"hello, (Modelica) world": Automated documentation of complex simulation models exemplified by expansion valves

Christian Vering Sven Hinrichs Moritz Lauster Dirk Müller

Institute for Energy Efficient Buildings and Indoor Climate, RWTH Aachen University, Germany,
cvering@eonerc.rwth-aachen.de

Abstract

The constantly increasing computing power enables the implementation of complex simulation models. Therefore, it is possible to create more detailed models to predict system behavior more accurately. Modelica, for example, has proven great suitability in modelling complex systems, because of its high degree of reusability. However, understanding these models is quite difficult and many simulation models are poorly documented. Consequently, it is very time-consuming to retrace given model structures especially for novice. The Unified Modeling Language (UML) provides a user-friendly and graphical structure for documentation to simplify working with existing simulation models. Hence, an algorithm (ADoCSM) is developed to automatically present the structure of a Modelica simulation model in UML. This algorithm is exemplarily applied to a refrigerant circuit expansion valve model. Thereby, we contribute to an increase of simulation model quality as well as simplifying the entry in the world of Modelica. ADoCSM and the expansion valve model are freely available on GitHub:

https://github.com/RWTH-EBC/ADoCSM
https://github.com/RWTH-EBC/AixLib/tree/issue590_ExpansionValve

Keywords: Modelica introduction, simplify modelling, automated model documentation

1 Introduction

In the last decades, modelling complex systems has gained importance. In engineering, we utilize modelling in order to support designing processes or to enable the application of sophisticated control strategies like model predictive control. Therefore, detailed simulation models with high software quality are necessary. However, careful modelling is time-consuming and the documentation of models is exhausting.

In the context of building energy systems' heat supply, heat pumps are awarded to be a key technology supporting the achievement of stated climate goals in this sector (EEA, 2016). The heat pumps' lifetime strongly depends on the operation of its compressor. Avoiding droplet impact within the compression process, the heat pumps' expansion valves adjust a level of superheat of the refrigerant at the compressor inlet (Jahnke, 2000). Thus, the expansion valve is very important for the lifetime of heat pumps. Hence, modelling expansion valves is essential to understand its behavior within the refrigerant circuit and increase the lifetime of the compressor by advanced expansion valve control.

Due to superposition of thermodynamic and fluid mechanic interactions, modelling expansion valves is challenging (Cao, 2016). In particular, the complexity of superposition makes good documentation necessary. Therefore, the expansion valve modelling and concurrent documentation offers a suitable application to show functionality of the presented algorithm ADoCSM ("Automated Documentation of Complex Simulation Models"). In order to ensure both a well-defined simulation model and a well-documented one, a modelling as well as a documentation language need to be chosen.

One suitable modelling language for thermal systems is Modelica, because of its DAE-based modelling options as well as its high degree of reusability.

Many approaches for modelling of thermal systems like expansion valves already exist and are utilized in literature. Thereby, two approaches are common. On the one hand, mathematical black box approaches are applied to ensure short simulation times by sufficient prediction accuracy for individual refrigerants (Müller, 2016). Going for modular and scalable models on the other hand, grey box approaches include physical behavior by applying fundamental equations to predict change of states. As a result, prediction accuracy can be increased for a wider range of refrigerants by simultaneous loose of computational speed (Müller, 2016). Joining presented advantages, we show automated documentation of a modular and scalable expansion valve simulation model.

A well-known and well-established model documentation approach is using the graphical Unified Modeling Language ("UML") (Weilkiens, 2006). In order to utilize UML for Modelica many tools already exist (Loeffler, 2006). However, an automation algorithm is not known.

Therefore, reducing entry barriers into both the structure of Modelica as well as our algorithm, we present an open-source solution that is applicable with a graphical

user interface (GUI). Hence, we want to educate users in understanding Modelica as well as to improve our algorithm to be suitable for many use-cases as easy as possible.
Thus, within this paper, we contribute to three main ideas by breaking up rather complicated structures:

1) Modular and scalable modelling approach for expansions valves on refrigerant level is set up in Section 2, combining thermodynamic and fluid mechanic effects by consideration of
   a. metastable coefficient of mass flow and transition of two-phase flow and
   b. choke effects.
2) In Section 3, we present the algorithm that translates Modelica code into a UML code, which enables a structured and automated documentation of complex simulation models.
3) Reducing entry barriers in Modelica and the use of the algorithm, we show in Section 4 a GUI that easily offers all functionalities of the current state of our work.

