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

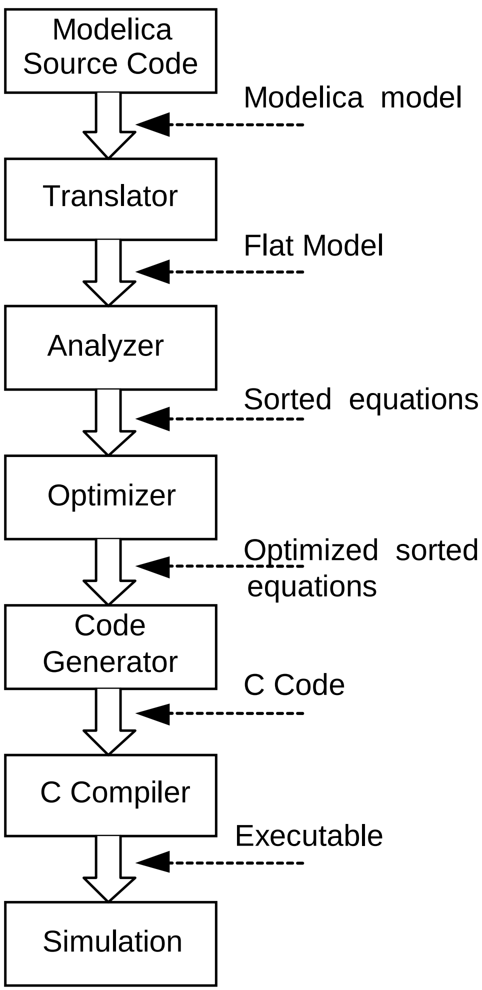
    ...

MODELICA

Model Compiler Translation Phases Extended with Debugging

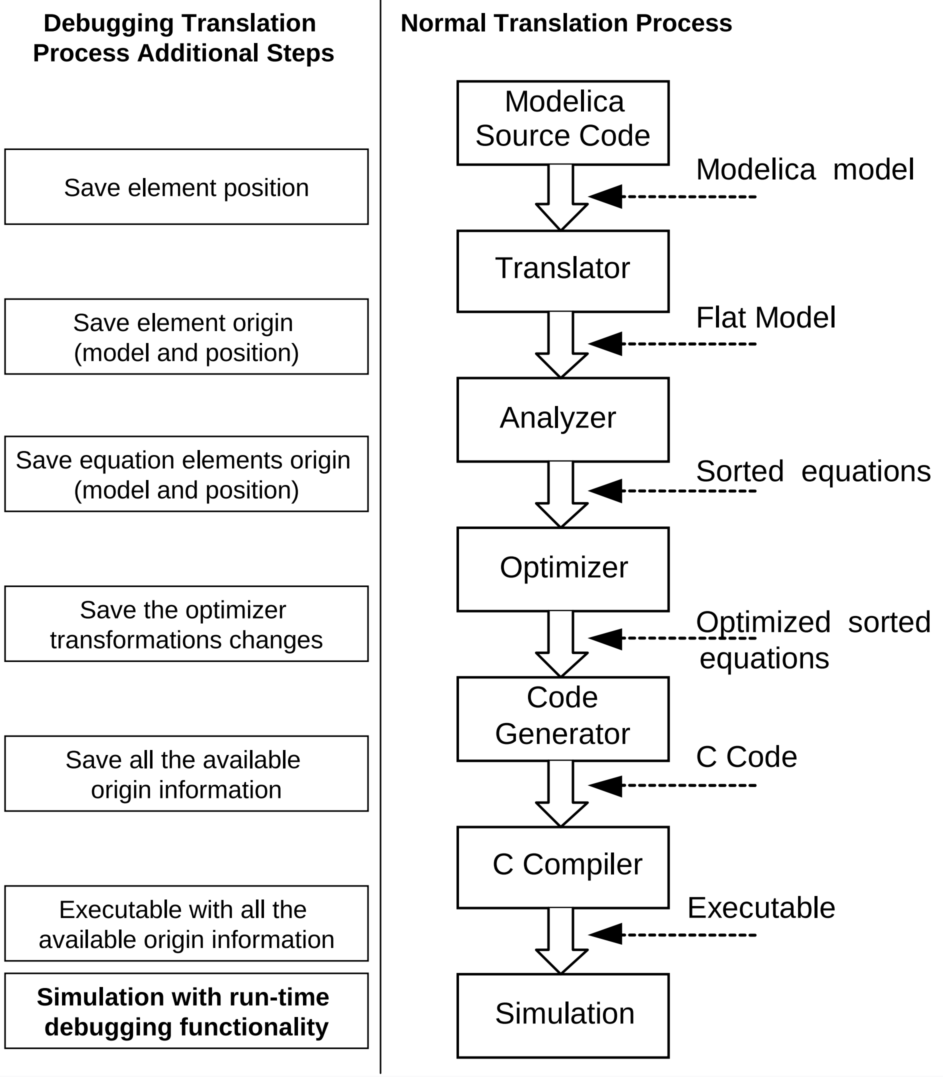
- Include debugging support within the translation process

Normal Translation Process



Model Compiler Translation Phases Extended with Debugging

- Additional step to provide needed debugging information



Example Symbolic Transformations with Compiler Debug Trace

- Complicated to understand source of some errors
- Efficient trace of transformations

