Simulating Complex Energy Systems With Modelica: A Primary Evaluation

Atiyah Elsheikh, Edmund Widl, Peter Palensky
Austrian Institute of Technology, Energy Department, Vienna, Austria
Email: {Atiyah.Elsheikh, Edmund.Widl, Peter.Palensky}@ait.ac.at

***Abstract*—Rapid physical modelling of complex energy systems is a requirement for coping with rapid changing technologies demanded by this field. The variety of model domains and component types required for describing such complex systems makes the modelling task very challenging. Moreover, while a wide set of advanced tools for modelling highly-specialized tasks and perspectives already exists, a tool that covers all perspectives and components of a complex energy system does not exist. For this purpose, a rather primary but comprehensive evaluation of employing the universal modelling language Modelica in modelling applications of complex energy systems is presented. The advantages of utilizing such an approach is emphasized on an abstract model that is composed of several typical components within a complex energy system.**

I. Introduction

Typical energy systems composed of consumption units, energy sources, transport etc. are facing rapid changes. Sooner or later, classical energy sources are going to be replaced by renewable ones. Meanwhile, power supply demands for increasing comfort requirements, for continuously growing industry and for development plans of ambitious cities need not only to remain satisfied but also to be efficiently exploited. Technologies are rapidly changing, modern buildings are equipped with thousands of sensors and intelligent agents enhanced with individual or centralized controllers able to respond to external environmental changes as well as changing market prices [1]. Visions for smart buildings and cities able to meet their own demands are evolving and are being seriously taken into consideration.

These expected rapid changes encourage model-driven research [2] in the domain of complex energy systems from a universal point of view. By achieving this goal, model-based investigations regarding security, power supply, load flow, discovering hidden aspects of considering new technologies etc. can be performed. In order to achieve this, complex energy systems need to be described by validated models. The first step is to construct models describing such systems. This is rather challenging due to the variety of domain categories [3]:

1) Physical World: energy generation and consumption, heating, cooling, etc.
2) Information Technology: sensors, controllers, communication, software, etc.
3) Roles and individual behaviour: Intelligent agents aiming at economically efficient energy consumption for individual units, market players controlling energy prices according to competition and wining strategies etc.
4) Stochastic behaviour: weather, consumers behaviour, etc.

The mentioned four categories of components that constitute an energy system can be classified under four categories of mathematical models: continuous models, discrete models, statistics models and even game theory models for self-deciding intelligent components. In advanced cases of energy systems, a comprehensive hybrid modelling paradigm with components of variable-structure dynamics [4] is required for describing such complex systems. Another challenging aspect arises in the dimension of energy systems when tens of thousands of components interact.

Unfortunately there is a lack of methods and tools that provide comprehensive description of such systems. While there are highly specialized tools for the various individual domains of the energy system [5], [6], there is no method or tool that combines all of the mentioned aspects with which model-based questions and decisions can be researched and investigated. In this paper, a comprehensive investigation for examining the descriptive power of the universal modelling language Modelica is demonstrated. The goal is not to be able to fully overcome the mentioned challenges but to examine the language features and its ability for coping with the mentioned rapid changes at the modelling level as well as its effectiveness when tackling many realistic challenges.

This paper is structured as follows: section II presents a quick background on Modelica. Section III introduces rather a simple mathematical model but it covers all mentioned aspects within complex energy systems. Section IV demonstrates the facilities of the language Modelica along the carried implementation of abstract model step by step. Basic attractive features of the language Modelica relevant for complex energy systems are emphasized. Section V demonstrates some simulation results and emphasizes accuracy and performance perspectives of the underlying simulation environment. Finally, conclusion and future works are given in section VI.

