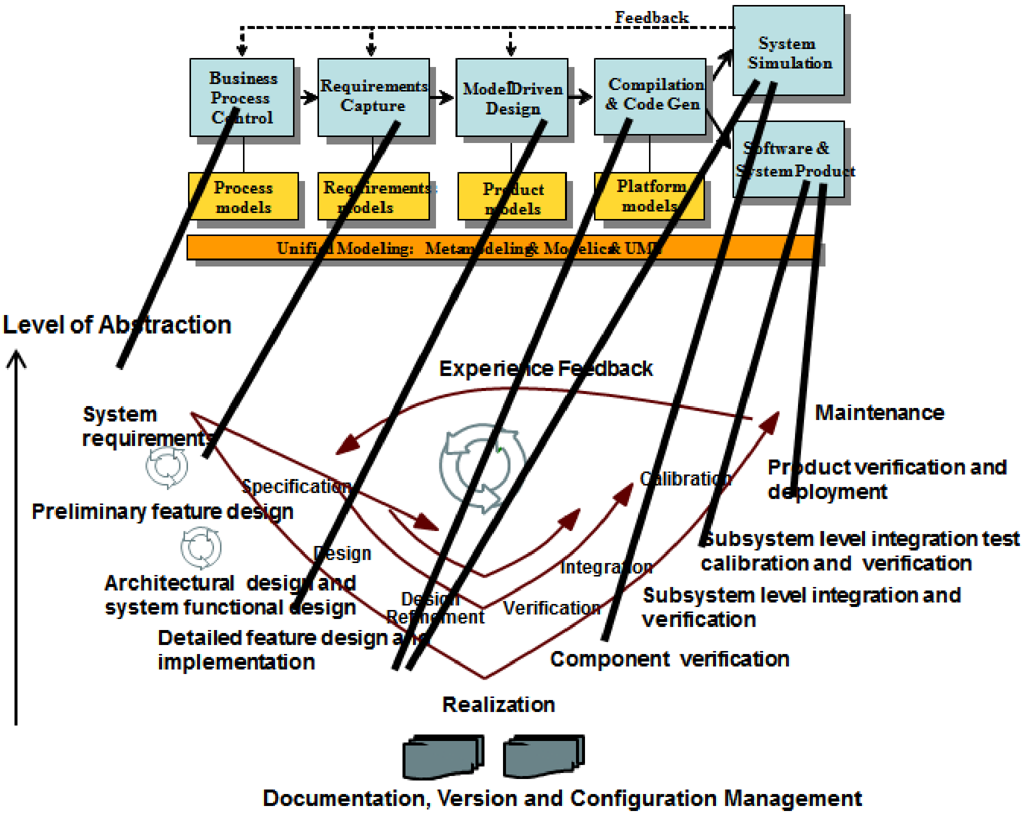
Integrated Modeling of CPS including Requirements: Open Source MBSE Tools Based on Modelica and UML



LCCC MBSE Workshop, Lund
May 4, 2015

Peter Fritzson
peter.fritzson@liu.se

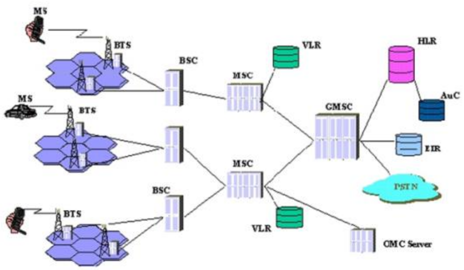
Vice Chairman of Modelica Association
Director of Open Source Modelica Consortium
Professor at Linköping University

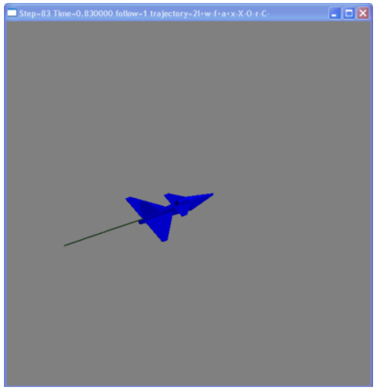
Linköpings universitet

MODELICA

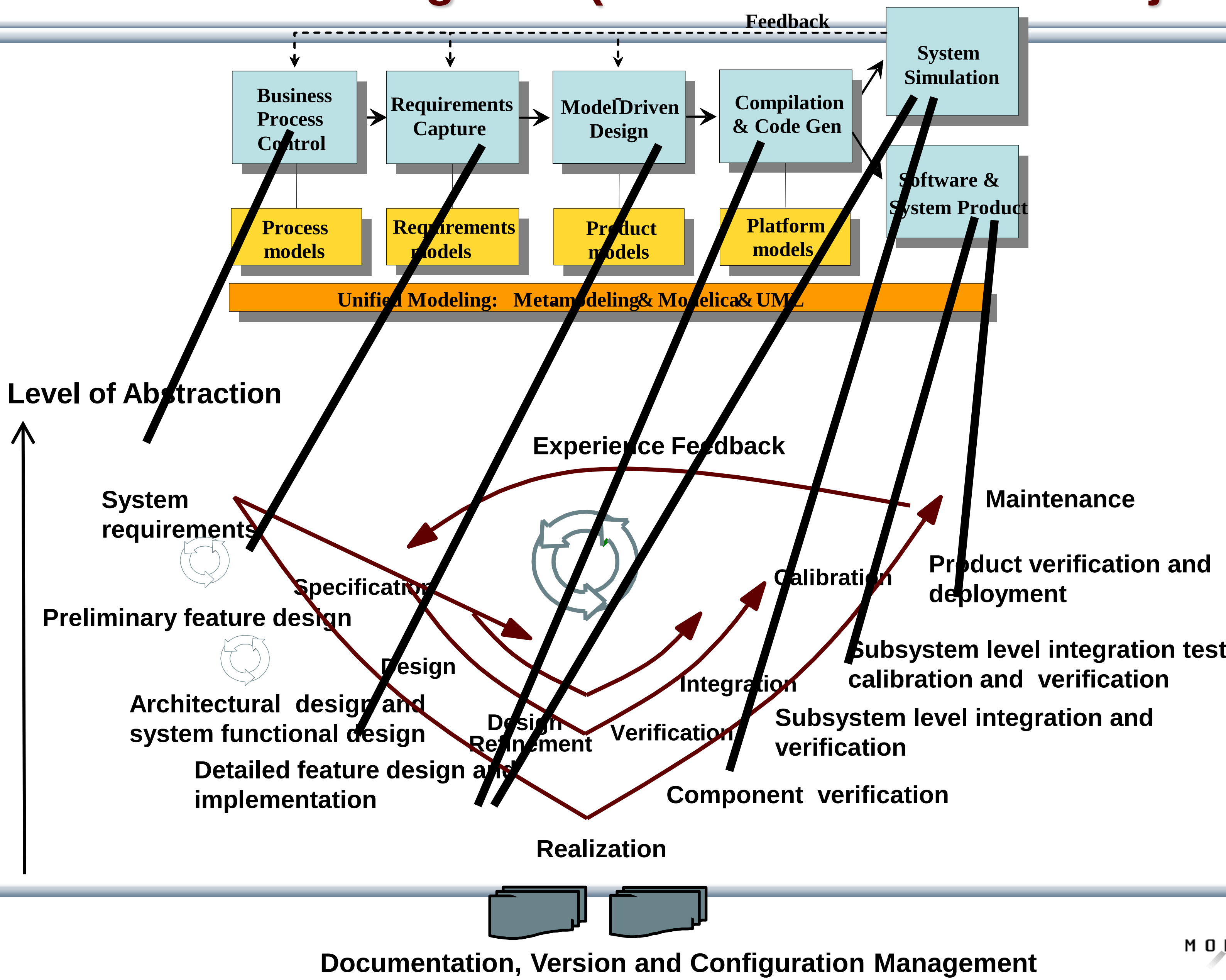
Industrial Challenges for Complex Cyber-Physical System Products of both Software and Hardware

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

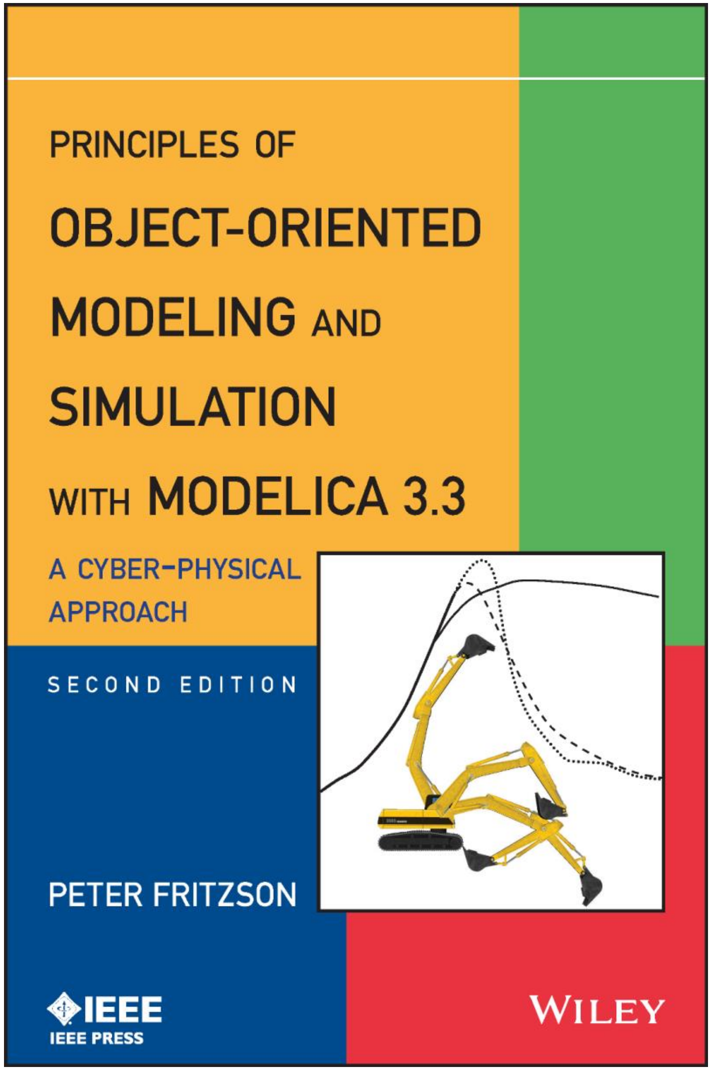




Open Source Model-Based Development Environment Covers Product-Design V – (OPENPROD ITEA2 Project)



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

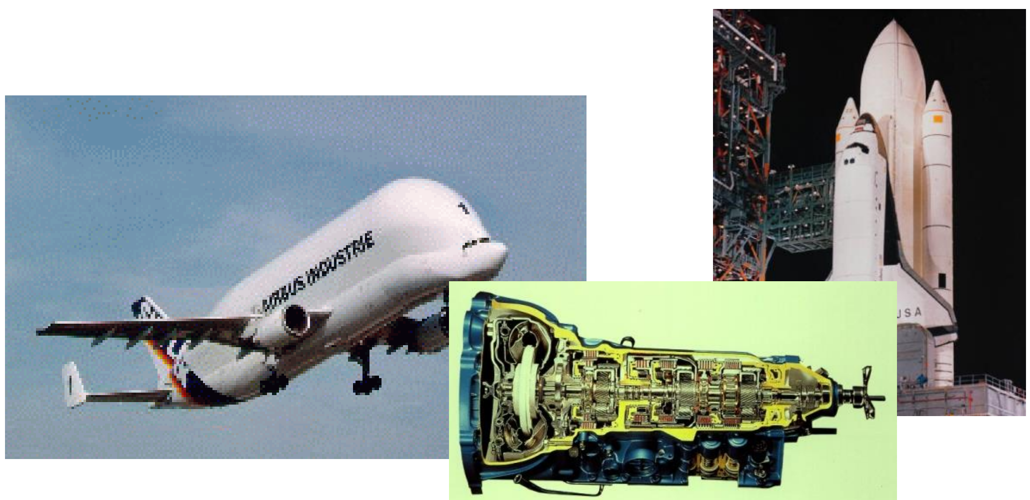
- OpenModelica
  - www.openmodelica.org
- Modelica Association
  - www.modelica.org

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

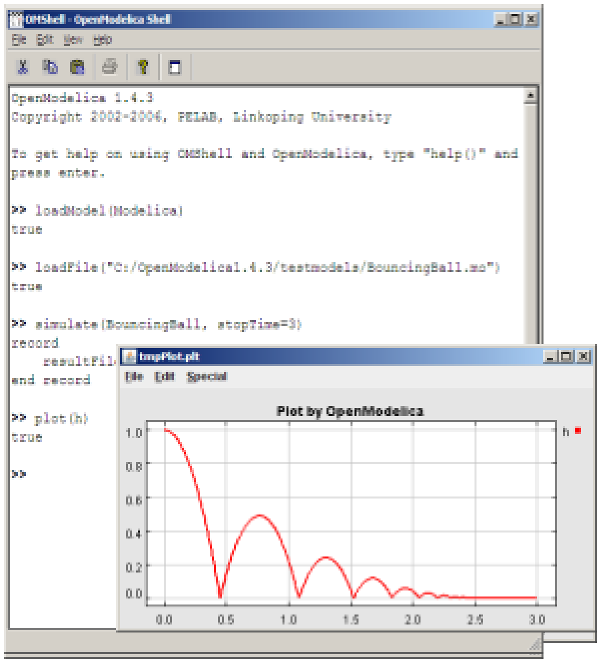
Introduction to the OpenModelica Environment

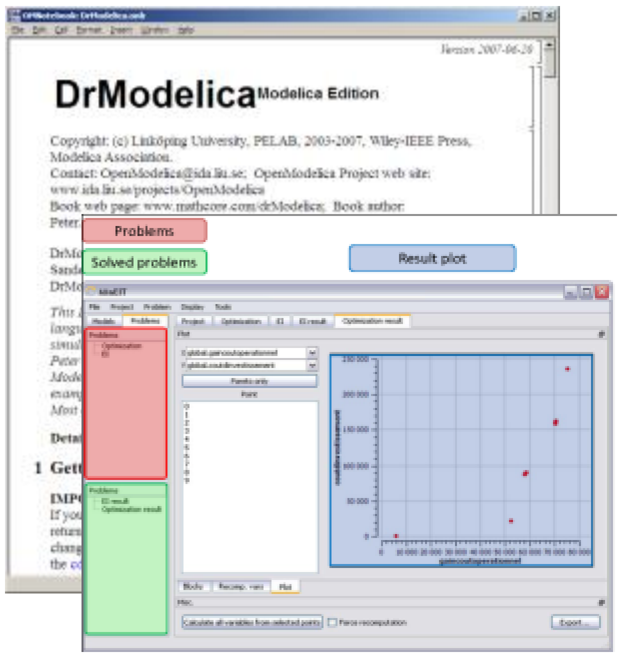


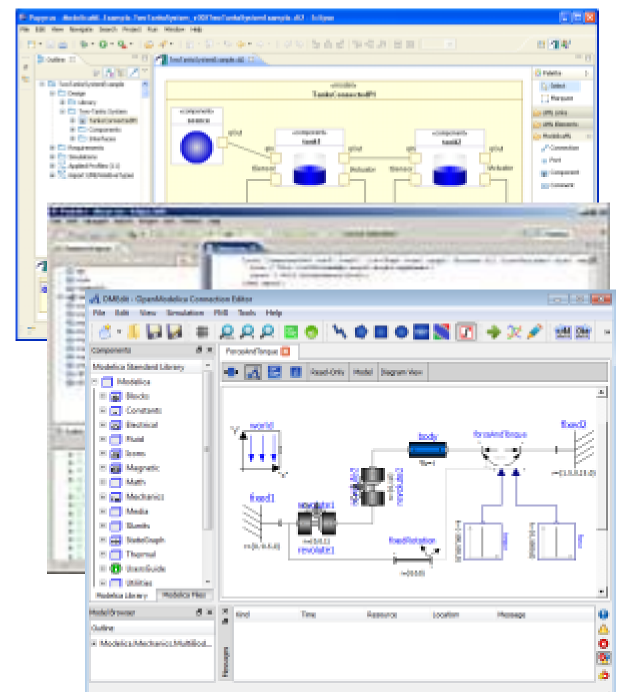
The OpenModelica Open Source Environment
www.openmodelica.org

- 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







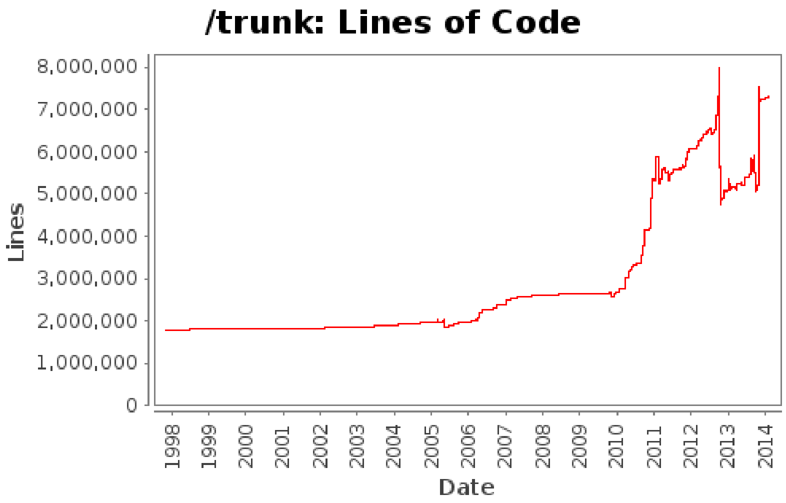
OSMC – International Consortium for Open Source Model-based Development Tools, 43 Members

