WebMWorks: A General Web-Based Modeling and Simulation Environment for Modelica

Liu Qi Xiong Tifan Liu Qinghua Chen Liping
CAD Center, Huazhong University of Science and Technology, Wuhan, China, 430074
luffy.lq@gmail.com xiongtf@hust.edu.cn liuqh@mail.hust.edu.cn chenlp@hustcad.com

Abstract

To meet the requirement of collaboration in the system-level modeling of multi-domain physical systems, a general web-based modeling and simulation environment, WebMWorks, is designed and implemented. It supports multi-user, multi-task and model sharing. Based on MWorks platform, the environment adopts SOA-based architecture and effectively solves the problems of sharing of simulation resources and reuse of the models. By application of RIA technologies, an interactive modeling and simulation environment based on the browser is constructed. This paper introduces the main characteristics and architecture of WebMWorks, and presents the operational effect of the system.

Keywords: visual modeling; web-based simulation; WebMWorks;

1 Introduction

Modelica is a non-proprietary, object-oriented, equation based language, and it has been widely applied because it is conveniently to express the model of complex physical system. Now, Modelica has become one kind of unified modeling standard for multi-domain physical systems. The system-level modeling of multi-domain physical systems using Modelica is difficult to be accomplished by an individual or individual enterprise because of its complexity and multidiscipline. So it is necessary to study the collaborative modeling using Modelica.

The web-based simulation is the integration of the web and simulation technology [1]. Compared to traditional simulation systems, the web-based simulation has many advantages [2], such as, wide usability, cross-platform capability, maintainability, upgradeability, and the sharing of the models.

With the openness, domain-independent and the unified expression of models, Modelica supports the reuse of simulation model based on the model framework at multiple levels. So we can study the web-based simulation for Modelica and realize the sharing of models and collaborative design in the complex system modeling, which has great practical significance.

At present, the related research is mainly around the virtual experiment and programming languages teaching. In reference [4] a web simulation environment UN-VirtualLab is presented, on which the virtual experiment can be defined by the administrator and the users can modify the experiment parameters to view the results in browser. In reference [5] a web version of the DrModelica is shown. In reference [6] a web-based teaching environment called OMWeb is presented. Student can send their exercises to compile and calculate on the server that contains many OMC (OpenModelica Compiler) wrappers and teachers can view the results. In reference [7] a web-based visual modeling environment was developed for electrical engineering experiment. Users can complete the experiment through connecting the custom experimental components, and the results of the experiment can be obtained after the simulation. However, all the studies presented are based on the specific purpose and lacks some features that are actually needed in industry and research, such as a general web-based visual graphical editor. To overcome these limitations, this paper describes a general web-based modeling and simulation environment: WebMWorks, on which users can easily perform system design, simulation and analysis in the browser. The WebMWorks is based on MWorks [10] and establishes the foundation to form the unified integrated platform for collaborative design and simulation.

2 MWorks Platform

MWorks is a general modeling and simulation platform for complex engineering systems, which provides the compiling and solving engines for WebMWorks.

The framework of MWorks platform is shown in Fig.1.

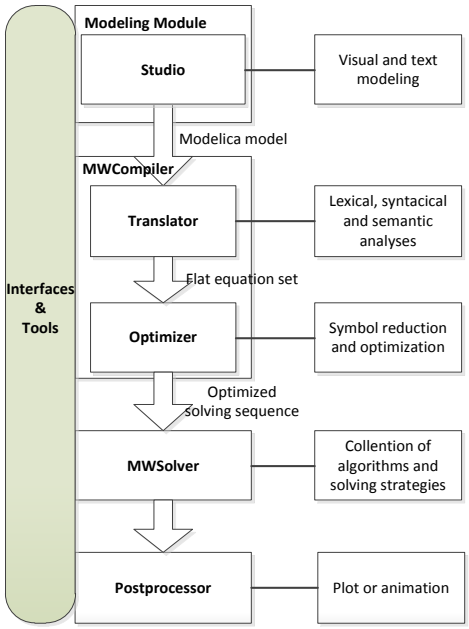

Figure 1: The framework and main process of MWorks

The platform is mainly composed of *Modeling Module*, *MWCompiler*, *MWSolver* and *Postprocessor*. A set of interfaces from MWorks are provided and can be called by external applications. In the WebMWorks, the *MWCompiler* and the *MWSolver* are called to perform the compilation, symbolic reduction and numeric-computation and results can be obtained for the simulation of the Modelica models.

