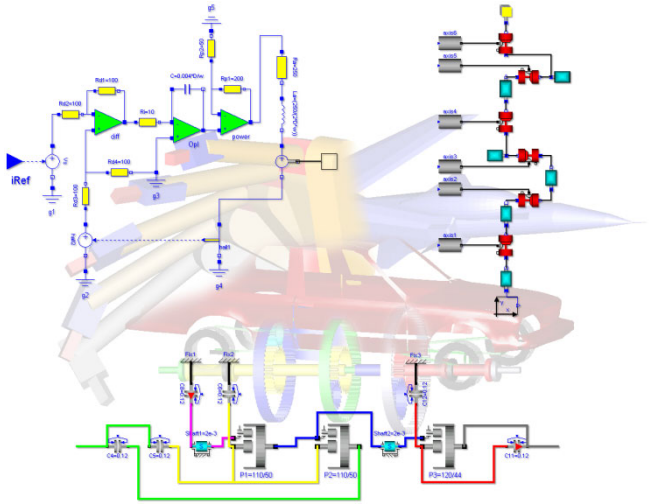
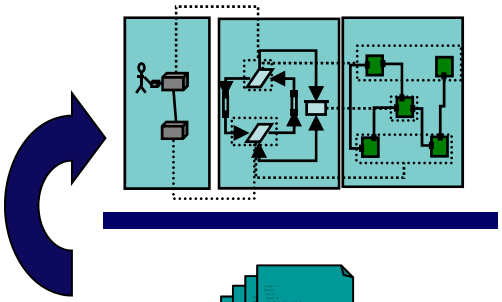
Research in Model-Based Product Development at PELAB in the MODPROD Center

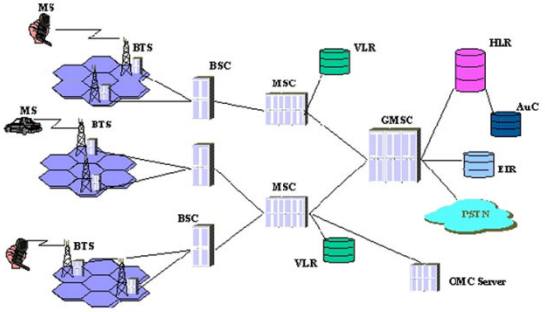
Presentation at MODPROD'2019
Department of Computer and Information Science
Linköping University

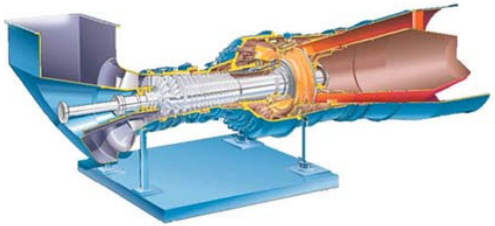
2019-02-05

Peter Fritzson, et al



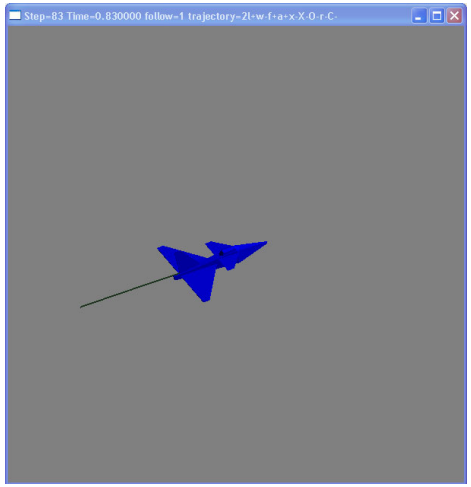


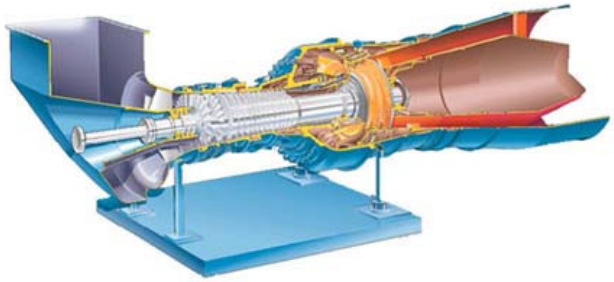


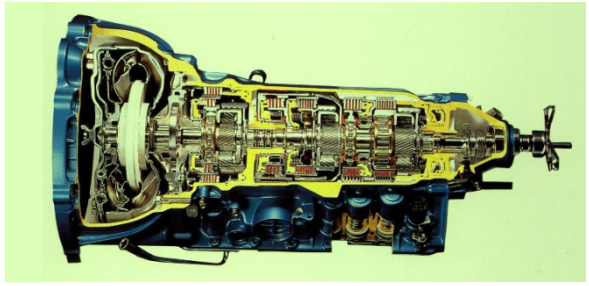


Examples of Complex Systems in Engineering

- Robotics
- Automotive
- Aircraft
- Mobile Phone Systems
- Business Software
- Power plants
- Heavy Vehicles
- Process industry

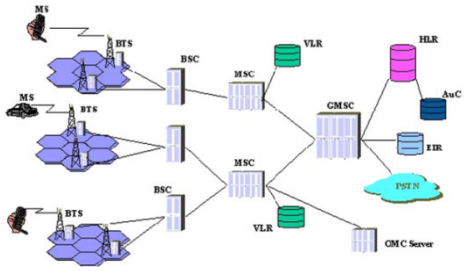






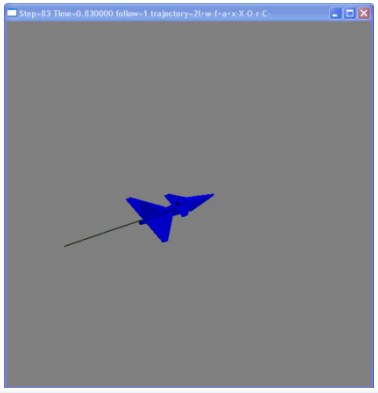
MODELICA pelab

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!