978-1-4673-1703-0/12/$31.00

II. Background to Modelica

Modelica [7] is a modern equation-based object-oriented modeling language for DAE-based physical models with continuously increasing attention from the scientific and engineering community. Modelica relies on the fact that no matter how complex a physical system is, it can be conceptually decomposed into structured hierarchies of elementary components. Each of these components described by basic laws of Physics can be independently implemented using intuitive equation-based syntax. These components are enhanced with special interfaces called connectors with which communication to the external world (i.e. other components) is well-defined according to common conservation laws present in physical domains (e.g. energy can be neither created nor destroyed). This is the main concept behind the so-called acausal modelling approach adopted by Modelica [8], which does not enforce the modeler to specify causal input-output relations among components and variables [9], in contrast to a block-diagram approach (e.g. Simulink). Consequently, once the implementation of independent components is provided, the physical construction of complex hierarchical systems becomes the matter of dragging, dropping and connecting components. Moreover, the resulting descriptive models are usually analogous to the conceptual topology of the underlying systems in reality.

The employment of Modelica has a lot of advantages. Firstly, relying on universal modeling concepts enables multidisciplinary modeling, an aspect that is obviously present in energy systems. Secondly, Modelica is enhanced with a lot of standardized libraries in many physical domains (e.g. Thermodynamics, Electrical Engineering) provided by the Modelica Standard Library (MSL). Additionally, further highly-specialized libraries for simplifying the formulation of complex hybrid systems exist (e.g. state graphs [10], state chart [11] and Petri nets [12], [13]). Thirdly the language facilities inherently support fast prototyping of physical models by encouraging the implementation of reusable, independent, and extensible components.

Furthermore, Modelica simulation environments are usually enhanced with advanced compilers that symbolically manipulate typically large DAE systems of higher indices using sophisticated algorithms for generating efficient solvable real-time simulation code with the help of advanced universal solvers (cf. Figure 1). These solvers are capable of handling stiff systems and hybrid systems with state and/or pre-known time events. With the help of common commercial and free open-source Modelica simulation environments developed from industry and academica like Dymola [14], OpenModelica [15] and MathModelica [16], the modeler can essentially focus on physical modeling rather than paying attention at the task of mathematical formulation.

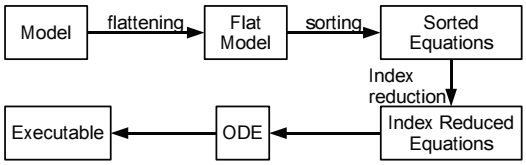

Fig. 1: An overview of a compact Modelica library

III. Description of a toy model

In this section, an easily scalable test model is introduced as presented in [3]. This model consists of many components of various types mentioned in the previous section. The model basically consists of the following components:

- Houses acting as energy consumption units
- Walls acting as isolator of the houses.
- A heater for each house as a medium for consuming power whenever it is switched on.
- An agent for each house controlling the heater settings for economical efficient power consumption
- A window for each house which gets opened according to random behaviour. In this case, the temperature of the house gets more influenced by the outside weather.
- A market component which controls the price of power and influences therefore the agents' decisions

Formal description is provided as follows.

Houses

Each house has a heater. Power is consumed whenever heaters are switched on. A house is acting as a thermal capacitor isolated by a thermal resistor (wall) from a thermal reservoir (ambient environment). A heater is controlled by an agent through a two-point controller. An agent specifies two temperature values $T_{\min}$ and $T_{\max}$ at which the agent turns on/switch off the heater. The following equations describes basic entities characterizing the temperature within houses:

- Thermal energy stored in thermal capacitor:

$$Q_{\text{store}} = \rho V C_{\text{th}} T_{\text{in}} \tag{1}$$

- Heat flow through thermal resistor:

$$\dot{Q}_{\text{loss}} = \frac{1}{R_{\text{th}}} \left(T_{\text{in}} - T_{\text{amb}}\right) \tag{2}$$

- Heat flow regulated by two-level controller:

$$\dot{Q}_{\text{heat}} = \begin{cases} 0 & \text{if heating is off,} \\ P_{\text{heat}} & \text{if heating is on.} \end{cases} \tag{3}$$

- Heat flow balance:

$$\dot{Q}_{\text{store}} = \dot{Q}_{\text{heat}} - \dot{Q}_{\text{loss}} \tag{4}$$

