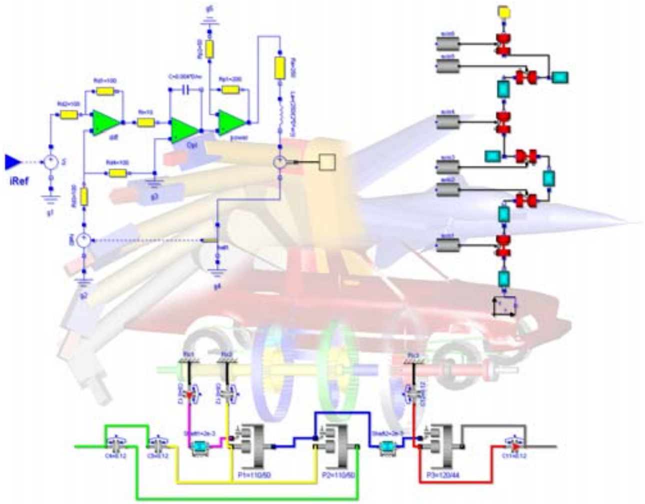
Research in Model-Based Product Development at PELAB in the MODPROD Center

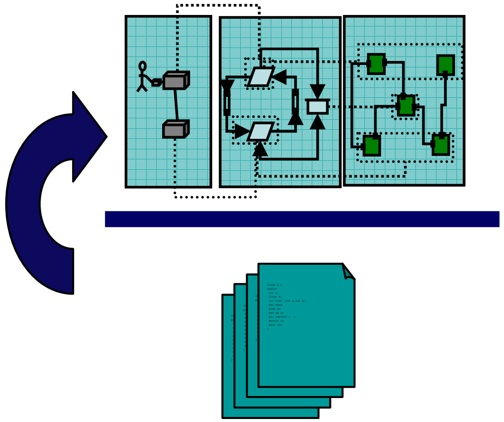
Presentation at MODPROD'2010

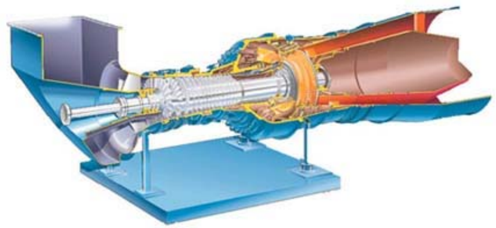
PELAB – Programming Environment Laboratory
Department of Computer and Information Science
Linköping University

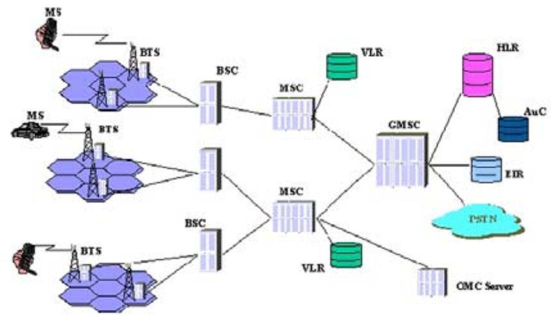
2010-02-09

Peter Fritzson





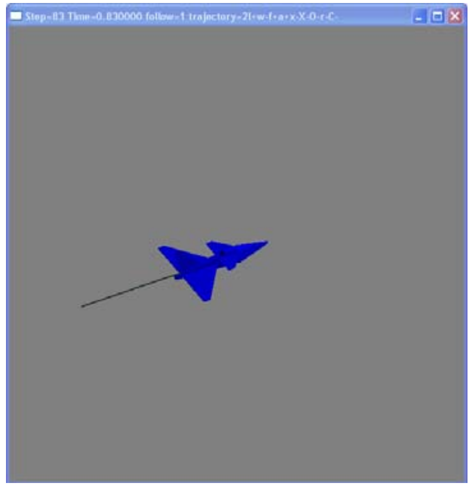




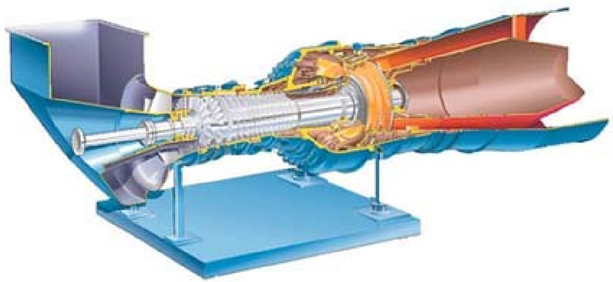
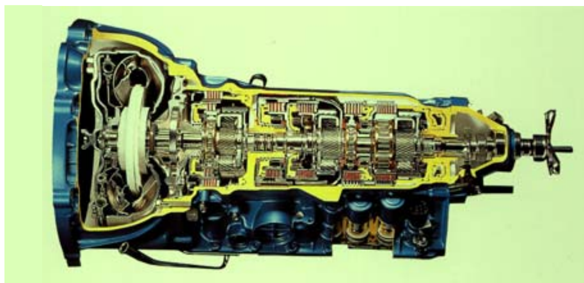
Examples of Complex Systems in Engineering

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

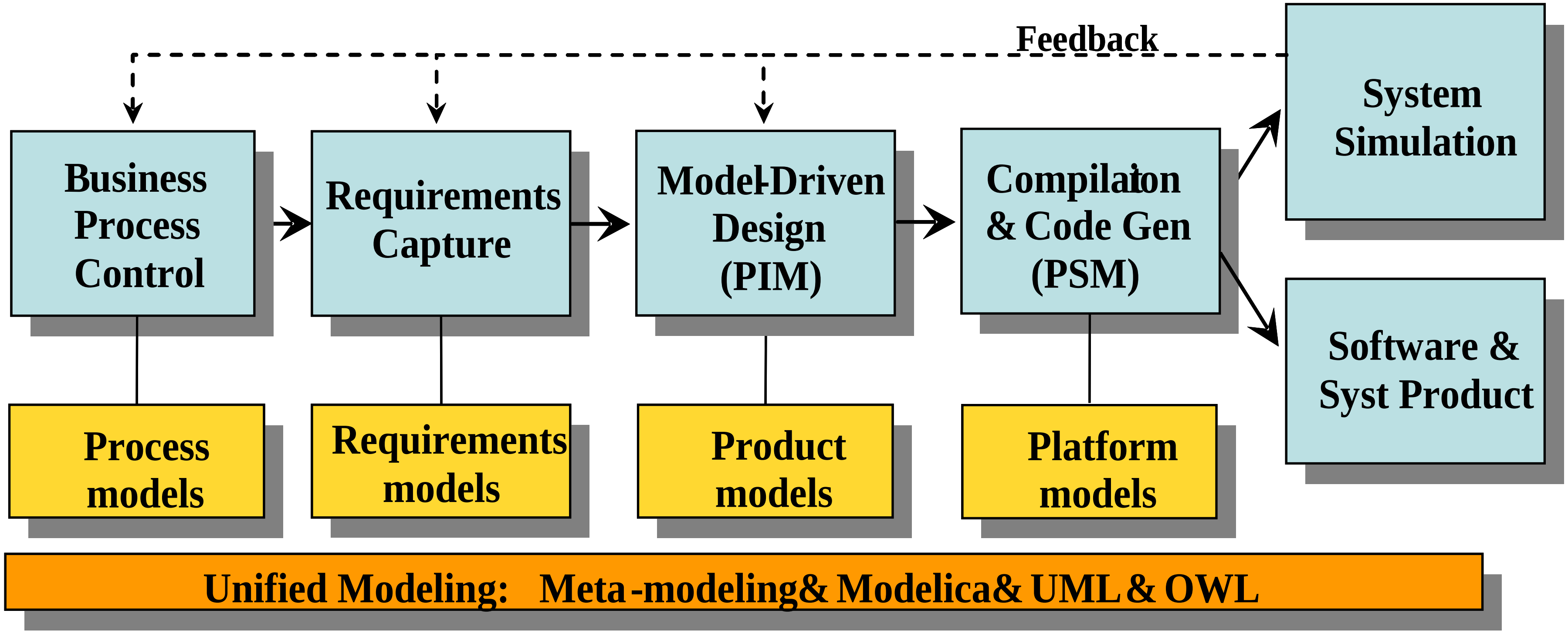






Vision of Integrated Model-Based Product Development



Vision of unified modeling framework for model-driven product development from platform independent models (PIM) to platform specific models (PSM)

Important Questions

- Design of modeling languages for modeling complex (physical) systems (**Modelica**) including precise semantics, extensibility, etc.
- How to engineer complex engineering systems of both **hardware and software** in a consistent and safe manner?
- Compilation of models for efficient (real-time) execution on **multi-core** architectures
- Traceability from **requirements** to models to implementation

Modeling-Language Design

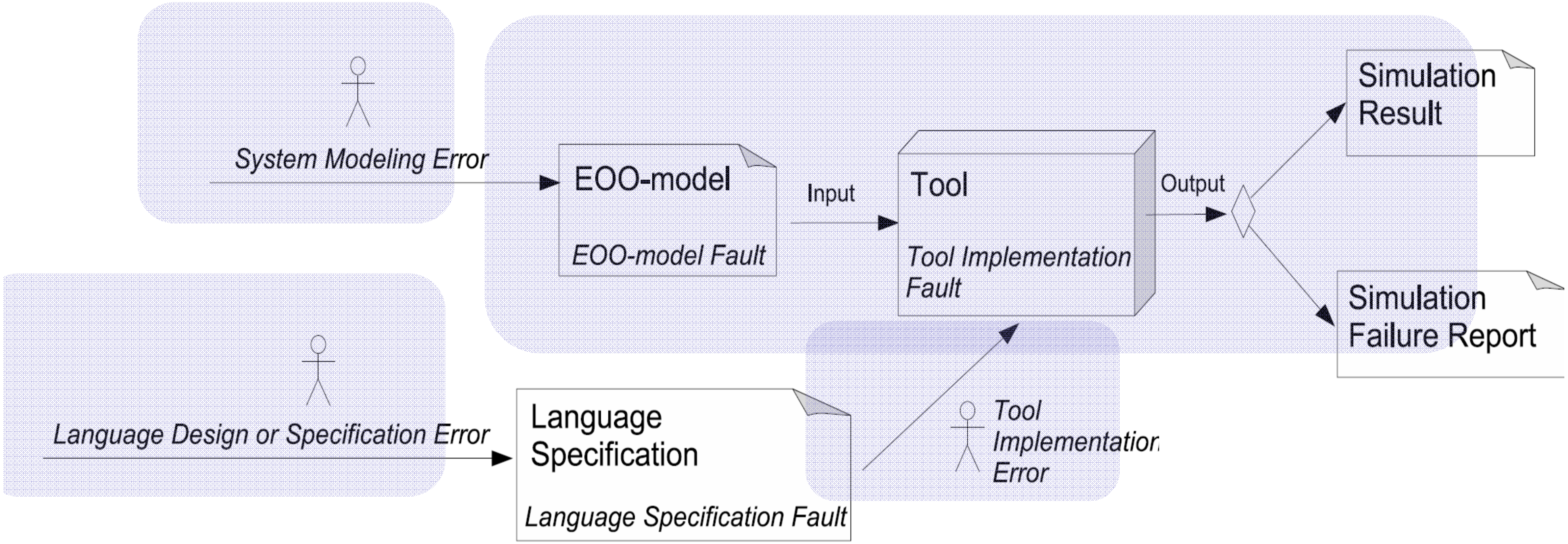
Modeling Support Environments

Modeling Language and Tool Research

- How can a **modeling language be designed** with **precise semantics** to avoid errors?
- Can the language be made extensible?
- Can it model itself (**meta-modeling**)?

- How should a user-supportive modeling environment be designed?