3 Scheme selection

As a prototype platform for collaborative design and simulation based on Modelica, WebMWorks should have the following features:

1) Support visual modeling based on web browser.
2) Support visualization of simulation results based on web browser.
3) Support multi-user and multi-task.
4) Support collaborative modeling based on model sharing

In order to realize feature 3) and 4), naturally, Modelica simulator should be located on the server side. Remote compilation and simulation of Modelica models also has been studied and achieved by many researchers [5-9].

Feature 1) is a foundation of the platform. In order to achieve this target, we have two choices: one is downloading Modelica model libraries to the client and analyze the libraries. The users can complete the system modeling with the information of each model which was obtained by reading its *model text*. Then the system model will be sent to the server and simulated. The other is putting the analysis program for Modelica models on the server side. The client gets the graphical information of each model from the server to realize the visual modeling. Then scenes of the system model are sent to the server. These scenes data will be resolved to *model text*, and finally simulated by calling the compiler.

The first scheme will consume much time when the libraries have a large volume. In this way, it is unable to achieve the advantages of the web system that users can use at anytime, anywhere. And it is also not conductive to protect the intellectual property, when the system contains some private model libraries. The second scheme needs to maintain a free communication between the client and the server in the process of modeling. In spite of this, in order to explore the full benefits of web-based simulation, the second scheme should be chosen.

4 System Design

4.1 System Architecture

To build a modeling and simulation architecture that have indifferent interfaces, is reusable and loosely-coupled, WebMWorks adopts the idea of *Serivce Oriented Architecture* (SOA) which is implemented by using WCF technology. WCF is a unified programming model provided by Microsoft for building service-oriented applications. With WCF, the core functions of the system is wrapped as a service, and called by the internal or external program. The system based on a layered architecture is shown in Fig.2.

- *Presentation tier*

  The presentation tier is a web portal which contains a Silverlight plug-in. It can be used to provide visual modeling and the visualization of the simulation results. Silverlight is a RIA (Rich Internet Application) technology released by Microsoft, with the capabilities of cross-browser and cross-platform.

- *Service tier*

  This tier consists of three independent servers, including the web server, the modeling server and the simulation server. By using WCF technology to package the functions of the program on the server, *the presentation tier* can get the variety of services from *the service tier*.

  Web server is used to host the client portal, containing the web pages and silverlight plug-

DOI 10.3384/ecp12076549

in package. It handles various requests from the client, and provides the services of user management and model management.

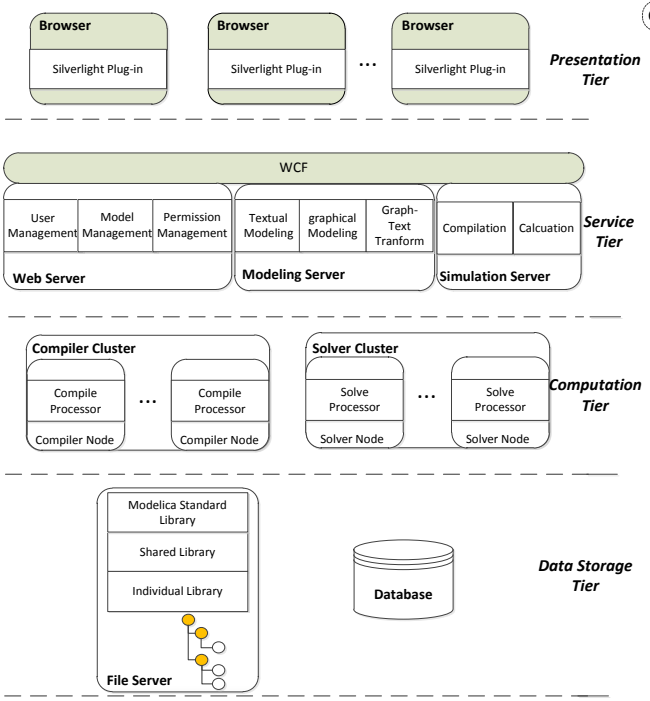

Figure 2: System architecture of WebMWorks

Modeling server receives requests from the user while modeling, including textual modeling request and graphical modeling request. It provides the services of textual modeling, graphical modeling and graph-text transformation.

