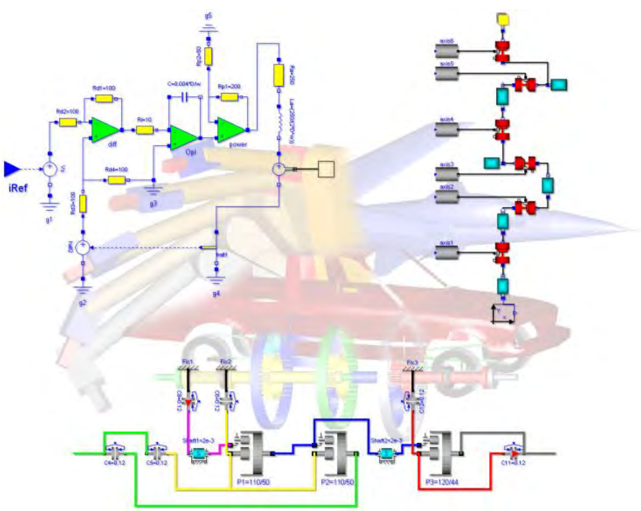
Research in Model-Based Product Development at PELAB and RISE in the MODPROD Center

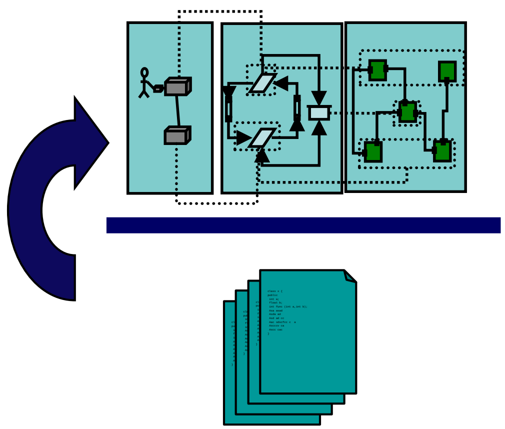
Presentation at MODPROD’2020

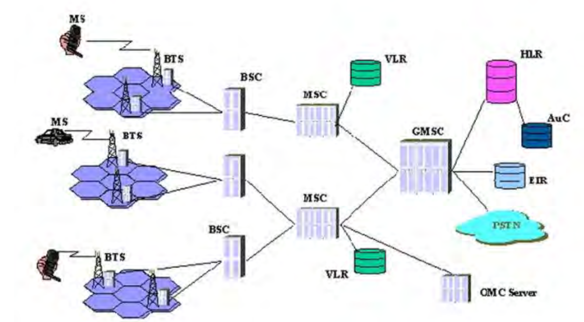
Department of Computer and Information Science
Linköping University

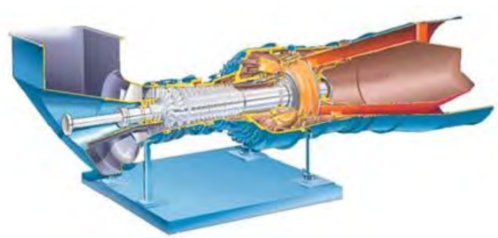
2020-02-05

Peter Fritzson, et al

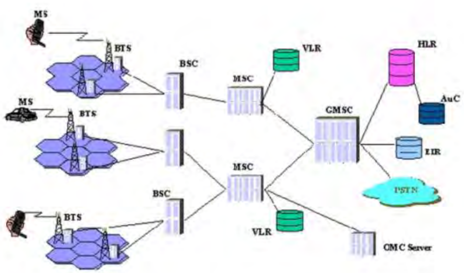








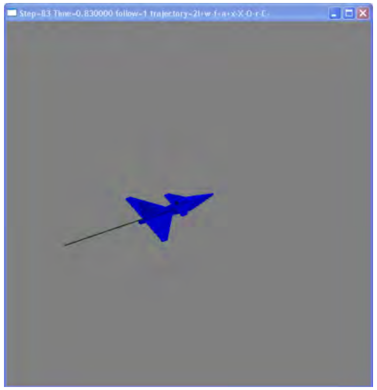
Industrial Challenges for Complex Products of both Software and Hardware



- Increased Software Fraction

- Embedded and real time constraints

- Higher demands on effective strategic decision making



Digitalization Revolution Happening Now!

Internet of Things, AI, CPS

Research

Large-Scale Modeling and Simulation

Modeling-Language Design

Model-Based Co-simulation with FMI and TLM

Model Debugging

Model-Based Fault Analysis

Embedded System Real-Time Modeling

Modeling Support Environments

Large-Scale, High Performance Model-Based Development

10 million equation goal!

Per Östlund, Adrian Pop, Martin Sjölund, Mahder Gebremedhin, John Tinnerholm,

Peter Fritzson, et al

High Performance Modelica Compilation Methods for Large Model Applications – A Quantum Leap!

- The **OpenModelica new compiler frontend** – a **large** effort to **redesign** and rewrite more than half of the compiler to gain high compilation **performance** and 100% Modelica semantics
- Uses **Model-centric** and **multiple phases design** principles
- OpenModelica 1.14.1 December 2019 – First release with New Frontend
- The New frontend is about **10 to 100 times faster** than the old compiler frontend.

- During 2020 – Further tuning and performance increases; enhanced compiler backend

Experimental OpenModelica Compiler in Julia
Goal – Flexible Just-in-time Compilation, variable structure

- Drugin 2019 , Developed a preliminary MetaModelica to Julia translator
- Translated most of the previous OM frontend
- Able to execute some translated MetaModelica functions
- Further performance tuning needed, integration with solver
- Goal – support variable structure system
- Goal – support large-scale models

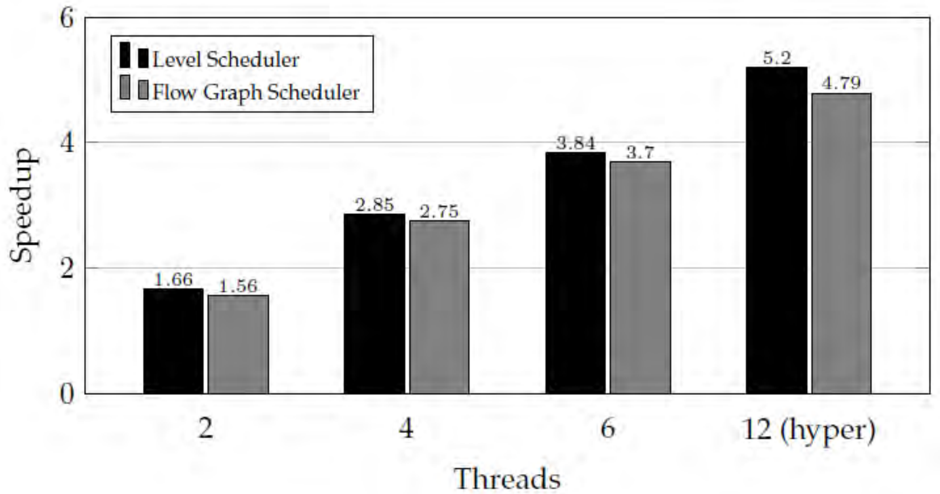
ParModAuto Parallelization (Release spring 2020)

