The OpenModelica Integrated Modeling, Simulation and Optimization Environment

Peter Fritzson[1], Adrian Pop[1], Adeel Asghar[1], Bernhard Bachmann[1], Willi Braun[2], Robert Braun[1], Lena Buffoni[1], Francesco Casella[3], Rodrigo Castro[6], Alejandro Danós[6], Rüdiger Franke[7], Mahder Gebremedhin[1], Bernt Lie[8], Alachew Mengist[1], Kannan Moudgalya[5], Lennart Ochel[1], Arunkumar Palanisamy[1], Wladimir Schamai[9], Martin Sjölund[1], Bernhard Thiele[1], Volker Waurich[4], Per Östlund[1]

[1]PELAB – Programming Environment Lab, Dept. of Computer and Information Science
Linköping University, SE-581 83 Linköping, Sweden
[2]FH Bielefeld, Bielefeld, Germany
[3]Dept. Electronics and Information, Politecnico di Milano, Milan, Italy
[4]TU Dresden, Dresden, Germany
[5]IIT Bombay, Mumbai, India
[6]Dept. Computer Science, Universidad de Buenos Aires, Argentina
[7]ABB AG, DE-68309 Mannheim, Germany
[8]University of South-Eastern Norway, Porsgrunn, Norway
[9]Danfoss Power Solutions GmbH & Co. OHG, Offenbach, Germany

peter.fritzson@liu.se, adrian.pop@liu.se

Abstract

OpenModelica is currently the most complete open-source Modelica- and FMI-based modeling, simulation, optimization, and model-based development environment. Moreover, the OpenModelica environment provides a number of facilities such as debugging; optimization; visualization and 3D animation; web-based model editing and simulation; scripting from Modelica, Python, Julia, and Matlab; efficient simulation and co-simulation of FMI-based models; compilation for embedded systems; Modelica-UML integration; requirement verification; and generation of parallel code for multi-ore architectures. The environment is based on Modelica and uses an extended version of Modelica for its implementation. This overview paper intends to give an up-to-date brief description of the capabilities of the system, and the main vision behind its development.

Keywords: Modelica, OpenModelica, MetaModelica, FMI, modeling, simulation, optimization, development, environment, compilation, embedded system, real-time

1 Introduction

The OpenModelica environment was the first open source Modelica environment supporting the Modelica modeling language (Modelica Association 2017) (Fritzson 2014). Its development started in 1997 resulting in the release of a flattening frontend for a core subset of Modelica 1.0 in 1998. After a pause of four years, the open source development resumed in 2002. An early version of OpenModelica is described in (Fritzson et al 2005). Since then the capabilities of OpenModelica have expanded enormously. The Open Source Modelica Consortium which supports the long-term development of OpenModelica was created in 2007, initially with seven founding organizations. The scope and intensity of the open source development has gradually increased. At the time of this writing the consortium has fifty-three supporting organizational members. The long-term vision for OpenModelica is an integrated and modular modeling, simulation, model-based development environment with additional capabilities such as optimization, sensitivity analysis, requirement verification, etc., which are described in the rest of this paper. The previous overview paper about OpenModelica was published 2005. The current paper intends to give a more up-to-date overview of the system and the vision and goals behind its development.

This paper is organized as follows. Section 2 presents the idea of integrated environment, Section 3 the goals for OpenModelica, Section 4 an overview of the OpenModelica environment, Section 5 and its subsections give more details about OpenModelica and its subsystems, Section 6 presents related work and Section 7 the conclusions.

2 Integrated Interactive Modeling and Simulation Environments

An integrated interactive modeling and simulation environment is a special case of programming environments with applications in modeling and simulation. Thus, it should fulfill the requirements both from general integrated interactive environments and

DOI
10.3384/ECP18154206

from the application area of modeling and simulation mentioned in the previous section.

The main idea of an integrated programming environment in general is that a number of programming support functions should be available within the same tool in a well-integrated way. This means that the functions should operate on the same data and program representations, exchange information when necessary, resulting in an environment that is both powerful and easy to use. An environment is interactive and incremental if it gives quick feedback, e.g., without re-computing everything from scratch, and maintains a dialogue with the user, including preserving the state of previous interactions with the user. Interactive environments are typically both more productive and more fun to use than non-interactive ones.

There are many things that one wants a programming environment to do for the programmer or modeler, particularly if it is interactive. Comprehensive software development environments are expected to provide support for the major development phases, such as:

- Requirements analysis
- Design
- Implementation
- Maintenance

A pure programming environment can be somewhat more restrictive and need not necessarily support early phases such as requirements analysis, but it is an advantage if such facilities are also included. The main point is to provide as much computer support as possible for different aspects of systems development, to free the developer from mundane tasks so that more time and effort can be spent on the essential issues.

Our vision for an integrated interactive modeling and simulation environment is to fulfill essentially all the requirements for general integrated interactive environments combined with the specific needs for modeling and simulation environments, e.g.:

- Specification of requirements, expressed as documentation and/or mathematics
- Design of the mathematical model
- Symbolic transformations of the mathematical model
- A uniform general language for model design, mathematics, and transformations
- Automatic generation of efficient simulation code
- Execution of simulations
- Debugging of models
- Design optimization
- Evaluation and documentation of numerical experiments
- Graphical presentation
- Model and system structure parameterization
- Variant and version handling, traceability

3 Goals for OpenModelica

The computational and simulation goals of the OpenModelica tool development include, but are not limited to, the following:

- Providing a complete open source Modelica-based industrial-strength implementation of the Modelica language, including modeling and simulation of equation-based models, system optimization, and additional facilities in the programming/modeling environment.
- Providing an interactive computational environment for the Modelica language. It turns out that with support of appropriate tools and libraries, Modelica is very well suited as a computational language for development and execution of numerical algorithms, e.g. for control system design and for solving nonlinear equation systems.

The research related goals and issues of the OpenModelica open source implementation of a Modelica environment include, but are not limited to, the following:

- Development of a *complete formal specification* and *reference implementation* of Modelica, including both static and dynamic semantics. Such a specification can be used to assist current and future Modelica implementers by providing a semantic reference, as a kind of reference implementation.
- *Language design*, e.g. to further *extend the scope* of the language, e.g. for use in diagnosis, structural analysis, system identification, integrated product development with requirement verification, etc., as well as modeling problems that require partial differential equations.
- *Language design* to *improve abstract properties* such as expressiveness, orthogonality, declarativity, reuse, configurability, architectural properties, etc.
- *Improved implementation techniques*, e.g. to enhance the performance of compiled Modelica code by generating code for parallel hardware.
- *Improved debugging* support for equation based languages such as Modelica, to make them even easier to use.
- *Improved optimization support*, with integrated optimization and modeling/simulation. Two kinds: parameter-sweep optimization based on multiple simulations; direct *dynamic optimization* of a goal function without lots of simulations, e.g., using collocation or multiple shooting.
- *Easy-to-use* specialized high-level (graphical) *user interfaces* for certain application domains.

- *Visualization* and animation techniques for interpretation and presentation of results.
- *Integrated requirement modeling and verification support.* This includes the ability to enter requirements formalized in a kind of Modelica style, and to verify that the requirements are fulfilled for selected models under certain usage scenarios.

The OpenModelica effort started by developing a rather complete formal specification of the Modelica language. This specification was developed in *Operational Semantics*, which still is the most popular and widely used semantics specification formalism in the programming language community. It was initially used as input for automatic generation of the Modelica translator implementations which are part of the OpenModelica environment. The RML compiler generation tool (our implementation of Operational Semantics) (Fritzson et al, 2009) was used for this task.

However, inspired by our vision of integrated interactive environments with self-specification of programs and data, and integrated modeling and simulation environments), in 2005 we designed and implemented an extension to Modelica called MetaModelica (Pop and Fritzson, 2006), (Fritzson, Pop, Sjölund, 2011). This was done in order to support language modeling and specification (including modeling the language itself), in addition to the usual physical systems modeling applications of Modelica, as well as applications requiring combined symbolic-numeric capabilities. Modeling the semantics in itself was also inspired by functional languages such as Standard ML (Milner 1997), and OCaml (OCaml org, 2018). Moreover, it was an investment into a future Modelica becoming a combined symbolic-numeric language such as Mathematica, but more efficient and statically strongly typed.

This language extension has a backwards-compatible Modelica-style syntax but was initially implemented on top of the RML compiler kernel. The declarative specification language primitives in RML with single-assignment pattern equations, possibly recursive case records (in MetaModelica called uniontypes) and match expressions, fit well into Modelica since it is a declarative equation-based language. In 2006 our whole formal specification of Modelica static and translational semantics, at that time about 50 000 lines, was automatically translated into MetaModelica. After that, all further development of the symbolic processing parts of the OpenModelica compiler (the run-time parts were mainly written in C), was done in MetaModelica.