MODELICA pelab

Research

Modeling-Language Design

Model-Based Co-simulation with FMI and TLM

Model Debugging

Model-Based Fault Analysis

Multi-Core based Simulation

Embedded System Real-Time Modeling

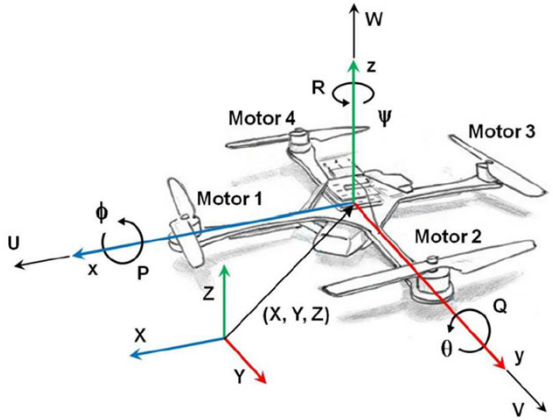
Modeling Support Environments

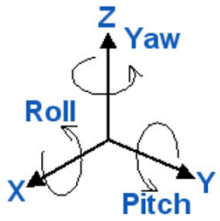
Digital Twins using Modelica and OpenModelica

Collaboration with
Modelicon InfoTech, Bangalore, India

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

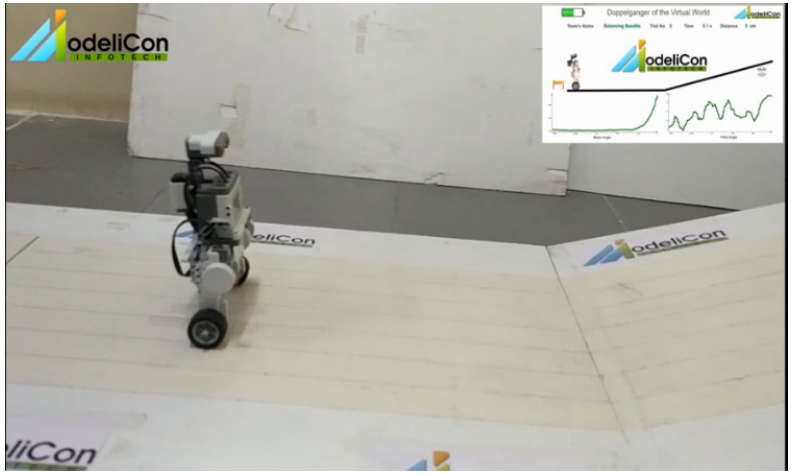
- UAV control and simulation
- Walking 2-wheel robot





UAV
Movie demo

All models and control software done using OpenModelica!



Walking 2-wheel Robot,

Movie demo

Talk Wednesday Afternoon!

MODELICA pelab

Large-Scale, High Performance Model-Based Development

Per Östlund, Adrian Pop, Martin Sjölund,

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
- Status January 2019, OMC with newfrontend simulates more than 95% of Modelica Standard Library
- The New frontend is about **10 to 100 times faster** than the old compiler frontend.

- Scientific paper accepted to the International Modelica Conference, Regensburg, March, 2019

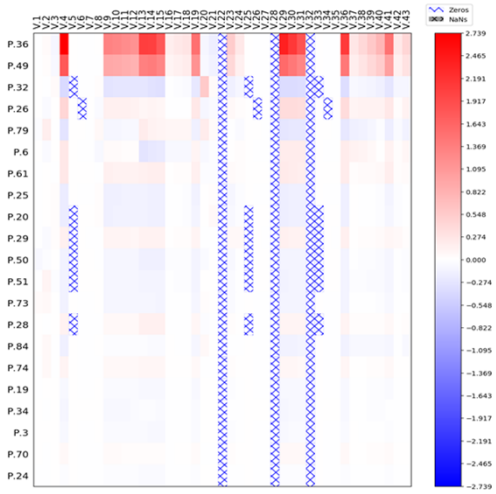
Simultaneous Param-based Sensitivity Analysis and Robust Optimization (collaboration with Univ. Buenos Aires)

- To define a sensitivity experiment:
  - The state variable to analyze
  - The set of parameters to perturb
  - The allowed perturbation intervals for each parameter

Paper published at EOOLT 2017 (prototype)

Planned OpenModelica Release spring 2019

- 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



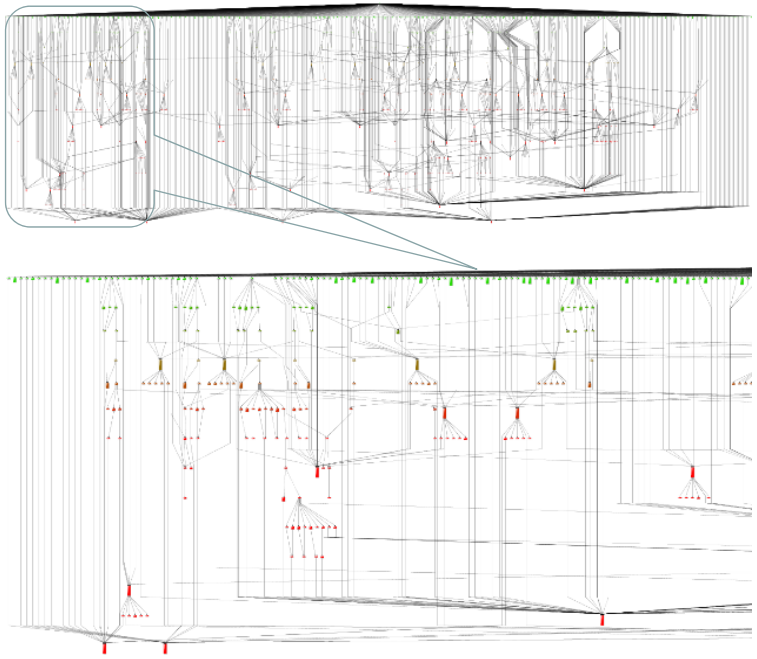
MODELICA pelab

Parallel Execution Compilation to MultiCore

Mahder Gebremedhin, Peter Fritzson

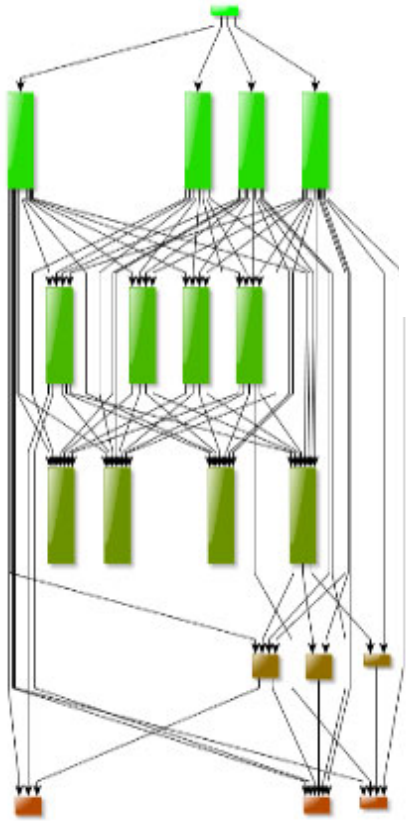
Compiling Models to Efficient Parallel Code
(scheduling on multiple cores)