Context of Using of Semantics for Checking



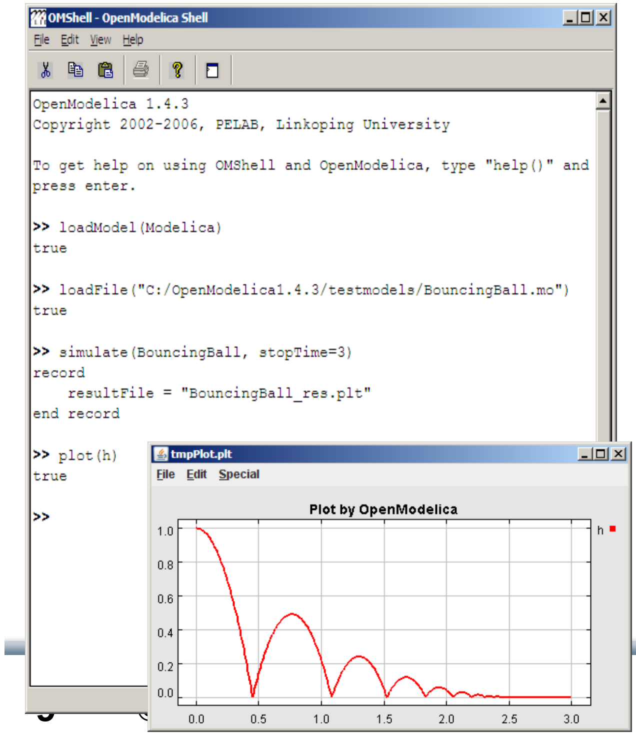
Our Current Semantics Work

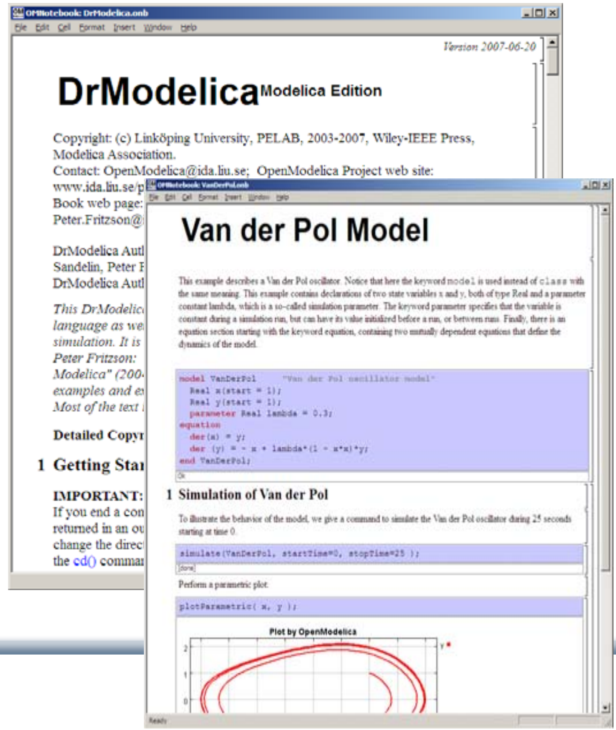
- Defining **Core language MKL** (Modeling Kernel Language) for expressing base semantics
- Other constructs should be expressible in the Core language
- Defining **extensible meta-modeling**/meta-programming language primitives (e.g. MetaModelica)

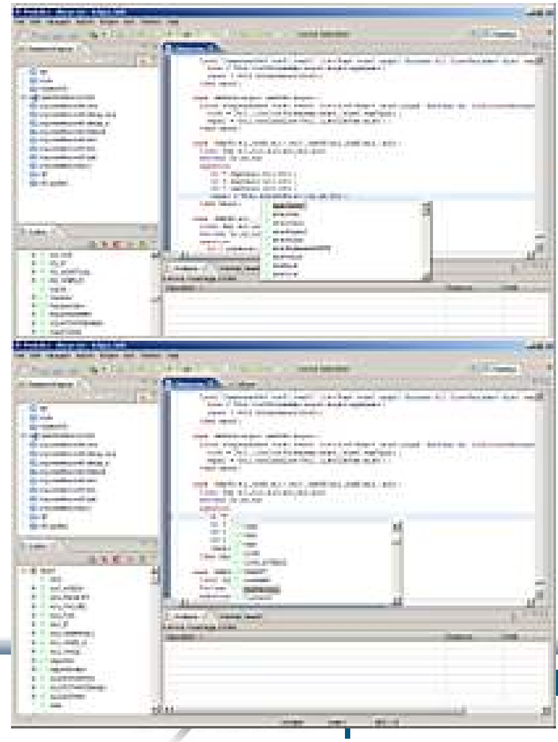
The OpenModelica Open Source Enviroment

www.openmodelica.org

- Advanced Interactive Modelica compiler (OMC)
  - Supports most of the Modelica Language
- Basic environment for creating models
  - OMShell – an interactive command handler   * ModelicaML UML Profile
  - OMNotebook – a literate programming notebook * MetaModelica transforms
  - MDT – an advanced textual environment in Eclipse







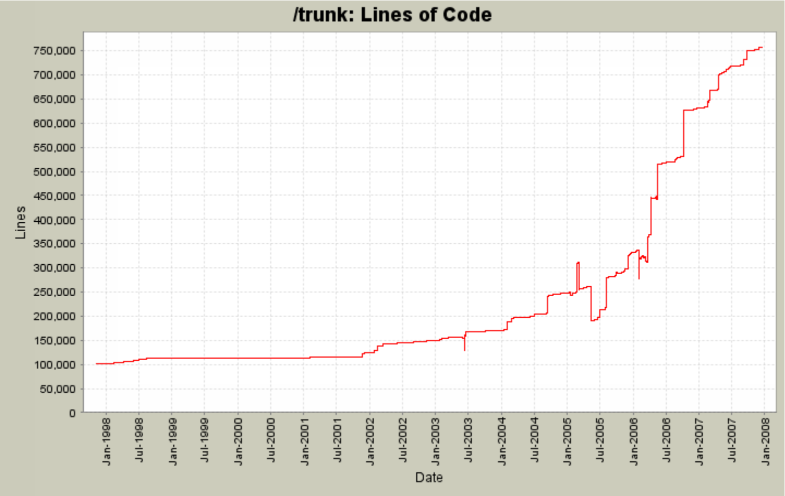
OSMC – International Consortium for Open Source Model-based Development Tools, 28 members Dec 2009

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 (16)

- **ABB Corporate Research**
- **Bosch-Rexroth AG, Germany**
- **Siemens Turbo Machinery AB**
- **CDAC, India**
- **Creative Connections, Prague**
- **Equa Simulation AB, Sweden**
- **Frontway AB**
- **IFP, Paris, France**
- **InterCAX, Atlanta, USA**
- **MostforWater, Belgium**
- **MathCore Engineering AB**
- **MapleSoft, Canada**
- **TLK Thermo, Germany**
- **VI-grade, Italy**
- **VTT, Finland**
- **XRG Simulation AB, Germany**

University members (12)

- **Linköping University, Sweden**
- **Ghent University, Belgium**
- **Hamburg University of Technology/TuTech, Germany**
- **Technical University of Braunschweig, Germany**
- **Université Laval, the modelEAU group, Canada**
- **Griffith University, Australia**
- **University of Queensland, Australia**
- **Politecnico di Milano, Italy**
- **Mälardalen University, Sweden**
- **Technical University Dortmund, Germany**
- **Technical University Dresden, Germany**
- **Telemark University College, Norway**

