

OpenModelica Development Environment with Eclipse Integration

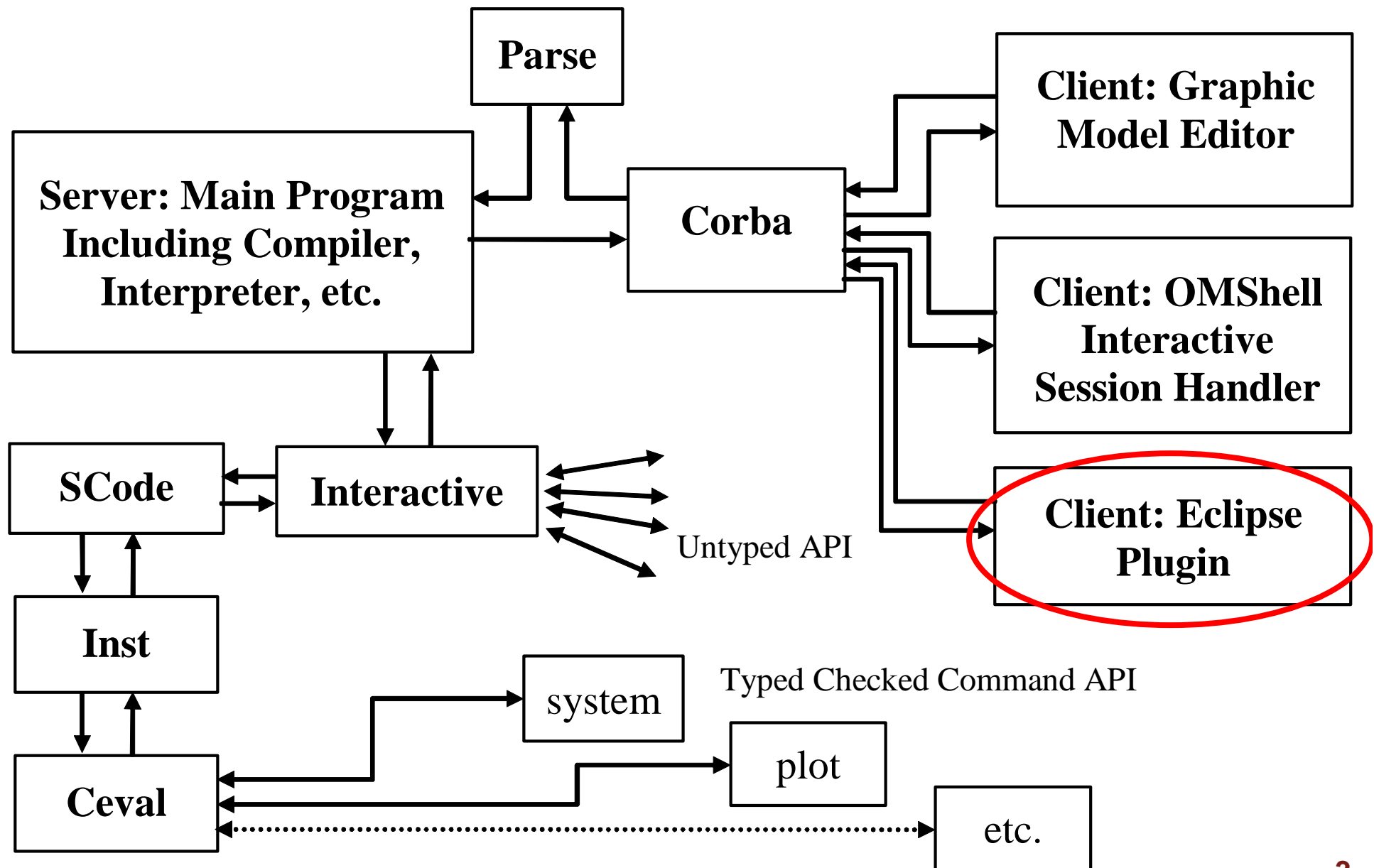
**Adrian Pop, Peter Fritzson, Andreas
Remar, Elmir Jagudin, David Akhvlediani**

Programming Environment Laboratory
Department of Computer and Information Science
Linköping University
2006-09-05

Modelica'2006, September 4-5,
Vienna, Austria

- Introduction
 - OpenModelica
- Eclipse Environment for Modelica/MetaModelica
 - Overview
 - Examples
- Conclusions and Future Work
- Demo

