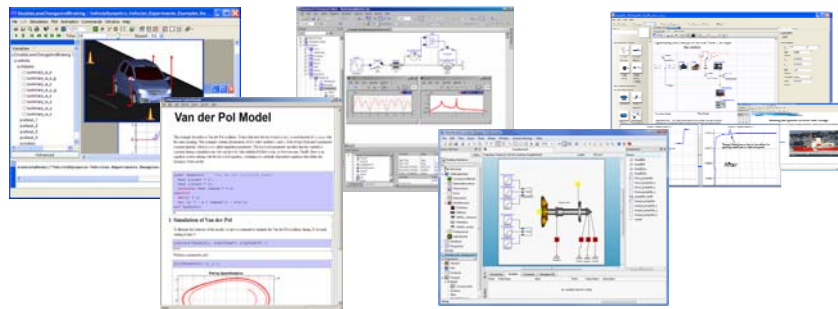
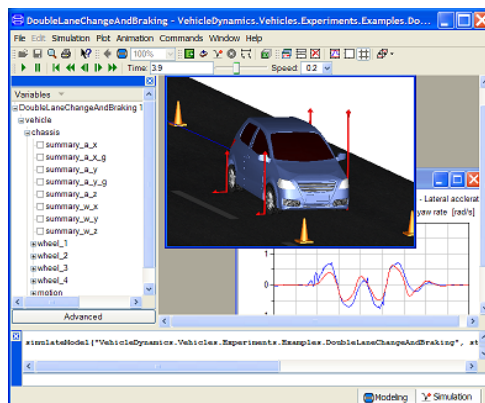


Modelica Environments and OpenModelica

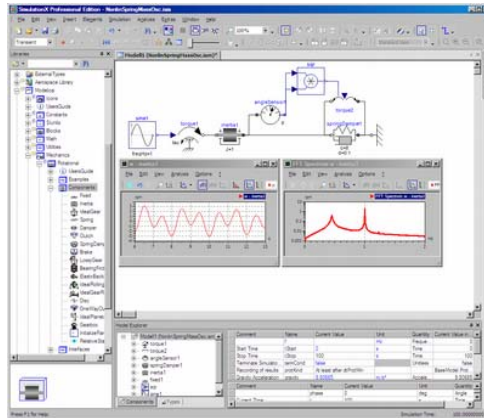


Dymola



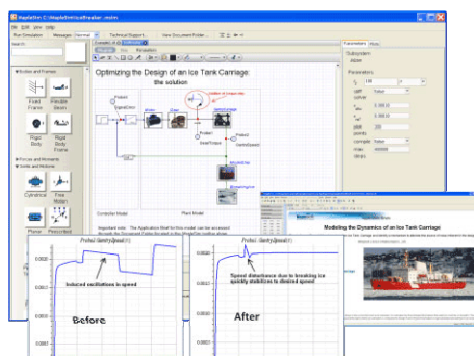
- Dynasim (Dassault Systemes)
- Sweden
- First Modelica tool on the market
- Main focus on automotive industry
- www.dynasim.com

Simulation X



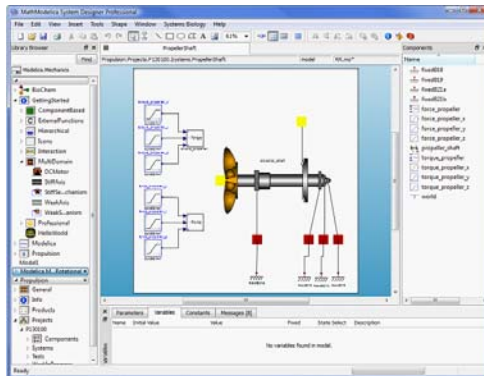
- ITI
- Germany
- Mechatronic systems
- www.simulationx.com

MapleSim



- Maplesoft
- Canada
- Recent Modelica tool on the market
- Integrated with Maple
- www.maplesoft.com