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



Industrial members

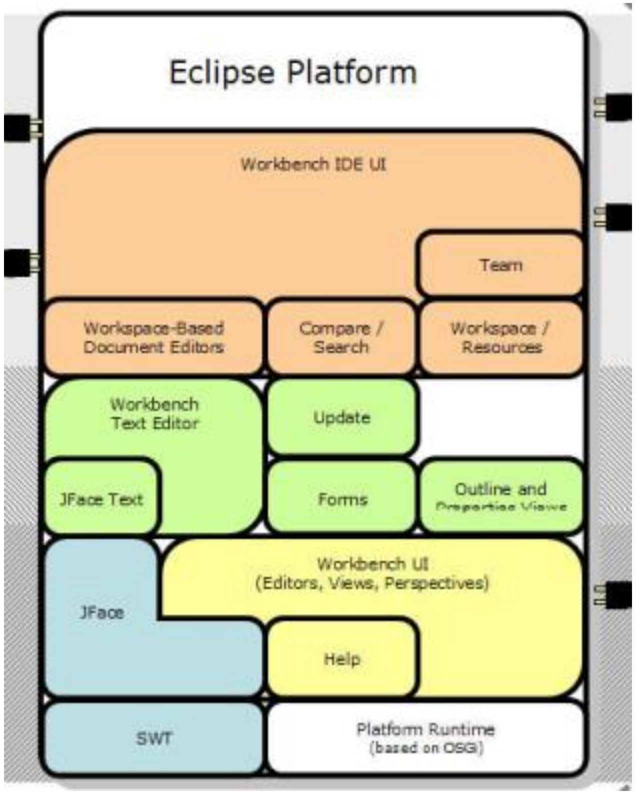
- ABB AB, Sweden
- Bosch Rexroth AG, Germany
- Siemens Turbo, Sweden
- CDAC Centre, Kerala, India
- Creative Connections, Prague
- DHI, Aarhus, Denmark
- EDF, Paris, France
- Equa Simulation AB, Sweden
- Fraunhofer IWES, Bremerhaven
- IFP, Paris, France
- ISID Dentsu, Tokyo, Japan
- ITI, Dresden, Germany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

University members

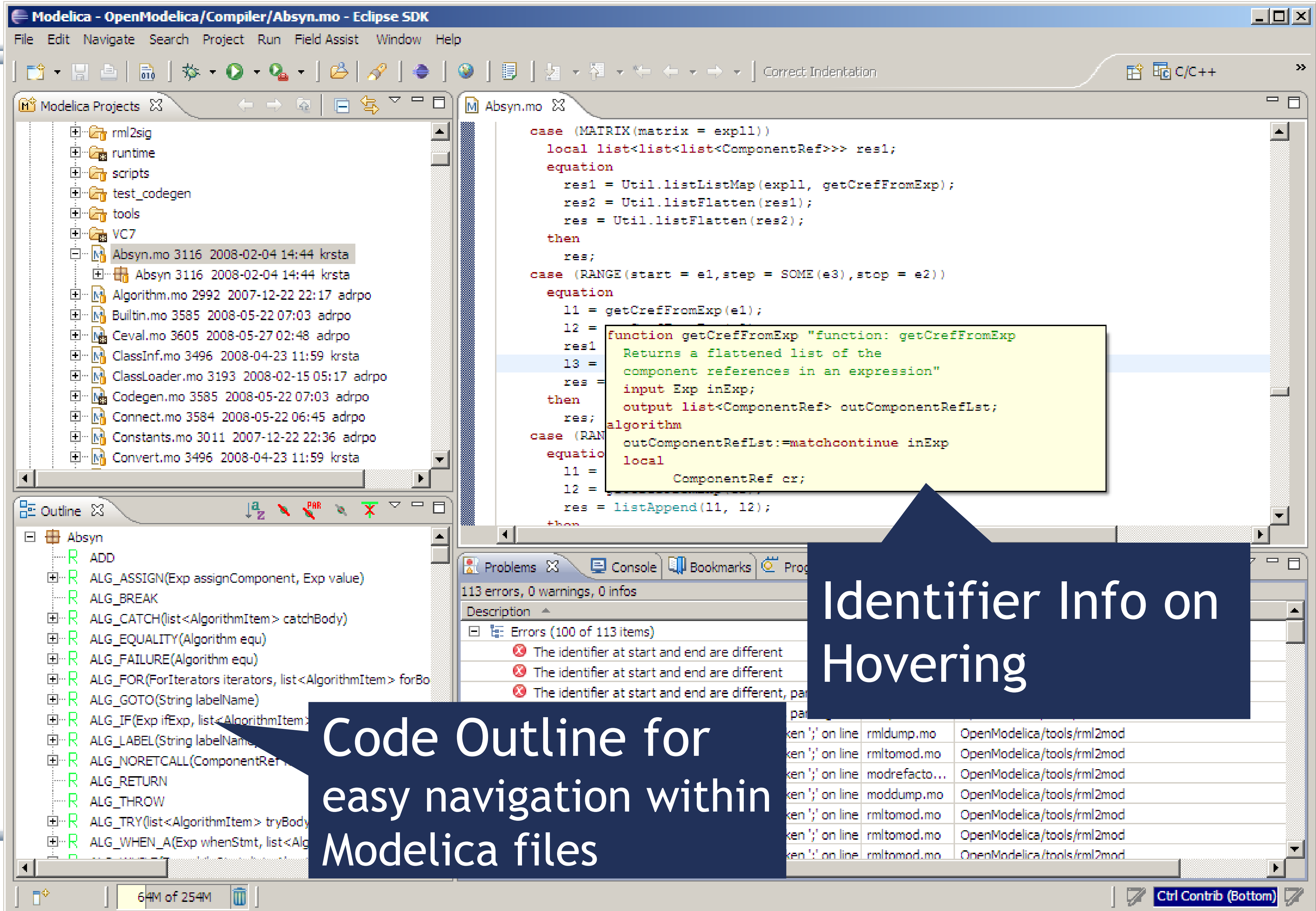
- Austrian Inst. of Tech, Austria
- UC Berkeley, USA
- TU Berlin, Insti UEBB, Germany
- FH Bielefeld, Bielefeld, Germany
- TU Braunschweig, Germany
- Univ Calabria, Italy
- TU Dortmund, Germany
- TU Dresden, Germany
- Université Laval, Canada
- Ghent University, Belgium
- Halmstad University, Sweden
- 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

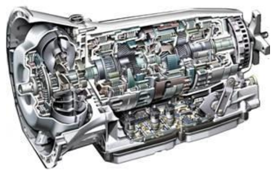


OpenModelica Eclipse MDT: Code Outline and Hovering Info

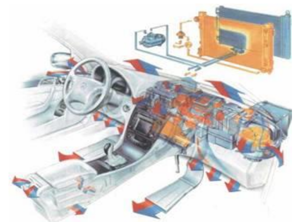


General Tool Interoperability & Model Exchange
Functional Mock-up Interface (FMI)

Engine with ECU

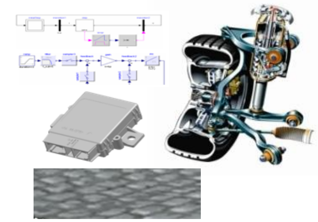


Gearbox with ECU



Thermal systems

Automated cargo door



Chassis components, roadway, ECU (e.g. ESP)

etc.

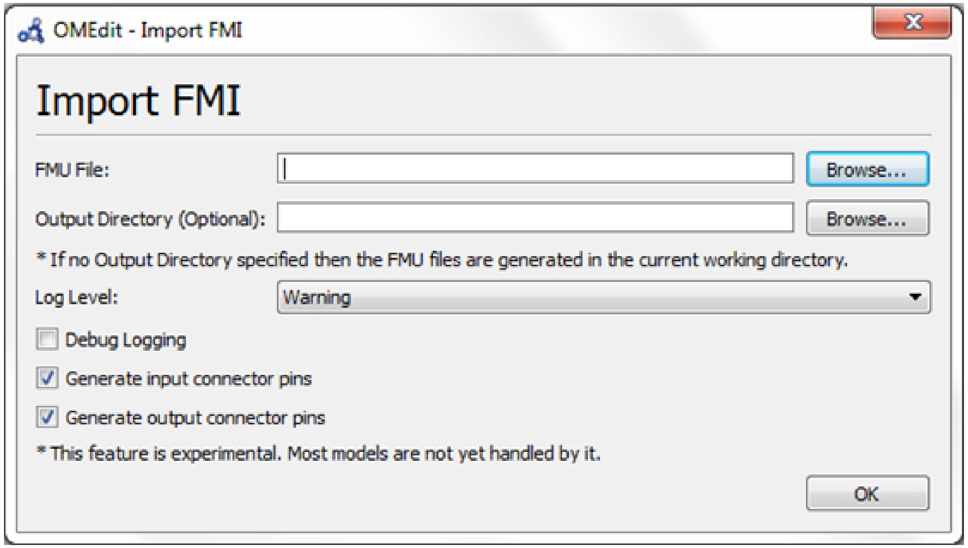
functional mockup interface for model exchange and tool coupling

courtesy Daimler

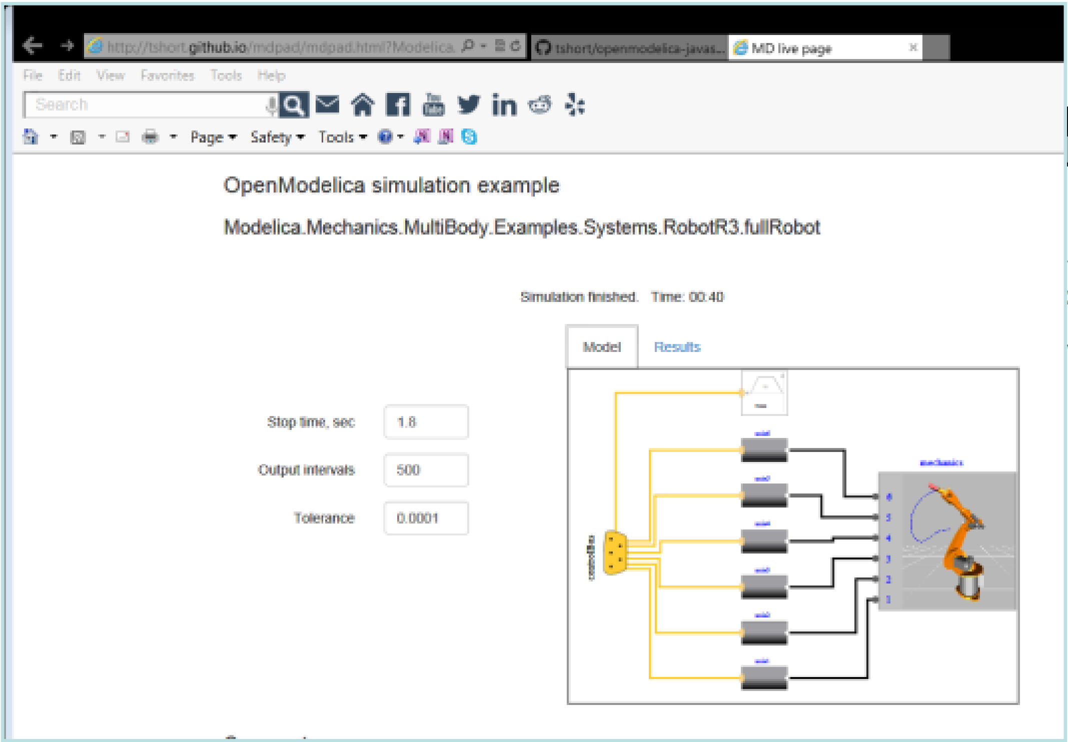
- FMI development was started by ITEA2 MODELISAR project. FMI is a Modelica Association Project now
- **Version 1.0**
- FMI for Model Exchange (released Jan 26,2010)
- FMI for Co-Simulation (released Oct 12,2010)
- **Version 2.0**
- FMI for Model Exchange and Co-Simulation (released July 25,2014)
- **>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

FMI in OpenModelica

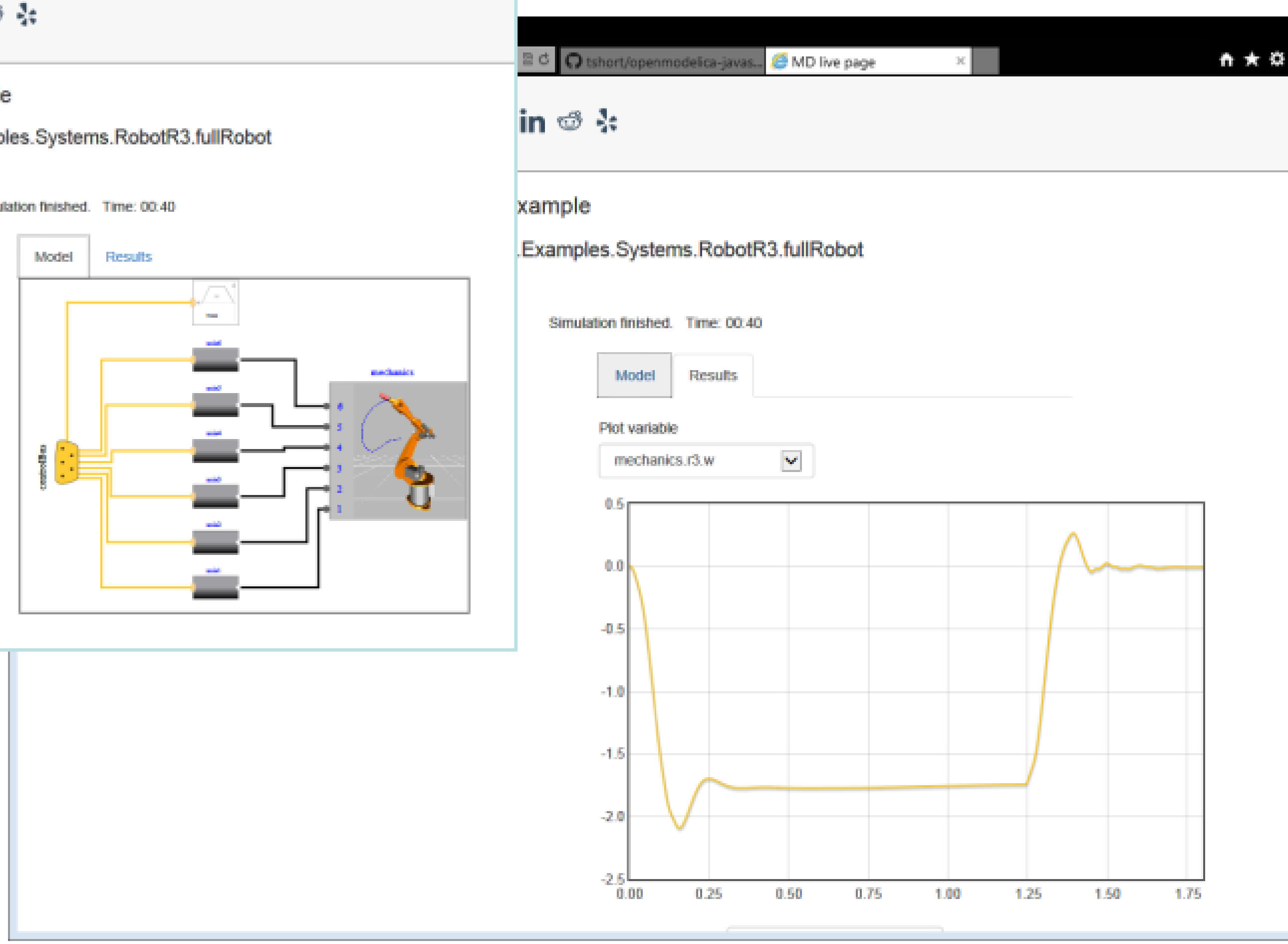
- FMI Model Exchange implemented (FMI 1.0 and FMI 2.0)
- A prototype of FMI 2.0 co-simulation is available
- Ongoing work to support full FMI 2.0 co-simulation
- The FMI interface is accessible via the **OpenModelica scripting environment** and the **OpenModelica connection editor**



