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

Presentation at MODPROD'2013
PELAB – Programming Environment Laboratory
Department of Computer and Information Science
Linköping University
2013-02-05

Peter Fritzson
Examples of Complex Systems in Engineering

- Robotics
- Automotive
- Aircraft
- Mobile Phone Systems
- Business Software
- Power plants
- Heavy Vehicles
- Process industry
Architecture of Integrated Model-Based Product Development

Open Standards – Modelica (HW, SW) and UML (SW)
Open Source Modelica Consortium

Open Source Effort
The OpenModelica Open Source Environment
www.openmodelica.org

- Advanced Interactive Modelica compiler (OMC)
  - Supports most of the Modelica Language
- 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
- OMOptim optimization subsystem
- ModelicaML UML Profile
- ParModelica extension
- OMPython – Python scripting

www.openmodelica.org
OSMC – International Consortium for Open Source Model-based Development Tools, 45 members Dec 2012

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 Corp Research, Sweden
• Bosch Rexroth AG, Germany
• Siemens PLM, California, USA
• Siemens Turbo, Sweden
• CDAC Centre, Kerala, India
• Creative Connections, Prague
• DHI, Aarhus, Denmark
• STEAG, Dehli, India
• Equa Simulation AB, Sweden
• Fraunhofer FIRST, Berlin
• Frontway AB, Sweden
• IFP, Paris, France
• GTI, USA
• ISID Dentsu, Tokyo, Japan
• ITI, Dresden, Germany
• Wolfram MathCore, Sweden
• Maplesoft, Canada
• TLK Thermo, Germany
• Sozhou Tongyuan, China
• VI-grade, Italy
• VTI, Linköping, Sweden
• VTT, Finland
• XRG Simulation, Germany

University members
• Linköping University, Sweden
• TU Berlin, Insti UEBB, Germany
• FH Bielefeld, Bielefeld, Germany
• TU Braunschweig, Germany
• Univ Calabria, Italy
• TU Dortmund, Germany
• TU Dresden, Germany
• Université Laval, Canada
• Georgia InstiTechnology, USA
• Ghent University, Belgium
• Griffith University, Australia
• TU Hamburg/Harburg Germany
• 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
• Telemark Univ College, Norway
• Univ. Ålesund, Norway
OpenModelica Environment Demo
General Tool Interoperability & Model Exchange
Functional Mock-up Interface (FMI)

The FMI development is result of the MODELISAR 29-partner project

- FMI development initiated by Daimler
- Improved Software/Model/Hardware-in-the-Loop Simulation, of physical models and of AUTOSAR controller models from different vendors for automotive applications with different levels of detail.
- Open Standard
- 14 automotive use cases for evaluation
- > 30 tool vendors are supporting it

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
Research

Modeling-Language Design

Model-based Optimization

Fault Analysis – Bayesian Networks

Debugging

Multi-Core based Simulation

Modeling Support Environments
OPENPROD – Large 28-partner European Project, 2009-2012
Vision of Cyber-Physical Model-Based Product Development

OPENPROD Vision of unified modeling framework for model-based product development.

Open Standards – Modelica (HW, SW) and UML (SW)
OPENPROD Model-Based Development Environment Covers Product-Design V

- Business Process Control
- Requirements Capture
- Model-Driven Design
- Compilation & Code Gen
- System Simulation
- Software & System Product

Unified Modeling: Metamodeling & Modelica & UML

Level of Abstraction

- System requirements
- Preliminary feature design
- Architectural design and system functional design
- Detailed feature design and implementation

Experience Feedback

Documentation, Version and Configuration Management

- Specification
- Design
- Design Refinement

Calibration

Integration

Verification

Component verification

Subsystem level integration and verification

Subsystem level integration test calibration and verification

Product verification and deployment

Maintenance

Realization

Product verification and deployment

Preliminary feature design

System requirements

Architectural design and system functional design

Detailed feature design and implementation

Verification

Experience Feedback

Realization

Product verification and deployment

Preliminary feature design

System requirements

Architectural design and system functional design

Detailed feature design and implementation

Verification
Business Process Control and Modeling

Process models
Requirements models
Product models
Platform models
Compilation & Code Gen
Software & System Product

