Technical Overview of OpenModelica and its Development Environment

Adrian Pop

2011-02-07

Open Source Modelica Consortium
Programming Environment Laboratory
Department of Computer and Information Science
Linköping University

www.OpenModelica.org
OpenModelica
- What is OpenModelica?
- The past and present

OpenModelica Technical Overview
- OMC, OMSHELL, OMNotebook,
  OMEdit, ModelicaML, SimForge

OpenModelica Development Environment
- MetaModelica (RML/OMC)
- The Eclipse Environment (MDT)

OpenModelica Latest Developments (2010-2011)
What is OpenModelica? (I)

- Advanced Interactive Modelica compiler (OMC)
  - Supports MLS v. 3.1 (without Media & Fluid)
- Basic environments for creating models
  - OMShell - an interactive command handler
  - OMNotebook - a literate programming notebook
  - OMEdit - Open Modelica Connection Editor (New)
  - MDT - an advanced textual environment in Eclipse
What Is OpenModelica? (II)

- Advanced Eclipse-based Development Environment
- **Modelica Development Tooling (MDT) - started in 2005**
  - Code Assistance, Debugging, Outline & a lot more
  - *Used heavily for OpenModelica development*
  - Used in 8 OpenModelica Development Courses (INRIA, PELAB)
- **ModelicaML UML/SysML integration (New)**

![System Modeling with ModelicaML](image1)
![Modelica Code Generation](image2)
![System Simulation with Modelica Tools](image3)
What is OpenModelica? (III)

- Open-source community services
  - Website and Support Forum
  - Version-controlled source base
  - Bug database (unfortunately)
  - Development courses
An incubator platform for research
- 4 PhDs since 2004 (Debugging, Parallelization, PDEs Extensions)
- 18 Master’s theses since 2004
- Both the students and the project benefit

Master theses at PELAB 2006-2011
- Refactoring/Parsing and Language extensions
- UML/SysML view of Modelica code
- 2D and 3D visualization tools
- Static and runtime debugging tools
- Advanced code generation and parallelization of simulation code
- Bootstrapping and Java Interface
- Function pointers
- NVIDIA Cuda parallel simulation
- OMEdit – Modelica Connection Editor

External Master theses
- Model based diagnostics at ISY (Dep. Of Electrical Engineering)
- Monte-Carlo simulation of Satellite Separation Systems at SAAB
- Interactive Simulations (EADS)
- Additional Solvers + Event handling (FH-Bielefeld)
- EADS - ModelicaML

A Base for commercial and open source products
- MathCore AB, Bosch Rexroth, InterCAX (MagicDraw SysML), VTT
OpenModelica Roadmap - Past

1997 - started as a master thesis
2003 - first usable internal version
2004 - first external version: OpenModelica 1.1
2005 - more development: OpenModelica 1.3.1

2006 - major milestone

- Translated the whole compiler to MetaModelica
- Integrated Development Environment for the compiler
- OpenModelica website started
- Moved the code repository to Subversion management
- Extended the OpenModelica environment with new tools
- 4 versions released during the year
- External people start using OpenModelica
  - ~ 200 downloads/month
  - first development course at INRIA
2007 - continued development and community involvement

- Improvement in website, support and documentation
- Answered ~1000 questions on the forum
- Portability is highly improved, ported to 4 platforms
  - Linux, Mac, Solaris, Windows (version 1.4.3)
- Improvement of the compiler development tools in Eclipse
- OpenModelica Community starts to react
  - contribute code & report bugs & request enhancements & participate in answering questions in the OpenModelica forum
  - participate at courses and workshops
- New server acquired for better community services
- Increased usage: ~600 downloads/month
- Open Modelica Consortium created in December 4
  - 4 months of work
  - 9 organizations as members already (3 Universities, 6 Companies)
  - discussions are ongoing with other 6 companies
2008 - Further work on the compiler

- Release 1.4.4 and 1.4.5
  - Linux, Mac, Solaris, Windows
- New Solver Interface
- Refactoring
- Dynamic loading of functions
- Merging of MathCore front-end code
- 744 commits in Subversion
- Much more other things I don’t remember
OpenModelica Roadmap – Past