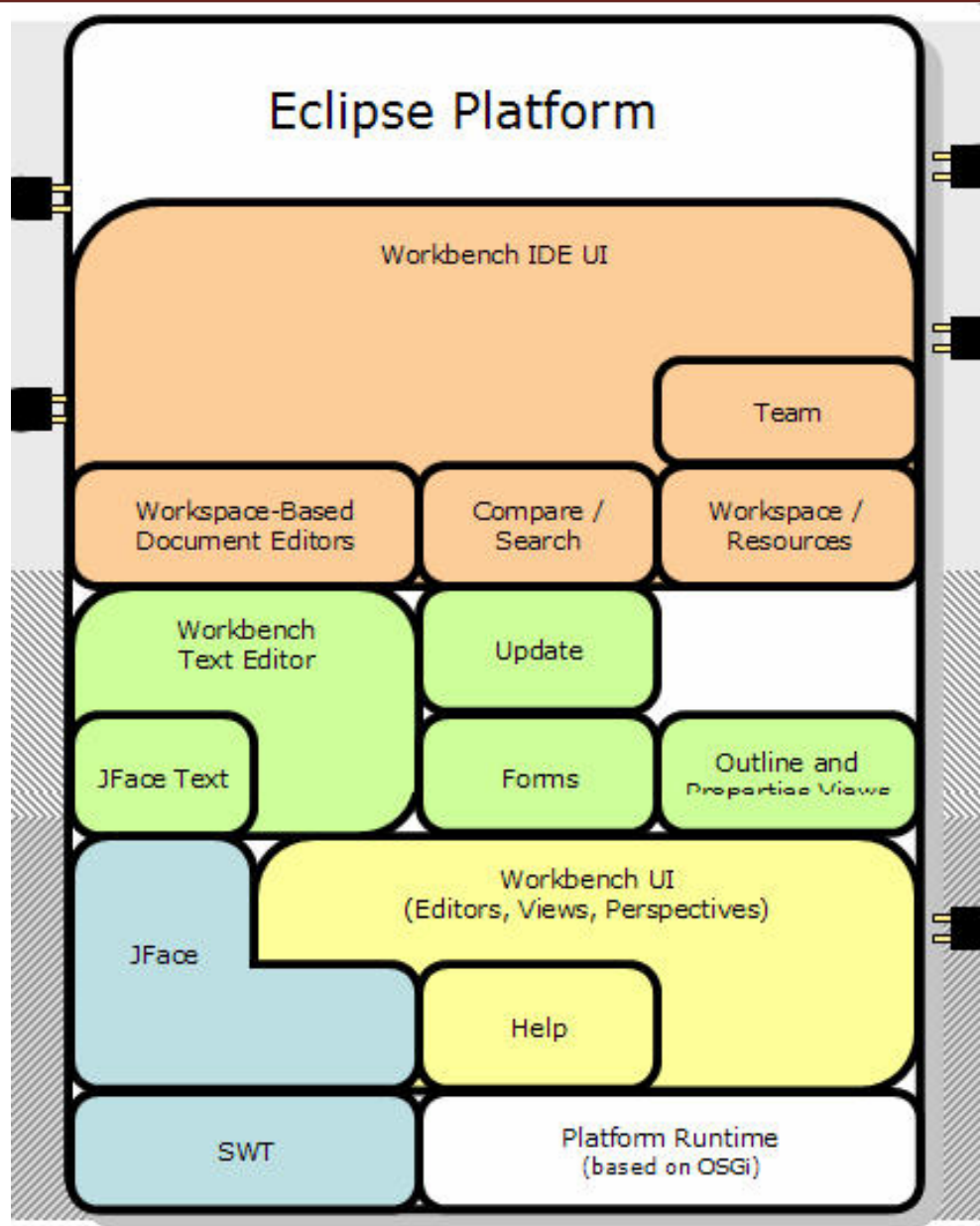
OpenModelica Context



Modelica Development Tooling (MDT)

- Supports textual editing of Modelica/MetaModelica code
- Was created to ease the development of the OpenModelica development (114232 lines of code) and to support advanced Modelica library development
- It has most of the functionality expected from a Development Environment
 - code browsing
 - code assistance
 - code indentation
 - code highlighting
 - error detection
 - automated build of Modelica/MetaModelica projects
 - debugging