OpenModelica Simulation in Web Browser Client



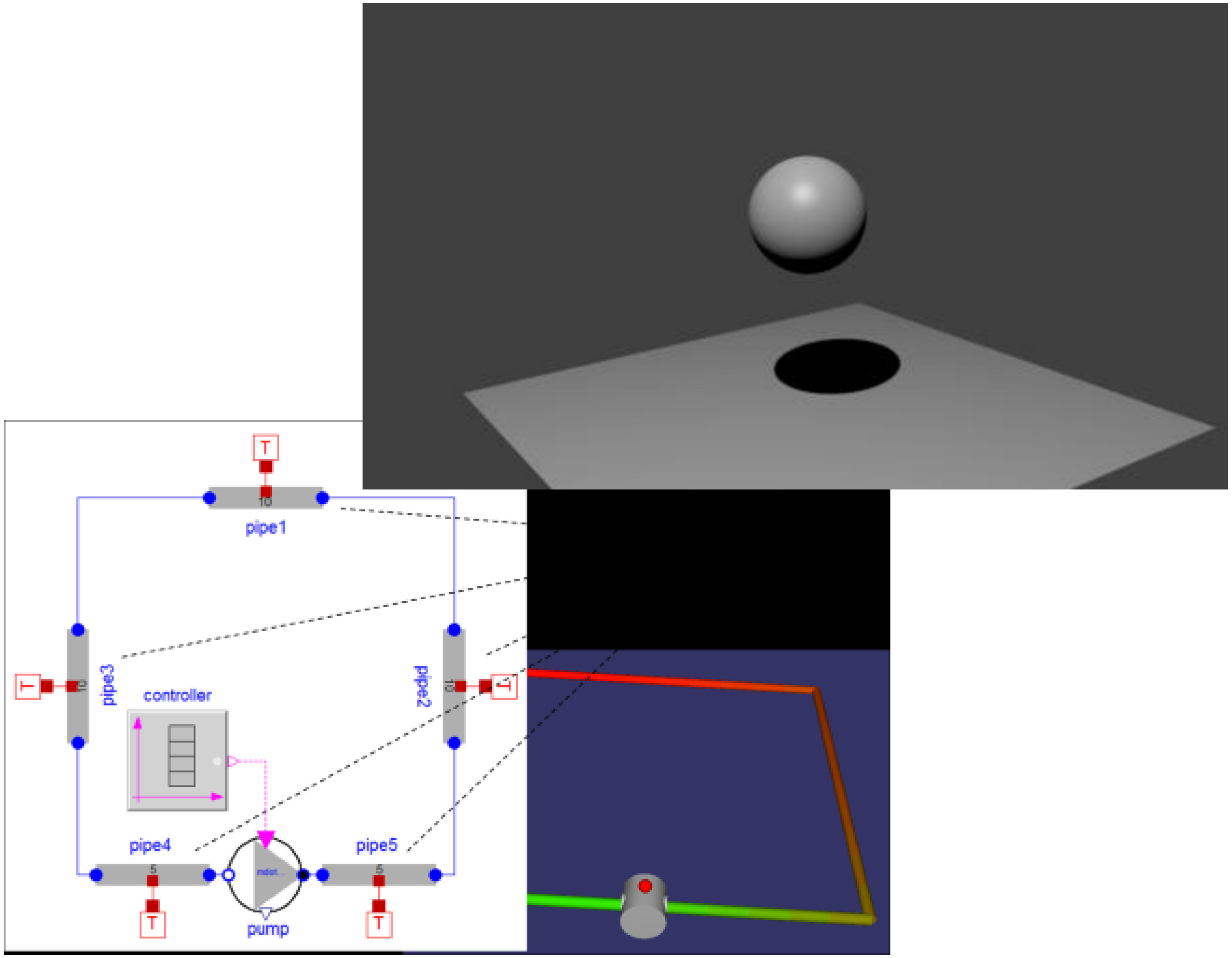
MultiBody RobotR3.FullRobot



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



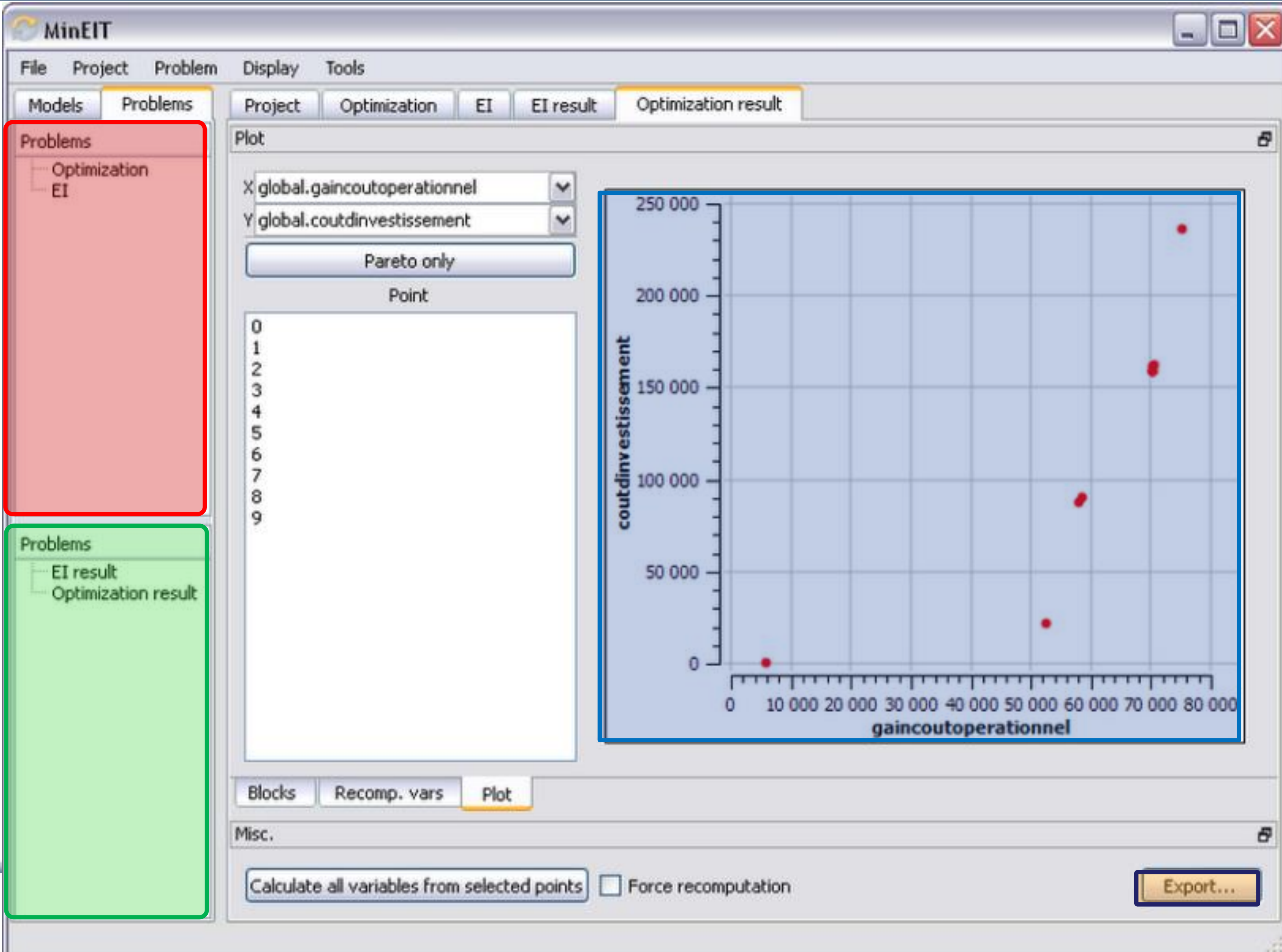
OMOptim – Parameter Sweep Design Optimization

Problems

Solved problems

Result plot

Export result data .csv



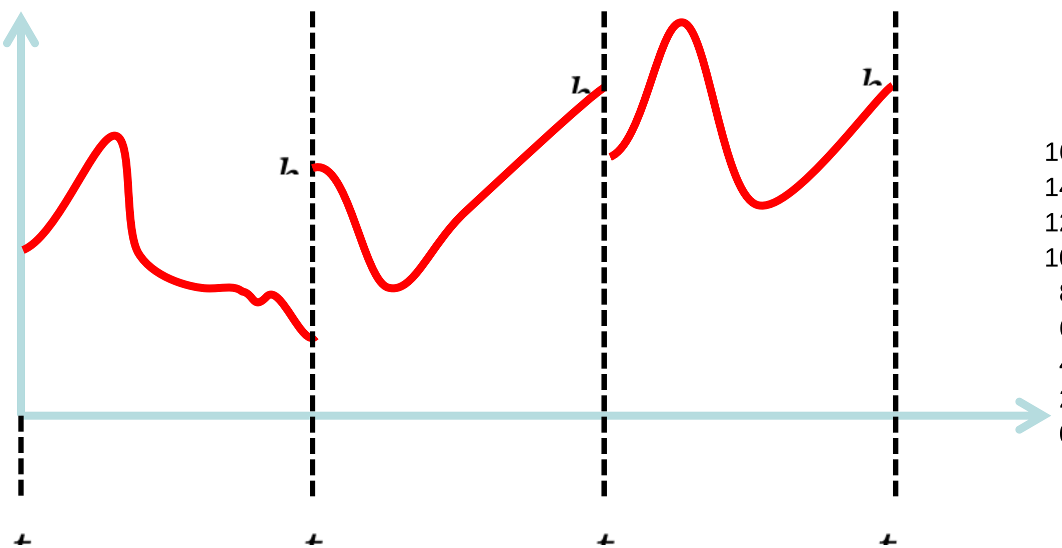
Here Pareto front optimiza-tion

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

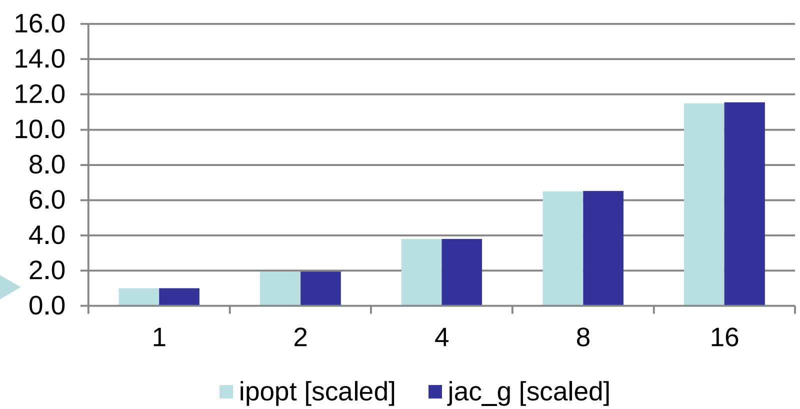
In OpenModelica 1.9.1 beta release Jan 2014.

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

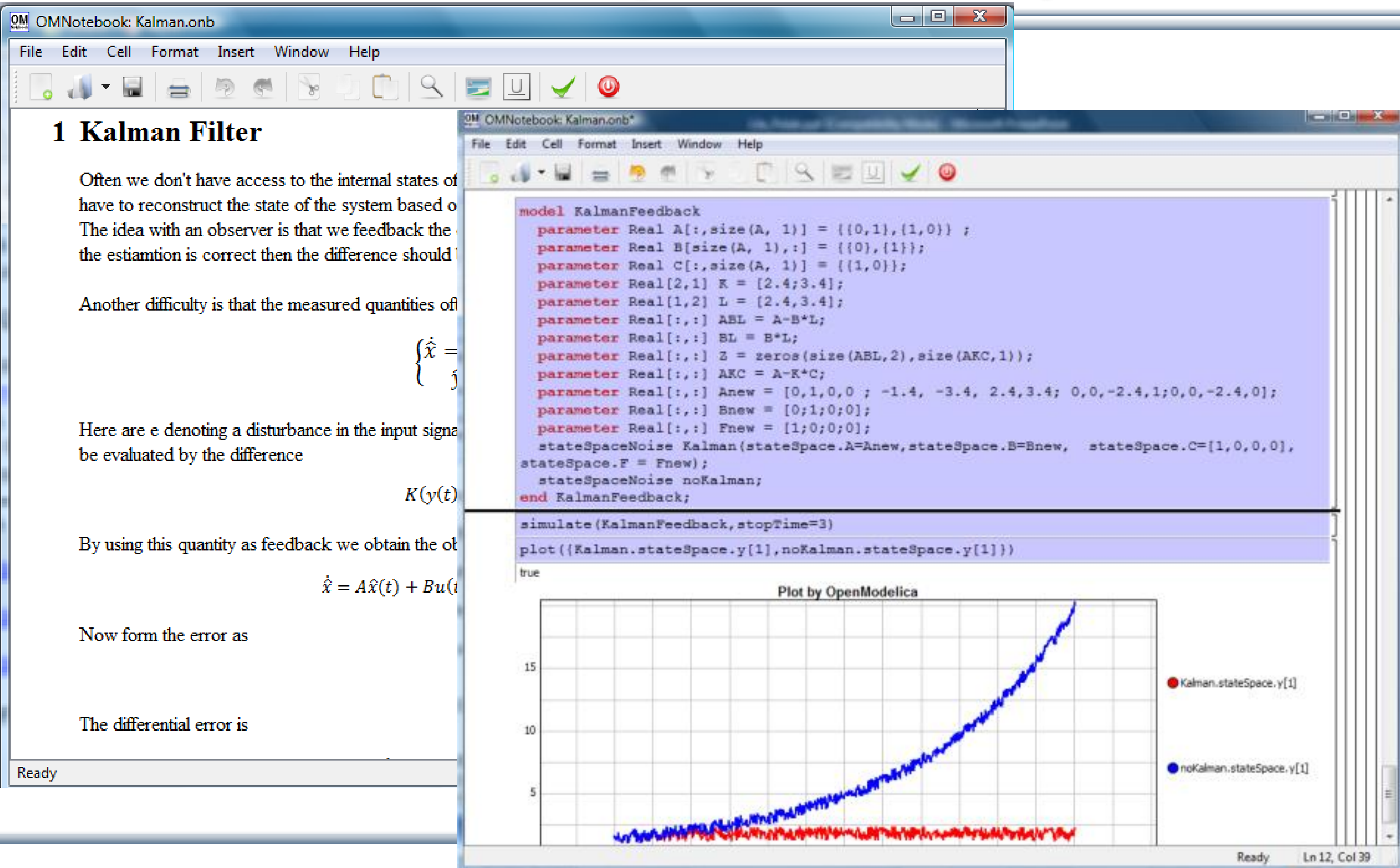


Example speedup, 16 cores:

MULTIPLE_COLLOCATION



OMnotebook Interactive Electronic Notebook Here Used for Teaching Control Theory

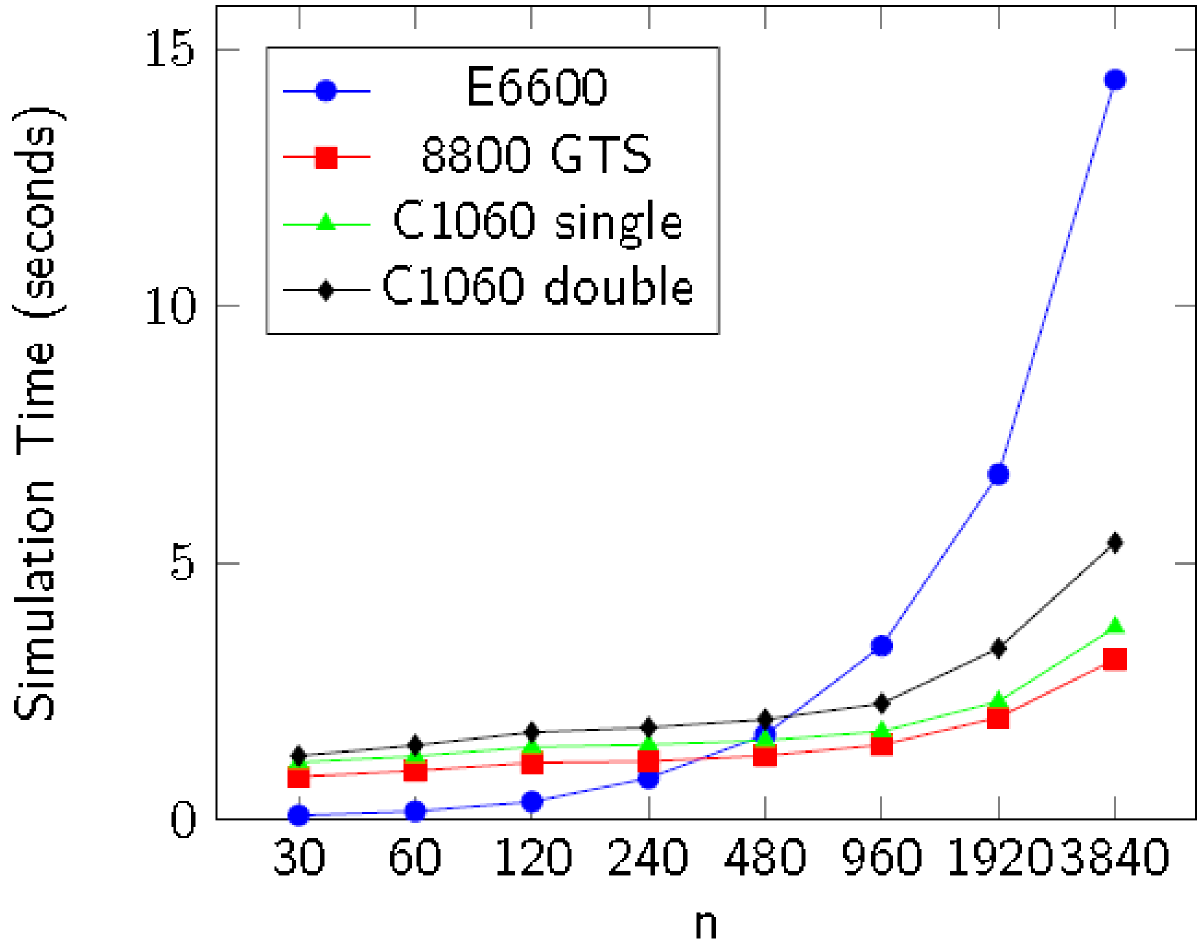


MetaModelica Language Extension for Model Transformations and Advanced Applications

