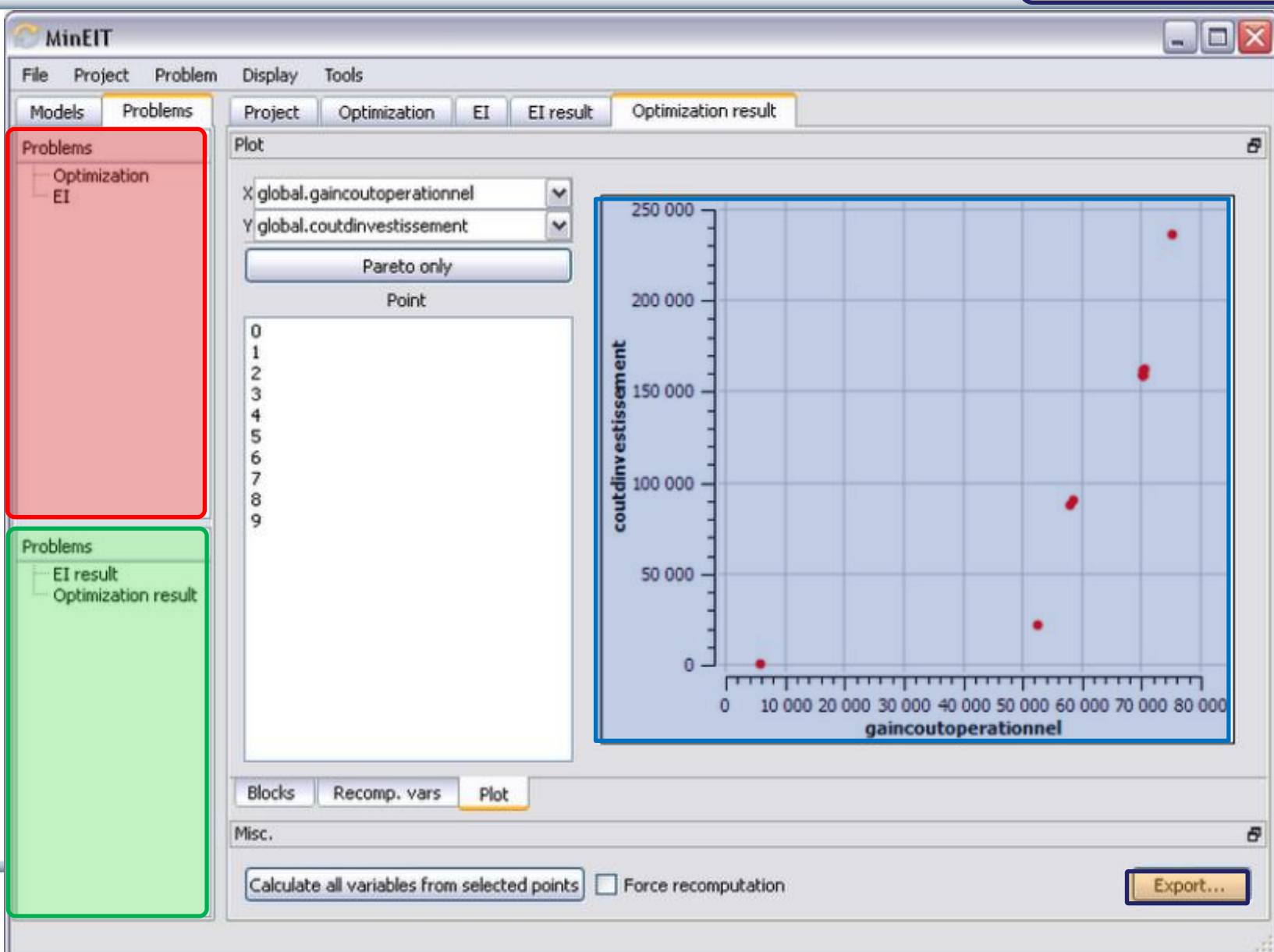
Problems

OMOptim – Parameter Sweep Design Optimization

Solved problems

Result plot

Export result data .csv



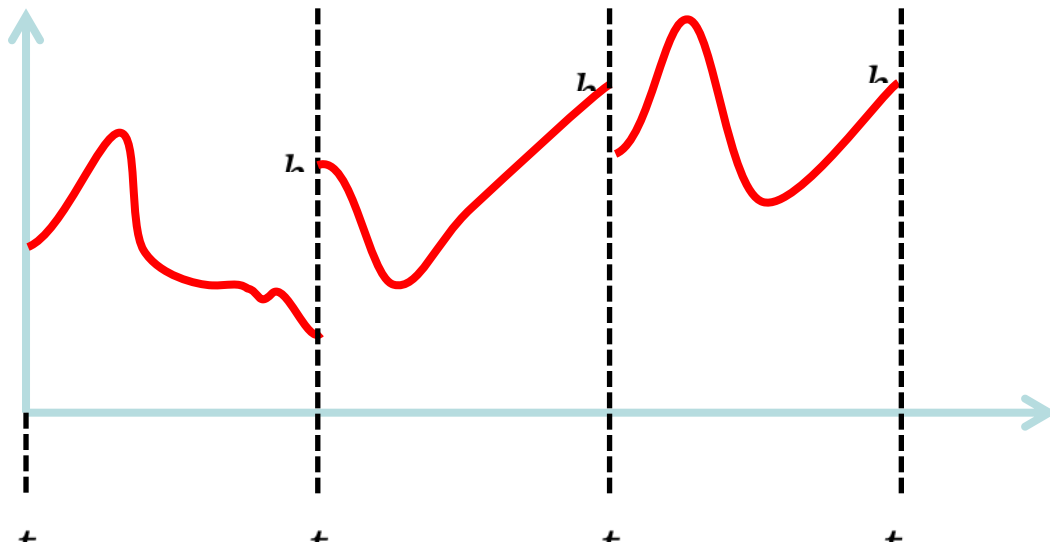
Here
Pareto
front
optimiza-
tion

Optimization of Dynamic Trajectories Using Multiple-Shooting and Collocation

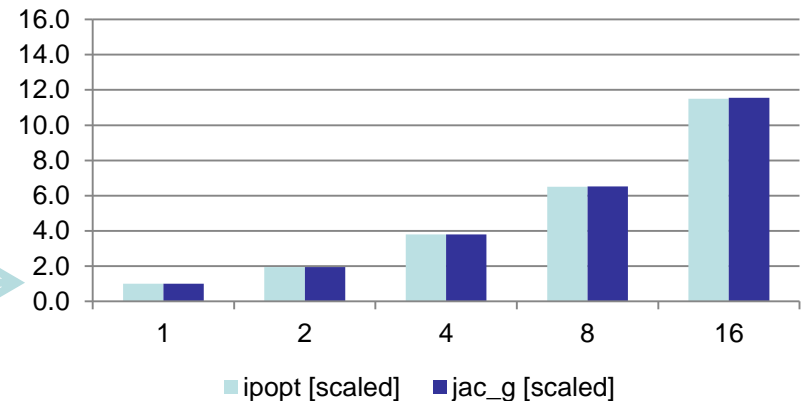
- Minimize a goal function subject to model equation constraints, useful e.g. for NMPC
- Multiple Shooting/Collocation
 - Solve sub-problem in each sub-interval

In OpenModelica 1.9.1
beta release Jan 2014.

$$x_i(t_{i+1}) = h_i + \int_{t_i}^{t_{i+1}} f(x_i(t), u(t), t) dt \approx F(t_i, t_{i+1}, h_i, u_i), \quad x_i(t_i) = h_i$$



Example speedup, 16 cores:
MULTIPLE_COLLOCATION



OMnotebook Interactive Electronic Notebook Here Used for Teaching Control Theory

1 Kalman Filter

Often we don't have access to the internal states of a system. We have to reconstruct the state of the system based on measurements. The idea with an observer is that we feedback the error. If the estimation is correct then the difference should be zero.

Another difficulty is that the measured quantities of

$$\begin{Bmatrix} \hat{x} \\ \hat{y} \end{Bmatrix}$$

Here e denoting a disturbance in the input signal. The error is evaluated by the difference

$$K(y(t) - \hat{y}(t))$$

By using this quantity as feedback we obtain the observer

$$\dot{\hat{x}} = A\hat{x}(t) + Bu(t) + K(y(t) - \hat{y}(t))$$

Now form the error as

The differential error is

The screenshot shows the OMNotebook interface with a code editor and a plot window. The code defines a Kalman filter model and simulates it over 3 seconds. The plot compares the state of the system with and without the Kalman filter.

```

model KalmanFeedback
  parameter Real A[:,size(A, 1)] = {{0,1},{1,0}};
  parameter Real B[size(A, 1),:] = {{0},{1}};
  parameter Real C[:,size(A, 1)] = {{1,0}};
  parameter Real[2,1] K = [2.4;3.4];
  parameter Real[1,2] L = [2.4,3.4];
  parameter Real[:,:] ABL = A-B*L;
  parameter Real[:,:] BL = B*L;
  parameter Real[:,:] Z = zeros(size(ABL,2),size(ACL,1));
  parameter Real[:,:] ACL = A-K*C;
  parameter Real[:,:] Anew = [0,1,0,0; -1.4, -3.4, 2.4,3.4; 0,0,-2.4,1;0,0,-2.4,0];
  parameter Real[:,:] Bnew = [0;1;0;0];
  parameter Real[:,:] Fnew = [1;0;0;0];
  stateSpaceNoise Kalman(stateSpace.A=Anew,stateSpace.B=Bnew, stateSpace.C=[1,0,0,0],
stateSpace.F = Fnew);
  stateSpaceNoise noKalman;
end KalmanFeedback;

simulate(KalmanFeedback,stopTime=3)

plot({Kalman.stateSpace.y[1],noKalman.stateSpace.y[1]})
  
```

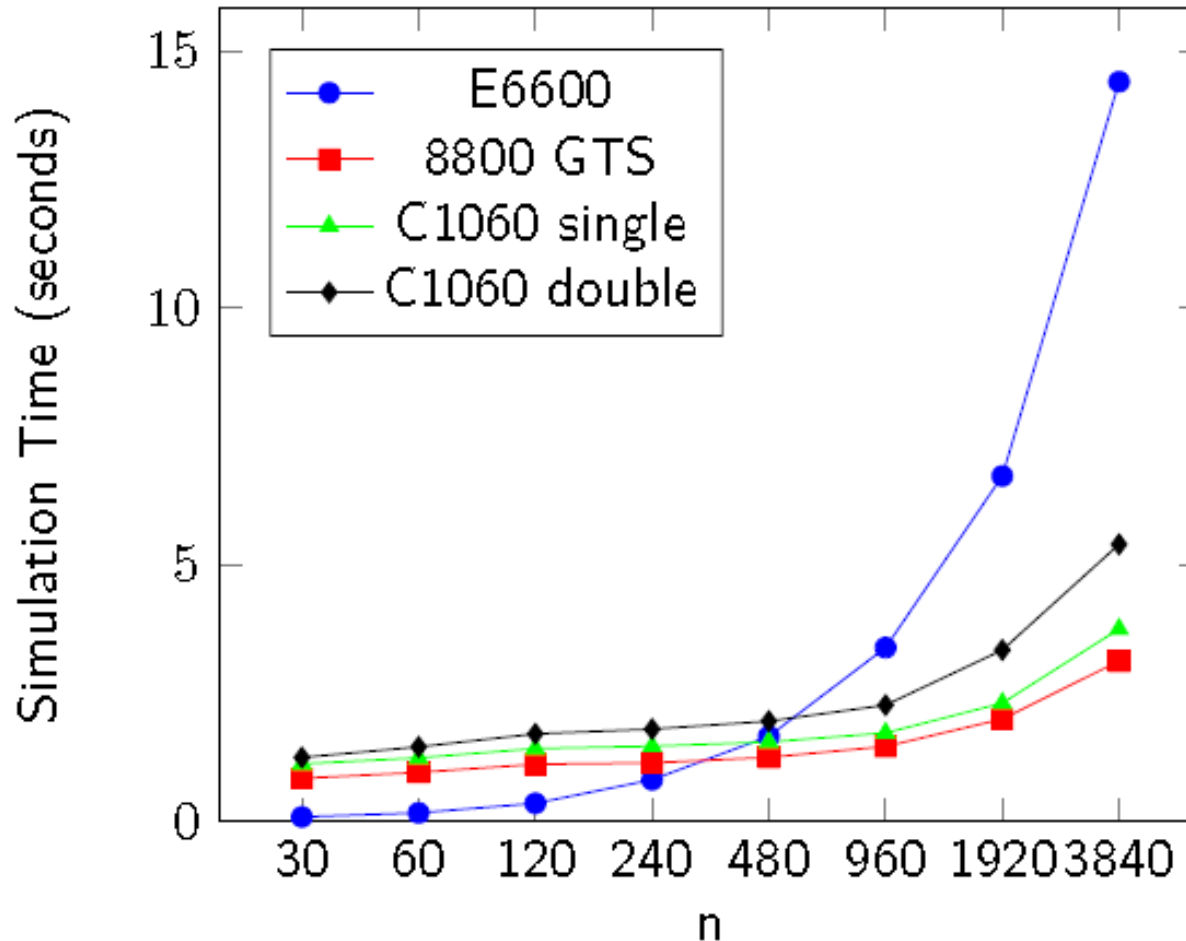
The plot, titled "Plot by OpenModelica", shows two data series over time from 0 to 3 seconds. The y-axis ranges from 0 to 15. The red series, labeled "Kalman.stateSpace.y[1]", remains near zero with small fluctuations. The blue series, labeled "noKalman.stateSpace.y[1]", starts near zero and increases exponentially to approximately 18 at 3 seconds.