Automatic AutoTuned Parallelization of Equation-based Models

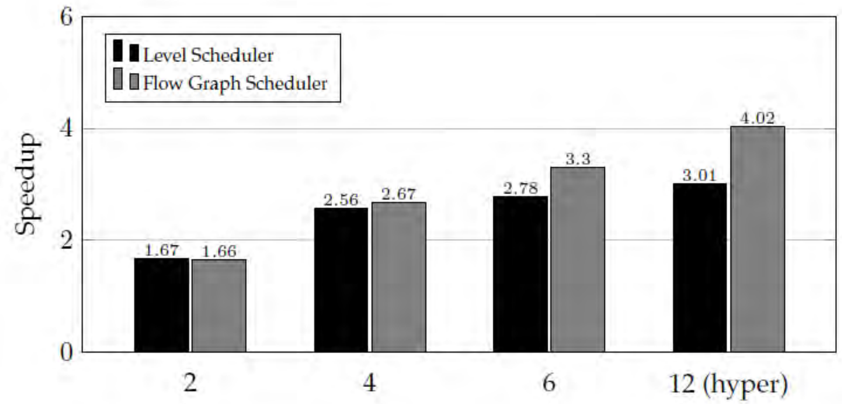
- **Parallelization for higher performance**

- Automatic **Parallelization**
- Automatic **clustering** of small tasks
- **Automatic load balancing** based on measurements, automatically adapts to changing load
- **Shared-memory** task parallelization

SteamPipe640 model, **Speedup 5.2 on 6 cores**:



BranchingDynamicPipes model, **Speedup 4 on 6 cores**:

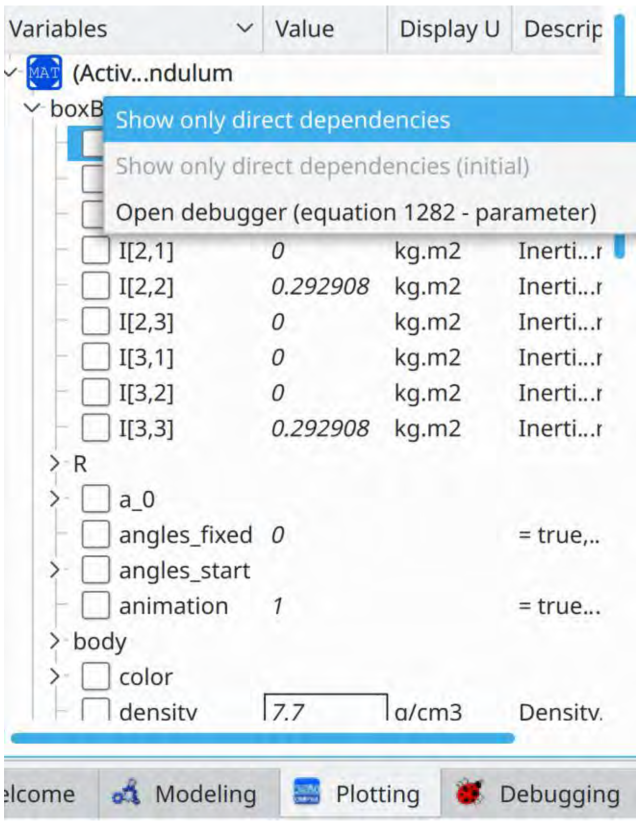


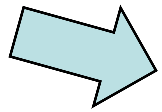
Enhance Modeling Ease-of-use! Model Debugging and Performance Analysis

Martin Sjölund,
Adrian Pop, Adeel Asghar
Dept Computer and Information Science
Linköping University

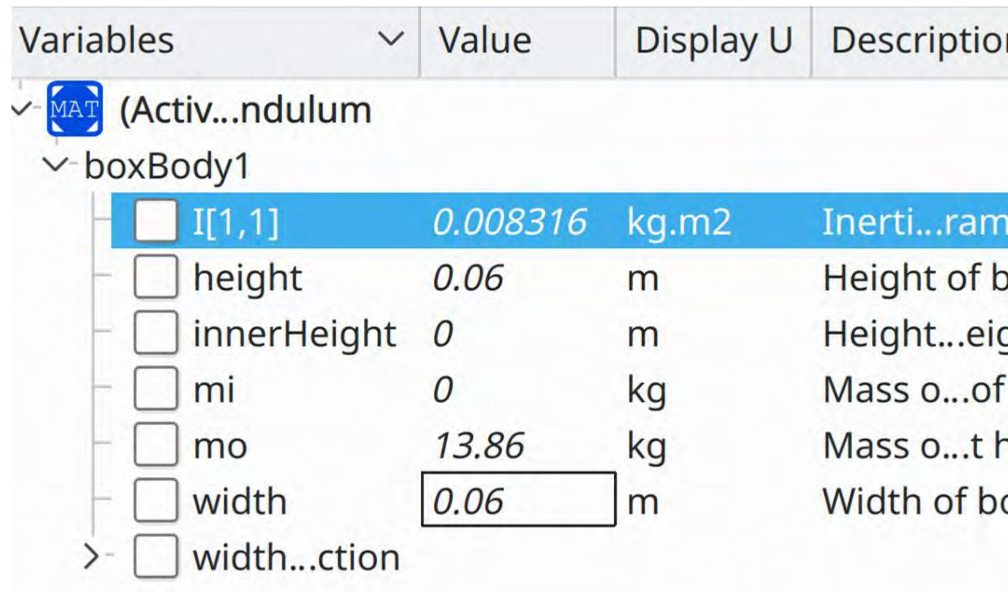
Enhanced OM Debugger that can trace (and plot) which variables and equations influence a variable