Simulation server receives requests of compilation and solution from the client. It provides the compilation service and computation service for the *presentation tier*. There are two message queues in this server, the compilation queue and the solution queue. These queues are realized using Microsoft Message Queuing which has many advantages: stability, priority of the message, security and so on. When a compilation or solution request is received, simulation server will put it into the corresponding queue. Then the server makes a balanced distribution of the requests to the nodes of compiler cluster or solver cluster.

- *Computation tier*

The tier consists of two computing cluster: *Compiler cluster* and *Solver cluster*.

Each node in *Compiler cluster* performs the same function, and there is one or more process programs called *CompileProcessor* for processing compilation tasks in it. The number of processors is determined by the service application on the simulation server. The workflow in the node is shown in Fig.3.

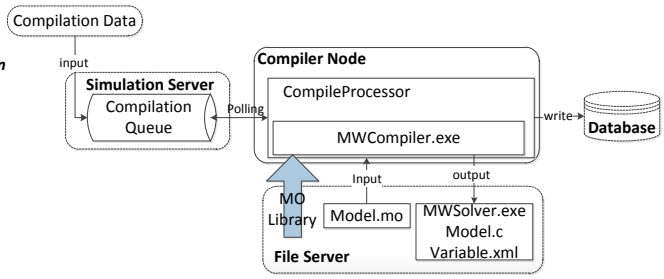

Figure 3: Workflow of each Compiler Node

When a compilation command is executed at the client-side, a request contains compilation data in XML format will be sent into the compilation queue on the simulation server. Compilation data contains several properties of the model to be compiled, including owner, ID, path and so on. *CompileProcessor* is a wrapper for *MWCompiler* which is the compiler of MWorks. It is constantly polling the compilation queue to get the compilation data. Then it calls the *MWCompiler* to load the MSL, the Shared library and the active user's library from the file server. The compiler takes the mo text of one model as input, and outputs the corresponding solver (MWsolver.exe), C code, and a XML file for the description of variables. Also, the compilation information will be stored to the database.

Similar to the *Compiler cluster*, *Solver cluster* is composed of several solver nodes which distribute the solution request. Each node can start one or more *SolveProcessor* to process the solution tasks. The workflow is shown in Fig.4.

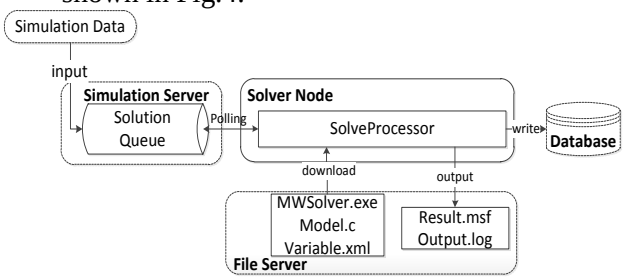

Figure 4: Workflow of each Solver Node

When a simulation command is executed at the client-side, a request contains simulation data in XML format will be sent into the solution queue on the simulation server. Simulation data contains three parts of information: the first is the model information, including owner, ID, path and so on; the second is the setting information of simulation, including start/stop time,

step length, algorithm; the third is the simulation parameters which refer to the modified parameters by users in the post-processing rather than the initial parameters. *SolveProcessor* will be constantly polling the solution queue to get the simulation data and then the *MWSolver* will be downloaded from the file server. *MWSolver* takes simulation data as input, and outputs the result file and the log file of the solution.

- *Data storage tier*

This tier consists of the database server and file server. Database server manages the information of the users and the metadata of Modelica models. The metadata contains the relationship between models and the properties of the Modelica models, such as type, name, path, description and owner. File server is used to store mo files of the models following the storage rules of Modelica models. It also stores SVG files, result files of the simulation, and kinds of intermediate files.

4.2 System Workflow

Fig.5 shows the typical procedure, from logining the system, to creating a new system model, and finally upto gaining the simulation results.