At the same time we embarked on an effort to completely integrate the MetaModelica language extension into the Modelica language and the OpenModelica compiler. This would enable us to support both Modelica and MetaModelica by the same compiler. This would allow modeling the Modelica tool and the OpenModelica compiler using its own language. This would get rid of the limitations of the RML compiler kernel and the need to support two compilers. Moreover, additional tools such as our Modelica debugger can be based on a single compiler.

Such an ability of a compiler to compile itself is called compiler *bootstrapping*. This development turned out to be more difficult and time-consuming than initially expected; moreover, developers were not available for a few years due resource limitations and other priorities. Finally, bootstrapping of the whole OpenModelica compiler was achieved in 2011. Two years later, in 2013, all our OpenModelica compiler development was shifted to the new bootstrapped compiler (Sjölund, Fritzson, Pop, 2014), (Sjölund, 2015), after automatic memory reclamation (garbage collection), separate compilation, and a new efficient debugger had been achieved for our new compiler platform.

More recently, we have had an effort to restructure and rewrite the frontend part of the OpenModelica compiler (OMC). The reasons were two-fold: to support the exact Modelica semantics required to simulate certain models even though the semantics at that time was not clearly specified by the Modelica language specification (Modelica Association 2017), and to achieve much higher compilation speed for large and complex models. This work turned out to more difficult than expected. Fortunately, recently, a lot of progress has been made and a release of a preliminary version of this new frontend as part of OMC now appears feasible late fall 2018.

4 The OpenModelica Environment

At the time of this writing, the interactive OpenModelica environment primarily consists of the following components and subsystems:

- *A graphical and textual model editor*, OMEdit. This is a graphical connection editor for component based model design by connecting instances of Modelica classes. The editor also provides text editing. Moreover, the OMEdit GUI provides a graphical user interface to simulation and plotting (OMPlot). See Section 5.2.
- *An interactive session handler*, OMShell, that parses and interprets commands and Modelica expressions for evaluation. The session handler also contains simple history facilities, and completion of file names and certain identifiers in commands. There is also a Python variant of the interactive session handler called OMPython that supports the same commands in Python. Very recently, similar session handlers for Julia, called OMJulia, and Matlab, called OMMatlab, have been implemented. See Section 5.10.
- *A Modelica compiler*, OMC, translating Modelica to lower level code such as C code, with a symbol table

DOI
10.3384/ECP18154206

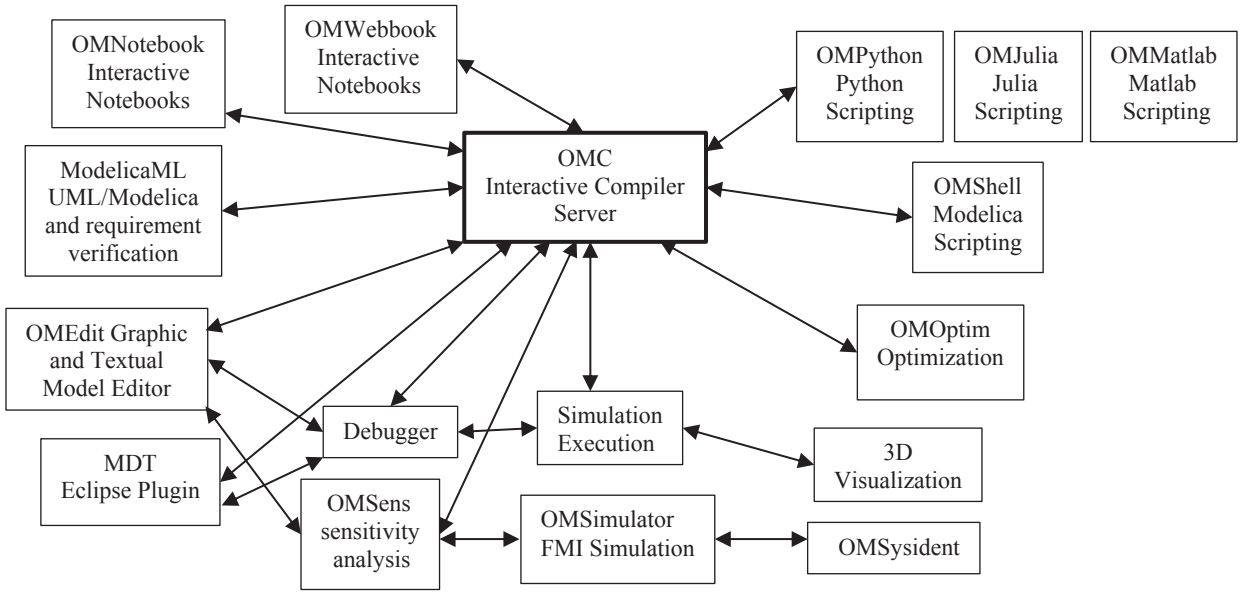

Figure 1. The architecture of the OpenModelica environment. Arrows denote data and control flow.

containing definitions of models, functions, and variables. Such definitions can be predefined, user-defined, or obtained from libraries. See Section 5.1. There is also a compilation mode to generate low-footprint code for embedded systems, see Section 5.13.

- *An execution and run-time module*. This module currently executes compiled binary code from translated models and functions. It includes numerical solvers as well as event handling facilities for the discrete and hybrid parts of the Modelica language. See Section 5.6.
- *Debuggers and performance analyzers*. These tools provide source-level Modelica debugging on equation models (Section 5.5), algorithmic model code (Section 5.4), as well as performance analysis of models (Section 5.5).
- *Textual model editors*. Any text editor can be used. Among the OpenModelica tools, text editing of models is supported by OMEdit (Section 5.2), by the OpenModelica MDT Eclipse plug-in (Section 5.6), and by the interactive electronic book OMNotebook (Section 5.7).
- *An interactive electronic book*, OMNotebook. This tool provides an active electronic book facility supporting chapters, sections, execution of simulation models, plotting, etc. One book, DrModelica, for teaching Modelica to the beginner, is automatically opened by default. The user can define his/her own books. This tool is useful for developing interactive course material. See Section 5.7.
- *Jupyter notebook for OpenModelica.* More recently, the Python-based Jupyter notebook has appeared, supporting a number of languages. Therefore we have also developed a Jupyter notebook for OpenModelica (OSMC 2018a) using Modelica scripting. However, Python scripting together with the OMPython package is used in the Jupyter notebooks presented in (Lie et al, 2016)
- *An interactive web-based electronic book, OMWebbook.* This is similar to OMNotebook, but model editing and simulation is in a web-browser. Simulation is performed by a simulation server. See Section 5.8.
- *An optimization module using parameter sweeps*, called OMOptim. This tool performs optimizations by running several simulations for different parameter settings while searching for the optimum value of a user-specified goal function. See Section 5.17.
- *A dynamic optimization module.* Direct optimization (without running lots of simulations) of a whole solution trajectory using collocation or multiple shooting. A goal function can be formulated to be optimized under the constraints of a selected model. See Section 5.17.
- *Requirement verification and ModelicaML Eclipse plug-in*. This plug-in contains a Modelica-UML profile that allows integrated requirement verification and cyber-physical hardware-software modeling by combining hardware modeling in Modelica with software modeling using UML. The tool contains a UML to Modelica translator that makes it possible to simulate combined UML-Modelica models. Moreover, automatic (dynamic) verification of formalized requirements against selected scenarios is supported by ModelicaML or by a Modelica-based approach without using UML. See Section 5.15 and Section 5.16.

