

Mathematical Modeling with Modelica

Teaching with Active Electronic Notebooks

Presentation in Berlin
2009-04-02

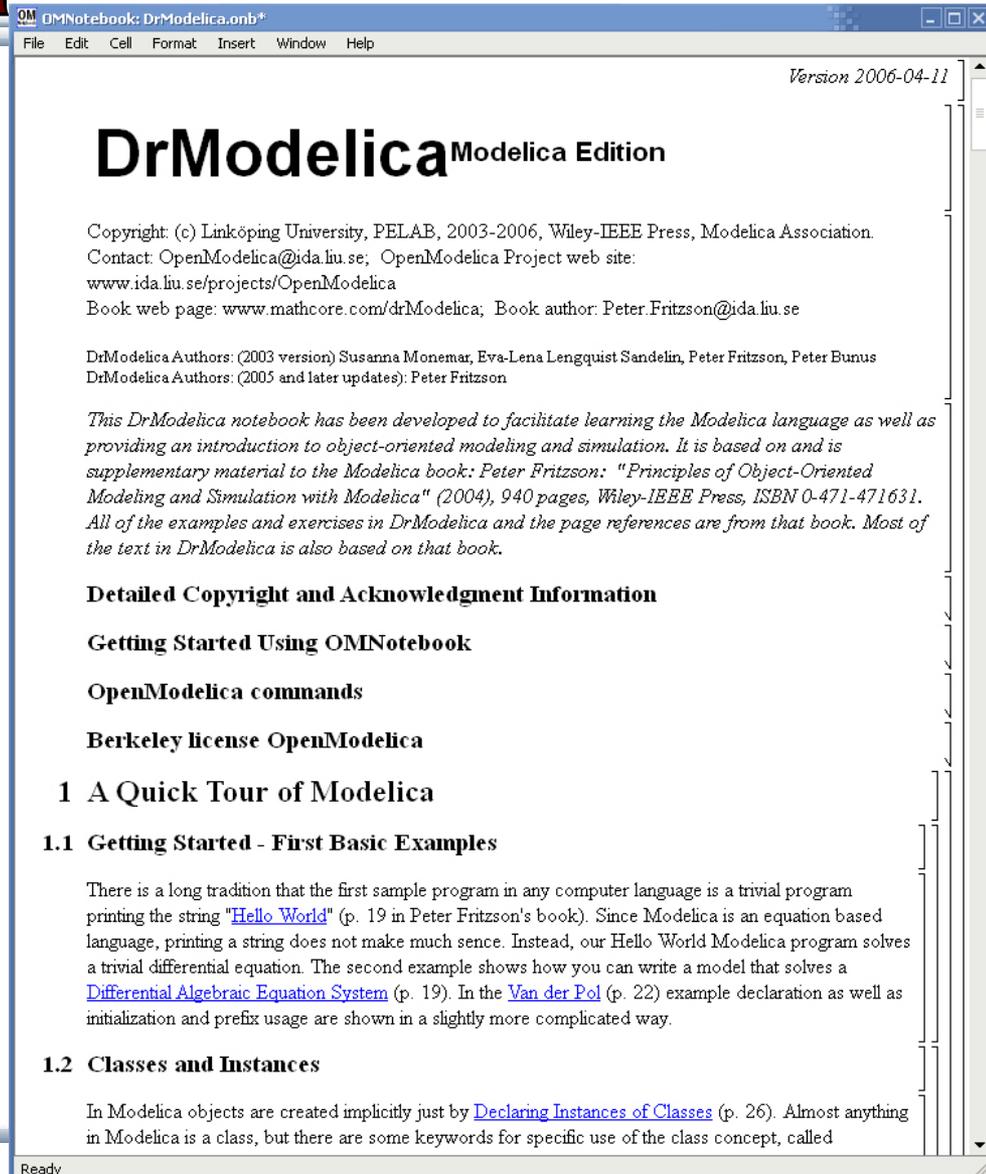
Bernhard Bachmann, Peter Fritzson,

Linköping University, Sweden

www.openmodelica.org

Teaching with Active Electronic Notebook in OpenModelica

- Primarily for teaching
- Interactive electronic book
- Platform independent (Windows, Linux, Mac)



OMNotebook: DrModelica.onb*
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 Project web site:
www.ida.liu.se/projects/OpenModelica
Book web page: www.mathcore.com/drModelica; Book author: Peter.Fritzson@ida.liu.se

DrModelica Authors: (2003 version) Susanna Monemar, Eva-Lena Lengquist Sandelin, Peter Fritzson, Peter Bunus
DrModelica Authors: (2005 and later updates): Peter Fritzson

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 Fritzson: "Principles of Object-Oriented Modeling and Simulation with Modelica" (2004), 940 pages, Wiley-IEEE Press, ISBN 0-471-471631. 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 Fritzson'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

Ready

Cells with both Text and Graphics

OMNotebook: Exercise1.nb

File Edit Cell Format Insert Window Help

Exercise 1

Using Algorithm Sections

Write a function, `Sum`, which calculates the sum of numbers, in an array of arbitrary size.

Write a function, `Average`, which calculates the average of numbers, in an array of arbitrary size. should use make a function call to `Sum`.