Modelica.Electrical.Spice3.Examples.Spice3BenchmarkFourBitBinaryAdde



Original task system of Four Bit Binary Adder model

1122 Tasks
1360 Edges



Task system after clustering for level scheduler

18 Tasks
72 Edges

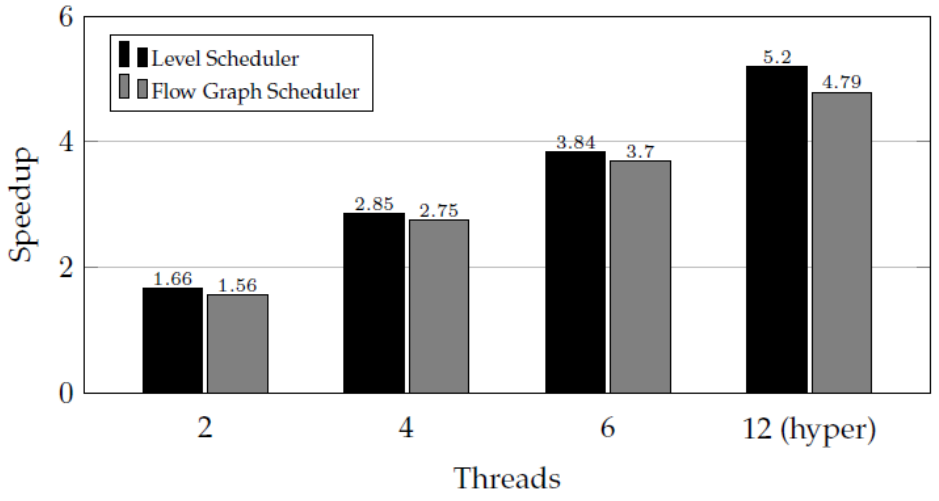
ParModAuto Parallelization (Release spring 2019)

Automatic AutoTuned Parallelization of Equation-based Models

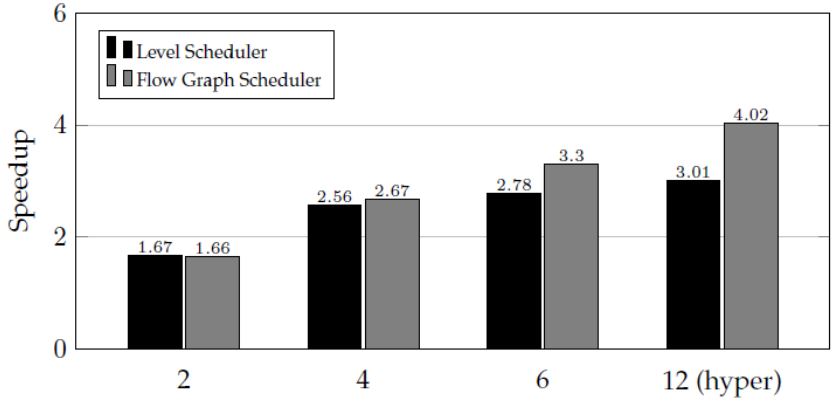
- Mahder Gebremedhin's PhD defended Jan 24, 2019

- Automatic **Parallelization**
- Automatic **clustering** of small tasks
- **Automatic load balancing** based on measurements, automatically adapts to changing load
- **Shared-memory** task parallelization
- Planned for OpenMOdelica release spring 2019

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



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



MODELICA pelab

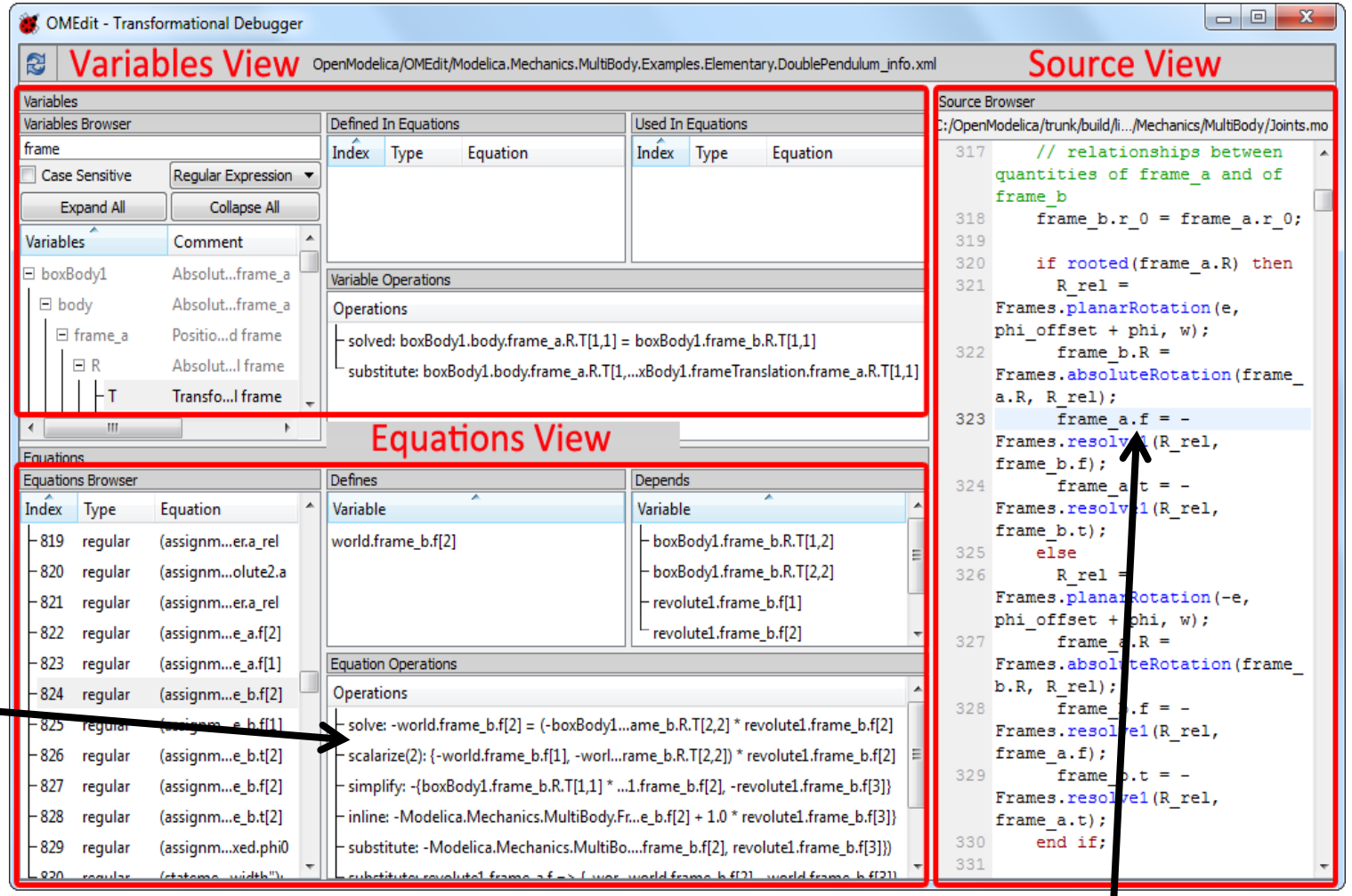
Model Debugging and Performance Analysis