- **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

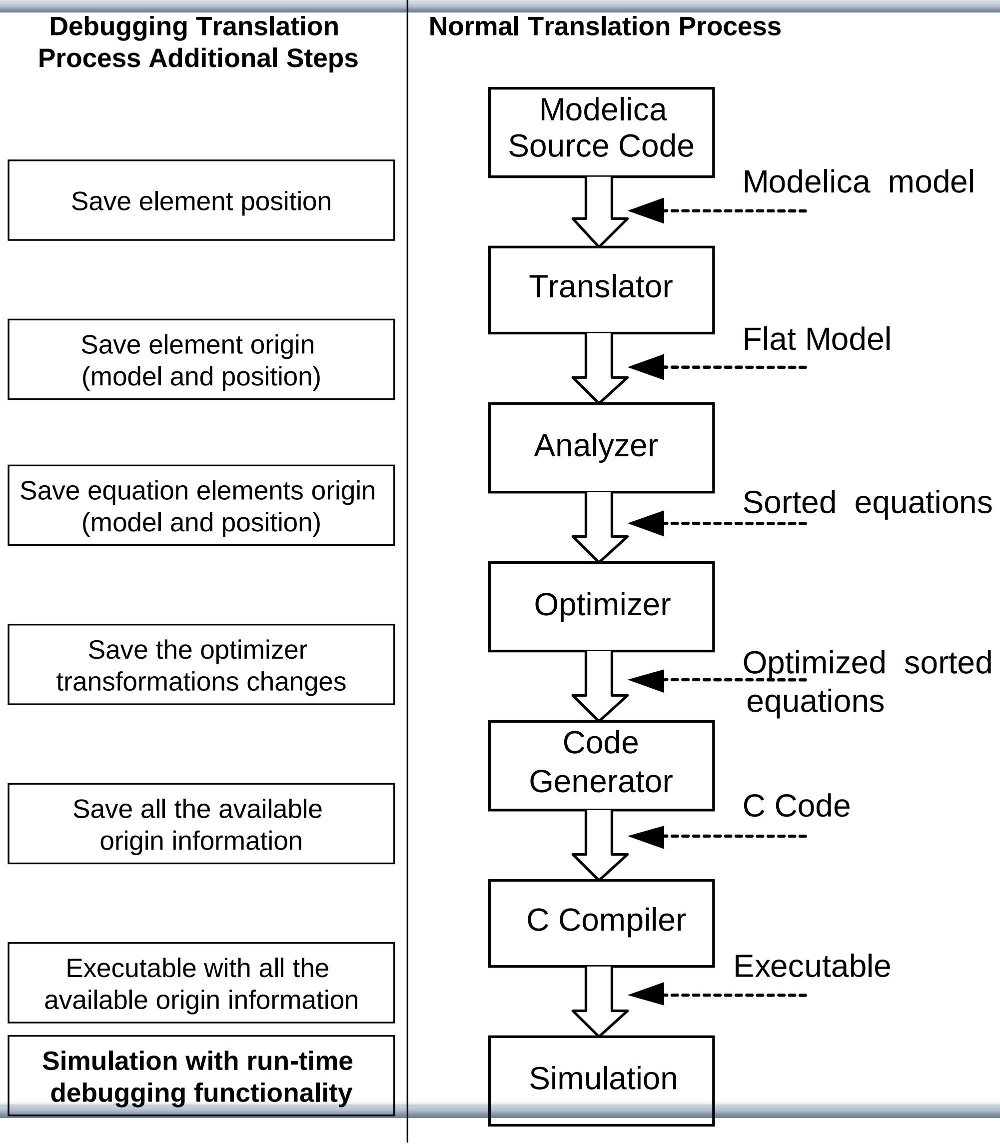
    residual[1] = 17.000400

    x[1] = 1.248448

    ...

Model Compiler Translation Phases Extended with Debugging

- Include debugging support within the translation process



Example Symbolic Transformations with Compiler Debug Trace

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

Example: 0 = y + der(x * time * z); z = 1.0;

(1) substitution:

y + der(x * (time * z))

=>

y + der(x * (time * 1.0))

(2) simplify:

y + der(x * (time * 1.0))

=>

y + der(x * time)

(3) expand derivative (symbolic diff):

y + der(x * time)

=>

y + (x + der(x) * time)

(4) solve:

0.0 = y + (x + der(x) * time)

=>

der(x) = ((-y) - x) / 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

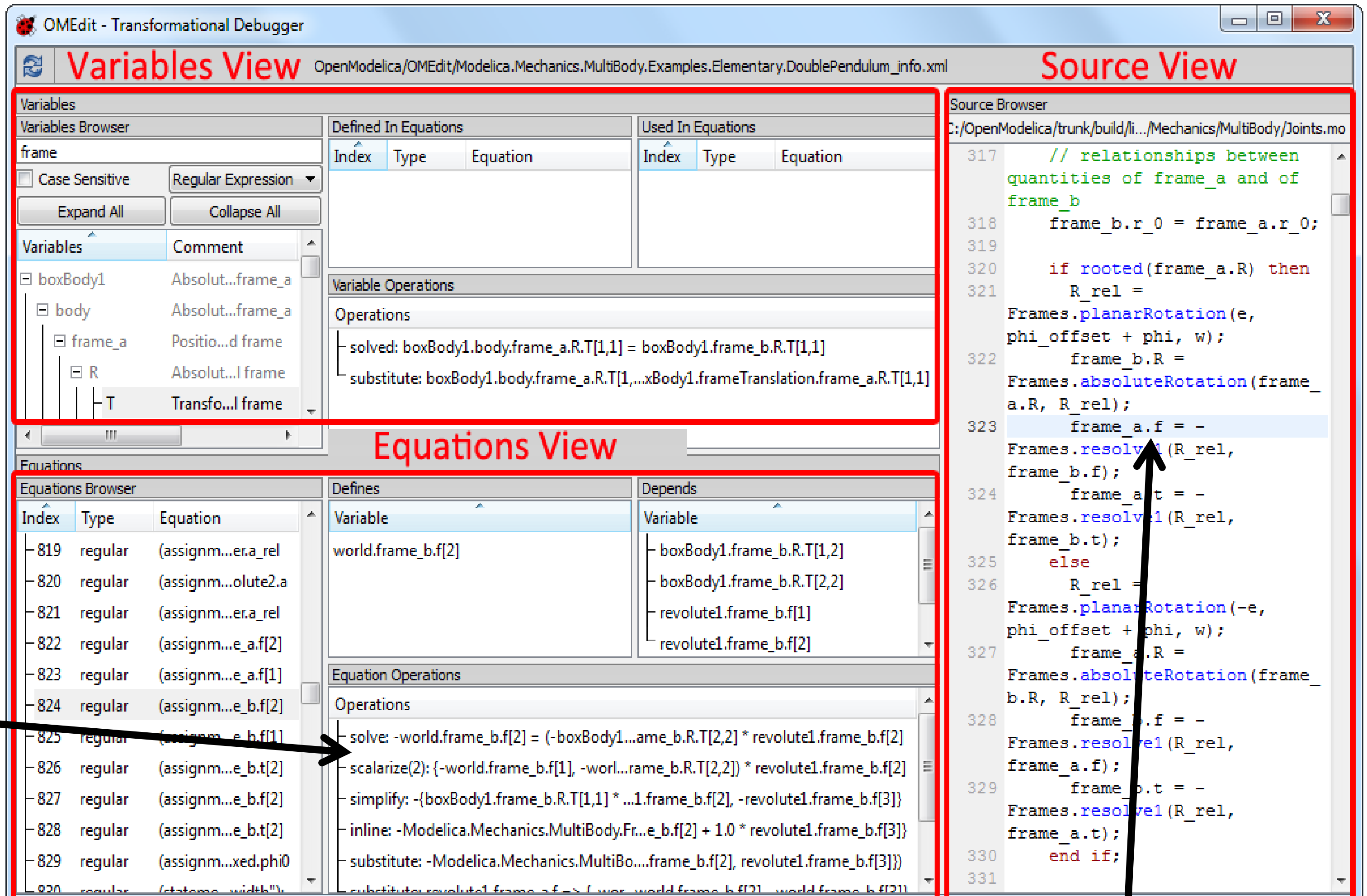
MSL 3.1 MultiBody DoublePendulum

# 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 occurances of aliased vars)

Integrated Static-Dynamic OpenModelica Equation Model Debugger

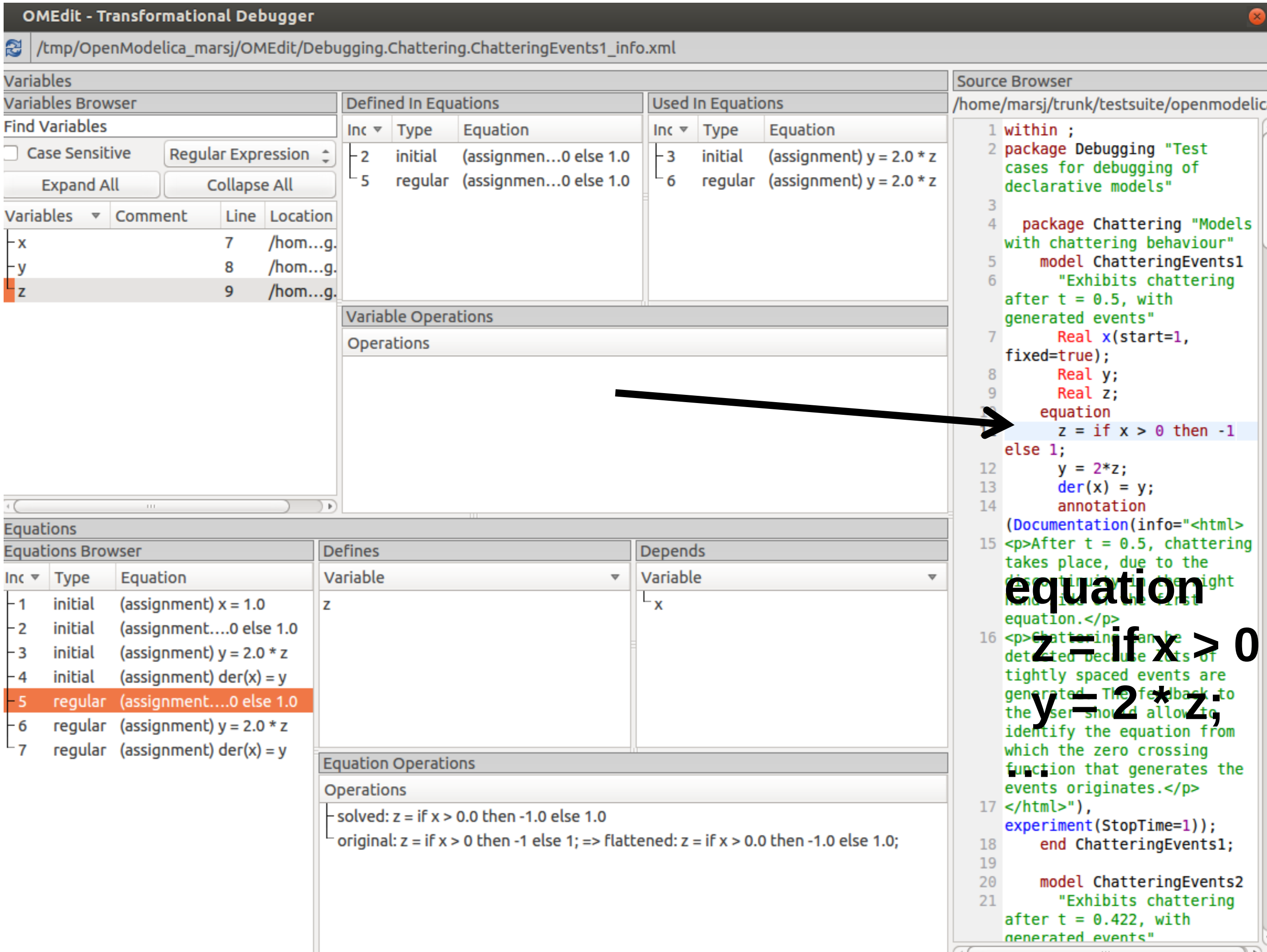
Efficient handling of **Large** Equation Systems

Showing equation transformations of a model:



Mapping dynamic run-time error to source model position

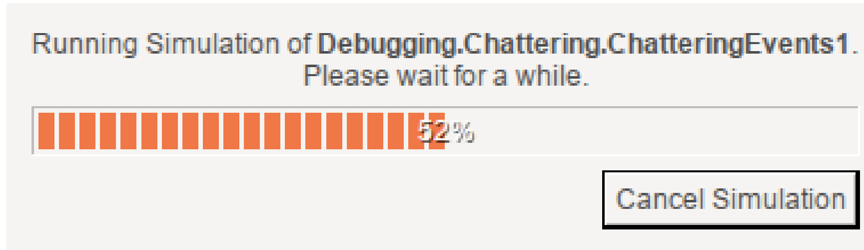
Example – Detecting Source of Chattering (excessive event switching) causing bad performance

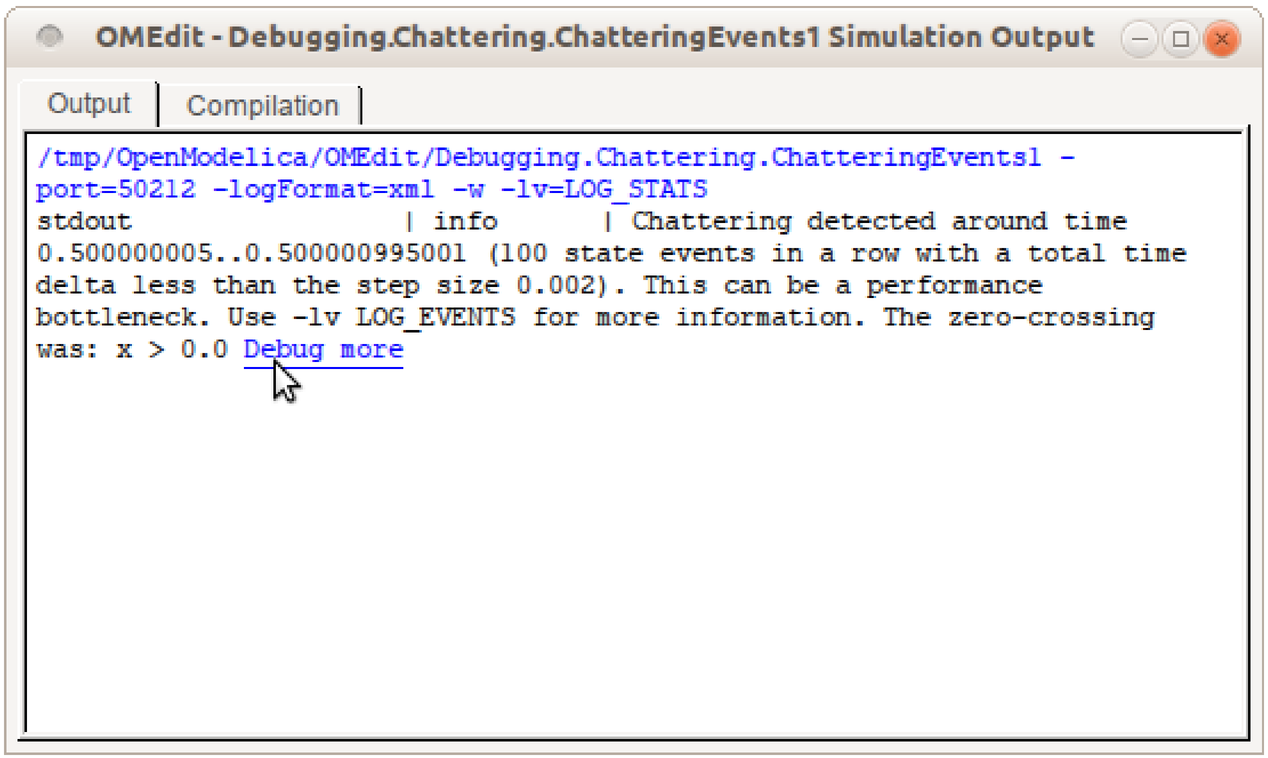



```
equation
  z = if x > 0 then -1 else 1;
  y = 2 * z;
  …
```