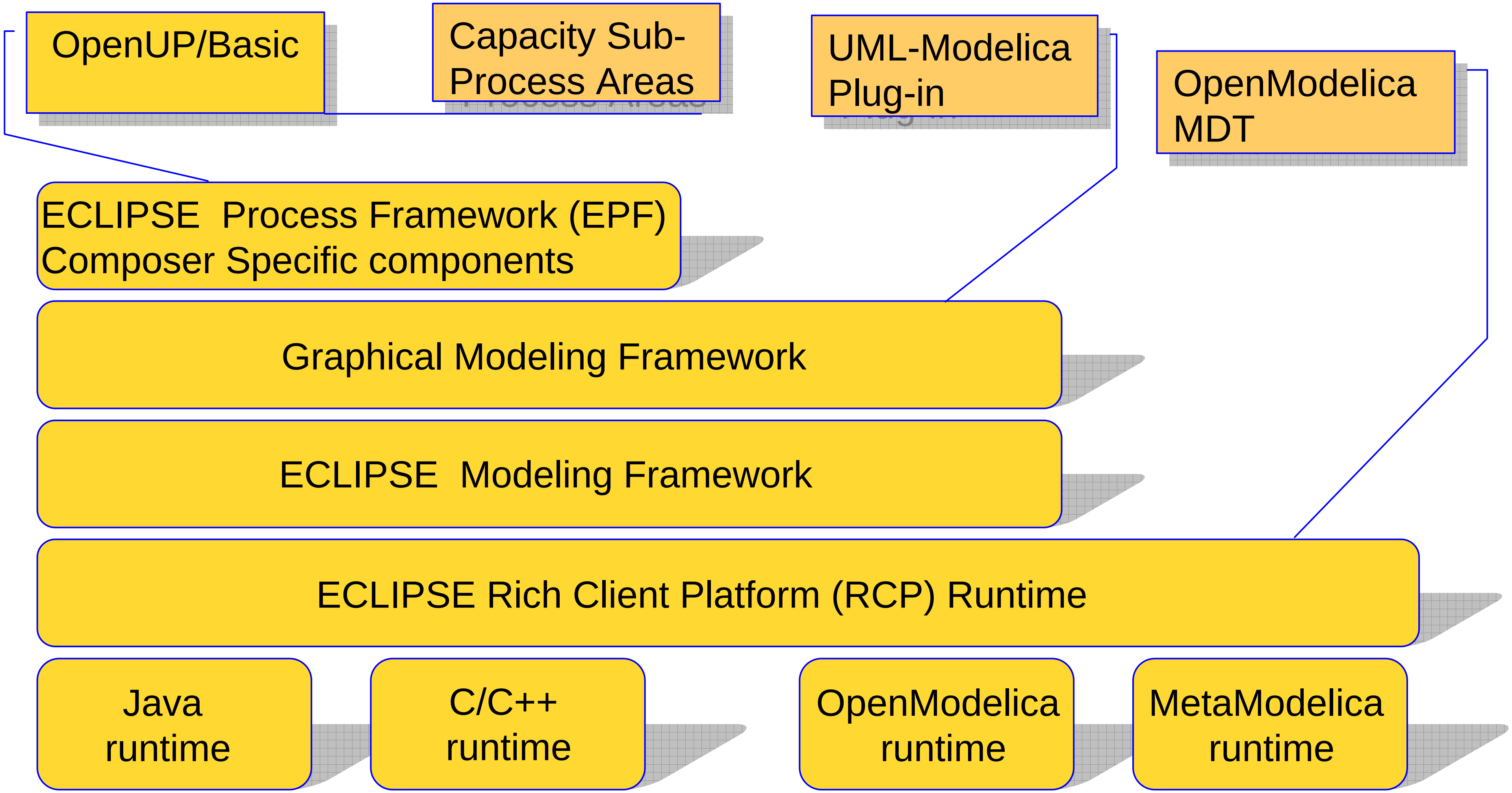
MODELICA pelab

Integrated Hardware-Software Modeling

ModelicaML
UML Profile for Modelica

SysML-Modelica Integration

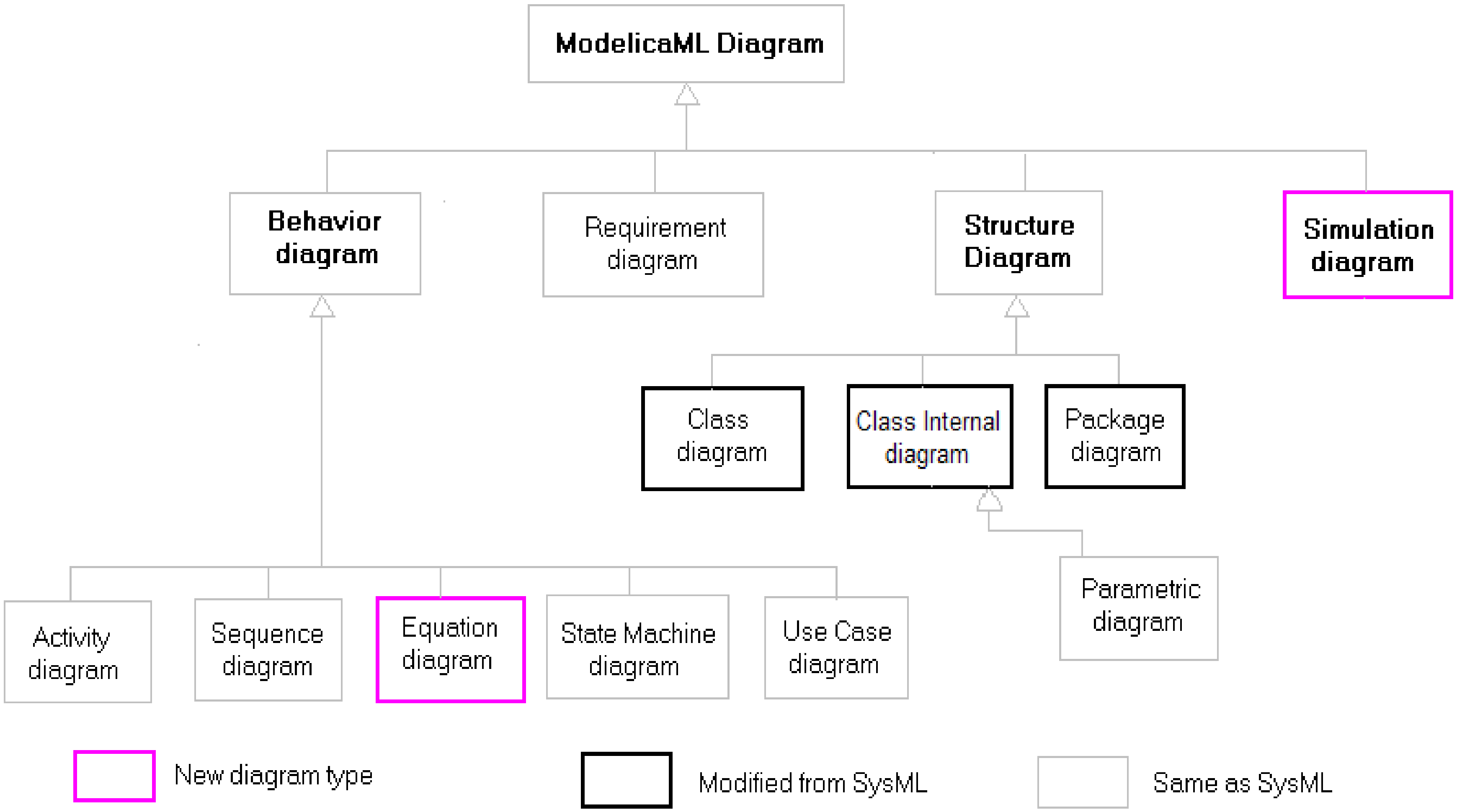
Using ECLIPSE as Integration Platform



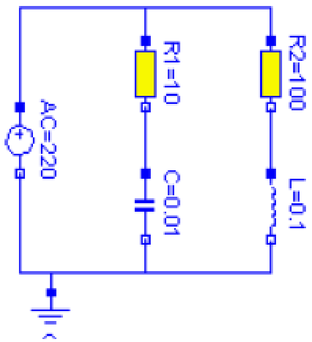
ModelicaML – UML Profile for Modelica 1st Generation

- Extension of SysML subset
- Features:
  - Supports Modelica constructs
  - Modelica generic class modeling
  - Modelica syntax in definitions
  - Equation-based modeling
  - Simulation modeling

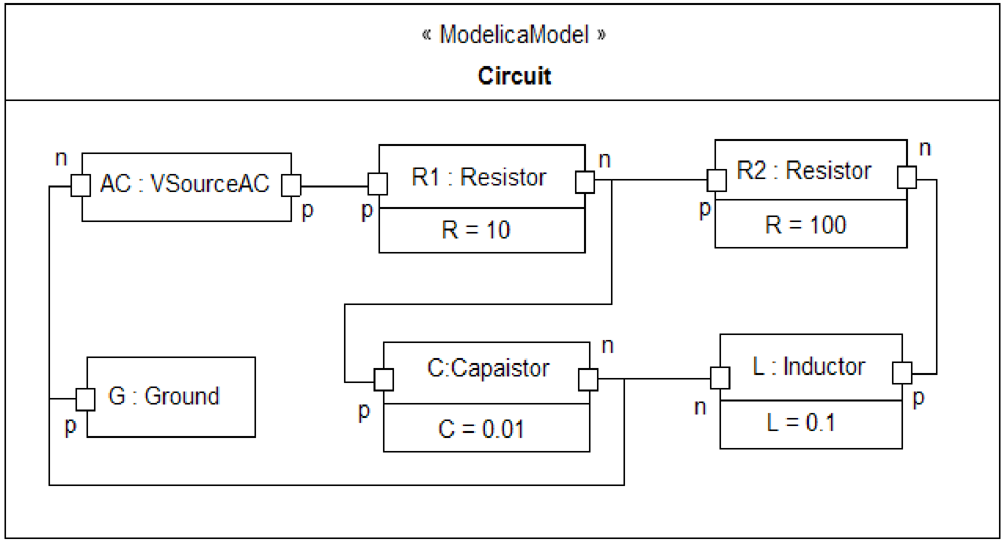
ModelicaML Diagrams – Overview



ModelicaML Class Internal Diagram



versus



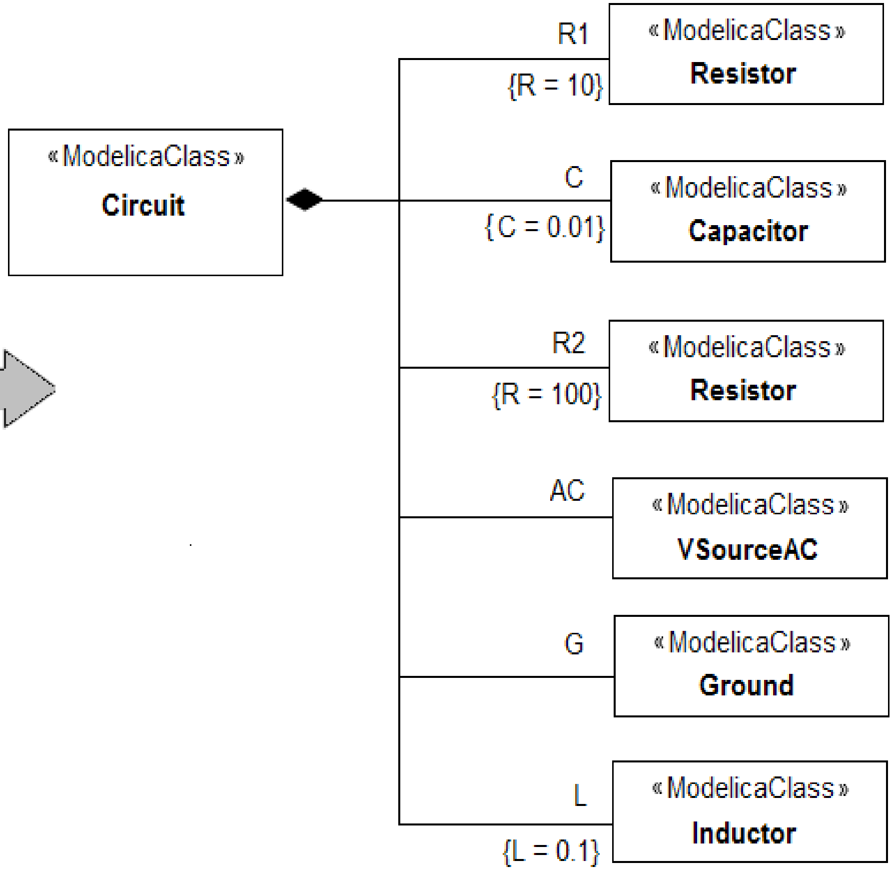
- Modelica Connection diagram
  - Better visual comprehension
  - Predefined connector locations
- Class Internal diagram
  - Nested models
  - Top-model parameters and variables
  - Flow direction
  - Other ModelicaML elements

ModelicaML Class Diagram

- Example

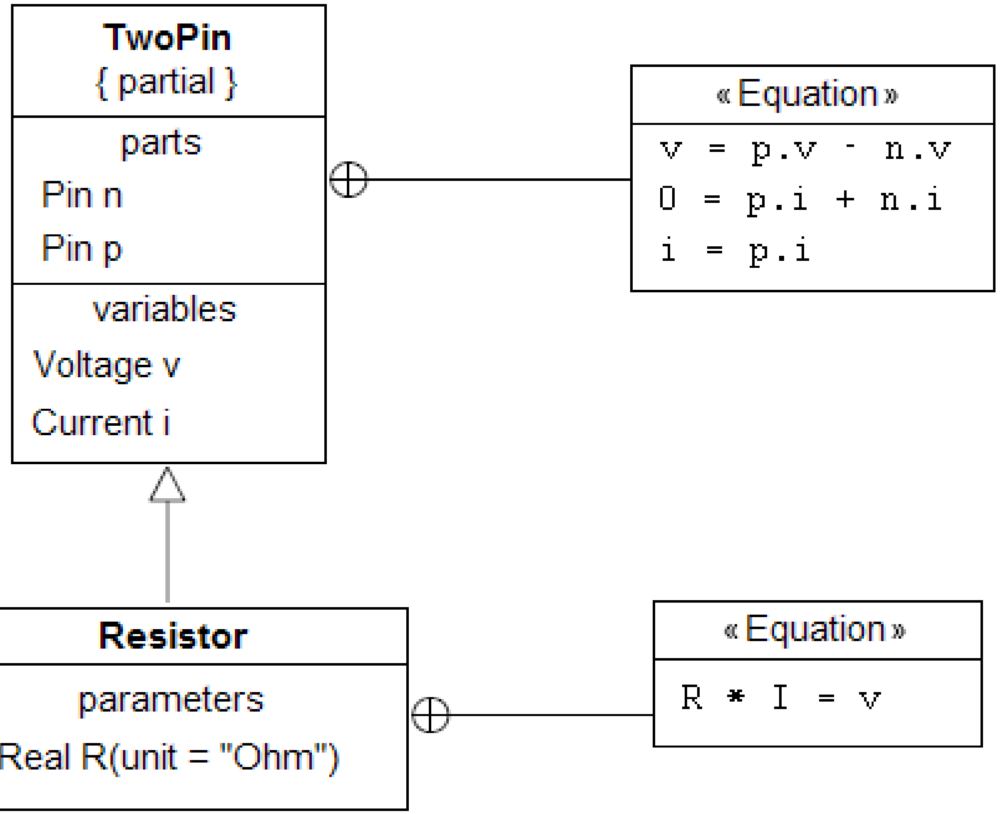
```
model circuit
  Resistor  R1(R=10);
  Capacitor C(C=0.01);
  Resistor  R2(R=100);
  Inductor  L(L=0.1);
  VsourceAC AC;
  Ground    G;

equation
  connect (AC.p, R1.p);
  connect (R1.n, C.p);
  connect (C.n, AC.n);
  connect (R1.p, R2.p);
  connect (R2.n, L.p);
  connect (L.n,  C.n);
  connect (AC.n, G.p);
end circuit;
```

