OpenModelica Environment and Modelica Overview

Peter Fritzson, Adrian Pop, Peter Aronsson

OpenModelica Course at INRIA, 2006 06 08

OpenModelica

- Goal: comprehensive modeling and simulation environment for research, teaching, and industrial usage
- Free, open-source for both academic and commercial use
- Now under Berkley New BSD open source license
- The OpenModelica compiler (OMC) now translated into MetaModelica
- Invitation for open-source cooperation around OpenModelica, tools, and applications

Background

Modelica – the Next Generation Modeling Language

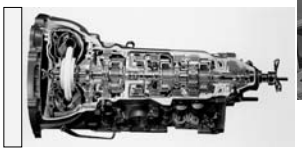
Examples of Complex Industrial Applications

- Robotics
- Automotive
- Aircraft
- Phone Systems
- Power plants
- Heavy Vehicles



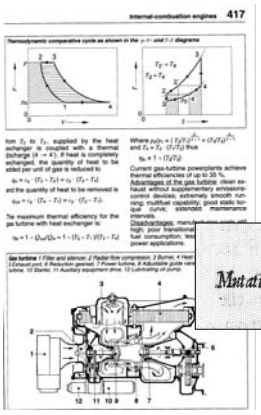






Stored Scientific and Engineering Knowledge

Model knowledge is stored in books and human minds which computers cannot access



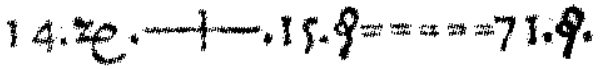
"The change of motion is proportional to the motive force impressed "
– Newton

Lex. II.

Mutationem motus proportionalem esse vi motrici impressæ, & fieri secundum lineam rectam qua vis illa imprimitur.

The Form – Equations

- Equations were used in the third millennium B.C.
- Equality sign was introduced by Robert Recorde in 1557



Newton still wrote text (Principia, vol. 1, 1686)
"The change of motion is proportional to the motive force impressed "

CSSL (1967) introduced a special form of "equation":

```
variable = expression
v = INTEG(F)/m
```

Programming languages usually do not allow equations!

Modelica – The Next Generation Modeling Language

Declarative language

Equations and mathematical functions allow acausal modeling, high level specification, increased correctness

Multi-domain modeling

Combine electrical, mechanical, thermodynamic, hydraulic, biological, control, event, real-time, etc...

Everything is a class

Strongly typed object-oriented language with a general class concept, Java & MATLAB-like syntax

Visual component programming

Hierarchical system architecture capabilities

Efficient, non-proprietary

Efficiency comparable to C; advanced equation compilation, e.g. 300 000 equations, ~150 000 lines on standard PC

Modelica Language Properties

- **Declarative** and **Object-Oriented**
- **Equation-based**; continuous and discrete equations
- **Parallel** process modeling of real-time applications, according to synchronous data flow principle
- **Functions** with algorithms without global side-effects (but local data updates allowed)
- **Type system** inspired by Abadi/Cardelli
- **Everything is a class** – Real, Integer, models, functions, packages, parameterized classes....

Object Oriented Mathematical Modeling with Modelica

- The static *declarative structure* of a mathematical model is emphasized
- OO is primarily used as a *structuring concept*
- OO *is not* viewed as dynamic object creation and sending messages
- *Dynamic model* properties are expressed in a *declarative way* through equations.
- Acausal classes supports *better reuse of modeling and design knowledge* than traditional classes