The MDT Eclipse Environment



Modelica Browser

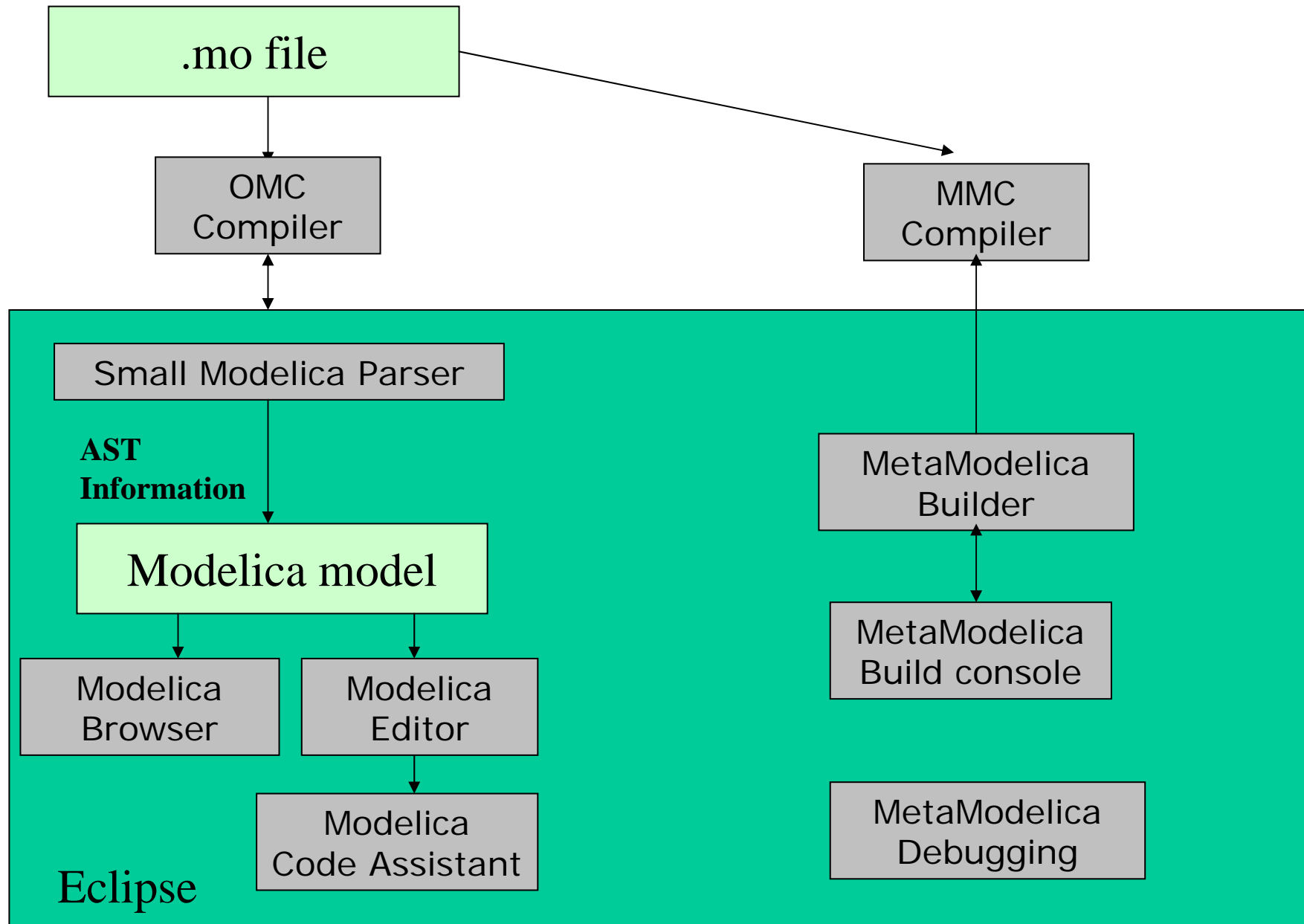
Modelica Editor

Modelica Code Assistant

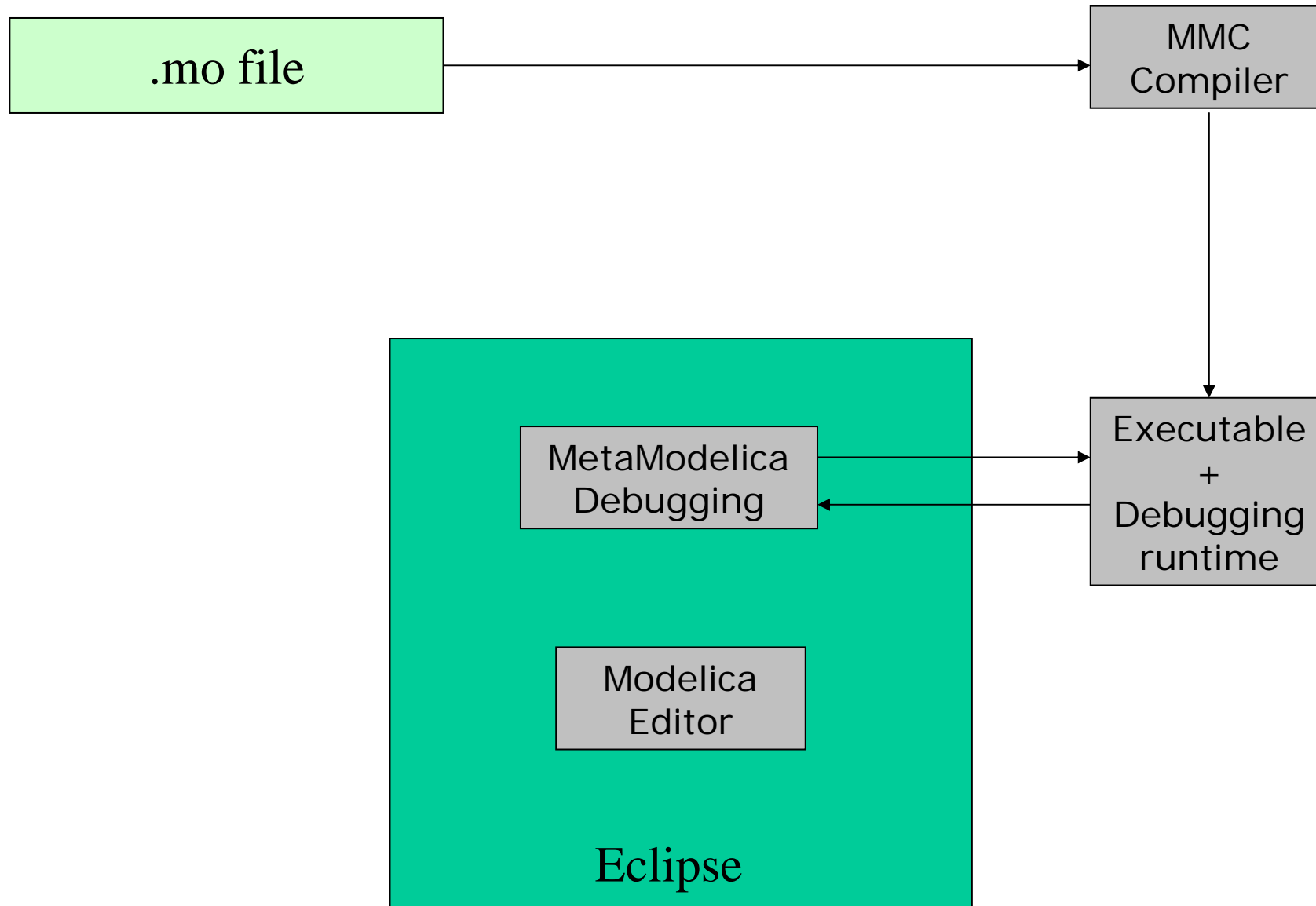
MetaModelica Debugging

Modelica Perspective

The MDT Eclipse Environment (II)



The MDT Eclipse Environment (III)



Creating Modelica projects (I)

The screenshot illustrates the steps to create a Modelica project in Eclipse. On the left, the Eclipse menu bar is visible with 'File' > 'New' > 'Project...' selected. A 'New Project' dialog box is open, showing a tree of wizards. The 'Modelica' folder is expanded, and the 'Modelica Project' wizard is selected. A red arrow points from the 'Modelica Project' wizard in the dialog to the 'New Modelica Project' wizard in the background. The background wizard shows the 'Project name' field filled with 'demo' and navigation buttons '< Back', 'Next >', 'Finish', and 'Cancel'.

Modelica - Eclipse SDK

File Edit Refactor Navigate Search Project Run Window Help

New Alt+Shift+N Project...

Open File...

Close Ctrl+F4

Close All Ctrl+Shift+F4

Save Ctrl+S

Save As...