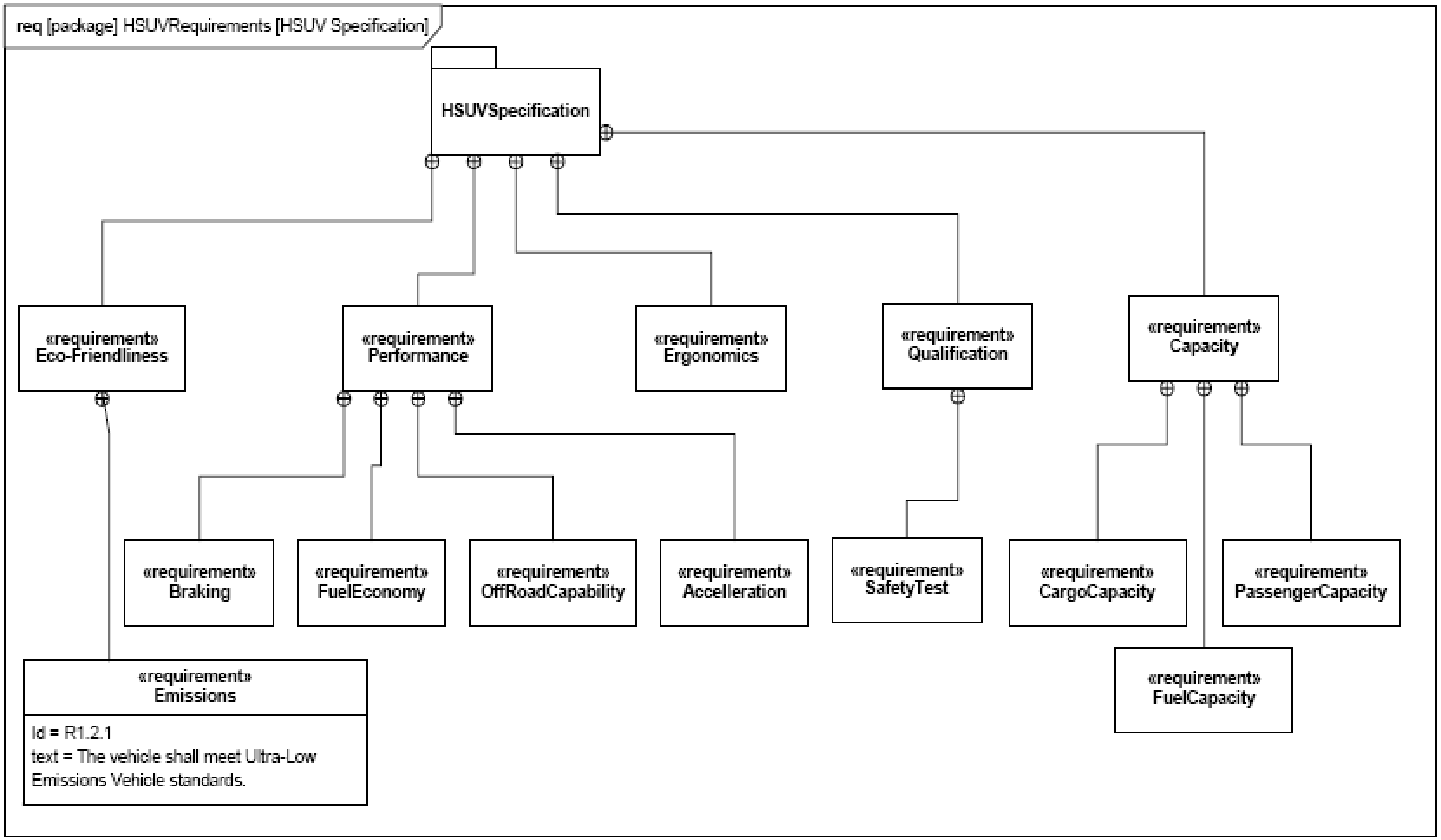


Equations Diagram

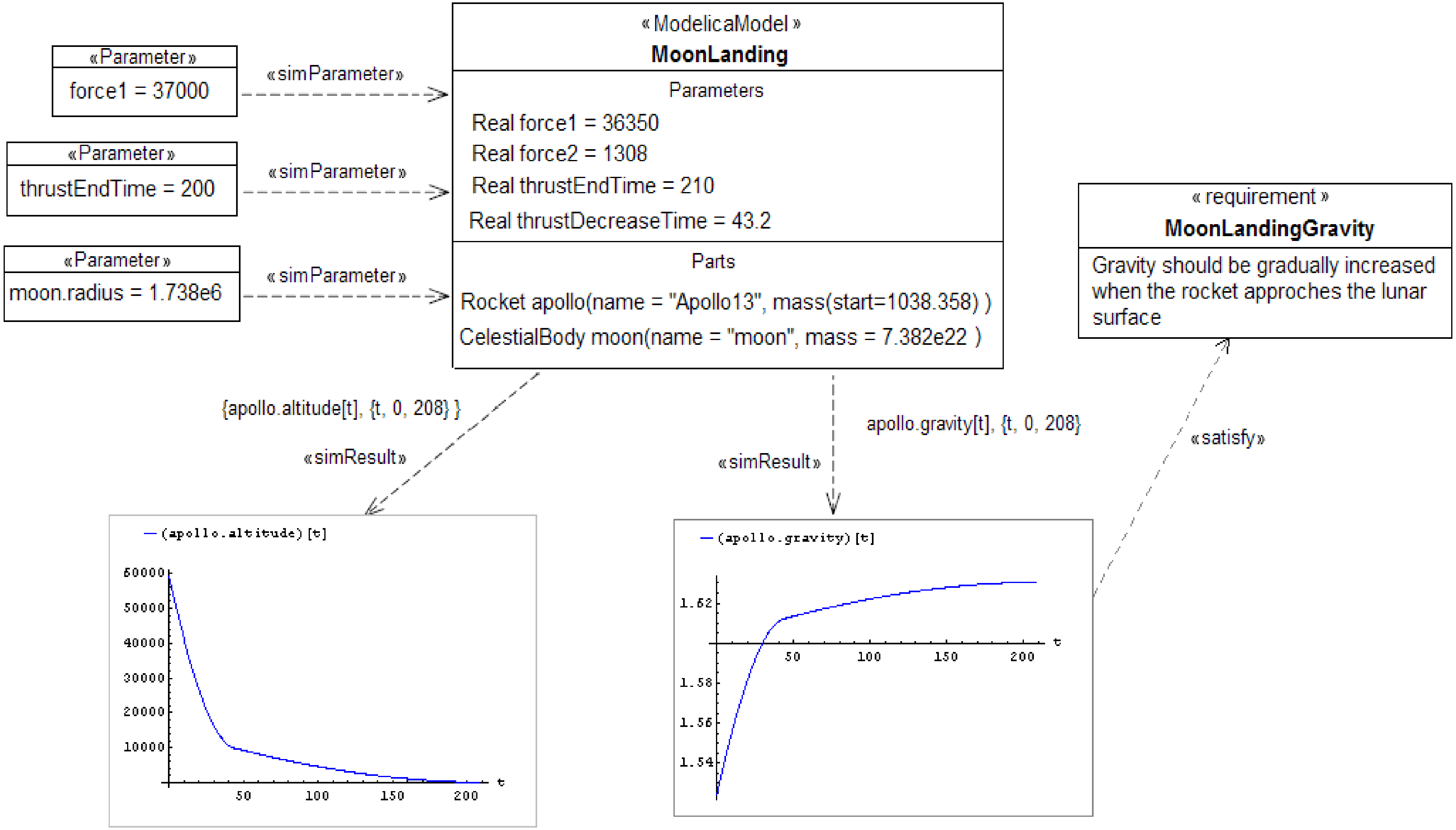
- Example



Requirements Diagram



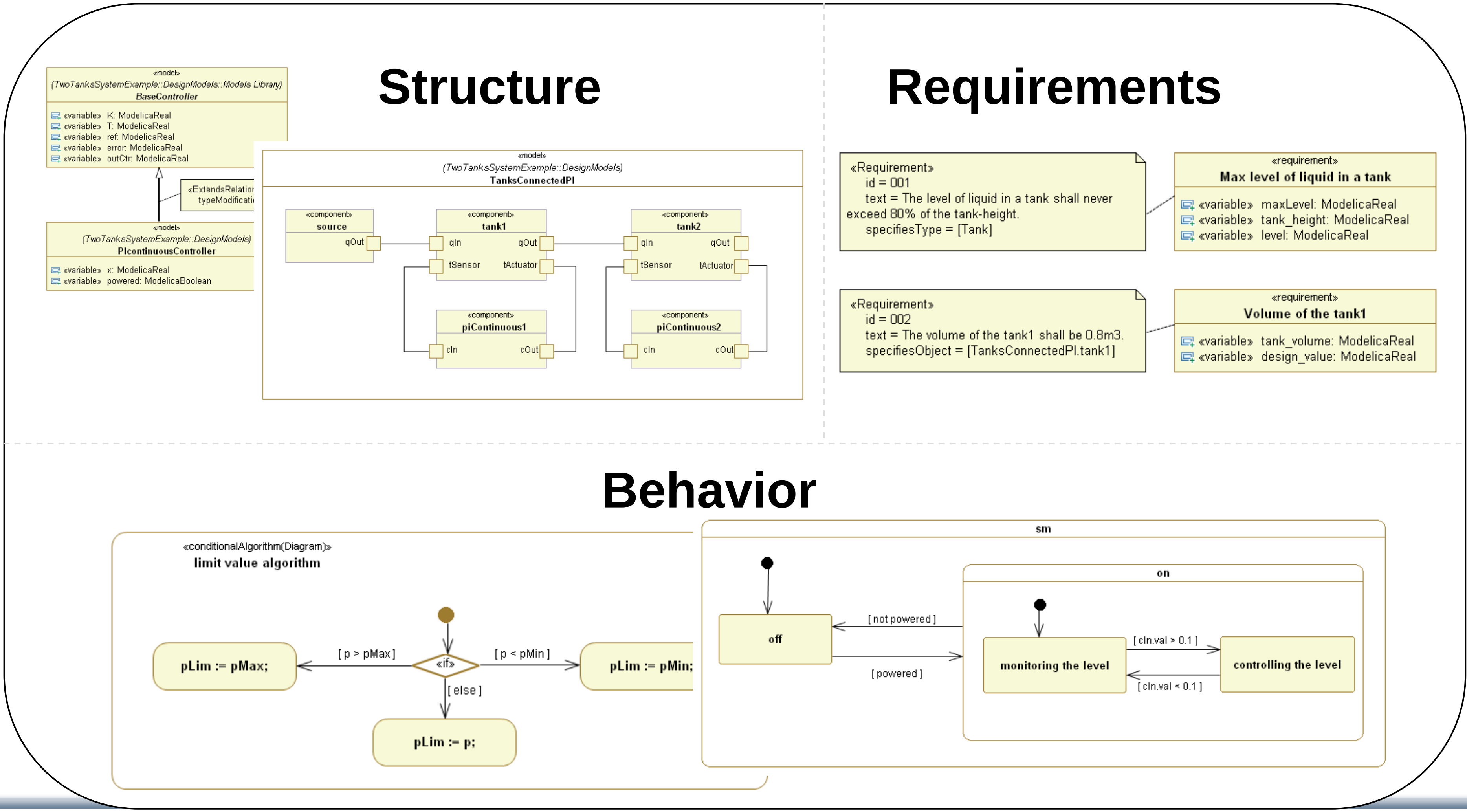
Simulation Diagram Introduced by ModelicaML



ModelicaML UML Profile, 2nd Generation SysML/UML to Modelica OMG Standardization (with Wladimir Schamai)