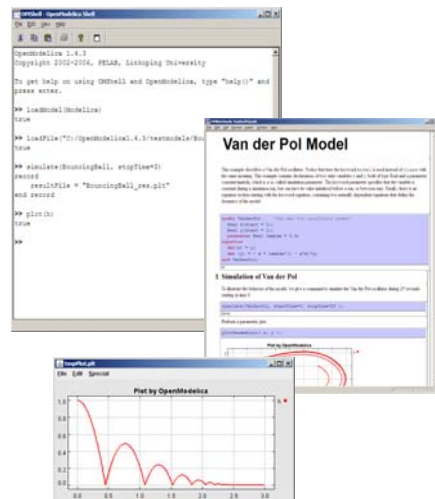
MathModelica



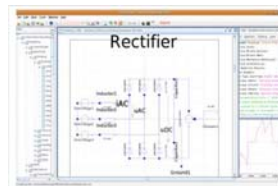
- MathCore
- Sweden
- Released 2006
- General purpose
- Mathematica connection
- www.mathcore.com

The OpenModelica Environment
www.OpenModelica.org

OpenModelica and simForge



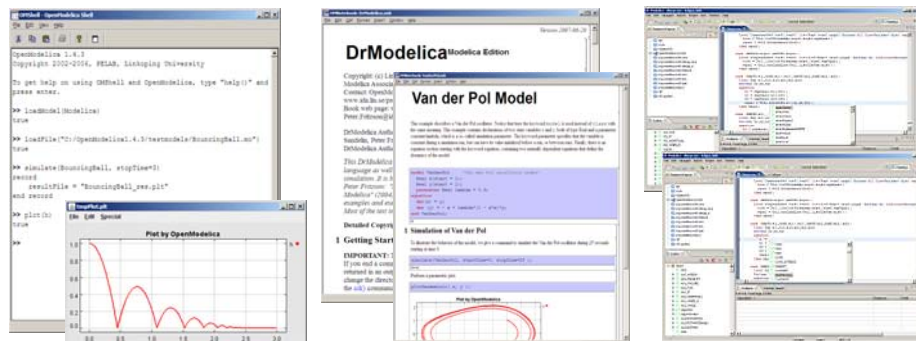
- OpenModelica
- Open Source Modelica Consortium (OSMC)
- Sweden and other countries
- Open source
- www.openmodelica.org



- Graphical editor simForge
- Politecnico di Milano, Italy
- Runs together with OpenModelica
- Open source

OpenModelica

- **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
 - **ModelicaML UML Profile**
 - **MetaModelica** extension

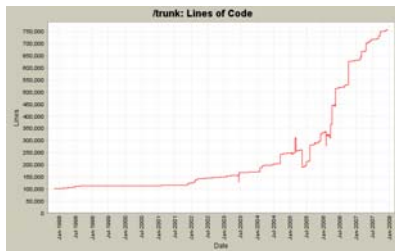


Open Source Modelica Consortium

Open-source community services

- Website and Support Forum
- Version-controlled source base
- Bug database
- Development courses
- www.openmodelica.org

Code Statistics



Founded Dec 4, 2007

Industrial members (12)

- ABB Corporate Research
- Bosch-Rexroth AG, Germany
- Siemens Turbo Machinery AB
- Creative Connections, Prague
- Equa Simulation AB, Sweden
- IFP, Paris, France
- MostforWater, Belgium
- MathCore Engineering AB
- MapleSoft, Canada
- TLK Thermo, Germany
- VTT, Finland
- XRG Simulation AB, Germany

University members (9)

- Linköping University, Sweden
- Hamburg University of Technology/TuTech, Institute of Thermo-Fluid Dynamics, Germany
- Technical University of Braunschweig, the Institut of Thermodynamik, Germany
- Université Laval, the modelEAU group, Canada
- Griffith University, Australia
- University of Queensland, Australia
- Politecnico di Milano, Italy
- Mälardalen University, Sweden
- Technical University Dresden, Germany

OMNotebook Electronic Notebook with DrModelica

- Primarily for teaching
- Interactive electronic book
- Platform independent

Commands:

- *Shift-return* (evaluates a cell)
- File Menu (open, close, etc.)
- Text Cursor (vertical), Cell cursor (horizontal)
- Cell types: text cells & executable code cells
- Copy, paste, group cells
- Copy, paste, group text
- Command Completion (shift-tab)

OMNotebook: DrModelica.aux

File Edit Cell Format Insert Window Help

Version 2006-04-11

DrModelica^{Modelica Edition}

Copyright: (c) Linköping University, PELAB, 2003-2006, Wiley-IEEE Press, Modelica Association.
 Contact: OpenModelica@ida.liu.se, OpenModelica.org Project web site:
www.ida.liu.se/projects/OpenModelica
 Book web page: www.mathcore.com/DrModelica, Book author: Peter.Fritzon@ida.liu.se

