*Proceedings of the 2021 Winter Simulation Conference*
*S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.*

# THE OPENMODELICA ENVIRONMENT FOR BUILDING DIGITAL TWINS OF SUSTAINABLE CYBER-PHYSICAL SYSTEMS

Peter Fritzson

Department of Computer and Information Science
Linköping University
Linköping, SE-58183, SWEDEN

## ABSTRACT

The Modelica modeling language and technology is being warmly received by the world community in modeling and simulation with major applications in virtual prototyping and digital twins of complex cyber-physical systems, which mix physical system dynamics with software (cyber) and networks. It is enabling a revolution in this area, based on its ease of use, visual design of models with combination of lego-like predefined model building blocks, its ability to define model libraries with reusable components, its support for modeling and simulation of complex applications involving parts from several application domains, and many more useful facilities. Adoption is further strengthened by the freely available open source OpenModelica environment for building digital twins and virtual prototypes as well as system analysis and optimization, especially relevant in transforming society into sustainability including applications in renewable energy and fossil-free transportation. This paper gives an overview of this technology as well as some applications.

## 1 INTRODUCTION

The human society on planet earth is entering a critical era. The environmental degradation in our current systems of production and consumption has reached critical levels. The effects from pollution including CO2 emissions causing global warming are becoming increasingly evident. Continuing on this path is an incredibly high risk as it could trigger non-linear, abrupt environmental change within planetary systems.

A transformative change of our society into sustainability is needed. This includes quick phasing out of fossil fuel solutions, increasing the use of renewable energy from clean sources such as solar and wind, switching from fossil-fueled vehicles to emission free electric transportation. Moreover, a circular economy needs to be developed, where material can be re-used instead of being turned into waste as in our current linear economy.

All this requires and thorough understanding and control of the complex systems involved as well as the transformative changes needed. In this endeavor, virtual prototypes and digital twins have an important role to play. They can be used to model complex systems, from cars to cities to human bodies, and simulate their functioning with an accuracy that allows the user to go directly from a virtual model to creation, without spending the years it normally takes to prototype and incrementally improve on existing designs.

What kind of technology is powerful enough to model complex systems including different kinds of hardware as well as software? We believe that the Modelica modeling technology together with the freely available OpenModelica tool environment is an important part of the answer to this question.

To give some idea of what we are discussing, below we show two concrete small examples of systems modeled as Modelica graphical models. The first (Figure 1) is a model of a small electric grid with some solar PV electric power, electric car charging, and house household electricity usage (Campillo et al 2015).

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

The second (Figure 2) is a sketch of a solar district heating system with thermal storage, complemented by some wind power and solar PV power.

Several Modelica libraries of pre-defined model components are available to help modeling such applications. Regarding electrical grids, for example, the open source PowerGrids library (Bartolini et al 2019) can be used. Low-temperature solar thermal systems modeling is described in (Hernandez-Albaladejo et al 2018), district heating applications can be modeled using the DistrictHeating library (Giraud et al 2015), and concentrated solar thermal power systems using the SolarTherm library (de la Calle et al 2018).

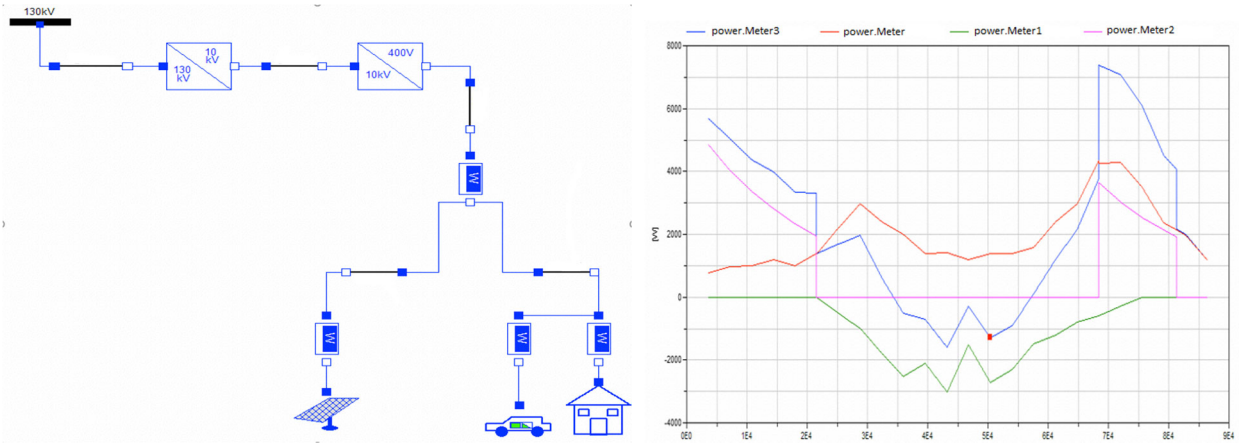



Figure 1: *Left*. Small Modelica grid model with some solar PV power, electric car charging, and house household electricity usage. *Right*. Simulation results. (Courtesy Javier Campillo).