1) *System initialization*. When the modeling sever is up, the modeling application loads Modelica Standard Library (MSL) and Shared library from the file server, and provides all needed services in the process of modeling.
2) *User environment initialization*. After successful login, the user will be assigned the appropriate permissions. Simultaneously the modeling application loads individual library of the user from the file server to the memory in.
3) *System graphical modeling*. In the modeling page, the user can create a system model in interactive environment, as in the traditional Modelica IDE, such as Dymona, MWorks or OMEdit. The information which required in the modeling like the icons, parameters, properties of the components are achieved through calling the services provided by the *service tier*.
4) *Model compilation*. First, the client submits the scene presentation of one system model in XML format to the modeling application and calls the graph-text transformation service provided by the application. The scene will be resolved to *mo text* in the modeling server and the *mo text* will be saved to the file server. Second, the client calls the compilation services to send the compilation request into the compilation queue. Finally, the *CompilerProcessor* gets the compilation data from the queue, compiles the mo file of the system model and generates the corresponding solver.
5) *Model solution*. The request of the simulation from the client will be put into the solution queue. Then the *SolveProcessor* in one solver node gets the simulation data from the queue and calls the *MWSolver* of the model to generate the result data of simulation.
6) *Post-proessing*. In the post-processing page, the user can monitor the process of simulation. When the solution is completed, the result data can be packed into XML and sent to the client and displayed on the page.

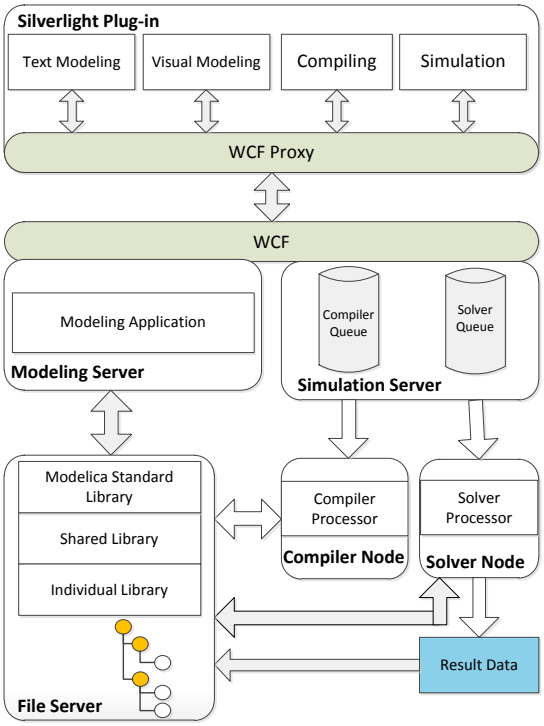

Figure 5: System workflow of modeling and simulation

DOI 10.3384/ecp12076549

5 System Implementation

5.1 Client: Modeling Page

The client is developed in a MVVM [12] architecture using the Silverlight technology. The vector graphics and asynchronous communication of Silverlight make it easy to create interactive graphical application in the browser. While a large number of graphical operations are transferred from the server to the client, the burden of the server is lightened, which allows the server with same hardware to handle more requests.

In the modeling page, a visual modeling environment, which is similar with MWork Studio, has been realized. It supports users to create, modify, delete, query and download models. Each model has three views: text view, icon view and diagram view. The screenshot of the modeling page is shown in Fig.6.

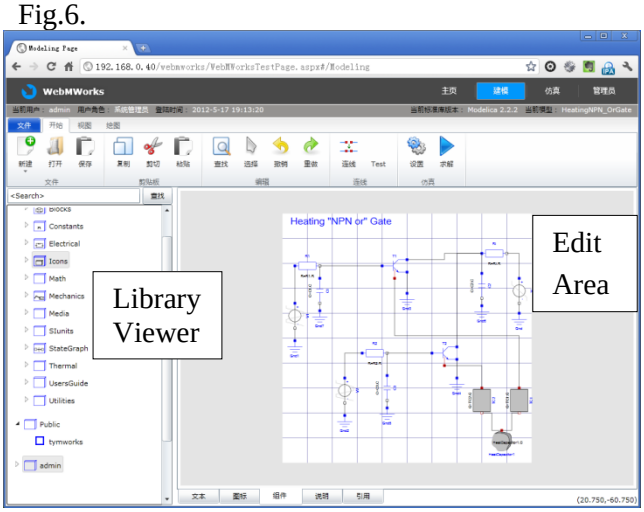

Figure 6: Modeling Environment in Browser

The library viewer contains three root nodes: “Modelica”, “Public” and “CurrentUserName”. The “Modelica” node represents a specific version of MSL. The user can decide which version of MSL to load in the login page. The “Public” node represents the models that the users shared. And the “CurrentUserName” Node is on behalf of the models owned by the user who has logged in. The models under the “Public” and “CurrentUserName” node are consistent with the version of the MSL. For example, when the user selects the version 2.2.2 of MSL, they have only loaded the models based on the Modelica Standard Library 2.2.2. The tree of the library viewer adopts the lazy loading mode, that is, only when the user expands one node, the next level nodes of the node are loaded.