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

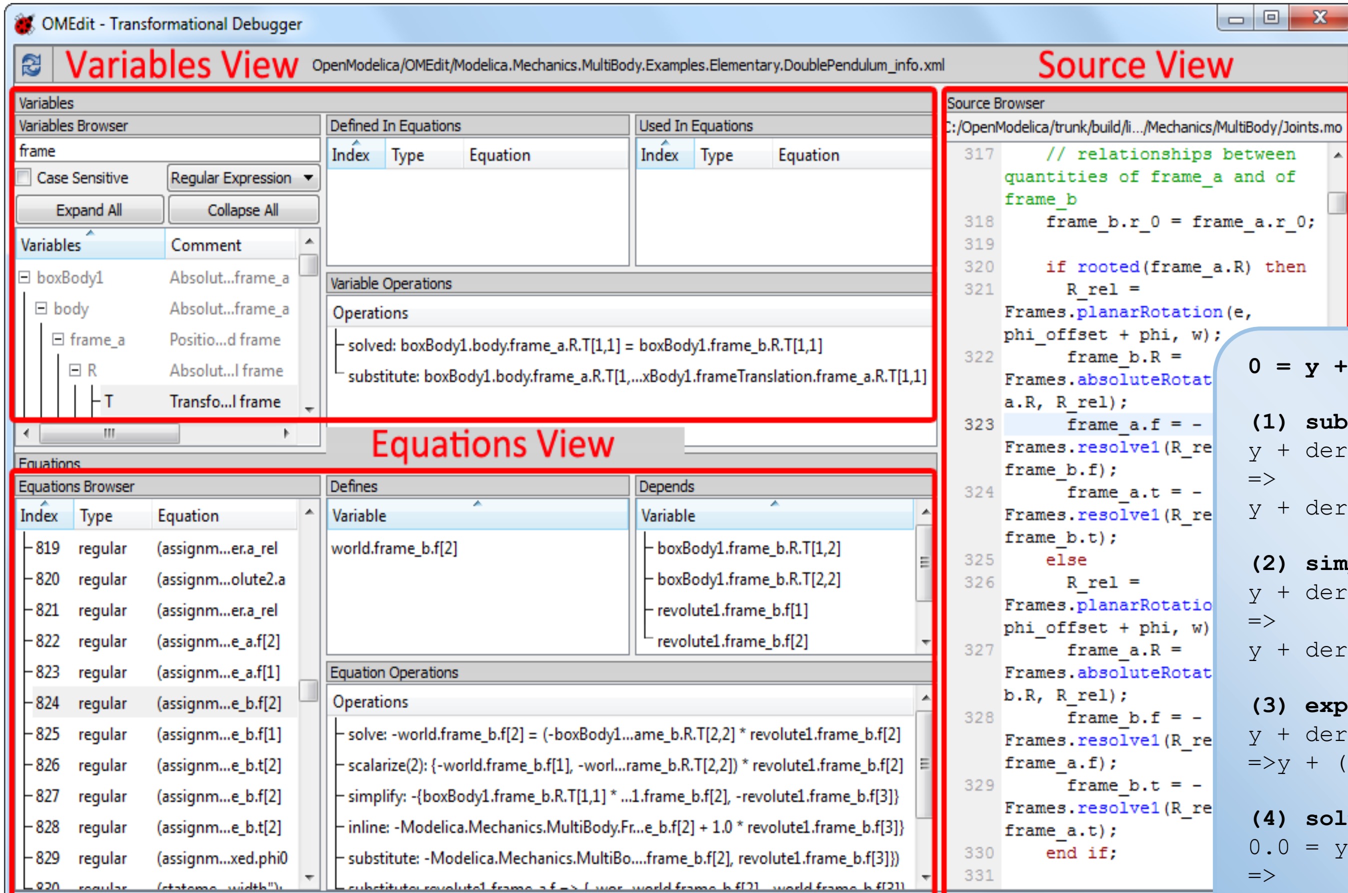
Properties of Transformation Trace

- 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)