MetaModelica Language Extension for Model Transformations and Advanced Applications

- **Large-scale** existing application – OpenModelica compiler written in MetaModelica, compiling itself
- MetaModelica language extension
 - single assignment equations (with opt. patterns)
 - **tree** data structures, garbage collection
 - **pattern** equations
 - **matching**, backtracking
 - Very **efficient portable** implementation (compiles to C)
- Now ongoing **standardization** in Modelica Association targeting Modelica 3.4

Faster Simulation – Compiling Modelica to Multi-Core

Speedup on NVIDIA, Modelica Model, Generated Code, n Problem Size



Part II

Equation-Based Model Dynamic Debugging

Need for Debugging Tools

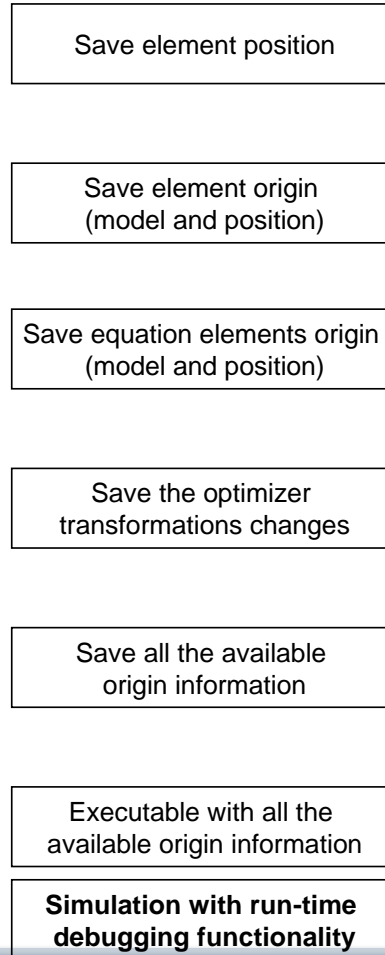
Map Low vs High Abstraction Level

- A **major part** of the total **cost** of software projects is due to testing and debugging
- US-Study 2002:
Software errors cost the US economy **annually~ 60 Billion \$**
- **Problem: Large Gap in Abstraction Level**
from **Equations** to **Executable Code**
- Example error message (hard to understand)
Error solving nonlinear system 132
time = 0.002
residual[0] = 0.288956
x[0] = 1.105149
residual[1] = 17.000400
x[1] = 1.248448
...

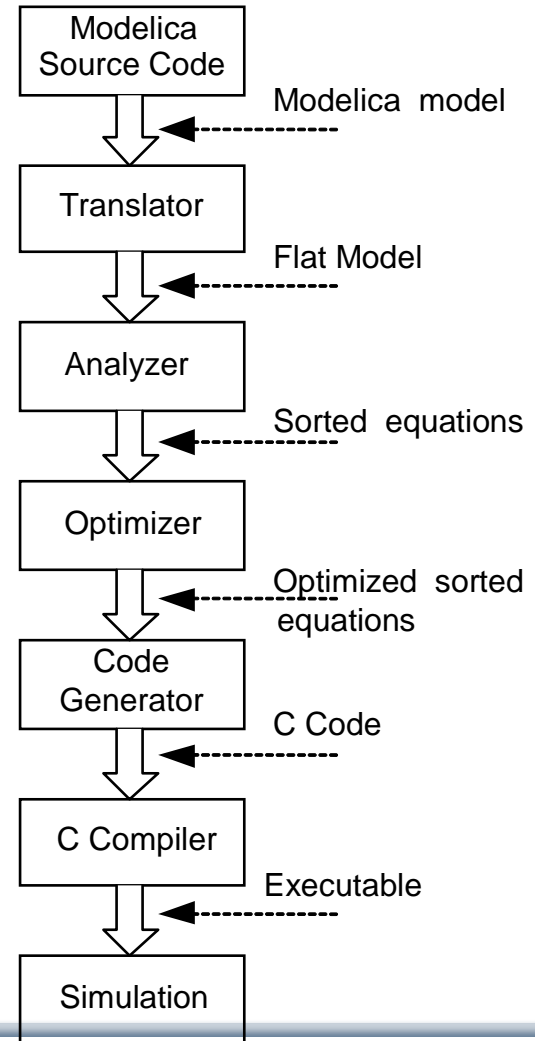
Model Compiler Translation Phases Extended with Debugging

- Include debugging support within the translation process

Debugging Translation Process Additional Steps



Normal Translation Process



Example Symbolic Transformations with Compiler Debug Trace

- **Complicated to understand source of some errors**
- **Efficient trace of transformations – low overhead**

Example: $0 = y + \text{der}(x * \text{time} * z); \quad z = 1.0;$

(1) substitution:

$$y + \text{der}(x * (\text{time} * z))$$

\Rightarrow

$$y + \text{der}(x * (\text{time} * 1.0))$$

(2) simplify:

$$y + \text{der}(x * (\text{time} * 1.0))$$

\Rightarrow

$$y + \text{der}(x * \text{time})$$

**(3) expand derivative
(symbolic diff):**

$$y + \text{der}(x * \text{time})$$

\Rightarrow

$$y + (x + \text{der}(x) * \text{time})$$

(4) solve:

$$0.0 = y + (x + \text{der}(x) * \text{time})$$

\Rightarrow

$$\text{der}(x) = ((-y) - x) / \text{time}$$

Properties of Transformation Trace

- Most equations have very **few** transformations on them
- Most of the interesting equations have a few
 - Still rather readable
- Some extra care to handle Modelica variable aliasing
- Very **efficient** implementation, max 1% overhead

MSL 3.1 MultiBody DoublePendulum

# Ops	Frequency	Comment
0	457	Parameters
1	89	Dummy eq & know var
2	720	Alias vars
3	479	Alias vars
4	124	Alias after simplify
5	25	Alias after simplify
6	99	Alias after simplify
7	55	Scalar eq
8	37	...
9	110	...
10	72	...
11	12	...
12	25	...
13	35	...
14	3	Known constant after many replacements
21	27	World object (3x3 matrix with many occurrences of aliased vars)

Integrated Static-Dynamic OpenModelica Equation Model Debugger

Efficient handling of Large Equation Systems

Showing equation transformations of a model:

The screenshot displays the OMEdit - Transformational Debugger interface, which is divided into three main panes: Variables View, Equations View, and Source View.

- Variables View:** Located at the top left, it shows a tree structure of variables (e.g., boxBody1, body, frame_a, R, T) and their properties. It includes a search bar and buttons for 'Expand All' and 'Collapse All'. Below the tree, there are sections for 'Defined In Equations' and 'Used In Equations', each with columns for Index, Type, and Equation. A 'Variable Operations' section is also present.
- Equations View:** Located at the bottom left, it shows a list of equations with columns for Index, Type, and Equation. Below this list, there are sections for 'Defines' and 'Depends', each with a list of variables. An 'Equation Operations' section is also present.
- Source View:** Located on the right, it shows the source code of the model. The code is color-coded and includes comments. A red box highlights a specific line of code (line 323) which is `frame_a.f = - Frames.resolve1(R_rel, frame_b.f);`. A black arrow points from this line to the corresponding equation in the Equations View.

Mapping dynamic run-time error to source model position

Example – Detecting Source of Chattering (excessive event switching) causing bad performance

The screenshot shows the OMEdit - Transformational Debugger interface. The main window displays the source code for a model named 'ChatteringEvents1'. The code is as follows:

```
1 within ;
2 package Debugging "Test
  cases for debugging of
  declarative models"
3
4 package Chattering "Models
  with chattering behaviour"
5 model ChatteringEvents1
6   "Exhibits chattering
  after t = 0.5, with
  generated events"
7   Real x(start=1,
  fixed=true);
8   Real y;
9   Real z;
10  equation
11  z = if x > 0 then -1
12  else 1;
13  y = 2*z;
14  der(x) = y;
15  annotation
16  (Documentation(info="<html>
17  <p>After t = 0.5, chattering
18  takes place, due to the
19  feedback loop in the right
20  hand side of the
21  equation.</p>
22  <p>Chattering can be
23  detected because lots of
24  tightly spaced events are
25  generated. The feedback to
26  the user should allow to
27  identify the equation from
28  which the zero crossing
29  function that generates the
30  events originates.</p>
31  </html>"),
32  experiment(StopTime=1));
33 end ChatteringEvents1;
34
35 model ChatteringEvents2
36   "Exhibits chattering
  after t = 0.422, with
  generated events"
```

The interface also shows the 'Variables' browser with the following table:

Variables	Comment	Line	Location
x		7	/hom...g
y		8	/hom...g
z		9	/hom...g

The 'Equations' browser shows the following table:

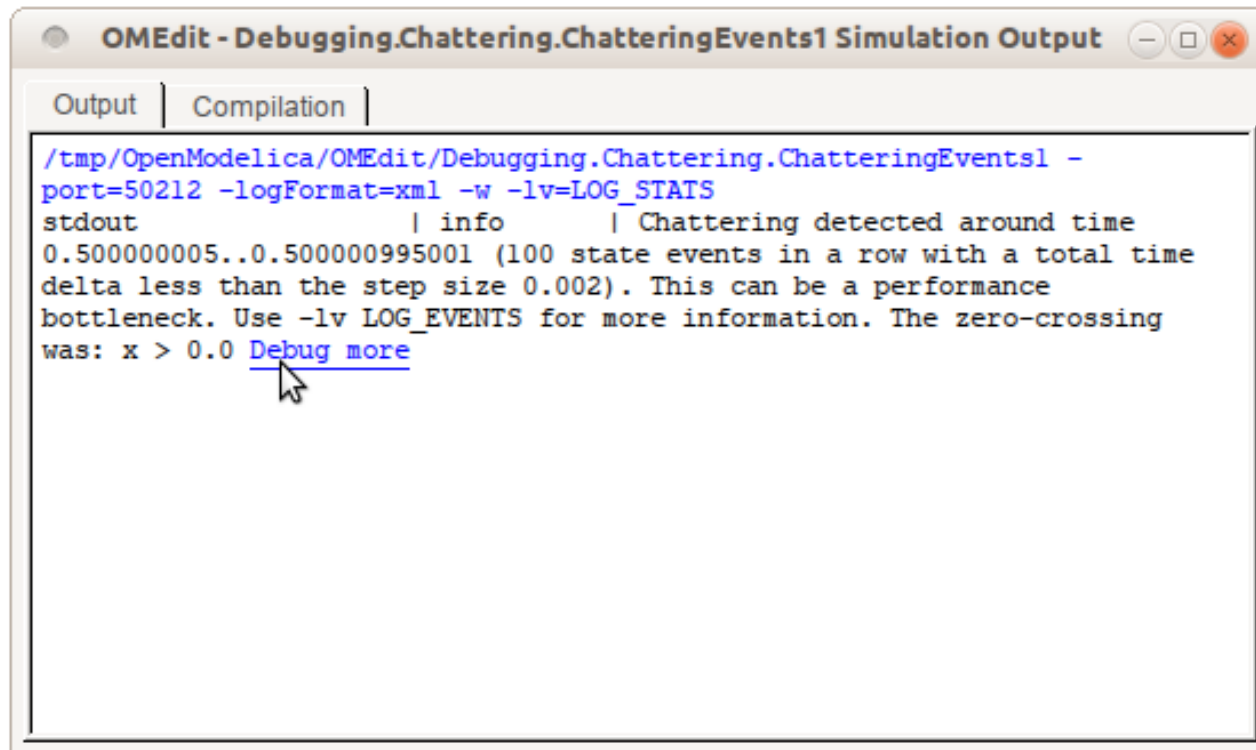
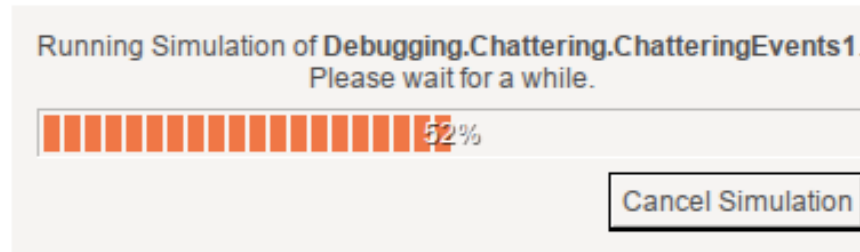
Inc	Type	Equation
-1	initial	(assignment) x = 1.0
-2	initial	(assignment) y = 2.0 * z
-3	initial	(assignment) der(x) = y
-4	initial	(assignment) y = 2.0 * z
-5	regular	(assignment) z = if x > 0 then -1 else 1.0
-6	regular	(assignment) y = 2.0 * z
-7	regular	(assignment) der(x) = y