Save All Ctrl+Shift+S

Revert

Move...

Rename... F2

Refresh F5

Convert Line Delimiters To

Print... Ctrl+P

Switch Workspace...

Import

Modelica Package

Modelica Class

Folder

File

Example...

Other...

New Project

Select a wizard

Create a new Modelica project.

Wizards:

- Plug-in Project
- C
- C++
- CVS
- Eclipse Modeling Framework
- EJB
- Functional Programming
- J2EE
- Java
- Modelica
 - Modelica Project
- Plug-in Development
- Simple
- Web
- Examples

Create a Modelica project

Create a Modelica project in the workspace.

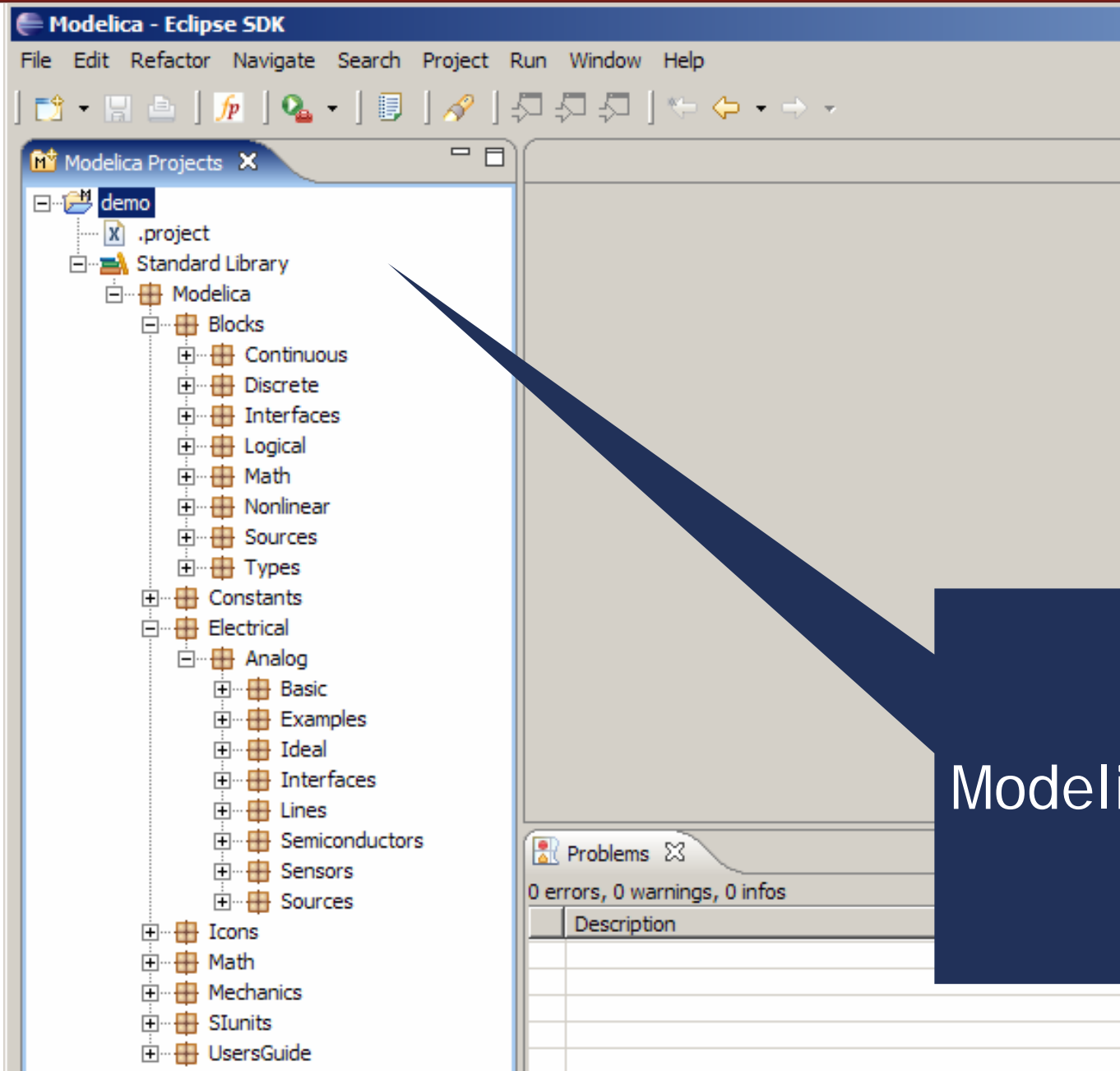
Project name: demo

< Back Next >

< Back Next > Finish Cancel

Creation of Modelica projects using wizards

Creating Modelica projects (II)



Modelica project

Creating Modelica packages

The image shows the Eclipse IDE interface for creating a new Modelica package. The 'New' menu is open, and the 'Modelica Package' option is selected. The 'New Modelica Package' wizard is displayed, with the following fields and options:

- Source folder: demo
- Package: (empty)
- Name: MyPackage
- Description: A Modelica Package
- is encapsulated package

The 'Finish' button is highlighted with a red arrow. A blue callout box on the left contains the text 'Creation of Modelica packages using wizards'.