2009

- Work mainly happened in OSMC (partially on a non-public branch)
- Front-end
  - Refactoring (OSMC)
  - Enumerations (OSMC)
  - Java Interface and Booststrapping (Martin Sjölund)
  - MultiBody flattening (OSMC)
  - Constraint connection graph breaking (VTT + OSMC)
  - Support for Modelica 3.x and 3.x annotations (OSMC)
- Back-end
  - Tearing in the back-end (Jens Frenkel)
  - Template Code Generation and CSharp backend (Pavol Privitzer, Charles University Prague)
  - Interactive Simulations (EADS)
  - C++ Code generation (Bosch Rexroth)
  - Java Interface and Booststrapping (Martin Sjölund)
  - Additional Solvers + Events (Willi Braun, FH-Bielefeld)
- General
  - New MDT based on Xtext (Antanas Pavlov, SysMO and BMW)
  - New ModelicaML + SysML prototype (EADS)
  - 1144 commits in subversion (Since 2009 to February 8, 2010)
  - Bug fixes (OSMC)
  - Release 1.5.0 and 1.5.0-RC_X (Linux, Mac, Solaris, Windows)
- More things I don’t remember
2010 - 2011

- Support for Modelica Standard Library 3.1 (Media & Fluid in works)

**Front-end**
- MultiBody flattening (OSMC)
- Support for Modelica 3.x and 3.x annotations (OSMC)
- Performance Enhancements
- Stream connectors
- Media & Fluid work is on the way

**Back-end**
- Back-end redesign (Jens, Willi, Martin, Per, Adrian, Kristian)
- Tearing in the back-end (Jens Frenkel)
- Template Code Generation and CSharp backend (Pavol Privitzer, Charles University Prague)
- Interactive Simulations (EADS)
- C++ Code generation (Bosch Rexroth)
- Additional Solvers + Events + Linearization (Willi Braun, FH-Bielefeld)

**General**
- OMEdit - new connection editor
- Bootstrapping OMC (90% finished)
- 2550 commits in subversion from 2010 to Feb. 7, 2011 (double than 2009-2010)
- Bug fixes ~300+ (OSMC)
- Release 1.6.0 (Linux, Mac, Windows)
- Downloads Windows (~16434), Linux (~8301), Mac (~2816)

More things I don’t remember
- ~ 4000K lines of code and tests, steady increase
Outline

- OpenModelica
  - What is OpenModelica?
  - The past and present

- OpenModelica Technical Overview
  - OMC, OMSHELL, OMNotebook,
  - OMEdit, ModelicaML, SimForge

- OpenModelica Development Environment
  - MetaModelica (RML/OMC)
  - The Eclipse Environment

- OpenModelica Latest Developments (2010-2011)
OMShell & OMNotebook

Demo?

```
OpenModelica 1.4.3
Copyright 2002-2006, PELAB, Linkoping University

To get help on using OMShell and OpenModelica, type "help()" and
press enter.

>> loadModel(Modelica)
true

>> loadFile("C:/OpenModelica1.4.3/testmodels/BouncingBall.mo")
true

>> simulate(BouncingBall, stopTime=3)
record
  resultfile = "BouncingBall_res.plt"
end record

>> plot(h)
true
```

Van der Pol Model

This example describes a Van der Pol oscillator. Notice that here the keyword model is used instead of class with
the same meaning. This example contains declarations of two state variables x and y, both of type Real and a parameter
constant lambda, which is a so-called simulation parameter. The keyword parameter specifies that the variable is
constant during a simulation run, but can have its value initialized before a run, or between runs. Finally, there is an
equation section starting with the keyword equation, containing two mutually dependent equations that define the
dynamics of the model.

```
model VanDerPol - "Van der Pol oscillator model"
  Real xstart = 1;
  Real ystart = 1;
  parameter Real lambda = 0.3;
  equation
dot(x) = y;
dot(y) = -x + lambda*(1-x^2)*y;
end VanDerPol;
```