The 'Equation Operations' section shows the following operations:

```
solved: z = if x > 0.0 then -1.0 else 1.0
original: z = if x > 0 then -1 else 1; => flattened: z = if x > 0.0 then -1.0 else 1.0;
```

An arrow points from the 'Equation Operations' section to the source code line 11: `z = if x > 0 then -1 else 1;`. A large text overlay on the right side of the image reads: **equation z = if x > 0 then -1 else 1; y = 2*z;**

Error Indication – Simulation Slows Down



Transformations Browser – EngineV6 Overview (11 116 equations in model)

Activities OMEdit Tue 12:06 sv Martin Sjöblund

OMEdit - Transformational Debugger
/tmp/OpenModelica_marsj/OMEdit/Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6_info.xml

Variables

Variables Browser

phi

Case Sensitive Regular Expression

Expand All Collapse All

Variables	Comment	Line	Location
phi	Exter...phi	6616	/usr/li...onal.mo
phi	Relat...ame_b	260	/usr/li...ints.mo
phi_offset	Relat...+ phi	242	/usr/li...ints.mo
Crank1	Absol...frame	11	/usr/li...mes.mo
body	Trans...frame	10	/usr/li...mes.mo
-phi	Dumm...body	805	/usr/li...arts.mo
-phi[1]	Dumm...body	805	/usr/li...arts.mo
-phi[2]	Dumm...body	805	/usr/li...arts.mo
-phi[3]	Dumm...body	805	/usr/li...arts.mo
-phi_d	= der(phi)	809	/usr/li...arts.mo
-phi_d[1]	= der(phi)	809	/usr/li...arts.mo
-phi_d[2]	= der(phi)	809	/usr/li...arts.mo

Defined in Equations

Index	Type	Equation
587	initial	(nonlinear)
5016	regular	(nonlinear)

Used in Equations

Inc	Type	Equation
...	regular	(assignment) cylinder...cos(cylinder3.B2.phi)
...	regular	(assignment) cylinder3... sin(cylinder3.B2.phi)
...	regular	(assignment) cylinder...sin(cylinder3.B2.phi)
...	regular	(assignment) cylinder...cos(cylinder3.B2.phi)
...	regular	(assignment) der(cyl...der3.Rod.body.w_a[1])
...	regular	(assignment) der(cyl...der3.Rod.body.w_a[1])
...	regular	(assignment) der(cyl...der3.Rod.body.w_a[1])

Variable Operations

Operations

Source Browser

```

386 Connections.branch(frame_a.R,
387 frame_b.R);
388 assert(cardinality(frame_a) > 0,
389 "Connector frame a of Revolute
390 joint is not connected");
391 assert(cardinality(frame_b) > 0,
392 "Connector frame b of Revolute
393 joint is not connected");
394
395 angle = phi_offset + phi;
396 w = der(phi);
397 a = der(w);
398
399 // relationships between quantities
400 of frame_a and of frame_b
401 frame_b.r_theta = frame_a.r_theta;
402
403 if rooted(frame_a.R) then
404   R_rel = Frames.planarRotation(e,
405 phi_offset + phi, w);
406   frame_b.R =
407 Frames.absoluteRotation(frame_a.R,
408 R_rel);
409   frame_a.f = -
410 Frames.resolve(R_rel, frame_b.f);
411   frame_a.t = -
412 Frames.resolve(R_rel, frame_b.t);
413 else
414   R_rel = Frames.planarRotation(-e,
415 phi_offset + phi, w);
416   frame_a.R =
417 Frames.absoluteRotation(frame_b.R,
418 R_rel);
419   frame_b.f = -
420 Frames.resolve(R_rel, frame_a.f);
421   frame_b.t = -
422 Frames.resolve(R_rel, frame_a.t);
423 end if;
424
425 // d'Alembert's principle
426 tau = -frame_b.t*e;
427
428 // Connection to internal

```

Equations

Equations Browser

Inc	Type	Equation
...	regular	(assignment) cylind...ylinder3.Cylinder.s
...	regular	(assignment) cylind...linder3.gasForce.L)
...	regular	(assignment) cylind...linder3.gasForce.x)
...	regular	(assignment) cylind...linder3.gasForce.V)
...	regular	(assignment) cylind...linder3.gasForce.L)
...	regular	(assignment) cylind...linder.s else 1e-06
...	regular	(assignment) cylind...k2.frame_b.R.T[2,3]
...	regular	(linear,r_rel_a = Fra...r_0 - frame_a.r_0);
...	regular	(linear,frame_b.r_0 =... * (s_offset + s);)
...	regular	(assignment) cylind...linder3.gasForce.x)
...	regular	(assignment) cylind...linder3.gasForce.p)
...	regular	(assignment) cylind...r3.gasForce.d ^ 2.0
...	regular	(assignment) cylind...linder3.gasForce.k)
...	regular	(assignment) cylind...ody.w_a[1] - load.w
...	regular	(assignment) der(c...r3.Rod.body.w_a[1])

Defines

Variable

der(cylinder3.B2.R_rel.T[3,3])

Depends

Variable

cylinder3.B2.phi
cylinder3.Rod.body.w_a[1]

Equation Operations

Operations

```

-solved: der(cylinder3.B2.R_rel.T[3,3]) = (-sin(cylinder3.B2.phi)) * cylinder3.Rod.body.w_a[1]
-substitute: (-sin(cylinder3.B2.phi)) * cylinder3.B2.w => (-sin(cylinder3.B2.phi)) * cylinder3.Rod.body.w_a[1]
-differentiate: dcos(cylinder3.B2.phi)/dtime = (-sin(cylinder3.B2.phi)) * der(cylinder3.B2.phi)
-differentiate: d(cylinder3.B2.R_rel.T[3,3])/dtime = der(cylinder3.B2.R_rel.T[3,3])
-scalarize(9): cylinder3.B2.R_rel.T = {{1.0, 0.0, 0.0}, [-0.0, c...B2.phi]] => cylinder3.B2.R_rel.T[3,3] = cos(cylinder3.B2.phi)
-simplify: cylinder3.B2.R_rel.T = {{1.0 * 1.0 + (1.0 - 1.0 * 1.0)...B2.phi}}, {0.0, -sin(cylinder3.B2.phi), cos(cylinder3.B2.phi)}}
-substitute: {{cylinder3.B2.e[1] * cylinder3.B2.e[1] + (1.0 - cy...2.phi), 0.0 * 0.0 + (1.0 - 0.0 * 0.0) * cos(cylinder3.B2.phi)}}
-inline: cylinder3.B2.R_rel = Modelica.Mechanics.MultiBody...[2] * cylinder3.B2.w, cylinder3.B2.e[3] * cylinder3.B2.w)
-original: R_rel = Frames.planarRotation(e, phi_offset + phi, w); => flattened:

```


Browsing Equation Transformation Chains

Closeup of EngineV6 Equations

Defines	Depends
Variable ▾	Variable ▾
der(cylinder3.B2.R_rel.T[3,3])	<ul style="list-style-type: none"> └ cylinder3.B2.phi └ cylinder3.Rod.body.w_a[1]
Equation Operations	
Operations	
<ul style="list-style-type: none"> └ solved: $\text{der}(\text{cylinder3.B2.R_rel.T}[3,3]) = (-\sin(\text{cylinder3.B2.phi})) * \text{cylinder3.Rod.body.w_a}[1]$ └ substitute: $(-\sin(\text{cylinder3.B2.phi})) * \text{cylinder3.B2.w} \Rightarrow (-\sin(\text{cylinder3.B2.phi})) * \text{cylinder3.Rod.body.w_a}[1]$ └ differentiate: $\text{dcos}(\text{cylinder3.B2.phi})/\text{dtime} = (-\sin(\text{cylinder3.B2.phi})) * \text{der}(\text{cylinder3.B2.phi})$ └ differentiate: $\text{dcylinder3.B2.R_rel.T}[3,3]/\text{dtime} = \text{der}(\text{cylinder3.B2.R_rel.T}[3,3])$ └ scalarize(9): $\text{cylinder3.B2.R_rel.T} = \{\{1.0, 0.0, 0.0\}, \{-0.0, \text{cylinder3.B2.phi}\}\} \Rightarrow \text{cylinder3.B2.R_rel.T}[3,3] = \cos(\text{cylinder3.B2.phi})$ └ simplify: $\text{cylinder3.B2.R_rel.T} = \{\{1.0 * 1.0 + (1.0 - 1.0 * 1.0) * \cos(\text{cylinder3.B2.phi}), \{0.0, -\sin(\text{cylinder3.B2.phi}), \cos(\text{cylinder3.B2.phi})\}\}$ └ substitute: $\{\{\text{cylinder3.B2.e}[1] * \text{cylinder3.B2.e}[1] + (1.0 - \text{cylinder3.B2.e}[1] * \text{cylinder3.B2.e}[1]) * \cos(\text{cylinder3.B2.phi}), 0.0 * 0.0 + (1.0 - 0.0 * 0.0) * \cos(\text{cylinder3.B2.phi})\}\}$ └ inline: $\text{cylinder3.B2.R_rel} = \text{Modelica.Mechanics.MultiBody.Rotation}(\text{cylinder3.B2.e}[1], \text{cylinder3.B2.e}[2], \text{cylinder3.B2.e}[3], \text{cylinder3.B2.w}, \text{cylinder3.B2.e}[3] * \text{cylinder3.B2.w})$ └ original: $\text{R_rel} = \text{Frames.planarRotation}(\text{e}, \text{phi_offset} + \text{phi}, \text{w}); \Rightarrow \text{flattened:}$ 	

Performance Profiling

(Here: Profiling all equations in MSL 3.2.1 DoublePendulum)

- ▶ Measuring performance of equation blocks to find bottlenecks
 - ▶ Useful as input before model simplification for real-time platforms
- ▶ Integrated with the debugger so it is possible to show what the slow equations compute
- ▶ Suitable for real-time profiling (less information), or a complete view of all equation blocks and function calls

Equations Browser							Defines
Index	Type	Equation	Executi	Max time	Time	Fraction	Variable
+ 876	regular	linear, size 2	4602	0.000501	0.0134	75.7%	damper.a_rel revolute2.frame_b.f[2]
- 836	regular	(assignment) ...evolute2.phi	1534	2.57e-05	0.000377	2.12%	
- 840	regular	(assignment) ...mper.phi_rel	1534	1.38e-05	0.000237	1.33%	
- 837	regular	(assignment) ...evolute2.phi	1534	8.38e-06	0.000235	1.32%	
- 841	regular	(assignment) ...mper.phi_rel	1534	8.48e-06	0.000192	1.08%	
- 849	regular	(assignment) ...mper.phi_rel	1534	8.04e-06	0.000146	0.824%	

OpenModelica Eclipse MDT Algorithmic Code Debugger

Debug - HelloWorld/SimulationModel.mo - Eclipse SDK

File Edit Navigate Search Project Run Window Help

Debug [Simulation Model [Modelica Development Tooling (MDT) GDB]]

- MDT
- Main Thread (stepping)
 - getValueMultipliedByTwo at simulationmodel.mo:13
 - eqFunction_3 at simulationmodel.mo:5
- C:\Users\adeas31\workspaceMDT\HelloWorld\SimulationModel.exe

List of Stack Frames

Name	Declared Type	Value	Actual Type
inValue	Real	1	double
outValue	Real	6.9453280720608359e-308	double

Variables View

```
model SimulationModel
  Real x(start = 1);
  Real y(start = 1);
  algorithm
    x := getValueMultipliedByTwo(x);
    y := x;
  end SimulationModel;

function getValueMultipliedByTwo
  input Real inValue;
  output Real outValue;
  algorithm
    outValue := inValue * 2;
  end getValueMultipliedByTwo;
```

Outline

- getValueMultipliedByTwo
 - inValue (Real - IN)
 - outValue (Real - OUT)
- SimulationModel
 - x
 - y

Console

Simulation Model [Modelica Development Tooling (MDT) GDB] C:\Users\adeas31\workspaceMDT\HelloWorld\SimulationModel.exe

Output View

Adding Breakpoints

File Name:

Line Number:

Enabled:

Ignore Count:

Condition

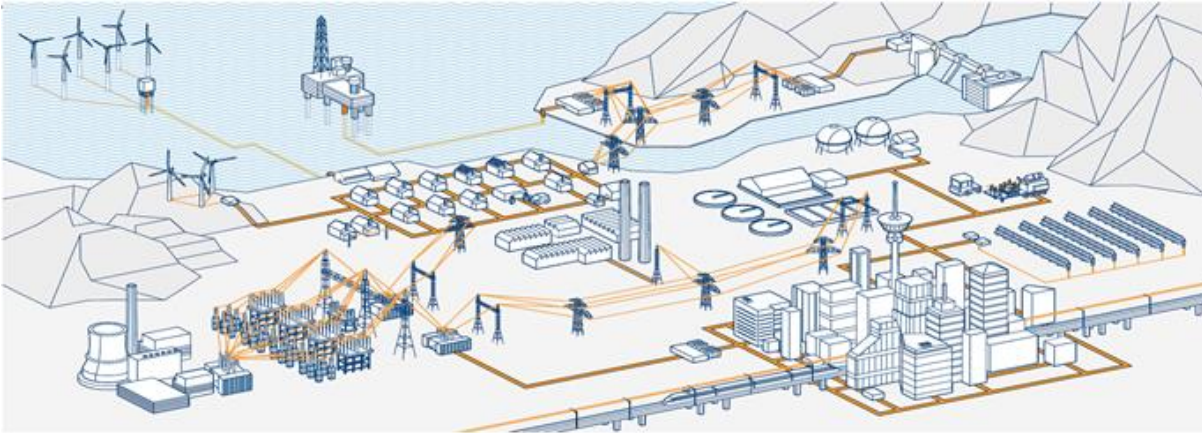
Expression:

Time:

Value: Equal Greater Greater Equal Less Less Equal

ABB Commercial Application Use of Debugger

- ABB OPTIMAX® provides advanced model based control products for power generation and water utilities.