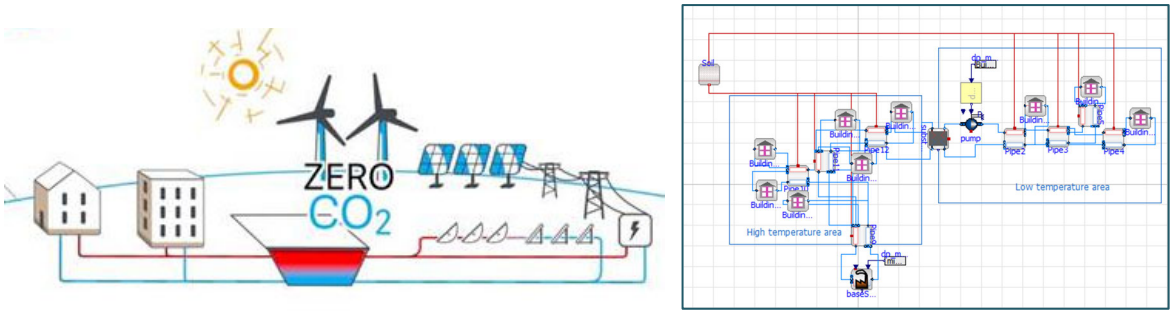



Figure 2: *Left*. Sketch of solar district heating system with thermal storage, wind power, and solar PV power. *Right*: small district heating model using the Modelica DistrictHeating library.

## 1.1 Modeling, Simulation, and Digital Twins

Before going into more details about the Modelica technology it is useful define some basic concepts. The concept of digital twin is closely related to modeling and simulation. In (Fritzson 2014) we have the following definitions:

- A *model* of a system is anything an "experiment" can be applied to in order to answer questions about that system.
- A *simulation* is an experiment performed on a model.

Artifacts represented by mathematical models in a computer are often called *virtual prototypes*. The process of constructing and investigating such models is virtual prototyping. Many people view a digital twin as a virtual prototype represented digitally. It is a virtual model that can be created in a computer, simulated,

analyzed, and tuned before building a physical counterpart. This is typical for model-based development of industrial products, and probably the most common application of the concept. Another interpretation is that the digital twin should interact in real-time with the physical world, as in the following definition of *virtual/digital twin* (Verzelen et al 2021):

> "A virtual twin is a real time virtual representation of a product, process, or a whole system that is used to model, visualize, predict and provide feedback on properties and performance, and is based on an underlying digital thread."

Such real-time applications have been available within the Modelica eco-system for some time, e.g., including on-line optimization (Franke 2002; Franke 2003) and model-predictive control (Zoltan et al 2007)

Yet another interpretation is more AI-inspired (Barricelli et al 2019), viewing a digital twin as a living, intelligent, and evolving model, being a virtual counterpart of a physical entity, and following the lifecycle of its physical twin. There should be continuous synchronization and communication between the two twins. This view is for example relevant for long-running autonomous systems such as robots.

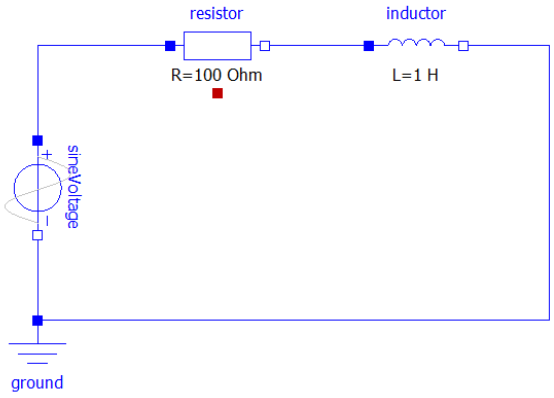
## 2 MODELICA LANGUAGE AND OPENMODELICA ENVIRONMENT

Modelica is an acausal equation-based modeling language for cyber-physical system modeling (Modelica Association 2021; Fritzson 2014; Fritzson et al 1998) standardized by Modelica Association. In Modelica, behavior is described declaratively using mathematical equations and functions. Object-oriented concepts are used to encapsulate behavior and facilitate reuse of model components. The acausal and object-oriented aspects of Modelica make it particularly well suited for code reuse through libraries. Modelica is superior to most other modeling tools and formalisms due to the following important properties:

- *Object-oriented modeling*. This technique makes it possible to create physically relevant and easy-to-use model components, which are employed to support hierarchical structuring, reuse, and evolution of large and complex models covering multiple technology domains.
- *Acausal modeling*. Modeling is based on equations instead of assignment statements as in traditional input/output block abstractions. Direct use of equations significantly increases reusability of model components, since components adapt to the data flow context in which they are used. This generalization enables both simpler models and more efficient simulation.
- *Physical modeling of multiple domains*. Model components can correspond to physical objects in the real world, in contrast to established techniques that require conversion to signal blocks. For application engineers, such "physical" components are particularly easy to combine into simulation models using a graphical editor.
- *Hybrid modeling*. Modeling of both continuous-time and discrete-time aspects of systems is supported in an integrated way. From Modelica 3.3, clocked discrete-time modeling is also supported for increased modeling precision and simulation performance.

A large set of Modelica libraries is available, under both free and commercial licenses. A few related to sustainable energy were mentioned previously in Section 1. The most important library is the Modelica Standard Library (MSL) (Modelica Association 2020). MSL version 4.0.0 released in 2020 contains about 1400 model components and 1200 functions from many domains.