Write a class, `LargestAverage`, that has two arrays and calculates the average of each of the compares the averages and sets a variable to true if the first array is larger than the second and other

Answer

Ready

OMNotebook: HelloWorld.onb*

File Edit Cell Format Insert Window Help

First Basic Class

1 HelloWorld

The program contains a declaration of a class called `HelloWorld` with two fields and one equation. The first field is the variable `x` which is initialized to a start value 2 at the time when the simulation starts. The second field is the variable `a`, which is a constant that is initialized to 2 at the beginning of the simulation. Such a constant is prefixed by the keyword `parameter` in order to indicate that it is constant during simulation but is a model parameter that can be changed between simulations.

The Modelica program solves a trivial differential equation: $x' = -a \cdot x$. The variable `x` is a state variable that can change value over time. The x' is the time derivative of `x`.

```
class HelloWorld
  Real x(start = 1);
  parameter Real a = 1;
equation
  der(x) = - a * x;
end HelloWorld;
```

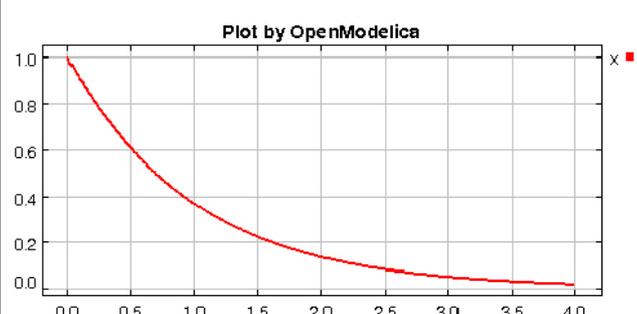
Ok

2 Simulation of HelloWorld

```
simulate( HelloWorld, startTime=0, stopTime=4 );
```

[done]

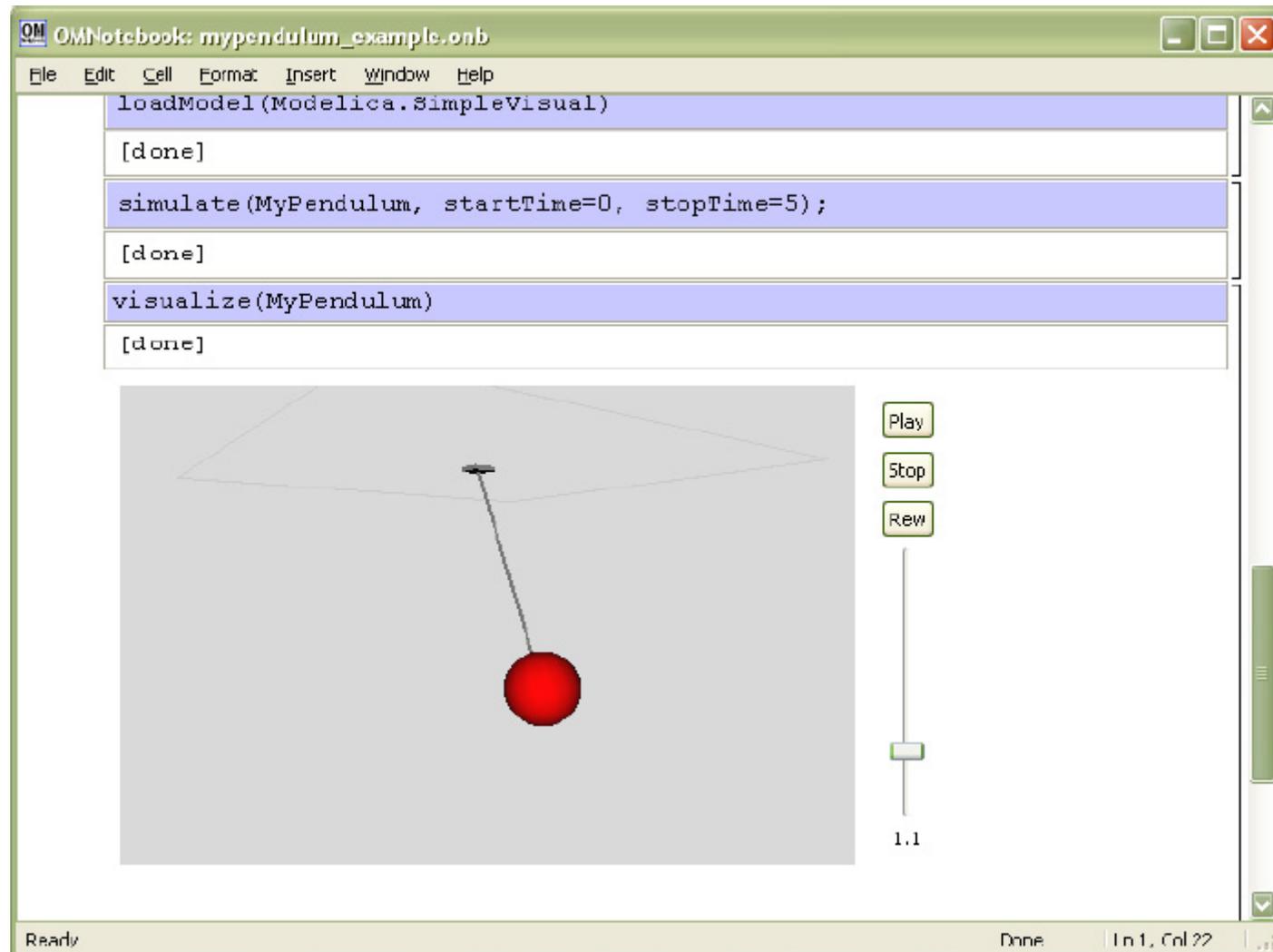
```
plot( x );
```



Plot by OpenModelica

Ready

Pendulum 3D Example – Animation



The screenshot displays the OMNotebook interface for a 3D pendulum simulation. The window title is "OMNotebook: mypendulum_example.onb". The menu bar includes File, Edit, Cell, Format, Insert, Window, and Help. The main area contains a code editor with the following commands:

```
loadModel(Modelica.SimpleVisual)  
[done]  
simulate(MyPendulum, startTime=0, stopTime=5);  
[done]  
visualize(MyPendulum)  
[done]
```