OpenModelica Equation Model Debugger

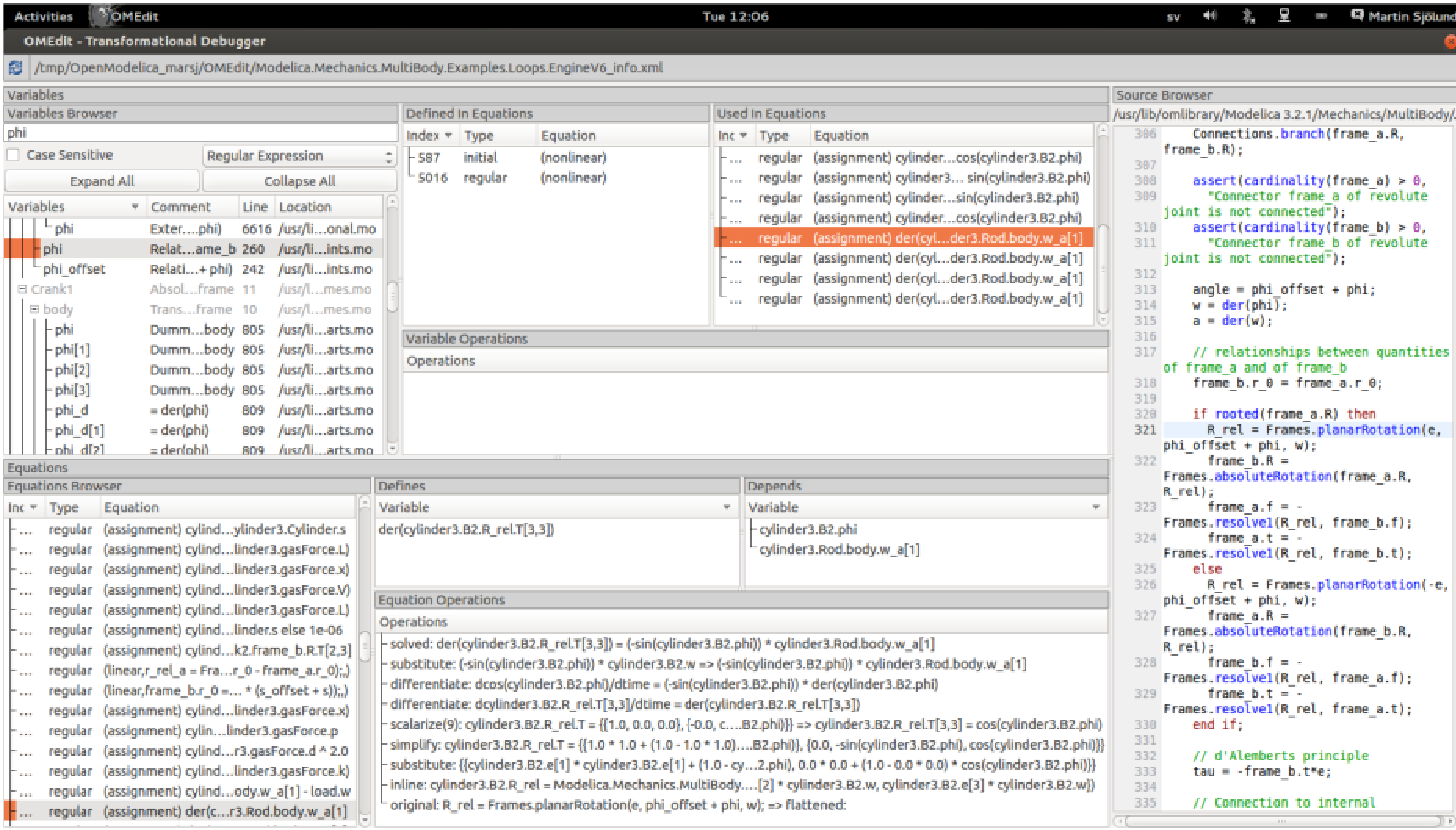


Showing equation transformations of a model:

```
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
time <> 0
```

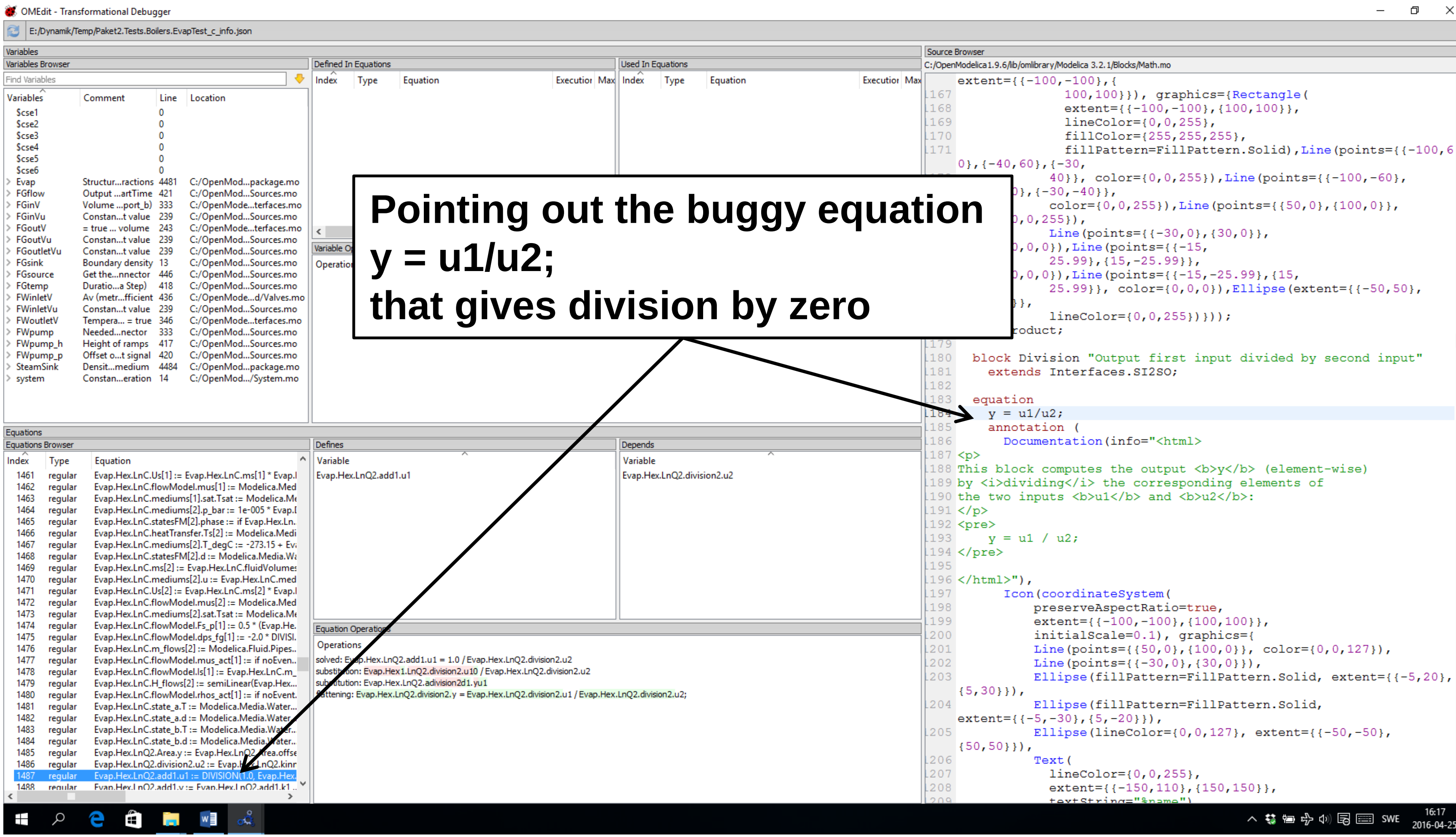
Mapping run-time error to source model position