Plot by OpenModelica

```
OMEdit - Demo? Maybe a movie!
The OMC Compiler

- Implemented mainly in MetaModelica and C/C++
- The compiler has 114 packages (in my local working copy)
// Parse the file and get an AST back
ast = Parse.parse(modelicaFile);

// Elaborate the file
scode = SCode.elaborate(ast);

// flatten the simplified code
(cache, dae1) = Inst.instantiate(Env.emptyCache, scode);

// Call the function that optimizes the DAE
optimizeDae(scode, ast, dae, dae, lastClassName);
Simulation Runtime Overview

Two libraries:

- **libc_runtime.a**
  - Runtime used by the generated functions in the model
  - Linked with the model

- **libsim.a**
  - Runtime used for simulations, it contains solver implementations and a main function for the simulation
OMC Simulation Runtime Library

DATA *globalData: simulation_runtime.h
simParams: start, stop, stepSize, outputSteps, tolerance, method

main: simulation_runtime.cpp

globalData = initializeDataStruc(FLAGS);
setLocalData(globalData);
read_input(globalData, simParams);
switch (method)
    "dassl": dassl_main(simParams);
    "euler": euler_main(simParams);
deInitializeDataStruct(DATA, FLAGS);

dassl_main: solver_dasrt.cpp
euler_main: solver_euler.cpp
read_input: simulation_input.cpp

OMC Generated Code

DATA *localData

initializeDataStruc
setLocalData
deInitializeDataStruc
OMC Simulation Runtime Library

DATA *globalData: simulation_runtime.h

simParams: start, stop, stepSize, outputSteps, tolerance, method
dassl main: solver_dasrt.cpp

// set the solver parameters and calculate step from
simParams
initializeEventData(); initializeResult(numpoints, globalData);
bound_parameters(); initial_function();
storeExtrapolationData();
initialize(init_method);
function_updateDependents();
CheckForInitialEvents(globalData->timeValue);
StartEventIteration(globalData->timeValue);
// calculate initial derivatives
functionODE();
// calculate initial output values
functionDAE_output(); functionDAE_output2();
// take a tiny step
tout = globalData->timeValue + epsilon;
function_updateDependents(); saveall(); emit();
calcEnabledZeroCrossings();
// call the solver for that tiny step
DDASTR(functionDAE_res, function_zeroCrossing, jroot);
checkForInitialZeroCrossings(jroot);
// check if we can continue the simulation
functionDAE_res(globalData); functionDAE_output();
// calculate the next step
tout = newTime(tout, step, stop);
// enter solver loop
storeExtrapolationData: simulation_runtime.cpp
initializeResult: simulation_result.cpp
emit: simulation_result.cpp
initializeEventData: simulation_events.cpp
CheckForInitialEvents: simulation_events.cpp
StartEventIteration: simulation_events.cpp
saveall: simulation_events.cpp
initialize: simulation_init.cpp

OMC Generated Code

DATA *localData

initializeDataStruc
setLocalData
deInitializeDataStruc

bound_parameters
initial_function
functionODE
functionDAE_output
functionDAE_output2
function_updateDependent
functionDAE_res
function_zeroCrossing
Outline

- OpenModelica
  - What is OpenModelica?
  - The past and present

- OpenModelica Technical Overview
  - OMC, OMSHell, OMNotebook
  - OMEdit, ModelicaML, SimForge

- OpenModelica Development Environment
  - MetaModelica
  - The Eclipse Environment

- OpenModelica Latest Developments (2009-2010)
OMC
- Implemented mainly in MetaModelica and C/C++

Modelica
- classes, models, records, functions, packages
- behavior is defined by equations or/and functions
- equations
  - differential algebraic equations and conditional equations

MetaModelica extensions
- local equations
- pattern equations
- match expressions
- high-level data structures: lists, tuples, option and union types
OpenModelica Context

Server: Main Program Including Compiler, Interpreter, etc.

Parse

Corba

Client: Graphic Model Editor

Client: OMSHELL Interactive Session Handler

Client: Eclipse Plugin

SCode

Interactive

Untyped API

Inst

Typed Checked Command API

Ceval

system

plot

etc.