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

Integrated Static-Dynamic OpenModelica Equation Model Debugger

Efficient handling of **Large** Equation Systems

Showing equation transfor mations of a model:



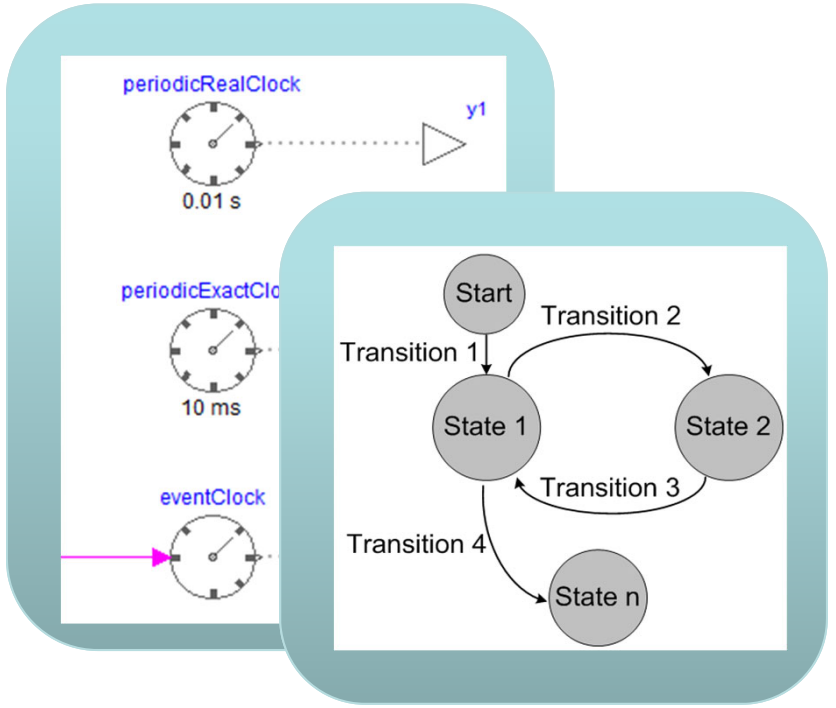
Mapping dynamic run-time error to source model position

MODELICA pelab

Ongoing Research on Debugging

Debugging of new features

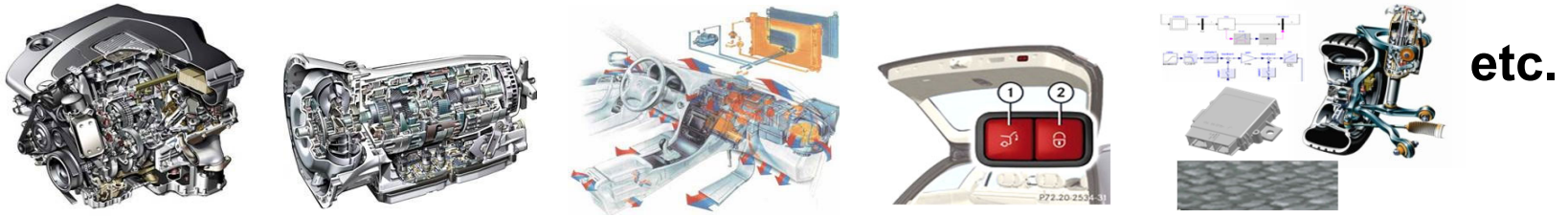
- **clocked** synchronous models
- **real-time debugging** and event tracing
- graphic support for **state machine** debugging



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)