In the end of the paper, we conclude the work and discuss the outlook for further developments.

2 Modeling expansion valves

Heat pumps are a key technology coupling sectors of heat and electricity regarding the building stock (Huchtemann, 2009). This enables a systematic electrification of heat supply in order to increase the flexibility of the energy system. Exploiting the whole potential of this technology, detailed simulation models are necessary to predict system behavior. Therefore, Modelica has proven great suitability.
Figure 1 depicts a schematic of a modular and scalable heat pump in Modelica (Dymola) that strictly separates physical system description from system control. Further information about this grey-box approach is published in Storek et al. (Storek, 2018) and Vering et al. (Vering, 2018).

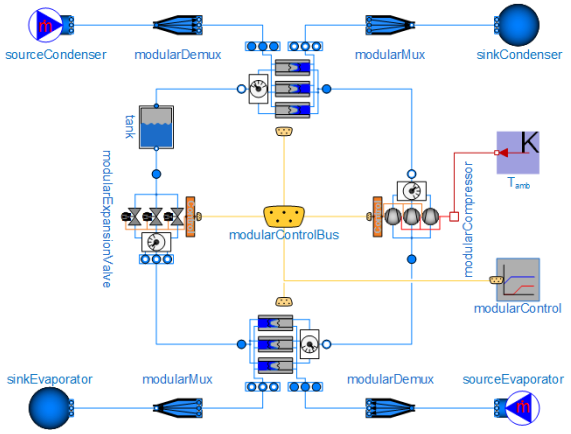


Figure 1: A modular and scalable heat pump in Modelica showing the main physical components separated from control blocks.

The expansion valve (EV) is an important component to ensure mass flow adjustment within the refrigerant circuit. The mass flow directly correlates to the refrigerant mass in the evaporator and dwell times in this component. Thus, EV allows controlling the level of superheated vapor at evaporator outlet and thereby at the inlet of the compressor.
Compressors' lifetime can be reduced by supplying it with non-superheated vapor, which causes droplet impact (Jahnke, 2000). Therefore, it is necessary to avoid those refrigerant conditions. Hence, controlling the level of superheat is important while operating a heat pump. As consequence, a resilient prediction of the fluid state at compressor inlet using simulation models requires careful modelling.
The expansion valve throttles the refrigerant from a high-pressure level to a lower one by decreasing the flow area with a positioning cylinder as shown in Figure 2. In this process, a superposition of thermodynamic and fluid mechanics effects occurs, which is a complex phenomenon (Huo, 2010). The main effects defining this process are choke, flash, cavitation and evaporation waves (Moreira, 2003). Modelling all physical interactions is not recommended in literature as well as just using the Bernoulli equation for an incompressible fluid and frictionless fluid flow (Cao, 2016).
In consistence to the whole heat pump model, we choose a grey-box modelling approach to estimate refrigerant states. Therefore, basic physical equations as well as some assumptions are covered and implemented. The mass flow $\dot{m}$ through an EV considering expansion by an expansion factor Y can be written as (Davies, 1973):

$$\dot{m} = C_\mathrm{d} Y A_\mathrm{th} \sqrt{2\,\rho_\mathrm{in}(p_\mathrm{in} - p_\mathrm{out})}, \tag{1}$$

$$Y = 1 - \frac{p_\mathrm{in} - p_\mathrm{out}}{3 F_\gamma X_\mathrm{T}}. \tag{2}$$

C_d describes the flow coefficient, A_th means flux area, ρ_in fluid density at inlet and p_i is the pressure of the fluid at inlet and outlet. $F_\gamma X_\mathrm{T}$ is the product of a specific heat ratio F_γ and a pressure drop factor of the valve X_T.

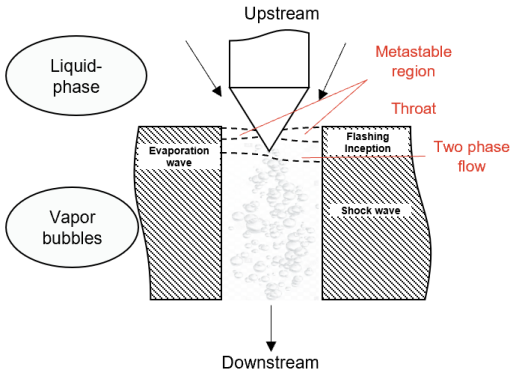


Figure 2: Schematic of fluid phenomina within an expansion valve showing the throtteling process for refrigerants with phase change.