Unified Modeling: Meta-modeling & Modelica & UML

Feedback

OpenModelica based simulation

Metso Business model & simulation
VTT Simantics Graphic Modeling Tool
Simulation of 3 strategies with outcomes

VTT Simantics Business process modeler
OpenModelica compiler & simulator
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
Model-Driven Whole-Product Design
SW/HW – UML/Modelica

Unified Modeling: Meta-modeling & Modelica & UML

Integrated SW/HW Modeling with ModelicaML/OpenModelica

- SW modeled in UML: statecharts, activity diagrams
- UML translated to Modelica allows integration with Modelica (HW) models

1. System Modeling with ModelicaML
2. Modelica Code Generation
3. System Simulation with Modelica Tools
Model-Driven Design, mostly with Modelica

New Capabilities

- Uncertainties modeling (EDF, LiU, MaCo)
- Improved Type System (LMS, LiU)
- Acausal analysis for design (INSA, PSA)
- Sensitivity analysis (Appedge, IFPEN)
- Model Debugger (LiU)
- Model Guidelines Checker (TUC, LiU)
- Model Simplifier (Bosch-Rexroth)

Modeling & Design Tools

- OpenModelica (LIU, UBiele, MaCo, et al)
- Wolfram System Modeler (MaCo)
- IDA Simulation (Equa)
- Simantics (VTT)
- BEAST (SKF)
- Amesim (LMS)
- BoGie (INSA)
- Jmodelica.org, Optimica Studio (Modelon)
- TeamCenter (Siemens)
- D&C System Simulator (Bosch-Rexroth)
Compilation and Code Generation

Unified Modeling: Meta-modeling & Modelica & UML

New Capabilities

- OMC QSS Codegen (ETH, LIU)
- OMC Inline Codegen (LIU, Ubiele)
- OMC Sparse Jacobians (UBiele)
- OMC Symbolic Linearization (Ubiele, MaCo)
- Multi-Rate Controller (Equa)
- Simulation Efficient Storage (Maco, BR)
- DSP Codegen (LIU, MaCo)
- Real-Time Code Gen

Model Compiling Tools

- OpenModelica Compiler (OMC) (LIU, UBiele, MaCo, et al)
- Wolfram System Modeler (MaCo)
- IDA Simulation (Equa)
- Amesim (LMS)

Business Process Control
- Process models
- Requirements models
- Requirements Capture

Product models
- Model-Driven Design

Platform models
- Compilation & Code Gen

System Simulation
- Software & System Product

Feedback
System Simulation

New Capabilities

- OMC QSS Solver (ETH, LIU)
- OMC Parallel Solver (LiU)
- OMC Modular Solver Runtime (BR, LiU)
- OMC Interactive Simulation (EADS, LIU)
- 3D Graphic Animation (Fhg)
- Sparse-Matrix DAE Solver (XRG, BR)
- Solver Selection (LiU, MaCo)
- Plexsim QSS Solver (ETH, Plexsim)
- BEAST Parallel Solver (SKF, LIU)
- Data Reconciliation (EDF, LIU, MaCo)

Simulation Tools

- OpenModelica Simulator (LIU, Ubiele, MaCo, et al)
- Wolfram Syst Modeler/ Simulator (MaCo)
- Plexsim Simulator (Plexsim)
- BEAST Simulator (SKF)
- xMOD Simulator (IFP)
- OMC Parallel Solver (LiU)
- OMC Modular Solver Runtime (BR, LiU)
- OMC Interactive Simulation (EADS, LIU)
- Sparse-Matrix DAE Solver (XRG, BR)
- Solver Selection (LiU, MaCo)
- Plexsim QSS Solver (ETH, Plexsim)
- BEAST Parallel Solver (SKF, LIU)
- Data Reconciliation (EDF, LIU, MaCo)
- Simulation Tools New Capabilities