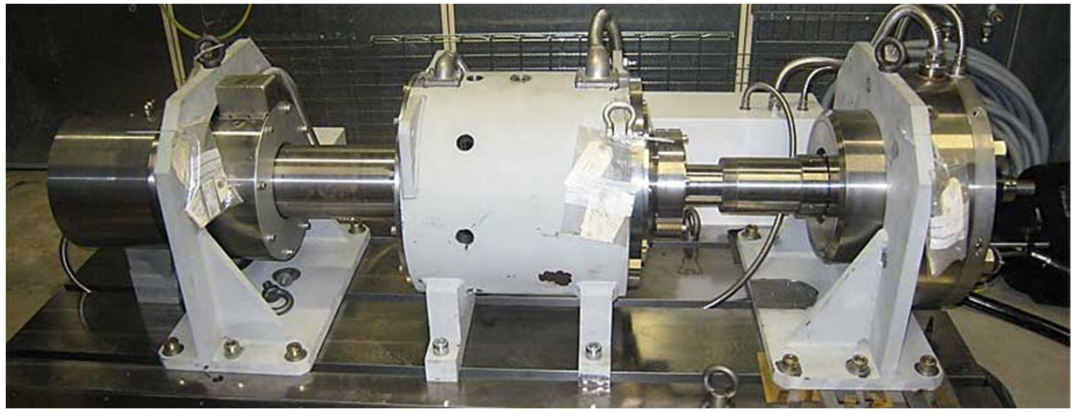


courtesy Daimler

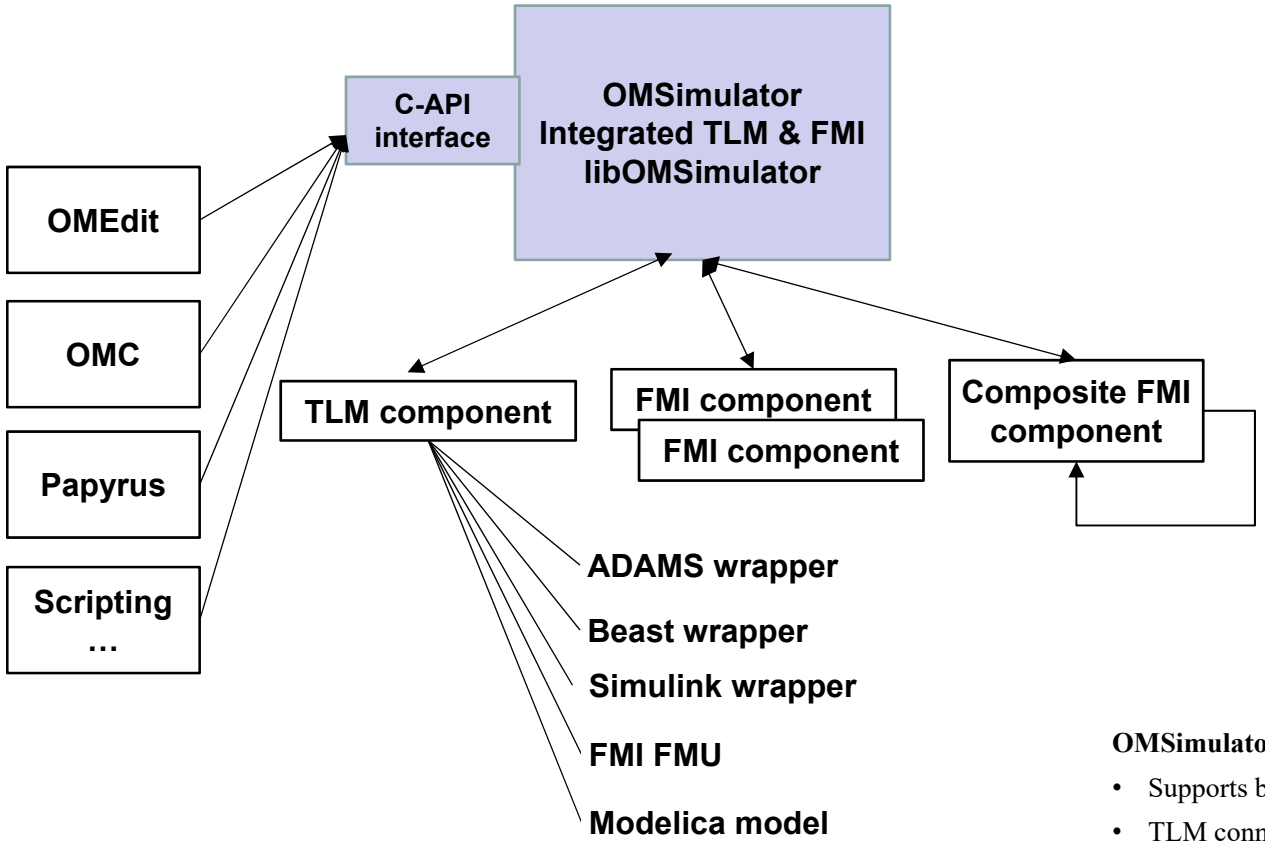
- FMI development was started by ITEA2 MODELISAR project. FMI is a Modelica Association Project now
- **Version 1.0**
- FMI for Model Exchange (released Jan 26,2010)
- FMI for Co-Simulation (released Oct 12,2010)
- **Version 2.0**
- FMI for Model Exchange and Co-Simulation (released July 25,2014)
- **>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

Enhanced FMI Co-simulation, Run-time, and Master Simulation Tool – Research in OPENCPS Project

- 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 – Integrated FMI and TLM-based Cosimulator/Simulator – part of OpenModelica



Main Framework Aspects

Unified co-simulation/simulation tool

- FMI 2.0 (model exchange and co-simulation)
- TLM (transition line modelling)
- Real-time and offline simulation

Standalone open source simulation tool with rich interfaces

- C/Java
- Scripting languages Python, Lua

Co-simulation framework as a solid base for engineering tools

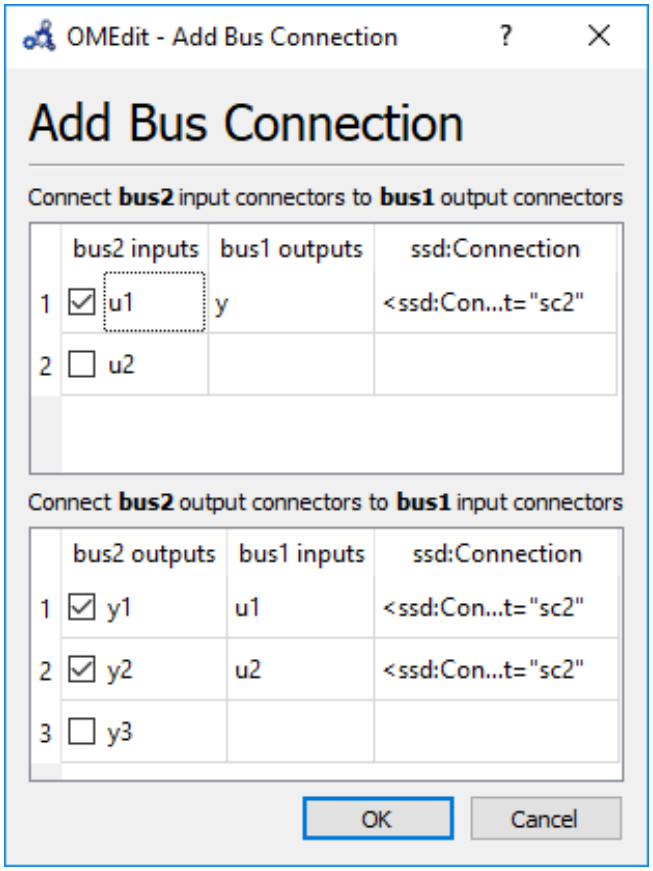
- Integration into OpenModelica/Papyrus
- Open for integration into third-party tools and specialized applications (e.g. flight simulators, optimization)

OMSimulator in OpenModelica 1.13.0

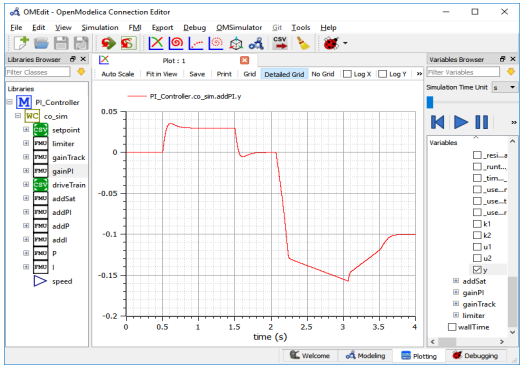
- Supports both FMI and TLM
- TLM connections are optional
- Co-simulation to multiple tools
- Composite model editor
- External API interface and scripting