Below the code editor is a 3D visualization of a pendulum with a red spherical bob. To the right of the visualization are control buttons: Play, Stop, and Rew. Below these buttons is a vertical slider with a value of 1.1. The status bar at the bottom shows "Ready" on the left, "Done" in the center, and "Ln 1, Col 22" on the right.

OMNotebook – Interactive WYSIWYG Book Software for Teaching Programming



Anders Fernström, Ingemar Axelsson, Peter Fritzson, Anders Sandholm, Adrian Pop

Common Teaching

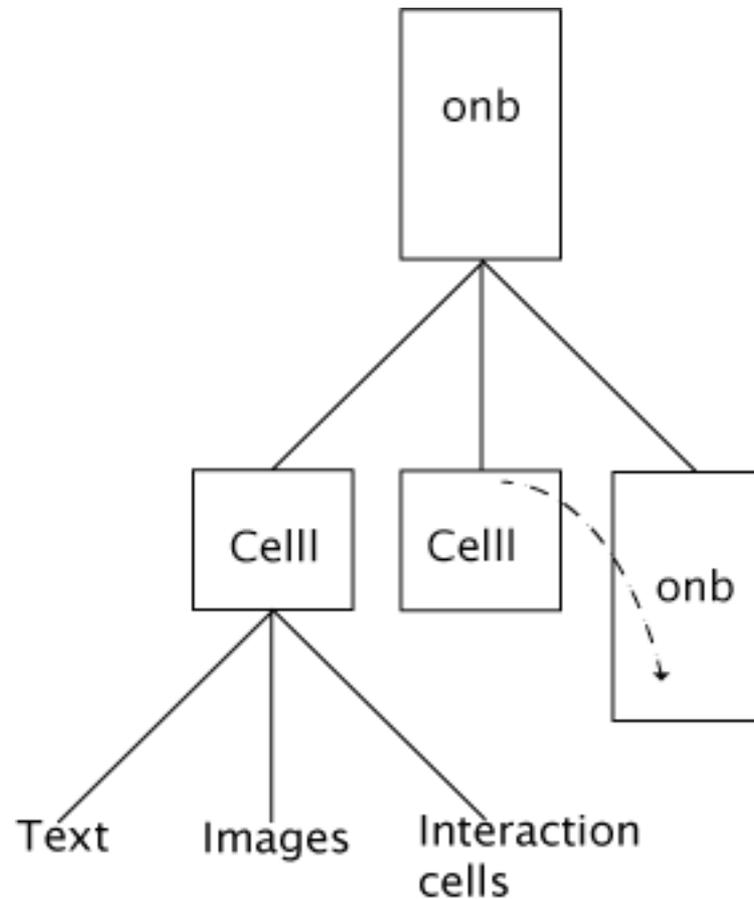
- Lectures
- Programming exercises
 - used to grasp the concept.

Traditional teaching methods too passive

Interactive Notebooks using Literate Programming

- Literate Programming
 - Source code are integrated with documentation in the same document
 - Mathematic notebooks
- Suitable for
 - teaching
 - experimentation
 - simulation scripting
 - model documentation
 - storage

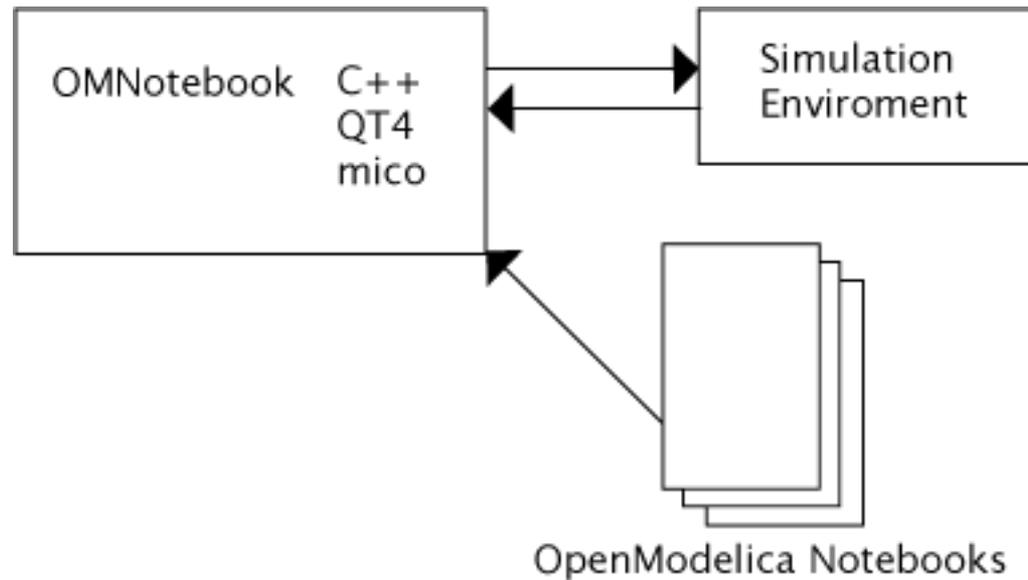
Tree Structured Hierarchical Document Representation