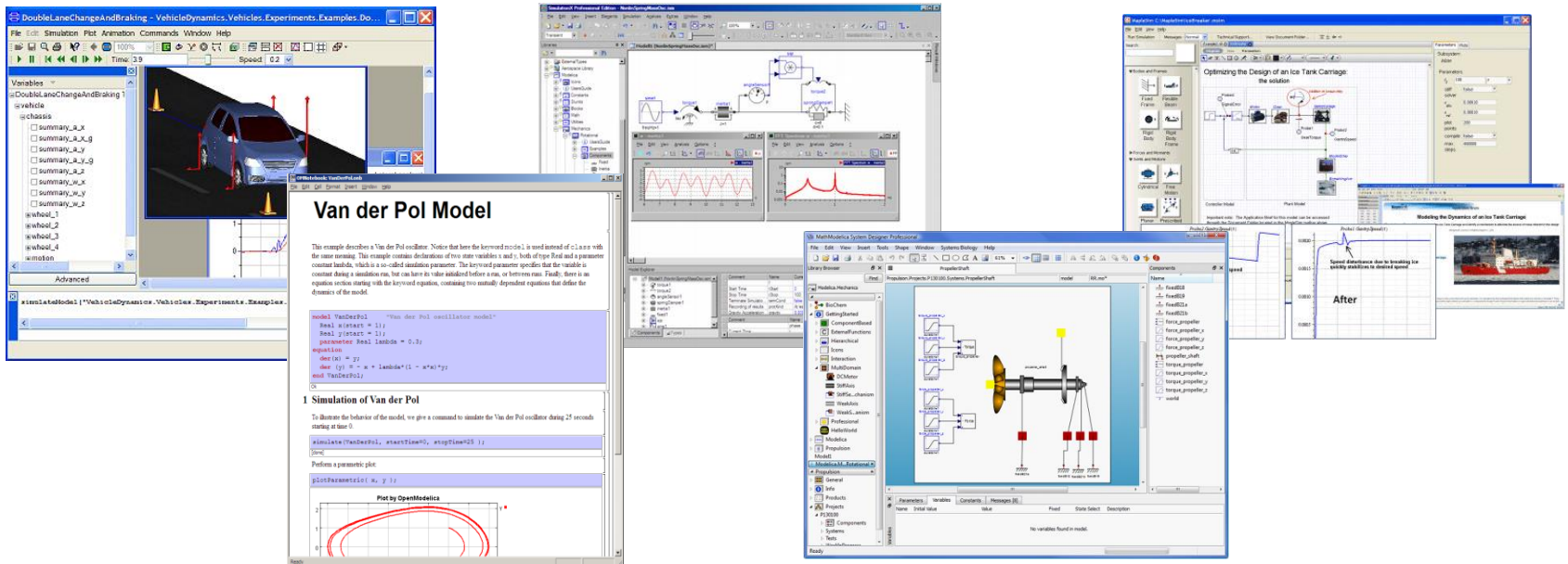


- ABB: “OpenModelica provides outstanding debugging features that help to save a lot of time during model development.”

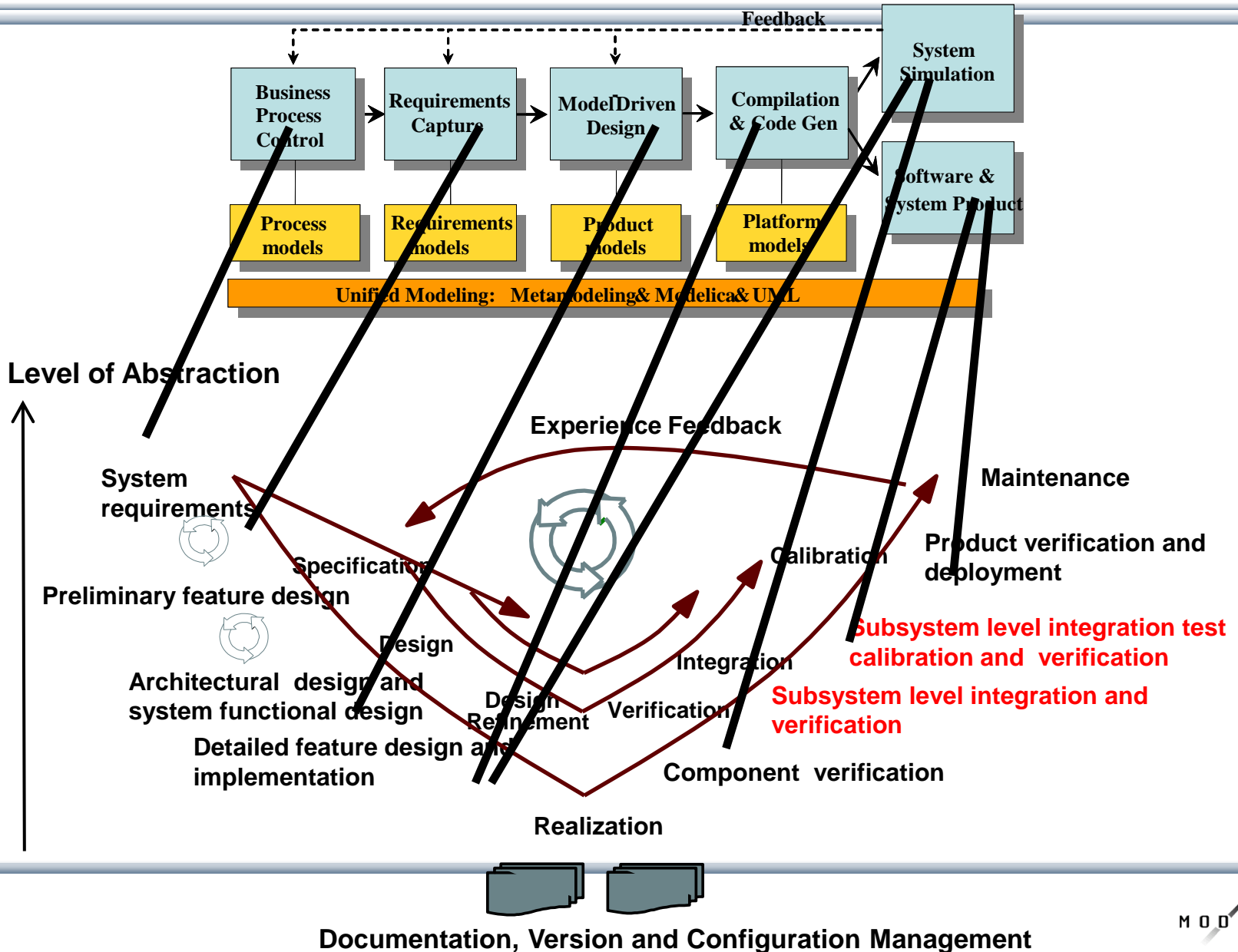
Part III

Dynamic Verification/Testing of Requirements vs Usage Scenario Models

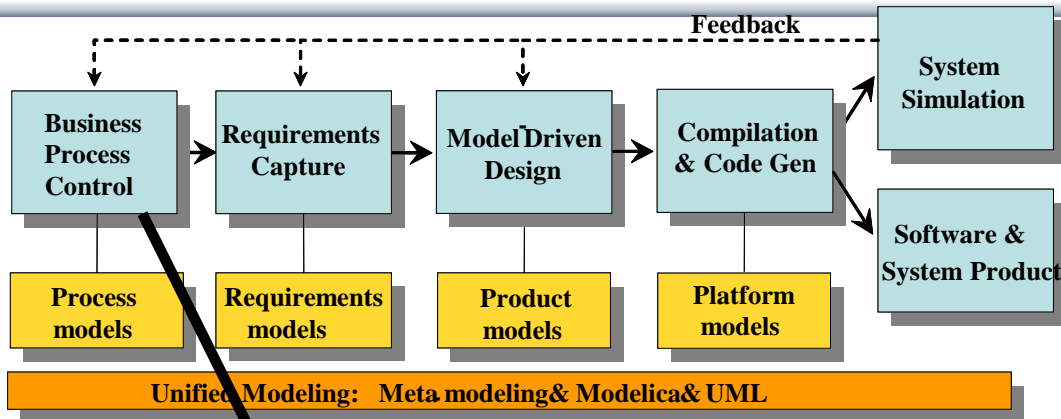
Wladimir Schamai, Lena Buffoni, Peter Fritzson and contributions from MODRIO partners



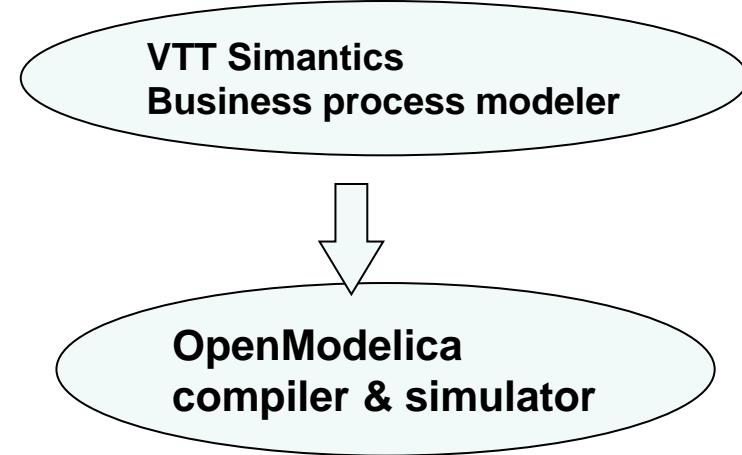
OpenModelica and Papyrus Based Model-Based Development Environment to Cover Product-Design V



Business Process Control and Modeling

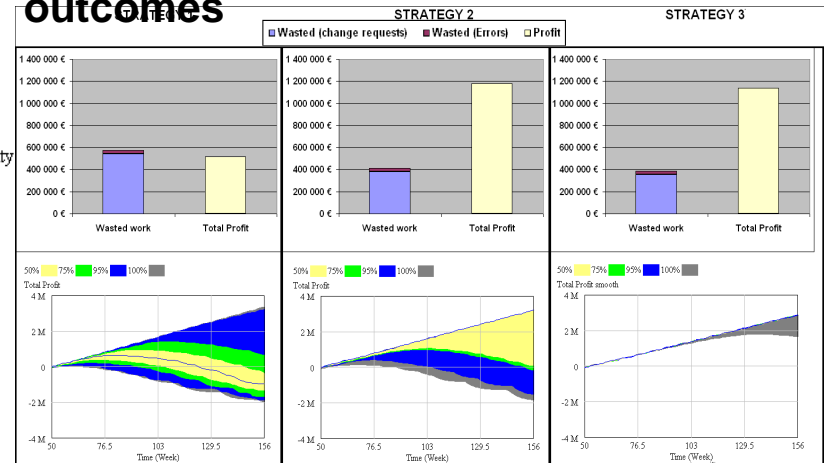
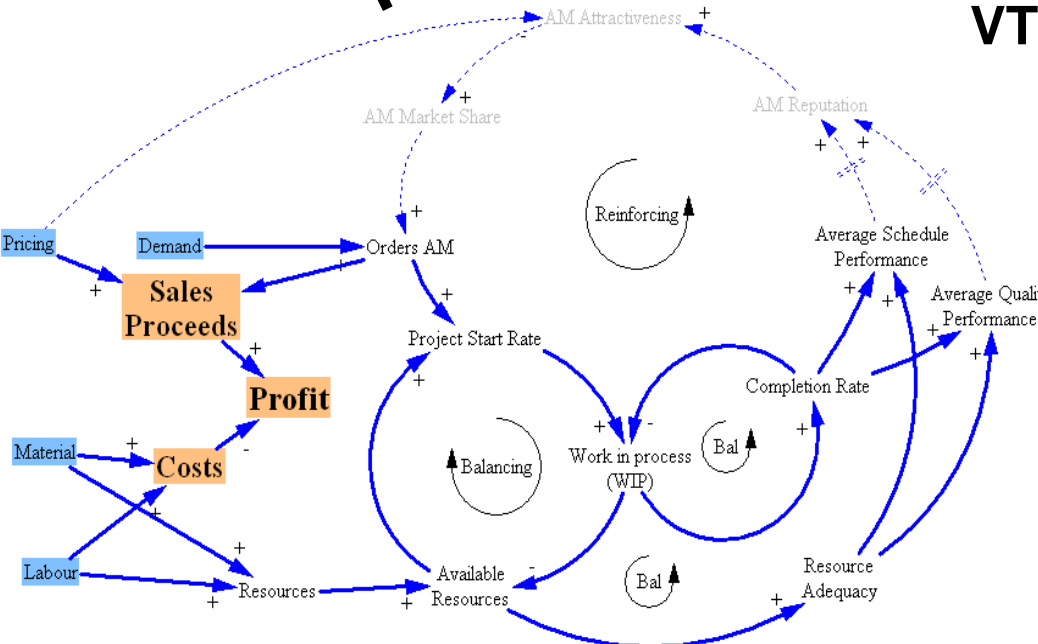