OMSimulator Simulation, SSP, and Tool Comparison

Adding SSP bus connections



FMI Simulation results in OMEdit



FMI Simulation Tool Comparison

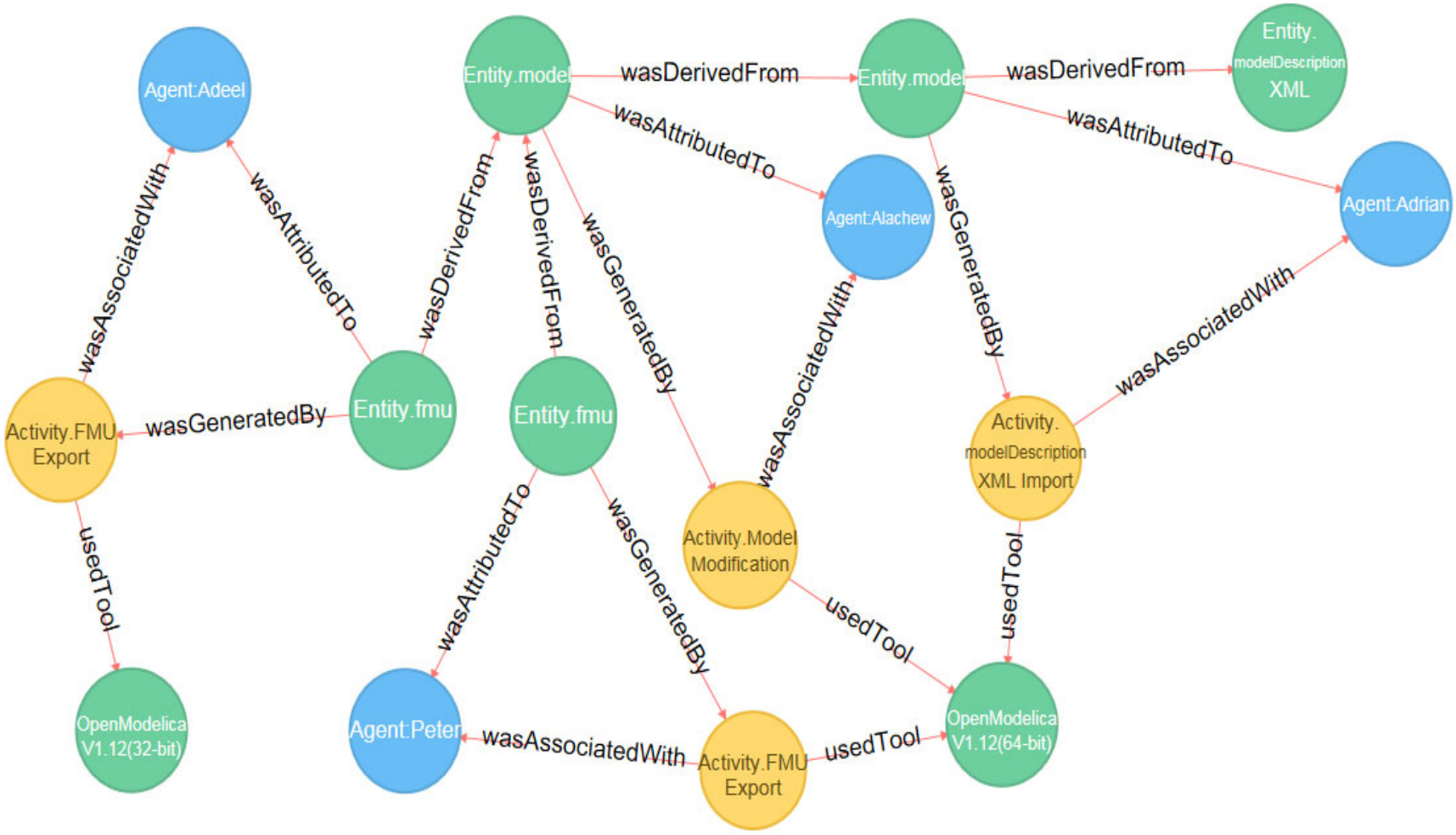
	OMSimulator	DACCOSIM	Simulink	PyFMI
Commercial	No	No	Yes	No
Open-source	OSMC-PL, GPL	AGPL2	No	LGPL
Lookup Table	Yes	Yes	Yes	No
Alg. Loops	Yes	Yes	No	Yes
Scripting	Python, Lua	proprietary	proprietary	Python
GUI	Yes	Yes	Yes	No
SSP	Yes	No	No	No
platform	Linux/Win/macOS	Linux/Win	Linux/Win/macOS	Linux/Win/macOS