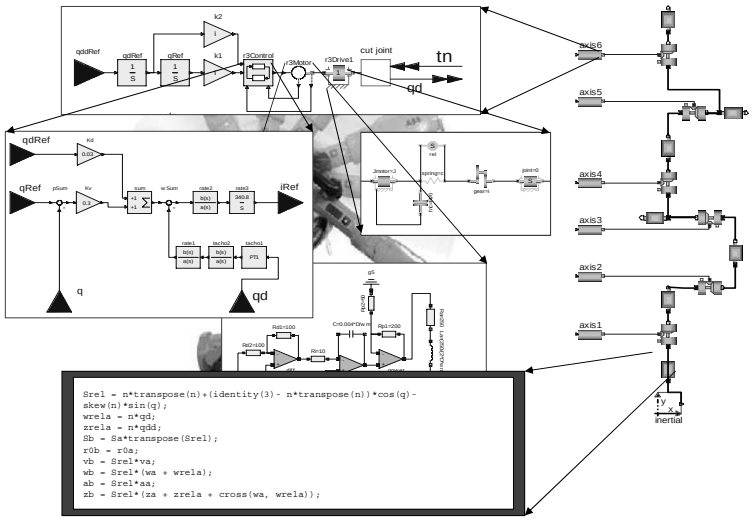
Brief Modelica History

- First Modelica design group meeting in fall 1996
  - International group of people with expert knowledge in both language design and physical modeling
  - Industry and academia
- Modelica Versions
  - 1.0 released September 1997
  - 2.0 released March 2002
  - Latest version, 2.2 released March 2005
- Modelica Association established 2000
  - Open, non-profit organization

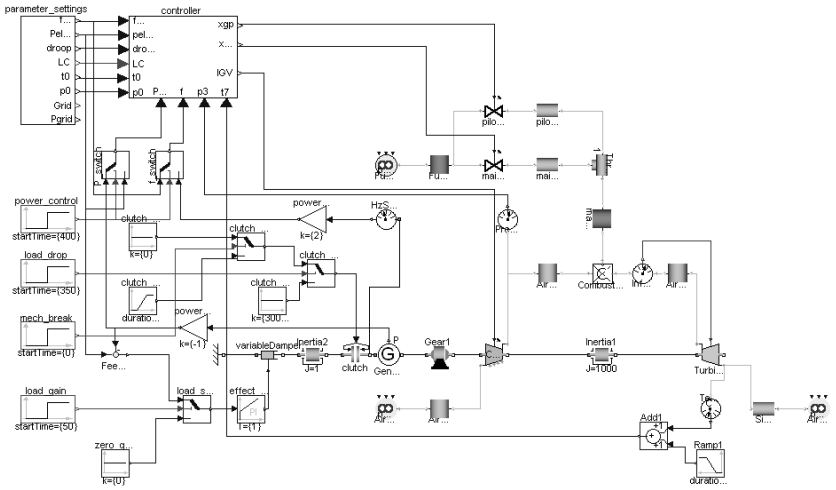
Modelica Conferences

- The 1st International Modelica conference October, 2000
- The 2nd International Modelica conference March 18-19, 2002
- The 3rd International Modelica conference November 5-6, 2003 in Linköping, Sweden
- The 4th International Modelica conference March 6-7, 2005 in Hamburg, Germany
- The 5th International Modelica conference September 4-5, 2006 in Vienna, Austria

Modelica Model Example – Industry Robot

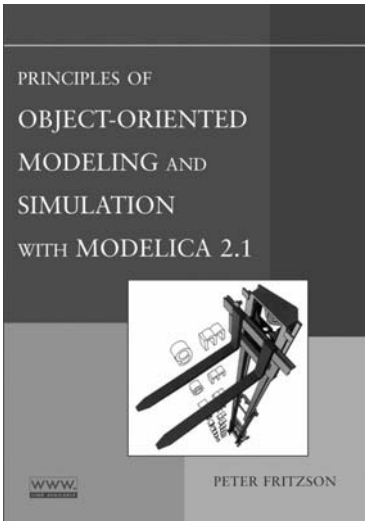


Modelica Model Example
GTX Gas Turbine Power Cutoff Mechanism



Courtesy of Siemens Industrial Turbomachinery AB

Recent Book, 2004



Peter Fritzson
Principles of Object Oriented Modeling and Simulation with Modelica 2.1