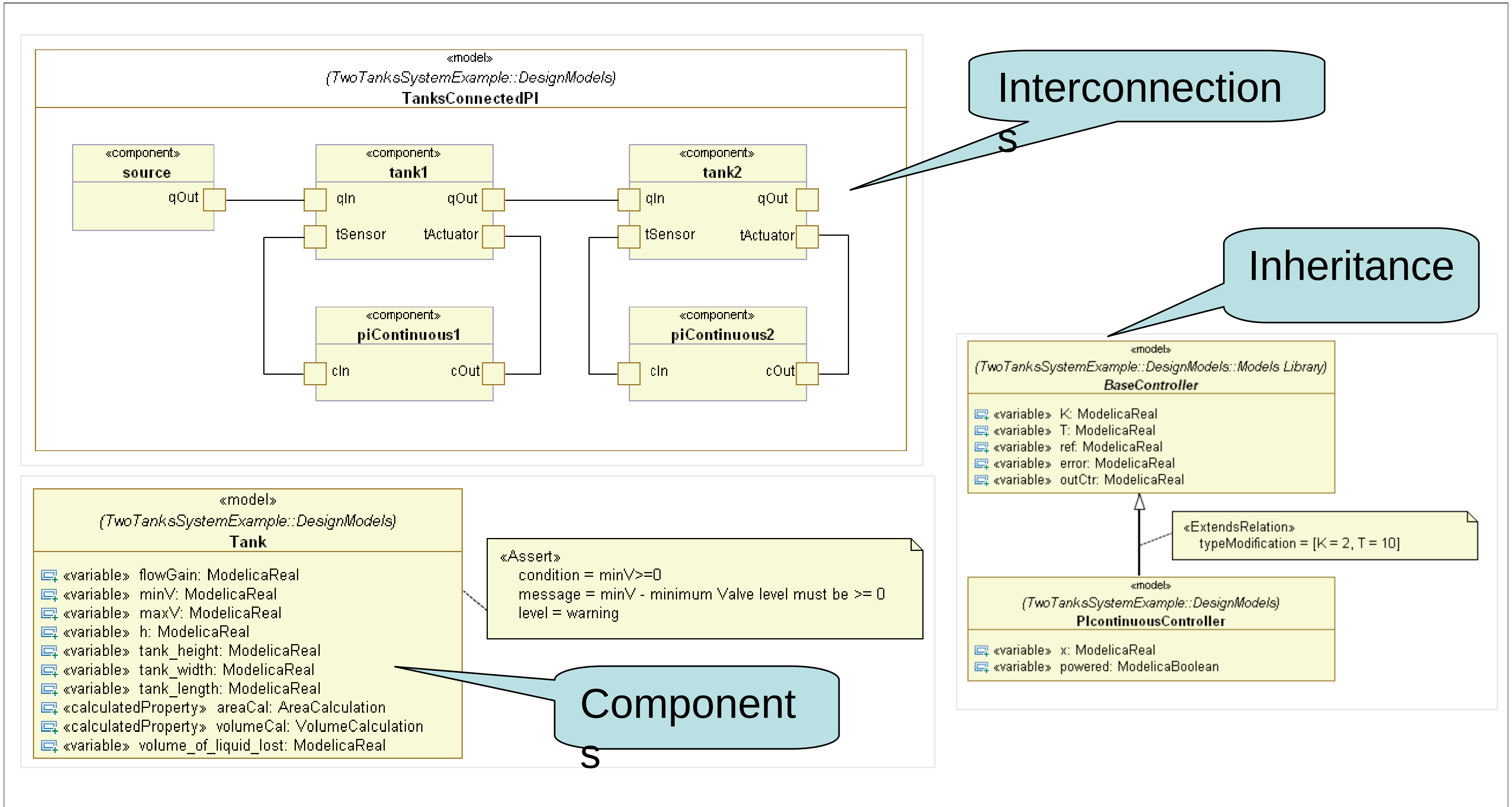
- ModelicaML is a UML Profile for SW/HW modeling
  - Applicable to “pure” UML or to other UML profiles, e.g. SysML
- Standardized Mapping UML/SysML to Modelica
  - Defines transformation/mapping for **executable** models
  - Being **standardized** by OMG
- ModelicaML
  - Defines graphical concrete syntax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

ModelicaML: Graphical Notation

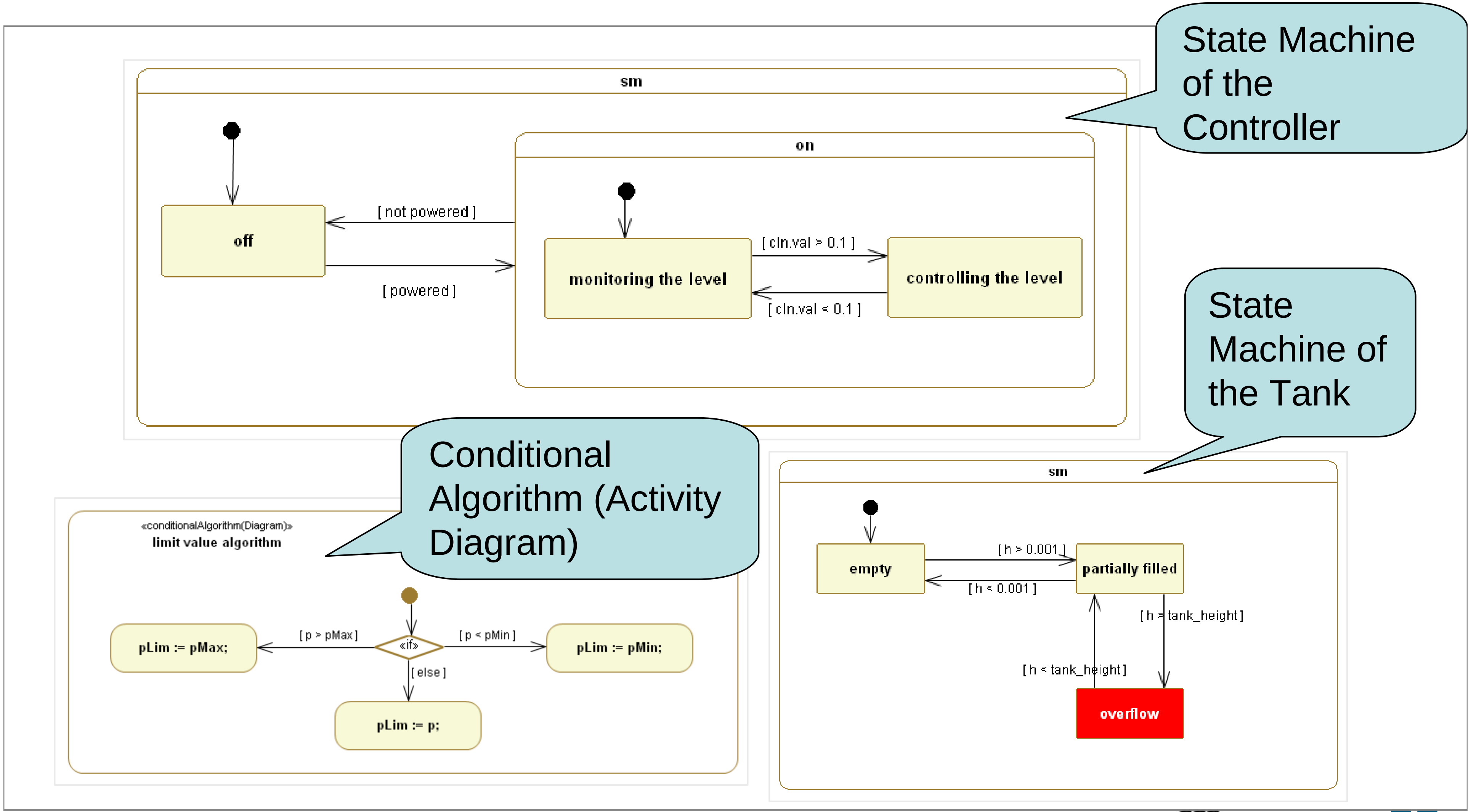


pelab

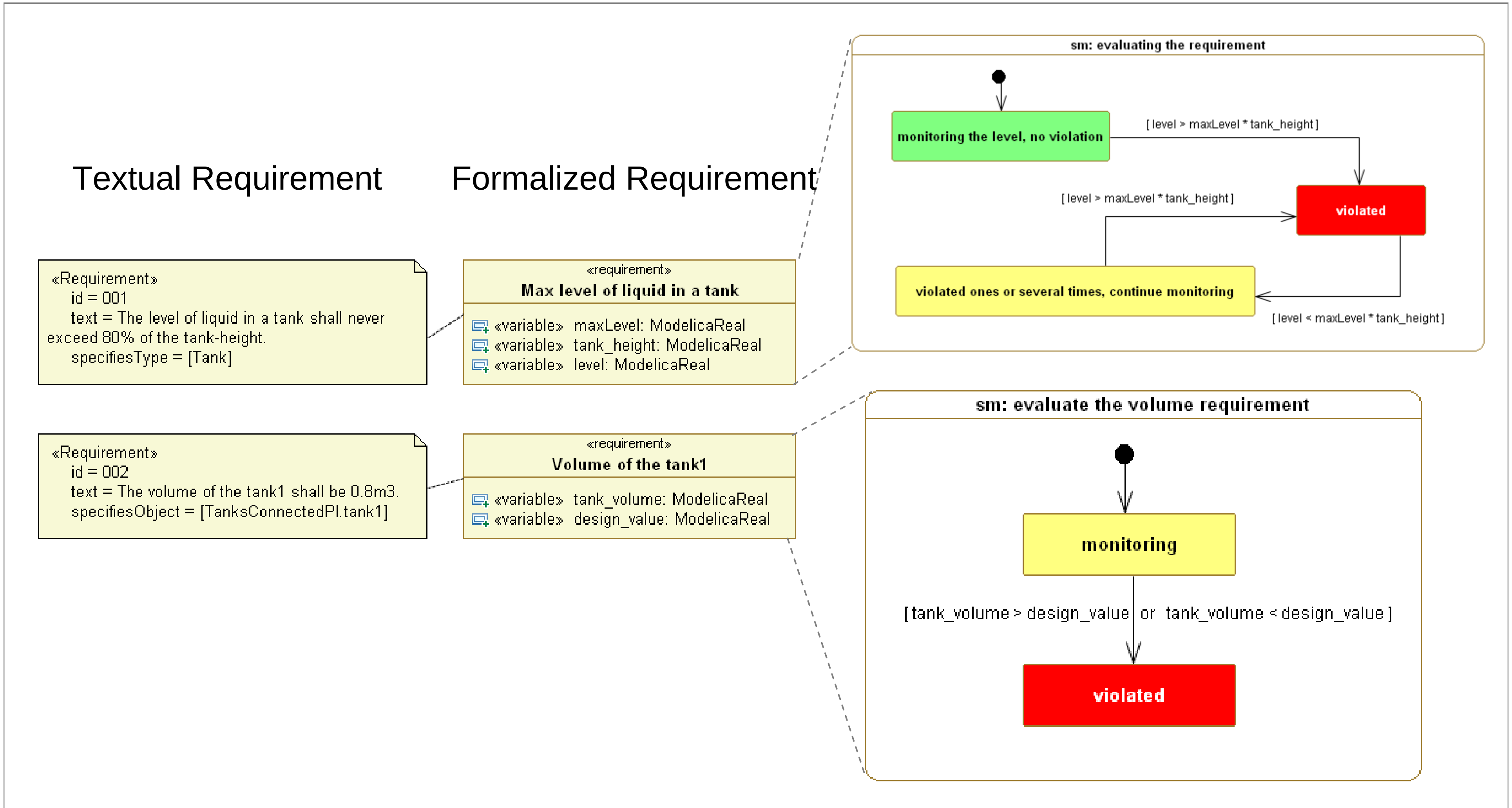
Example: Representation of System Structure