	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

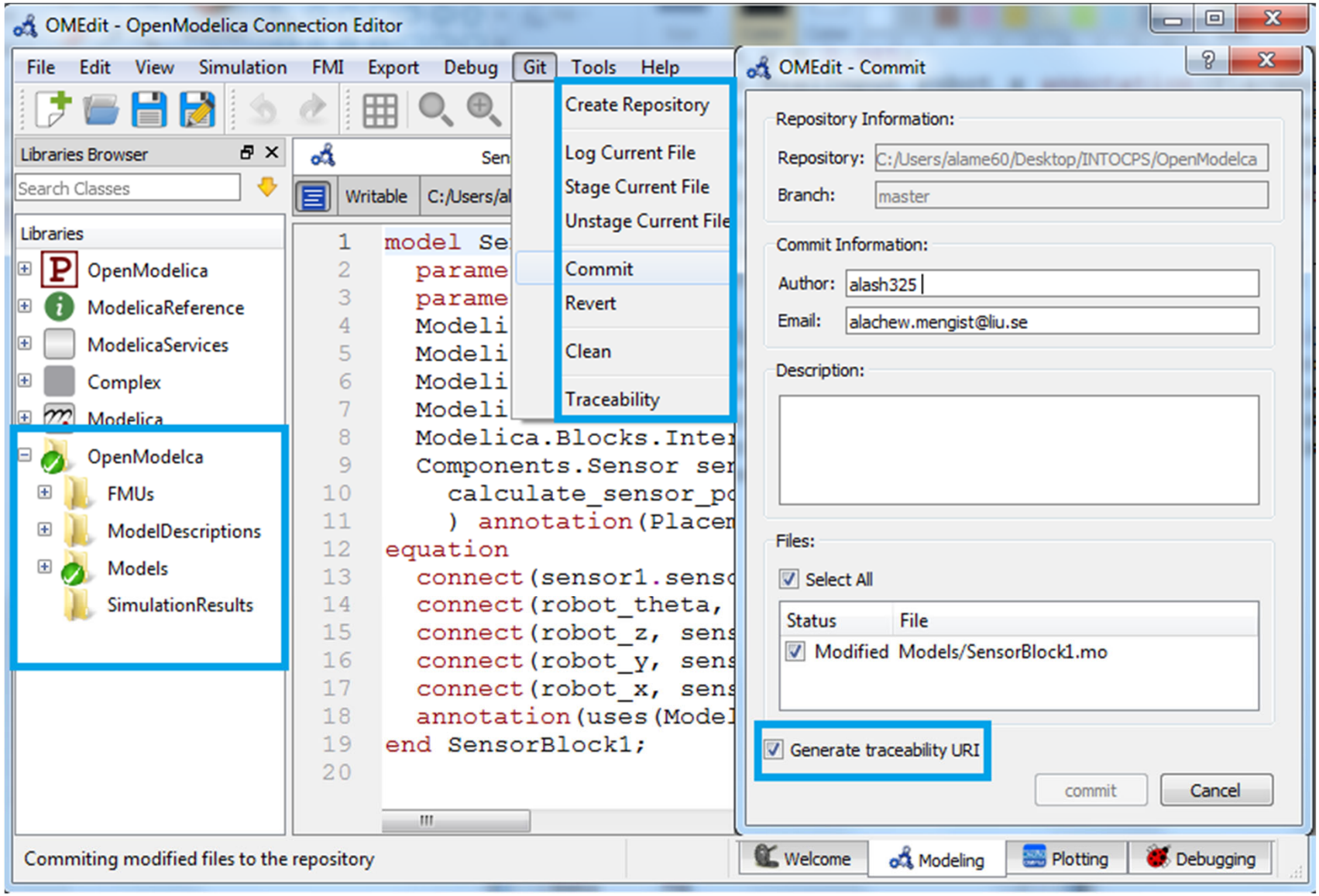
Model Management and Traceability

Adrian Pop, Alachew Mengist, Peter Fritzson

Traceability Information collected by OpenModelica

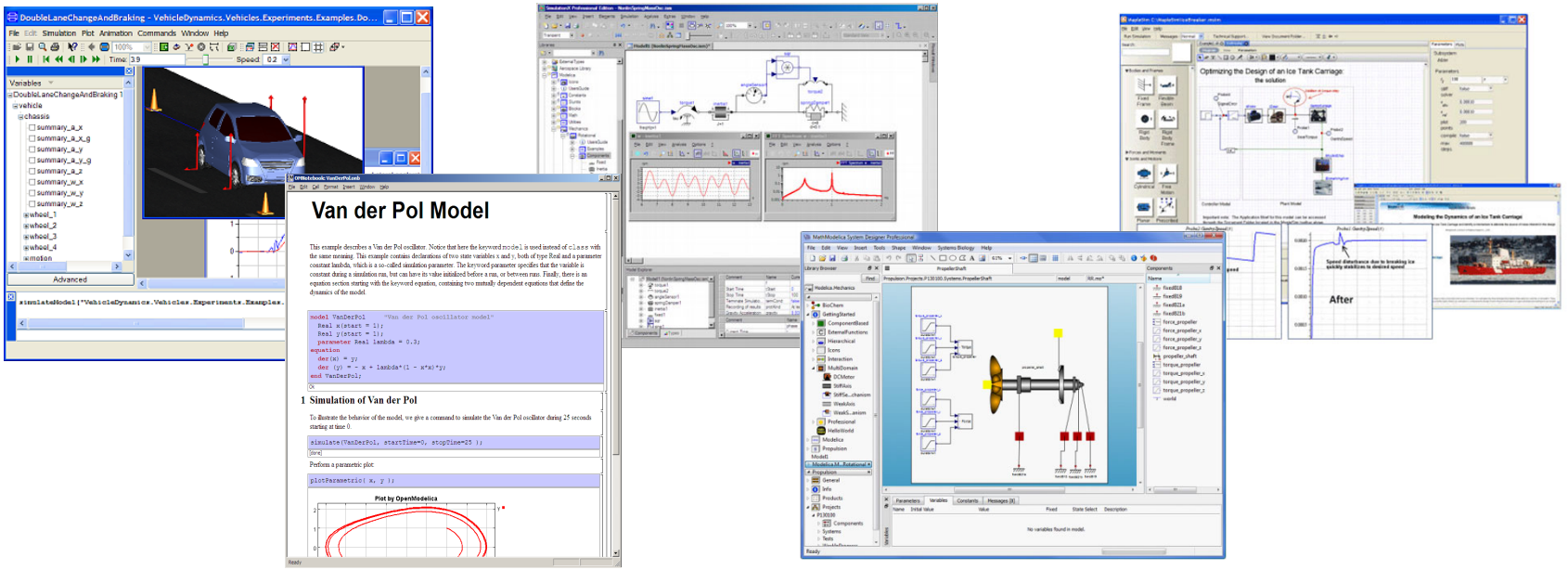


Model Management with Git Integration



Dynamic Verification/Testing of Requirements vs Usage Scenario Models

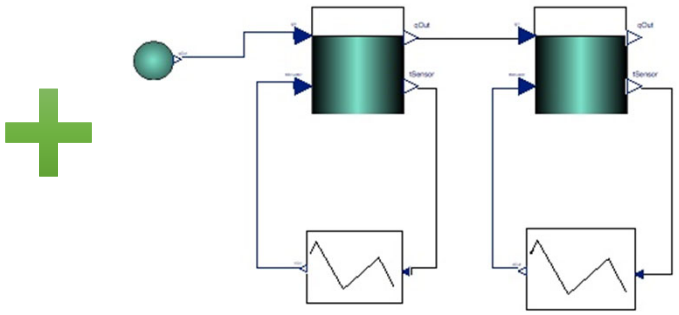
Lena Buffoni et al



Testing a single verification model in Modelica

- 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.
- …

Design alternative:
two tank model



Design alternative:
two tank model

Start with constant flow and increase at t=150

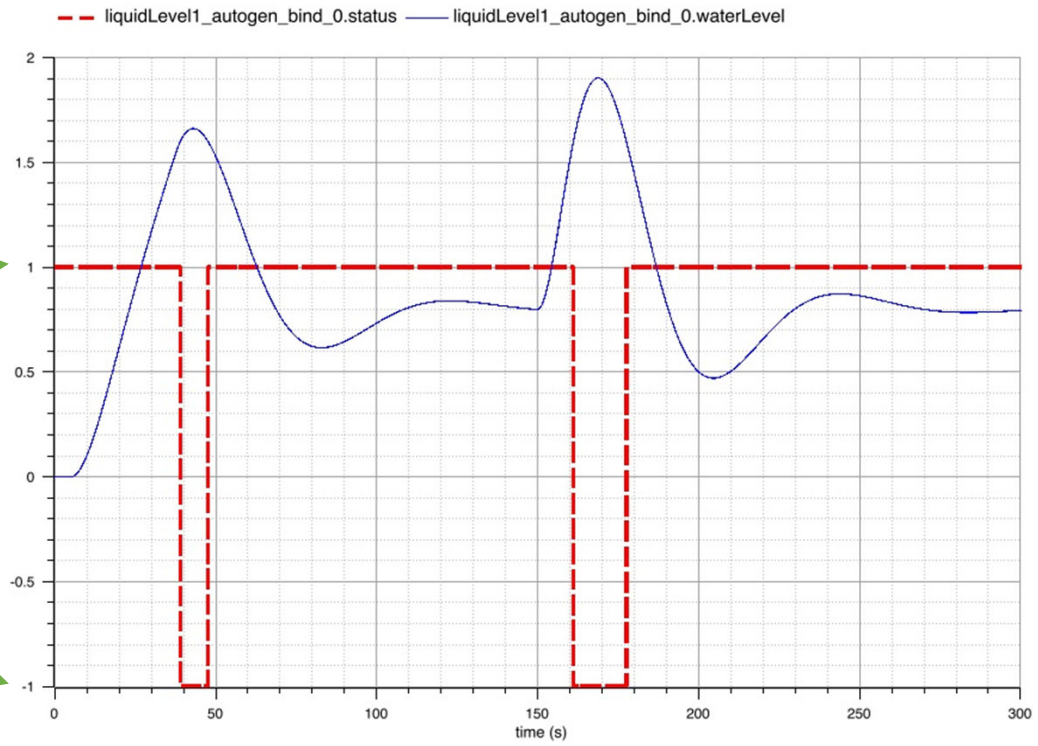
One possible test scenario

Analyzing a single requirement status