Wiley-IEEE Press

940 pages

Book web page:
`www.mathcore.com/drmodelica`

The OpenModelica Environment

OpenModelica End-Users vs. Developers

- OpenModelica End-Users
  - People who use OpenModelica for modeling and simulation
- OpenModelica Developers
  - People who develop/contribute to parts in the OpenModelica environment including the OpenModelica compiler

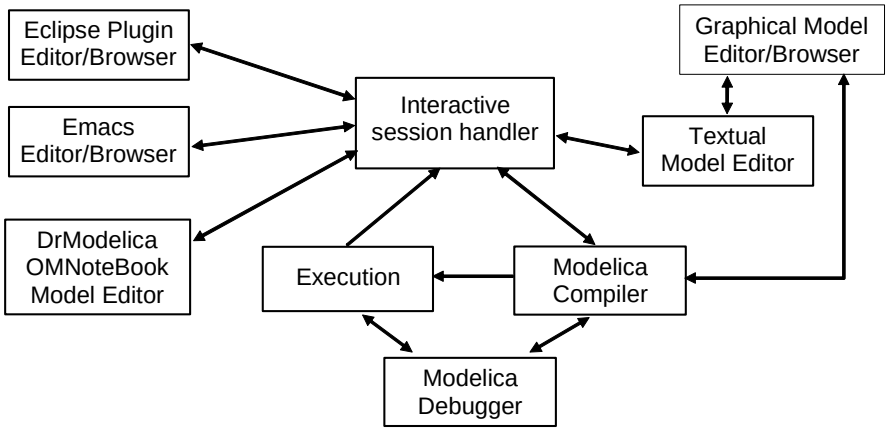
OpenModelica End-User Subsystems

- OpenModelica End-User Subsystems – a pre-packaged kit containing tools needed for modeling, simulation, teaching
- OpenModelica Compiler (OMC) – compiles and executes/simulates Modelica models
- OMShell – interactive session handler for Modelica scripting
- OMNotebook – interactive electronic notebook for Modelica teaching (with DrModelica), scripting, and documentation
- OpenModelica MDT – Eclipse Plugin (Modelica Development Tooling), e.g. for library development (c.f. JDT – Java, CDT,)
- Graphic Model Editor from MathCore
  (only binary, but free for university usage)

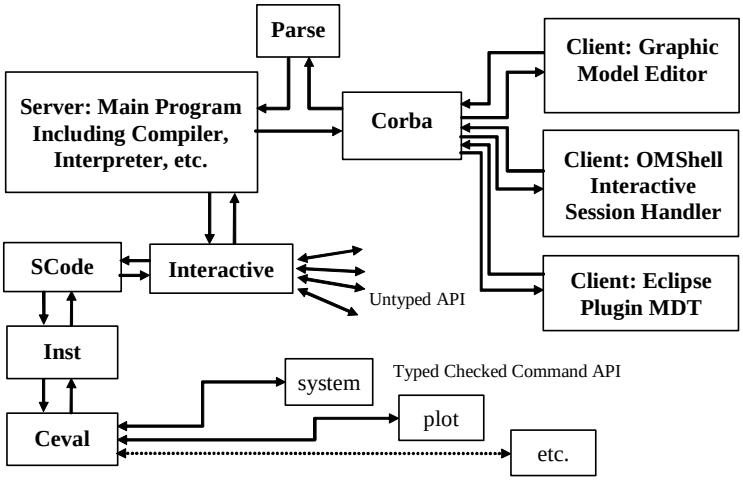
OpenModelica Development Toolkit (OMDev) – to Simplify Open Source Development