OpenModelica based simulation

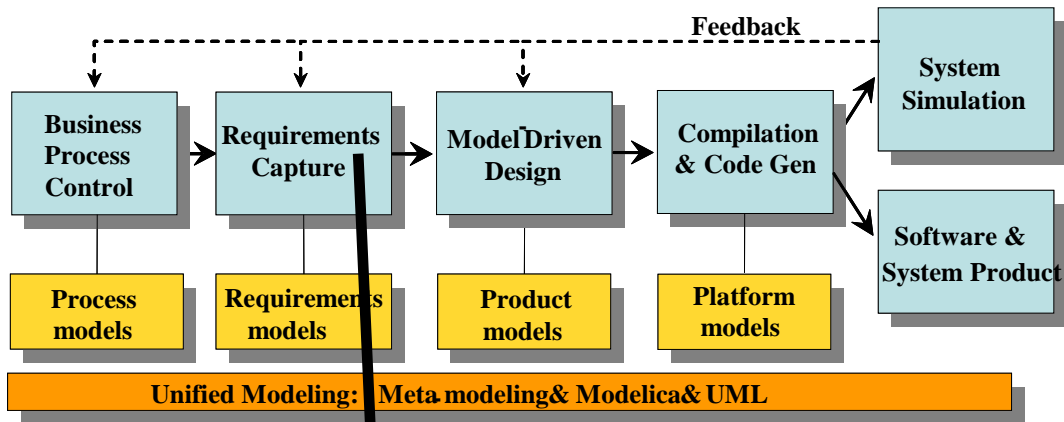


Metso Business model & simulation
VTT Simantics Graphic Modeling Tool

Simulation of 3 strategies with outcomes



Requirement Capture



vVDR (virtual Verification of Designs against Requirements)

in ModelicaML UML/Modelica Profile, part of OpenModelica

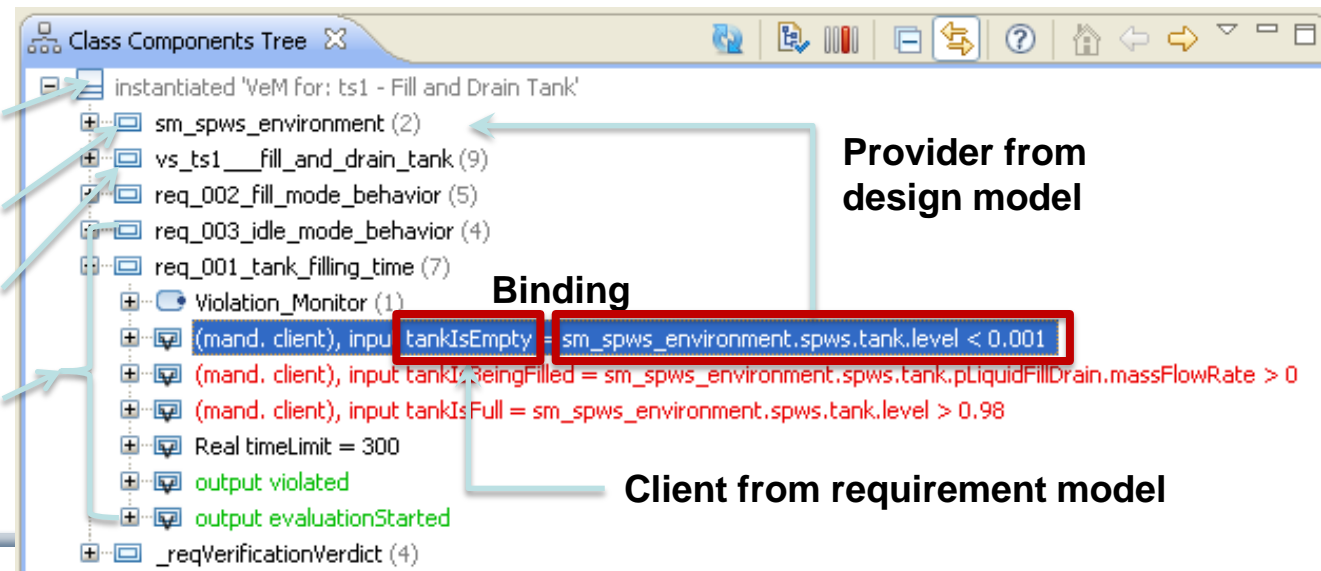
OpenModelica based simulation

Verification Model

Design Model

Scenario Model

Requirement Models



Provider from design model

Binding

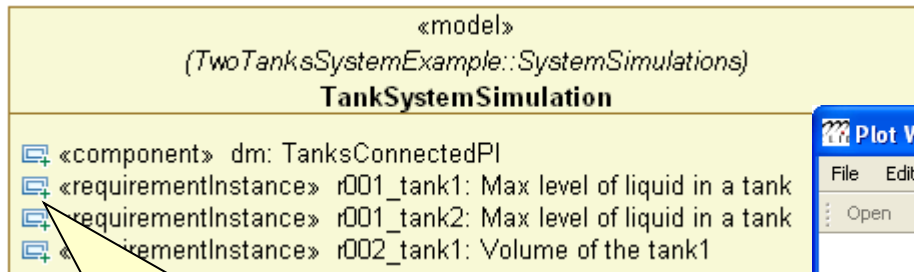
Client from requirement model

OpenModelica – ModelicaML UML Profile

Based on Open-Source Papyrus UML and OpenModelica

- ModelicaML is a UML Profile for SW/HW modeling
 - Applicable to “pure” UML or to other UML profiles, e.g. SysML
- Standardized Mapping UML/SysML to Modelica
 - Defines transformation/mapping for **executable** models
 - Being **standardized** by OMG
- ModelicaML
 - Defines graphical concrete syntax (graphical notation for diagram) for representing Modelica constructs integrated with UML
 - Includes graphical formalisms (e.g. State Machines, Activities, Requirements)
 - Which do not yet exist in Modelica language (extension work ongoing)
 - Which are translated into executable Modelica code
 - Is defined towards generation of executable Modelica code
 - Current implementation based on the Papyrus UML tool + OpenModelica

Example: Simulation and Requirements Evaluation

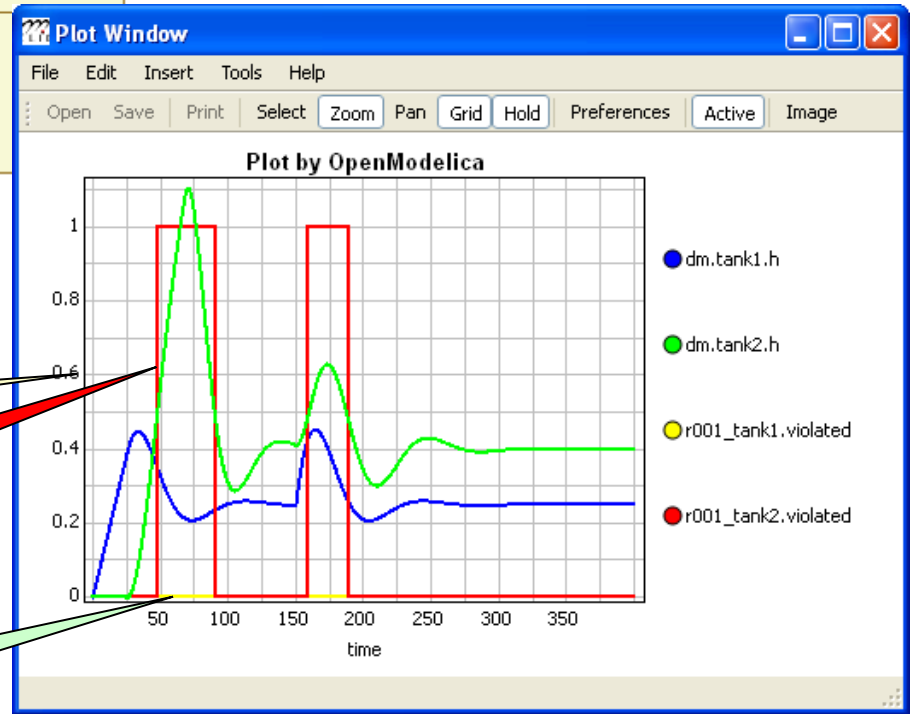


Req. 001 is instantiated 2 times (there are 2 tanks in the system)