The edit area acts as visual modeling, text modeling or icon-editing area. The icon view provides basic graphics drawing, and the user can edit the icon of the model in this view. In the Diagram view, the model can be dragged and dropped from the library viewer to this view, and then the model will be instantiated to a component. The users can complete the system modeling by connecting components.

5.2 Client: Post-processing Page

In the Post-processing page, the visualization of simulation results, which is similar with the MWorks Simulator, has been realized. It offers simulation management, setting up and viewing the results of several simulation cases at the same time. The screenshot of the Post-processing page is shown in Fig.7：

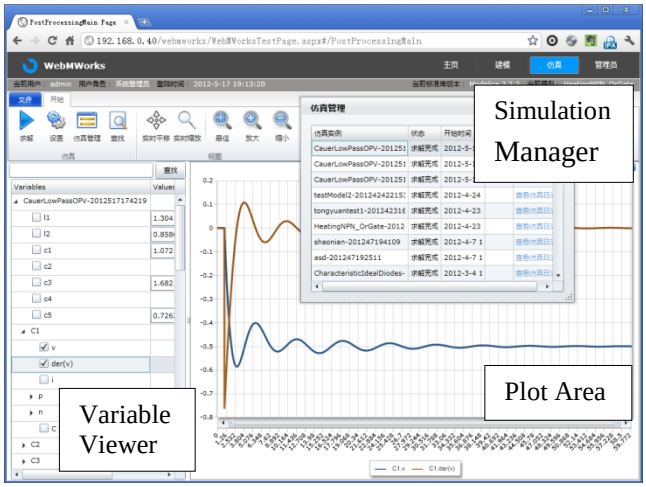

Figure 7: Post-Processing Environment in Browser

The *simulation manager* can monitor and control which state the model is in the compiling and simulation queue. The possible states are: queuing, processing, finishing and failure. Also, the simulation log can be shown in the window.

The variable nodes in variable viewer also adopt the lazy loading mode. When the variable nodes are chosen, the set of dots will be downloaded and shown in the plot Area on the right.

5.3 Modeling Application

In the graphical modeling client, the graph which is shown in icon/diagram view defined by Modelica annotations. To get Modelica annotation, the client of WebMWorks used .NET platform to re-implement some functions of modeling module of MWorks, call the interface of compiling through p/invoke, and at last realize loading Modelica model library. The package of the function which the client modeling need is achieved by WCF, and can be shown to provide visual modeling service.

The modeling application can provide the WCF interface which can get SVG of icon view of model (scalable vector graphics, which is based on XML).

For example, *GetIconSVG (Modelica.Electrical.Analog.Basic.Resistor)* can get the SVG of icon view of the Resistor Model.