T_{in} and T_{amb} are the inside and outside temperature, ρ and V are the density and the cubic content of the inside volume, C_{th} is the thermal capacity of the inside volume and R_{th} is the thermal resistance of the walls computed as:

$$\frac{1}{R_{\text{th}}} = \frac{\lambda \phi V}{d} \tag{5}$$

978-1-4673-1703-0/12/$31.00 ©2013 IEEE

The walls' thickness and thermal conductivity are given by d and λ, ϕ is the ratio of the outside walls' area to the inside volume's cubic content (form factor)

Stochastic events

For demonstrating events and random static behaviour each house may get ventilated by a window that gets opened and closed according to statistically random behaviour. Each time a window gets opened, a heat transfer from the environment to the house takes place according to a change in thermal energy:

$$\dot{Q}_{\text{window}} = G_{\text{vent}} \left(T_{\text{in}} - T_{\text{amb}}\right) \tag{6}$$

where G_{vent} is the (very large) additional heat conductance value representing additional ventilation. Two different time constants, τ_{open} and τ_{close}, are used to specify the average waiting time for a window to be opened and then closed again, respectively. To model independent random events, the waiting times are drawn from exponential distributions, according to the corresponding time constants.

Market

The market component determines the price per consumed kWh according to the total energy consumption of all houses according to:

$$p = p_0 + \langle \bar{E}_{\text{bec}} \rangle_n \times p_1 \tag{7}$$

E_{bec} is the consumed energy per house in the last 30 minutes, $\bar{E}_{\text{bec}}$ is the mean value of E_{bec} of all houses and $\langle . \rangle_n$ denotes the average over n periods. The terms p_0 and p_1 are constants.

Agents

An agent determines at which temperature a heater is turned on or switched off at a lower goal temperature T_{min} or an upper goal temperature T_{max}, respectively. In case the energy price exceeds a certain level p_{max}, the agent decreases the upper goal temperature from T_{max} to T_{alt}.

Weather

The temperature of the outside temperature is described by a sinusoidal pattern as:

$$T_{\text{amb}} = \bar{T}_{\text{amb}} + \Delta T_{\text{amb}} \sin\left(\omega t + \varphi\right) \tag{8}$$

The parameters ω and ϕ are chosen such that the minimum temperature occurs every day at midnight, see table I for a complete list of parameter values.

IV. Implementation in Modelica

Basic overview

For managing a large number of components and models, Modelica enables the modeler to organize his models within a library of structured hierarchies of subpackages. For the considered simple model a compact library providing the basic components for realizing the implementation of the given test model is shown in figure 2. There are basically three packages: *Types, Interfaces* and *Components*.

TABLE I: Simulation parameters

Parameter	Value	Unit
ρ	1.2041 †	kg m^{-3}
V	100 - 300	m^3
C_{th}	430.587 ‡	J kg^{-1} K^{-1}
P_{heat}	1.4 - 1.6	kW
ϕ	0.3 - 0.8	m^{-1}
d	0.25 - 0.35	m
λ	0.1 - 0.20	W K^{-1} m^{-1}
G_{vent}	150 - 250	W K^{-1}
T_{min}	18 - 20	°C
T_{max}	25 - 28	°C
T_{alt}	21 - 24	°C
τ_{open}	3	h
τ_{close}	15	min
Δt_{meter}	30	min
n	4	
p_0	0.15	c.u. *
p_1	0.10	c.u. kWh^{-1} *
p_{max}	0.18 - 0.20	c.u. *
$\bar{T}_{\text{amb}}$	10	°C
ΔT_{amb}	5	°C

* c.u. = currency unit
† corresponds to dry air at 20°C and 1 atm
‡ assumes ideal gas model