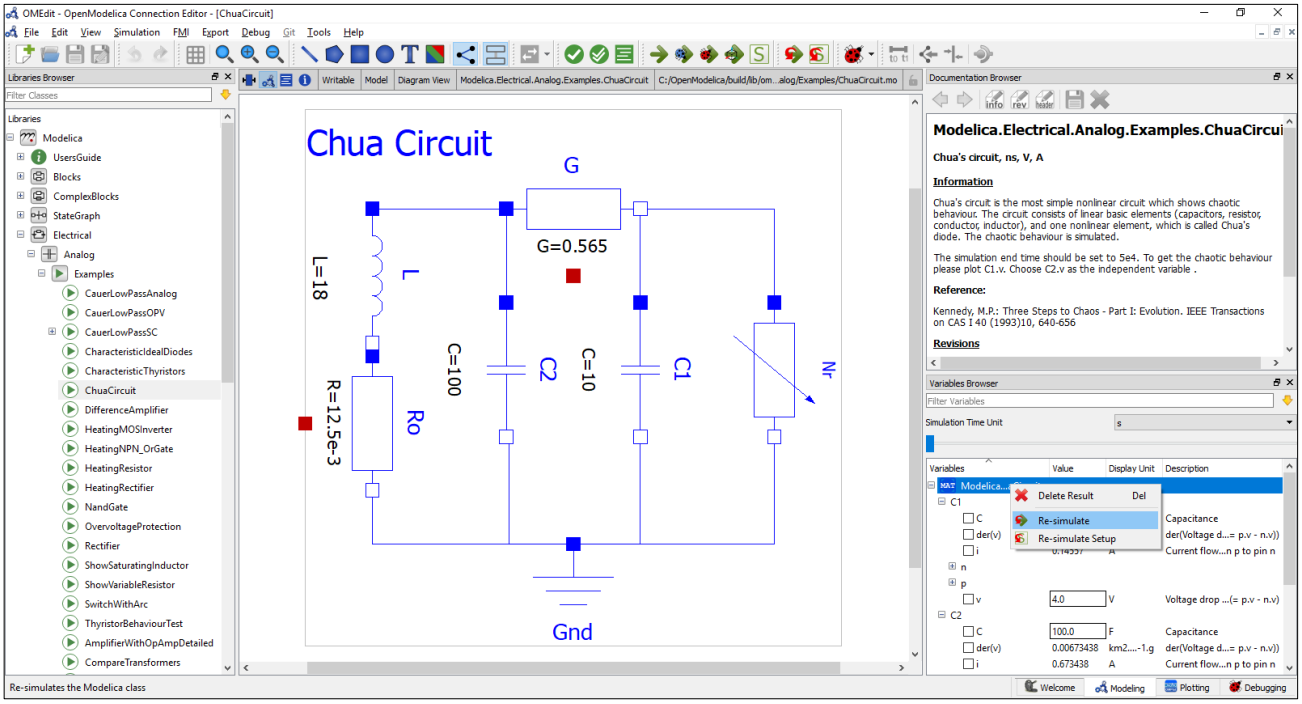

Figure 2. OMEdit on the Modelica.Electrical.Analog. Examples.ChuaCircuit model. *Center*: Model connection diagram. *Upper right*: information window. *Lower right*: plot variable browser will a small popup re-simulate menu on top.

- *MDT Eclipse plug-in.* The MDT (Modelica Development Tooling) Eclipse plug-in for Modelica library and model compiler textual development, project support, cross-referencing, building executables, debugging, etc. See Section 5.6.
- *3D animation visualization.* This is provided by a special module in OpenModelica, and uses the standard Modelica MBS library 3D graphical annotations. See Section 5.3.
- *FMI Import and Export.* A model (including models from other tools, even non-Modelica ones) can be imported or exported according to the FMI (Functional Mockup Interface) standard as an FMU (Functional Mockup Unit).
- *OMSimulator* FMI-based simulation and co-simulation subsystem. This recently added subsystem, which also can be run stand-alone separated from the OpenModelica compiler, supports efficient simulation and co-simulation of single or composite FMUs. FMUs can also be connected using a graphical editor to form composite FMUs. See Section 5.12.
- *OMSens.* Sensitivity analysis subsystem that allows both single-parameter and multi-parameter analysis, the latter based on robust optimization techniques. Specification of the analysis and display of results can be made interactively via OMEdit in the current prototype. An early prototype not yet integrated in OMEdit is described in (Danós et al, 2017).
- *OMSysIdent.* A parameter system identification module, using system identification vs measurement data to determine the best model parameter values for a certain model (OSMC 2018c).
- *MetaModelica language extension.* This is used for modeling/specification of languages (including the Modelica language) and for Modelica programming of model transformations (Pop and Fritzson, 2006), (Fritzson, Pop, Sjölund, 2011). Related to this, there are discussions in the Modelica Design group about possible extensions to the Modelica language that would enable definition some language constructs in a Modelica core library instead of being hardcoded in the compiler.
- *Parallelization and ParModelica language extension.* ParModelica is used for explicit algorithmic parallel Modelica programming with compilation to both multi-core CPU platforms and GPGPU platforms (including NVIDIA). See Section 5.18.

5 OpenModelica Subsystems

The relationships between the main OpenModelica subsystems is depicted above in Figure 1. Their functionality is briefly described in the following.

DOI
10.3384/ECP18154206

5.1 OMC – The OpenModelica Model Compiler

OMC is the OpenModelica compiler which translates Modelica models into C/C++ code (or Java or C# code using experimental code generators), which is compiled and executed to perform simulations. The OpenModelica compiler is generated from formal specifications in RML (earlier) or MetaModelica (currently). At the time of this writing the OpenModelica compiler (OMC) is generated from a specification of about two hundred thousand lines of MetaModelica. Moreover, OMC is able to compile itself, i.e., it is bootstrapped.

5.2 OMEdit – the OpenModelica Graphic Model Editor and Simulator GUI

OMEdit is the OpenModelica graphic model editor (Figure 2), (Asghar et al, 2011). In addition to graphic/textual model editing and browsing, it also provides model text editing, simulation, parameter update, debugging, and plotting capabilities.

Using OMEdit to perform simulations and plotting simulation results is depicted in Figure 3 below.

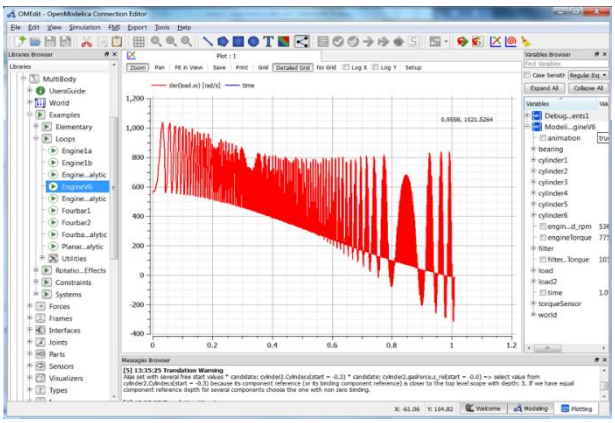

Figure 3. OpenModelica simulation of the V6Engine model with 11000 equations. Plotting simulation results using OMEdit. *Left*: Model browser. *Right*: Plot variable browser. *Bottom*: message browser window.

5.3 3D Animation and Visualization

The OpenModelica 3D animation and visualization is a built-in feature of OMEdit to animate based on 3D shapes defined by the MSL Muilti-Body library. It provides visualization of simulation results and animation of geometric primitives and CAD-files. There is also support for FMI-based visualization (Waurich and Weber, 2017).

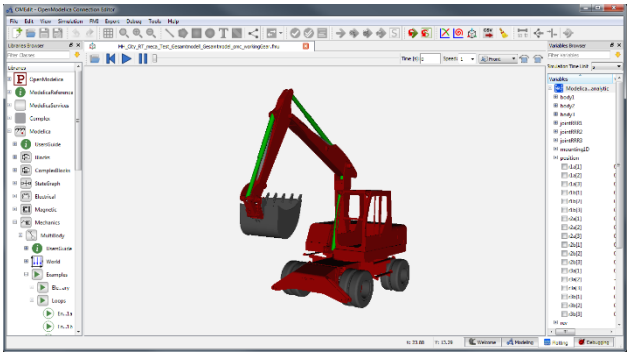

Figure 4. OpenModelica 3D animation of a simulated excavator.

5.4 The OpenModelica Algorithm Debugger

The OpenModelica algorithm debugger (Figure 5), (Pop, 2008), (Sjölund, 2015) is available for use either from OMEdit or from the MDT Eclipse plug-in. The debugger provides traditional debugging of the algorithmic part of Modelica, such as setting breakpoints, starting and stopping execution, single-stepping, inspecting and changing variables, inspecting all kinds of standard Modelica data structures as well as MetaModelica data structures such as trees and lists.

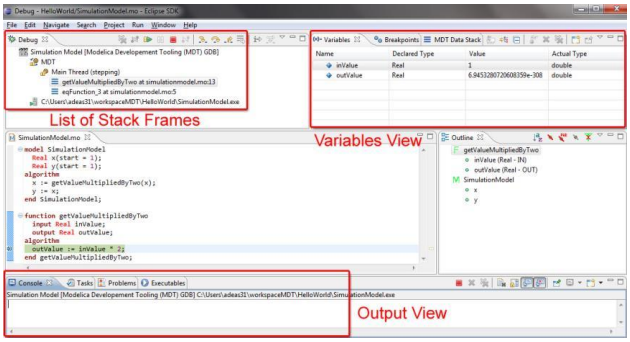

Figure 5. The OpenModelica algorithmic code debugger viewed from the MDT Eclipse plug-in. The OMEdit version of the debugger looks about the same. A breakpoint has been set in the function which is called from the small model called SimulationModel.