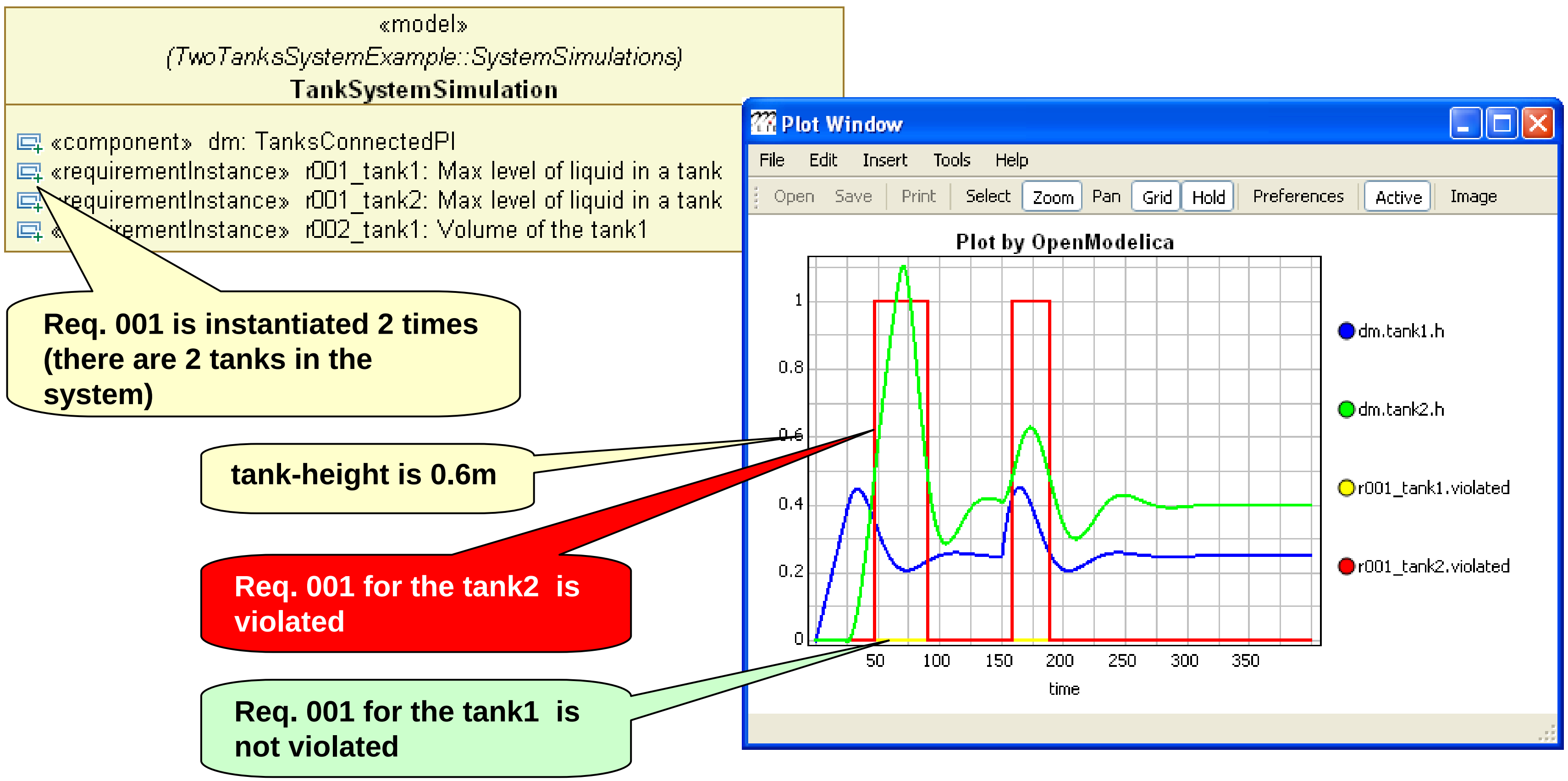
Example: Representation of System Behavior



Example: Representation of System Requirements

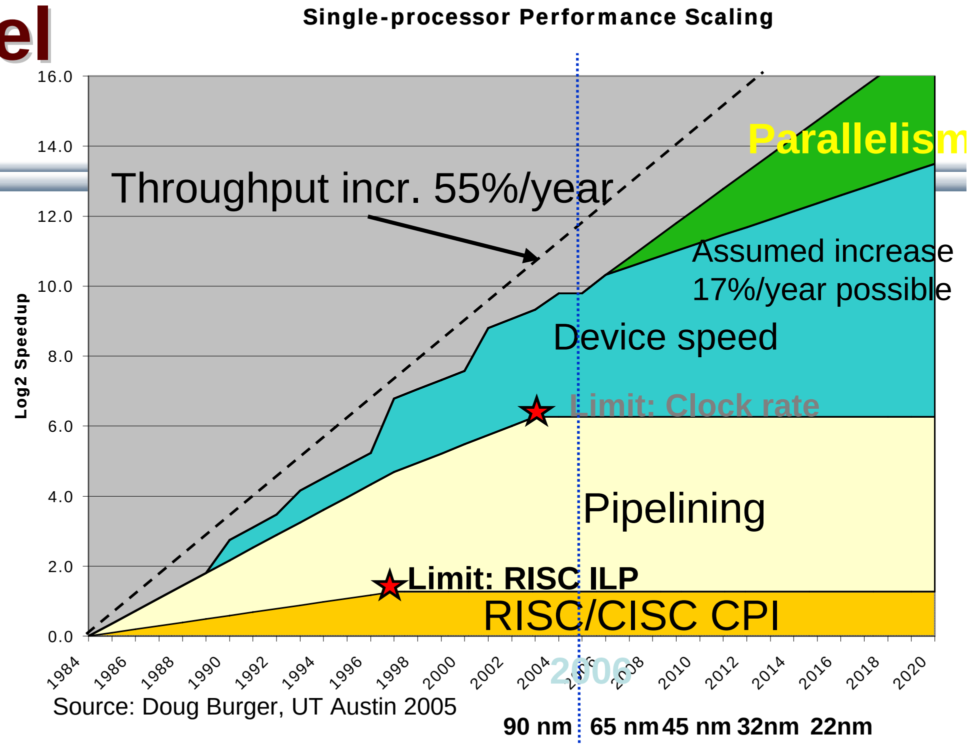


Example: Simulation and Requirements Evaluation



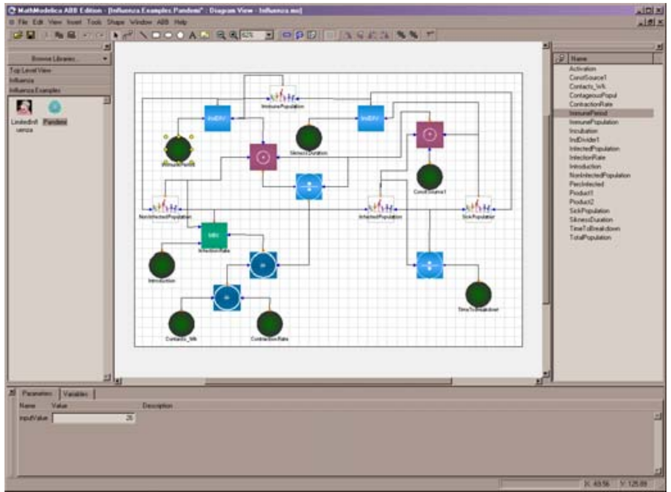
Parallel Execution
Compilation to MultiCore

Towards High-Level Parallel Modeling and Simulation



- Simulations are time-consuming
- Moore's "Law": (since 1965)
  - #devices per chip area doubles every 18 months
  - CPU clock rate also doubled every 18 months – until 2003, then: heat and power issues, limited ILP, ...
    → superscalar technology has reached its limits, only (thread-level) parallelism can increase throughput substantially
- The consequence:
  Chip multiprocessors (+ clusters)
  - Multi-core, PIM, ... (for general-purpose computing)
- Need parallel programming/modeling/parallelization
  - Automatic parallelization
  - Explicit parallel modeling and parallel programming

Towards Easy-to-Use Modeling & Simulation using Parallel Computers



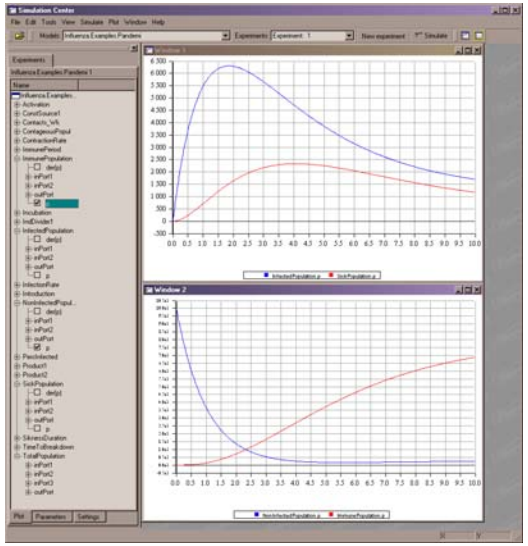
Modeling using drag'n'drop

Translation

Parallel Simulation Code

Parallel Execution

Visualization



Integrating Parallelism and Mathematical Models Three Approaches

- **Automatic Parallelization of Mathematical Models (ModPar)**
  - Parallelism over the numeric solver method.
  - Parallelism over time.
  - **Parallelism over the model equation system**
    - ... with fine-grained task scheduling

- **Coarse-Grained Explicit Parallelization Using Components**
  - The programmer partitions the application into computational components using strongly-typed communication interfaces.
    - Co-Simulation, Transmission-Line Modeling (TLM)

- **Explicit Parallel Programming**
  - Providing general, easy-to-use explicit parallel programming constructs within the *algorithmic* part of the modeling language.
    - NestStepModelica

Modelica Simulations – Parallelization Approach

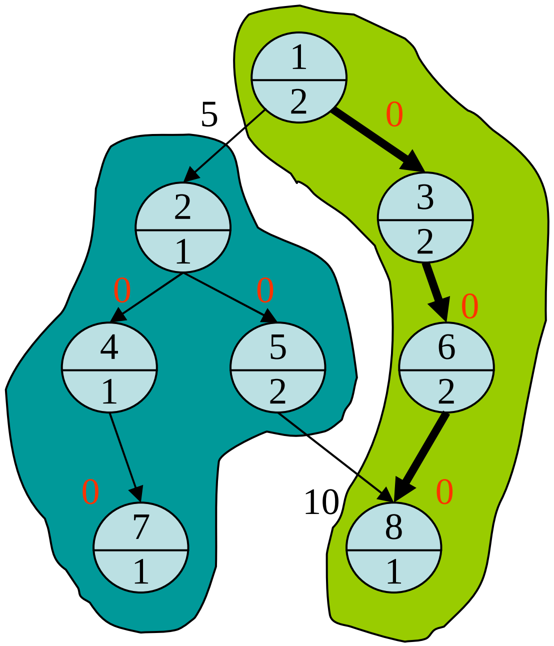
- Simulation = solution of (hybrid) DAEs from models