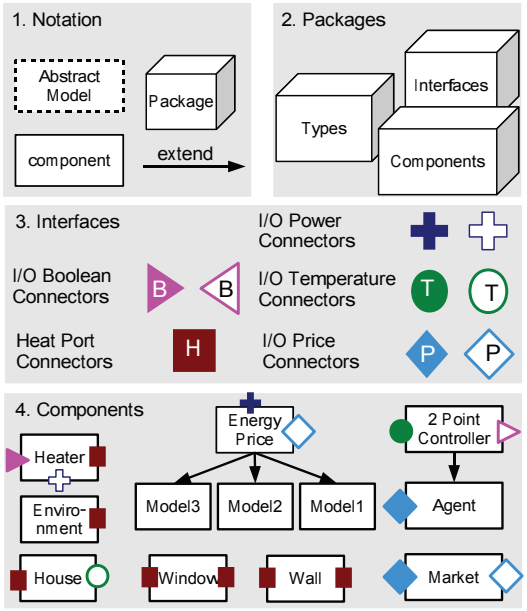

Fig. 2: An overview of a compact Modelica library

Physical types

Within the package *Types*, types for physical units declarations are provided. For instance, types for physical quantities are realized as follows:

Listing 1: Implementation of physical types

```
type Power = Real(final unit = "W",
                  final quantity = "Power");
type PowerConsumption = Real(final unit="W/s",
                             min=0);
```

Note that many of the used physical types are already provided by the MSL. The associated physical units are usually

978-1-4673-1703-0/12/$31.00 ©2013 IEEE

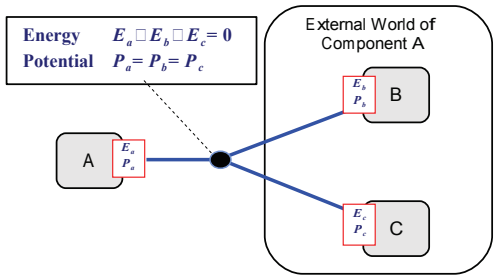

Fig. 3: An overview of a compact Modelica library

employed for performing physical unit checking at compile time. At simulation runtime, boundaries checking can be performed. This certainly enhances modeling at early stage of development and avoids common mistakes.

Connections

The package *Interfaces* provides the s.c. connectors acting as communication ports with which elementary components can be connected for building up complex models. A typical implementation of a connector (for instance the HeatPort within the MSL) looks as follows:

Listing 2: Heat Port

```
connector HeatPort
        "Thermal port for 1-dim heat transfer"
  Types.Temperature T "Port temperature";
  flow Types.HeatFlowRate Q_flow
     "Heat flow rate (positive if flowing from
     outside into  the component)";
end HeatPort;
```

Within a typical connector, two types of variables are declared, potential variables (e.g. voltage) and flow variables (e.g. current). Only connectors with identical declaration can be connected together. In this case, a connection point as shown in Figure 3 assembles two kinds of equations: trivial identity equations for potential variables and sum-to-zero equations for flow variables. The former type of equations describes quantities that have to be instantaneously equal among all components while the latter type of equations describes the flow of energy among components (e.g. the flow of heat). I/O connectors consisting only of potential variables are functioning in a similar manner as within Simulink (e.g. I/O boolean connectors for switching).

Components: A house as an example

The package *Components* provides the basic components for the test model. For instance the implementation of a house component looks as follows:

Listing 3: The house component

```
model House
  "Temperature of a thermal capacitor (house)"
  Types.Temperature T(start=20+273.15,
                      displayUnit="degC");
  parameter Types.ThermalCapacity Cth = 430.578
    "Heat capacity";
  parameter Types.Density ro = 1.2041 ;
```

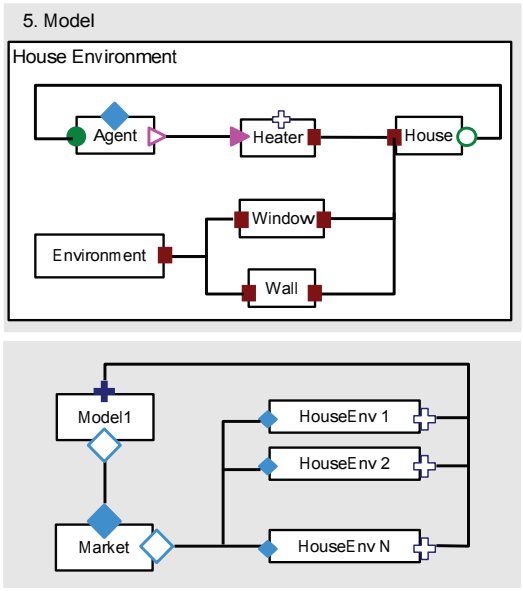

Fig. 4: An implementation of the test model in Modelica

```
  parameter Types.Volume volume = 200 ;
  Interfaces.OutputTemperature outputTemperature;
  Interfaces.HeatPort port_a;
equation
  outputTemperature = T;
  T = port_a.T;
  ro*volume*Cth*der(T) = port_a.Q_flow;
end House;
```

