OBJECT-ORIENTED MODELLING AND VERIFICATION AIDED BY MODEL SIMPLIFICATION TECHNIQUES

Anton Sodja **Borut Zupančič**

Faculty of Electrical Engineering, University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
anton.sodja@fe.uni-lj.si borut.zupancic@fe.uni-lj.si

ABSTRACT

Object-oriented modeling approach brought efficient model reuse and thus possibility to create rich model libraries which enable rapid development of large heterogeneous models. However, verification and debugging of large complex models is becoming and increasingly challenging task.Furthermore, model should not be more complicated as needed for a given purpose. A suitable component describing a subsystem in sufficient detail should be selected from a library which might contain several components describing the same system but with different level of detail.

Benefits of model simplification techniques for object-oriented model development are discussed in this paper. A modeler may help himself with them in a decision making process of how detailed components should be used (e.g., how complicated model should be) and also as an assistance for verifying model by some informal verification methods. Simplified model should be represented in the same way as original, therefore, two simplification techniques are discussed, simplification of object-diagrams and simplification of equations, which are the usual representation of models in Modelica, one of the commonly used object-oriented modeling language today.

Keywords: nonlinear model simplification, model verification, Modelica

1. INTRODUCTION

Dynamic models are an important part of many engineering applications. Various purposes which models have in engineering applications include assistance for system and control design, explaining complex-system behavior and help in operators training.

The purpose of the model also determines a desirable complexity of the model. Modeling is an iterative process and the complexity of the model usually increases during the process as well as the model purpose tends to change. In the early stages of model development relatively simple conceptual models are used which help examine some general design situations. The error bounds on model prediction are relatively large and validation with experimental data is usually not possible due to the lack of measurements. Therefore, the model is only verified with general design experience and comparison with earlier models of systems with similar characteristics (Murray-Smith, 2009).

Later in the model building process, as more data becomes available, more complex description of the system are integrated into the model. However, more thorough validation may even indicate that some parts of the model are unnecessarily detailed and model is consequentially simplified.

The model building process may be also carried out in reverse: all known details about the systems are included into a model and in the further modeling phases the unnecessarily parts of the model are identified and removed (Murray-Smith, 2009).

The emergence of the object-oriented modeling exacerbated the difficulty of assuring proper model complexity and model-verification in some respects. Object-oriented models are built of inter-connected components (models of subsystems). Large collections of various pre-prepared components are available in designated model libraries (Tummescheit, 2002; Andres et al., 2009; Cellier, 1991). Therefore, a very complex model can be built easily. However, this kind of modeling paradigm has some pitfalls, for example, when the modeler is not familiar with the assumptions made at formulation of the components, they can be used incorrectly, e.g., resulting model is invalid due to incompatible components (especially when they come from different libraries). Furthermore, at different modeling stages, different level of model complexity is required and hence a set of components describing the same subsystem but with different level of detail must be present. Another option is to include switches into the components which enables to turn off unnecessary level of details (Casella et al., 2006).

When a model consists of many components (as it is the case with large models) it can be quite intricate to determine how complex each component should be. The complexity of the components with low impact on overall

model behavior should be of course kept low to bound overall model complexity.

2. MODEL SIMPLIFICATION TECHNIQUES

Engineers use experience and intuition to determine important parts of the model which have the highest impact on system's dominant dynamics or on the model's simulation response in specific scenario. In an attempt to diminish reliance on subjective factors such as experience, numerous model simplification and order-reduction methods have been developed (Schwarz et al., 2007; Sommer et al., 2008; Lall et al., 2003; Louca, 1998; Chang et al., 2001). Model simplification techniques consist of running a series of simulations, ranking the individual coordinates or elements by the appropriate metric and removing those that fall below a certain threshold (Chang et al., 2001). The choice of the ranking metric and simplification steps depends on modeling formalism used and may be limited by modeling domains. A special class of model simplification methods are those that produce *proper models*, i.e., simplified models have physically meaningful variables and parameters.

It is obvious how model simplification tools can help us determine if the model is too complicated: if model could be simplified a lot without loosing too much accuracy, it is clearly too complicated.

Model simplification methods can also facilitate model verification when less formal approaches such as desk-checking are used (Sodja and Zupančič, 2010). Even rather small models and their simulation results can be to large to be human interpretable and understandable. Hence, physically interpretable simplified models could be used instead to provide a deeper insight into the system's behavior needed for model verification. In some cases when only a part of the model (a submodel) is under consideration, it is desired that only this submodel could be simplified.