New functionality to show direct variable dependencies





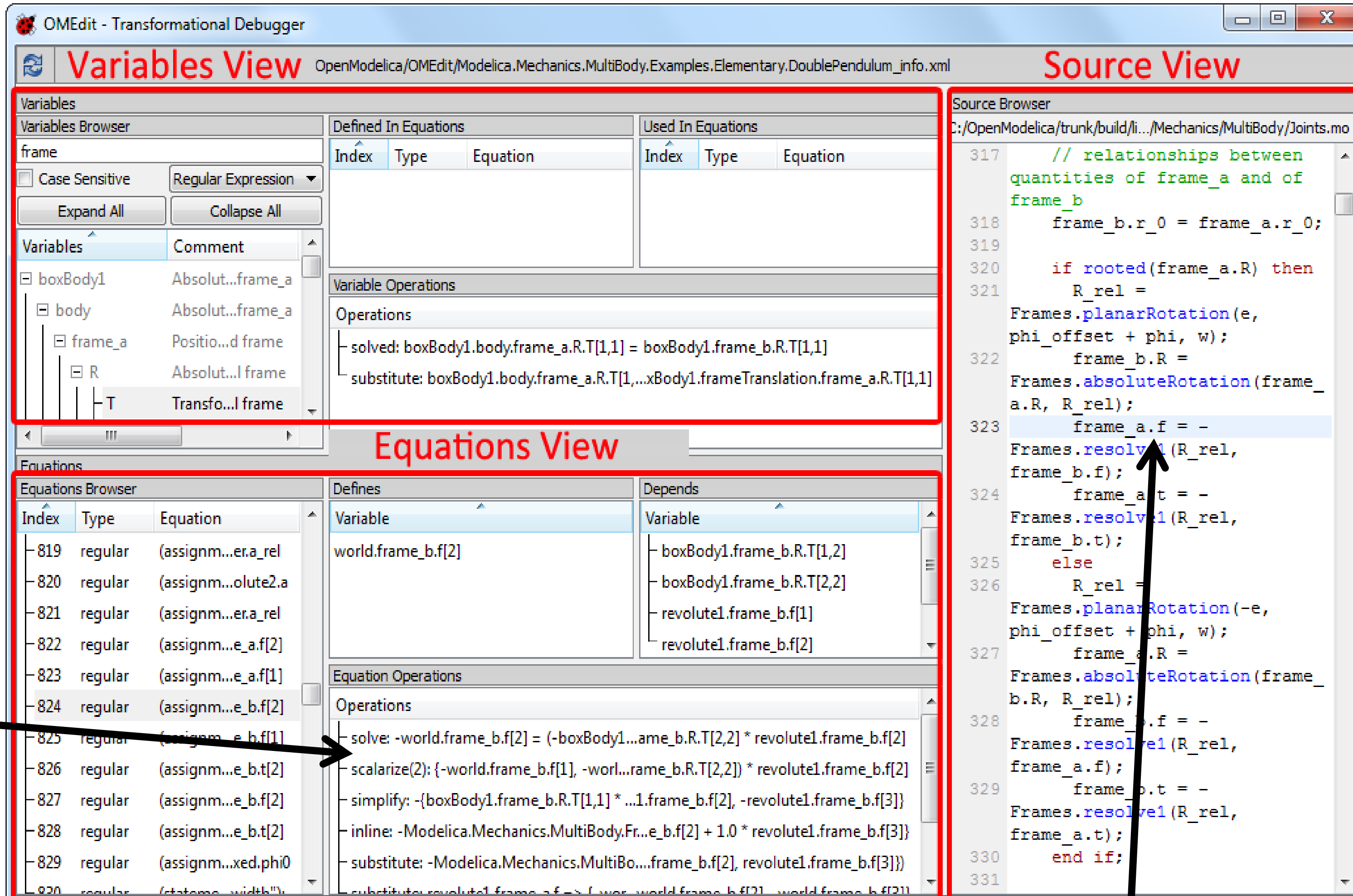
List of Variables directly influencing:



Integrated Static-Dynamic OpenModelica Equation Model Debugger

Efficient handling of **Large** Equation Systems

Showing equation transformations of a model:



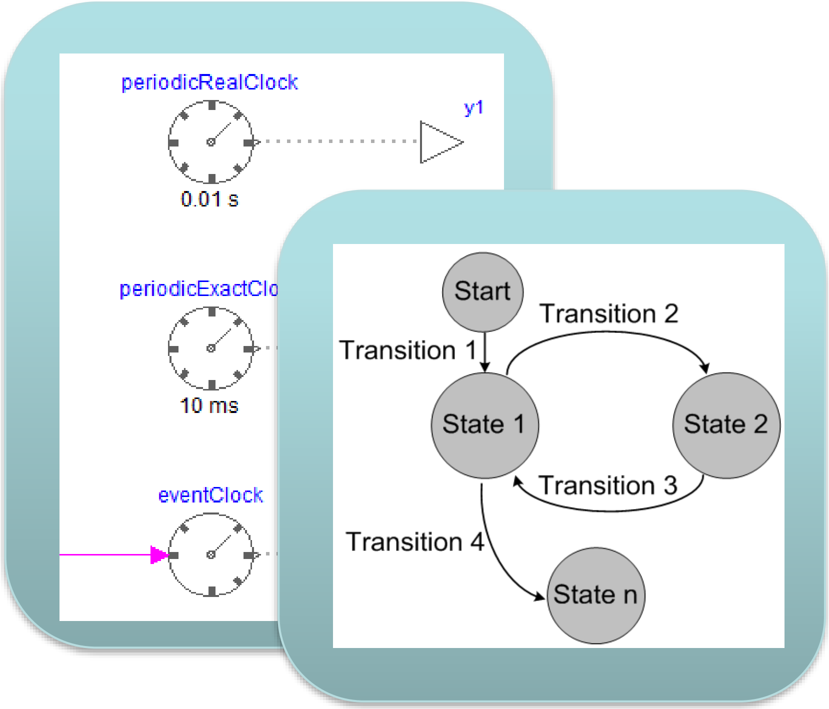
Mapping dynamic run-time error to source model position

pelab

Further Ongoing Research on Debugging

Debugging of new features

- **clocked** synchronous models
- **real-time debugging** and event tracing
- graphic support for **state machine** debugging



Digital Twins using Modelica and OpenModelica

Collaboration with
Modelicon InfoTech, Bangalore, India
and GI-LIFT AB, Linköping

Adeel Asghar, Martin Sjölund, Peter Fritzson

More Sustainable Foestry – Digital Twin of Balloon-Assisted UAV – Collaboration with GI-LIFT AB and Modelicon