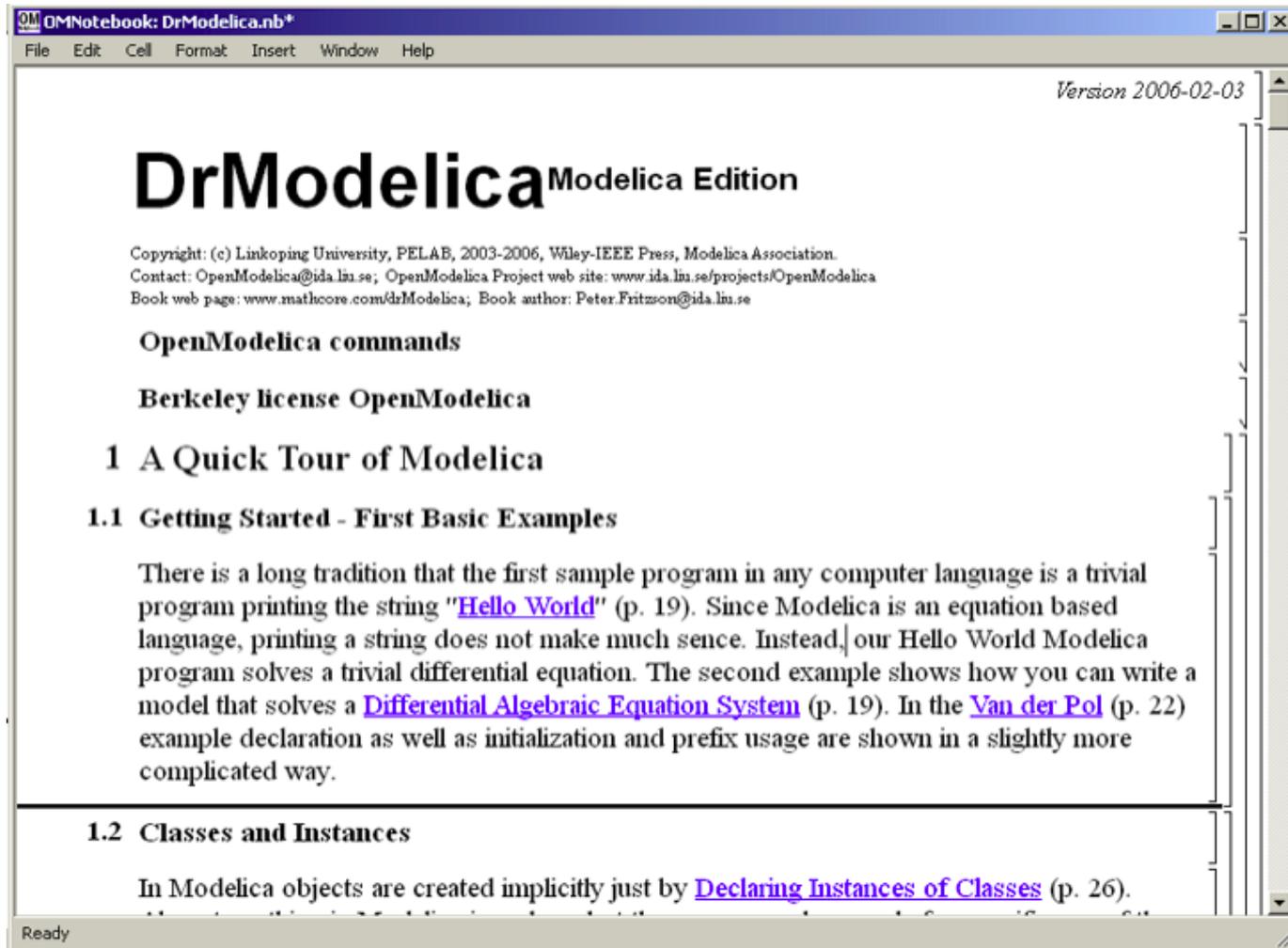
OMNotebook

- User can add, change and remove anything in a notebook.
- Save and reopen documents.
- User can create an entirely new notebook with his/her own program
- Syntax highlighting
- Keyword
- Hyperlinks
- Command completion

Connecting Notebooks with OpenModelica



DrModelica



DrModelica

- Textbook example exist in the notebook, make it easier to learn a complex equation language.
- One of the first open source software to create interactive WYSIWYG books for teaching and learning programming.