The MDT Eclipse Environment (I)
The MDT Eclipse Environment (II)

- .mo file
- OMC Compiler
- Small Modelica Parser
- AST Information
- Modelica model
  - Modelica Browser
  - Modelica Editor
  - Modelica Code Assistant
- Eclipse
- MetaModelica Builder
  - MetaModelica Build console
- MetaModelica Debugging
The MDT Eclipse Environment (III)

- .mo file
- MMC Compiler
- Eclipse
  - MetaModelica
  - Debugging
    - Modelica Editor
  - Executable + Debugging runtime
Creating Modelica projects (I)

Creation of Modelica projects using wizards
Creating Modelica packages

Creation of Modelica packages using wizards
Creating Modelica classes

Creation of Modelica classes, models, etc, using wizards
Code browsing for easy navigation within Modelica files. Automatic update on file save.
Error detection (I)

Parse error detection on file save
Error detection (II)

Semantic error detection on compilation
Code assistance (I)

Code Assistance on imports
Code assistance (II)

Code Assistance on assignments
Code Assistance on function calls
Code indentation
Code Outline and Hovering Info

Code Outline for easy navigation within Modelica files

Identifier Info on Hovering
Eclipse Debugging Environment

- Type information for all variables
- Browsing of complex data structures
Eclipse environment for ModelicaML

1. System Modeling with ModelicaML
2. Modelica Code Generation
3. System Simulation with Modelica Tools

- Tutorial tomorrow at ModProd 2011!
OpenModelica
  - What is OpenModelica?
  - The past and present

OpenModelica Technical Overview
  - OMC, OMShell, OMNotebook

OpenModelica Development Environment
  - MetaModelica
  - The Eclipse Environment

OpenModelica Latest Developments (2010-2011)
Latest Developments (2010-2011)

2010 - 2011 - Most focus on MSL 3.1 support & some performance

- Support for Modelica Standard Library 3.1 (Media & Fluid in works)

**Front-end**
- MultiBody flattening (OSMC)
- Support for Modelica 3.x and 3.x annotations (OSMC)
- Performance Enhancements
- Stream connectors
- Media & Fluid work is on the way

**Back-end**
- Back-end redesign (Jens, Willi, Martin, Per, Adrian, Kristian)
- Tearing in the back-end (Jens Frenkel)
- Template Code Generation and CSharp backend (Pavol Privitzer, Charles University Prague)
- Interactive Simulations (EADS)
- C++ Code generation (Bosch Rexroth)
- Additional Solvers + Events + Linearization (Willi Braun, FH-Bielefeld)

**General**
- OMEdit - new connection editor
- Bootstrapping OMC (90% finished)
- 2550 commits in subversion from 2010 to Feb. 7, 2011 (double than 2009-2010)
- Bug fixes ~300+ (OSMC)
- Release 1.6.0 (Linux, Mac, Windows)
- Downloads Windows (~16434) , Linux (~8301), Mac (~2816)

**Much More things I don’t remember**
The most evil Modelica Standard Library: Media

- Everything in the Modelica Language Specification is used
  - partial functions in partial packages
  - full packages in partial packages used via the fully qualified path
  - redeclare replaceable model extends x.
  - functions using redelclare replaceable function extends used to set constants in partial packages.

- ... and then some more unspecified things
  - [https://trac.modelica.org/Modelica/ticket/488](https://trac.modelica.org/Modelica/ticket/488) Unspecified function in Modelica.Fluid (should pose no issue)
  - [https://trac.modelica.org/Modelica/ticket/482](https://trac.modelica.org/Modelica/ticket/482) Illegal lookup in Modelica.Media (fixed by Hubertus in a branch)
Action plan to support Media & Fluid

- Simplify flattening (instantiation) by preprocessing phases
  - Remove imports (100%)
  - Remove extends (95%)
  - Remove redeclare (1%)
  - Perform dependency analysis (0%)

- Handle record constants (10%)

- Any other unknown issues

- Hopefully Media & Fluid flattening will start working for Modelica Conference 2011 (March 20)
Next presentation by Martin Sjölund
Thank You!

Questions?

OpenModelica Project
http://www.OpenModelica.org