- OMDev is a pre-packaged pre-compiled kit containing all tools needed for OpenModelica development. Just unpack and start working on your platform. (Windows, (Linux))
- MetaModelica Compiler (MMC) – for developing OMC
- OpenModelica Compiler (OMC) – for browsing support
- Eclipse plugin MDT – (Modelica Development Tooling), e.g. for compiler (OMC) development
- Pre-compiled Corba (MICO) for tool communication
- Packaged Gnu compiler (Mingw version for Windows)
- Emacs mode
- Online (web) Subversion for version handling
- Online (web) Bugzilla for bug reporting
- Automatic regression testing using a test suite
- (Soon: release of interactive debugger)

OpenModelica Environment Architecture



OpenModelica Client-Server Architecture



Released in OpenModelica 1.4.0

- OpenModelica compiler/interpreter – OMC
- Interactive session handler – OMShell
- OpenModelica Notebook with DrModelica – OMNotebook
- OpenModelica Eclipse plugin MDT

- Preliminary versions:
  - Graphic Editor – Beta version available
  - Debugger – soon released, beta version being improved
  - Emacs mode – available

OpenModelica Compiler/Interpreter

- New version (1.4.0) released May 15, 2006
- Currently implemented in 100 000 lines of MetaModelica
- Includes code generation, BLT-transformation, index reduction, connection to DASSL, etc.
- Most of the Modelica language including classes, functions, inheritance, modifications, import, etc.
- Hybrid/Discrete event support

Corba Client-Server API

- Simple text-based (string) communication in Modelica Syntax
- API supporting model structure query and update

Example Calls:
Calls fulfill the normal Modelica function call syntax.:

```
saveModel("MyResistorFile.mo",MyResistor)
```

will save the model MyResistor into the file “MyResistorFile.mo”.

For creating new models it is most practical to send a model, e.g.:

```
model Foo   end Foo;
```

or, e.g.,

```
connector Port   end Port;
```


Some of the Corba API functions

saveModel(A1<string>,A2<cref>)	Saves the model (A2) in a file given by a string (A1). This call is also in typed API.
loadFile(A1<string>)	Loads all models in the file. Also in typed API. Returns list of names of top level classes in the loaded files.
loadModel(A1<cref>)	Loads the model (A1) by looking up the correct file to load in `$MODELICAPATH`. Loads all models in that file into the symbol table.
deleteClass(A1<cref>)	Deletes the class from the symbol table.
addComponent(A1<ident>,A2<cref>, A3<cref>,annotate=<expr>)	Adds a component with name (A1), type (A2), and class (A3) as arguments. Optional annotations are given with the named argument `annotate`.
deleteComponent(A1<ident>, A2<cref>)	Deletes a component (A1) within a class (A2).
updateComponent(A1<ident>, A2<cref>, A3<cref>,annotate=<expr>)	Updates an already existing component with name (A1), type (A2), and class (A3) as arguments. Optional annotations are given with the named argument `annotate`.
addClassAnnotation(A1<cref>, annotate=<expr>)	Adds annotation given by A2(in the form `annotate= classmod(...))` to the model definition referenced by A1. Should be used to add Icon Diagram and Documentation annotations.
getComponents(A1<cref>)	Returns a list of the component declarations within class A1: `{{Atype,varidA,"commentA"},{Btype,varidB,"commentB"}, {...}}`
getComponentAnnotations(A1<cref>)	Returns a list `{...}` of all annotations of all components in A1, in the same order as the components, one annotation per component.
getComponentCount(A1<cref>)	Returns the number (as a string) of components in a class, e.g return `"2"` if there are 2 components.
getNthComponent(A1<cref>,A2<int>)	Returns the belonging class, component name and type name of the nth component of a class, e.g. `"A.B.C,R2,Resistor"`, where the first component is numbered 1.
getNthComponentAnnotation(A1<cref>,A2<int>)	Returns the flattened annotation record of the nth component (A2) (the first is has no 1) within class/component A1. Consists of a comma separated string of 15 values, see Annotations in Section 2.4.4 below, e.g `"false,10,30,..."`
getNthComponentModification(A1<cref>,A2<int>)??	Returns the modification of the nth component (A2) where the first has no 1) of class/component A1.
getInheritanceCount(A1<cref>)	Returns the number (as a string) of inherited classes
getNthInheritedClass(A1<cref>,	Returns the type name of the nth inherited class of a class. The first class has number 1.