DrModelica

DEMO



Student Evaluation of DrModelica

- DrModelica was evaluated by approx. 10 students in a Modelica Course fall 2004

Usability Evaluation – Ten Usability Heuristics

- **1. Visibility of system status:**
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- **2. Match between system and the real world:**
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- **3. User control and freedom:**
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
- **4. Consistency and standards:**
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- **5. Error prevention:**
Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

Usability – Ten Usability Heuristics

- **6. Recognition rather than recall:**
Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- **7. Flexibility and efficiency of use:**
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- **8. Aesthetic and minimalist design:**
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- **9. Help users recognize, diagnose, and recover from errors:**
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- **10. Help and documentation:**
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Evaluation, Page 1 of 2

(Done 2004, on an early version of DrModelica)

- The evaluation gave many valuable results. The evaluators found that learning how to use DrModelica was easy in general. However, realizing how some of the functionality works was, according to the evaluators, not so intuitive. For example it can be hard to discover the ability to collapse and expand sections. Though, once it was known how to use the functionality they found easy.
- Furthermore, according to the evaluators it might be confusing that a link in some cases opens a new window and in other cases refers to another chapter in the same window. This is a problem concerning heuristic number 4.
- Another problem, when being linked to another page, is that there is no feedback telling the user that a new page has appeared in front of the previous one. This is a problem mostly concerning heuristics number 1, 2 and 3.

Evaluation Page 2 of 2

(Done 2004, on an early version)

- When a new window is opened in front of the other the user is not properly informed about what is going on, since there is no feedback that the window was just being opened (see heuristic number 1).
- This involves another problem, taking the user back to the former window. This is currently resolved by closing the window, but it would be better solved by having a “back”-button, following real-world conventions (see heuristics 2 and 3).
- Heuristics number 5, 8 and 9 concern dialogues and error messages, none of which exist in neither DrModelica nor OpenModelica, but that is why the environment does not have a need for it. Heuristic number 10 concerns help and documentation. There is a help section on how to start using DrModelica, which was appreciated by the users.
- The evaluators also found that DrModelica was less intimidating than other programming environments, since the user is presented with an environment similar to a document showing only a small amount of functionality. This leads the user to believe that DrModelica is a reading material. However, after using the material for a while the user discovers that DrModelica could be used for programming as well.
- A common approach adopted by many programming environments is to lead the user in the opposite direction, by presenting all functionality from the beginning. This approach can have a discouraging effect on the user

Future Improvements

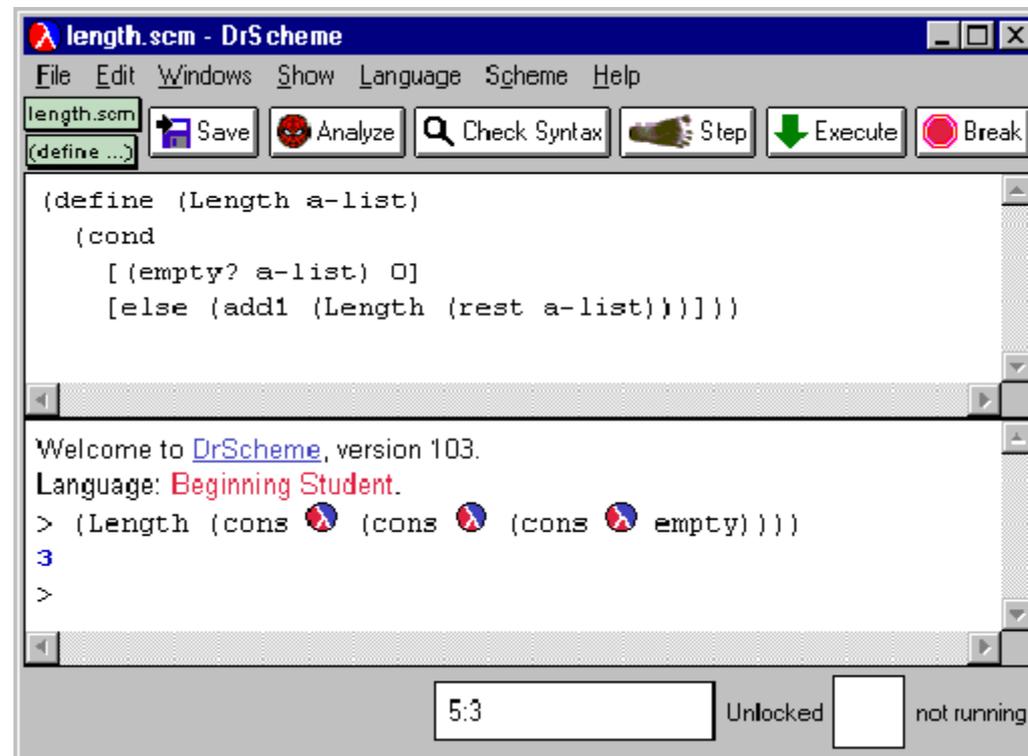
(Comments on 2004 version)

- Considering the results of the evaluation and comparing our work with related work we have discovered some possible improvements that can be implemented in the future. Here follows a list of these improvements:
- A suggestion from the students, attending the Modelica graduate course, is to extend DrModelica to contain more exercises on simple as well as more complex constructs in order for the student to get more practice.
- Since it can be difficult to learn how to use the functionality in DrModelica, an idea is to make an introductory exercise for practicing the basics step by step instead of just reading a long introductory text.
- Links between files containing different variants of the same term should be added. Currently the exercises in the material mainly concern language specific constructs, it would be desirable to add exercises reflecting the purpose of Modelica. The material needs to be extended with more exercises in general.
- Features, like parenthesis matching and keyword highlighting, used in DrScheme and DrJava, would be helpful when programming.