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



MODELICA

Equation Model Debugger on Siemens Model
(Siemens Evaporator test model, 1100 equations)

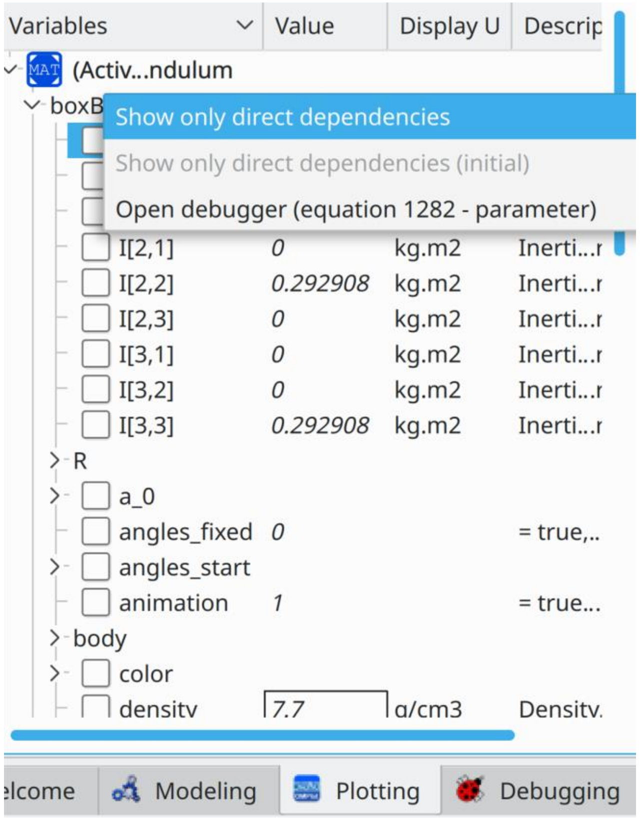


Pointing out the buggy equation
y = u1/u2;
that gives division by zero

MODELICA

New OM Debug function that can trace (and plot) which variables and equations influence a variable

New menu choice to show direct dependencies



List of Variables directly influencing:

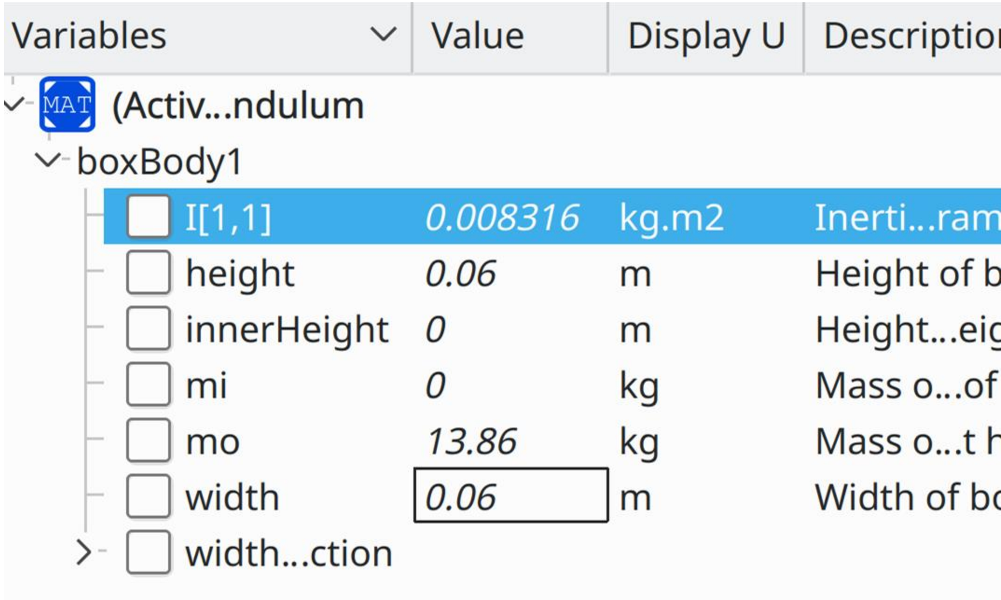
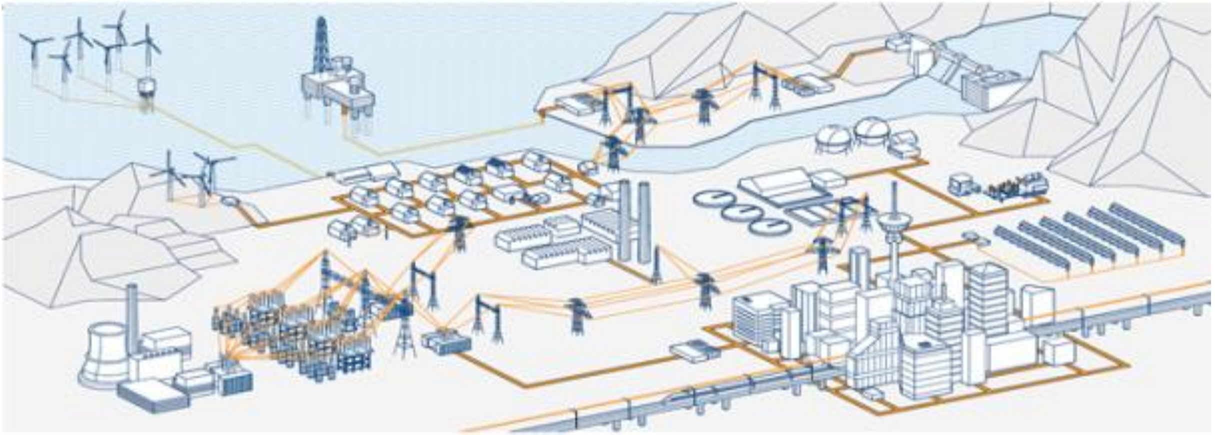