DrModelica Authors (2003 version): Susanna Moenar, Eva-Lena Ljungqvist Sandelin, Peter Fritzon, Peter Blom
 DrModelica Authors (2005 and later updates): Peter Fritzon

This DrModelica notebook has been developed to facilitate learning the Modelica language as well as providing an introduction to object-oriented modeling and simulation. It is based on and is supplementary material to the Modelica book: Peter Fritzon: "Principles of Object-Oriented Modeling and Simulation with Modelica" (2004), 940 pages, Wiley-IEEE Press, ISBN 0-471-47163-1. All of the examples and exercises in DrModelica and the page references are from that book. Most of the text in DrModelica is also based on that book.

Detailed Copyright and Acknowledgment Information

Getting Started Using OMNotebook

OpenModelica commands

Berkeley license OpenModelica

1 A Quick Tour of Modelica

1.1 Getting Started - First Basic Examples

There is a long tradition that the first sample program in any computer language is a trivial program printing the string "Hello, World" (p. 19 in Peter Fritzon's book). Since Modelica is an equation based language, printing a string does not make much sense. Instead, our Hello World Modelica program solves a trivial differential equation. The second example shows how you can write a model that solves a [Differential Algebraic Equation System](#) (p. 19). In the [Van der Pol](#) (p. 22) example declaration as well as initialization and prefix usage are shown in a slightly more complicated way.

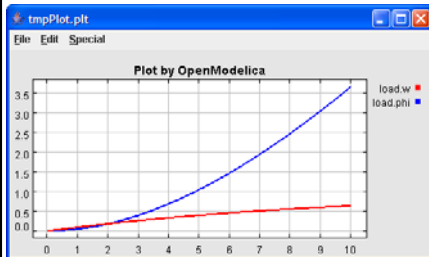
1.2 Classes and Instances

In Modelica objects are created implicitly just by [Declaring Instances of Classes](#) (p. 26). Almost anything in Modelica is a class, but there are some keywords for specific use of the class concept, called

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})
```

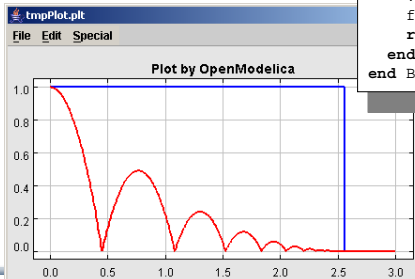
```
model dcmotor
  Modelica.Electrical.Analog.Basic.Resistor r1(R=10);
  Modelica.Electrical.Analog.Basic.Inductor il;
  Modelica.Electrical.Analog.Basic.EMF emf1;
  Modelica.Mechanics.Rotational.Inertia load;
  Modelica.Electrical.Analog.Basic.Ground g;
  Modelica.Electrical.Analog.Sources.ConstantVoltage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

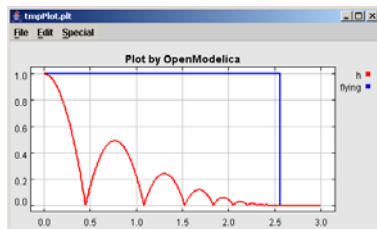
```
model BouncingBall
  parameter Real e=0.7 "coefficient of restitution";
  parameter Real g=9.81 "gravity acceleration";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";
  Boolean flying(start=true) "true, if ball is flying";
  Boolean impact;
  Real v_new;
equation
  impact=h <= 0.0;
  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 MModelica Script file
- Example:

```
>> runScript("sim_BouncingBall.mos")
```

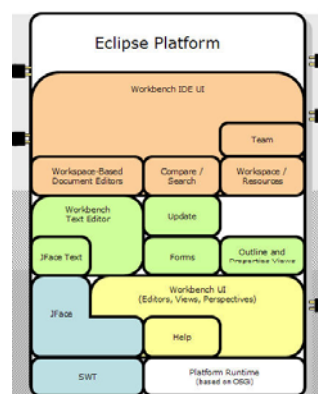


The file `sim_BouncingBall.mos` :

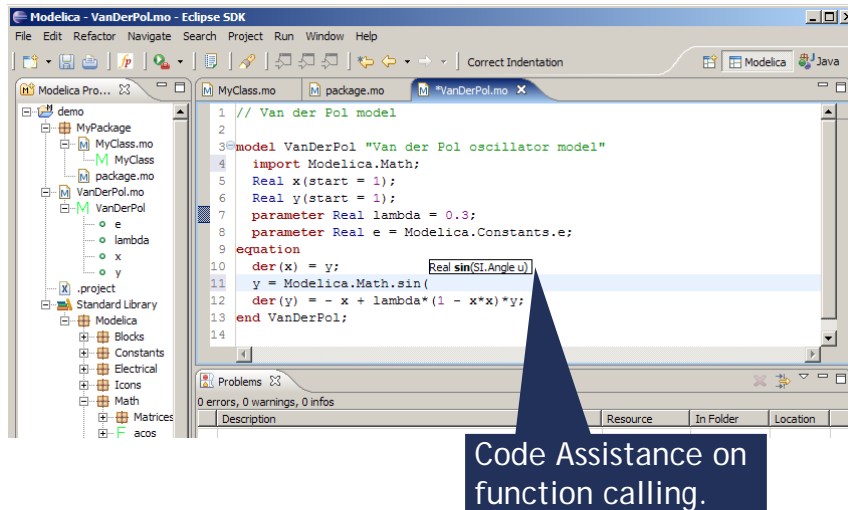
```
loadFile("BouncingBall.mo");  
simulate(BouncingBall, stopTime=3.0);  
plot({h, flying});
```

OpenModelica MDT – Eclipse Plugin

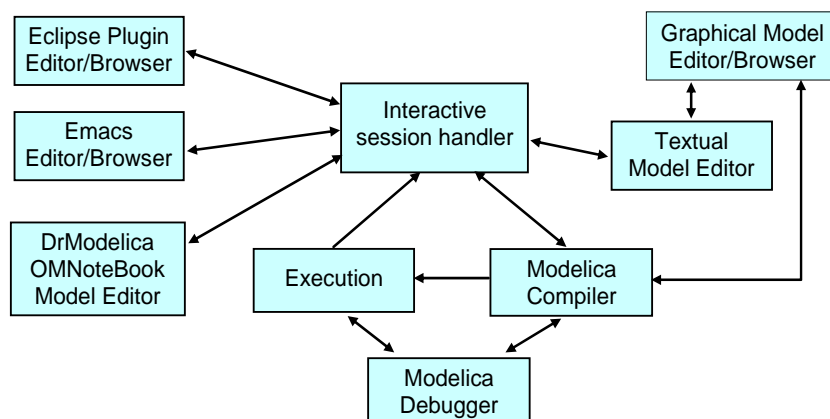
- Browsing of packages, classes, functions
- Automatic building of executables;
separate compilation
- Syntax highlighting
- Code completion,
Code query support for developers
- Automatic Indentation
- Debugger
(Prel. version for algorithmic subset)



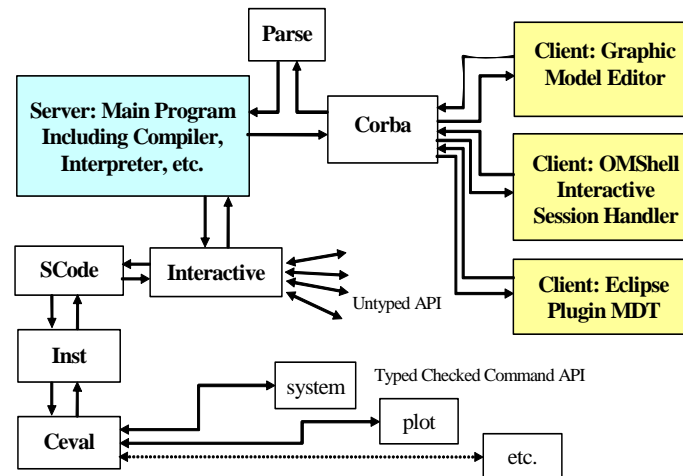
OpenModelica MDT – Usage Example



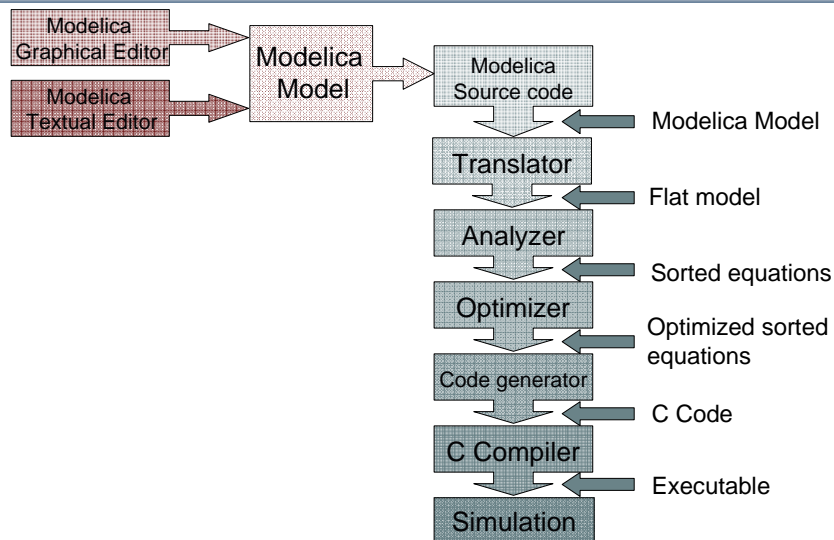
OpenModelica Environment Architecture



OpenModelica Client-Server Architecture



Translation of Models to Simulation Code



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

<code>saveModel (A1<string>,A2<cref>)</code>	Saves the model (A2) in a file given by a string (A1). This call is also in typed API.
<code>loadFile (A1<string>)</code>	Loads all models in the file. Also in typed API. Returns list of names of top level classes in the loaded files.
<code>loadModel (A1<cref>)</code>	Loads the model (A1) by looking up the correct file to load in \$MODELICAPATH. Loads all models in that file into the symbol table.
<code>deleteClass (A1<cref>)</code>	Deletes the class from the symbol table.
<code>addComponent (A1<ident>,A2<cref>, A3<cref>,annotate=<expr>)</code>	Adds a component with name (A1), type (A2), and class (A3) as arguments. Optional annotations are given with the named argument <code>annotate</code> .
<code>deleteComponent (A1<ident>, A2<cref>)</code>	Deletes a component (A1) within a class (A2).
<code>updateComponent (A1<ident>, A2<cref>, A3<cref>,annotate=<expr>)</code>	Updates an already existing component with name (A1), type (A2), and class (A3) as arguments. Optional annotations are given with the named argument <code>annotate</code> .