Avoid clear-cut damage

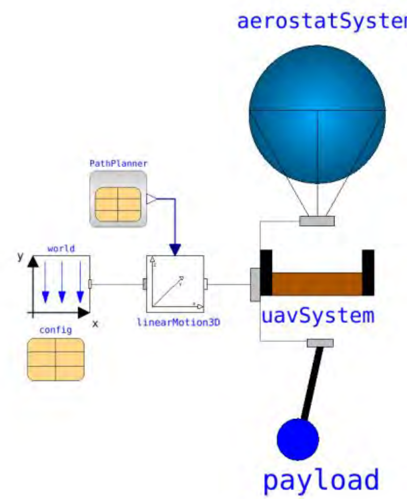
Digital Twin Using OpenModelica

Instead high-powered Electric Ballon-assisted UAV lifting system
(patent pending, GI-LIFT)

Timber

Branches

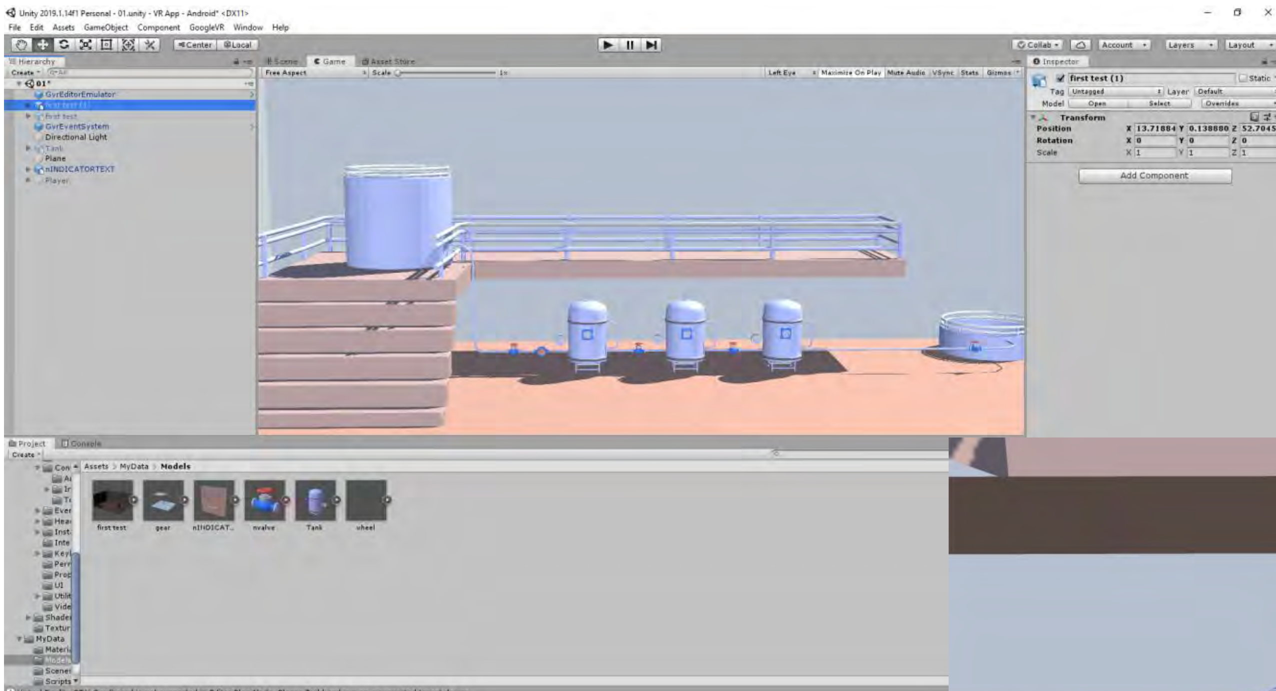
Whole tree



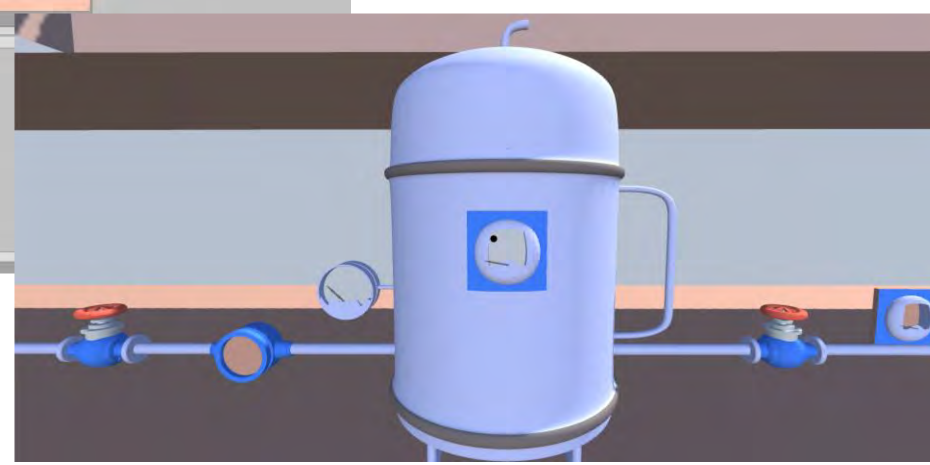
Test-Flight of Balloon-Assisted UAV – Outside Linköping – by GI-LIFT AB

Integration with Unity 3D Visualization in VAL – Virtual Automation Lab

Development environment integrated with OpenModelica



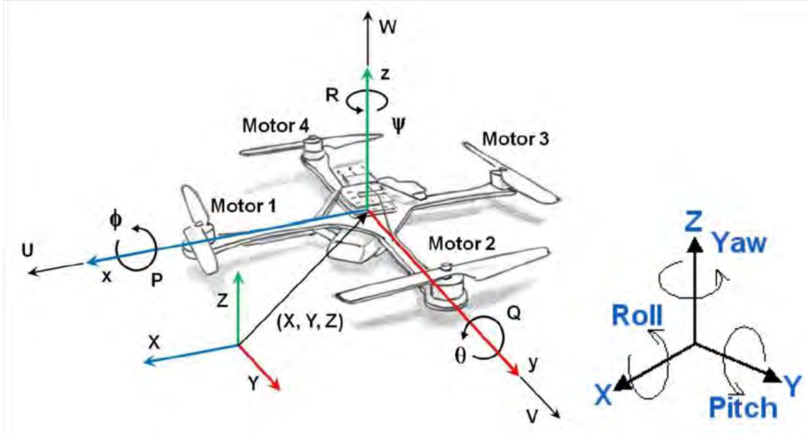
VR Model – Unity 3D



Developed by Modelicon and BMSCE
in Bangalore, India

Digital Twin OpenModelica Applications by Modelicon (Bangalore) Model-based Control of UAVs and Walking Robots