5.5 The OpenModelica Equation Model Debugger and Performance Analyzer

The OpenModelica equational model debugger (Figure 6), (Pop, Sjölund, et al, 2014), (Sjölund, 2015) is available for use from OMEdit. It provides capabilities for debugging equation-based models, such as showing and explaining the symbolic transformations performed on selected equations on the way to executable simulation code. It can locate the source code position of an equation causing a problem such as a run-time error, traced backwards via the symbolic transformations. Moreover, a *performance analyzer tool* is also included in OpenModelica and integrated with the debugger.

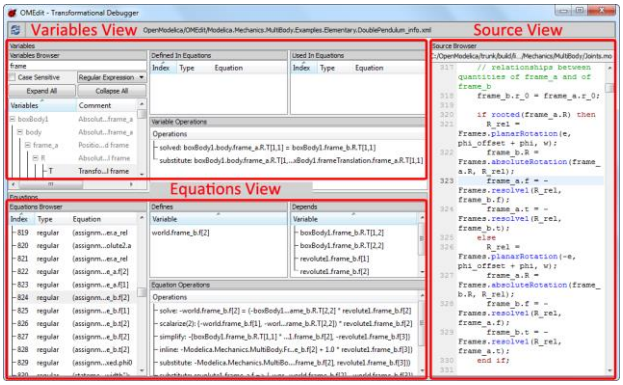

Figure 6. The OpenModelica equation model debugger. *Left*: equations view where equations and symbolic transformations can be vied. *Right*: source view where the erroneous equation is pointed out.

5.6 Run-time Solver Module and DAEMode

The OpenModelica execution and run-time solver module executes compiled binary code from translated models and functions. It includes numerical solvers as well as event handling facilities for the discrete and hybrid parts of the Modelica language.

A recent extension of this module is the DAEMode used for solving very large models. This is part of an emerging trend in Modelica tools of handling large-scale models, with hundreds of thousands or possibly millions of equations, (Casella, 2015). OpenModelica has pioneered this field by introducing sparse solvers in the solution chain: KLU for linear algebraic equations, Kinsol for nonlinear algebraic equations, and IDA for causalized differential equations. It also introduced the direct use of IDA as differential-algebraic equation solver, skipping the traditional causalization step, which is computationally more efficient for certain classes of systems. The largest system handled so far is an electro-mechanical power system model with about 600.000 differential-algebraic equations, (Braun et al, 2017).

5.7 OMNotebook and DrModelica

OMNotebook (Figure 7) (Fernström et al, 2006) is a book-like interactive user interface to OpenModelica primarily intended for teaching and course material. It supports sections and subsections to any level, hiding and showing sections and cells, interactive evaluation and simulation of Modelica models and plotting results. The DrModelica (Lengquist-Sandelin, 2003) interactive Modelica teaching course was the first main application, at that time based on Mathematica notebooks.

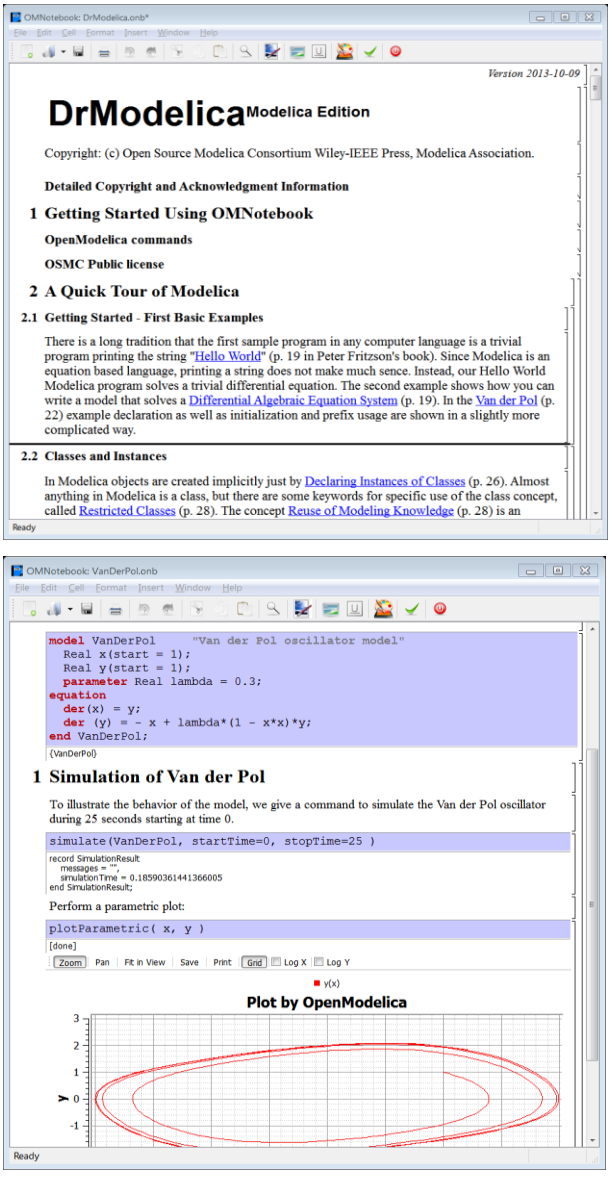

Figure 7. The OMNotebook electronic notebook showing part of the DrModelica document (course-material) for learning Modelica. *Top*: The DrModelica document start page. *Bottom*: The VanDerPol sub-document showing a cell with a Modelica model, simulation commands, and plot results.

5.8 OMWebbook – Interactive Web-based Editable and Executable Book

OMWebbook (Figure 8) (Moudgalya et al, 2017), (Fritzson et al, 2018), is an interactive web-based electronic book. This is similar to OMNotebook, but textual model editing and simulation is performed in a web-browser. Simulation is performed by a dedicated simulation server. Thus, the user need not install OpenModelica on a computer. Editing and simulation can even be done from smartphones or tablets.

DOI
10.3384/ECP18154206

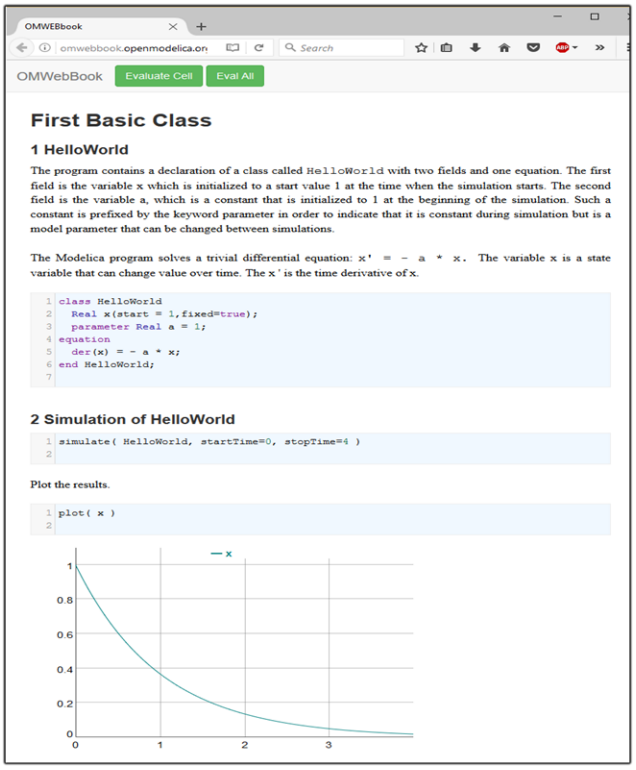

Figure 8. OMWebbook with editable models, simulations, and plots.

5.9 MDT Eclipse Plug-in

The MDT (Modelica Development Tooling) Eclipse plug-in (Figure 9) (Pop et al, 2006), (Pop 2008), is an Eclipse-based textual development environment for Modelica and MetaModelica model development.

It provides the usual facilities for software development such as browsing, building, cross referencing, syntax checking, and showing useful information such as types, function signatures, etc.

MDT is primarily used for development of medium to large scale Modelica projects, such as Modelica libraries written in standard Modelica and the OpenModelica compiler (currently containing more than 200 packages) written in MetaModelica.