Summary of DrModelica

- DrModelica is a material intended to facilitate the learning of Modelica
- DrModelica is based on Literate programming
- The evaluation shows that Literate programming is a suitable methodology for teaching Modelica
- Hopefully DrModelica will increase the usage of Modelica, and facilitate learning modeling and simulation

Related Work – DrScheme



The screenshot shows the DrScheme IDE window titled "length.scm - DrScheme". The menu bar includes File, Edit, Windows, Show, Language, Scheme, and Help. The toolbar contains buttons for Save, Analyze, Check Syntax, Step, Execute, and Break. The main text area contains the following Scheme code:

```
(define (Length a-list)
  (cond
    [(empty? a-list) 0]
    [else (add1 (Length (rest a-list)))]))
```

Below the code, the REPL shows the following interaction:

```
Welcome to DrScheme, version 103.
Language: Beginning Student.
> (Length (cons (cons (cons empty)))
3
>
```

The status bar at the bottom indicates the cursor is at line 5, column 3, the window is Unlocked, and the program is not running.

Related Work – DrJava

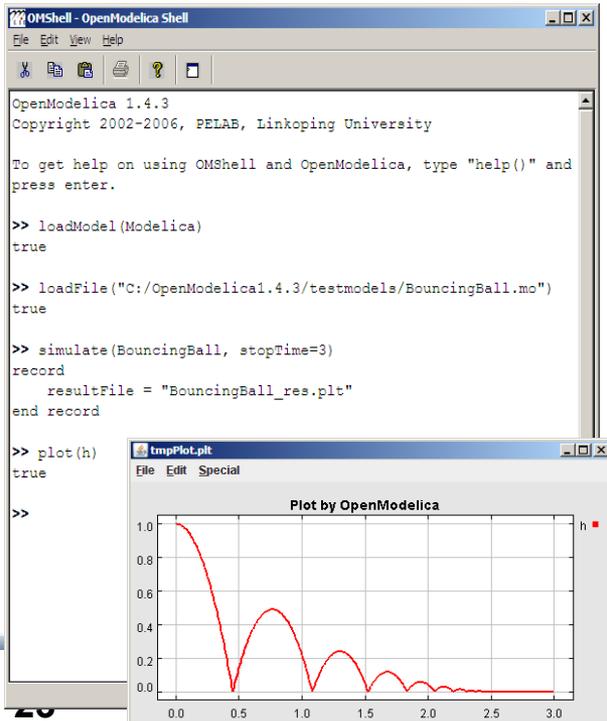
The screenshot displays the DrJava IDE interface. The main editor window shows the source code for a class named `List`. The code includes a constructor `Cons(int f, List r)`, a `getLength()` method, and a `toStringHelp()` method. A breakpoint is set at line 80, which is the `return` statement in the `getLength()` method. The console window at the bottom shows the following output:

```
Welcome to DrJava.
> List myList = new Cons(7, Empty.ONLY);
> myList.getLength()
Breakpoint hit in class Cons [line 80]
```

The stack window shows the current stack frame for `Cons.getLength` at line 80, along with the frames for `sun.reflect.NativeMethodAccessorImpl.invoke0`, `sun.reflect.NativeMethodAccessorImpl.invoke`, and `sun.reflect.DelegatingMethodAccessorImpl.invoke`.

What is OpenModelica? www.openmodelica.org

- Advanced Interactive Modelica compiler (OMC)
 - Supports most of the Modelica Language
- Basic environment for creating models
 - OMShell – an interactive command handler * ModelicaML UML Profile
 - OMNotebook – a literate programming notebook
 - MDT – an advanced textual environment in Eclipse



OMShell - OpenModelica Shell

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

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

>> loadModel(Modelica)
true

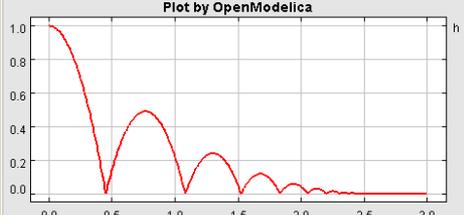
>> loadFile("C:/OpenModelica1.4.3/testmodels/BouncingBall.mo")
true

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

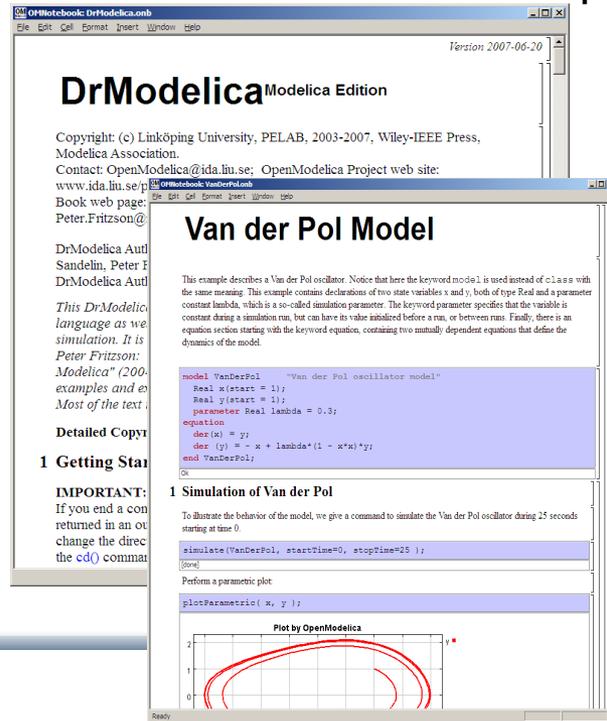
>> plot(h)
true
```

tmpPlot.plt

Plot by OpenModelica



The plot shows a red curve representing the height h of a bouncing ball over time. The x-axis ranges from 0.0 to 3.0, and the y-axis ranges from 0.0 to 1.0. The curve starts at (0, 1.0), reaches a minimum of 0.0 at approximately t=0.5, and then oscillates with decreasing amplitude, crossing the x-axis at approximately t=1.0, 1.5, and 2.0.



DrModelica Modelica Edition

Version 2007-06-20

Copyright: (c) Linköping University, PELAB, 2003-2007, Wiley-IEEE Press, Modelica Association.
Contact: OpenModelica@ida.liu.se; OpenModelica Project web site: www.ida.liu.se/projects/OpenModelica
Book web page: www.ida.liu.se/projects/OpenModelica
Peter.Fritzon@ida.liu.se

DrModelica AutL Sandelin, Peter Fritzon
DrModelica AutL

This DrModelica language as we simulation. It is Peter Fritzon: Modelica" (2006) examples and exercises. Most of the text.

Detailed Copy

1 Getting Started

IMPORTANT: If you end a command returned in an output window, change the direction of the `cd()` command.

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 x(start = 1);
  Real y(start = 1);
  parameter Real lambda = 0.3;
equation
  der(x) = y;
  der(y) = -x + lambda*(1 - x*x)*y;
end VanDerPol;
```

