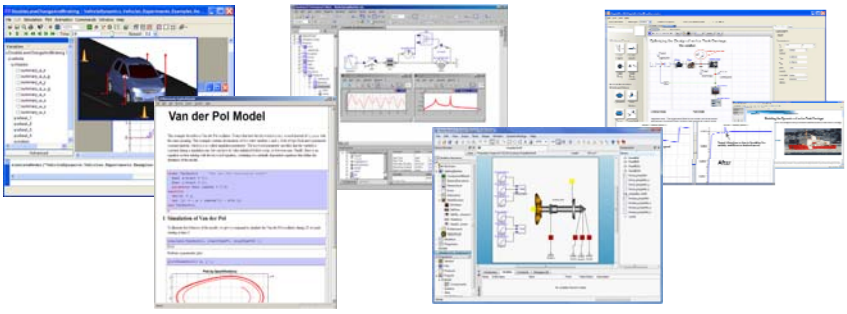
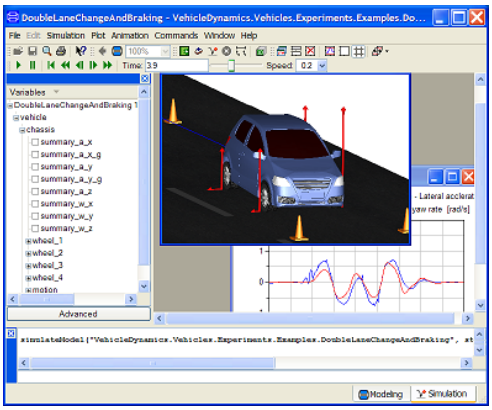
Modelica Environments and OpenModelica

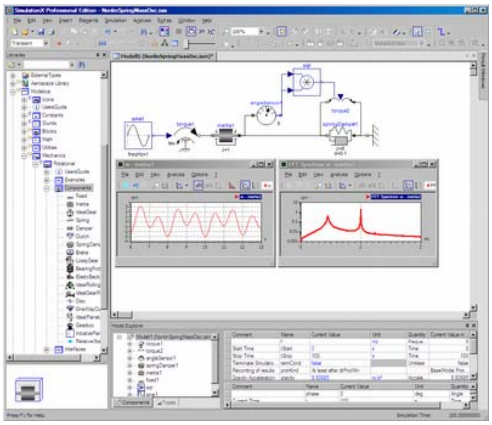


Dymola



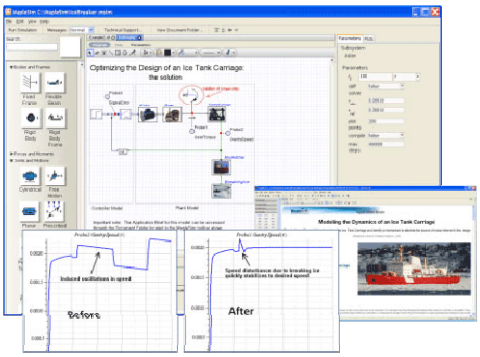
- Dynasim (Dassault Systemes)
- Sweden
- First Modelica tool on the market
- Main focus on automotive industry
- www.dynasim.com

Simulation X



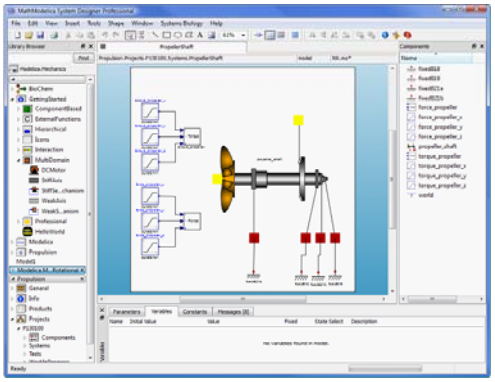
- ITI
- Germany
- Mechatronic systems
- www.simulationx.com