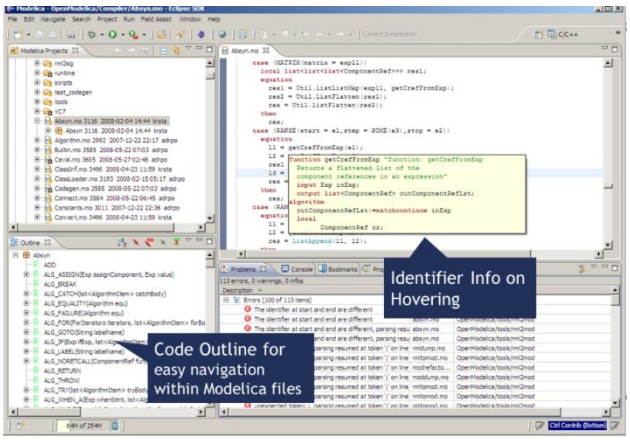

Figure 9. The OpenModelica MDT (Modelica Development Tooling) Eclipse plug-in.

5.10 Python, Julia, and Matlab Scripting

Scripting APIs to OpenModelica is also provided for the languages Python (Python 2018), Julia (Julia org, 2018), and Matlab (MathWorks 2018), through the subsystems OMPython (Lie et al, 2016), OMJulia and OMMatlab (OSMC 2018a). This gives the user the possibility to use Modelica together with the rich set of facilities and libraries in these languages, e.g. for tasks such as control design and post processing of simulation results.

5.11 Spoken Tutorials for OpenModelica

A number of interactive audio-video based spoken tutorials (www.spoken-tutorial.org) have been developed which provide step-by-step teaching about how to use OpenModelica and develop simple models. (Moudgalya et al, 2017). They are not part of the OpenModelica installer and system, but mentioned here since they provide important functionality to learn OpenModelica and Modelica.

5.12 OMSimulator – FMI-based Simulation and Composite Model Editor

OMSimulator, version 2.0, is an OpenModelica subsystem that provides efficient simulation and co-simulation of models in the (pre-compiled) FMI standard FMU (Functional Mockup Unit) form. Thus, models from non-Modelica tools compiled into FMUs can also be included and simulated. Furthermore, models that cannot be exported as FMUs can be integrated in a simulation using tool-to-tool co-simulation. This is provided via wrappers to models in tools such as ADAMS, Beast, Simulink, Hopsan, or co-simulation of FMUs with embedded solvers. The system can optionally be used with TLM (Transmission Line Modeling) connectors, which give numerically more stable co-simulation.

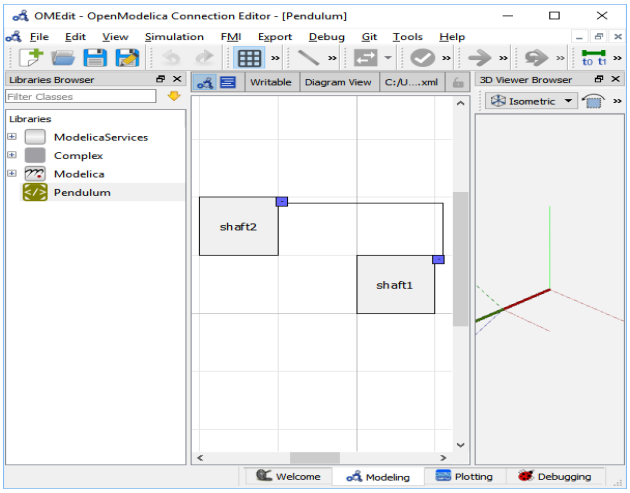

Figure 10. The OpenModelica OMSimulator composite model editor including 3D animation.

The earlier version OMSimulator 1.0, was already available in OpenModelica 1.12.0 released 2017

(Fritzson, Braun, and Hartford, 2018), (OSMC 2018b), but in that version TLM-connectors were mandatory and pure FMI model-exchange based simulation was missing.

Moreover, OMSimulator contains a composite model editor integrated in OMEdit, that allows combining external models (FMUs, both model-exchanged and co-simulated ones) into new composite models.

This editor, an extension of OMEdit (Figure 10), also provides 3D visualization of connected mechanical model components which can be FMUs, Modelica models, etc., or co-simulated components. 3D animation of simulated FMUs is possible (right part of Figure 10). A composite model is saved as an XML file according to the SSP (Systems and Structure Parameterization) standard (Modelica Assocaition 2018), (OSMC 2018c).

5.13 Embedded System Support

OpenModelica provides code generation of real-time controllers from Modelica models, for small foot-print platforms such as Arduino boards. (Berger et al, 2017), or in tools for RexRoth PLCs (Menager et al, 2014).

One example of code generation to small targets is the Single board heating system (Figure 11) from IIT Bombay (Arora, Kannan Moudgalya, and Malewar, 2010). It is used for teaching basic control theory, and usually controlled by a serial port (set fan value, read temperature, etc). OpenModelica can generate code targeting the ATmega16 on the board.

The program size is 4090 bytes including LCD driver and PID-controller (out of 16 kB flash memory available). The ATmega16 we target has 1 kB SRAM available for data (stack, heap, and global variables). In this case, only 130 bytes is used for data variables.

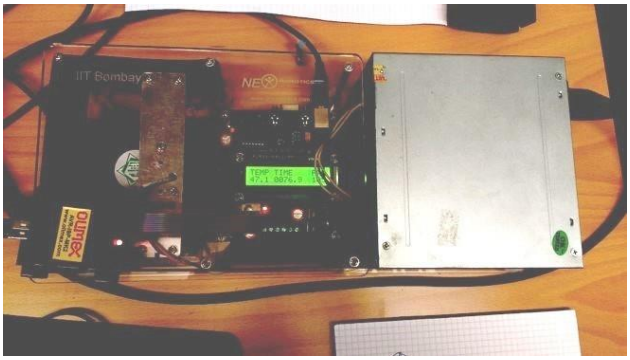

Figure 11. The SBHS (Single Board Heating System), an example embedded target system for OpenModelica.

To simplify interfacing of low-level devices from Modelica, OpenModelica supports the Modelica_DeviceDrivers library (Thiele, Beutlich, Waurich, Sjölund, and Bellmann, 2017), which is a free library for interfacing hardware drivers that is developed primarily for interactive real-time simulations. It is cross-platform (Windows and Linux). Using this library, modeling, parameterization and configuration can be done at a high level of abstraction using Modelica, avoiding the need for low level C programming.

5.14 Model-based Control, Synchronous Modelica, and C++ Run-time

OpenModelica is one of the (currently) two Modelica tools that support synchronous Modelica (Modelica Association, 2017), implemented both on top of the OpenModelica C run-time and C++ run-time. This can be used for model-based control, using Modelica and FMI, (Franke et al, 2017), and using the OpenModelica C++ run-time (Franke et al, 2015).

5.15 ModelicaML Modelica-UML Profile and Eclipse Plug-in

ModelicaML (Figure 12), (Schamai, 2013), (Schamai et al, 2014) is an Eclipse plug-in and Modelica-UML profile for the description of system architecture and system dynamic behavior. It is based on an extended subset of the OMG Unified Modeling Language (UML) as well as Modelica, and is designed for Modelica code generation from graphical models such as state machines and activity diagrams, supporting hardware/software co-modeling and system requirement verification against selected scenarios. The current prototype has not been updated recently and only works together with an old version of Eclipse.

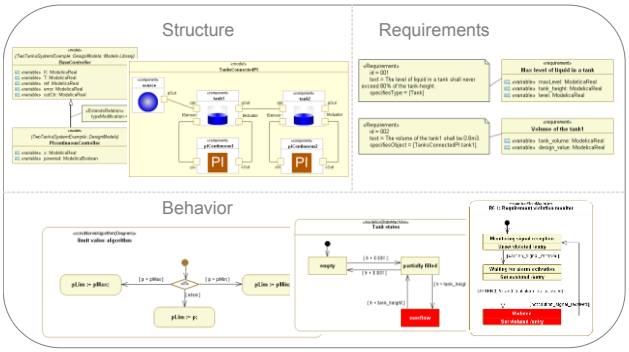

Figure 12. The ModelicaML Eclipse plug-in and UML-Modelica profile for integrated software-hardware modeling and requirements verification.

5.16 Requirement Verification

OpenModelica supports requirement verification using the vVDR approach (virtual Verification of Designs vs. Requirements), (Schamai, 2014<, Schamai et al, 2015). It was first introduced in the ModelicaML Eclipse plug-in mentioned previously, using a combination of UML and Modelica for requirement specification. Recently, a Modelica-only version of vVDR has been designed and implemented in OpenModelica, using requirement specification in Modelica, and a vVDR Modelica library (Buffoni et al, 2014; Buffoni et al, 2017).