Creating Modelica classes

The image shows the Eclipse IDE interface for Modelica. On the left, the 'Modelica Projects' view shows a project named 'demo' with a sub-package 'MyPackage'. A context menu is open over 'MyPackage', and the 'New Modelica Class' option is selected. A red arrow points from this menu item to the 'New Modelica Class' wizard dialog box in the foreground. The wizard has the following fields and options:

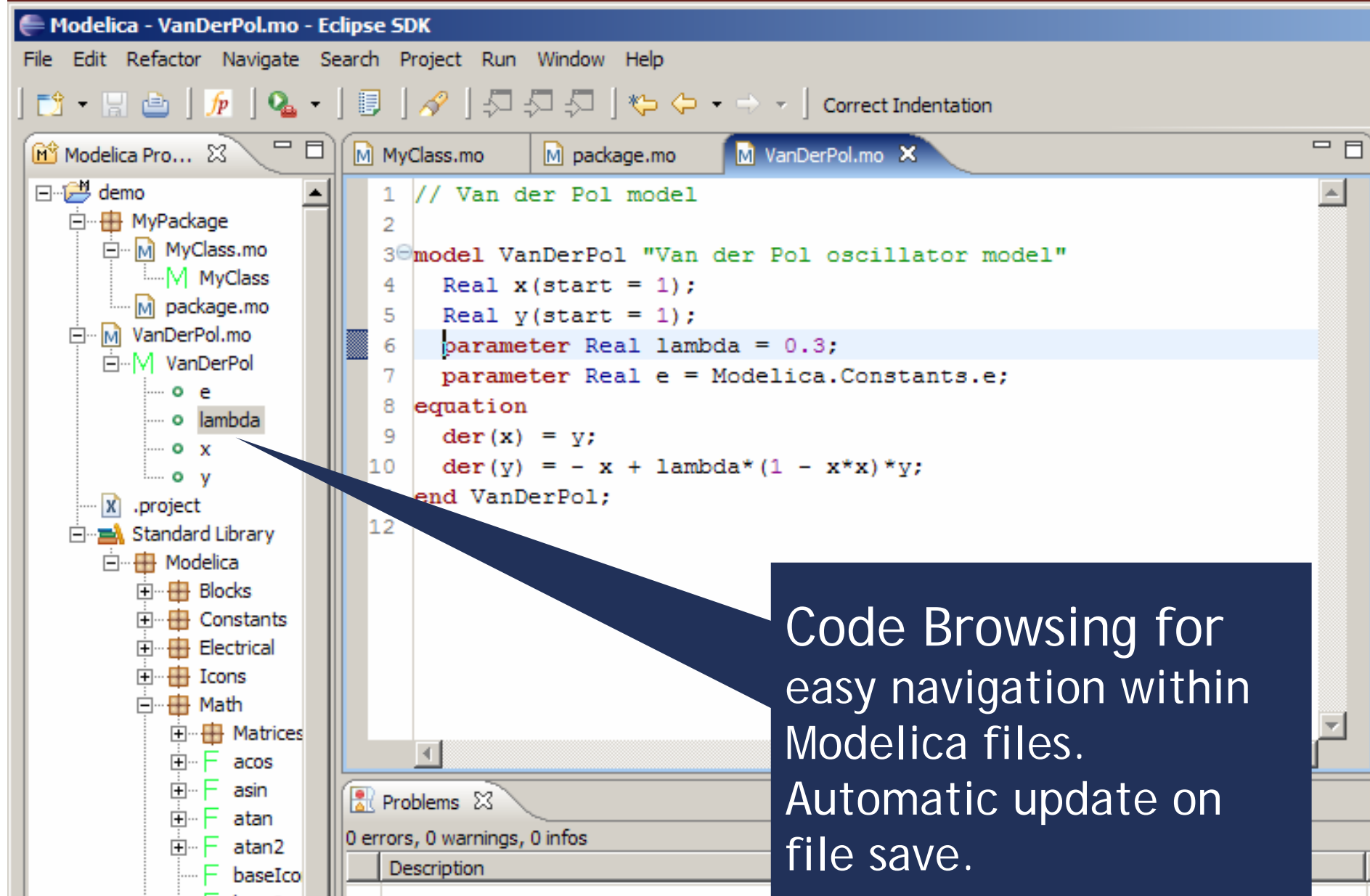
- Source folder: demo/MyPackage
- Package: MyPackage
- Name: MyClass
- Restriction: model
- Modifiers: include initial equation block, is partial class, have external body

At the bottom of the wizard are 'Finish' and 'Cancel' buttons. A red arrow points from the 'Finish' button to the code editor on the right. The code editor shows the following code in 'MyClass.mo':

```
1 within MyPackage;  
2  
3 model MyClass  
4  
5 equation  
6  
7 end MyClass;
```

Creation of Modelica classes, models, etc, using wizards

Code browsing



The screenshot displays the Eclipse IDE interface for a Modelica project. The left-hand side shows a project explorer with a tree view of the project structure. The 'demo' project contains a 'MyPackage' folder, which includes 'MyClass.mo', 'MyClass', and 'package.mo'. Below this is the 'VanDerPol.mo' file, which contains a 'VanDerPol' model. The 'VanDerPol' model has several parameters: 'e', 'lambda', 'x', and 'y'. The 'lambda' parameter is currently selected in the tree view. The main editor window shows the code for 'VanDerPol.mo'. The code is as follows:

```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4   Real x(start = 1);
5   Real y(start = 1);
6   parameter Real lambda = 0.3;
7   parameter Real e = Modelica.Constants.e;
8 equation
9   der(x) = y;
10  der(y) = - x + lambda*(1 - x*x)*y;
11 end VanDerPol;
12
```

The 'lambda' parameter is highlighted in blue in the code editor. A callout box points to the 'lambda' parameter in the tree view and contains the following text:

Code Browsing for easy navigation within Modelica files. Automatic update on file save.