3. SIMPLIFICATION OF MODELICA MODELS

Modelica modeling language was designed for efficient modeling of large, complex and heterogeneous physical systems (Modelica Association). Model is usually decomposed into several hierarchical levels. On the bottom of the hierarchy there are submodels of basic physical phenomena which are most commonly stated as a set of (acausal) differential-algebraic equations. It is thus most conveniently that these equations can be entered directly (e.g., without a need for any kind of manipulation or even transformation to some other description formalism). On higher hierarchical levels, the model is described graphically by schematics (i.e., *object diagrams*) and the obtained scheme efficiently reflects the topology of the system. Such model representation in Modelica is thus understandable also to domain specialists who do not have a profound knowledge about computer simulation.

Prior to simulation, model must be translated. First, model is *flattened* (i.e., hierarchical structure of a model is mapped into a set of differential, algebraic and discrete equations together with the corresponding variable declarations (Modelica Association)), then some further modifications (e.g., tearing of algebraic loops and DAE index reduction) are performed so that model can be brought into the form required by numerical solvers.

There are no tools known to the authors which support simplification of Modelica models directly. Model must be thus exported to external tools using either *Functional Mockup Interface* (Blochwitz et al., 2011) or by reparsing flattened model (if this feature is supported by modeling environment). This kind of an approach have many downsides, because model is flattened or even other translation steps are performed (algebraic-loop tearing and index reduction) before the export. Therefore much information about the organization of the model (for example, hierarchical structure) is lost and the simplified models may not be convenient for verification purposes. Furthermore, it may be very intricate and laborious to simplify only certain submodel and evaluate it together with whole model.

We believe that model simplification tool should be closely integrated into modeling environment and representation of a simplified model should represented in the same way as original model. Models are in Modelica generally represented graphically (i.e., object diagrams) on a higher levels and textually (i.e., equations) on the lowest. Therefore, there are needed two classes of simplification techniques, simplification of object-diagrams and simplification of equation sets. In many cases only rankings of elements might be sufficient and simplification of the model could be done manually if needed. Simplification of submodels should be supported as only subset of model might be under consideration.

4. SIMPLIFICATION OF MODELICA OBJECT-DIAGRAMS

Object diagrams consist of connected components (submodels). A connection defines interactions which are determined by types of connectors (i.e., ports) which are used in components. Connector are rather loosely defined in Modelica. In general, a list of variables with some qualifications (e.g., causality, type of variable: intensive – extensive, etc.) is defined, but it can also have a hierarchical structure (Modelica Association).

4.1. Choice of ranking metric

Although different domains are modeled with rather different schemes and connections, acausal connections for modeling physical interactions are of special interest. Each (dynamic) interaction between physical systems results in an energy exchange between the systems. So it is very intuitive to chose the energy-based metrics for the simplification of a physical systems models.

Metric we chose Louca (1998) is defined by Eq. 1. It is the integral of absolute net energy flow that element exchanges with environment in time interval $[t_1,t_2]$.

$$\text{Activity} = \int_{t_1}^{t_2} |\sum_i p_i(t)| \cdot dt \qquad (1)$$

In Eq. 1, $p_i(t)$ designates an energy flow through the boundary of the element (in Modelica usually modeled with *connection*).

4.2. Determination of energy-flows in object diagrams

Modelica object diagrams, when modeling physical systems, share many similarities with bond graphs, which can be efficiently used for object-oriented acausal modeling. Therefore it is possible to adapt most of bond-graph simplification techniques to Modelica's object diagrams. Of course the energy concept in bond graphs is much more unified in comparison with different Modelica libraries. So we analyzed the energy interactions between components in different Modelica libraries at the beginning.

Connectors usually contain a pair of effort and flow variables. However, their product is not necessarily an energy flow like in bond graph formalisms. This can be seen by inspecting Modelica Standard Library (Mod, 2010) where elementary connector definitions for almost all physical domains are gathered:

- Interaction between components in analog circuits (*Modelica.electric*) is determined by voltage v and current i, the latter is a flow variable, and the power of the interaction is the product of both variables: $p = v \cdot i$.

- Similar features has also the connector in *Modelica.Magnetic*, which is composed of variables for magnetic potential difference V_m and magnetic flux Φ, an effort and flow variables respectively. The power of the connection is the product of both variables: $p = V_m \cdot \Phi$.