Platforms

- All OpenModelica GUI tools (OMShell, OMNotebook, ...) are developed on the Qt4 GUI library, portable between Windows, Linux, Mac
- Both compilers (OMC, MMC) are portable between the three platforms

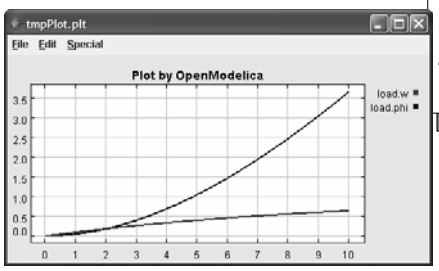
- Windows – currently main development and release platform
- Linux – available
- Mac (Berkeley) Unix – planned

Interactive Session Handler – on dcmotor Example

(Session handler called OMShell – OpenModelica Shell)

```
>>simulate(dcmotor,startTime=0.0,stopTime=10.0)
>>plot({load.w,load.phi})
```

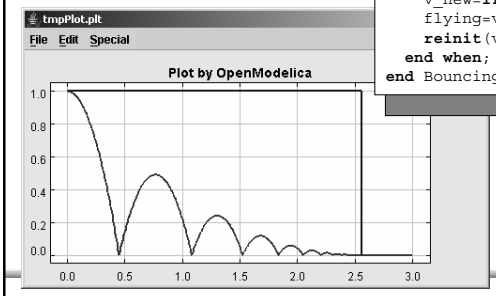
```
model dcmotor
  Modelica.Electrical.Analog.Basic.Resistor r1(R=10);
  Modelica.Electrical.Analog.Basic.Inductor i1;
  Modelica.Electrical.Analog.Basic.EMF emf1;
  Modelica.Mechanics.Rotational.Inertia load;
  Modelica.Electrical.Analog.Basic.Ground g;
  Modelica.Electrical.Analog.Sources.ConstantVoltage v;
equation
  connect(v.p,r1.p);
  connect(v.n,g.p);
  connect(r1.n,i1.p);
  connect(i1.n,emf1.p);
  connect(emf1.n,g.p);
  connect(emf1.flange_b,load.flange_a);
end dcmotor;
```



Event Handling by OpenModelica – BouncingBall

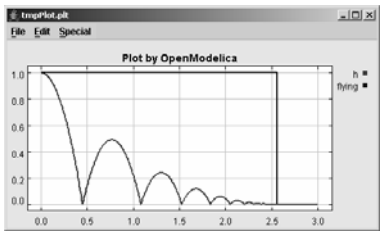
```
>>simulate(BouncingBall,
           stopTime=3.0);
>>plot({h,flying});
```

```
model BouncingBall
  parameter Real e=0.7 "coefficient of restitution";
  parameter Real g=9.81 "gravity acceleration";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";
  Boolean flying(start=true) "true, if ball is flying";
  Boolean impact;
  Real v_new;
equation
  impact=h <= 0.0;
  der(v)=if flying then -g else 0;
  der(h)=v;
  when {h <= 0.0 and v <= 0.0,impact} then
    v_new=if edge(impact) then -e*pre(v) else 0;
    flying=v_new > 0;
    reinit(v, v_new);
  end when;
end BouncingBall;
```



Run Scripts in OpenModelica

- RunScript command interprets a .mos file
- .mos means MOdelica Script file
- Example:
  `>> runScript("sim_BouncingBall.mos")`