ABB Industry Use of OpenModelica Debugger

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



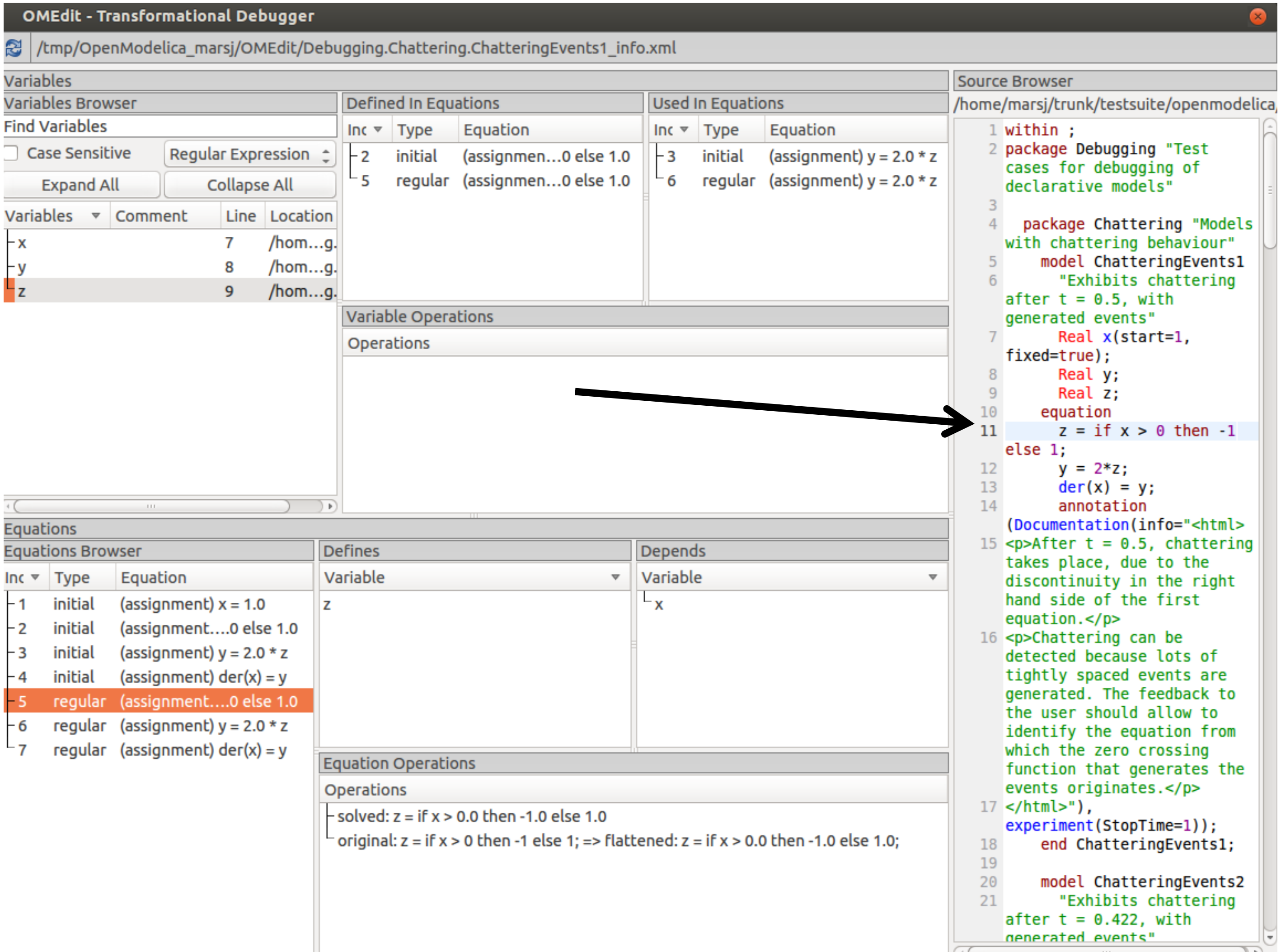
- ABB: "*ABB uses several compatible Modelica tools, including OpenModelica, depending on specific application needs.*"
- ABB: "*OpenModelica provides outstanding debugging features that help to save a lot of time during model development.*"