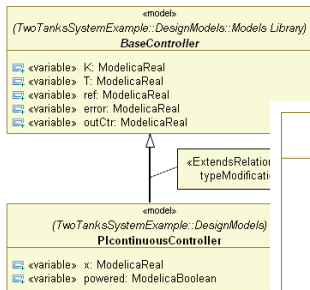
tank-height is 0.6m

Req. 001 for the tank2 is violated

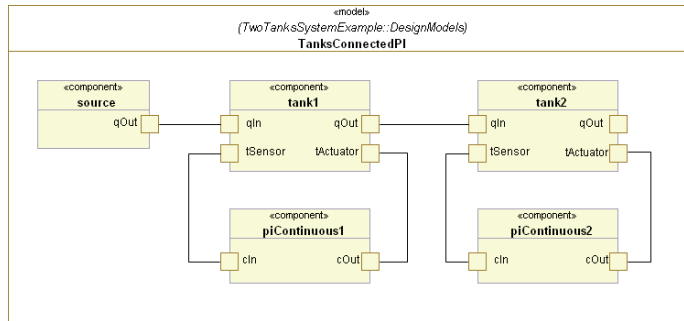
Req. 001 for the tank1 is not violated



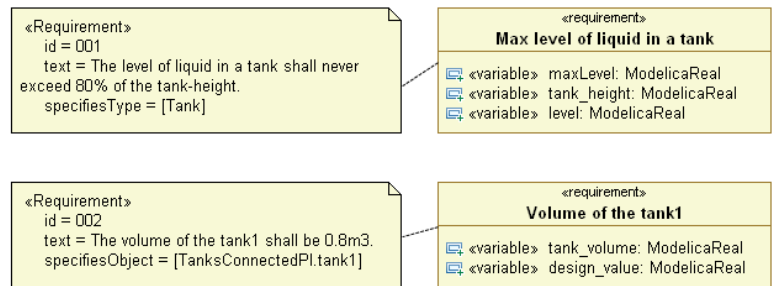
ModelicaML: Graphical Notation



Structure

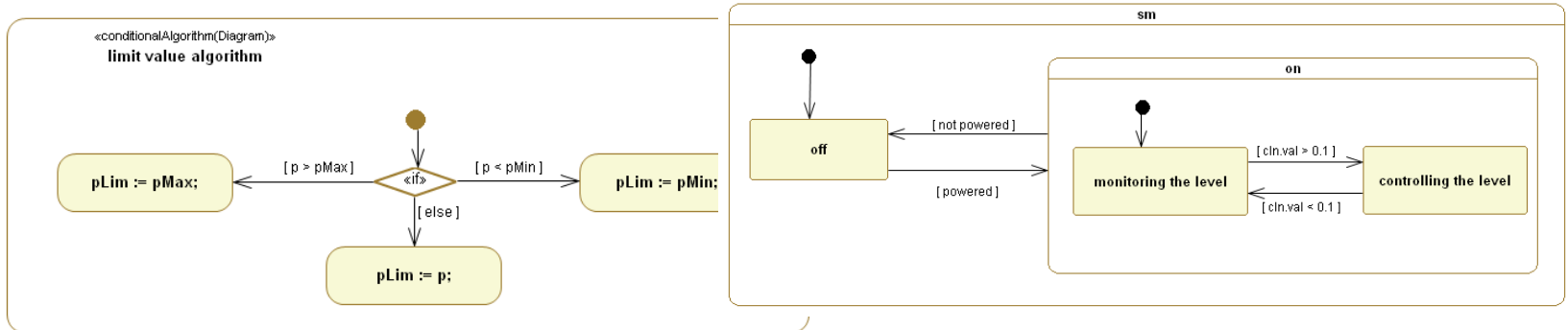


Requirements

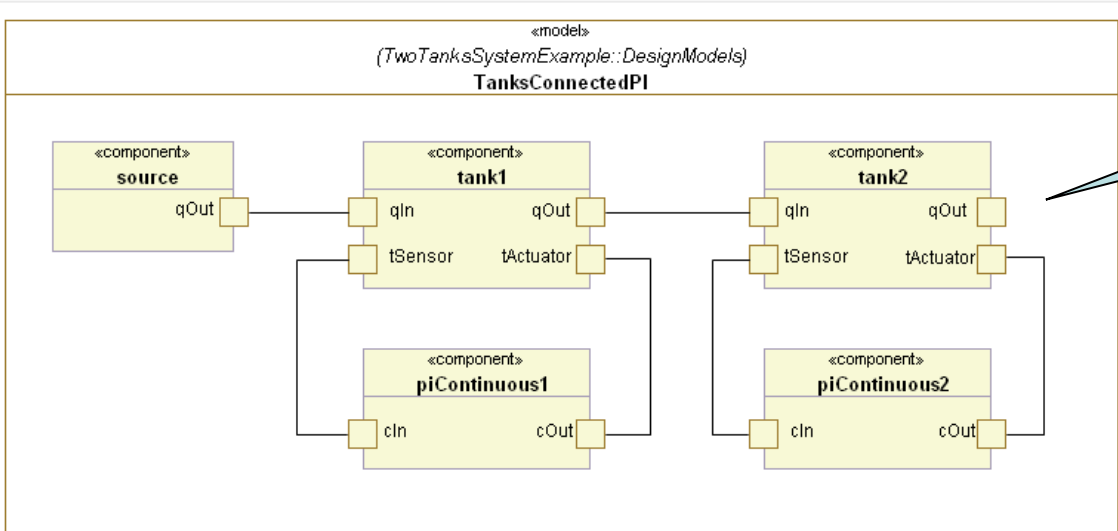


a

Behavior

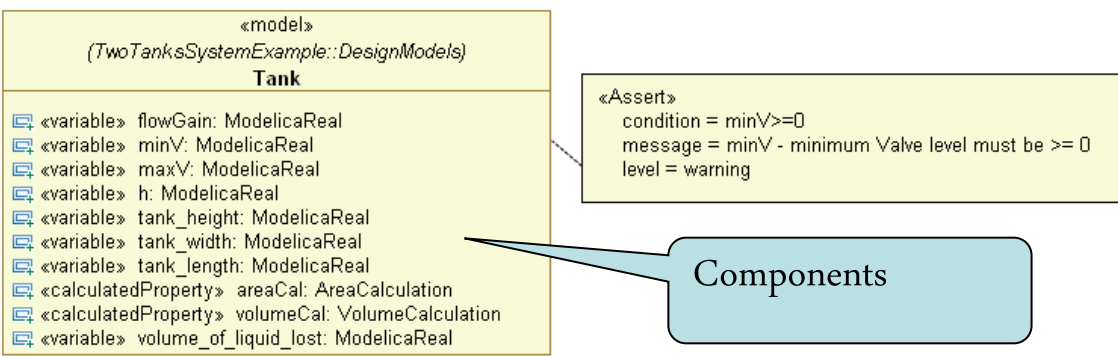


Example: Representation of System Structure

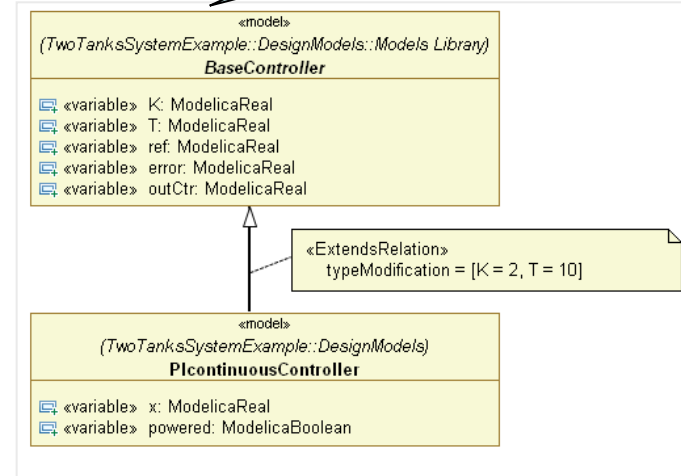


Interconnections

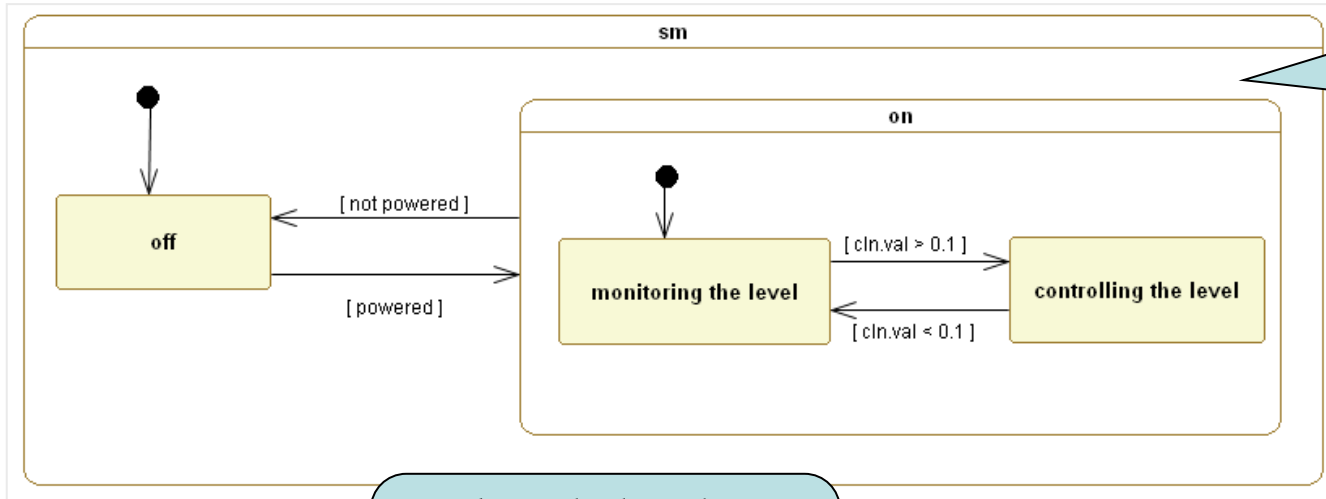
Inheritance



Components



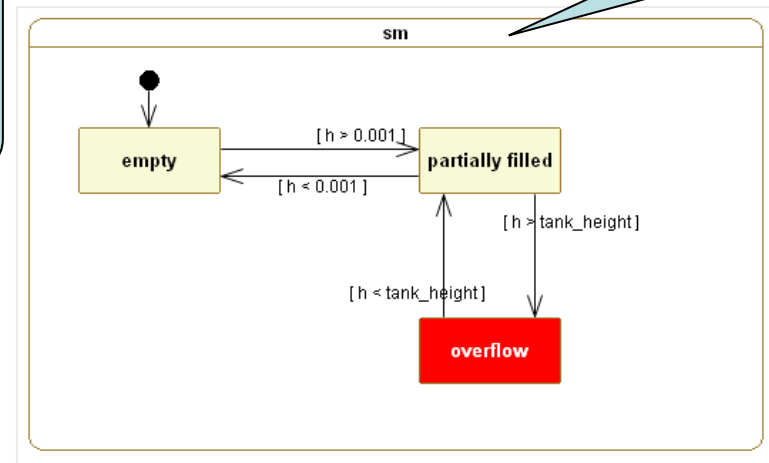
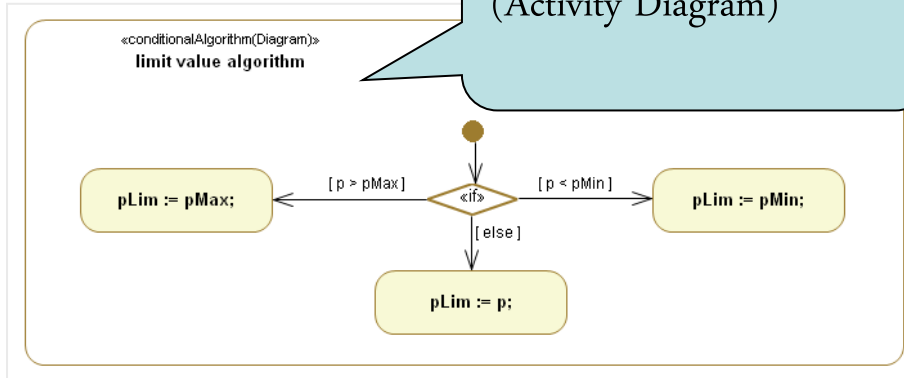
Example: Representation of System Behavior



State Machine of the Controller

State Machine of the Tank

Conditional Algorithm (Activity Diagram)



Example: Representation of System Requirements

Textual Requirement

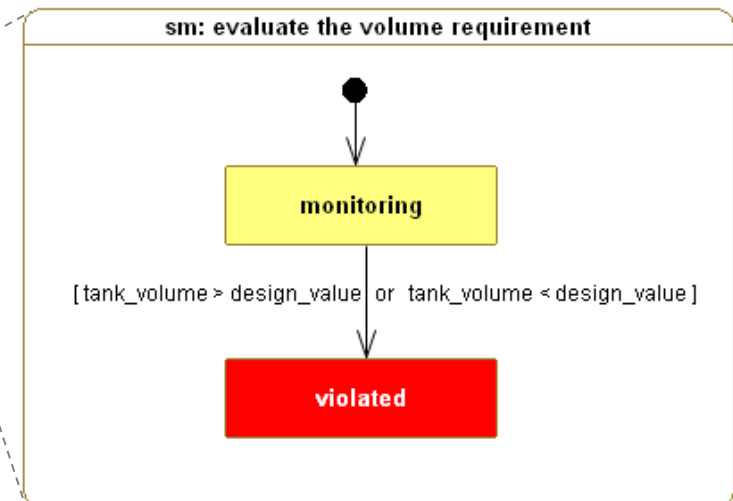
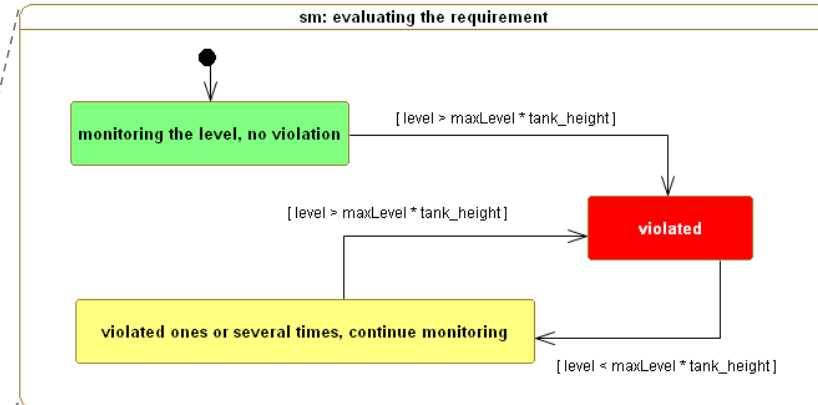
«Requirement»
id = 001
text = The level of liquid in a tank shall never exceed 80% of the tank-height.
specifiesType = [Tank]