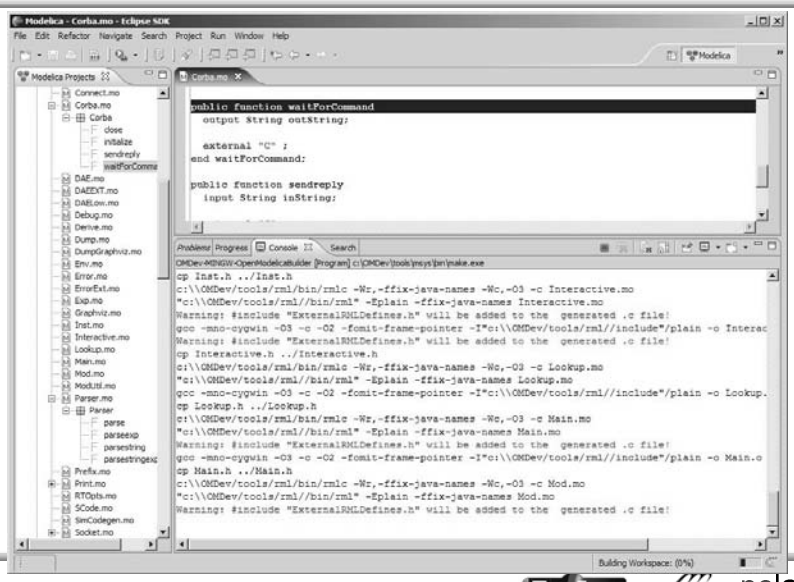
The file sim_BouncingBall.mos :

```
loadFile("BouncingBall.mo");
simulate(BouncingBall, stopTime=3.0);
plot({h,flying});
```

OpenModelica MDT – Eclipse Plugin

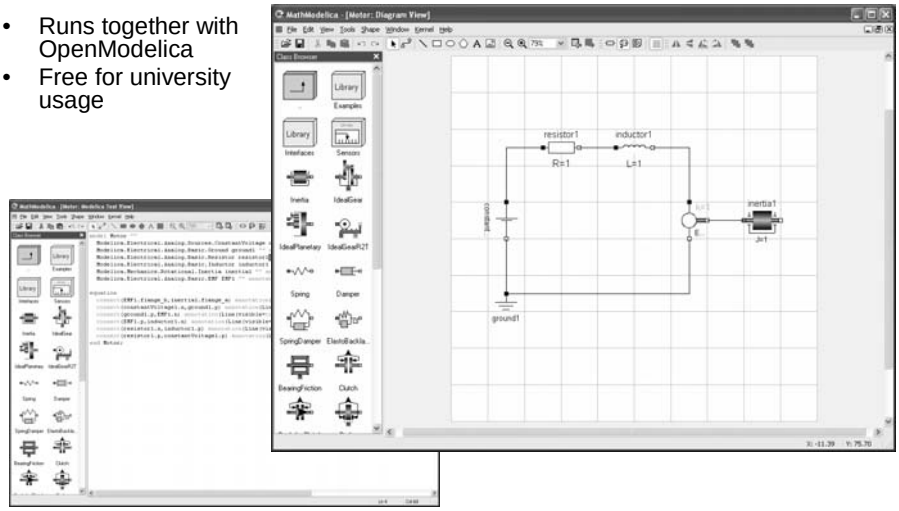
- Browsing of Modelica/MetaModelica packages, classes, functions
- Automatic building of executables
- Separate compilation
- Syntax highlighting
- Code completion, Code query support for developers
- Automatic Indentation