The Modelica language support of both visual and textual views of the same model is demonstrated by the example in Figure 3. Since the visual view is defined by standardized graphical annotations, both the visual view and the textual view are preserved if the model is moved between Modelica tools and allowing both visual and textual model editing.




```
model RLCircuit
  Ground ground;
  Resistor resistor(R = 100);
  Inductor inductor(L = 1);
  SineVoltage sineVoltage;
equation
  connect(sineVoltage.p, resistor.p);
  connect(resistor.n, inductor.p) ;
  connect(inductor.n, ground.p);
  connect(sineVoltage.n, ground.p);
```

Figure 3: Graphical vs textual view of the same Modelica model. *Left*: A simple RL-circuit is modeled using Modelica graphical connection diagrams. *Right*: Textual view of RL-circuit Modelica model.

Cyber-physical modeling including multiple domains is illustrated in Figure 4. The model contains parts from three domains, two physical and one cyber: an electric part using components from the Modelica.Electrical library, a mechanical part using components from the Modelica.Mechanical.Rotational library, and a control (cyber) part using the Modelica.Blocks library.

Modeling in Modelica of both continuous-time and discrete-time aspects of systems is possible in an integrated way. See Section 3.1 for some more details regarding this topic. From Modelica language version 3.3 and later modeling using clocked discrete-time constructs is also supported for increased modeling precision and simulation performance. This is illustrated in Figure 5.

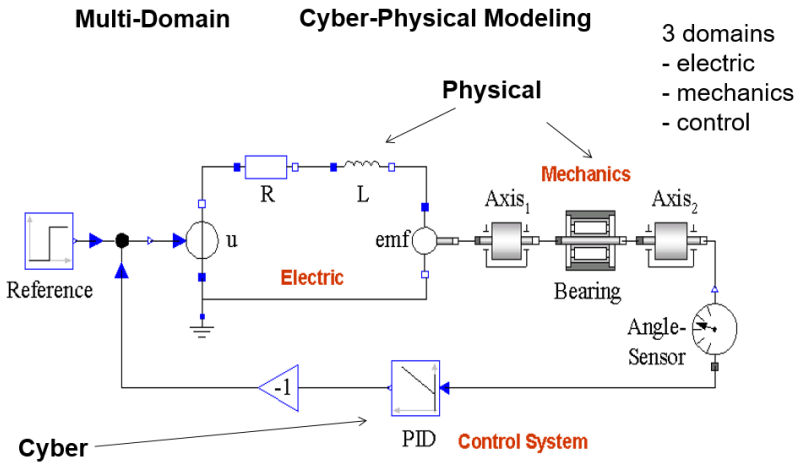



Figure 4: A simple example of a multi-domain cyber-physical model in Modelica. Two physical domains are present: electrical and mechanical, and one software (cyber) domain for the control system part.

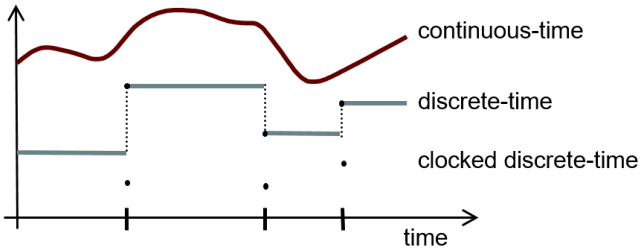



Figure 5: Illustration of hybrid modeling in Modelica, allowing combinations of continuous-time, discrete-time, and clocked discrete-time variables.

## 2.1 The OpenModelica Environment

OpenModelica (Fritzson et al 2020) is an open-source Modelica- and FMI-based modeling, simulation, optimization, model-based analysis, and development environment. It includes a number of facilities such as textual and graphical model editing, simulation, optimization and sensitivity analysis (Fritzson 2020), debugging (Pop et al 2014)), visualization and 3D animation (Section 2.2), requirement verification, web-based model editing and simulation, scripting from Python (Python Software Foundation 2018), Julia (Julialang 2018), Matlab (MathWorks 2018), and Modelica itself; efficient simulation and co-simulation of FMI-based (Section 3.3) models using its OMSimulator subsystem. There are also commercial proprietary Modelica tools, e.g., Dymola (Dassault Systemes 2018). A full list of Modelica tools is available (Modelica Association 2021).

The most important subsystems are the OpenModelica Compiler (OMC) and the OMEdit graphical connection editor and user interface for simulation, plotting, and debugging. OMC is implemented in MetaModelica, an extended version of Modelica that allows symbolic transformations to be specified and efficiently executed Fritzson et al 2019), which is useful e.g., for compilation purposes. Models are compiled to efficient C or C++ code. Experimental Java and C# code generators have also been developed.

Modelica models can be created and edited graphically, by dragging, dropping and connecting together existing model components from libraries, or textually using ordinary text editing.

Figure 6 illustrates the graphical user interface. To the left is the library browser, in the center is the model, shown graphically or textually, here a Chua Circuit. The upper right pane shows model documentation, and lower right pane displays the plot variable browser, to select which variables should be plotted.

Figure 7 shows OpenModelica simulating the Chua Circuit and plotting two variables, the C1.v and C2.v which are selected in the plot variable browser to the right.