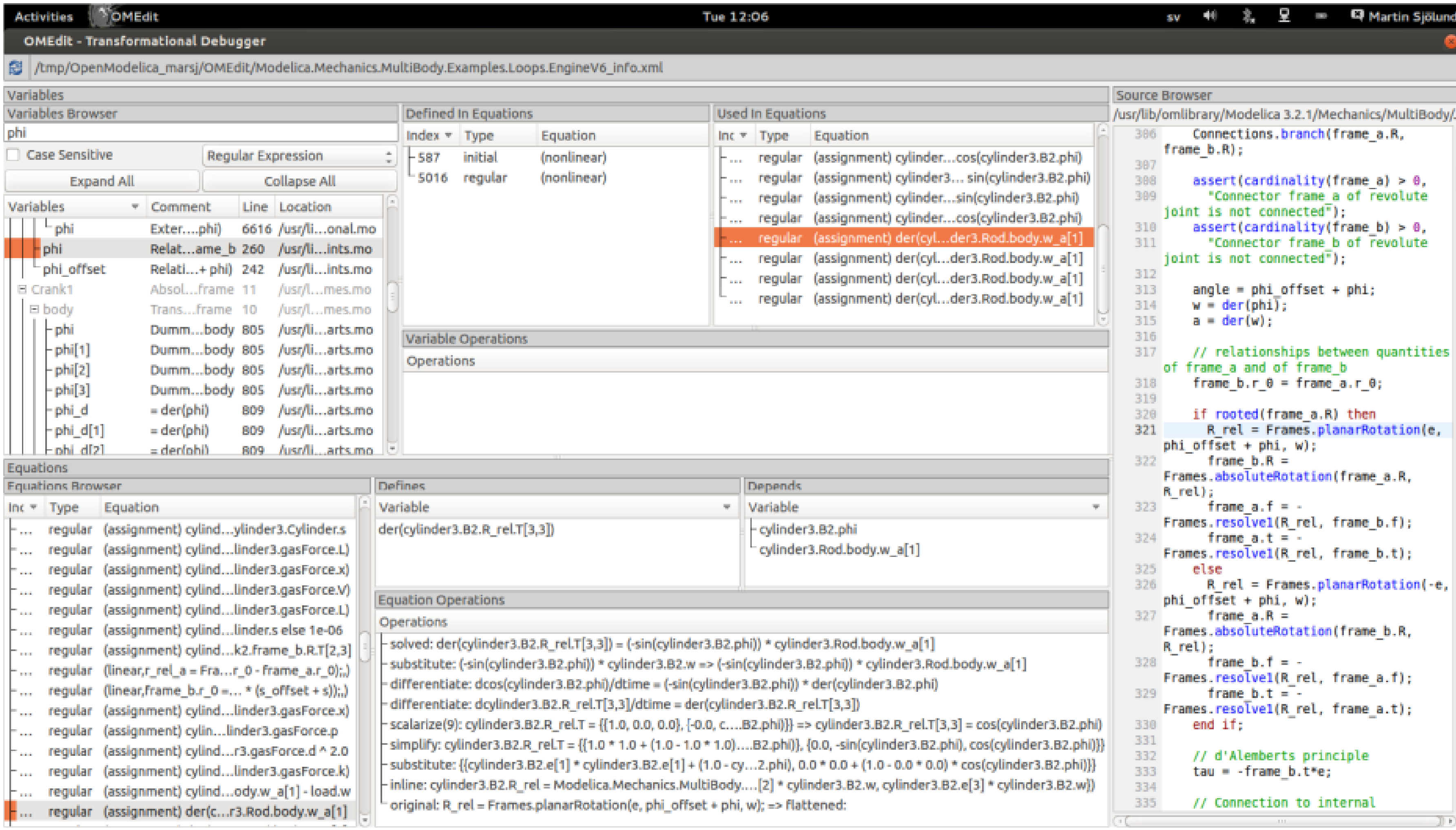

MODELICA

Error Indication – Simulation Slows Down





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



MODELICA

Browsing Equation Transformation Chains
Closeup of EngineV6 Equations

Defines	Depends
Variable	Variable
der(cylinder3.B2.R_rel.T[3,3])	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: dcylinder3.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:
```


MODELICA

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

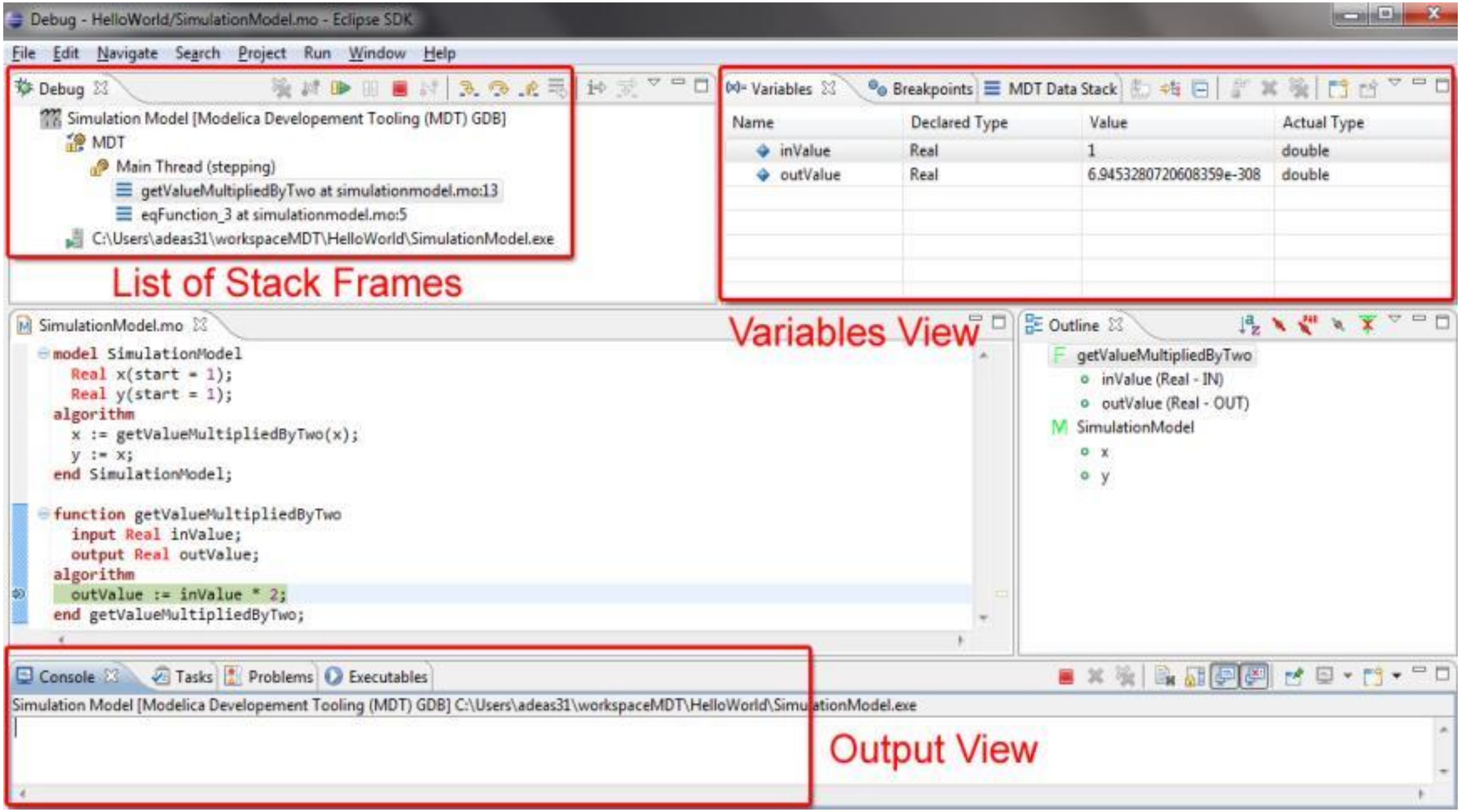
Equations Browser

Index	Type	Equation	Executi	Max time	Time	Fraction
876	regular	linear, size 2	4602	0.000501	0.0134	75.7%
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%

Defines

Variable
damper.a_rel
revolute2.frame_b.f[2]

OpenModelica Eclipse MDT Algorithmic Code Debugger



MODELICA

Adding Breakpoints

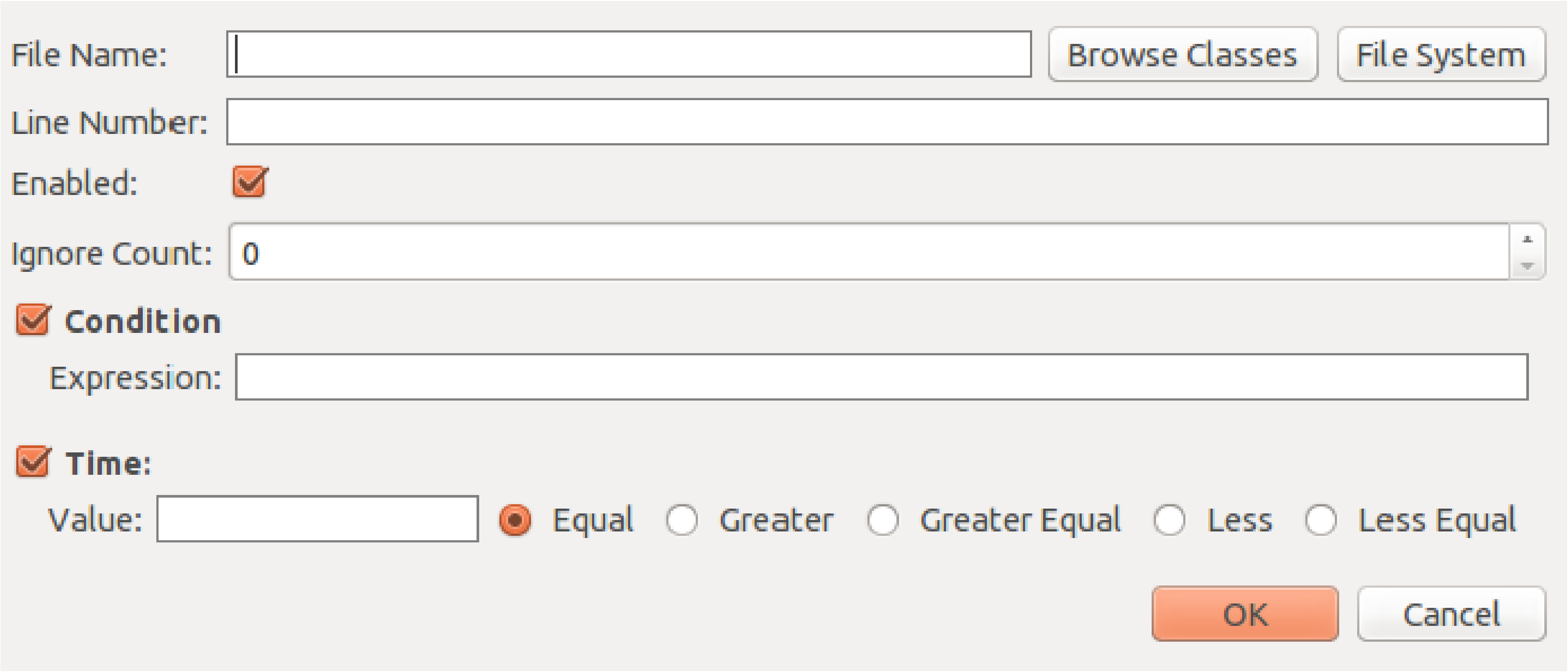
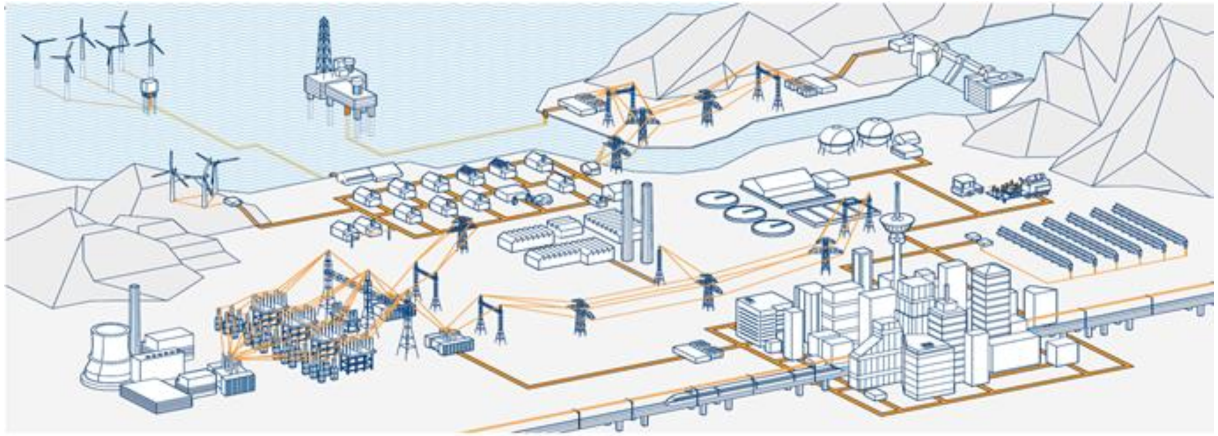


ABB Commercial Application Use of Debugger

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

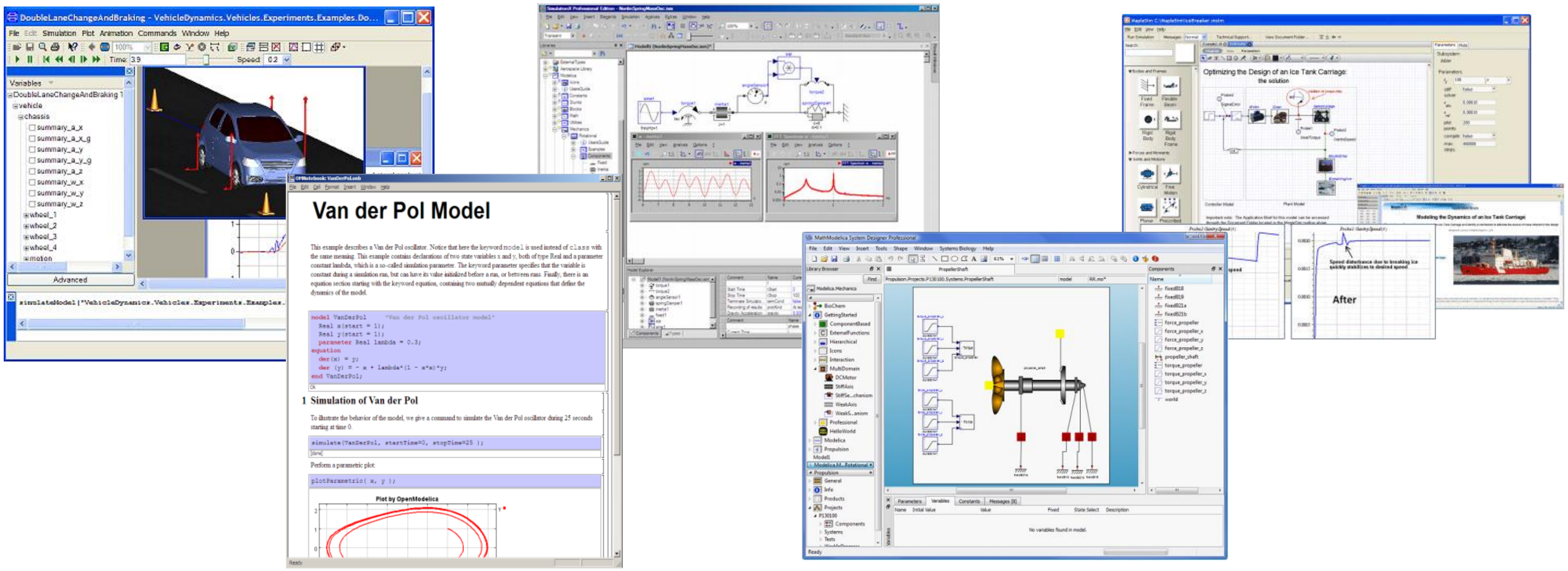


- ABB: “OpenModelica provides outstanding debugging features that help to save a lot of time during model development.”

Part III

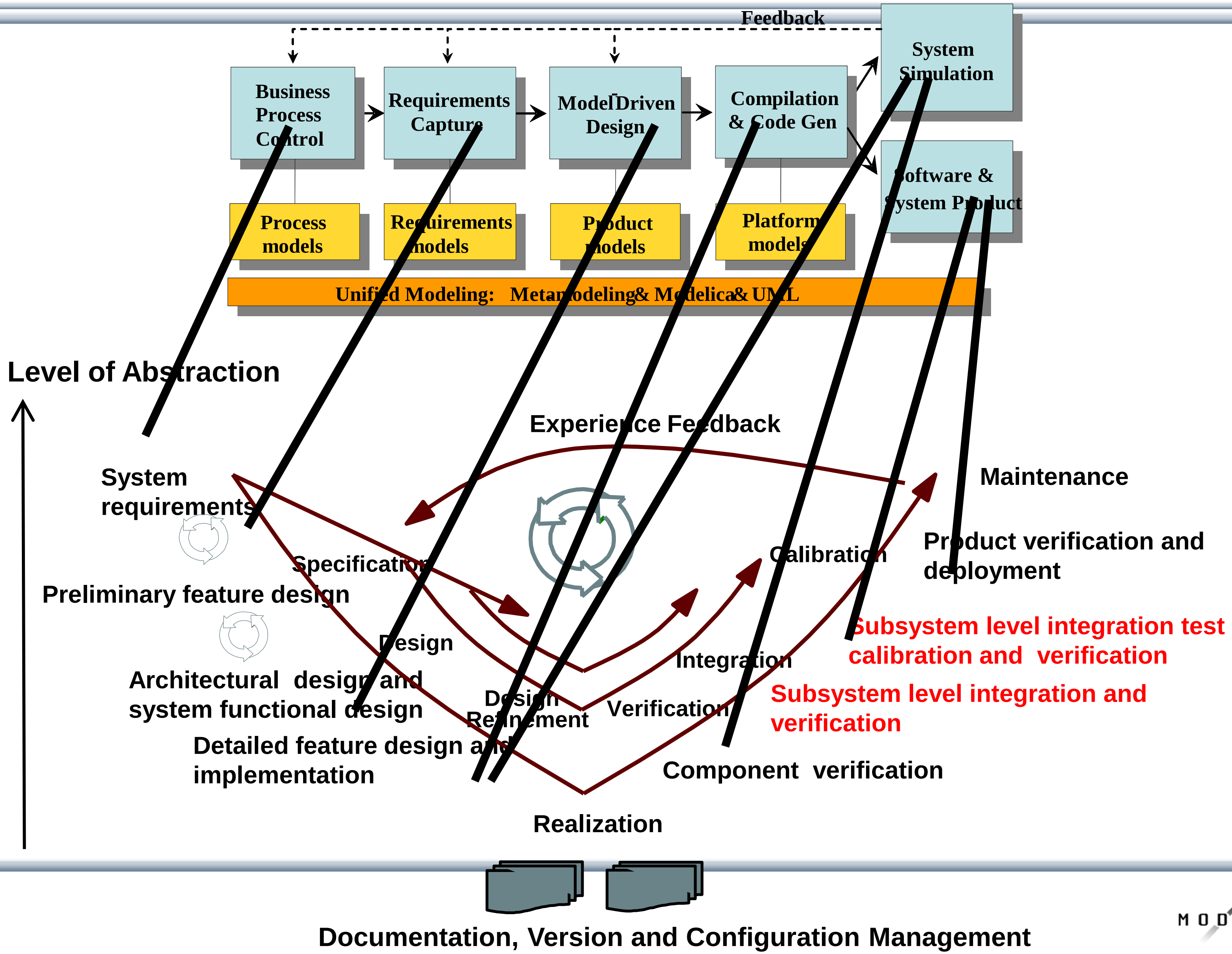
Dynamic Verification/Testing of Requirements vs Usage Scenario Models

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

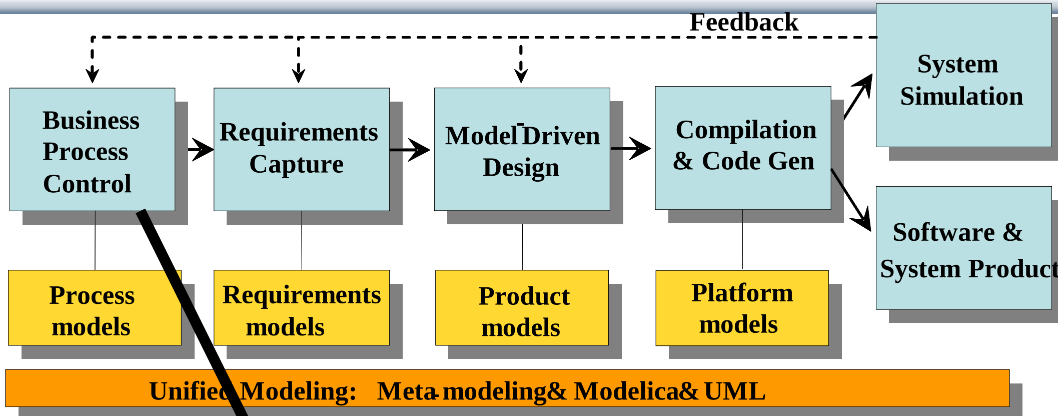


MODELICA

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



Business Process Control and Modeling



VTT Simantics
Business process modeler

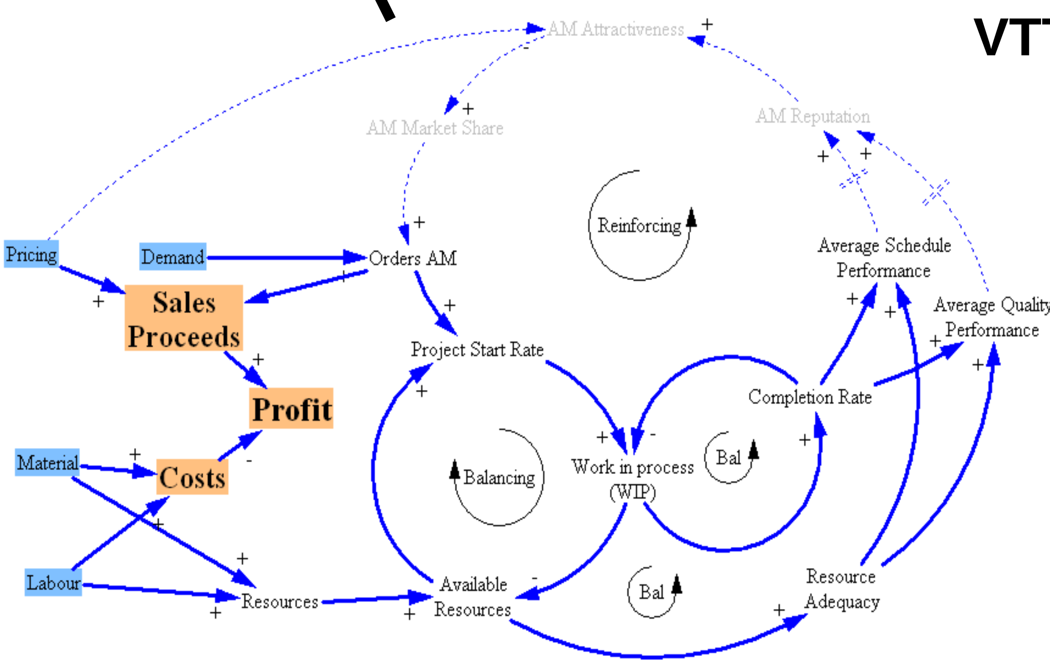
OpenModelica
compiler & simulator

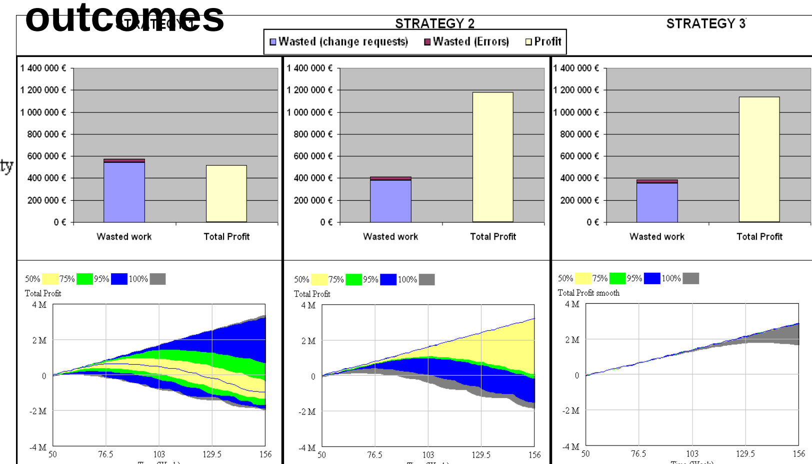
OpenModelica based simulation

Metso Business model & simulation

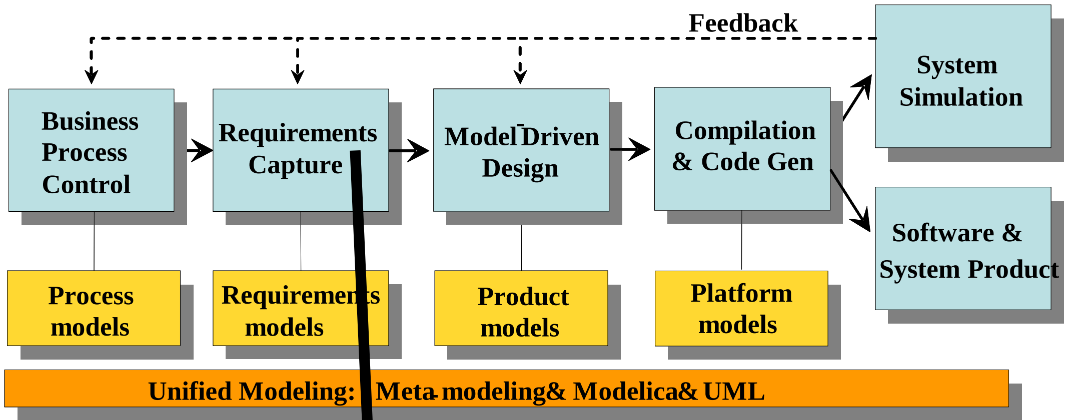
VTT Simantics Graphic Modeling Too

Simulation of 3 strategies with outcomes