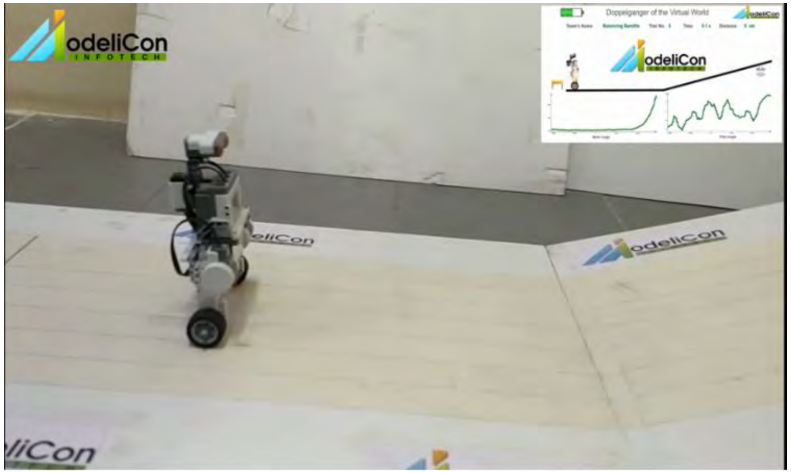
- UAV control and simulation
- Walking 2-wheel robot



UAV

Movie demo

All models and control software done using OpenModelica!



Walking 2-wheel Robot,

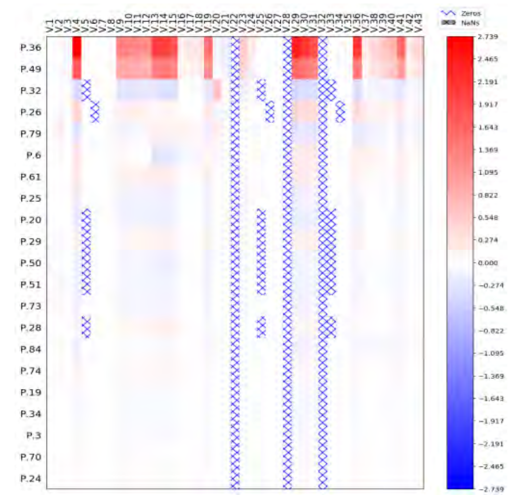
Movie demo

Simultaneous Param-based Sensitivity Analysis and Robust Optimization (collaboration with Univ. Buenos Aires)

Paper published at
EOOLT 2017 (prototype)

Planned OpenModelica
Release spring 2019

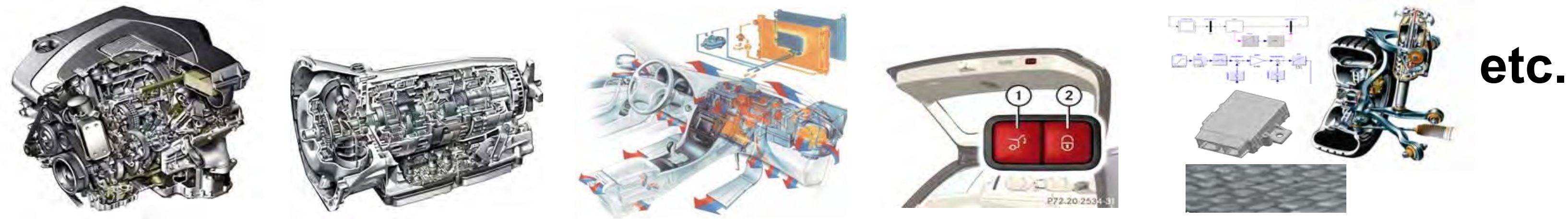
- To define a sensitivity experiment:
  - The state variable to analyze
  - The set of parameters to perturb
  - The allowed perturbation intervals for each parameter
- Main goal: pinpoint a small number of parameters that produce the largest deviations when perturbed within narrow ranges around their default values
- To select parameters and their intervals is not a trivial task
  - Responsibility relies completely on the expertise of the user
  - Enabling all parameters can lead to very costly experiments
- Use a top-N subset of parameters from a ranked list
  - obtained using individual parameter-based analysis
- Using CURVIF robust derivative-free model building method for few function evaluations
- Heat-map visualization of parameter influence



Co-simulation, FMI, Model Composition

Lennart Ochel, Robert Braun, Adeel Asghar, Adrian Pop, Arunkumar Palanisamy, Peter Fritzson

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)

functional mockup interface for model exchange and tool coupling

courtesy Daimler

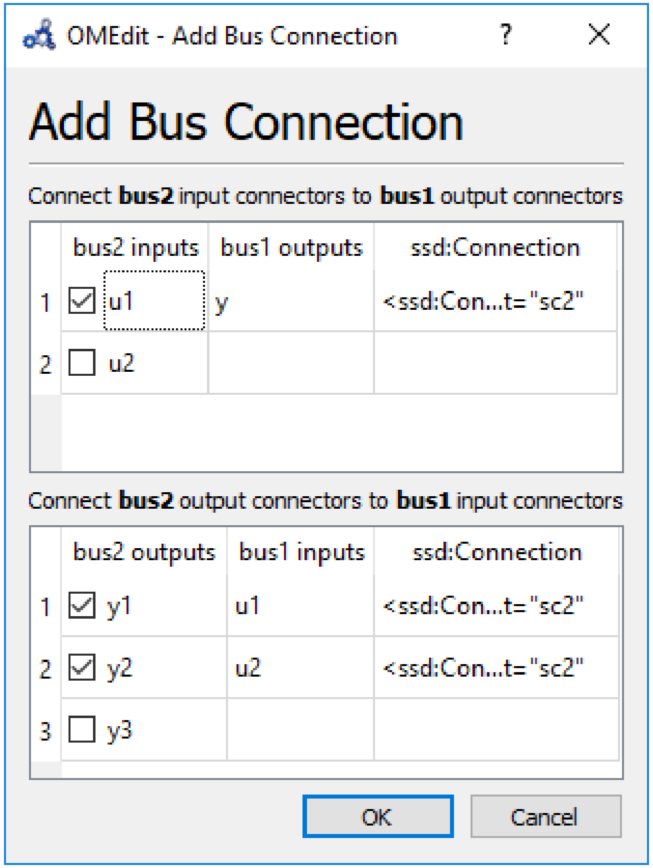
- FMI development was started by ITEA2 MODELISAR project. FMI is a Modelica Association Project now
- **Version 1.0**
- FMI for Model Exchange (released Jan 26,2010)
- FMI for Co-Simulation (released Oct 12,2010)
- **Version 2.0**
- FMI for Model Exchange and Co-Simulation (released July 25,2014)
- **> 100 tools** supporting it (https://www.fmi-standard.org/tools)