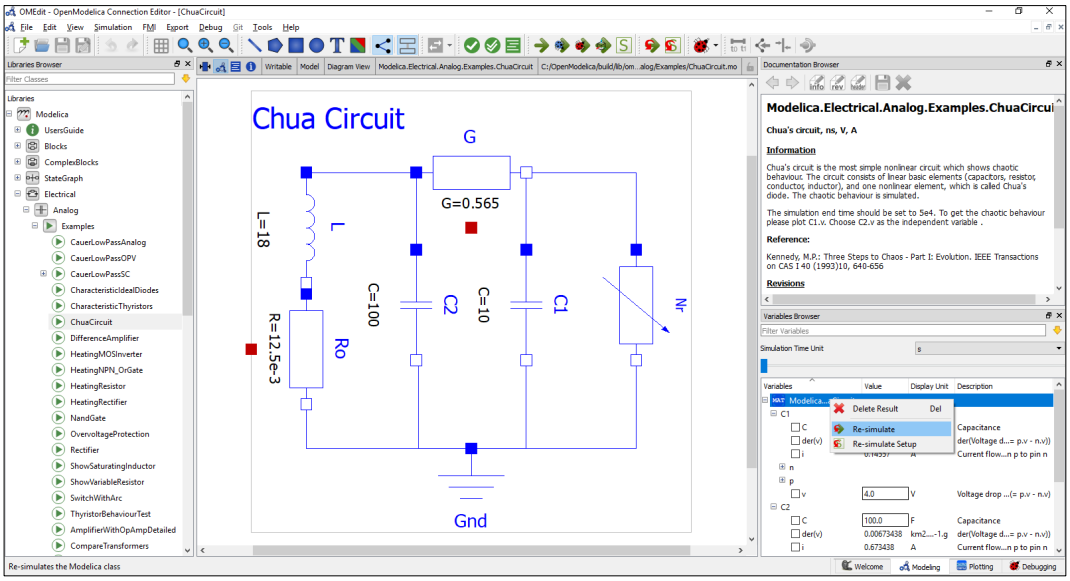



Figure 6: OpenModelica graphical editor OMEdit on a Chua Circuit Modelica model. *Upper right:* model information pane. *Lower right*: plot variable pane.

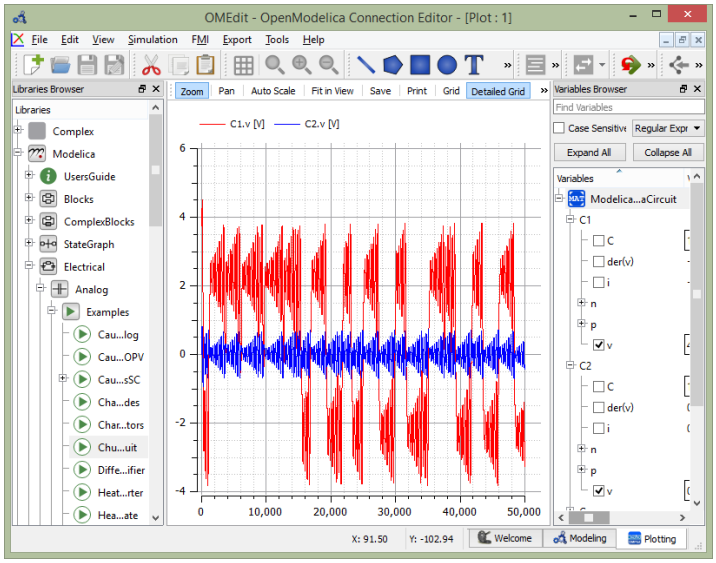



Figure 7: OpenModelica simulation and plotting of Chua Circuit.

## 2.2 3D Visualization

The Modelica language standard includes definitions standardized graphical annotations. Some of these annotations can be used to define 3D shapes of physical objects. There are standard annotations for a number of shapes such as cylinders, rods, etc. The OpenModelica 3D animation and visualization is based on 3D shapes defined by the Modelica Multi-Body library. It provides visualization of simulation results and animation of geometric primitives and CAD-files. OpenModelica generates a scene description XML-file which assigns model variables to visualization shape attributes. The scene description file can also be used to generate a visualization controlled by an FMU (Section 3.3) either in OMEdit, Figure 8, or in an external visualization tool as Unity 3D, Figure 8, (Waurich and Weber, 2017). In combination with the

Modelica_DeviceDrivers Library (Thiele et al. 2017), interactive simulations with visual feedback and 3D-interactions can be implemented for training, development and testing purposes.