Feedback

Unified Modeling: Meta-modeling & Modelica & UML
Interoperability

Unified Modeling: Meta-modeling & Modelica & UML

Business Process Control
- Process models

Requirements Capture
- Requirements models

Model-Driven Design
- Product models

Compilation & Code Gen
- Platform models

System Simulation

Software & System Product

Feedback

New Capabilities

FMI 1.0 Prototypes (LiU, Modelon, MaCo, TLK, TUC)
FMI 2.0 Subset Co-simulator (LiU)
FMI 2.0 Sparse Jacobians (Ubiele, Modelon)
XML Generation (Modelon, LIU)
Real-Time Co-Sim Platform (IFP, TLK, TUC)
Flat Modelica export/import (LiU, Equa)
OPC Data Access Server (VTT)
PDM System Integration (Siemens)

M&S Tools, PLM Tools

OpenModelica (LIU, Ubiele, MaCo, et al)
xMOD (IFP)
Wolfram System Modeler (MaCo)
TISC Co-Simulator for xIL (TLK)
Jmodelica.org, Optimica Studio (Modelon)
TeamCenter (Siemens)

System Simulation

Software & System Product

Feedback

New Capabilities

FMI 1.0 Prototypes (LiU, Modelon, MaCo, TLK, TUC)
FMI 2.0 Subset Co-simulator (LiU)
FMI 2.0 Sparse Jacobians (Ubiele, Modelon)
XML Generation (Modelon, LIU)
Real-Time Co-Sim Platform (IFP, TLK, TUC)
Flat Modelica export/import (LiU, Equa)
OPC Data Access Server (VTT)
PDM System Integration (Siemens)

M&S Tools, PLM Tools

OpenModelica (LIU, Ubiele, MaCo, et al)
xMOD (IFP)
Wolfram System Modeler (MaCo)
TISC Co-Simulator for xIL (TLK)
Jmodelica.org, Optimica Studio (Modelon)
TeamCenter (Siemens)
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?
Efficient Traceable Model-Based Dynamic Optimization – EDOp Research Project

- Integrated modeling & optimization in OpenModelica
  - Equation oriented tool, generation of efficient parallel code
  - Based on OpenModelica, cooperation with Open-Source Consortium
  - Including model error traceability and localization

- Efficient solution methods for model-based optimization problems
  - Methods can exploit structure of the problem and their equations
  - Exploit models for parallel computing
  - Utilize equations to analytically compute gradients and hessians, faster search routines

- Optimized Heavy Industrial Applications
  - Trucks (Scania), Wheel Loaders (Volvo CE)
  - Bearing applications in mechanical systems (e.g. wind power, SKF)
  - Siemens: Gas turbines, Frontway: Process industry
OpenModelica Prototype Parallel Multiple-Shooting and Collocation Dynamic Trajectory Optimization

- Minimize a goal function subject to model equation constraints, useful e.g. for NMPC
- Multiple Shooting/Collocation
  - Solve sub-problem in each sub-interval

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

\[ x_i(t_i) = h_i \]

Example speedup, 16 cores:

MULTIPLE_COLLOCATION
OpenModelica Optimization Subsystem OMOptim

- Parameter optimization
- Currently using genetic optimization algorithms in OMOptim 0.9.
- Work on more parameter optimization
- Work on dynamic optimization using multi-core computers
Bayesian Network Based Fault and Failure Analysis – Cooperation with Scania, Mattias Nyberg

- Developing a Fault and Failure analysis tool for the engineers, containing the following features:
- Modelica/UML/SysML-like intuitive GUI design interface with simulation
- Functionality
  - Generate Failure Mode and Effect Analysis, FMEA table
  - Generate Fault Tree, FTA table
  - Answer to Queries
  - Probability of hazards
  - Root cause of hazards
  - Infer hazards, given the root cause
Bayesian Network Based Fault and Failure Analysis
Association of Services to the Components
Need for Debugging Tools
Problems and Motivation

• 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 $**

• **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
  ...