Enhanced FMI Co-simulation, Run-time, and Master Simulation Tool

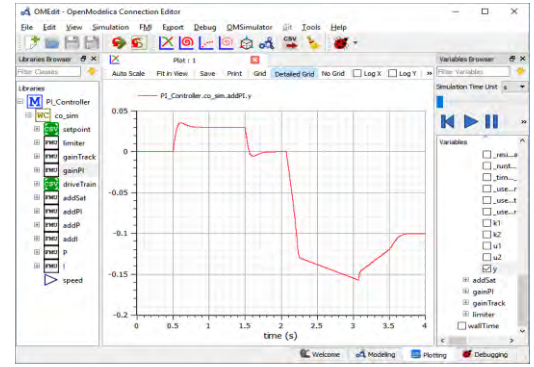
- Further **extensions** to the FMI standard to support TLM-based co-simulation including support for SKF mechanical bearing models
- **Enhanced run-time** for efficient co-simulation of FMUs, including FMUs from OpenModelica and Papyrus
- General **Master** simulation tool support for FMI

OMSimulator Simulation, SSP, and Tool Comparison

Adding SSP bus connections



FMI Simulation results in OMEdit



FMI Simulation Tool Comparison

	OMSimulator	DACCOSIM	Simulink	PyFMI
Commercial	No	No	Yes	No
Open-source	OSMC-PL, GPL	AGPL2	No	LGPL
Lookup Table	Yes	Yes	Yes	No
Alg. Loops	Yes	Yes	No	Yes
Scripting	Python, Lua	proprietary	proprietary	Python
GUI	Yes	Yes	Yes	No
SSP	Yes	No	No	No
platform	Linux/Win/macOS	Linux/Win	Linux/Win/macOS	Linux/Win/macOS

	Dymola	PySimulator	FMI Go!	FMI Composer
Commercial	Yes	No	No	Yes
Open-source	No	BSD	MIT	No
Lookup Table	Yes	Yes	Yes	Yes
Alg. Loops	Yes	Yes	Yes	Yes
Scripting	proprietary	Python	Go	No
GUI	Yes	Yes	No	Yes
SSP	No	No	Yes	Yes
platform	Linux/Win	Linux/Win	Linux/Win/macOS	Linux/Win/macOS

Future Developments: FMI 3.0

Ports and Icons

Help the user to build consistent systems from FMUs and render the systems more intuitively with better representation of structured ports (for instance busses and physical connectors) in the modelDescription.xml.

Array variables

Allow FMUs to communicate multi-dimensional variables and change their sizes using structural parameters.

Clocks and Hybrid Co-Simulation

Introduces clocks for synchronization of variables changes across FMUs. Allows co-simulation with events.

Binary Data Type

Adds an opaque binary data type to FMU variables to allow, for instance, efficiently exchanging of complex sensor data.

Intermediate Variable Access

Allow access to intermediate input and output values between communication time points from the FMU to disclose relevant subsystem behavior for analysis or advanced co-simulation master algorithms for enhanced numerical stability.

Source code FMUs

Adding more information to the modelDescription.xml file to improve automatic import of source code FMUs.

Numeric Variable Types

Adds 8, 16, 32 and 64-bit signed and unsigned integer and single precision floating point variable types to improve efficiency and type safety when importing / exporting models from the embedded, control and automotive domains.