Formalized Requirement

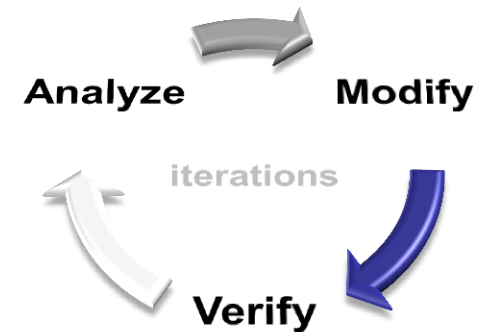
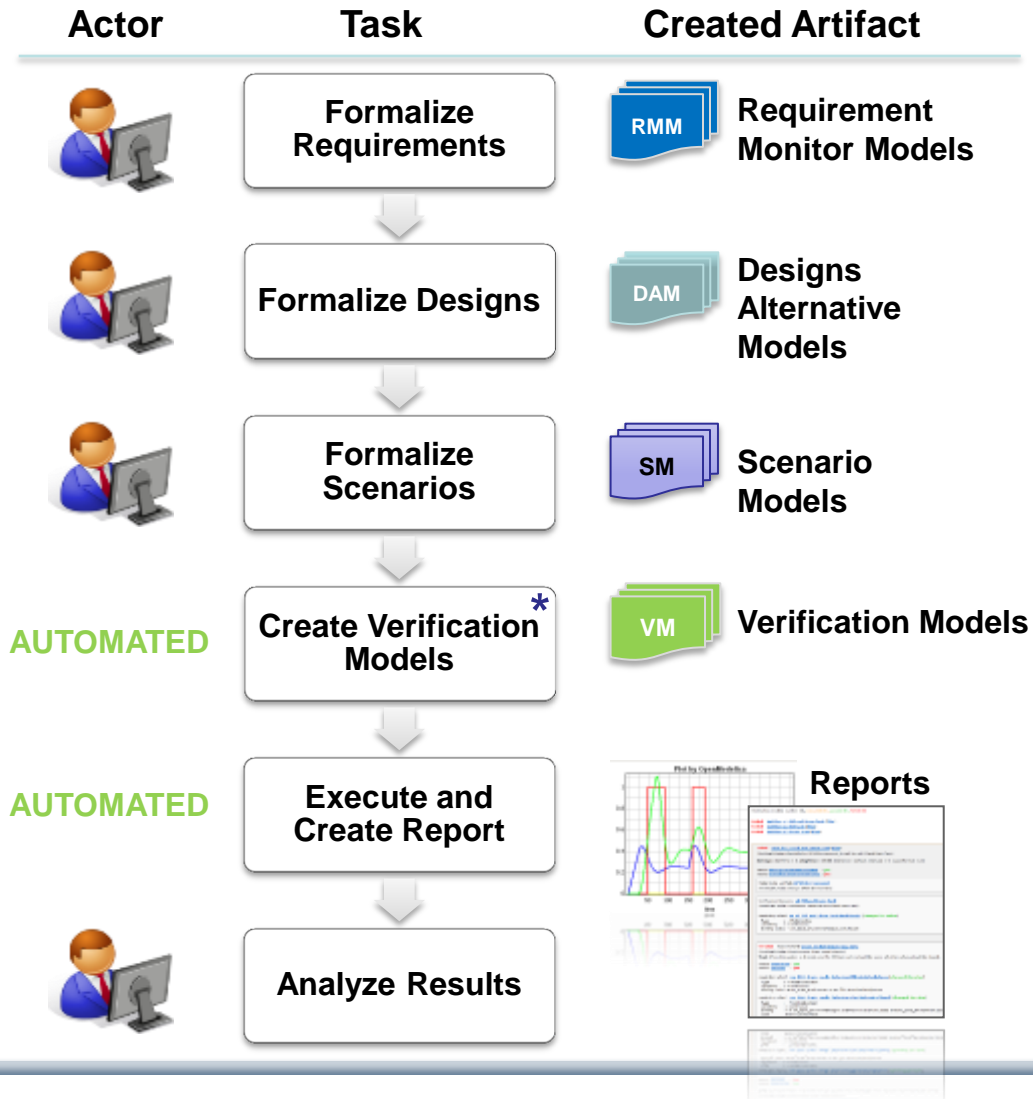
«requirement»
Max level of liquid in a tank
«variable» maxLevel: ModelicaReal
«variable» tank_height: ModelicaReal
«variable» level: ModelicaReal

«Requirement»
id = 002
text = The volume of the tank1 shall be 0.8m3.
specifiesObject = [TanksConnectedPI.tank1]

«requirement»
Volume of the tank1
«variable» tank_volume: ModelicaReal
«variable» design_value: ModelicaReal



vVDR Method – virtual Verification of Designs vs Requirements

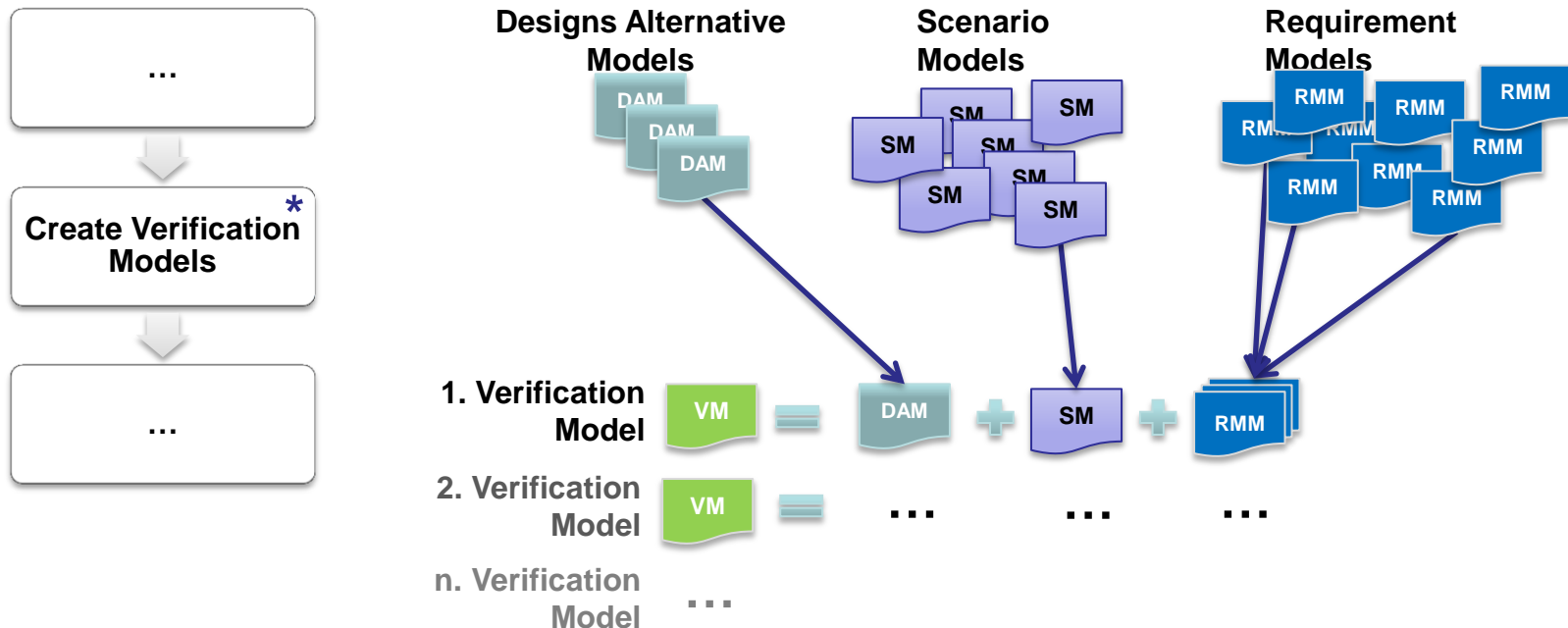


Goal: Enable on-demand verification of designs against requirements using automated model composition at any time during development.

Challenge

We want to verify **different design alternatives** against **sets of requirements** using **different scenarios**. Questions:

- 1) How to **find valid combinations** of **design alternatives**, **scenarios** and **requirements** in order to enable an automated composition of verification models?
- 2) Having found a valid combination: How to **bind all components correctly**?

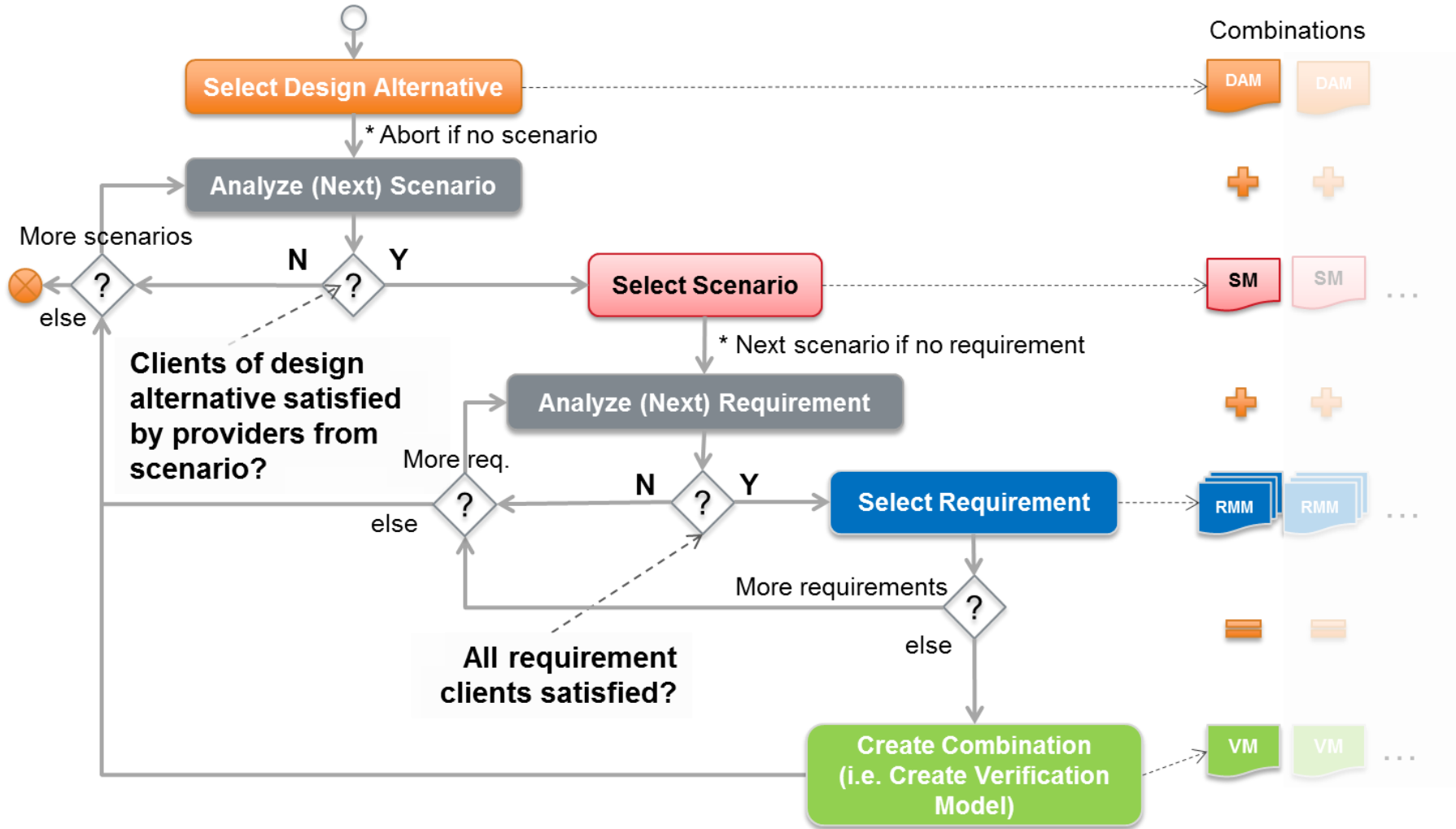


Composing Verification Models

main idea

- Collect all **scenarios**, **requirements**, import **mediators**
- Generate/compose *verification models* automatically:
 - Select the **system model** to be verified
 - Find all **scenarios** that can stimulate the selected system model (i.e., for each mandatory client check whether the binding expression can be inferred)
 - Find **requirements** that are implemented in the selected system model (i.e., **check** whether for **each requirement** for all mandatory clients binding expressions can be inferred)
- Present the list of scenarios and requirements to the user
 - The user can select only a subset of scenarios or requirements he/she wishes to consider

Generating/Composing Verification Models algorithm

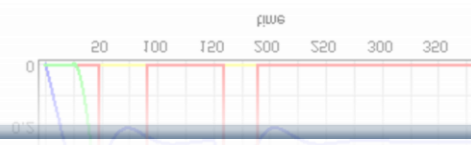


Simulation and Report Generation in ModelicaML

Verification models are simulated.

The generated **Verification Report** is a prepared summary of:

- Configuration, bindings
- Violations of requirements
- etc.



Verification models number (3), executed (3), passed (0), failed (3)

Failed [VeM for: s1-Fill and Drain Tank \(Plot\)](#)

Failed [VeM for: s2-Fill tank \(Plot\)](#)

Failed [VeM for: s3-Drain tank \(Plot\)](#)

Failed [VeM for: s1-Fill and Drain Tank \(Plot\)](#)

(ModelicaMLModel::GenVeMs for: SPWS Environment_1::VeM for: s1-Fill and Drain Tank)

Settings: startTime = 0, stopTime = 1500, tolerance = default, intervals = 0, outputFormat = plt

verdict [allRequirementsEvaluated](#) : yes

verdict [someRequirementsViolated](#) : yes

Model to be verified: [SPWS Environment](#)

(ModelicaMLModel::Design::SPWS Environment)

Verification Scenario: [s1-Fill and Drain Tank](#)

(ModelicaMLModel::Verification Scenarios::s1-Fill and Drain Tank)

mandatory client: [vs s1 fill and drain tank.tankHeight](#) (changed its value)

Type : = ModelicaReal

Variability : = continuous

Binding code : = sm_spws_environment.spws.tank.height

Violated Requirement: [Drain mode behavior \(ID 004\)](#)

(ModelicaMLModel::Requirements::Drain mode behavior)

Text: When the system is drained only the fill/drain valve should be open, all other valves should be closed.

verdict [evaluated](#) : yes

verdict [violated](#) : yes

mandatory client: [req_004_drain_mode_behavior.fillDrainValveIsOpen](#) (changed its value)

Type : = ModelicaBoolean

Variability : = continuous

Binding code : = sm_spws_environment.spws.fillDrainValve.isFullyOpen

mandatory client: [req_004_drain_mode_behavior.otherValvesAreClosed](#) (changed its value)

Type : = ModelicaBoolean

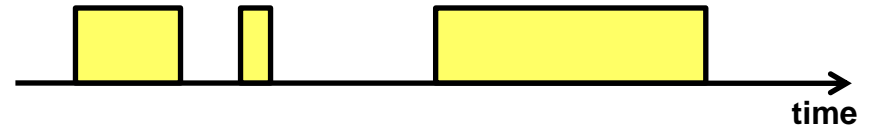
Variability : = continuous

Binding code : = if sm_spws_environment.spws.overflowValve.isFullyClosed and sm_spws_environment.spws.supplyVavle.isFullyClosed
code then true else false

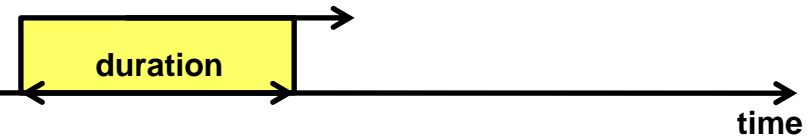
Continuous and Discrete Time Locators for Time-related Requirements – Work in MODRIO project by EDF, LIU, DLR, DS, ..