MODELICA

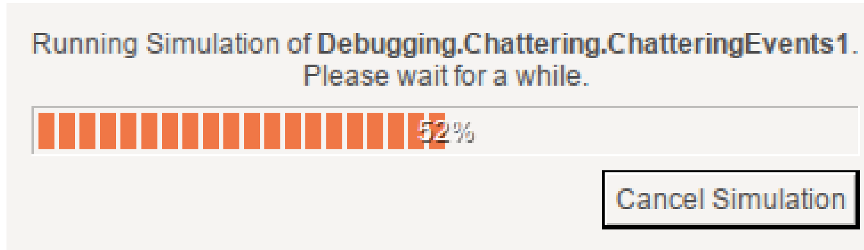
Equation Debugging Summary

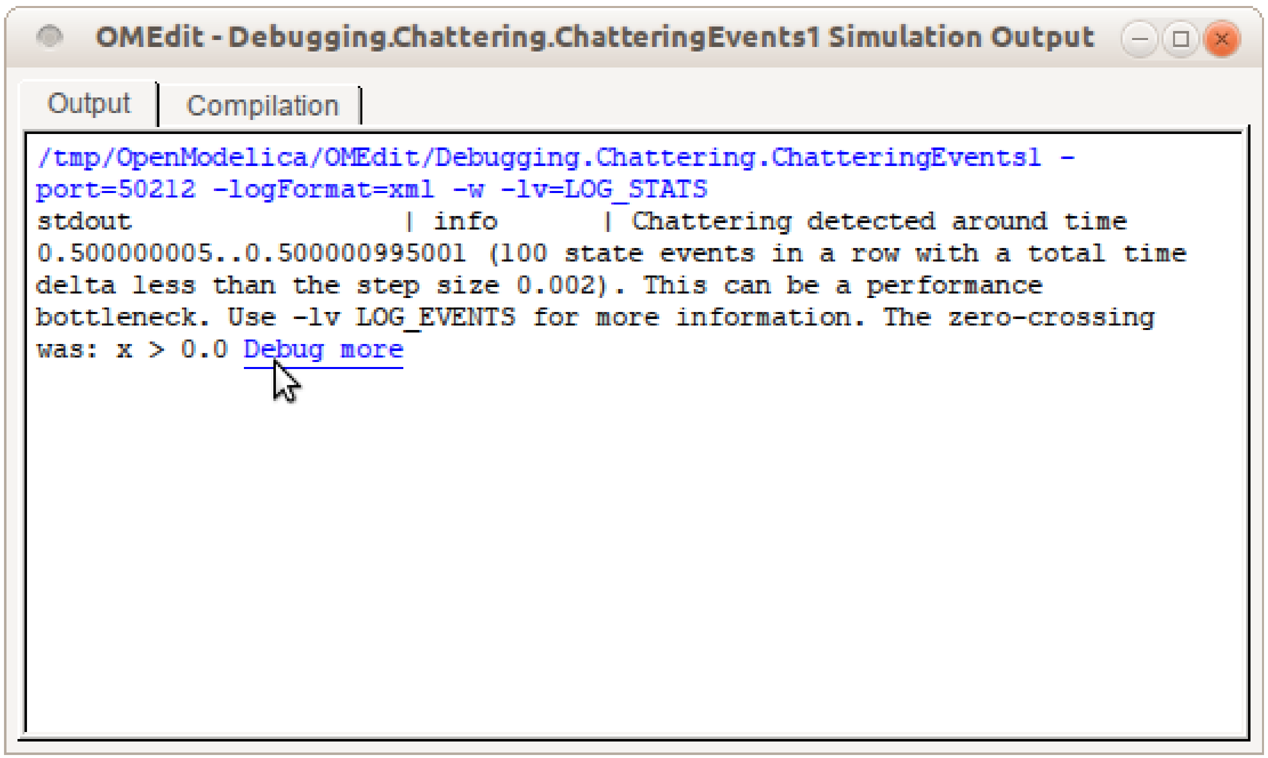
- Debugging **equation-based** models present new **challenges**
- **Equation** systems are **transformed** symbolically to a form hard for the user to understand
- Maintain and **explain** a **mapping** between the **low** level and the **high** level model
- **The first integrated static/dynamic debugger of any Modelica tool**

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



Error Indication – Simulation Slows Down



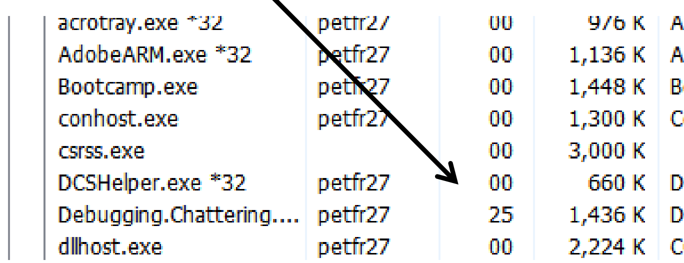


Exercise – Equation-based Model Debugger

In the model ChatteringEvents1, chattering takes place after t = 0.5, due to the discontinuity in the right hand side of the first equation. Chattering can be detected because lots of tightly spaced events are generated. The debugger allows to identify the (faulty) equation that gives rise to all the zero crossing events.

```
model ChatteringEvents1
  Real x(start=1, fixed=true);
  Real y;
  Real z;
equation
  z = noEvent(if x > 0 then -1 else 1);
  y = 2*z;
  der(x) = y;
end ChatteringNoEvents1;
```