The test model

Using the presented library, the test model can be constructed as intuitively clarified in figure 4. The textual implementation (usually automatically generated) is simply a one-to-one mapping of the physical model as shown as follows:

Listing 4: The environment of a house

```
model HouseEnvir9
  "Non-isolated House and Temperature controlled
  by 2-point controller"
  Agent agent;
  House house;
  Heater heater;
  Wall wall;
  Environment environment;
  Window window;
equation
  connect(house.outputTemperature,
                  agent.inputTemperature);
  connect(agent.switch, heater.switch);
  connect(heater.port_b, house.port_a);
  connect(environment.port_b, wall.port_a);
  connect(wall.port_b, house.port_a);
  connect(environment.port_b, window.port_a);
  connect(window.port_b, house.port_a);
end HouseEnvir9;
```

Note that the flexibility of the language allows for different implementation flavors from other perspectives of viewing the model structure [17].

978-1-4673-1703-0/12/$31.00 ©2013 IEEE

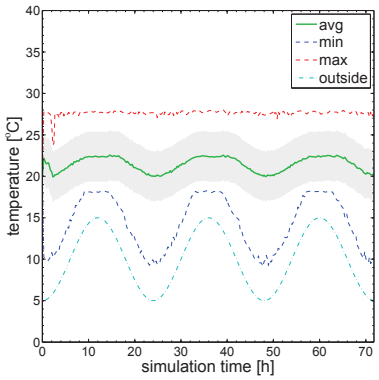

Fig. 5: Average temperature profile for 1000 buildings simulated using a variable step size integrator.

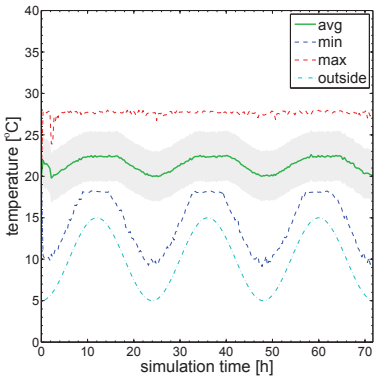

Fig. 6: Average temperature profile for 1000 buildings simulated using a fixed step Runga-Kutte method.

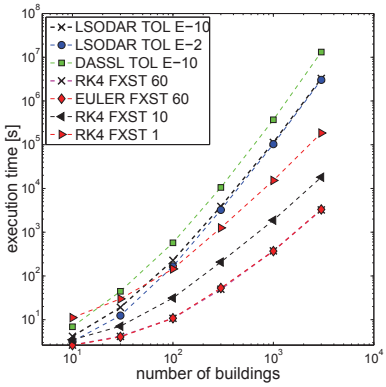

Fig. 7: Overview of elapsed computing times for various integrators.

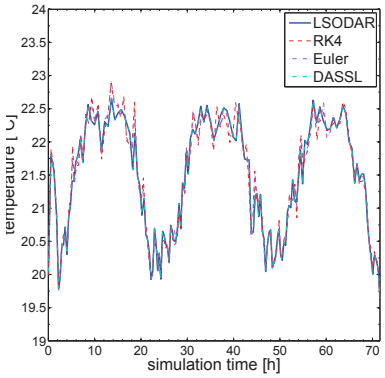

Fig. 8: Comparison of the continuous average temperature profile for different integrators.