Eclipse MDT in Action – Browsing and Building OMC



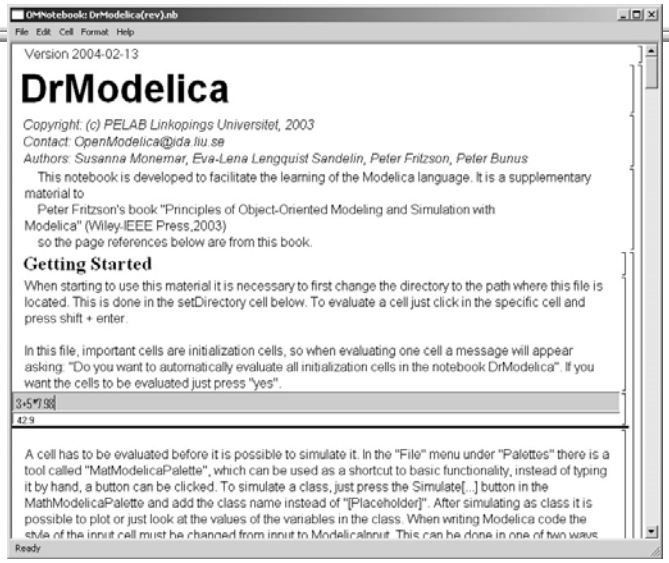
New Graphic Model Editor
(From MathCore; runs on Windows, Linux)

- Runs together with OpenModelica
- Free for university usage

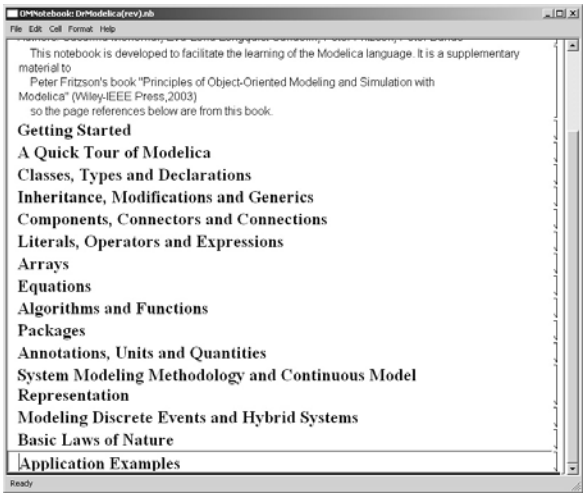


OpenModelica Simple Electronic Notebook with DrModelica

- Primarily for teaching
- OMNotebook Does not need Mathematica

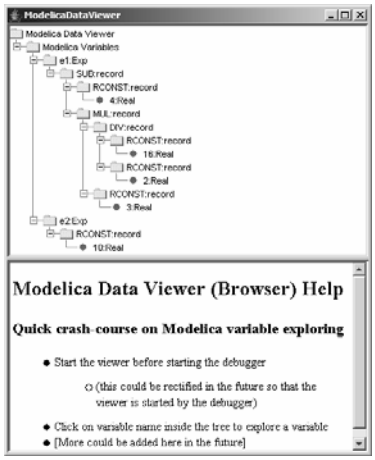


Interactive Contents in DrModelica Contains Examples and Exercises from Modelica Book



Recent Book, 2004:

OpenModelica Algorithmic Code Debugger (prel.)



Meta-Modelica Compiler (MMC)

- Supports extended subset of Modelica
- Used for development of OMC
- Some MetaModelica Language properties:
  - Modelica syntax and base semantics
  - Pattern matching (named/positional)
  - Local equations (local within expression)
  - Recursive tree data structures
  - Lists and tuples
  - Garbage collection of heap-allocated data
  - Arrays (with local update as in standard Modelica)
  - Polymorphic functions
  - Function parameters to functions
  - Simple builtin exception (failure) handling mechanism

Conclusions

- OpenModelica version 1.3.1 released Nov 2005
- Recent OpenModelica version 1.4.0 released May 15, 2006
  - OpenModelica in MetaModelica
  - Many bugfixes
  - OpenModelica MDT Eclipse plugin
  - Graphic model editor (available for beta testing)
- Cooperation and feedback welcome!

- **www.ida.liu.se/projects/OpenModelica Download OpenModelica**
- **www.mathcore.com/DrModelica Modelica book page**
- **www.modelica.org Modelica Association**
- **Emails: {petfr,adrpo,petar}@ida.liu.se, OpenModelicaInterest@ida.liu.se**