Error detection (I)

The screenshot shows the Eclipse IDE interface with the following components:

- Project Explorer:** Shows a project named 'demo' containing a package 'MyPackage' with files 'MyClass.mo' and 'package.mo', and a class 'VanDerPol' with variables 'e', 'x', and 'y'. A 'Standard Library' is also visible.
- Code Editor:** Displays the content of 'VanDerPol.mo'. The code is as follows:

```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4   Real x(start = 1);
5   Real y(start = 1);
6   parameter Real lambda = 0.3;
7   parameter Real e = Modelica.Constants.e;
8 equation
9   der(x) = y;
10  der(y) = - x + lambda*(1 - x*x)*y;
11 end VanDerPol;
12
```

Line 6 is highlighted in blue, and a red 'X' icon is visible in the left margin next to it.
- Problems View:** Located at the bottom, it shows '1 error, 0 warnings, 0 infos'. The error table is as follows:

Description	Resource	In Folder	Location
unexpected token: lambda, parsing resumed at token ';' on line 6, column 29	VanDerPol.mo	demo	line 6

Parse error
detection on
file save

Error detection (II)

The screenshot shows the Eclipse IDE with the following components:

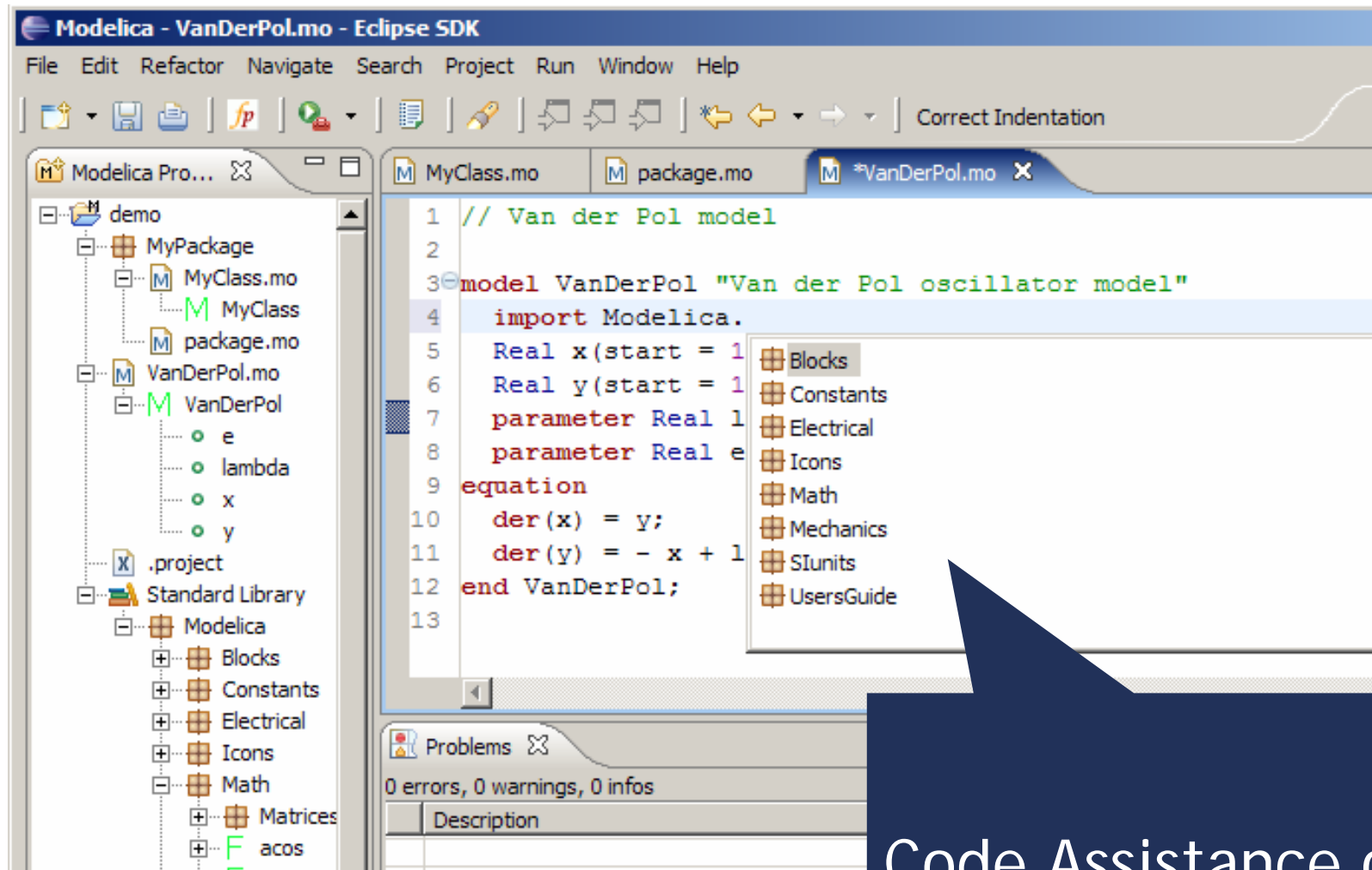
- Modelica Projects:** A tree view on the left showing a project structure with folders like 'Compiler', 'absyn_builder', 'doc', 'modpar', 'omc_debug', 'omc_release', 'report', 'rml2mmo', 'rml2sig', 'runtime', 'scripts', 'test_codegen', 'tools', 'VC7', 'winruntime', and files like 'Absyn.mo', 'Algorithm.mo', 'Builtin.mo', 'Ceval.mo', 'ClassInf.mo', 'ClassLoader.mo', 'Codegen.mo', 'Connect.mo', 'Corba.mo', 'DAE.mo', 'DAEEXT.mo', 'DAELow.mo', 'Debug.mo', and 'Derive.mo'.
- Absyn.mo:** The main editor window showing the following code:

```
69 public
70 uniontype Program "- Programs, the top level construct
71   A program is simply a list of class definitions declared at top
72   level in the source file, combined with a within statement that
73   indicates the hieractical position of the program.
74 "
75 record PROGRAM
76   list<Class> classes "classes ; List of classes" ;
77   Withi within_ "within ; Within statement" ;
78 end PROGRAM;
79
```
- Problems/Console:** The bottom panel shows the error log with the following text:

```
<terminated> OMDev-MINGW-OpenModelicaBuilder [Program] c:\OMDev\tools\msys\bin\make.exe
cp -p ../Static.mo Static.mo
cp -p ../SimCodegen.mo SimCodegen.mo
cp -p ../Values.mo Values.mo
cp -p ../System.mo System.mo
/c/OMDev//tools/rml/bin/rmlc -v -Wc,-O3 -c Absyn.mo
"/c/OMDev//tools/rml//bin/rml" -Eplain Absyn.mo
Absyn.mo:77.5-77.9 Error: unbound type constructor Withi
Error: StaticElaborationError
make[2]: Leaving directory `~/c/bin/...
make[1]: Leaving directory `~/c/bin/...
make[2]: *** [Absyn.h] Error 1
make[1]: *** [omc_release] Error 2
make: *** [omc] Error 2
```