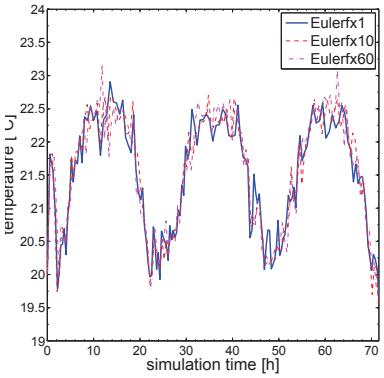

Fig. 9: Comparison of the continuous average temperature profile for different fixed step sizes.

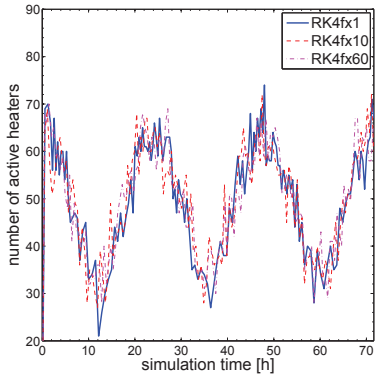

Fig. 10: Comparison of the discrete number of active switches profile for different step sizes.

Replaceable components

Many useful features and advanced constructs for model abstractions, model templates, inheritance, arrays, variable structure systems and event handling are employed within the overall implementation. For instance, it is possible to parametrize the types of components for instantiating further models from one model:

Listing 5: Redeclaration of classes

```
model Market = Market(
        redeclare class EPrice=Model2);
```

In this way different models adopting several types for energy prices can be adopted. This is also useful for supporting rapid prototyping on the basis of a top-down modelling approach when less detailed elementary components are replaced with fully detailed complex subsystems.

V. Simulation Results

In this section basic simulation results are shown. All simulations have been performed on a DELL™PowerEdge™R415 machine with two AMD Opteron 4184 CPUs (64 bit, six cores, 2.8 GHz) and 128 GB RAM working memory. In all simulations a time span of three days has been simulated. The Modelica simulation environment Dymola 7.4 is used on the mentioned hardware.

By simulating the model, many state profiles concerning the power consumption in a single building unit, the overall power consumption and its response w.r.t. the variable energy prices as well as temperature profiles can be demonstrated. Figures 5 and 6 show the average temperature profile of 1000 houses simulated by the variable step size integrator LSODAR with relative tolerance of 10^{-6} and a Runga-Kutta method of order four and a fixed step equal to one second [3]. With the availability of several types of integrators each of which could be employed either with strict or relaxed accuracy settings, it is advantageous to examine all of these w.r.t. runtime and accuracy performance. Figure 7 shows the runtime performance of many of the available methods with

978-1-4673-1703-0/12/$31.00 ©2013 IEEE

different settings.

While the performance of fixed-step integrators with relatively large-step sizes are much better than variable-step integrators, the accuracy of the results needs to be examined. For instance, figure 8 shows the temperature profile of a chosen house for different methods with strict settings (relative tolerance of 10^{-10} and fixed step of 1 for dynamic and fixed integrators respectively). The comparison reveals that the choice of the type of method is quite insignificant for these types of models and the less efficient fixed-step methods can be employed without facing stability or accuracy problems. Furthermore, figures 9 and 10 demonstrate the continuous temperature profile within a single house and the discrete number of active heaters, respectively for a simulation with 100 houses. Both figures demonstrate the accuracy of fixed step methods with different step sizes. While enlarging step-sizes leads to improved runtime performance, this comes with the price of less accurate results. The selection of the right integrator with the right settings depends largely on the desired demand of runtime performance and accuracy.

VI. Conclusion and Outlook

This work presented a primary evaluation of the language Modelica on a toy model. It has been shown that Modelica provides powerful facilities for describing models composed of various categories of components belonging to different types of physical domains. This represents a fundamental characteristic of the multidisciplinary scientific domain of complex energy systems. A fundamental advantage of Modelica is that its design supports rapid prototyping of model components. This is a further requirement for ongoing work aiming at constructing advanced realistic models by considering:

- An electric grid with non-ideal power sources, losses, renewable energy sources, etc.
- Combined Heat and Power (CHP elements with a thermal grid, economic choice between electric or thermal heating.
- Individual user behaviour depending on time, history, demographic categories, etc.
- More complex weather models with geographic aspects.
- Dynamic markets for energy/fuel depending on season, consumption, etc.
- Enhance all data and parameters with meta-data like confidence level, geographic tags, ownership, etc.
- Parallel/distributed computing on multi-core environments, clusters and Graphic Processing Unit (GPU).