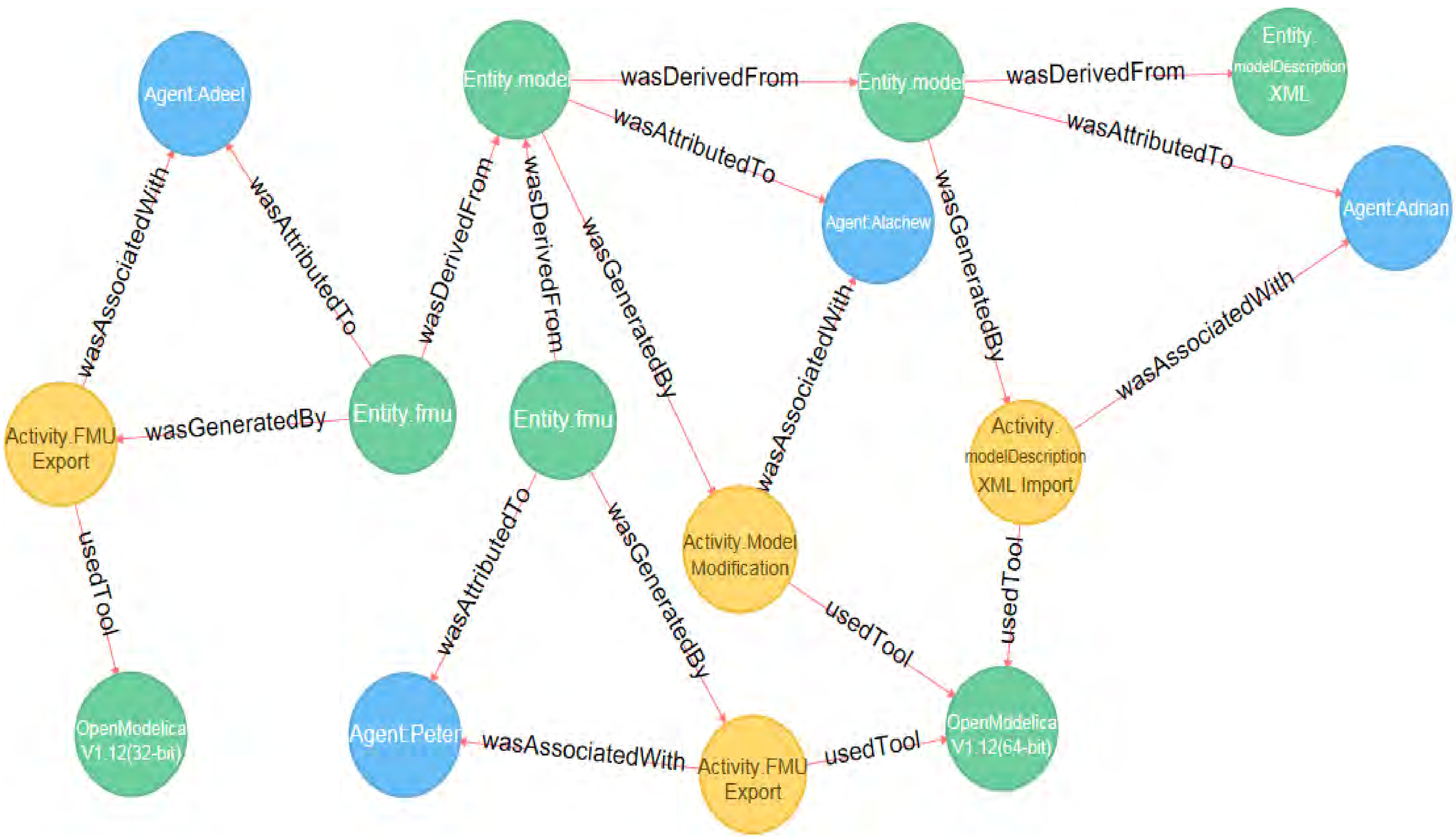
Extra directory

Adding a new folder in the ZIP Archive representing an FMU, providing additional data to travel with the FMU which can be modified by different tools, allowing for layered standards