- A Continuous Time Locator(CTL) specifies one or more time intervals

- Time intervals have a duration

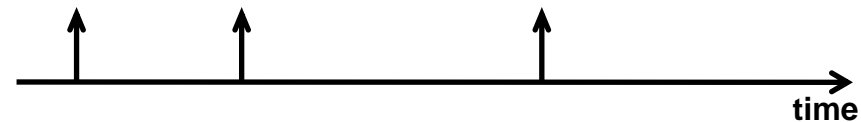


- They usually have a position in time, but a sliding time window defines any time period of a given duration



- A Discrete Time Locator (DTL) defines one or more positions in time and has no duration

- An event is associated with a DTL that specifies when the event occurred



- The difference between events and DTLs is that a DTL is not an object

- That position may be relative to the initialisation of the system or to another DTL

Time Locators Expressed in Modelica

Special FORML-L syntax	Standard Modelica syntax
<i>duringAny</i> duration	duringAny (duration)
<i>after</i> event	after (event)
<i>after</i> event1 <i>untilNext</i> event2	afterUntil (event1, event2)
<i>after</i> event <i>for</i> duration	afterFor (event, duration)
<i>after</i> event <i>within</i> duration	afterWithin (event, duration)
<i>until</i> event	until (event)
<i>every</i> duration1 <i>for</i> duration2	everyFor (duration1, duration2)
<i>when</i> condition <i>changes</i>	Maps to Modelica if

From Text to Simulated Requirement

– Modelica Extended with new Operators

From a text requirement expressing a condition:

A - In the absence of any Backup Power Supply (BPS) component failure or in the presence of a single sensor failure, when the BPS is not under maintenance, in case of loss of MPS, and if safety injection is required, Set1 must be powered within 20 s

```
model P2a extends Condition;
```

```
  input ConditionStatus bPSNeeded, sARequired, set1Powered;
```

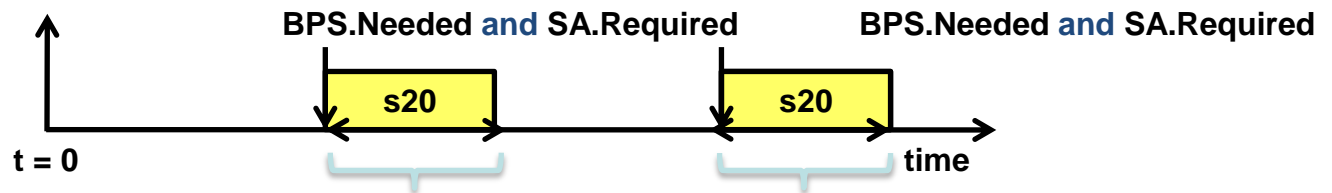
```
equation
```

```
  status = if afterWithin (bPSNeeded == notViolated and  
                        sARequired == notViolated, 20) then
```

```
    if set1Powered == notViolated then
```

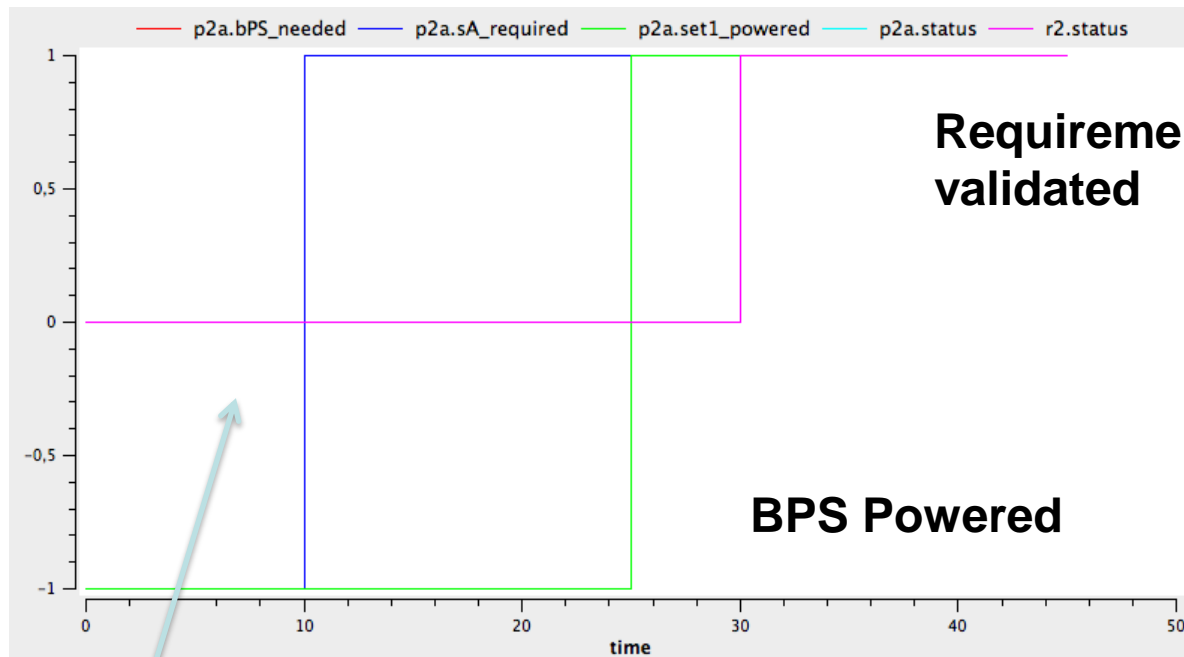
```
      notViolated else violated else undefined;
```

```
end P2a;
```



Set1.Powered must become true within the timeframe s20 and remain true afterwards

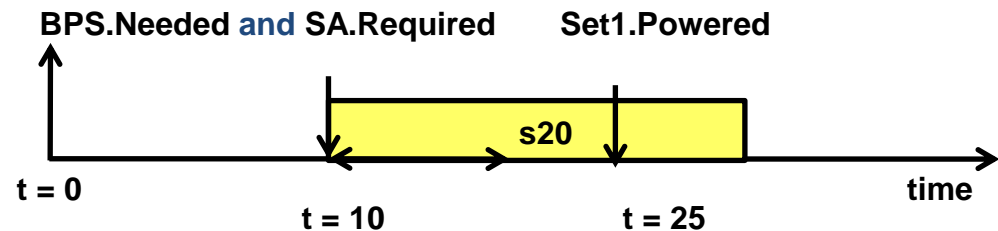
From Text to Simulated Requirement – Requirement not Violated – OpenModelica Simulation



**3-valued logic
prototype:**
1 – true
0 – false
-1 – undefined

**Requirement
undefined
outside the
specified time
window**

Within 20s



Industrial Use Case for Requirements Verification and Model Composition in ModelicaML

OPENPROD-Project Case Study, performed 2012; presented 2013

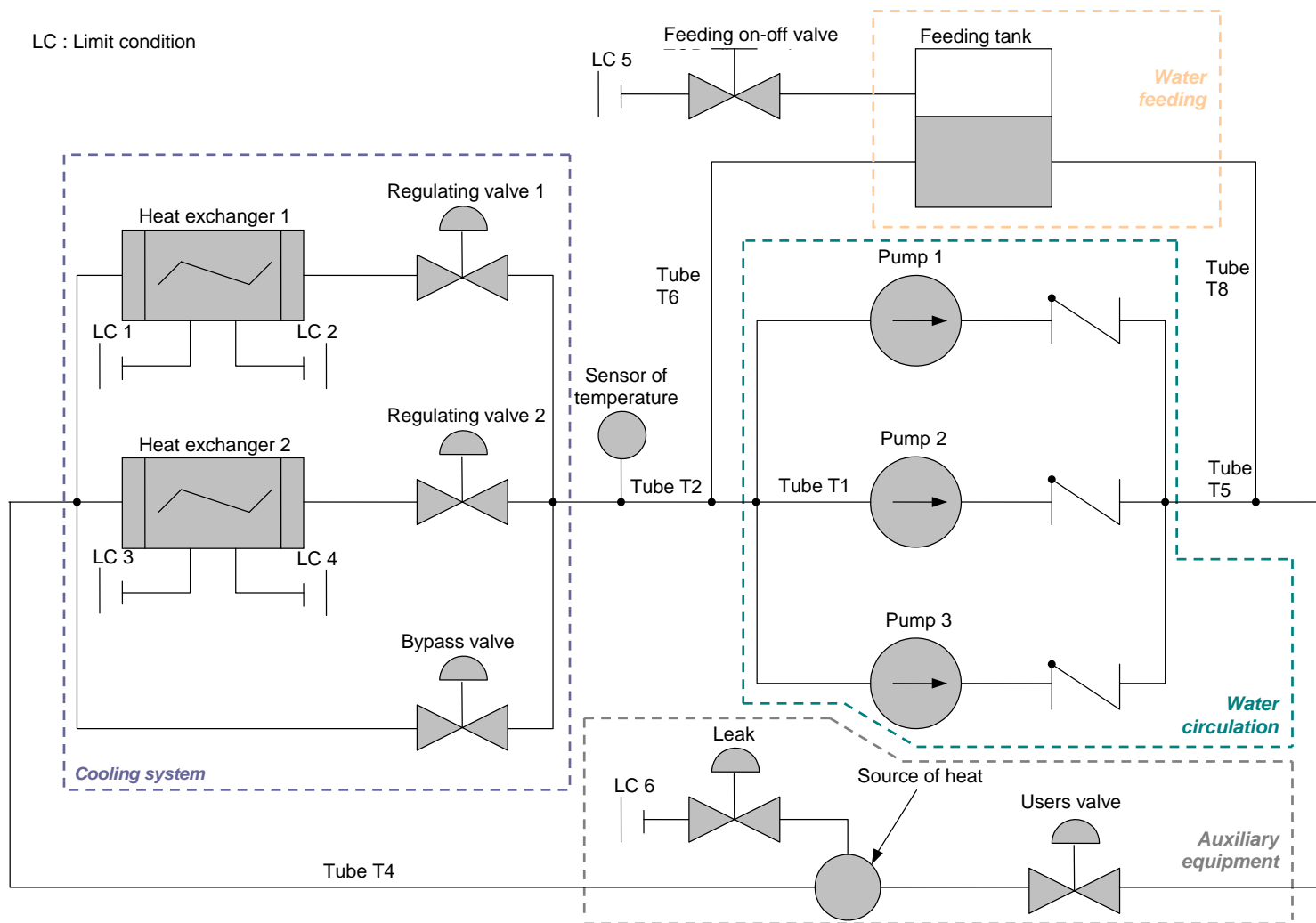
- Wladimir SCHAMAI (EADS Innovation Works, Germany)
- Peter Fritzson (Linköping University)

- Audrey JARDIN (EDF - R&D, France)
- Daniel BOUSKELA (EDF - R&D, France)

•Mar. 2013



EDF Use Case – System Description of SRI system (Intermediate Cooling System) in turbine hall of a nuclear power plant



System Requirements

- *#002*: The set point of the SRI water temperature must be held at a minimum value of 17°C.
- *#003*: In a normal operating mode, the water temperature of the SRI circuit should be between $T_s - e$ and $T_s + e$ (T_s : set point temperature).
- *#0083*: A pump must not start more than 3 times per hour.
- *#013*: In a normal operating mode, there must not be less than 2 operating pumps during more than 2s.
- *#007*: The water temperature must not vary more than 10°C/hour.

SRI Case Study Conclusion and Lessons Learnt

- Showed **applicability of vVDR method** to **realistic industrial applications**
- ModelicaML is a **promising prototype** implementation of the vVDR method, needs improved usability and stability
- Lessons learnt:
 - **Formalized requirements** should be **tested separately** in order to ensure correctness
 - **Model validity asserts** must be included
 - Parameterized **requirement monitors** can be re-used as **library components** (later realized in MODRIO project)
- Work is **continued** in the ITEA2 **MODRIO** project
 - Stochastic aspects (model uncertainties, tolerances in requirements, ...) should be taken into account