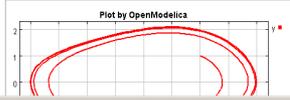
1 Simulation of Van der Pol

To illustrate the behavior of the model, we give a command to simulate the Van der Pol oscillator during 25 seconds starting at time 0.

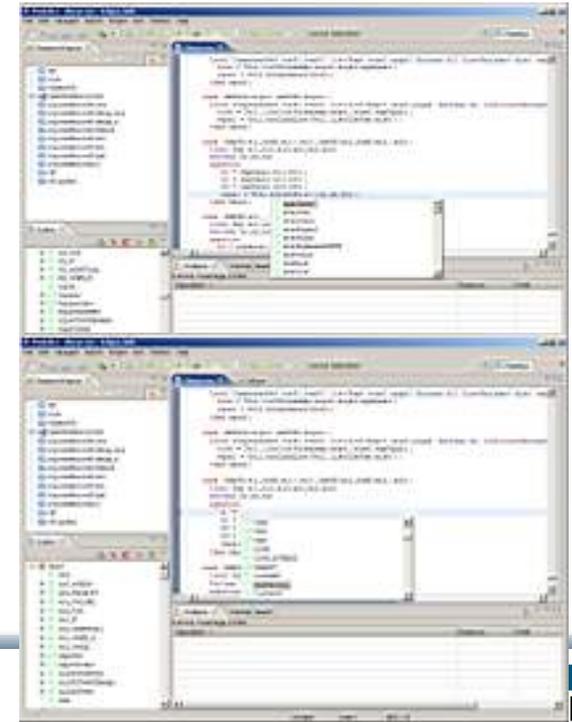
```
simulate(VanDerPol, startTime=0, stopTime=25);
```

Perform a parametric plot

```
plotParametric(x, y);
```



The plot shows a red curve representing the state variables x and y over time. The x-axis ranges from 0.0 to 3.0, and the y-axis ranges from 0.0 to 2.0. The curve starts at (0, 1.0) and oscillates with a period of approximately 1.5 seconds, with the amplitude of the oscillations decreasing over time.



The screenshot shows the Eclipse IDE with a Modelica model being edited. The editor displays the model code, and the IDE interface includes a project explorer, a console, and a plot window. The plot window shows a red curve representing the state variables x and y over time, similar to the plot in the previous screenshot.

What is OpenModelica? (II)