MapleSim



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

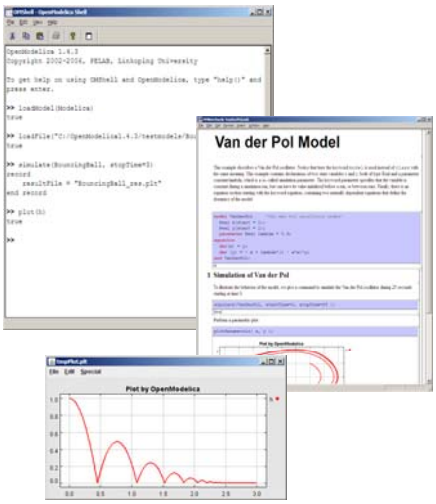
MathModelica



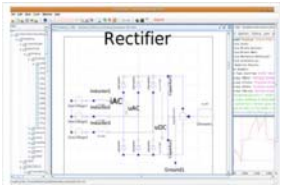
- MathCore
- Sweden
- Released 2006
- General purpose
- Mathematica connection
- www.mathcore.com

The OpenModelica Environment
www.OpenModelica.org

OpenModelica and simForge



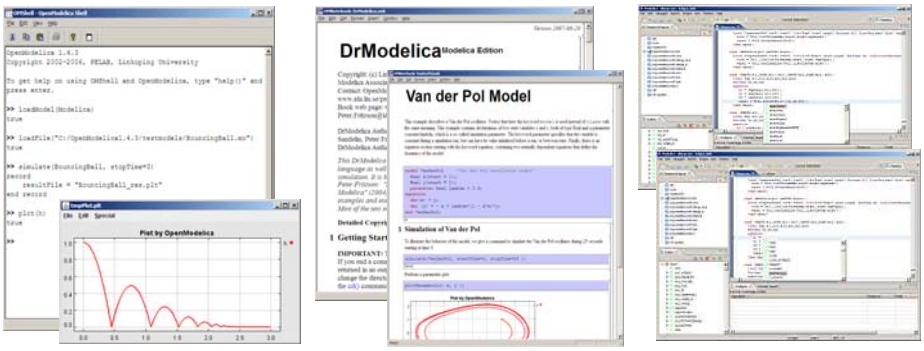
- OpenModelica
- Open Source Modelica Consortium (OSMC)
- Sweden and other countries
- Open source
- www.openmodelica.org



- 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

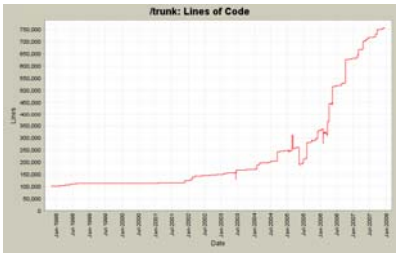


Open Source Modelica Consortium

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)

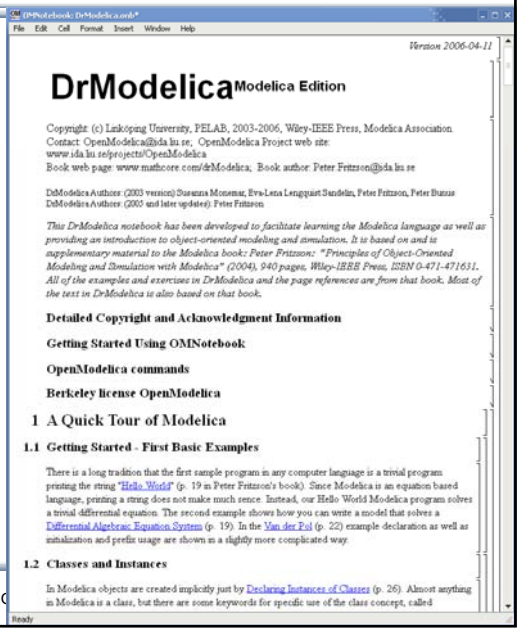
- **Linköping University, Sweden**
- **Hamburg University of Technology/TuTech, Institute of Thermo-Fluid Dynamics, Germany**
- **Technical University of Braunschweig, the Institut of Thermodynamik, Germany**
- **Université Laval, the modelEAU group, Canada**
- **Griffith University, Australia**
- **University 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:

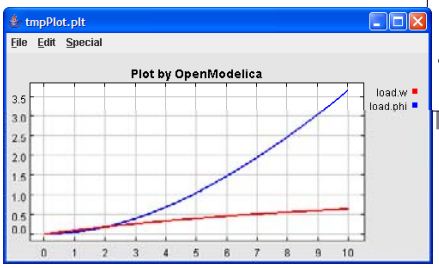
- *Shift-return (evaluates a cell)*
- File Menu (open, close, etc.)
- Text Cursor (vertical), Cell cursor (horizontal)
- Cell types: text cells & executable code cells
- Copy, paste, group cells
- Copy, paste, group text
- Command Completion (shift-tab)



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})
```

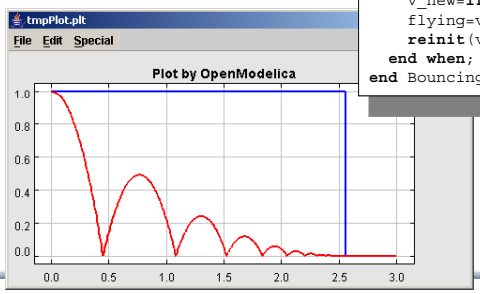
```
model dcmotor
  Modelica.Electrical.Analog.Basic.Resistor r1(R=10);
  Modelica.Electrical.Analog.Basic.Inductor i1;
  Modelica.Electrical.Analog.Basic.EMF emf1;
  Modelica.Mechanics.Rotational.Inertia load;
  Modelica.Electrical.Analog.Basic.Ground g;
  Modelica.Electrical.Analog.Sources.ConstantVoltage 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