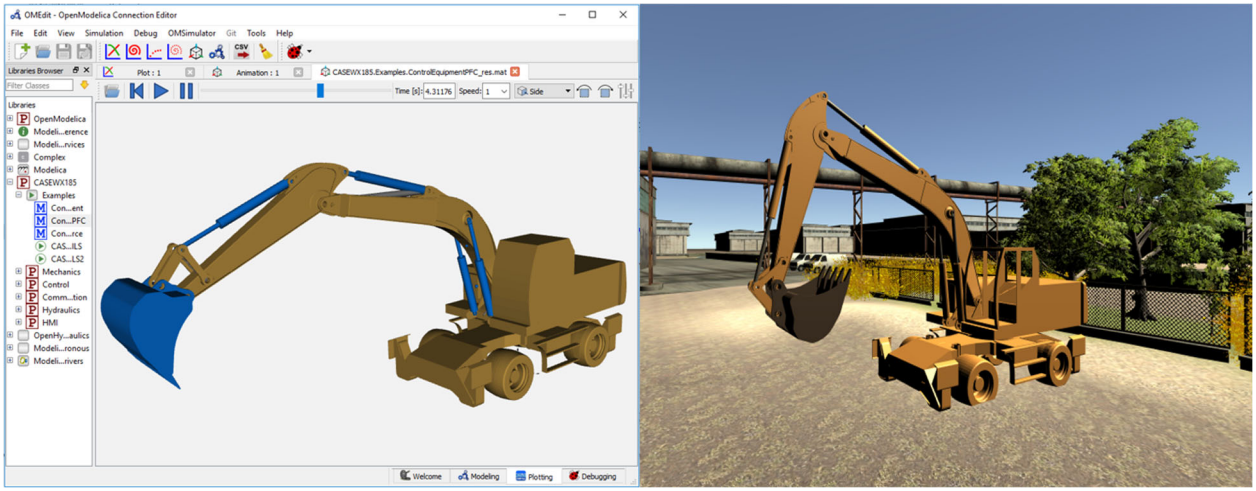


Figure 8: OpenModelica 3D animation of a simulated excavator in OMEdit and in unity 3D. (Courtesy Volker Waurich).

## 3 EMBEDDED REAL-TIME SYSTEMS

OpenModelica provides code generation of real-time controllers from Modelica models (Sjölund 2015), e.g., for small foot-print platforms such as Arduino boards 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 9) from IIT Bombay (Arora et al 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 compiled from a PID model in Modelica. 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 9: 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 et al. 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.

### 3.1 Event Handling

Real-time systems usually need to deal with events. How are events *created*? Events may occur naturally, created in the physical world external to the modeled system, or are created internally in the simulated computer model through various mechanisms, see Figure 10.

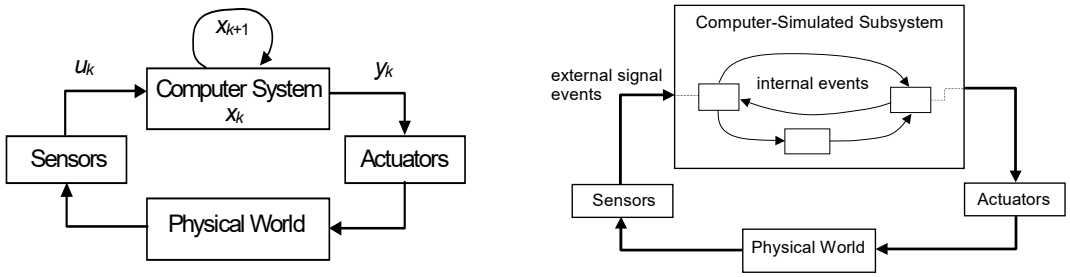



Figure 10: *Left*. External events are related to external input variables whereas internal events are related to internal model variables. *Right*: A sampled system where a computer obtains sampled inputs $u_k$ and controls the continuous physical world through the output signals $y_k$. The current discrete state is represented by $x_k$, and the computed state to be used at the next sample event is $x_{k+1}$.

A simple type of discrete-time Modelica model is the sampled data model often used in applications because of its simplicity and favorable analytical properties. A sampled model is characterized by its ability to periodically sample continuous input variables, calculate new outputs y that can influence the physical world as well as continuous parts of the model, and update discrete-time state variables x. Both the output variables y and the state variables x keep their values between the sample events since they are discrete-time variables. Such a sampled system model is schematically depicted in Figure 10. A sampled system model has only one kind of event, the sample event, and can be represented by the following state space equations:

$$x_{k+1} = f(x_k, u_k, t_k, t_{k+1})$$
$$y_k = h(x_k, u_k, t_k)$$

Here $x_k$ is a sequence of state vectors or scalars, $u_k$ is a sequence of sampled input vectors or scalars, and $y_k$ is a sequence of output vectors or scalars at points in time $t=t_0,t_1,...t_k,t_{k+1}...$ etc. In a real-time environment, periodic sampling is usually assumed with $t_k=kT$, $k=0,1,2,3,\ldots$where $T$ is the fixed sample interval. At any sample event occurring at time $t_k$, the model should compute $x_{k+1}$ and $y_k$ depending on $u_k$ and $x_k$. It is important that the input $u_k$ can propagate through the system without any time delay, since the input might be the output of another subsystem.

The following Modelica model is a simple periodic sampler with a sampling period T that is constant and defined as a parameter that can be changed by the model user. As we remarked previously, this model has only one kind of event, the sampling event. We use the built-in function sample in the when-condition sample(0,T) to periodically generate sampling events with a period time T. This is a simple model using state space equations:

```
model SimplePeriodicSampler
parameter Real T=1  "Sample period";
  input     Real u    "Input used at sample events";
  discrete output Real y  "Output computed at sample events";
```

```
protected
  discrete Real x;     // discrete-time state variable
equation
  when sample(0,T) then
    x = f(pre(x),u);   // state update expression
    y = h(pre(x),u);   // output expression
  end when;
end SimplePeriodicSampler;
```

### 3.2 Sustainability Modeling of Societies Using System Dynamics

An interesting kind of models related to sustainability deal with our society, the human population, and interaction with nature. When this is applied to the human race living on and interacting with the earth, our world, those models are often called World models. Several models were developed mainly during the 1970's, with some later updates (Meadows et al 2004) aiming at understanding the complexity of the interactions between global societies with their physical environment. One characteristic was their generality and complexity, spanning several subsystems (demographic, energy, economy, industry, agriculture, minerals, etc.) with varied levels of detail. Several of these models use the System Dynamics modeling approach, which is a more graphic form of expressing rather ordinary differential equations and functions. System dynamics modeling is also possible in Modelica using the System Dynamics Modelica Library (Cellier 2008).