Uses 25% CPU



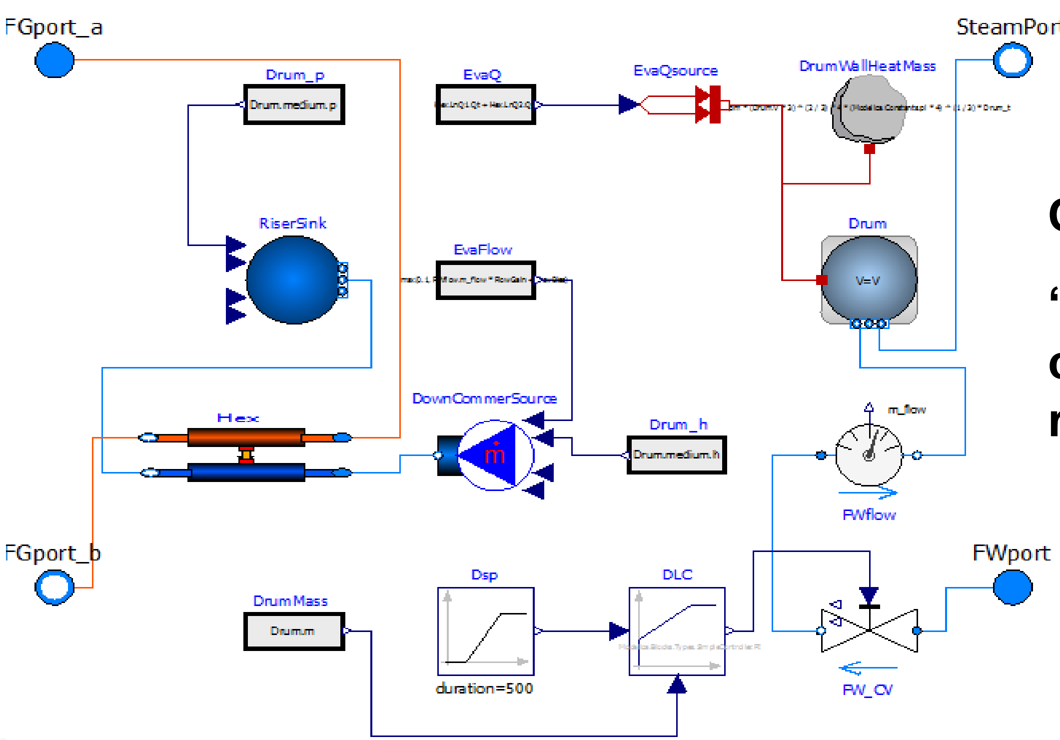
- Switch to OMEdit text view (click on text button upper left)
- Open the Debugging.mo package file using OMEdit
- Open subpackage Chattering, then open model ChatteringEvents1
- Simulate in debug mode (transformational debugger)
- Click on the button Debug more (see prev. slide)
- Possibly start task manager and look at CPU. Then click stop simulation button

Performance Analysis

Performance Profiling for faster Simulation

(Here: Profiling equations of Siemens Drum boiler model with evaporator

- Measuring **performance** of equation blocks to find bottlenecks
  - Useful as input before model simplification for real-time applications
- Integrated with the debugger to **point out the slow equations**
- Suitable **for real-time profiling** (collect less information), or a complete view of all equation blocks and function calls