It is a simulation-based approach that can be used to verify (Figure 13) different design alternatives against sets of requirements using different scenarios. The tool

DOI
10.3384/ECP18154206

automatically generates verification models in Modelica, performs the simulations, compares the results, and generates a report about verification results.

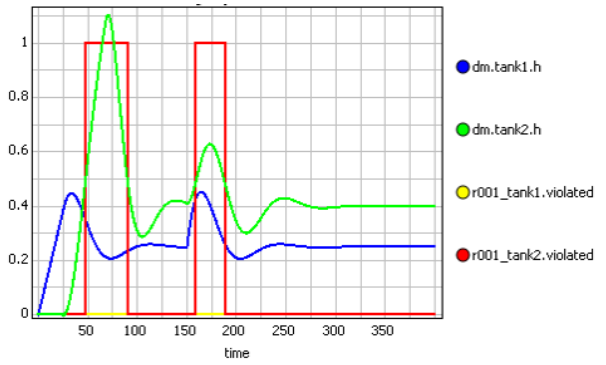

Figure 13. Simulation-based requirement verification for the two-tanks example. Requirement r001 regarding the level of tank2 is violated twice (shown in red).

5.17 Design Optimization

Two forms of design optimization tool support are available with OpenModelica: a) the traditional parameter sweep static design optimization using many simulation runs, or b) direct dynamic optimization of a full trajectory. The first method is supported by the OMOptim tool (Figure 14), (Thieriot et al, 2011).

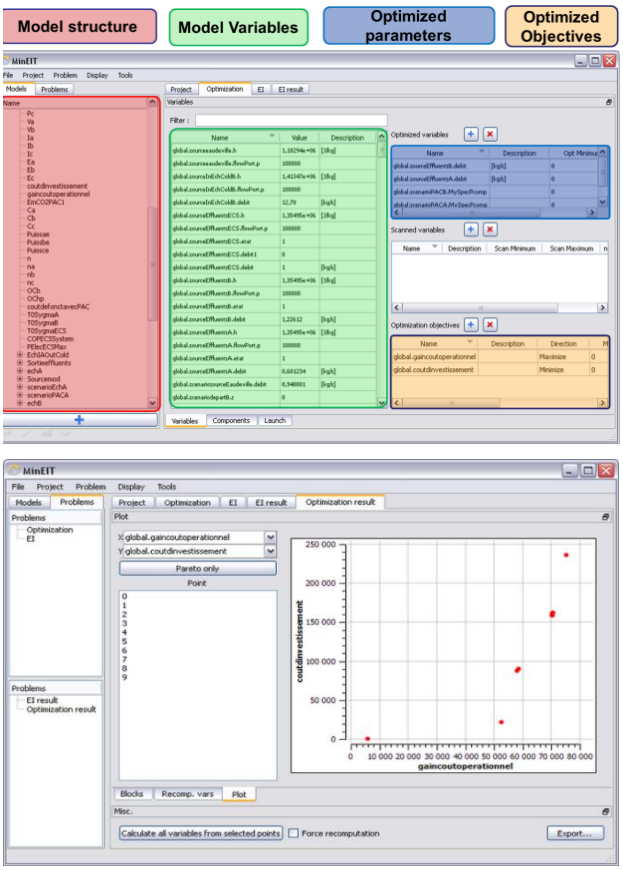

Figure 14. The OpenModelica OMOptim tool for parameter sweep optimization. *Top*: selecting variables, objectives, parameters. *Bottom*: A result plot with a Pareto optimization of two goal functions.

The second approach, dynamic optimization (Figure 15), (Bachmann et al, 2012), (Åkesson, 2008), formulates an optimization problem directly on a whole trajectory which is divided into trajectory segments (Figure 15, top) whose shapes are determined by coefficients which are initially not determined.

During the optimization process these coefficients are gradually assigned values which make the trajectory segments adjust shape and join into a single trajectory with a shape that optimizes the goal function under the constraints of fulfilling the model equations. Figure 15 (bottom) shows the relative speedup of performing dynamic optimization of a goal function for a small BatchReactor model using parallel versions of the dynamic optimization methods multiple shooting and multiple collocation running on a multi-core computer. Optimization algorithms from the Ipopt library (Wächter and Biegler, 2006), are employed for part of the optimization mechanism.

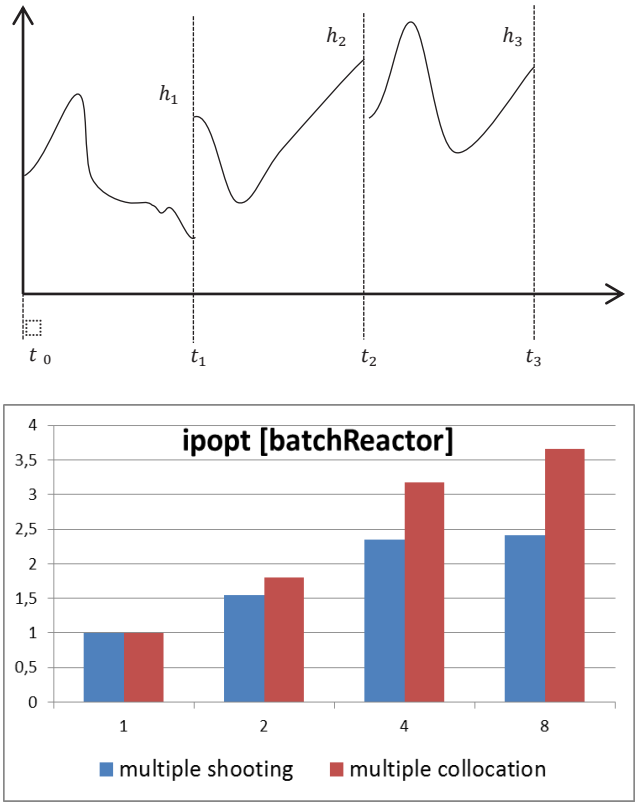

Figure 15. *Top*: Dynamic optimization formulates the whole trajectory in terms of trajectory segments whose shapes are adjusted during optimization. *Bottom*: Relative speedups and computation times of the complete dynamic optimization process for the BatchReactor example model using parallel multiple shooting or multiple collocation in OpenModelica on 1, 2, 4, and 8 cores.

5.18 Parallelization and Multi-Core

Work on generating parallel code from Modelica models has been ongoing for OpenModelica during several years. Automatic extraction of task parallelism and automatic scheduling is one approach that has been

investigated (Figure 16), (Aronsson 2006), (Walther et al, 2014). Another approach is the ParModelica language extension (Gebremedhin 2011) that allows generation of OpenCL code for data-parallel platforms such as the NVIDIA graphics cards. A third approach, which is now integrated with the above approaches is dynamic load balancing partly based on the running simulation. (Gebremedhin and Fritzson, 2017).

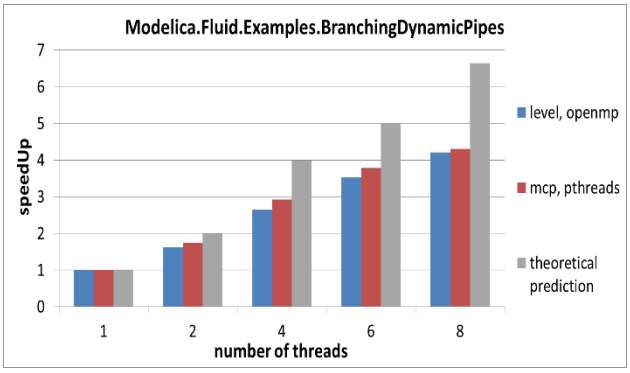

Figure 16. Example of speedup of parallel code from OpenModelica for the Fluid.Examples.Branching. DynamicPipes model.

6 Related Work

Since OpenModelica is a Modelica environment it has of course been influenced by other Modelica tools. The most influential of these tools is Dymola (Elmqvist et al, 1996), (Brück et al, 2002), (Dassault Systèmes 2013), which was the first full-scale industrial-strength Modelica environment. Certain aspects have also been influenced by the MathModelica environment (Fritzson 2006), later renamed and further developed to Wolfram System Modeler (Wolfram Research 2018), InterLisp, Mathematica (Wolfram 2003), and ObjectMath (Fritzson, et al, 1995) have influenced the design of OpenModelica as an integrated symbolic-numeric environment. Recently, the rapidly developing symbolic-numeric Julia language (Bezanson 2017), (Julia org, 2018) has appeared, with similar goals as MetaModelica.