- Connectors used for modeling of 1-D translational and rotational mechanics consist of position s and angle ϕ respectively, and force f and torque τ respectively. However the product of connector's effort and flow variables is no longer the power. For determination of the power of the connection, displacement variable has to be differentiated: $p = \frac{d}{dt}s \cdot f$ and $p = \frac{d}{dt}\phi \cdot \tau$ for translational and rotational mechanics respectively.

- In *Modelica Multibody* library, which deals with 3-D mechanics, effort and flow variables are no longer scalars, they are 6-dimensional vectors, so a state of a free-body (having 6 degree-of-freedom) can be determined. Furthermore, due to computational restrictions, implementation of connector takes into account also a suitable selection of a frame of reference (forces, torques and orientation are expressed in local, while position is in global frame of reference). A definition of the connector is the following:

```
connector Frame
SI.Position r_0[3];
Frames.Orientation R;
flow SI.Force f[3];
flow SI.Torque t[3];
end Frame;
```

  The position is determined with the variable *r_0*, while the orientation R is a structure containing the transformation matrix T from global to local frame of the reference and the vector of angular velocities ω in the local frame of reference. Forces and torques are given by vectors f and t respectively. The power of the connection can be calculated by the expression: $p = \frac{d}{dt}(\mathbf{T} \cdot r_o) \cdot f + \omega \cdot t$, where again, there is a need to differentiate the position after transformation to local frame.

- Connector for modeling the heat transfer in 1-D consists of the effort variable temperature T and the flow variable for heat-flow rate Q_{flow}. The energy transfer is in this case equal to flow variable $p = Q_{flow}$.

- Library *Modelica.Fluid* deals with modeling of heat and mass transfer. The connector used in library's components which covers also mass transfer is implemented as following:

```
connector FluidPort
replaceable package Medium =
Modelica.Media.Interfaces.PartialMedium;

flow Medium.MassFlowRate m_flow;
Medium.AbsolutePressure p;
stream Medium.SpecificEnthalpy
    h_outflow;
stream Medium.MassFraction
    Xi_outflow[Medium.nXi];
end FluidPort;
```

  Besides effort and flow variables, pressure p and mass-flow rate m_{flow} respectively, the connector includes also additional information about properties of the substance which is being exchanged in the interaction modeled by a connection of type *FluidPort*: specific enthalpy h and composition of substance (vector of mass fractions X_i if substance is a mixture). The thermodynamic state of the substance is uniquely determined by the variables of connector and all the other (thermodynamic) properties can be calculated by using functions provided by package *Medium* which is a parameter of the connector. However, thermal diffusion is not covered by this connector (it is neglected).

  Energy flow associated with the connector is composed of thermal, hydraulic and chemical term and could be calculated as following (in Modelica By

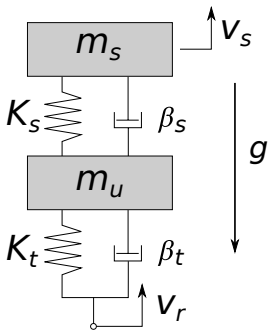

Figure 1: Scheme of a car suspension system.

Element	Activity [J]	Relative [%]	Accumulated [%]
gravityForce_s	2,270.06	37.06	37.06
spring_s	1,763.33	28.79	65.85
ground	795.02	12.98	78.82
mass_s	787.65	12.86	91.68
damper_s	198.82	3.25	94.93
spring_t	192.57	3.14	98.07
gravityForce_t	92.98	1.52	99.59
mass_t	24.53	0.40	99.99
damper_t	0.53	0.01	100.00
displacement_s	0.00	0.00	100.00
displacement_t	0.00	0.00	100.00

Table 1: Ranking of components when model from Fg. 2 is fed by input shown in Fg. 3.

Use of Chemo-bonds, 2009): $p = \dot{m} \cdot s \cdot T + \dot{m} \cdot p/\rho + \sum \mu_i \cdot \dot{N}_i$. Quantities specific entropy s, temperature T, density ρ, chemical potential μ_i and molar flow $\dot{N}_i$ can be calculated from thermodynamical state equations provided by package *Medium*.

4.3. Ranking of elements

Although it is possible to calculate energy flow of the connector from the variables of the connector, this is not always possible to do from simulation results, because some variables can not be available. For example, the derivative of a position or an angle in the connector of the library for 1-D mechanics may not be available if this variable is not chosen as a state variable. This implies instrumentation of the model, i.e., additional auxiliary variables and equations are inserted into the model.