Two simulations are depicted in Figure 11, showing a collapse scenario and a sustainable scenario for our current world society. A more thorough explanation of these scenarios is available in (Meadows 2004). These simulations are strongly connected to the current world sustainability and climate crisis. A more thorough discussion about world models, both simple and complex, can be found in Chapter 15 of (Fritzson 2014).

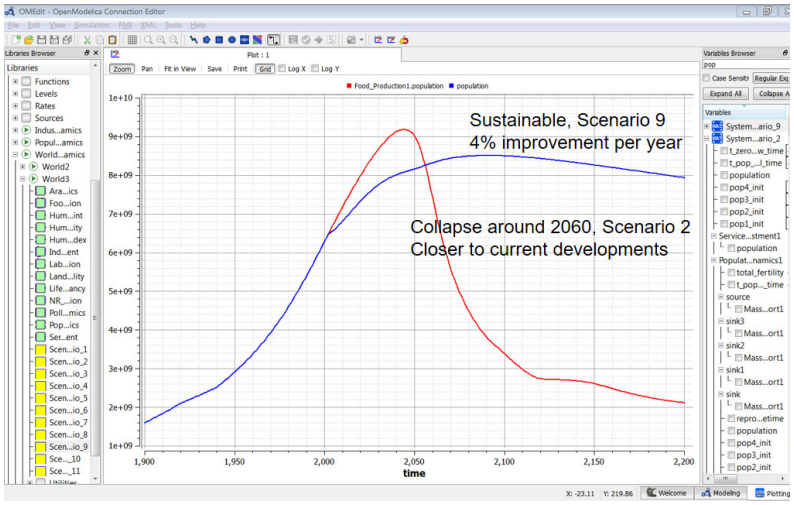



Figure 11: OpenModelica system dynamics simulation of planet Earth world society using the Modelica World3 model (Meadows et al 2004; Cellier 2008). Scenario 2 shows collapse of the world population due to pollution, destruction of arable land, etc., whereas scenario 9 demonstrates the possibility of transition to sustainability.

### 3.3 Interoperability Using the FMI Standard

To increase interoperability between tools and exchange of models, to ensure that tools, languages, and models can be maintained over time, and encourage cooperation between tool developers and the industry, it is important to rely on open standards as much as possible.

The FMI (Functional Mockup Interface) standard (Modelica Association 2017) specifies a way of describing and packaging causal models in either compiled binary or source-code (C code and XML descriptors) form. Many tools (including Modelica tools) support exporting models from their native modeling representation into FMI form. The standard is widely used in industry, especially the automotive industry which initially pushed the development in order to be able to simulate system models consisting of models from multiple tools and modeling formalisms, as depicted in Figure 12.

Today, the Modelica Association is maintaining the standard and continuously developing it further. A model or simulation unit is called FMU (Functional Mockup Unit) according to the standard. Regarding export from Modelica tools, compared to a Modelica model which is usually acausal, an exported model in FMU form is less general since it is causal - the causality of ports has to be fixed.

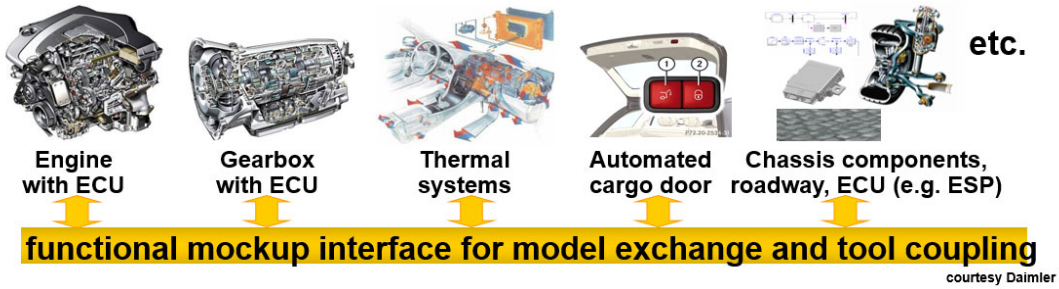



Figure 12: Automotive industry applications of FMI, allowing models from several domains to be simulated together.

SSP (Structure and System Parameterization) (Modelica Association 2018) is a complementary standard to FMI, that specifies how FMUs can be connected to create composite FMUs, and how they can be parameterized. Both FMI and SSP are standardized by Modelica Association. Editing and simulating composite FMUs using the OpenModelica OMSimulator tool is depicted in Figure 13.