7 Conclusions

OpenModelica has been developed into a powerful open source tool suite for modeling, simulation, and model-based development. Still some challenges are being worked on and remain to be addressed, for example very large models with several million equations. The debugger can be further improved to find additional kinds of numeric/symbolic errors. Integration aspects between tool functionalities can be further enhanced. Just-in-time compilation would improve the system's interactive properties. Two large recent OpenModelica efforts are the OMC new frontend development for 100% coverage and greatly enhanced compilation speed, and the OMSimulator tool for efficient large-scale FMI-based simulation. Recently OMJulia has been introduced that provides OpenModelica access from Julia. More powerful integration options between Julia and OpenModelica are also being considered in order to benefit from the Julia libraries and infrastructure.

Acknowledgements

This work has been supported by Vinnova in the ITEA OPENPROD, MODRIO, and OPENCPS projects and in the Vinnova RTISIM project. Support from the Swedish Government has been received from the ELLIIT project. The OpenModelica development is supported by the Open Source Modelica Consortium. Many students, researchers, engineers have contributed to the OpenModelica system. There is not room here to mention all these people, but we gratefully acknowledge their contributions.

References

Peter Aronsson. Automatic Parallelization of Equation-Based Simulation Programs. Linköping Studies in Science and Technology, Ph.D. Thesis No. 1022. June 14, 2006. URN: urn:nbn:se:liu:diva-7446

Adeel Asghar, Sonia Tariq, Mohsen Torabzadeh-Tari, Peter Fritzson, Adrian Pop, Martin Sjölund, Parham Vasaiely, and Wladimir Schamai. An Open Source Modelica Graphic Editor Integrated with Electronic Notebooks and Interactive Simulation. In *Proc. of the 8th International Modelica Conference* 2011, pp. 739–747. Modelica Association, March 2011.Linköping University, Sweden, 2010.

Inderpreet Arora, Kannan Moudgalya, Sachitanand Malewar. A low cost, open source, single board heater system. In *Proc. 4th IEEE International Conference on E-Learning in Industrial Electronics* (ICELIE), Nov 7-10, 2010. IEEE Xplore, DOI: 10.1109/ICELIE.2010.5669868

Bernhard Bachmann, Lennart Ochel, Vitalij Ruge, Mahder Gebremedhin, Peter Fritzson, Vaheed Nezhadali, Lars Eriksson, Martin Sivertsson. Parallel Multiple-Shooting and Collocation Optimization with OpenModelica. In *Proceedings of the 9th International Modelica Conference* (Modelica'2012), Munich, Germany, Sept.3-5, 2012

Lutz Berger, Martin Sjölund, Bernhard Thiele. Code generation for STM32F4 boards with Modelica device drivers. In *Proc.of 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Munich, Germany, Dec 1, 2017. Published by ACM Digital Library. doi:10.1145/3158191.3158204

Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59: 65–98. 2017 doi: 10.1137/141000671.

Willi Braun, Francesco Casella, and Bernhard Bachmann Solving Large-scale Modelica Models: New Approaches and Experimental Results using OpenModelica, In *Proc 12th Int. Modelica Conference,* May 15-17, 2017, Prague, Czech Republic, pp. 557-563, doi:10.3384/ecp17132557

Dag Brück, Hilding Elmqvist, Sven-Erik Mattsson, and Hans Olsson. Dymola for Multi-Engineering Modeling and Simulation. In *Proceedings of the 2nd International*

Modelica Conference, Oberpfaffenhofen, Germany, Mar. 18–19, 2002

Lena Buffoni and Peter Fritzson. Expressing Requirements in Modelica. In *Proceedings of the 55th Scandinavian Conference on Simulation and Modeling* (SIMS'2014), available at www.scan-sims.org. Aalborg, Denmark, Oct 21-22, 2014.

Lena Buffoni, Adrian Pop, and Alachew Mengist. Traceability and Impact Analysis in Requirement Verification. In *Proc.of 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Munich, Germany, Dec 1, 2017. Published by ACM Digital Library. doi:10.1145/3158191.3158207

Francesco Casella. Simulation of Large-Scale Models in Modelica: State of the Art and Future Perspectives. In *Proceedings of the 11th International Modelica Conference*, Sept 21-23 2015, Versailles, France, pp. 459-468, doi:10.3384/ecp15118459

Alejandro Danós, Willi Braun, Peter Fritzson, Adrian Pop, Hugo Scolnik, and Rodrigo Castro. Towards an OpenModelica-based Sensitivity Analysis Platform Including Optimization-driven Strategies. In *Proc.of 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Munich, Germany, Dec 1, 2017. Published by ACM Digital Library. doi:10.1145/3158191.3158206

Hilding Elmqvist, Dag Bruck, and Martin Otter. *Dymola—User's Manual*. Dynasim AB, Research Park Ideon, SE-223 70, Lund, Sweden, 1996

Dassault Systèmes. *Dymola. Systems Engineering Overview*. https://www.3ds.com/products-services/catia/products/dymola/ Accessed Sept. 3, 2018.

Anders Fernström, Ingemar Axelsson, Peter Fritzson, Anders Sandholm, Adrian Pop. OMNotebook – Interactive WYSIWYG Book Software for Teaching Programming. In *Proc. of the Workshop on Developing Computer Science Education – How Can It Be Done*?. Linköping University, Dept. Computer & Inf. Science, Linköping, Sweden, March 10, 2006.

Rüdiger Franke, Marcus Walther, Niklas Worschech, Willi Braun, and Bernhard Bachmann. Model-based Control with FMI and a C++ Runtime for Modelica. In *Proceedings of the 11th International Modelica Conference*, Versailles, France, September 21-23, 2015. Published by LIU Electronic Press. doi:10.3384/ecp15118339

Rüdiger Franke, Sven Erik Mattsson, Martin Otter, Karl Wernersson, Hans Olsson, Lennart Ochel, and Torsten Blochwitz. Discrete-time Models for Control Applications with FMI. In *Proceedings of the 12th International Modelica Conference*, Prague, Czech Republic, May 15-17, 2017. Published by LIU Electronic Press. doi: 10.3384/ecp17132507

Peter Fritzson, Lars Viklund, Dag Fritzson, and Johan Herber. High Level Mathematical Modeling and Programming in Scientific Computing, IEEE Software, pp 77–87, July 1995.

Peter Fritzson, Peter Aronsson, Håkan Lundvall, Kaj Nyström, Adrian Pop, Levon Saldamli, and David Broman. The OpenModelica Modeling, Simulation, and Software Development Environment. In *Simulation News Europe*, 44/45, December 2005. See also: http://www.openmodelica.org. An earlier version in *Proceedings of the 46th Conference on Simulation and Modelling of the Scandinavian Simulation Society* (SIMS2005), Trondheim, Norway, October 13-14, 2005.

Peter Fritzson. MathModelica - An Object Oriented Mathematical Modeling and Simulation Environment. *Mathematica Journal*, Vol 10, Issue 1. February. 2006.

Peter Fritzson, Adrian Pop, David Broman, Peter Aronsson. Formal Semantics Based Translator Generation and Tool Development in Practice. In *Proc. of the 20th Australian Software Engineering Conference* (ASWEC 2009), Gold Coast, Queensland, Australia, April 14 – 17, 2009.

Peter Fritzson, Adrian Pop, and Martin Sjölund. *Towards Modelica 4 Meta-Programming and Language Modeling with MetaModelica 2.0*. Technical reports in Computer and Information Science, No 10, Linköping University Electronic Press. February 2011. URL http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-68361

Peter Fritzson. *Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*. 1250 pages. ISBN 9781-118-859124, Wiley IEEE Press, 2014.

Peter Fritzson, Bernhard Bachmann, Kannan Moudgalya, Francesco Casella, Bernt Lie, Jiri Kofranek, Massimo Ceraolo, Christoph Nytsch Geusen, Luigi Vanfretti, (editors). *Introduction to Modelica with Examples in Modeling, Technology, and Applications*. Published by Linköping University Electronic Press, series "Linköping University Interdisciplinary Studies" with ISSN 1650-9625. On-line: http://omwebbook.openmodelica.org/ Accessed Sept 3, 2018.

Dag Fritzson, Robert Braun, and Jan Hartford. Composite modelling in 3-D mechanics utilizing Transmission Line Modelling (TLM) and Functional Mock-up Interface (FMI) *Modeling, Identication and Control*, Vol. 39, No. 3, pp. 179-190, 2018.