A callout box with a blue background and white text points to the error message in the console, containing the text: "Semantic error detection on file save".

Code assistance (I)



Code Assistance on imports.

Code assistance (II)

The screenshot shows the Eclipse IDE with the Modelica SDK. The main editor displays the following code:

```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4   import Modelica.Math;
5   Real x(start = 1);
6   Real y(start = 1);
7   parameter Real lambda = 0.3;
8   parameter Real e = Modelica.Constants.
9 equation
10  der(x) = y;
11  der(y) = - x + lambda*(1 - x*x)*y;
12 end VanDerPol;
13
```

Line 8 is selected, and a code completion popup is visible, listing constants from the Modelica.Math package. The constant 'e' is highlighted. A blue arrow points from the text 'Code Assistance on assignments.' to the 'e' in the popup.

The left sidebar shows a project structure with a 'demo' package containing 'MyPackage', 'VanDerPol.mo', and 'Standard Library'. The 'Problems' window at the bottom shows '0 errors, 0 warnings, 0 infos'.

Code Assistance on assignments.

Code assistance (III)

The screenshot shows the Eclipse IDE with the following components:

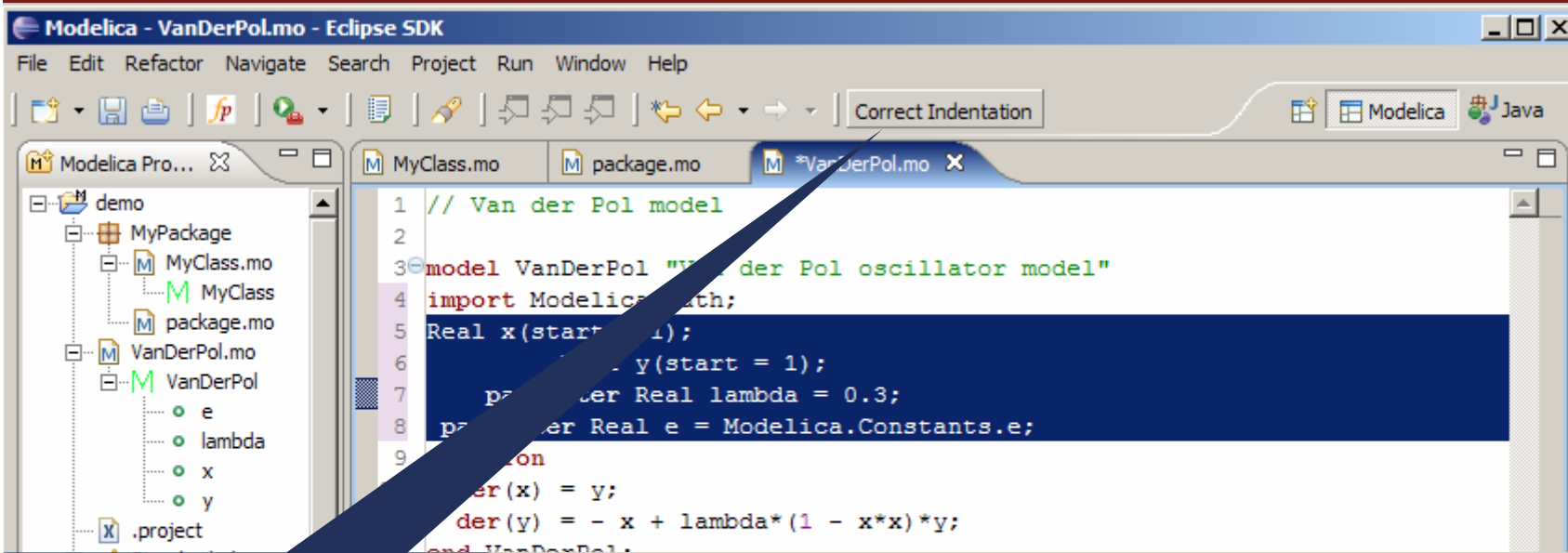
- Project Explorer:** Shows a project named 'demo' containing a package 'MyPackage' with files 'MyClass.mo', 'package.mo', and 'VanDerPol.mo'. The 'VanDerPol' class is expanded to show parameters 'e', 'lambda', 'x', and 'y'.
- Editor:** Displays the code for 'VanDerPol.mo'. The code is as follows:

```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4   import Modelica.Math;
5   Real x(start = 1);
6   Real y(start = 1);
7   parameter Real lambda = 0.3;
8   parameter Real e = Modelica.Constants.e;
9   equation
10    der(x) = y;
11    y = Modelica.Math.sin(
12    der(y) = - x + lambda*(1 - x*x)*y;
13  end VanDerPol;
14
```

Line 11 is highlighted, and a tooltip shows 'Real sin(SI.Angle u)'. A blue arrow points from this tooltip to a dark blue callout box at the bottom right.
- Problems View:** Shows '0 errors, 0 warnings, 0 infos'.
- Table:** A table with columns 'Description', 'Resource', 'In Folder', and 'Location' is partially visible at the bottom.

