



Modelica on The Web

Tamas Kecskes, Patrik Meijer, Janos
Sztipanovits, Peter Fritzson, Adrian
Pop, Arunkumar Palanisamy

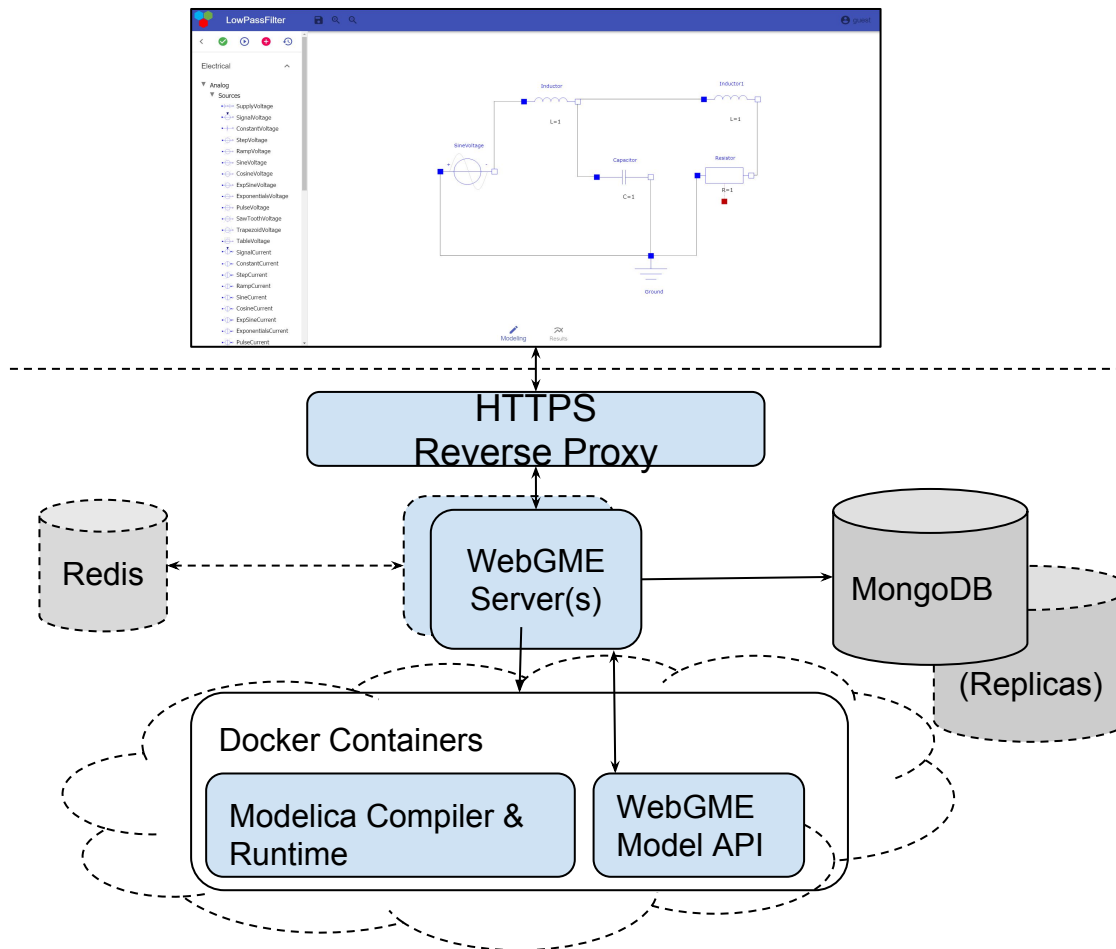


Motivation

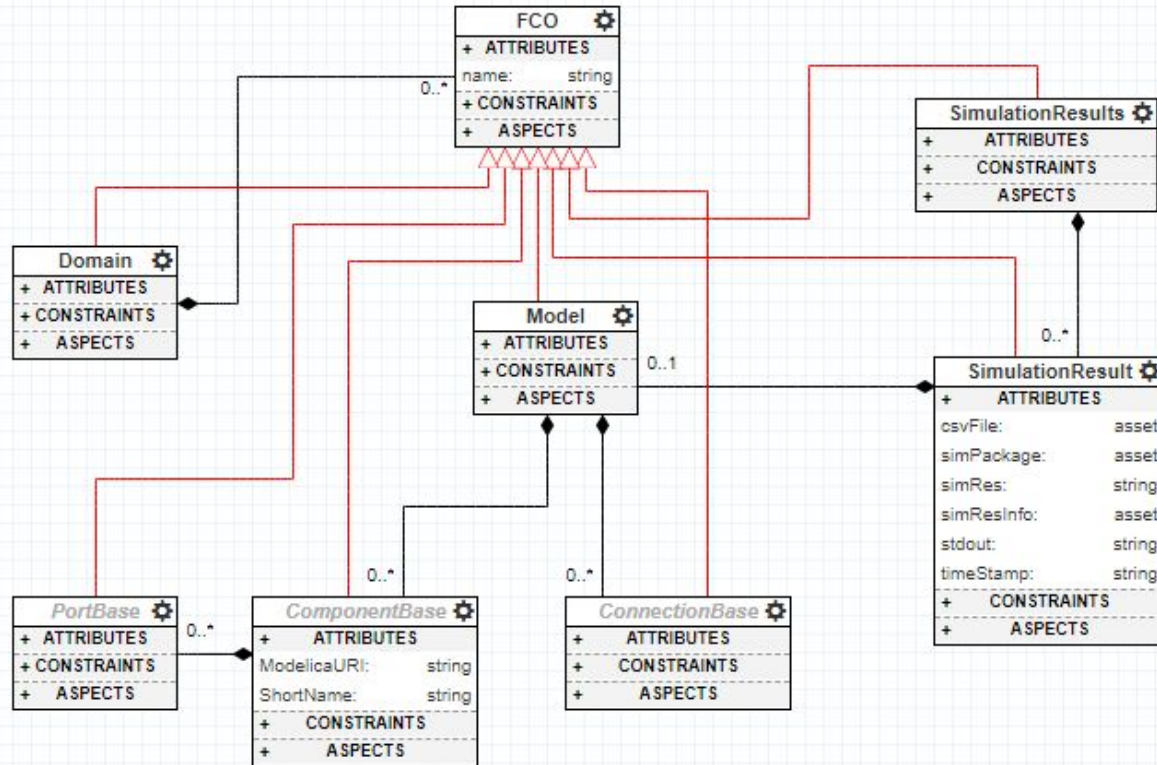


- Drawbacks of current Modelica solutions
 - Missing integrated collaboration features
 - No web presence
- WebGME was built around collaboration and web presence
- The meta-programmable modeling environment WebGME can model Modelica Standard Language and integrate many of its features into a Design Studio
- Tailored modeling and simulation experience for beginner Modelica users
- Foundation for post-simulation analysis (optimization, surrogate model training)