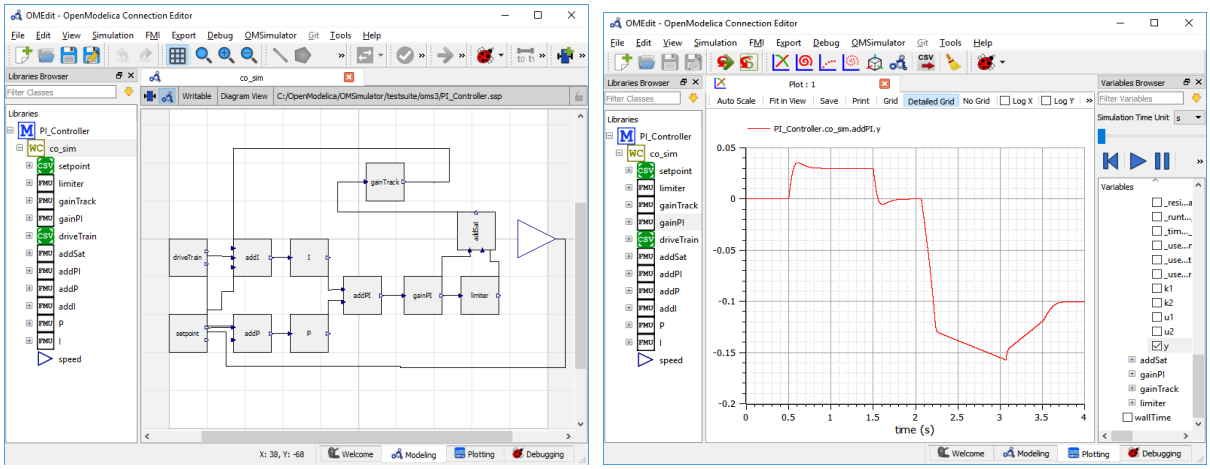



Figure 13: The OpenModelica OMSimulator composite model editor (*left*) and simulator (*right*).

## 4 CONCLUSIONS

This paper presents a quick overview of some aspects of the Modelica technology and the open source OpenModelica tool suite. We have put special emphasis on the use of Modelica, OpenModelica and FMI for the transformation of our society that is needed for sustainability and to avoid collapse, including some examples of sustainable energy as well as world modeling. The Modelica and FMI standards with support

of cyber-physical system modeling, simulation, analysis together with the freely available OpenModelica tool suite are ideally positioned to help in creating virtual prototypes and digital twins needed to transform our society into sustainability.

## ACKNOWLEDGEMENTS

This work has been supported by the Swedish Government in the ELLIIT project, and by Vinnova in the OPENPROD, MODRIO, OPENCPS, EMPHYSIS, EMBRACE, and EMISYS projects. Support has also been received from the Swedish Strategic Research foundation (SSF) in the LargeDyn project. The OpenModelica development is supported by the Open Source Modelica Consortium

## REFERENCES

Arora, I., Moudgalya, K., and Malewar, S. (2010). A low cost, open source, single board heater system. In *Proc. 4th IEEE International Conference on E-Learning in Industrial Electronics*. IEEE, November. doi:10.1109/icelie.2010.5669868.

Barricelli, B., Casiraghi, E., and Fogli, D. (2019) A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. In *IEEE Access*, vol. 7, pp. 167653-167671, 2019, doi: 10.1109/ACCESS.2019.2953499.

Bartolini, A., Casella, F., and Guironnet, A.. (2019). Towards Pan-European Power Grid Modelling in Modelica: Design Principles and a Prototype for a Reference Power System Library. In *Proc. of the 13th International Modelica Conference*, Regensburg, Germany, March 4–6, 2019. DOI: 10.3384/ecp19157627 Library: https://github.com/PowerGrids

de la Calle, A., et al. (2018) SolarTherm: A New Modelica Library and Simulation Platform for Concentrating Solar Thermal Power Systems. *Simul. Notes Eur.* 28.3 (2018): 101-103. Library: https://github.com/modelica-3rdparty/Solar

Campillo, J., Barberis, S., Traverso, A., Kyprianidis, K. and Vassileva, I. (2015) Open-Source Modelling and Simulation of Microgrids and Active Distribution Networks. In *Proc. Sustainable Places* 2015, September 16-18, Savona, Italy. URN urn:nbn:se:mdh:diva-29347

Cellier, F. (2008) World3 in Modelica. Creating System Dynamics Models in the Modelica Framework. In *Proceedings of the 6th International Modelica Conference*, Bielefeld, Germany.

Dassault Systemes. (2018) Dymola. Systems Engineering Overview, 2018. URL https://www:3ds:com/products-services/catia/products/dymola/. Accessed: September, 2018.

Franke, R. (2002) Formulation of Dynamic Optimization Problems Using Modelica and their Efficient Solution. In *Proceedings of the 2nd International Modelica Conference*, Oberpfaffenhofen, Germany, March. 18–19, 2002.

Franke, R., Rode, M., and Krüger, K. (2003) On-line Optimization of Drum Boiler Startup. In *Proceedings of the Modelica'2003 Conference*, Linköping, Sweden, November.

Fritzson, P. (2014) Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach, 2 ed., 1250 pp. Wiley-IEEE Press.

Fritzson, P., Pop, A., Sjölund, M., and Asghar, A. (2019) MetaModelica – A Symbolic-Numeric Modelica Language and Comparison to Julia. In *Proc. of the 13th International Modelica Conference*, Regensburg, Germany, March. doi:10.3384/ecp19157289.