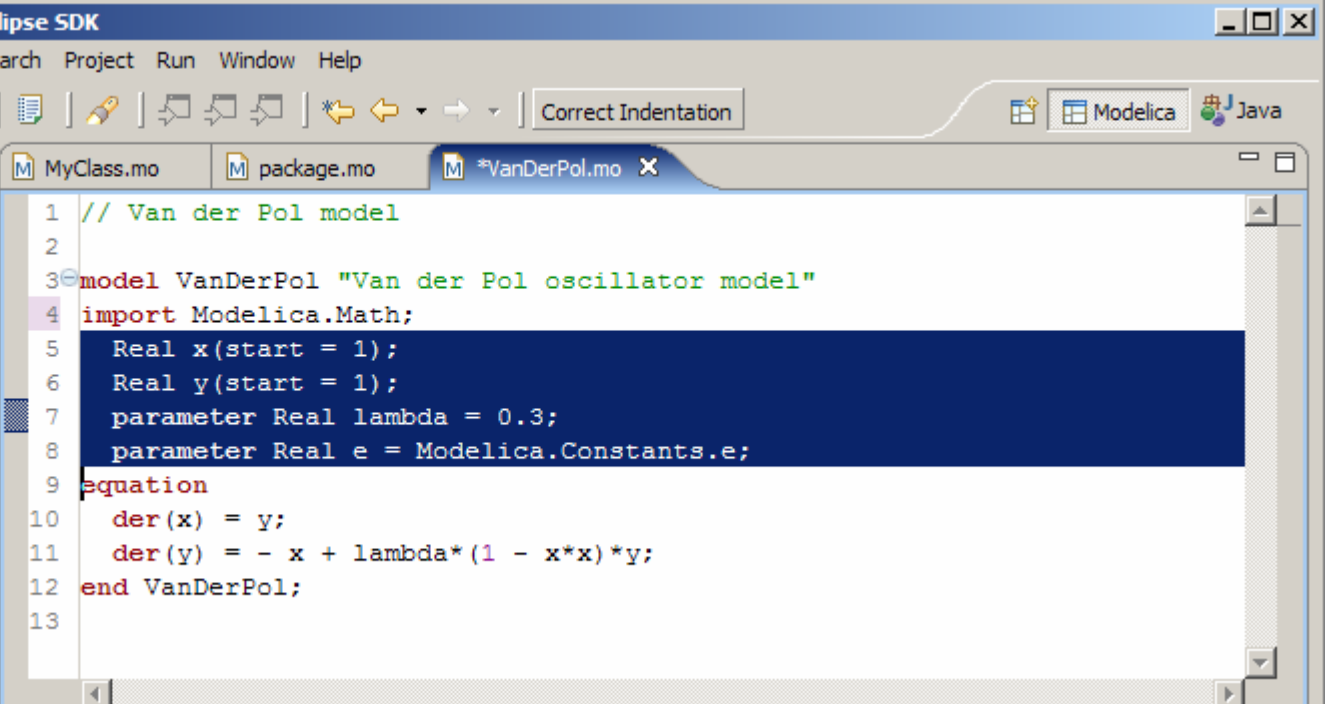
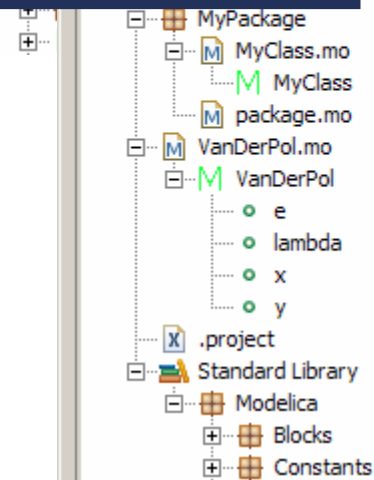
Code Assistance on function calling.

Code indentation



```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4 import Modelica.Math;
5 Real x(start = 1);
6 Real y(start = 1);
7 parameter Real lambda = 0.3;
8 parameter Real e = Modelica.Constants.e;
9
10 equation
11   der(x) = y;
12   der(y) = - x + lambda*(1 - x*x)*y;
13 end VanDerPol;
```

Code
Indentation



```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4 import Modelica.Math;
5   Real x(start = 1);
6   Real y(start = 1);
7   parameter Real lambda = 0.3;
8   parameter Real e = Modelica.Constants.e;
9
10  equation
11    der(x) = y;
12    der(y) = - x + lambda*(1 - x*x)*y;
13 end VanDerPol;
```

Code folding

The image displays two screenshots of the Eclipse IDE interface, illustrating the process of code folding in a Modelica file. The top screenshot shows the source code for a Van der Pol model in the editor, with the parameter declaration line highlighted. A blue arrow points from this line to the bottom screenshot, where the same line is now collapsed into a single line with a small square icon at the end, demonstrating the 'code folding' feature.

Top Screenshot (Expanded Code):

```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
4 import Modelica.Math;
5   Real x(start = 1);
6   Real y(start = 1);
7   parameter Real lambda = 0.3;
8   parameter Real e = Modelica.Constants.e;
9 equation
10  der(x) = y;
11  der(y) = -x + lambda*(1 - x*x)*y;
12
13
```

Bottom Screenshot (Collapsed Code):

```
1 // Van der Pol model
2
3 model VanDerPol "Van der Pol oscillator model"
13
```

Code Folding

Conclusions and Future work

- Conclusions
 - advanced Modelica/MetaModelica Eclipse Environment
 - project, package, class, model management
 - code browsing and assistance
 - integrated debugging
- Future Work
 - support refactorings
 - better code checking
 - better code navigation (hyperlinks, go to definition)
 - faster debugging
 - more code assistance
 - code templates
 - UML view of Modelica/MetaModelica Code

Demo

Thank you!
Questions?

<http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html>