<code>addClassAnnotation (A1<cref>, annotate=<expr>)</code>	Adds annotation given by A2(in the form <code>annotate= classmod(...)</code>) to the model definition referenced by A1. Should be used to add Icon Diagram and Documentation annotations.
<code>getComponents (A1<cref>)</code>	Returns a list of the component declarations within class A1: { {Atype, varidA, "commentA"}, {Btype, varidB, "commentB"}, {...} }
<code>getComponentAnnotations (A1<cref>)</code>	Returns a list { ... } of all annotations of all components in A1, in the same order as the components, one annotation per component.
<code>getComponentCount (A1<cref>)</code>	Returns the number (as a string) of components in a class, e.g return "2" if there are 2 components.
<code>getNthComponent (A1<cref>,A2<int>)</code>	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.
<code>getNthComponentAnnotation (A1<cref>,A2<int>)</code>	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,..."
<code>getNthComponentModification (A1<cref>,A2<int>)??</code>	Returns the modification of the nth component (A2) where the first has no 1) of class/component A1.
<code>getInheritedClasses (A1<cref>)</code>	Returns a list of the names of inherited classes of a class.
<code>getNthInheritedClass (A1<cref>, A2<int>)</code>	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. Also used for development
- Mac – available

OpenModelica – Recent Developments

- Dec 2008. OSMC Board decides to focus on improving the OpenModelica compiler for Modelica libraries during 2009
- Dec 2008. MathCore contributes 1 man-year worth of source code for the flattening frontend.
- Jan-Sept 2009. Development mostly on the compiler frontend
- Sept 2009. OpenModelica release 1.5, containing approx 2 man-years development compared to version 1.4.5. (Beta release available today).