Fritzson, P., Pop, A., Abdelhak, K., Asghar, A., Bachmann, B., Braun, W., Bouskela, D., Braun, R., Buffoni, L., Casella, F., Castro, R., Franke, R., Fritzson, D., Gebremedhin, M., Heuermann, A., Lie, B., Mengist, A., Mikelsons, L., Moudgalya, K., Ochel, L., Palanisamy, A., Ruge, V., Schamai, W., Sjölund, M., Thiele, B., Tinnerholm, J., and Östlund, P. (2020). The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control*. 2020;41(4):241-295, November DOI 10.4173/mic.2020.4.1

Giraud, L., et al. (2015) Presentation, validation and application of the DistrictHeating Modelica library. (2015) In *Proc. of the 11th International Modelica Conference*. Linköping University Electronic Press, 2015.

Hernandez-Albaladejo, G., and Urquia, A. (2018) Modelling of low-temperature solar thermal systems with Modelica. *IFAC-PapersOnLine* 51.2: 783-788 Library: https://github.com/modelica-3rdparty/Soltermica

Julialang. *Julia Language Documentation*, Release 1.0. (2018) URL https://julialang:org. Accessed:2020-09-11.

MathWorks. Matlab Product Overview. (2018) URL: https://www:mathworks:com/products/matlab:html. Accessed: September, 2018.

Meadows, D., Randers, J., and Meadows, D. (2004) *Limits to Growth: The 30-year Update*. Chelsea Green Publishing, 3rd edition.

Menager, N., Worschech, N., and Mikelsons, L. (2014) Toolchain for Rapid Control Prototyping using Rexroth Controllers and Open Source Software. In *Proc. of the 10th International Modelica Conference*, Lund, Sweden, March. doi:10.3384/ecp14096371.

Modelica Association. (2017) Functional Mock-Up Interface, Version 2.0. Interface Specification. 2017. URL: https://fmi-standard.org/downloads/ Accessed on 20 January 2020.

Modelica Association. (2018) SSP – MA Project for System Structure and Parameterization of Componentsf or Virtual System Design. URL: https://www:modelica:org/projects . Accessed: September, 2018.

Modelica Association. (2020) Modelica Standard Library version 4.0. URL: https://doc.modelica.org/ and https://github.com/modelica/ModelicaStandardLibrary/releases/tag/v4.0.0 Accessed June 2020.

Modelica Association. (2021) *Modelica - A Unified Object-oriented Language for Physical Systems Modeling - Language Specification Version 3.5*, February 18, 2021. https://modelica.org/documents/MLS.pdf

Modelica Association. (2021) Modelica tools. URL: https://modelica.org/tools.html Accessed April 2021.

Nagy, Z., Mahn, B., Franke, R., Allgöwer, F. (2007) Real-Time Implementation of Nonlinear Model Predictive Control of Batch Processes in an Industrial Framework. In: Findeisen R., Allgöwer F., Biegler L.T. (eds) *Assessment and Future Directions of Nonlinear Model Predictive Contro*l. Lecture Notes in Control and Information Sciences, vol 358. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-72699-9_38

Pop, A., Sjölund, M., Asghar, A., Fritzson, P., and Casella, F. (2014) Integrated Debugging of Modelica models. *Modeling, Identification and Control*, 35(2):93--107.

Python Software Foundation. (2018) Python Programming Language. URL https://www:python:org/ .Accessed: Sept, 2018.

Sjölund, M. (2015) *Tools and Methods for Analysis, Debugging, and Performance Improvement of Equation-Based Models*. Ph.D. thesis, Linköping University, Department of Computer and Information Science. doi:978-91-7519-071-6.

Thiele, B., Beutlich, T., Waurich, V., Sjölund, M., and Bellmann, T. (2017) Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library. In Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15–17.

Verzelen, F., Lacy, P., and Stacey, N. Designing disruption: The critical role of virtual twins in accelerating sustainability. https://www.3ds.com/sites/default/files/2021-01/dassault-systemes-and-accenture-virtual-twin-and-sustainability.pdf, Accessed March, 2021

Waurich, V. and Weber, J. Interactive FMU-Based Visualization for an Early Design Experience. In *Proc. of the 12th International Modelica Conference*, Prague, Czech Republic, May 2017. doi:10.3384/ecp17132879.

## AUTHOR BIOGRAPHIES

PETER FRITZSON is Professor Emeritus and research director of the Programming Environment Laboratory, at Linköping University, Linköping, Sweden. He earned his Ph.D. in Software Technology and Software Engineering from Linköping University. He is also vice director of the Open Source Modelica Consortium, vice director of the MODPROD center for model-based product development, and until year 2020 vice chairman of the Modelica Association, organizations he helped to establish. During 1999-2007 he served as chairman of the Scandinavian Simulation Society, and secretary of the European simulation organization, EuroSim. His research interests are in programming languages, tools and environments; multi-core computing; compilers and compiler generators, high level specification and modeling languages with emphasis on tools for object-oriented modeling and simulation where he is one of the founders of the Modelica language. He has authored more than 319 technical publications, including 21 books/proceedings. Main book: Principles of Object Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 1250 pages. Wiley IEEE Press, 2014 His email address is peter.fritzson@liu.se. His website is www.ida.liu.se/~petfr