Debugger Integrated in Eclipse OpenModelica MDT Environment

- Eclipse plugin MDT (Modelica Development Tooling) is the integrated development environment
- Debugger is a debug plug-in within MDT
Breakpoint Support in the Debugger
Equation Model Transformation Debugger

- Complicated to understand source of some errors
- Trace transformations through the compiler

\[ 0 = y + \text{der}(x \times \text{time} \times z); \ z = 1.0; \]

(1) subst:
\[ y + \text{der}(x \times (\text{time} \times z)) \]
\[ \Rightarrow \]
\[ y + \text{der}(x \times (\text{time} \times 1.0)) \]

(2) simplify:
\[ y + \text{der}(x \times (\text{time} \times 1.0)) \]
\[ \Rightarrow \]
\[ y + \text{der}(x \times \text{time}) \]

(3) expand derivative (symbolic diff):
\[ y + \text{der}(x \times \text{time}) \]
\[ \Rightarrow \]
\[ y + (x + \text{der}(x) \times \text{time}) \]

(4) solve:
\[ 0.0 = y + (x + \text{der}(x) \times \text{time}) \]
\[ \Rightarrow \]
\[ \text{der}(x) = ((-y) - x) / \text{time} \]
\[ \text{time} \neq 0 \]
Integrated Hardware-Software Modeling

ModelicaML

UML Profile for Modelica

SysML-Modelica Integration
Using ECLIPSE as Integration Platform

- OpenUP/Basic
- Capacity Sub-Process Areas
- UML-Modelica Plug-in
- OpenModelica MDT

- ECLIPSE Process Framework (EPF)
- Composer Specific components

- Graphical Modeling Framework

- ECLIPSE Rich Client Platform (RCP) Runtime
  - Java runtime
  - C/C++ runtime
  - OpenModelica runtime
  - MetaModelica runtime

- OpenModelica
- MDV
- Graphical Modeling Framework
- UML-Modelica Plug-in
- OpenUP/Basic Capacity Sub-Process Areas
ModelicaML – UML Profile for Modelica 1st Generation (2006-2008)

• Extension of SysML subset
• Features:
  • Supports Modelica constructs
  • Modelica generic class modeling
  • Modelica syntax in definitions
  • Equation-based modeling
  • Simulation modeling
ModelicaML UML Profile, 2nd Generation
SysML/UML to Modelica OMG Standardization
(with Wladimir Schamai, from 2009 until Now)

• 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

Structure

Requirements

Behavior

- Structure:
  - Variables: V, Modelica
  - Variables: T, Modelica
  - Variables: m0, Modelica
  - Variables: m1, Modelica

- Behavior:
  - limit value algorithm:
    - \( p \leq p_{\text{Max}} \)
    - \( p > p_{\text{Min}} \)
    - \( p = p_{\text{Min}} \)

- Requirements:
  - Requirement: Max level of liquid in a tank
    - ID: 001
    - Text: The level of liquid in a tank shall never exceed 80% of the tank height.
    - Specifier: Tank

  - Requirement: Volume of the tank
    - ID: 002
    - Text: The volume of the tank shall be 0.6m^3.
    - Specifier: TankConnected
    - Specifier: Tank1
Example: Representation of System Structure

- **Interconnections**
- **Inheritance**
- **Components**
Example: Representation of System Behavior

State Machine of the Tank

State Machine of the Controller

Conditional Algorithm (Activity Diagram)
Example: Representation of System Requirements

Textual Requirement

- ID: 001
  - Text: The level of liquid in a tank shall never exceed 80% of the tank-height.
  - Specified Object: [Tank]

Formalized Requirement

- Max level of liquid in a tank
  - Variables:
    - maxLevel: ModelicaReal
    - tank_height: ModelicaReal
    - level: ModelicaReal

- Volume of the tank
  - Variables:
    - tank_volume: ModelicaReal
    - design_value: ModelicaReal

Diagram:

1. Evaluating the Requirement
   - Condition: level > maxLevel * tank_height
     - If violated, continue monitoring.

2. Evaluating the Volume Requirement
   - Condition: tank_volume > design_value or tank_volume < design_value
     - Violated
Example: Simulation and Requirements Evaluation