Ranking is done as post-processing of simulation results. *Activity* of each element required for ranking is calculated with Eq. 1. Each hierarchical level is considered separately.

After the ranking of the elements is available, model can be simplified by removing all the elements that fall below certain threshold (value of *activity* in our case). However, our current implementation provides only results of ranking in a printed form (a table). The ranking table can be then used to simplify the model manually. Automatic simplification is the matter of future investigations.

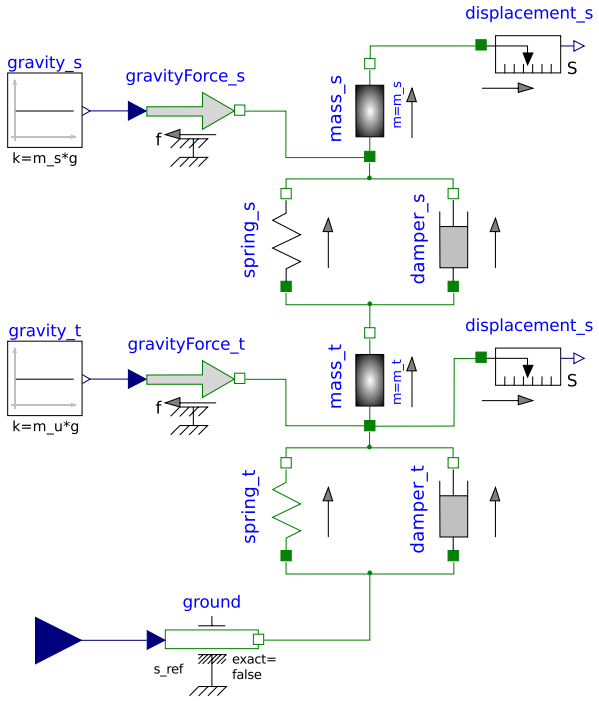

Figure 2: Car suspension system: model represented by a Modelica object diagram.

4.4. Example

The model from Fg. 2 is excited by signal depicted in Fg. 3. Components of the model are ranked with activity metrics (Fg. 1) and results are shown in table 1. The second column of table 1 consists of activities of all components calculated with Eq. 1. The third column contains relative activities components (the sum equals 100%) and the last column shows the accumulated relative activities. This column very illustratively shows how many components has to be taken into account for a reasonable accuracy.

The aim of the ranking tables is to simplify the model in Fg. 2 by removing components from the bottom of the tables 1. However, a high accumulated relative activity of the remaining components (e.g., components not removed) do not necessary imply high similarity of original and simplified model responses. It is necessary to validate the simplification comparing the original and simplified responses.

5. SIMPLIFICATION OF MODEL'S EQUATIONS

From a mathematical point of view, models in Modelica are systems of hybrid differential-algebraic equations (DAE). Therefore, it some cases we might want to investigate these equations directly. In all modeling environments supporting Modelica, models can be printed in *flat* form (i.e., as a system of equations). However, this can yield complex expression even for relatively small models, so symbolic model reduction techniques are applicable to achieve favorable representation.

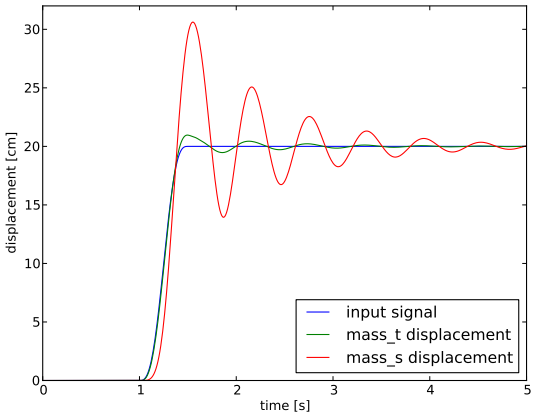

Figure 3: A car hits a smooth curb: low-frequency excitation signal is given as an input to the model in Fg.2. Also responses – displacement of an unsprung (*mass_t*) and sprung mass (*mass_s*) are depicted.

5.1. Simplification strategies

There are many mature simplification methods available for linear systems (Eitelberg, 1981; Fodor, 2002; Innis and Rexstad, 1983), while there is a lack of more general simplification methods that could be applied effectively on multi-domain models described as a set of nonlinear DAE.