Each extension step would be followed by a comprehensive investigation w.r.t. scalability and numeric performance. Meanwhile fundamental challenges concerning adoption of advanced hybrid paradigms, co-simulation with highly-specialized tools and improving the runtime performance are desired requirements for progressive research. All these are currently under study as an ongoing work aiming at setting up a usable high-performance environment to investigate complex energy systems. The application scenarios that will serve as use cases are smart grid functions like demand response, multi-agent energy balance communities or energy storage management and long term scenarios like the impact of renewable energies and other new technologies.

References

[1] P. Palensky and D. Dietrich, "Demand side management: demand response, intelligent energy systems, and smart loads," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 381–388, 8 2011.

[2] M. Broy, M. Feilkas, M. Herrmannsdoerfer, S. Merenda, and D. Ratiu, "Seamless model-based development: From isolated tools to integrated model engineering environments," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 526 –545, April 2010.

[3] P. Palensky, E. Widl, and A. Elsheikh, "Simulating complex energy systems: challenges, tools and methods," *submitted to IEEE Transactions on Industrial Informatics*, 2012.

[4] D. Zimmer, "An application of sol on variable-structure systems with higher index," in *Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009*, 2009.

[5] M. Ilic, L. Xie, U. Khan, and J. Moura, "Modeling future cyber-physical energy systems," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, july 2008, pp. 1 –9.

[6] C. Macana, N. Quijano, and E. Mojica-Nava, "A survey on cyber physical energy systems and their applications on smart grids," in *Innovative Smart Grid Technologies (ISGT Latin America), 2011 IEEE PES Conference on*, oct. 2011, pp. 1 –7.

[7] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica*. Wiley-IEEE Computer Society Pr, 2003.

[8] H. Elmqvist and S. E. Mattsson, "Modelica - the next generation modeling language: An international design effort," Passau, Germany, 1997.

[9] S. E. Mattsson and H. Elmqvist, "Modelica - an international effort to design the next generation modeling language," Gent, Belgium, 1997.

[10] K.-E. M. Otter and I. D. Årzén, "Stategraph-a modelica library for hierarchical state machines," in *Proceeding of the 4th International Modelica Conference*, Hamburg, Germany, 2005.

[11] J. Ferreira and J. Estima de Oliveira, "Modelling hybrid systems using statecharts and Modelica," in *The 7th IEEE International Conference on Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA '99.*, vol. 2, 1999, pp. 1063 –1069 vol.2.

[12] S. Proß and B. Bachmann, "A petri net library for modeling hybrid systems in openmodelica," in *Proceeding of the 7th International Modelica Conference*, Como, Italy, 2009.

[13] P. J. Mosterman, M. Otter, and H. Elmqvist, "Modeling petri nets as local constraint equations for hybrid systems using modelica," in *In: proceedings of the Summer Computer Simulation Conference -98*, 1998, pp. 314–319.

[14] `http://www.dymola.com`, Dassault Systèmes.

[15] P. Fritzson, P. Aronsson, A. Pop, H. Lundvall, K. Nystrom, L. Saldamli, D. Broman, and A. Sandholm, "Openmodelica - a free open-source environment for system modeling, simulation, and teaching," in *Proceedings of the 2006 IEEE International Conference on Control Applications, 2006 IEEE Conference on Computer-Aided Control Systems Design, 2006 IEEE International Symposium on Intelligent Control (ISIC)*, 2006, pp. 1588 –1595.

[16] `http://www.mathcore.com`, MathCore Engineering AB - Wolfram Research.

[17] W. Wiechert, S. Noack, and A. Elsheikh, "Modeling languages for biochemical network simulation: Reaction vs equation based approaches," *Advances in Biochemical Engineering / Biotechnology*, 2010.

978-1-4673-1703-0/12/$31.00 ©2013 IEEE