Requirement Capture



vVDR (virtual Verification of Designs against Requirements)

in ModelicaML UML/Modelica Profile, part of OpenModelica

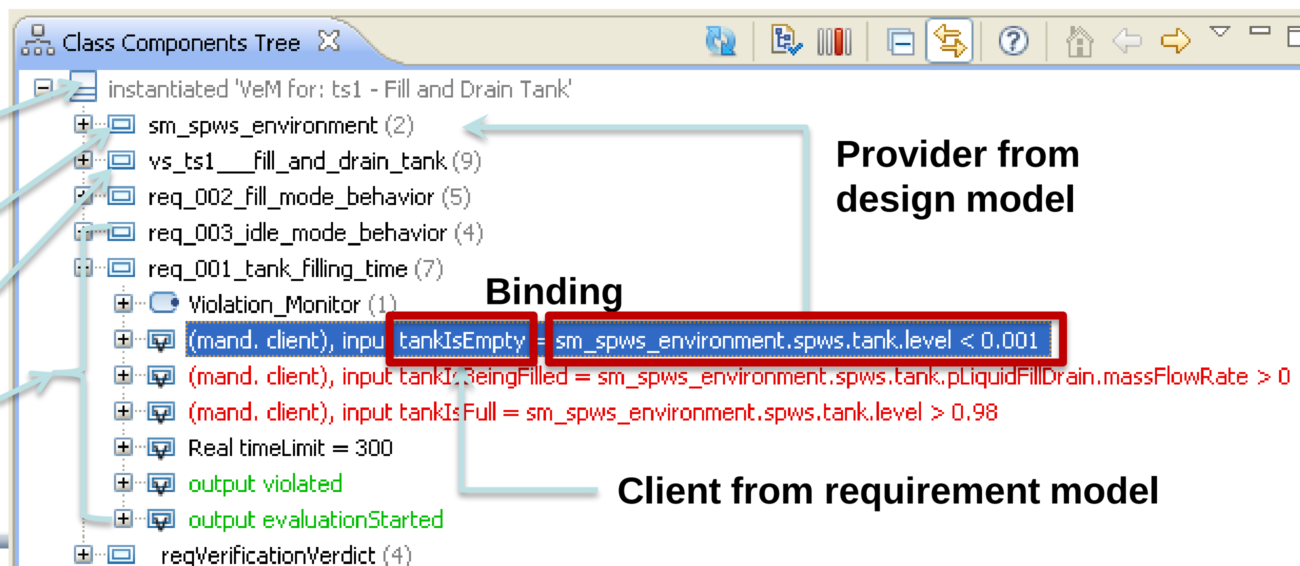
OpenModelica based simulation

Verification Model

Design Model

Scenario Model

Requirement Models



Binding

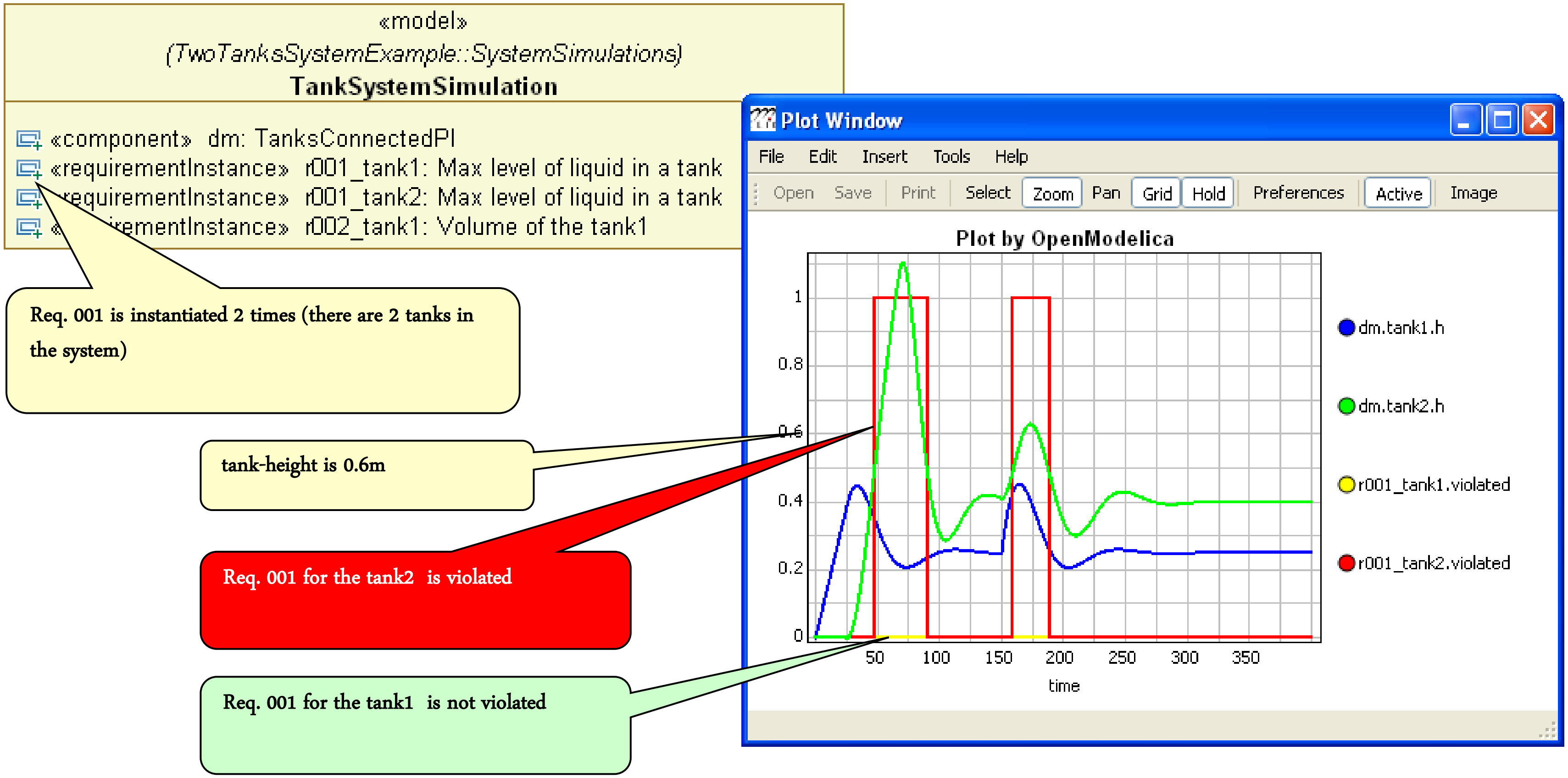
Provider from design model

Client from requirement model

OpenModelica – ModelicaML UML Profile
Based on Open-Source Papyrus UML and OpenModelica

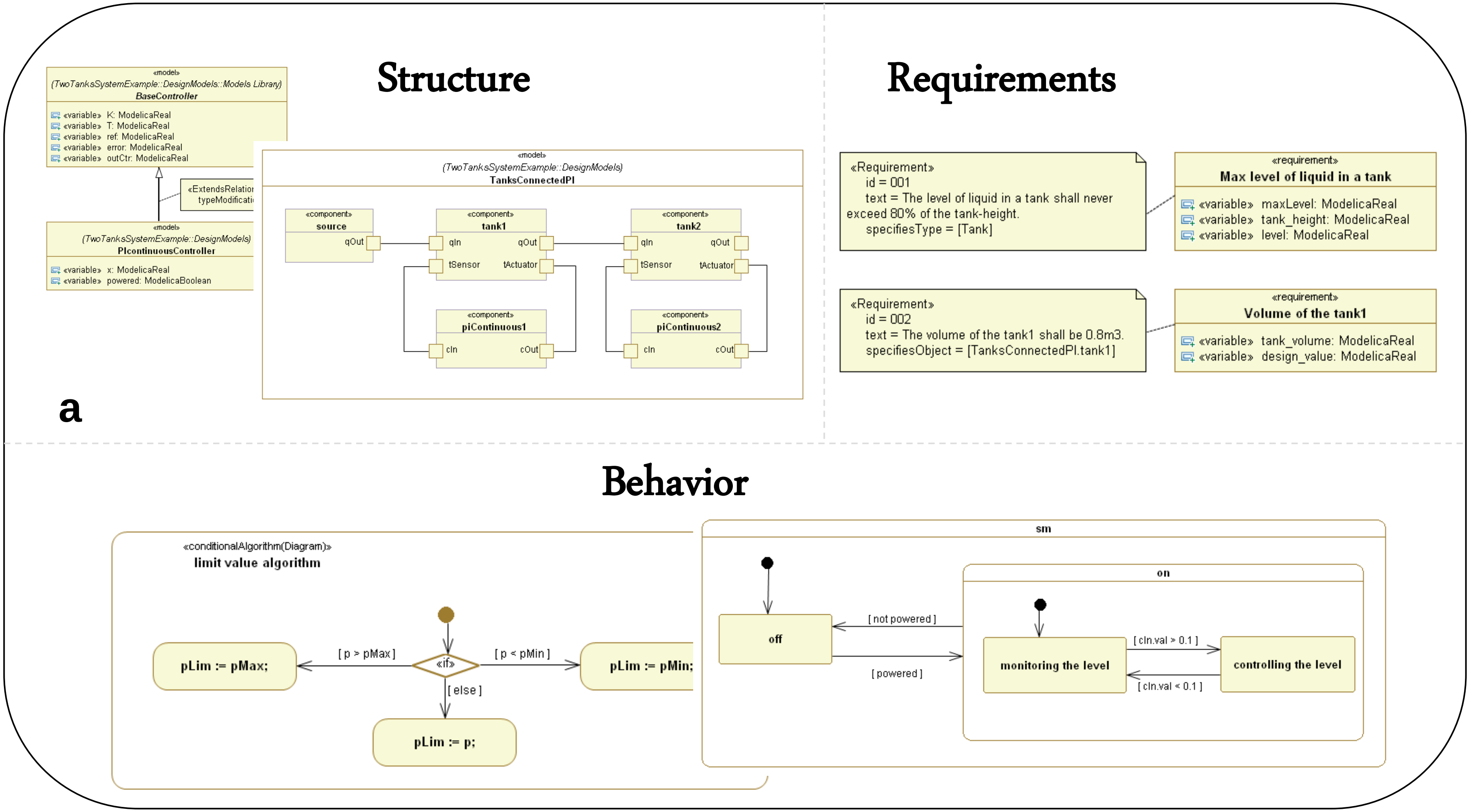
- ModelicaML is a UML Profile for SW/HW modeling
  - Applicable to "pure" UML or to other UML profiles, e.g. SysML
- Standardized Mapping UML/SysML to Modelica
  - Defines transformation/mapping for **executable** models
  - Being **standardized** by OMG
- ModelicaML
  - Defines graphical concrete syntax (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

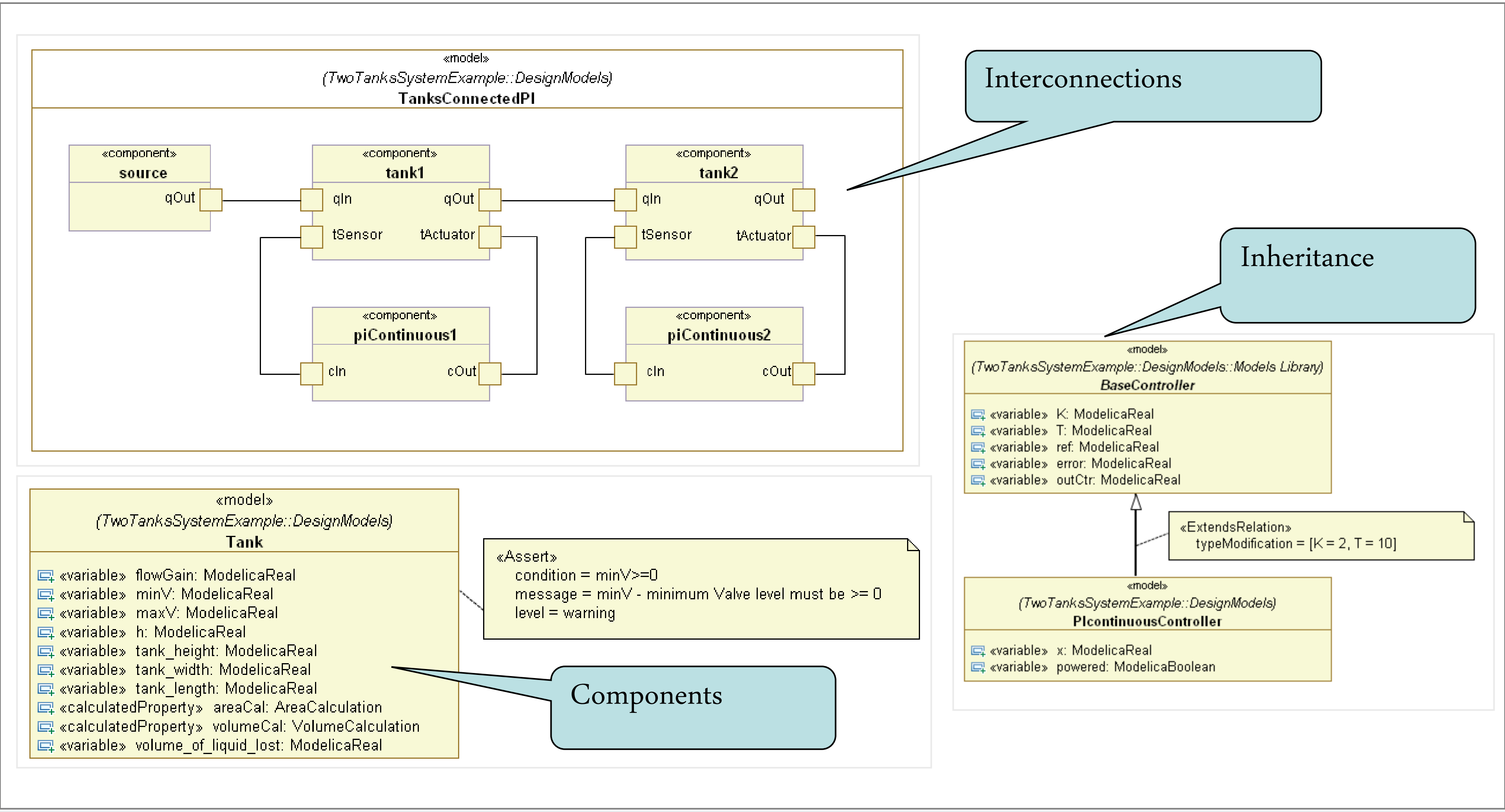


MODELICA

ModelicaML: Graphical Notation

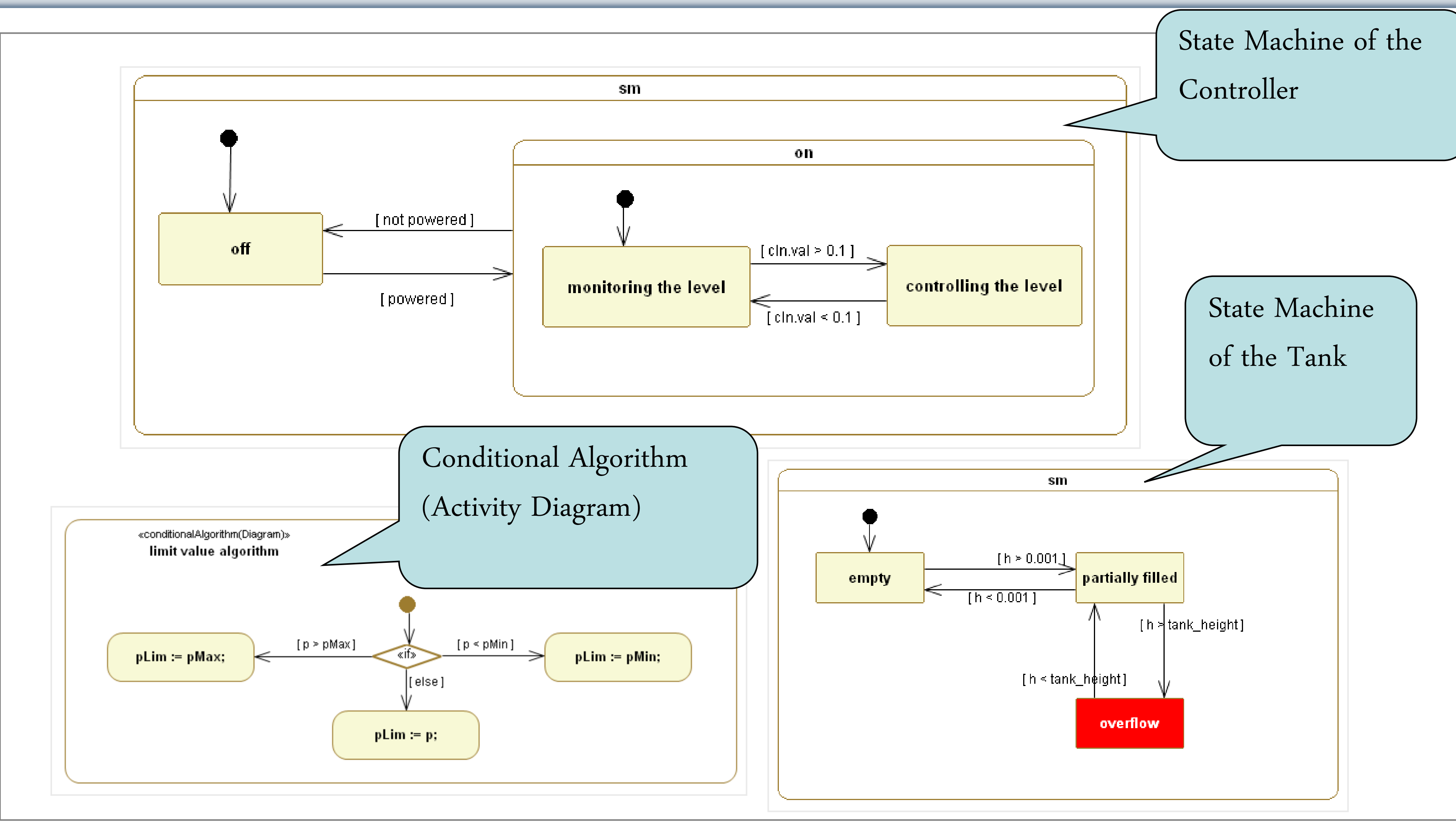


Example: Representation of System Structure

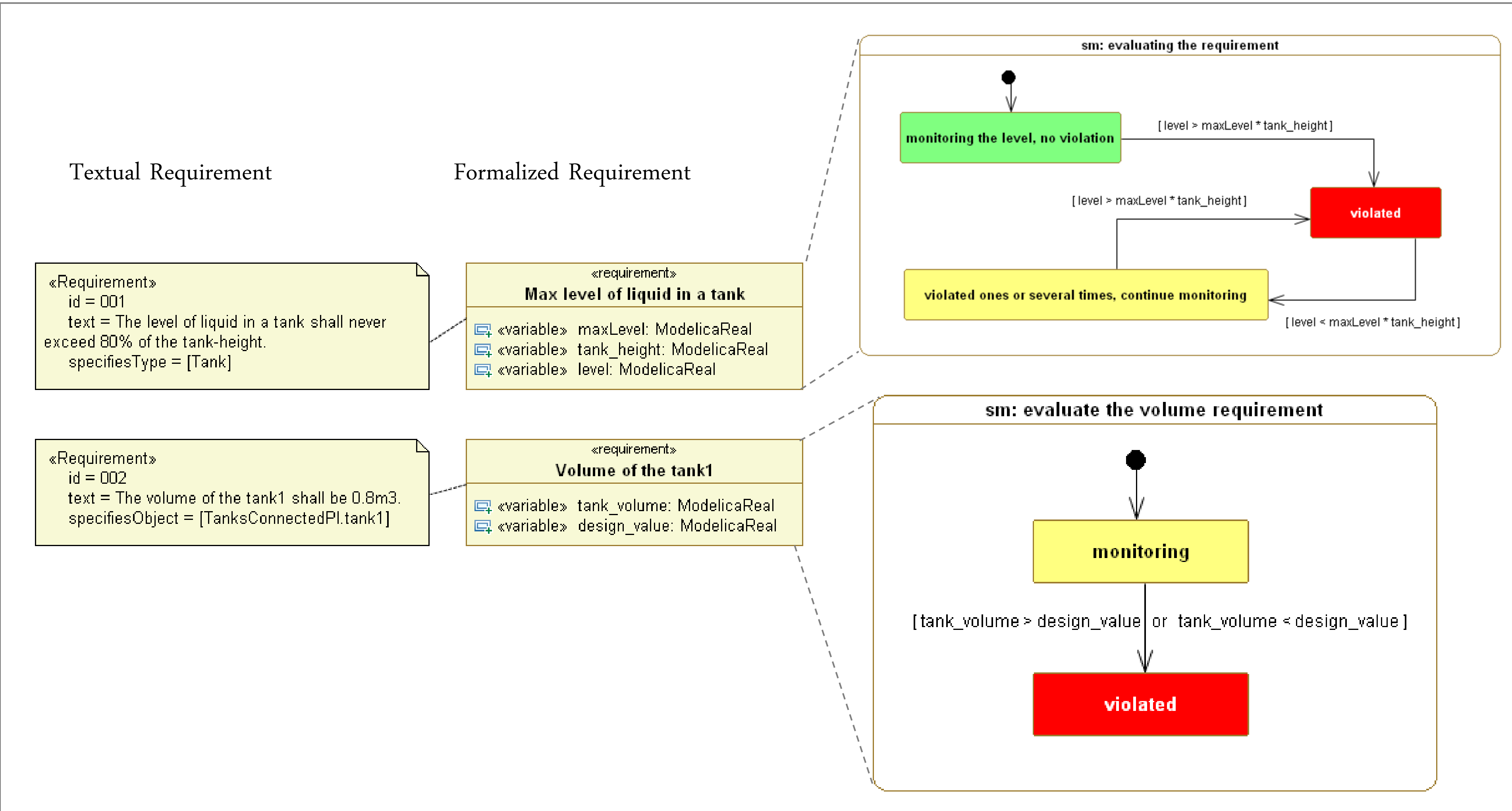


MODELICA

Example: Representation of System Behavior

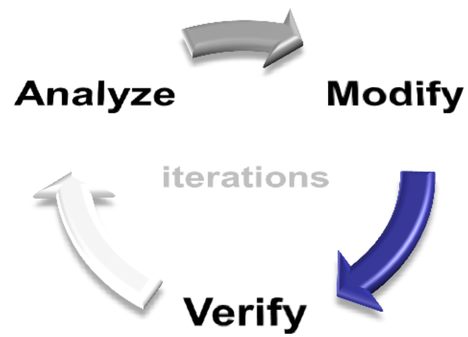


Example: Representation of System Requirements



vVDR Method –
virtual Verification of Designs vs Requirements

Actor	Task	Created Artifact
	Formalize Requirements	RMM Requirement Monitor Models
	Formalize Designs	DAM Designs Alternative Models
	Formalize Scenarios	SM Scenario Models
AUTOMATED	Create Verification Models *	VM Verification Models
AUTOMATED	Execute and Create Report	Reports
	Analyze Results	

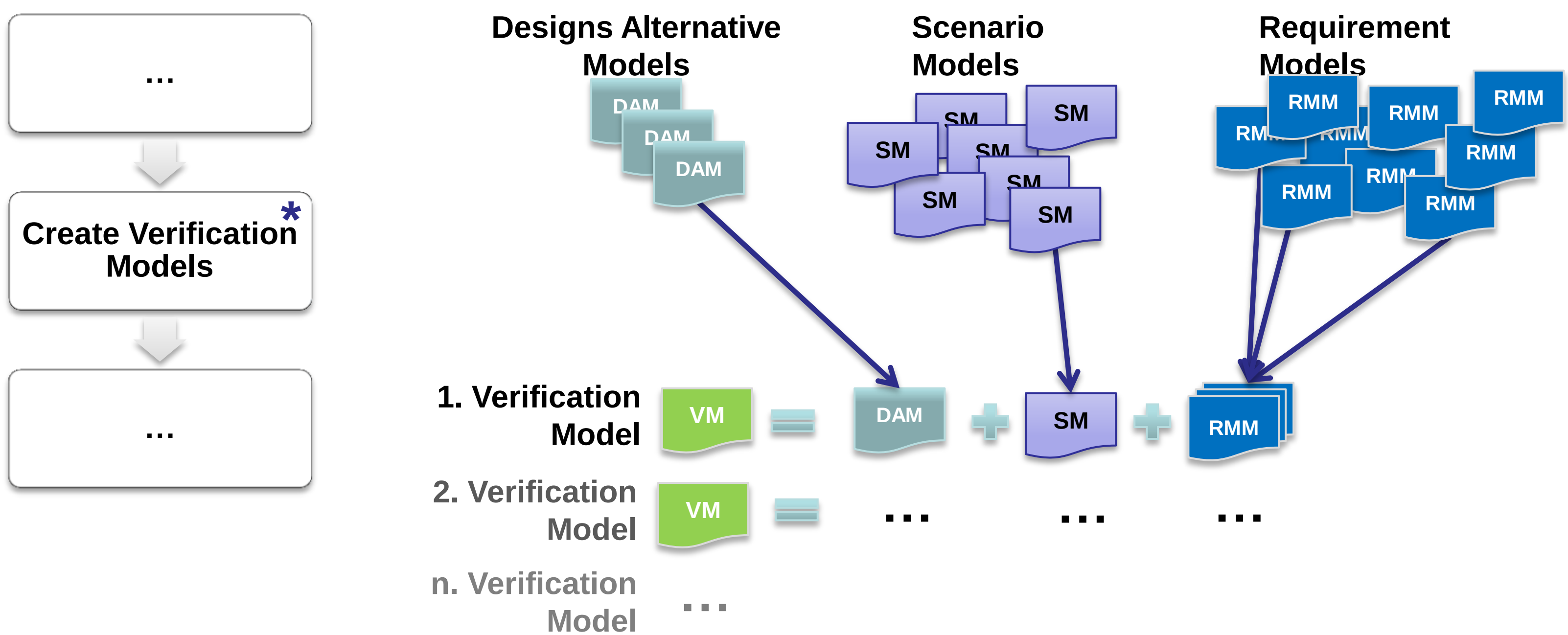


Goal: Enable on-demand verification of designs against requirements using automated model composition at any time during development.

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**?



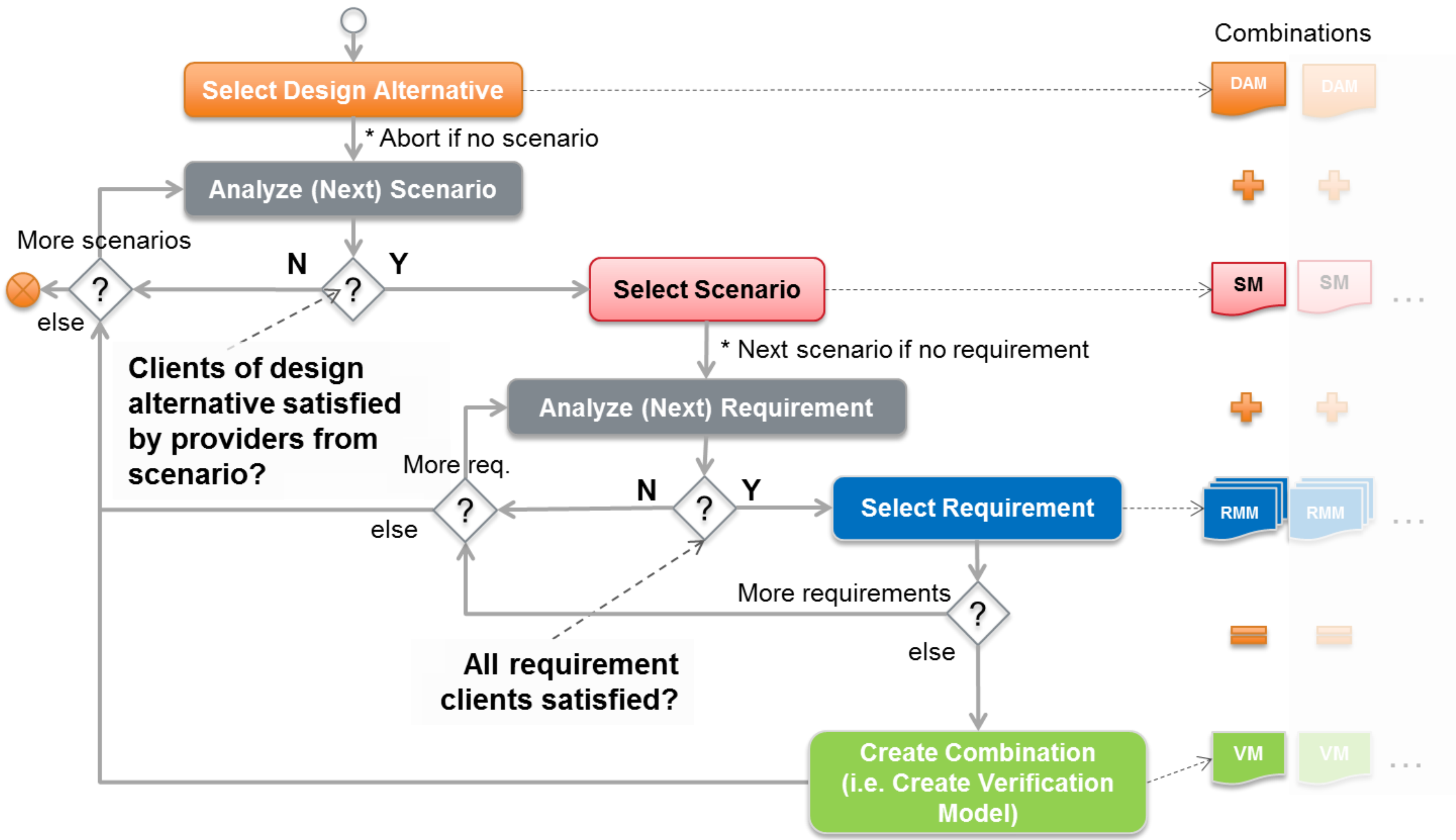