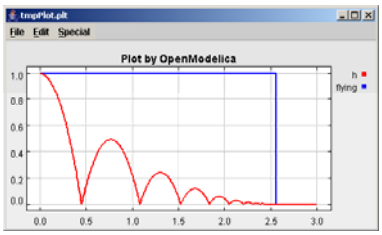
```
>>simulate(BouncingBall,
           stopTime=3.0);
>>plot({h,flying});
```

```
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")`

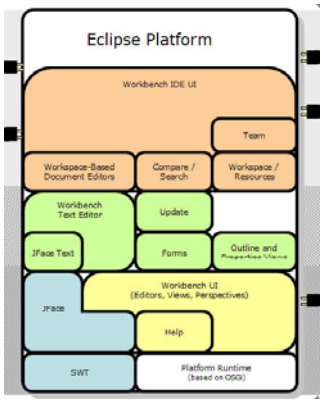


The file sim_BouncingBall.mos :

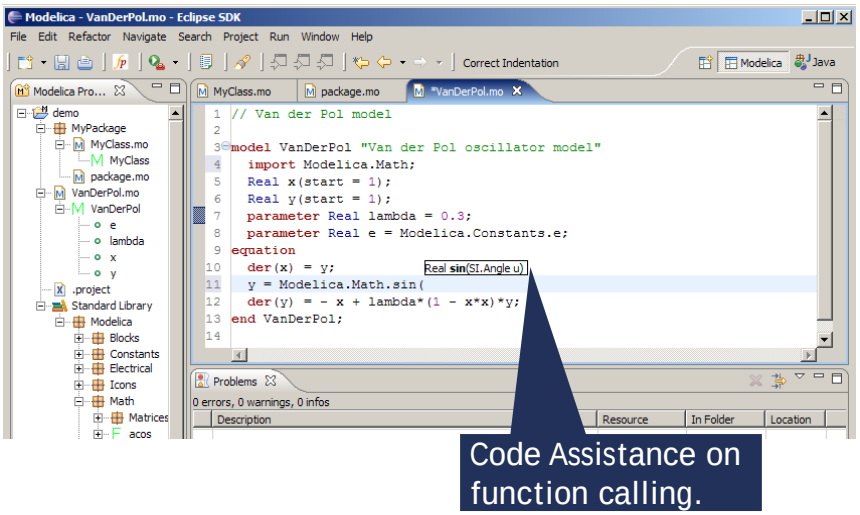
```
loadFile("BouncingBall.mo");
simulate(BouncingBall, stopTime=3.0);
plot({h,flying});
```


OpenModelica MDT – Eclipse Plugin

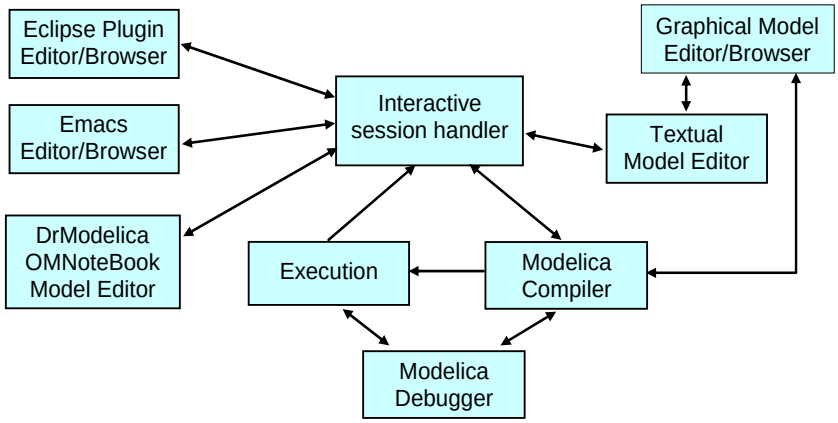
- Browsing of packages, classes, functions
- Automatic building of executables; separate compilation
- Syntax highlighting
- Code completion,
  Code query support for developers
- Automatic Indentation
- Debugger
  (Prel. version for algorithmic subset)



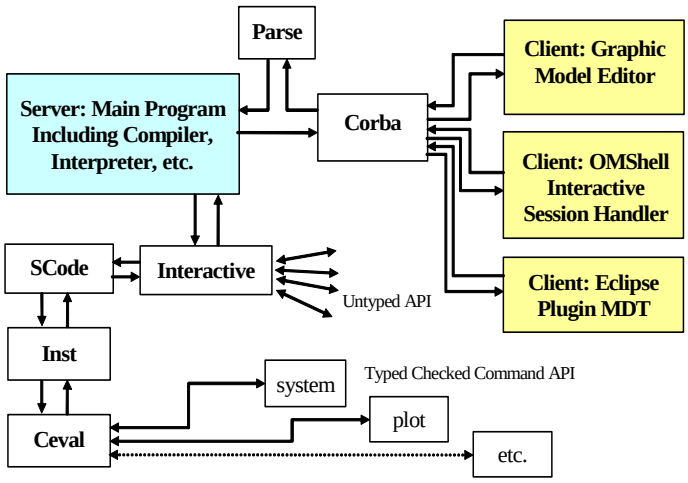
OpenModelica MDT – Usage Example



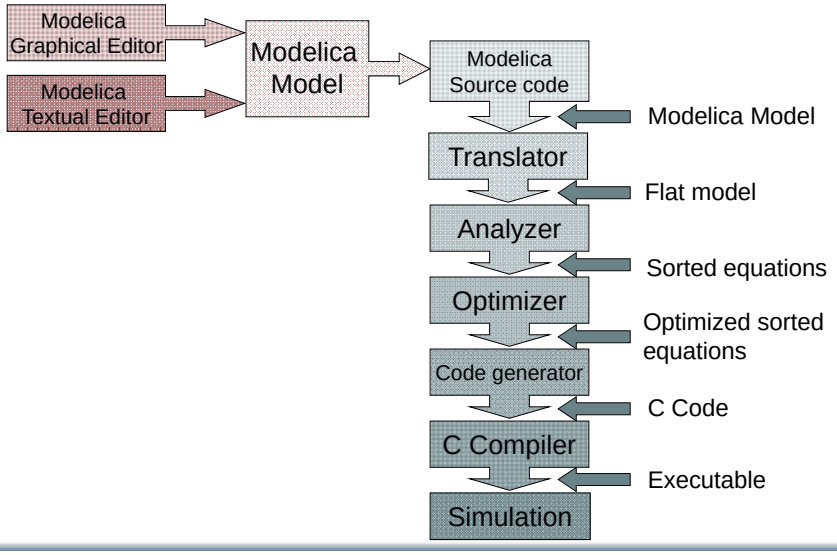
OpenModelica Environment Architecture



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

`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 (string) of inherited classes of a class.
`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. 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).