$$g(\dot{X}, X, Y, t) = 0$$

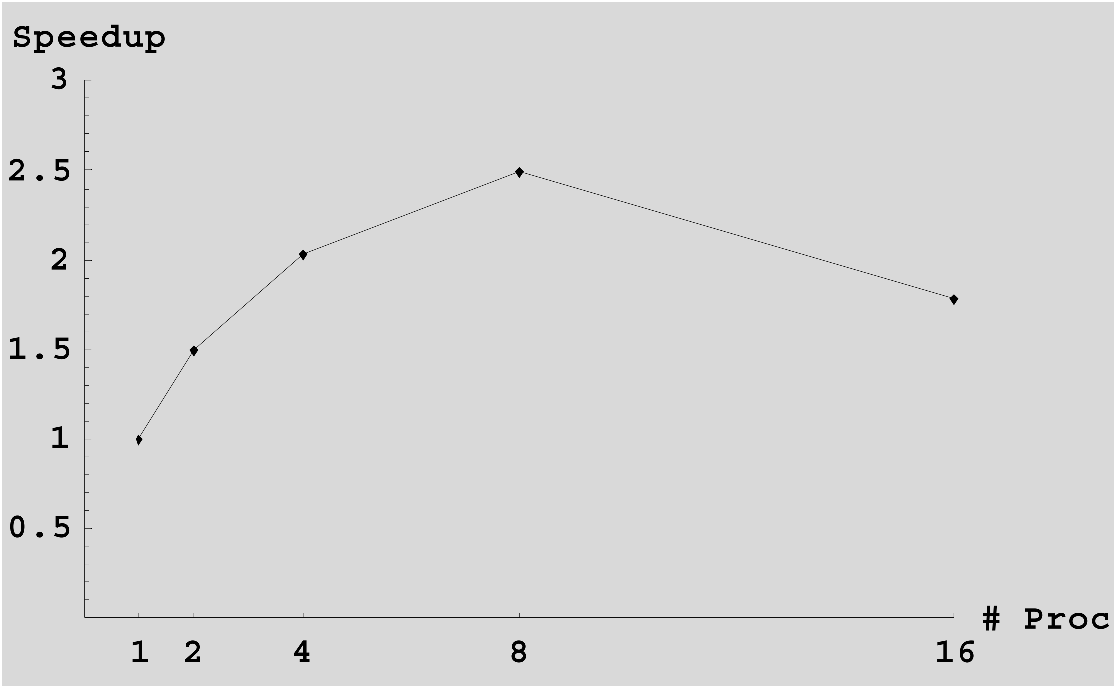
$$h(X, Y, t) = 0$$

- In each step of numerical solver:
  - Calculate $\dot{X}$ in g (and Y in h)
- Parallelization approach: perform the **calculation** of $\dot{X}$ in **parallel**
  - Called *parallelization over the system*.
- Drawback: Numeric solver might become bottle-neck

Example – Task Graphs and Parallelized Application

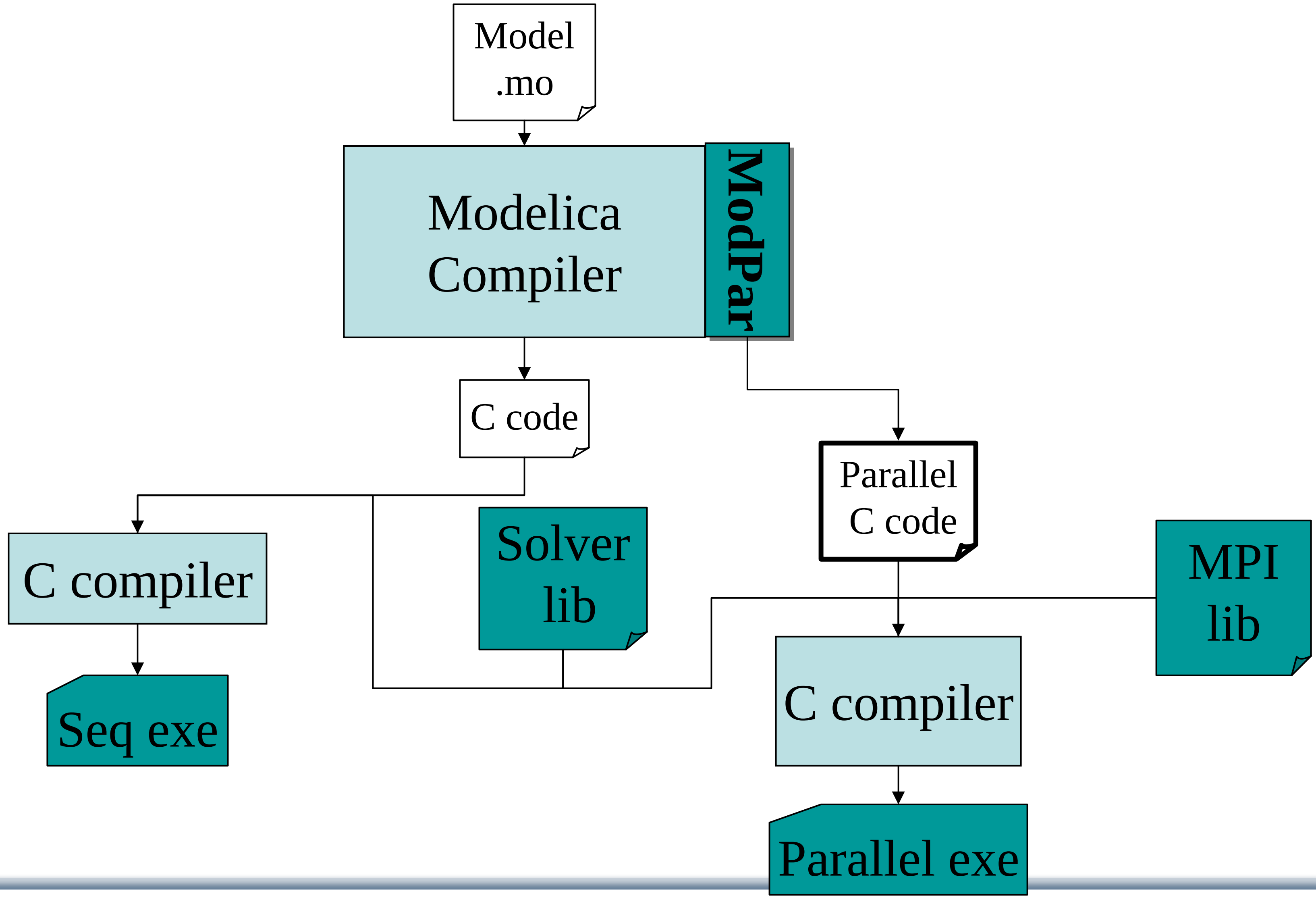


Clustered Task Graph

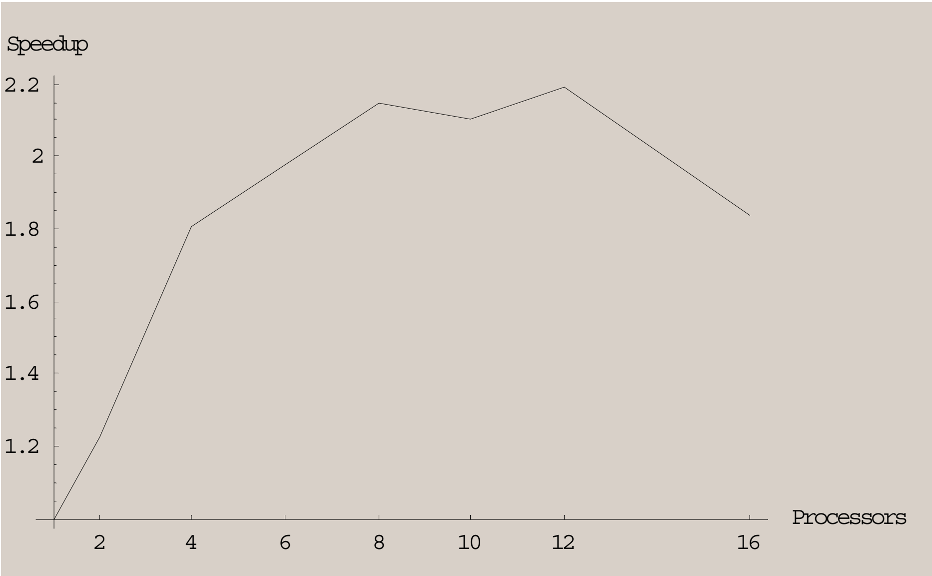


Thermofluid Pipe Application

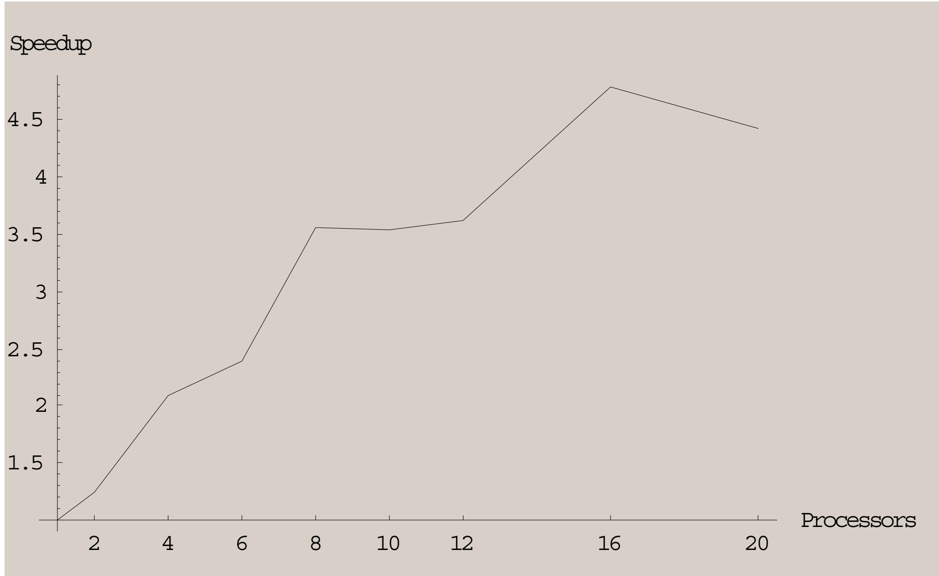
The ModPAR Parallelization Tool – Part of the OpenModelica Environment



Speedup Results on Flexible Shaft



Linux Cluster (SCI network)
(monolith.nsc.liu.se)



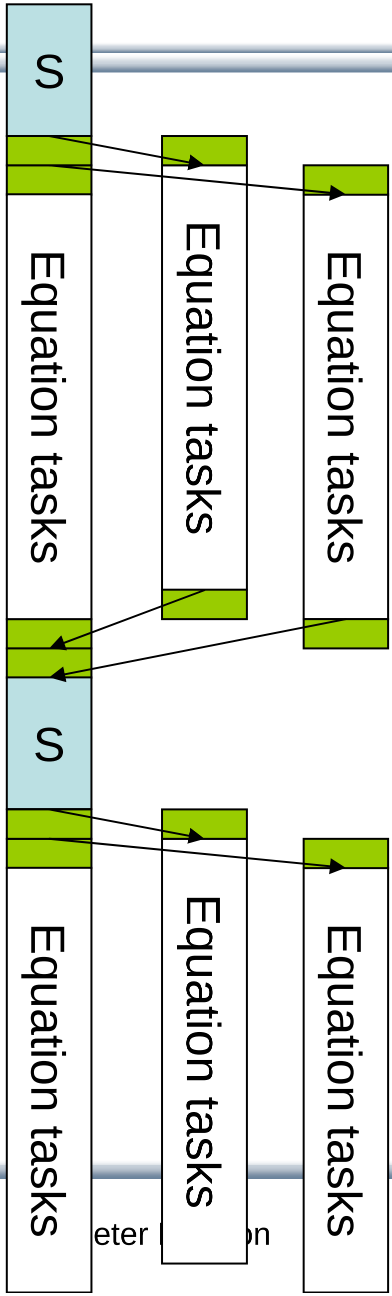
SGI Altix 3700 Bx2
(mozart.nsc.liu.se)

Modified Approach Automatic Fine-Grained Parallelization Using Software Pipelining and Solver Inlining