Composing Verification Models

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 or scenarios or requirements he/she wishes to consider

Generating/Composing Verification Models

algorithm

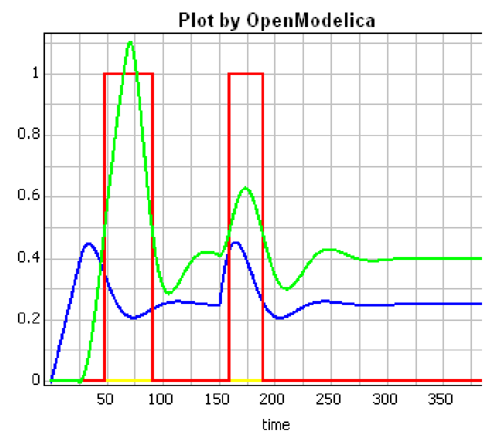


Simulation and Report Generation in ModelicaML

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)

madantory 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

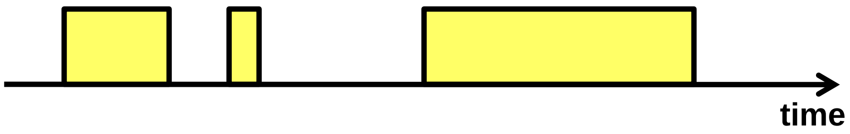
madantory client: req_004_drain_mode_behavior.fillDrainValveIsOpen (changed its value)
Type : = ModelicaBoolean
Variability : = continuous
Binding code : = sm_spws_environment.spws.fillDrainValve.isFullyOpen

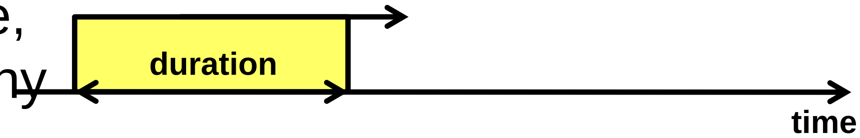
madantory client: req_004_drain_mode_behavior.otherValvesAreClosed (changed its value)
Type : = ModelicaBoolean
Variability : = continuous
Binding code : = if sm_spws_environment.spws.overFlowValve.isFullyClosed and sm_spws_environment.spws.supplyVavle.isFullyClosed then true else false

MODELICA

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

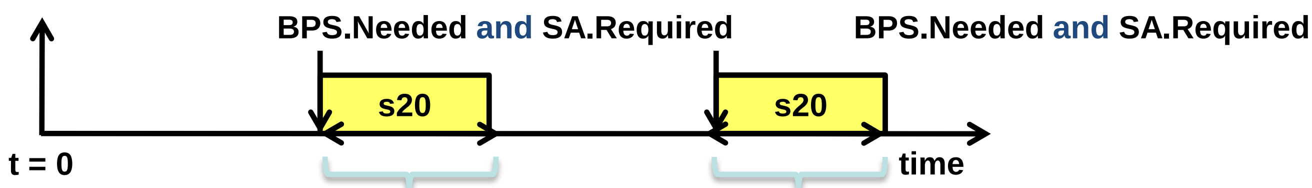
Special FORML-L syntax	Standard Modelica syntax
duringAny duration	**duringAny**(duration)
after event	**after**(event)
after event1 ***untilNext*** event2	**afterUntil**(event1, event2)
after event ***for*** duration	**afterFor**(event, duration)
after event ***within*** duration	**afterWithin**(event,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

From Text to Simulated Requirement – Modelica Extended with new Operators

From a text requiremen expressing a condition:

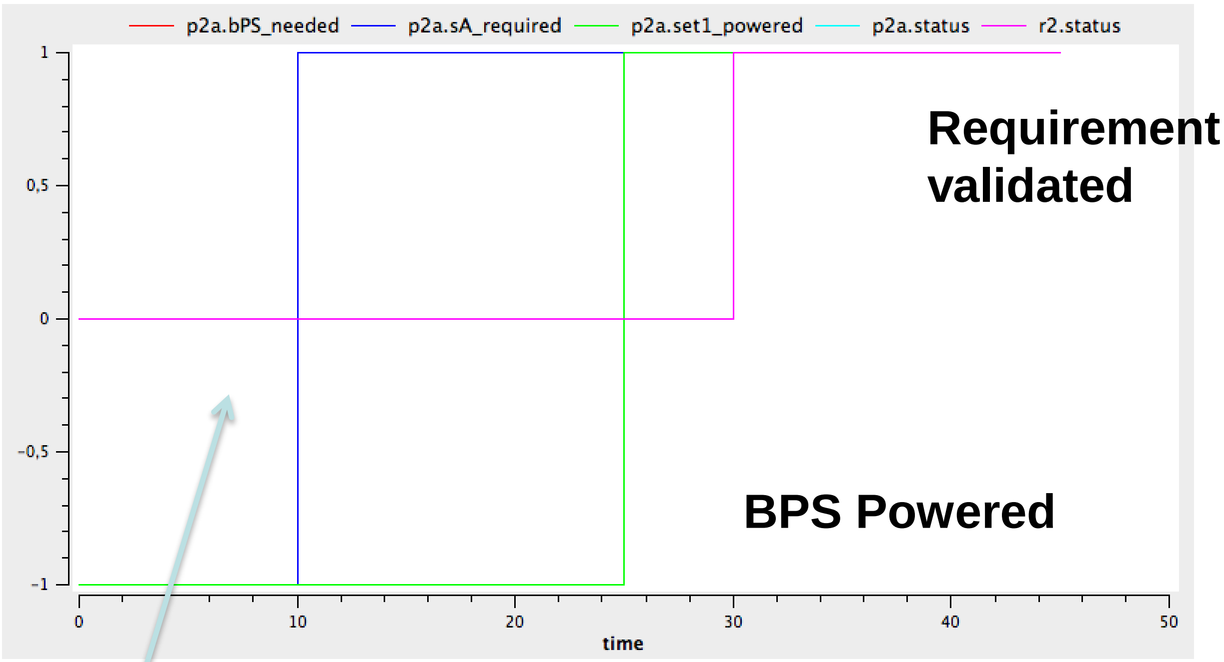
A - In the absence of any Backup Power Supply (BPS) component failure or in the presence of a single sensor failure, when the BPS is not under maintenance, in case of loss of MPS, and if safety injection is required, Set1 must be powered within 20 s