Most commonly employed simplification strategies for nonlinear DAE combine model order reduction techniques (i.e., deletion of variables and variables' time derivatives or substitution of variables with constant values) and an approximation of single terms, for example, linearization, deletion or substitution of the term with constant value, etc. (Sommer et al., 2008; Wichmann et al., 1999).

The order of the applied simplification steps is determined by ranking. The most apparent ranking metric is estimation of reduced-model error for selected variables (variables of interest). One possible approach is to repeat simulation after for each possible simulation step which would yield a perfect ranking. However, this method is too time consuming to have any practical meaning.

Another option is to use energy-based ranking metric as in case of object-diagram simplification. In method suggested by (Chang et al., 2001) it is required that Lyapunov function of the system is known which is a rather harsh restriction.

The metric (Wichmann et al., 1999) suggest for simplification of nonlinear DAE systems obtained in analog circuit design estimates the error caused by simplification step (e.g., term deletion, substitution of the term with a constant value, term linearization) and it is done in two parts: the DC analysis and AC analysis. The former requires calculation of several *design points*, i.e. steady-states of the system at different inputs.

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{0},\boldsymbol{y},\boldsymbol{u},t)=0 \tag{2}$$

A set of nonlinear algebraic equations (3) is obtained for each *design point* by solving original equation system (2) with $\dot{x}=0$ and given u.

$$\boldsymbol{F}(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{y},\boldsymbol{u})=0 \tag{3}$$

Values of variables at steady-state of the modified system $\tilde{\boldsymbol{F}}(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{y},\boldsymbol{u},t)$ (where a single term in one of equations is changed) are estimated by performing single Newton-Ralphson iteration (4) and the solution of the original system is used as a guess value.

$$\begin{bmatrix}\boldsymbol{x}^{(1)}\\ \boldsymbol{y}^{(1)}\end{bmatrix}=\begin{bmatrix}\boldsymbol{x}^{*}\\ \boldsymbol{y}^{*}\end{bmatrix}-\mathbf{J}_{\tilde{F}}^{-1}(\boldsymbol{x}^{*},\dot{\boldsymbol{x}}^{*},\boldsymbol{y}^{*})\cdot\tilde{\boldsymbol{F}}(\boldsymbol{x}^{*},\dot{\boldsymbol{x}}^{*},\boldsymbol{y}^{*}) \tag{4}$$

The error estimation ε_i is then calculated by equation (5).

$$\begin{aligned}\varepsilon_i &= ||\,[x^{*},y^{*}]^{T}-[x^{(1)},y^{(1)}]^{T}\,||\\ &= ||\,\mathbf{J}_{\tilde{F}}^{-1}(\boldsymbol{x}^{*},\dot{\boldsymbol{x}}^{*},\boldsymbol{y}^{*})\cdot\tilde{\boldsymbol{F}}(\boldsymbol{x}^{*},\dot{\boldsymbol{x}}^{*},\boldsymbol{y}^{*})\,||\end{aligned} \tag{5}$$

To further reduced computational costs, inverse Jacobian matrix is computed only once for original system at each design point and inverse Jacobians of the modified system are obtained by Sherman-Morisson formula (6).

$$\mathbf{J}_{\tilde{F}}^{-1}=\mathbf{J}_{F}^{-1}-(1+\boldsymbol{v}^{T}\mathbf{J}_{F}^{-1}\boldsymbol{e}_l)^{-1}\mathbf{J}_{F}^{-1}\boldsymbol{e}_l\boldsymbol{v}^{T}\mathbf{J}_{F}^{-1} \tag{6}$$

Finally, error estimations ε_i are combined in equation (7).

$$\varepsilon=||\,[\varepsilon_1,\ldots,\varepsilon_n]^{T}\,|| \tag{7}$$

The latter part of the metric, the AC analysis, requires linearization of the DAE system at selected design points and then transfer functions are computed. The resulting transfer functions are then simplified using methods for linear-systems simplification.

However, the simplification method of (Wichmann et al., 1999) is limited on DAE systems representing analog electrical circuits and analogue systems. The most impractical limitation preventing its use for more general multi-domain models in Modelica is that it requires calculation of *design points*, i.e. solving usually large nonlinear system of equations.