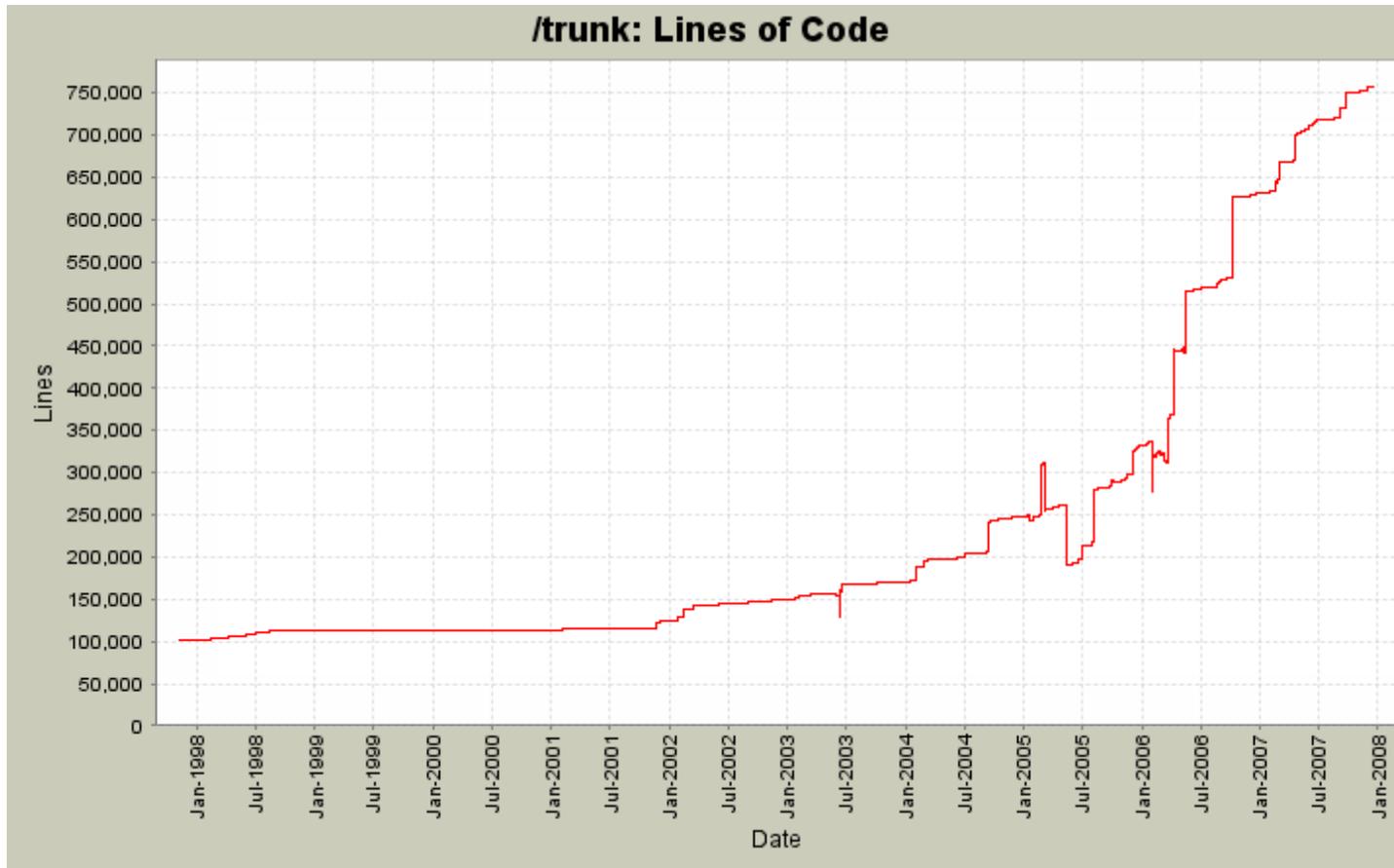
- Open-source community services
 - Website and Support Forum
 - Version-controlled source base
 - Bug database
 - Development courses

The screenshot shows the OpenModelica Project website. The header includes the logo for PELAB (Programming Environment Laboratory) and the Department of Computer and Information Science (IDA) at Linköping University (LIU). Navigation tabs include Home, Goal, Research, Documentation, Download OMC/MDT/Nightly-Builds, Bugs, License, Registration, Forum, Contact, and Developer Pages. The main content area is divided into sections: Goal, News, and Tutorial. The Goal section states the project's aim to create a complete Modelica modeling, compilation, and simulation environment. The News section lists releases from 2007-07-13 to 2005-05-02. The Tutorial section offers an introduction to object-oriented modeling and simulation with OpenModelica, accompanied by a book cover and a diagram.

The screenshot displays a 'Log Messages' window for the OpenModelica project. It shows a table of revisions with columns for Revision, Actions, Author, Date, and Message. The messages describe updates to the OMCShell, fixes to the runtime, and changes to the source code. Below the table, there are sections for 'Action' and 'Path' showing file operations like 'Modified', 'Deleted', and 'Added'. The window also includes a search bar and a 'Show All' button.

The screenshot shows a 'Bugzilla - Bug List' window. It features a search bar and navigation links for Home, View, Search, and Reports. The main content area displays a table of bugs with columns for ID, Opened, Sev, Pri, Assignee, Reporter, Status, Resol, OS, and Summary. The table lists 188 bugs found, with the first few entries showing details like 'Faulty errormsg for assignment statement in an equation section' and 'Interaktiv anrop av en funktion med Strangparameter'.

OpenModelica Statistics



- Mature code base
- ~ 750k lines of code, doubled since 2005

Open Source Modelica Consortium

- Created in Dec 4, 2007
 - Non-profit, non-governmental organization
 - Purpose
 - Developing and promoting the development and usage of the OpenModelica open source implementation and environment for model-driven applications
 - Members Dec 2008
 - 11 industry/institute members, 8 university members, 22 individual members
-
- **Bosch-Rexroth AG, Germany**
 - **ABB Corporate Research AB, Sweden**
 - **Siemens Industrial Turbomachinery, Sweden**
 - **Equa Simulation AB, Sweden**
 - **TLK Thermo, Germany**
 - **VTT, Finland**
 - **MostforWater, Belgium**
 - **MapleSoft, Canada,**
 - **Emmeskay Inc., USA**
 - **IFP, Paris, France**
 - **MathCore Engineering AB**
- **Linköping University, Sweden**
 - **Technical Univ of Hamburg-Harburg, Germany**
 - **Technical Univ of Braunschweig, Germany**
 - **Université Laval, Canada**
 - **University of Queensland, Australia**
 - **Griffith University, Australia**
 - **Politecnico di Milano, Italy**
 - **Mälardalen University, Sweden**

Conclusions

- New OpenModelica version 1.4.5 contains flexible 2D graphics and 3D animation

- www.openmodelica.org Download OpenModelica
- Emails: petfr@ida.liu.se, OpenModelica@ida.liu.se