Architecture

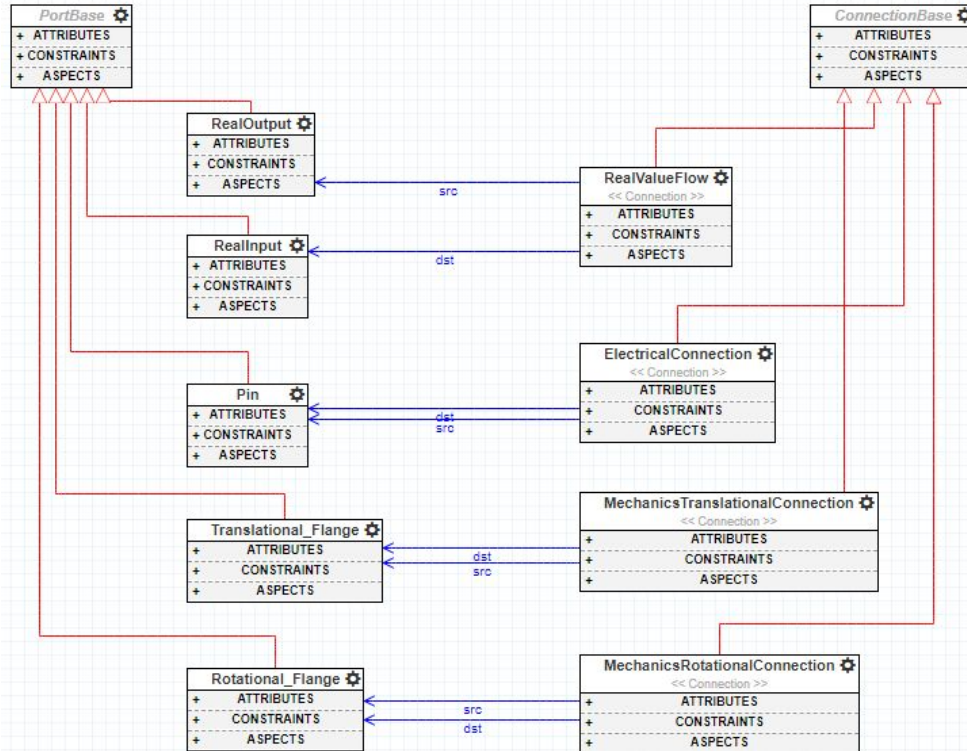


The Modelica Metamodel



- The basic elements of the domain that allow model creation and simulation
- FCO is the root of inheritance tree in WebGME
- Every new component can be defined by inheriting from the Component/Connection/Port bases
- The inheritance feature of WebGME allows extension of the domain later

The Modelica Metamodel



- Connections in WebGME are defined by the use of two specific pointers (src and dst)
- Pointers are directed one-to-one associations
- These definitions will govern the UI of the end-user, so that only well-formed models could be created



WebGME-DSS



- The user interface is clean and simple, focusing on model creation
- The start page allows project creation (by listing available modelica domains) or opening projects that are accessible to the user
- At project creation domains of MSL are selected (existing projects can later be updated with more domains)

The screenshot shows the WebGME-DSS web application interface. At the top, there is a dark header bar with the GME logo on the left and a 'guest' user indicator on the right. Below the header, a large dark banner contains the text 'Welcome to WebGME-DSS' and a paragraph explaining that it is a graphical editor for Modelica with simulation support using JModelica.org or OpenModelica. Below this banner, there are two main content areas. The left area is titled 'Electrical Analog' and features an image of electronic components. The right area is titled 'Rotational Mechanics' and features an image of a train engine. Both areas have a 'CREATE' button and a 'LEARN MORE' link. To the right of these two areas, there is a section titled 'CURRENT PROJECTS' which lists 'ElectricalAnalog', 'MassSpringDamper', 'TranslationalMechanics', and 'RotationalMechanics', each with a small thumbnail icon.

guest

Welcome to WebGME-DSS

WebGME Dynamic Systems Studio is a graphical editor for Modelica® with simulation support using [JModelica.org](#) or [OpenModelica](#). This current deployment runs JModelica.org on the backend.

Electrical Analog

This package contains packages for analog electrical components. Including basic components such as resistor, capacitor, conductor, inductor, transformer, gyrator etc. It also includes sources and sensors.

[CREATE](#) [LEARN MORE](#)

Rotational Mechanics

Library Rotational is a free Modelica package providing 1-dimensional, rotational mechanical components to model in a convenient way drive trains with frictional losses.

[CREATE](#) [LEARN MORE](#)

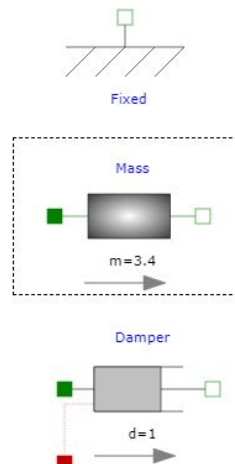
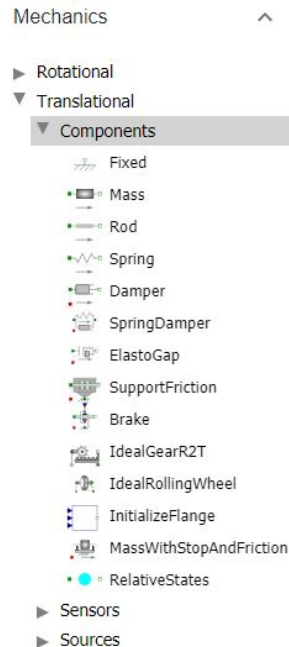
CURRENT PROJECTS

- ElectricalAnalog
- MassSpringDamper
- TranslationalMechanics
- RotationalMechanics



WebGME-DSS - Modeling View

- Standard editor features are available (create by drag-and-drop, edit parameters of elements, connect compatible ports)
- The user can initiate simulations from here
- Automatic history of the model is available (version control)
- Every change will generate an entry in the history
- The user can make additional entry points in the history with message
- Different versions of the project can be compared
- Changes can be reverted and earlier versions can be restored



Parameters

Mechanics.Translational.Components.Mass

Mass

m=3,4

name

Mass

modifiers

Enter optional modifiers, e.g., phi(start=1), w(start=2)

