Modelica Environments and OpenModelica

Dymola

- Dynsim (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

Founded Dec 4, 2007
Industrial members (12)
- ABB Corporate Research
- Bosch-Rexroth AG, Germany
- Siemens Turbo Machinery AB
- Creative Connections, Prague
- Eriks 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 Thermofluid Dynamics, Germany
- California State University, Dominguez Hills, USA
- Technical University of Braunschweig, the Institute of Thermodynamics, Germany
- Université Laval, the modellEAU 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.Electrical.Analog.Basic.Ground g;
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:

  \[
  \text{>> runScript("sim\_BouncingBall.mos")}
  \]

  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

Code Assistance on function calling.

OpenModelica Environment Architecture

- Eclipse Plugin Editor/Browser
- Emacs Editor/Browser
- DrModelica OMNoteBook Model Editor
- Interactive session handler
- Execution
- Modelica Compiler
- Modelica Debugger
- Graphical Model Editor/Browser
- Textual Model Editor
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>saveModel(A1&lt;strings&gt;,A2&lt;cref&gt;)</td>
<td>Saves the model (A2) in a file given by a string (A1). This call is also in typed API.</td>
</tr>
<tr>
<td>loadFile(A1&lt;cref&gt;)</td>
<td>Loads all models in the file. Also in typed API. Returns list of names of top level classes in the loaded file.</td>
</tr>
<tr>
<td>loadModel(A1&lt;cref&gt;)</td>
<td>Loads the model (A1) by looking up the correct file to load in $MODELICAPATH. Loads all models in that file into the symbol table.</td>
</tr>
<tr>
<td>deleteClass(A1&lt;cref&gt;)</td>
<td>Deletes the class from the symbol table.</td>
</tr>
<tr>
<td>addComponent(A1&lt;ident&gt;,A2&lt;cref&gt;,A3&lt;cref&gt;,annotate=&lt;expr&gt;)</td>
<td>Adds a component with name (A1), type (A2), and class (A3) as arguments. Optional annotations are given with the named argument annotate.</td>
</tr>
<tr>
<td>deleteComponent(A1&lt;ident&gt;,A2&lt;cref&gt;)</td>
<td>Deletes a component (A1) within a class (A2).</td>
</tr>
<tr>
<td>updateComponent(A1&lt;ident&gt;,A2&lt;cref&gt;,A3&lt;cref&gt;,annotate=&lt;expr&gt;)</td>
<td>Updates an already existing component with name (A1), type (A2), and class (A3) as arguments. Optional annotations are given with the named argument annotate.</td>
</tr>
<tr>
<td>addClassAnnotation(A1&lt;cref&gt;,annotate=&lt;expr&gt;)</td>
<td>Adds annotation given by $\text{annotate=} \text{classmod(...)}$ to the model definition referenced by $\text{A1}$. Should be used to add Icon Diagram and Documentation annotations.</td>
</tr>
</tbody>
</table>
| getComponents(A1<cref>) | Returns a list of the component declarations within class A1: 
\[
\{\text{Atype}, \text{varidA}, \text{commentA}\}, \{\text{Btype}, \text{varidB}, \text{commentB}\}, \ldots
\]
| 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) where the first 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 n-th component (A2) where the first 1) of class/component A1. |
| getInheritanceCount(A1<cref>) | Returns the number (as a string) of inherited classes of a class. |
| getNthInheritedClass(A1<cref>,A2<int>) | 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).