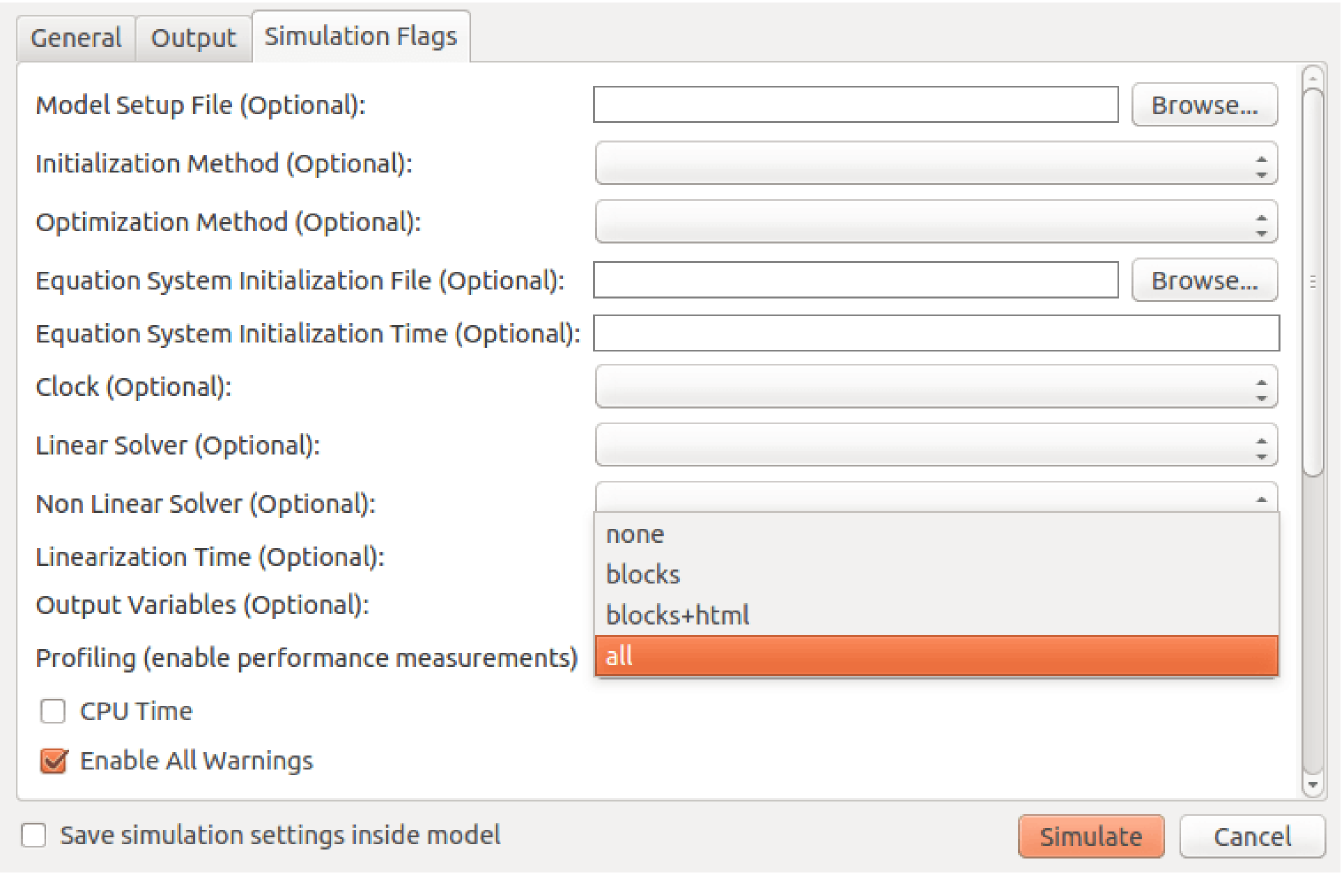


Conclusion from the evaluation:

"…the profiler makes the process of performance optimization radically shorter."

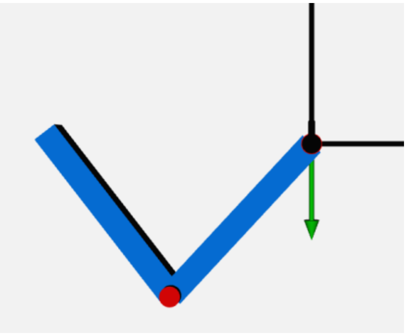
Using the Performance Profiler on DoublePendulum

- When running a simulation from OMEdit, it is possible to enable profiling information, which can be combined with the transformations browser.



Set this in simulation Setup

DoublePendulum in MultiBody library



Using the Performance Profiler on the DoublePendulum model

- When profiling the DoublePendulum example from MSL, the following output below is a typical result. This information clearly shows which system takes longest to simulate (a linear system, where most of the time overhead probably comes from initializing LAPACK over and over).

Equations Browser

Index	Type	Equation	Executions	Max time	Time	Fraction
+ 876	regular	linear, size 2	4602	0.000199	0.0582	86.2%
836	regular	(assignment) revolute2.R_rel.T[2,2] = cos(revolute2.phi)	1534	8.25e-05	0.000491	0.728%
837	regular	(assignment) revolute2.R_rel.T[2,1] = -sin(revolute2.phi)	1534	7.29e-05	0.000422	0.625%
841	regular	(assignment) boxBody1.frame_...[2,1] = -sin(damper.phi_rel)	1534	7.1e-05	0.000395	0.585%
840	regular	(assignment) boxBody1.frame_...T[2,2] = cos(damper.phi_rel)	1534	7.08e-05	0.000361	0.535%
839	regular	(assignment) revolute2.R_rel.T[1,1] = cos(revolute2.phi)	1534	7.33e-05	0.000303	0.449%
842	regular	(assignment) boxBody1.frame_b.R.T[1,2] = sin(damper.phi_rel)	1534	7.45e-05	0.000303	0.449%
838	regular	(assignment) revolute2.R_rel.T[1,2] = sin(revolute2.phi)	1534	7.11e-05	0.0003	0.444%
849	regular	(assignment) boxBody1.frame_...T[1,1] = cos(damper.phi_rel)	1534	7.29e-05	0.000286	0.424%
827	regular	(assignment) revolute1.tau = (-damper.d) * revolute1.w	1534	6.84e-05	0.000274	0.406%

Defines

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

Performance Profiler Exercise

- Try the profiler on this model. Results in Equations Browser, enlarge the window, click on Fraction to sort in ascending/descending order.

```
model ProfilingTest
  function f
    input Real r;
    output Real o = sin(r);
  end f;
  String s = "abc";
  Real x = f(x) "This is x";
  Real y(start=1);
  Real z1 = cos(z2);
  Real z2 = sin(z1);
equation
  der(y) = time;
end ProfilingTest;
```

Equations Browser

Index	Type	Equation	Executioi	Max time	Time	Fraction
21	regular	non-linear, unkno...tion variables: 1	1006	7.24e-05	0.000664	44.6%
19	regular	non-linear (torn),...ation variables: 1	1006	7.87e-05	0.000635	42.7%
22	regular	(assign) der(y) := time	540	8e-07	1.71e-05	1.15%
9	initial	non-linear (torn),...ation variables: 1	8	5.3e-06	5.3e-06	0.356%
11	initial	non-linear, unkno...tion variables: 1	3	1.5e-06	1.5e-06	0.101%
2	initial	(assign) s := "abc"	2	2e-07	2e-07	0.0134%
1	initial	(assign) y := $START.y	1	1e-07	1e-07	0.00672%
12	initial	(alias) 22	0	0	0	0%
23	parameter	(alias) 2	0	0	0	0%