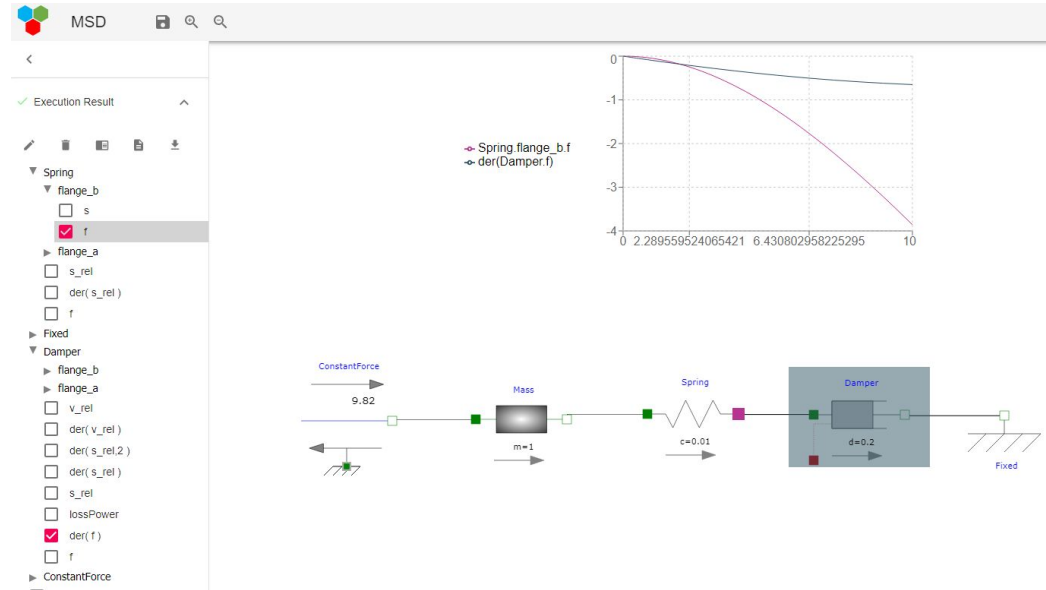
L [m]

1

Length of component, from left flange to right flange (= flange_b.s - flange_a.s)

WebGME-DSS - Simulation View

- Is activated automatically whenever a new simulation is initiated on the modeling view
- Each simulation preserves the version of the model it was run on, allows for traceability of stored results
- Selection is also visually reinforced by coloring the corresponding part of the model (allows easy tracking of the plotted variables)





Availability



- Currently a proof of concept implementation
- Available as Open Source (under MIT license) at <https://github.com/webgme/webgme-dss>
- It is also live and anyone can try it out at <https://cps-vo.org/group/modelica>
- The virtual organization has other design studios as well
- That deployment uses JModelica.org as a Modelica back-end, but when someone downloads the source, OpenModelica can be configured as backend also



Future Plans



- Extending the coverage of Modelica Standard Language components
- Separate mode for building up components (currently system level composition)
- Extend with textual editing
- Integration with Modia/Julia (beyond Modia code generation)



Future with OMWebbook



```
1 model VanDerPol      "Van der Pol oscillator model"
2   Real x(start = 1,fixed=true);
3   Real y(start = 1,fixed=true);
4   parameter Real lambda = 0.3;
5 equation
6   der(x) = y;
7   der (y) = - x + lambda*(1 - x*x)*y;
8 end VanDerPol;
```

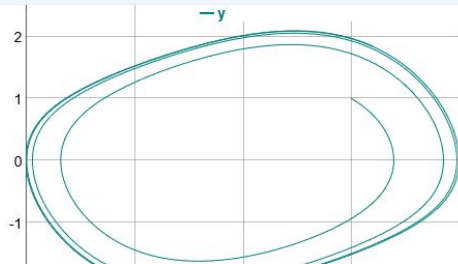
1 Simulation of Van der Pol

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.

```
1 simulate(VanDerPol, startTime=0, stopTime=25 )
2
```

Perform a parametric plot:

```
1 plotParametric( x, y )
2
```



- OMWebbok combines notebook and textual capabilities with computation (like Wolfram notebook or Jupyter notebook)
- <http://omwebbook.openmodelica.org/>
- Just like plotting, WebGME's visual editor could be embedded into a notebook and the user could choose the way how the models are defined (either visually or textually)
- Or even switching among different representations
- Also, the models then could be stored so alternatively be used among multiple notebooks