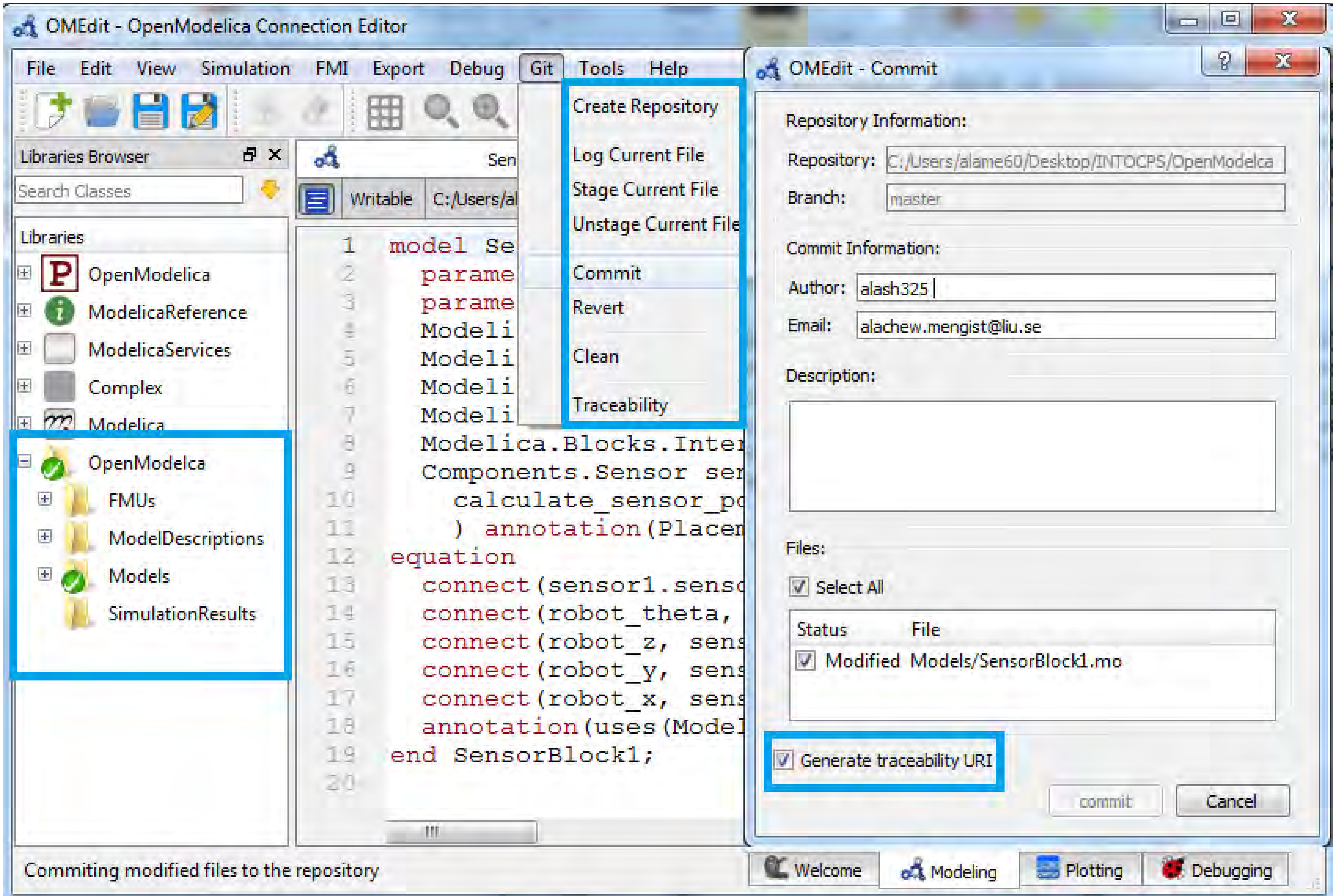
Model Management and Traceability

Adrian Pop, Alachew Mengist, Peter Fritzson

Traceability Information collected by OpenModelica

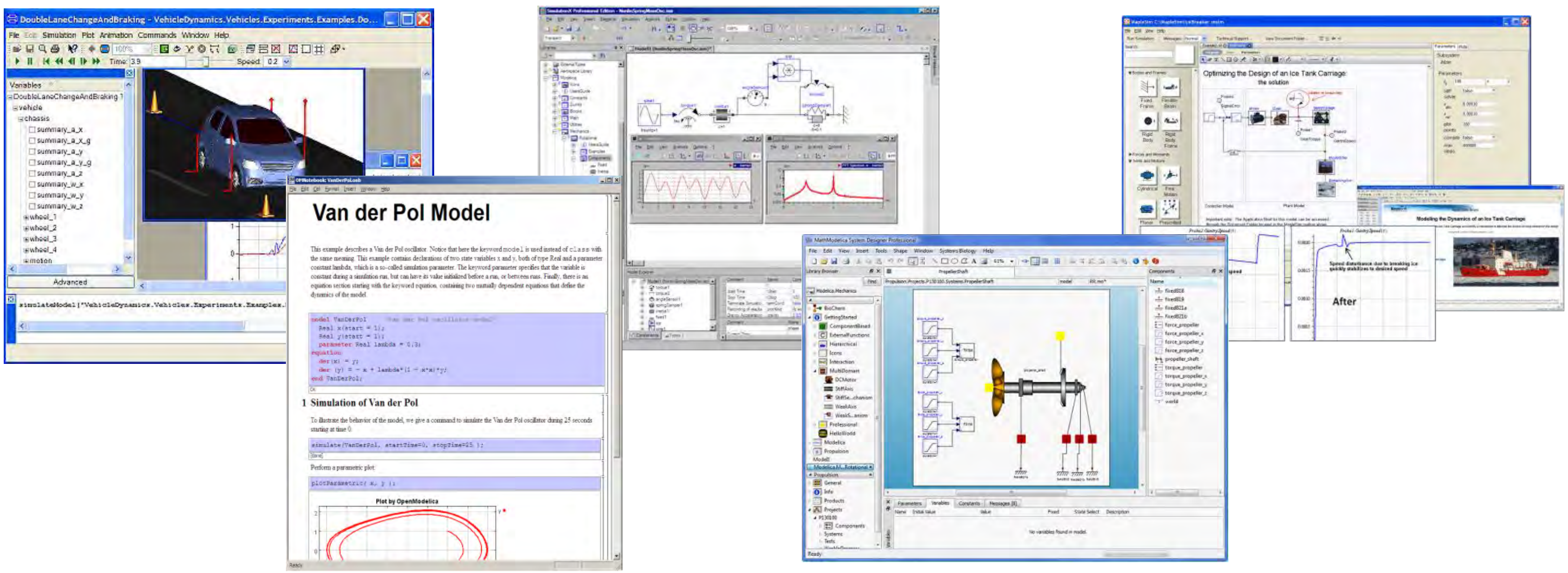


Model Management with Git Integration



Dynamic Verification/Testing of Requirements vs Usage Scenario Models EMBRACE project starting 2020

Lena Buffoni et al

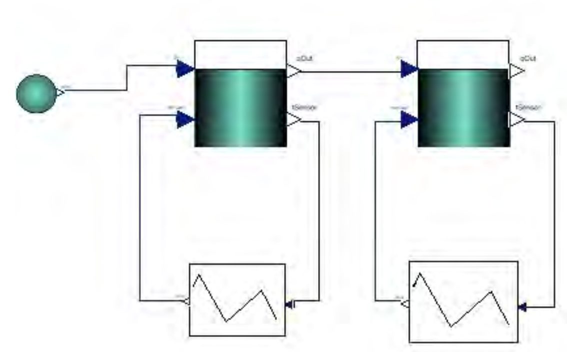


Testing a single verification model in Modelica

In EMBRACE project – develop CRML standardized Requirement language

- Req. 001: The volume of each tank shall be at least 2 m3.
- Req. 002: The level of liquid in a tank shall never exceed 80% of the tank height.
- Req. 003: After each change of the tank input flow, the controller shall, within 20 seconds, ensure that the level of liquid in each tank is equal to the reference level with a tolerance of ± 0.05 m.
- …

+



+

Start with constant flow and increase at t=150

Design alternative:
two tank model

Design alternative:
two tank model

One possible test scenario

Model-based Development Tooling for Embedded Systems

Project EMPHYSIS, EMISYS

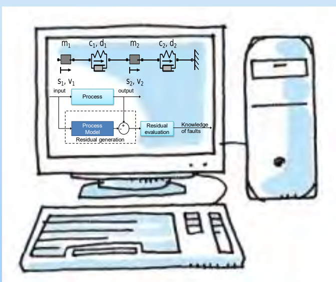
EMbedded systems with PHYSIcal models In production code Software

Lennart Ochel, Martin Sjölund, Adrian Pop, et al
Dept Computer and Information Science
Linköping University

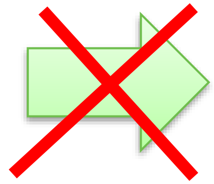
Technology Gap between Modeling and Simulation Tools and Embedded Software

ITEA 3

Physical Modelling Tools:
High-level modeling,
Model libraries
symbolic manipulation
solvers, advanced numerics



SimulationX Powered by ITI
Dymola
AMESim
MapleSim Advanced System-Level Modeling
OpenModelica
etc.



No automation,
Models
re-implemented
(hand-coded)

ECU code generation tools.
(Simulink, with special extensions (target link), ASCET)

Signal-flow oriented,
with strong restrictions
(e.g., no continuous states)

THE C PROGRAMMING LANGUAGE
ASCET

Currently the design flow for physical models in ECU software is **interrupted**

Bridging the gap between modelling and simulation tools and embedded systems through a new interface definition (eFMI)

Physical modeling tools

SimulationX
Powered by ITI

Dymola

AMESim

MapleSim
Advanced System-Level Modeling

OM

efmi
10%

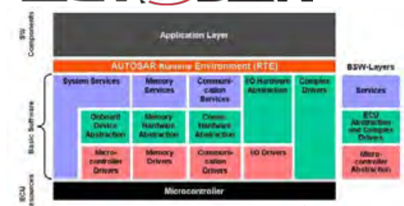
Classical
fmi

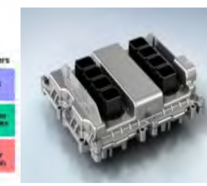
ECU-code generators

ETAS
dSPACE

Classical ECU ASW
90%

AUTOSAR





Additional numerical functions

AUTOSAR BSW numerical functions

Offline and HiL simulation of SW and plant models

efmi fmi fmi

Seamless model-based design of ECU-Software based on physical models.

EMPHYSIS

Sponsored by the Federal Ministry of Education and Research

Embedded Systems Real-time Control Code Generation Using OpenModelica

Martin Sjölund et al
Dept Computer and Information Science
Linköping University

OpenModelica Code Generators for Embedded Real-time Code

- A **full-fledged** OpenModelica-generated source-code FMU (Functional Mockup Unit) code generator
  - Can be used to **cross-compile FMUs** for platforms with more available memory.
  - These platforms can **map** FMI inputs/outputs to analog/digital I/O in the importing FMI master.
- A very **simple code generator** generating a **small footprint** statically linked executable.
  - Not an FMU because there is no OS, filesystem, or shared objects in microcontrollers.

Use Case: SBHS (Single Board Heating System)

Single board heating system (IIT Bombay)

- Use for teaching basic control theory
- Usually controlled by serial port (set fan value, read temperature, etc)
- OpenModelica can generate code targeting the ATmega16 on the board (AVR-ISP programmer in the lower left). Program size is 4090 bytes including LCD driver and PID-controller (out of 16 kB flash memory available).

Thanks for Listening!