Peter Fritzson, Bernhard Bachmann, Kannan Moudgalya, Francesco Casella, Bernt Lie, Jiri Kofranek, Massimo Ceraolo, Christoph Nytsch Geusen, Luigi Vanfretti, (editors). *Introduction to Modelica with Examples in Modeling, Technology, and Applications* Published by Linköping University Electronic Press, series "Linköping University Interdisciplinary Studies" with ISSN 1650-9625. On-line: http://omwebbook.openmodelica.org/ Accessed Sept 3, 2018.

Mahder Gebremedhin. *ParModelica: Extending the Algorithmic Subset of Modelica with Explicit Parallel LanguageConstructs for Multi-core Simulation*. Master Thesis, Department of Computer and Information Science, Linköping University, Oct. 2011. URN: urn:nbn:se:liu:diva-71612

Mahder Gebremedhin and Peter Fritzson. Parallelizing Simulations with Runtime Profiling and Scheduling, In *Proc..of 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, Munich, Germany, 2017. Published by ACM.

Julia org. *Julia Language Documentation*, Release 1.0. Accessed August 31, 2018, www.julialang.org

Eva-Lena Lengquist-Sandelin, Susanna Monemar, Peter Fritzson, and Peter Bunus. DrModelica - An Interactive Tutoring Environment for Modelica. In *Proceedings of the*

3rd International Modelica Conference, Nov. 3-4, Linköping, Sweden, 2003

Bernt Lie, Sudeep Bajracharya, Alachew Mengist, Lena Buffoni, Arunkumar Palanisamy, Martin Sjölund, Adeel Asghar, Adrian Pop, Peter Fritzson. API for Accessing OpenModelica Models From Python. In *Proceedings of the 9th Eurosim Congress on Modelling and Simulation*, EuroSim2016, Oulu, Finland, September 12-16, 2016. Published by IEEE, ISBN 978-1-5090-4119-0, pp. 707--713; http://eurosim2016.

MathWorks. Matlab product overview. https://www.mathworks.com/products/matlab.html Accessed Sept 3, 2018.

Nils Menager, Niklas Worschech, and Lars Mikelsons. A Toolchain for Rapid Control Prototyping using Rexroth Controllers and Open Source Software. In *Proceedings of the 10th International Modelica Conference* (Modelica'2014), Lund, Sweden, March.10-12, 2014.

Robert Milner, Mads Tofte, Robert Harper, and David MacQueen, *The Definition of Standard ML - Revised*. MIT Press. ISBN: 0-262-63181-4. Year 1997

Modelica Association. *Modelica: A Unified Object-oriented Language for Physical Systems Modeling, Language Specification Version 3.4*. April 10, 2017. URL http://www.modelica.org/

Modelica Association. SSP – MA Project for System Structure and Parameterization of Components for Virtual System Design. https://www.modelica.org/projects Accessed Sept 3, 2018.

Kannan Moudgalya, Bhargava Nemmaru, Kaushik Datta, Priyam Nayak, Peter Fritzson, and Adrian Pop. Large Scale Training through Spoken Tutorials to Promote and use OpenModelica. . In *Proceedings of the 12th International Modelica Conference* (Modelica'2017), Prague, Czech Republic, May, 15-17, 2017.

OCaml org. OCaml web site. https://ocaml.org/ Accessed Sept 3, 2018.

OSMC. *OpenModelica Users Guide*, latest version. https://www.openmodelica.org/doc/OpenModelicaUsersGuide/latest/ Accessed September 3, 2018a.

OSMC. OMSimulator 1.0 documentation. Chapter 6 in https://www.openmodelica.org/doc/OpenModelicaUsersGuide/OpenModelicaUsersGuide-v1.12.0.pdf Accessed September 3, 2018b.

OSMC. OMSimulator 2.0 documentation: https://openmodelica.org/doc/OMSimulator/html/ Accessed September 3, 2018c.

Adrian Pop, Peter Fritzson, Andreas Remar, Elmir Jagudin, and David Akhvlediani. OpenModelica Development Environment with Eclipse Integration for Browsing, Modeling, and Debugging. In *Proceedings of the 5th International Modelica Conference* (Modelica'2006), Vienna, Austria, Sept. 4-5, 2006.

Adrian Pop and Peter Fritzson, MetaModelica: A Unified Equation-Based Semantical and Mathematical Modeling Language. In D. Lightfoot and C. Szyperski, editors, *Modular Programming Languages*, Vol. 4228 of Lecture Notes in Computer Science, pages 211{229. Springer Berlin / Heidelberg, 2006.DOI:10.1007/11860990 14.

Adrian Pop. *Integrated Model-Driven Development Environments for Equation-Based Object-Oriented Languages*. Ph.D. Thesis. Linköping Studies in Science and Technology, Dissertation No. 1183, June 5, 2008.

Adrian Pop and Peter Fritzson. MetaModelica: A Unified Equation-Based Semantical and Mathematical Modeling Language. In *Proceedings of Joint Modular Languages Conference* 2006 (JMLC2006) LNCS 4228, Springer Verlag. Jesus College, Oxford, England, Sept 13-15, 2006.

Adrian Pop, Martin Sjölund, Adeel Asghar, Peter Fritzson, Francesco Casella. Integrated Debugging of Modelica Models. *Modeling, Identication, and Control*, Vol 35, No 2, pp. 93-107, DOI: http://dx.doi.org/10.4173/mic.2014.2.3, ISSN 1890-1328, Aug 2014.

Python Software Foundation. Python Programming Language web page. https://www.python.org/ Accessed Sept 3, 2018.

Wladimir Schamai. *Model-Based Verification of Dynamic System Behavior against Requirements - Method, Language, and Tool*. Linköping Studies in Science and Technology, Dissertation No. 1547, Nov 12, 2013. DOI: 10.3384/diss.diva-98107

Wladimir Schamai, Lena Buffoni, Peter Fritzson. An Approach to Automated Model Composition Illustrated in the Context of Design Verification. *Modeling, Identification and Control*, Vol. 35, No. 2, pp. 79-91, ISSN 1890-1328, Aug. 2014

Wladimir Schamai, Lena Buffoni, Nicolas Albarello, Pablo Fontes De Miranda, and Peter Fritzson. An Aeronautic Case Study for Requirement Formalization and Automated Model Composition in Modelica. In *Proceedings of the 11th International Modelica Conference* (Modelica'2015), Paris, France, September, 21-23, 2015

Martin Sjölund, Peter Fritzson, and Adrian Pop. Bootstrapping a Compiler for an Equation-Based Object-Oriented Language. DOI: 10.4173/mic.2014.1.1. *Modeling, Identification and Control*, Vol 35, No 1, pp 1-19, 2014.

Martin Sjölund. *Tools and Methods for Analysis, Debugging, and Performance Improvement of Equation-Based Models*. Ph.D. Thesis. Linköping Studies in Science and Technology, Dissertation No. 1664, June 1, 2015.

Bernhard Thiele, Thomas Beutlich, Volker Waurich, Martin Sjölund, and Tobias Bellmann. Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library. In *Proc. of the 12th Int. Modelica Conference*, Prague, Czech Republic, May 2017.

Hubert Thieriot, Maroun Nemer, Mohsen Torabzadeh-Tari, Peter Fritzson, Rajiv Singh, and John John Kocherry. Towards Design Optimization with OpenModelica Emphasizing Parameter Optimization with Genetic Algorithms. In *Proceedings of the 8th International Modelica Conference* (Modelica'2011), Dresden, Germany, March.20-22, 2011.

Marcus Walther, Volker Waurich, Christian Schubert, and Ines Gubsch. Equation based parallelization of Modelica models. In *Proceedings of the 10th International Modelica Conference* (Modelica'2014), Lund, Sweden, March.10-12, 2014.

Volker Waurich and Jürgen Weber. Interactive FMU-Based Visualization for an Early Design Experience. In *Proc. of*

the 12th Int. Modelica Conference, Prague, Czech Republic, May 2017.

Stephen Wolfram. *The Mathematica Book*, 5th Ed. Wolfram Media, Inc, 2003.

Wolfram Research. Wolfram System Modeler Documentation and Overview. http://www.wolfram.com/system-modeler/ Accessed September 3, 2018.

Andreas Wächter and Lorenz. Biegler, On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, *Mathematical Programming* 106 (2006) 25-57. Also: (Ipopt) https://projects.coin-or.org/Ipopt

Johan Åkesson. Optimica—An Extension of Modelica Supporting Dynamic Optimization. In *Proc. of 6th International Modelica Conference* 2008. Modelica. Association, March 2008