Req. 002: The level of liquid in a tank shall never exceed 80% of the tank height.

Requirement not violated

Requirement violated



Model-based Development Tooling for Embedded Systems

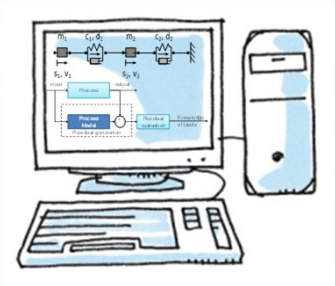
ITEA3 project EMPHYSIS

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

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



SIMULATION X 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)

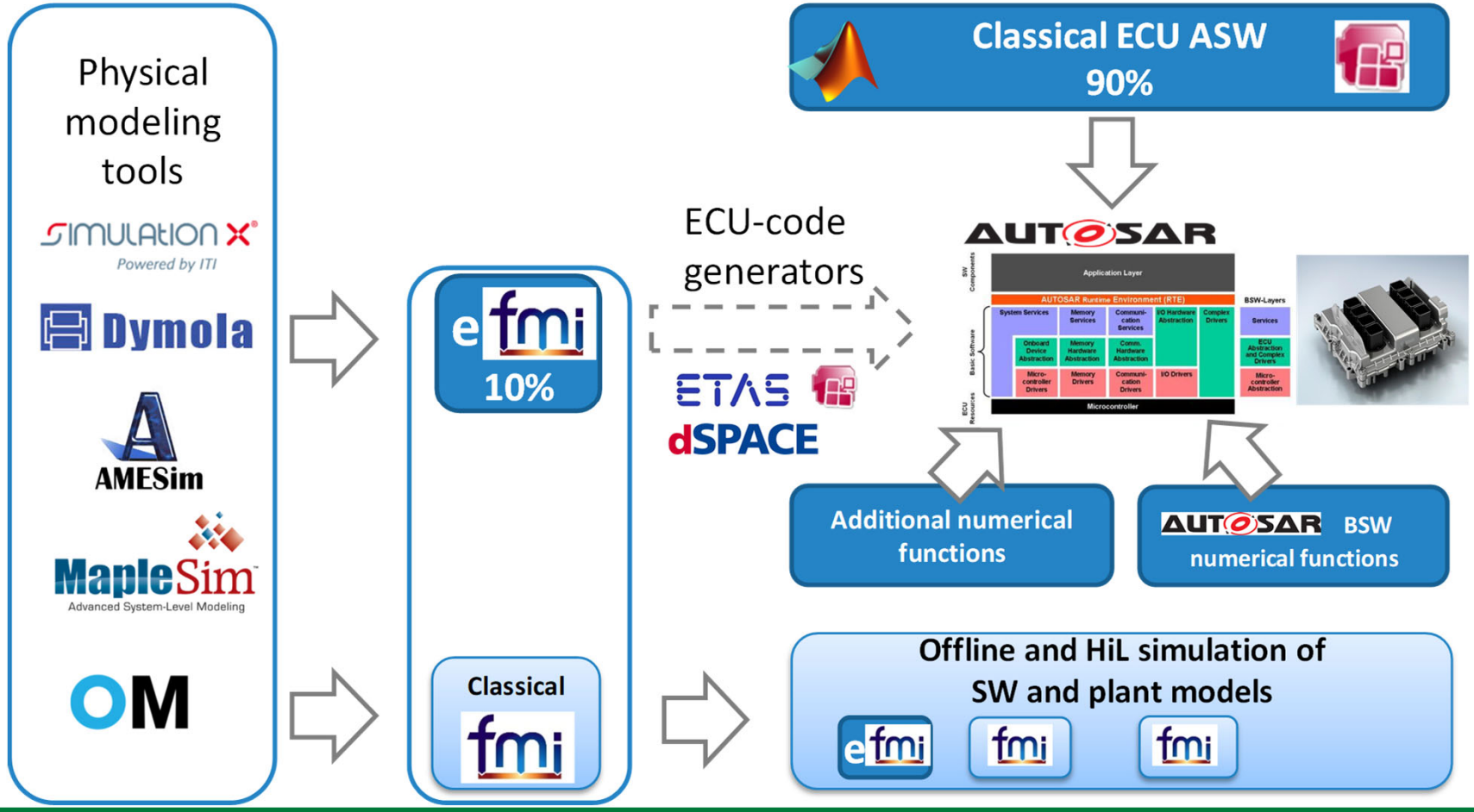
THE C PROGRAMMING LANGUAGE
ASCET

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

EMPHYSIS

SPONSORED BY THE Federal Ministry of Education and Research

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



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

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!