```
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

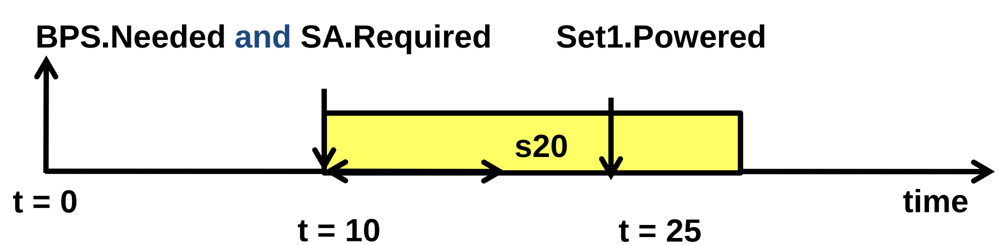
From Text to Simulated Requirement – Requirement not Violated – OpenModelica Simulation



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

Within 20s

Requirement undefined outside the specified time window



MODELICA

Industrial Use Case for Requirements Verification and Model Composition in ModelicaML

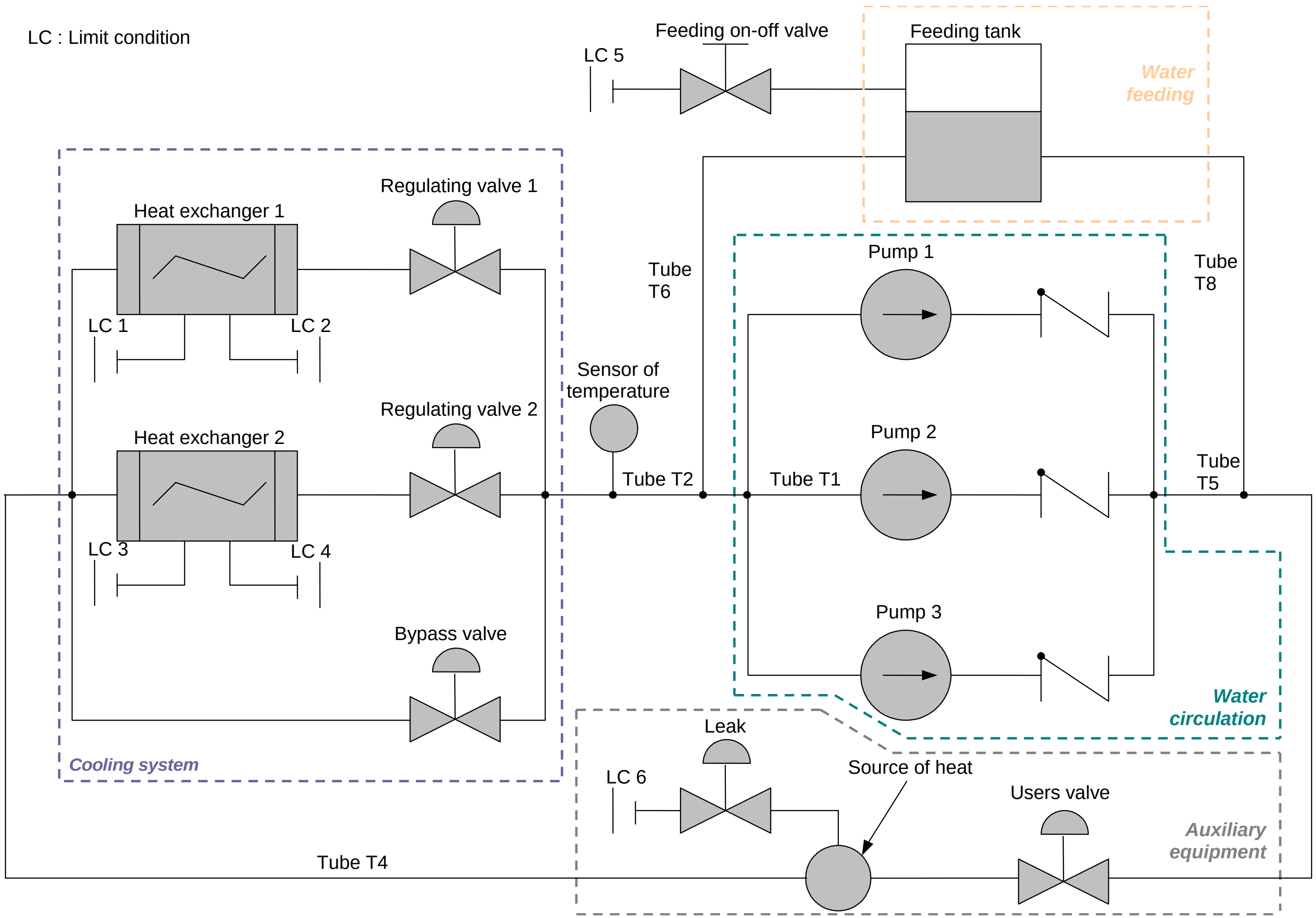
OPENPROD-Project Case Study, performed 2012; presented 2013

•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 Ts - e and Ts + e (Ts : 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