New Modified Approach – Inlining and Pipelining

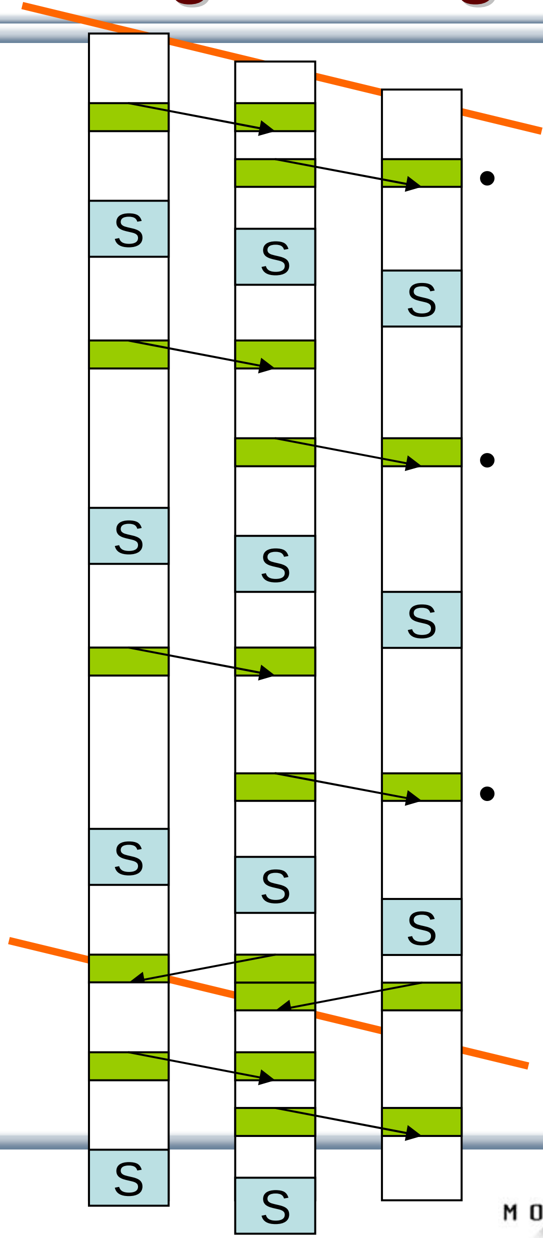
- Try to keep **communication as close** as possible
- Only **communicate in one direction** inside a time step.
- **Solver Inlining** – distribute the solver across all the processors
- Some **parallelism** across the **method** – parallel evaluation of Runge-Kutta step

Task Merging vs New Approach with Pipelining/Inlining



Use a graph rewriting system to merge tasks into larger tasks, based on latency and bandwidth.

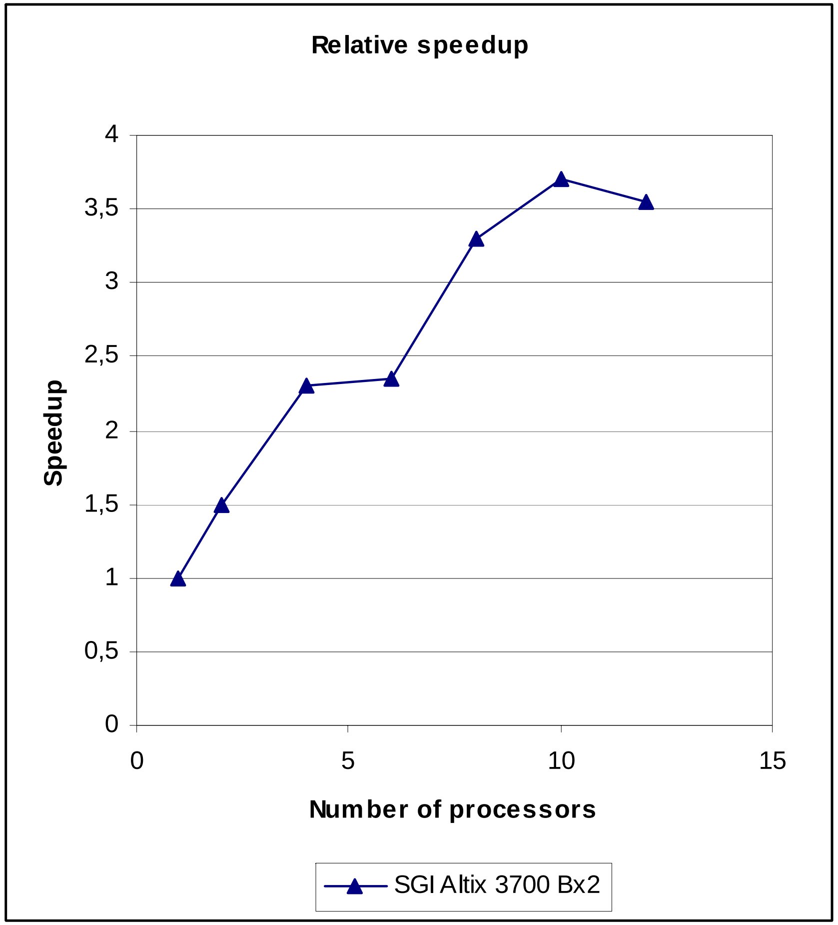
Some tasks are duplicated to avoid communication within a step



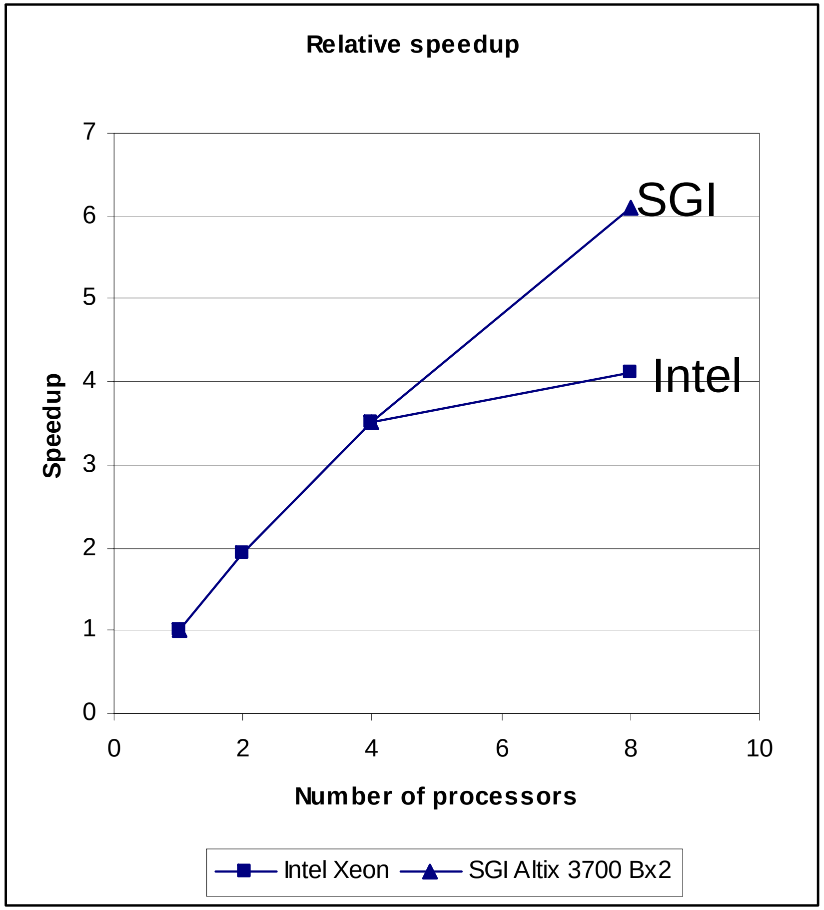
- Try to keep communication as close as possible
- Only communicate in one direction inside a time step.
- Solver Inlining – distribute the solver across all the processors

Measurements
(100000 steps Flexible Shaft Model)

Task-merging, MPI, SGI Altix

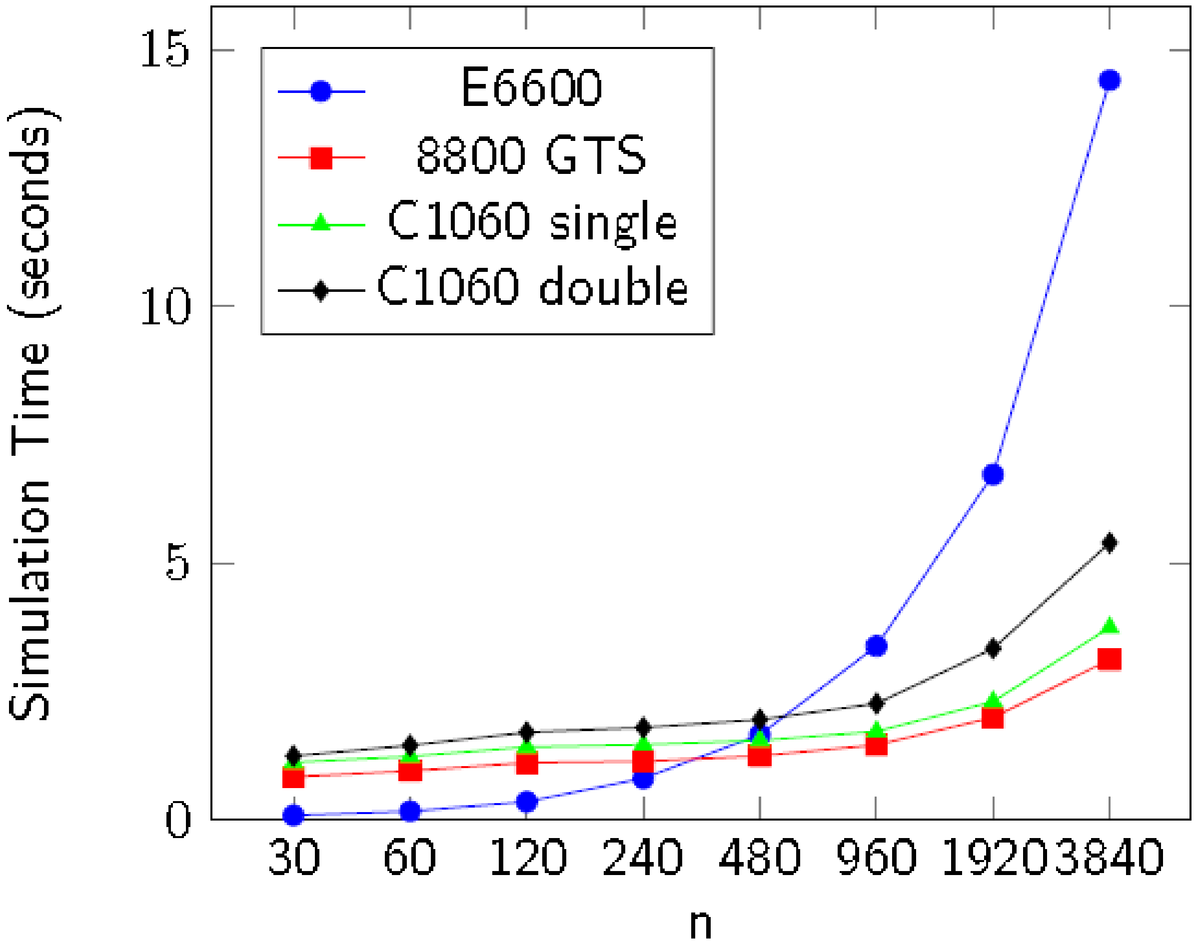


Pipelined, Pthreads, SGI, Intel Xeon



Recent Speedup Measurements on NVIDIA (nov 2009)

Modelica Model, Generated Code, Function of Problem Size



New 2 TeraFlop Parallel Platform to PELAB/LIU

- Just ordered: An NVIDIA Fermi 2050 2 Teraflop peak parallel platform, delivery in May-Sept 2010.
- Use, e.g. in research on compiling Modelica to MultiCore

Summary of MODPROD Research in PELAB

- Modeling language design (semantics, type systems, meta-modeling, extensibility)
- Modelica-SysML integration
- Requirements traceability, also Non-functional requirements
- Compilation to multi-core platforms