Could be this method extended to handle simplify transients of nonlinear DAE system directly (i.e., without linearization at selected *design points*)? Influence of a simplification on the equations' transient solution could be predicted by performing single Newton-Ralphson iteration of equation system (2) at different time instants of the transients and then combine the obtained error estimates as suggested by the (Wichmann, 2003). However, this kind of error estimation is much more difficult as in case of purely algebraic nonlinear system (system at steady-state), because only local integration error is estimated and the elimination of low-ranked terms often results in an unstable system. (Wichmann, 2003) does not report how this problem was solved.

6. CONCLUSION

As the complexity of the models continuously increases, it is important than contemporary modeling environments include tools which help understand and cope with these models effectively. One class of those tools are also model simplification techniques.

Models which can be built in Modelica can be very heterogeneous and include submodels from different physical domains. On contrary, most model simplification techniques require strict modeling formalism and are limited on certain physical domains. They are thus not easily applicable on most models in Modelica. As it was shown in the paper, simplification methods developed for bond-graphs can be easily adapted for simplification of Modelica's object-diagrams. However, the simplification of the model on equation level is much more difficult and there are no publicly published simplification methods known to the author which could be efficiently used for a simplification of wide variety of Modelica models.

REFERENCES

M. Andres, D. Zimmer, and F. E. Cellier. Object-oriented decomposition of tire characteristics based on semi-empirical models. In *Proceedings of the 7th Modelica Conference*, pages 9–18, Como, Italy, 2009.

T. Blochwitz et al. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of 8th International Modelica Conference*, Dresden, Germany, 2011.

F. Casella et al. The modelica fluid and media library for modeling of incompressible and compressible thermo-fluid pipe networks. In *Proceedings of the 5th Modelica Conference*, pages 631–640, Vienna, Austria, 2006.

F. E. Cellier. *Continous System Modeling*. Springer - Verlag, New York, 1991.

Samuel Y. Chang, Christopher R. Carlson Carlson, and J. Christian Gerdes. A lyapunov function approach to energy based model reduction. In *Proceedings of the ASME Dynamic Systems and Control Division – 2001 IMECE*, pages 363–370, New York, USA, 2001.

E. Eitelberg. Model reduction by minimizing the weighted equation error. *International Journal of Control*, 34(6):1113–1123, 1981.

I. K. Fodor. A survey of dimension reduction techniques. Technical report, Lawrence Livermore National Laboratory, 2002. Technical Report UCRL-ID-148494.

Modeling Chemical Reactions in Modelica By Use of Chemo-bonds. Cellier, f. e. and greifeneder, j. In *Proceedings of the 7th Modelica Conference*, pages 142–150, Como, Italy, 2009.

G. Innis and E. Rexstad. Simulation model simplification techniques. *Simulation*, 41(1):7–15, 1983.

Sanjay Lall, Petr Krysl, et al. Structure-preserving model reduction for mechanical systems. *Physica D*, 284: 304–318, 2003.

Loucas Sotiri Louca. *An Energy-based Model Reduction Methodology for Automated Modeling*. PhD thesis, University of Michigan, 1998.

Modelica Standard Library 3.1, User's Guide. Modelica Association, 2010. `https://www.modelica.org/libraries/Modelica`.

Modelica Association. *Modelica Specification, version 3.2*, 2010. `http://www.modelica.org/documents/ModelicaSpec32.pdf`.

J. D. Murray-Smith. Simulation model quality issues in product engineering: A review. *Simulation News Europe*, 19(2):47–57, 2009.

P. Schwarz et al. A tool-box approach to computer-aided generation of reduced-order models. In *Proceedings EUROSIM 2007*, Ljubljana, Slovenia, 2007.

A. Sodja and B. Zupančič. Model verification and debugging of eoo models aided by model reduction techniques. In *Proceedings of the EOOLT 2010*, pages 117–120, Oslo, Norway, 2010.

Ralf Sommer, Thomas Halfmann, and Jochen Broz. Automated behavioral modeling and analytical model-order reduction by application of symbolic circuit analysis for multi-physical systems. *Simulation Modelling Practice and Theory*, 16:1024–1039, 2008.

Hubertus Tummescheit. *Design and Implementation of Object-Oriented Model Libraries using Modelica*. PhD thesis, Lund Institute of Technology, 2002.

T. Wichmann. Transient ranking methods for the simplfication of nonlinear dae systems in analog circuit design. *Proceedings in Applied Mathematics and Mechanics*, 2:448–449, 2003.

T. Wichmann et al. On the simplification of nonlinear dae systems in analog circuit design. In *Proceedings of CASC'99*, Munich, Germany, 1999.