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<svg width="200" height="200">
 <g transform="matrix(1 0 0 -1 100 100)">
  <line x1="-60" y1="50" x2="60" y2="50" style="stroke:rgb(0,0,255);fill:none;stroke-width:0.5;" />
  <line x1="-40" y1="30" x2="40" y2="30" style="stroke:rgb(0,0,255);fill:none;stroke-width:0.5;" />
  <line x1="-20" y1="10" x2="20" y2="10" style="stroke:rgb(0,0,255);fill:none;stroke-width:0.5;" />
  <line x1="0" y1="90" x2="0" y2="50" style="stroke:rgb(0,0,255);fill:none;stroke-width:0.5;" />
  <g transform="matrix(1 0 -0 -1 0 0)">
   <text x="-78.8363876342773" y="53.113208770752" font-size="48.3018867924528" style="fill:rgb(0,0,255)">%name</text>
  </g>
  <g transform="matrix(7.54979e-009 0.1 0.1 -7.54979e-009 9.53674e-007 100)"
 isConnector="true" name="p" type="Modelica.Electrical.Analog.Interfaces.Pin" description=""
 parent="Modelica.Electrical.Analog.Basic.Ground">
   <rect x="-100" y="-100" width="200" height="200" style="stroke:rgb(0,0,255);fill:rgb(0,0,255);stroke-width:0.5;" />
  </g>
 </g>
</svg>
```

Figure 8: Icon SVG of the Resistor model

The client can call the interface of sever through the proxy of WCF to get the SVG which then will be displayed in the icon view of *Resistor* model.

5.4 Multi-task implementation

Compilation and calculation of the model based on Modelica are always time-consuming. In the case of multi-user, when a model is compiled by compiler, others cannot be compiled along. (MWCompiler does not yet support compiling more than one model at the same time).

So, the compilation and calculation of WebMWorks process in queue is formed to solve concurrent calculation and compilation of multi-tasking and multi-user. In this method, user can submit multi-simulation task in the client, and quit the system after submitting tasks, which explore the full benefits of the web systems

6 Conclusions and future work

This article presents a general web-based modeling and simulation environment: WebMWorks. The design and implementation of the environment are described. Through transplanting the stand-alone tools of modeling and simulation for Modelica into the Web, the usage scenarios are greatly expanded:

For the enterprise and research organizations, the co-design and co-simulation could be achieved, based on the tools of modeling and simulation on web and through combining with model sharing and workflow management.

For individuals, the characteristics of the Web system, such as cross-platform capability, wide accessibility, would improve the efficiency of modeling and simulation. Model sharing and reuse, multi-tasking also helps to improve the speed of the modeling and simulation.

But, the prototype of WebMWorks lacks of several functions compared with MWorks. In the future, it should be improved gradually in the following areas.

- Enforce the capability of data transport, especially compression and decompression of the transporting data;
- Add the function of highlighting and code folding for textual modeling;
- Improve safety of the model store;
- Add the function of 3D visualization in post-processing.

References

[1] Tummescheit H. Design and Implementation of Object-Oriented Model Libraries using Modelica. Lund, Sweden: PhD thesis, Department of Automatic control, Lund Institute of Technology, 2002.

[2] James Byrne, Cathal Heavey, P.J. Byrne. A review of Web-based simulation and supporting tools. *Simulation Modelling Practice and Theory 18 (2010) 253–276.*

[3] Zhou Fan-li, Guo Jun-feng, Zhao Jian-jun, Chen Li-ping. Reusability of Modleica Simulation Model. System Simulation Technology & Application (Vol 11).

[4] Oscar Duarte. UN-VirtualLab : A web simulation environment of OpenModelica models for educational purposes. *Proceedings 8th Modelica Conference, Dresden, Germany, March 20-22, 2011*

[5] Eva-lena Lengquist S , Susanna Monemar , Peter Fritzson , Peter Bunus DrModelica – A WebBased Teaching Environment for Modelica In Proceedings of the 44th Scandinavian Conference on Simulation and Modeling (SIMS'2003)

[6] Mohsen Torabzadeh-Tari, Zoheb Muhammed Hossain, Peter Fritzson, Thomas Richter. OMWeb – Virtual Web-based Remote Laboratory for Modelica in Engineering Courses. *Proceedings 8th Modelica Confe-*

rence, Dresden, Germany, March 20-22, 2011.

[7] Zhengyin Shi, Shenglin Zhao, Shan-an Zhu. An Internet-based Electrical Engineering Virtual Lab: Using Modelica for Unified Modeling. Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on 27-29 May 2011.

[8] Sven Meyer zu Eissen, Benno Stein Realization of Web-based simulation services Computers in Industry 57 (2006) 261–271

[9] Björn Johansson. COMPUTATIONAL METHODS APPLIED TO MODELICA SIMULATION MODELS IN A WEB BASED FRAMEWORK. Proceedings of IDETC/CIE ,September 24-28, 2005, Long Beach, California USA

[10] F.-L. Zhou, L.-P. Chen, Y.-Z. Wu, J.-W. Ding, J.-J. Zhao, Y.-Q. Zhang. MWorks: a Modern IDE for Modeling and Simulation of Multi-domain Physical Systems Based on Modelica. *Modelica 2006, Vienna Austria, September 2006*

[11] Sven Meyer zu Eissen, Benno Stein. WEB-BASED SIMULATION:APPLICATION SCENARIOS AND REALIZATION ALTERNATIVES. Proceedings of the TMCE 2004, April 13–17, 2004

[12] http://en.wikipedia.org/wiki/Model_View_ViewModel.

DOI 10.3384/ecp12076549