Req. 001 is instantiated 2 times (there are 2 tanks in the system)

- tank-height is 0.6m
- Req. 001 for the tank2 is violated
- Req. 001 for the tank1 is not violated
- 1. Select requirements to be verified using simulations (or logic tools)
- 2. Formalize textual requirements

- Support Requirements Analyst in selecting requirements
- 3. Create system design models to be verified against requirements
- Support System Tester in deciding which requirements are to be verified using which test cases
- Support System Tester in linking requirement properties to design model properties

- 4. Create test modes including test cases and decide which requirement are to be verified using which test cases
- 5. Simulate and observe requirement violations
- 6. Analyze simulation results and create a Simulation Summary Report
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
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
Modelica + OpenCL = ParModelica

function parvar
    Integer m = 1024;
    Integer A[m];
    Integer B[m];
    parglobal Integer pm;
    parglobal Integer pn;
    parglobal Integer pA[m];
    parglobal Integer pB[m];
    parlocal Integer ps;
    parlocal Integer pSS[10];
end parvar;

algorithm
    B := A; //copy to device
    pA := A; //copy from device
    B := pA; //copy device to device
    pm := m;
    n := pm;
    pn := pm;
end parvar;

ParModelica – Modelica Parallel Algorithmic Programming
Here: parglobal and parlocal Variables; Execution on GPUs

<table>
<thead>
<tr>
<th>Memory Regions</th>
<th>Accessible by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Memory</td>
<td>All work-items in all work-groups</td>
</tr>
<tr>
<td>Constant Memory</td>
<td>All work-items in all work-groups</td>
</tr>
<tr>
<td>Local Memory</td>
<td>All work-items in a work-group</td>
</tr>
<tr>
<td>Private Memory</td>
<td>Private to a work-item</td>
</tr>
</tbody>
</table>
ParModelica Matrix Multiplication using *Kernel* function

Gained speedup

- Intel Xeon E5520 CPU (16 cores) 26
- NVIDIA Fermi-Tesla M2050 GPU (448 cores) 115

Speedup comparison to sequential algorithm on Intel Xeon E5520 CPU

### Speedup

<table>
<thead>
<tr>
<th>Parameter M (Matrix sizes MxM)</th>
<th>CPU E5520 (Serial)</th>
<th>CPU E5520 (Parallel)</th>
<th>GPU M2050 (Parallel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.61</td>
<td>0.137</td>
<td>1.215</td>
</tr>
<tr>
<td>128</td>
<td>4.36</td>
<td>0.17</td>
<td>1.217</td>
</tr>
<tr>
<td>256</td>
<td>13.41</td>
<td>0.438</td>
<td>1.274</td>
</tr>
<tr>
<td>512</td>
<td>24.76</td>
<td>2.36</td>
<td>2.36</td>
</tr>
<tr>
<td>114.67</td>
<td></td>
<td>17.66</td>
<td>17.66</td>
</tr>
</tbody>
</table>

Simulation Time (second)

- CPU E5520 (Serial) 0.093 0.741 5.875 58.426 485.234
- CPU E5520 (Parallel) 0.137 0.17 0.438 2.36 17.66
- GPU M2050 (Parallel) 1.215 1.217 1.274 1.625 4.057
New Scalable OpenModelica Parallel Code Generator

- Ongoing work
- Both **task** parallelism and **data** parallelism
- Handling non-expanded **arrays** efficiently
- Includes use of TLM-partitioning for more parallelism

- OPENMP based parallelization prototype for equation-based models in OpenModelica

- ParModelica - Generating **OpenCL** code for platform independence

- **Template** based code generator
Summary of MODPROD Research at PELAB

- Modeling language design (semantics, type systems, meta-modeling, extensibility)
- Model-based efficient optimization (multi-core)
- Fault estimation & traceability; Bayesian networks
- Modelica-UMLSysML integration
- Requirements traceability
- Debugging of models
- Compilation to multi-core platforms
- Compilation and performance measurements for real-time simulation