DOI 10.3384/ecp19157709

For a fluid flow with constant density without phase change Equation 1 is simplified by $Y = 1$ to Bernoulli Equation (Li, 2013).
Modelling a choked mass flow with constant EV inlet pressure p_{in} Equation 1 can be written as

$$\dot{m} = \frac{2}{3} C_{\mathrm{d}} A_{\mathrm{th}} \sqrt{2\, \rho_{\mathrm{in}} p_{\mathrm{in}} F_{\gamma} X_{\mathrm{T}}}. \qquad (3)$$

It is obvious, that a two phase mass flow is always lower than for a one phase flow. Regarding detailed modelling approaches of the flow coefficient, we refer to (Li, 2013).
We implemented Equation 3 in Modelica to model the physical behavior of an expansion valve and the choke effect of the refrigerant considering two phase flow.
Showing the main governing equations, interdependencies between different variables occur, which are related to the reformulation of the problem. The modeler exactly knows which variable and equation stands for what. However, making a model open-source available, not only the modeler uses the model, but also end users. Simplifying the use of a model, a structured documentation is necessary. Within this work we chose UML to be a suitable graphical way of documentation because of its dissemination and establishment. Enabling the use of UML, an algorithm that translates Modelica code into an UML readable format is required. This algorithm is presented in Section 3.

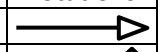
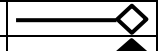
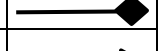
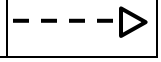
The expansion valve model is freely available on GitHub:
https://github.com/RWTH-EBC/AixLib/tree/issue590_ExpansionValve

3 Automated Documentation of Complex Simulation Models

ADoCSM stands for "Automated Documentation of Complex Simulation Models". It is a tool that scans Modelica libraries and translates them into a code, which can be interpreted by UML language. Therefore, we use the freely available tools PlantUML and Papyrus because they support 9 (Papyrus 13) different UML diagrams and they are easily extendable (PlanUML, 2018).

These tools need to be interconnected via a parser to translate Modelica main functions into UML readable and interpretable code.
The parser is based on the Python programming language and has to recognize keywords from Modelica that relate to various Modelica functions.
We started implementing four main relations "Inheritance", "Aggregation", "Composition" and "Polymorphism" that are summarized in Table 1, which are key ideas of object-oriented modelling and allow a high level of reusability. Inheritance passes all methods and attributes from a parent to the child class and is initialized with the keyword "extends". In the composition several objects are assembled into a new overall system and is initialized with the "*modelname"*. The relation polymorphism is used to specify interchangeable classes or object types that can be exchanged later in the parameterization.

Table 1: Relations in Modelica that the parser finds and translates into UML code.

Relation	Modelica key word	UML Notations
Inheritance	extends	
Aggregation	outer / inner	
Composition	model	
Polymorphism	replaceable model	

In a pre-processing step, a structured reformulation of the Modelica code decreases the parser's error rate. Hence, we introduce two steps before starting the parser. In a first step, the path of the Modelica model is defined. This allows a systematic naming of packages and classes. After that, we translate a well-defined Modelica code into an easily parser readable formulation. Therefore, we choose a structure that follows these designs:

<< Variable >> << Comment >> << Annotation >>
<< Method >> << Comment >> << Annotation >>

Using the new code, the workflow of the parser is shown in Figure 3. Further reducing error rates, there is a first check whether a library exists. After that there is a lookup for the existing files.

Within this, file packages are declared. A package with its models is taken to be translated into the UML notation. Therefore, different key words are translated into a PlantUML format:

- Method
- Attribute
- Stereotype (model, record, type, block, function, connector)

Using these three inputs, the parser defines all relations within this model and repeats these steps for the whole list of files.
After passing all steps, the parser analyses the next layer i. The user defines the number of layers beforehand. Finally, the Modelica code is translated into a UML readable notation that can be interpreted e.g. by PlantUML. The converted Modelica code can now be loaded into the PlantUML tool and will be graphically generated in the UML form.
Thereby, the model structure can be broken up in order to get an overview of the model.

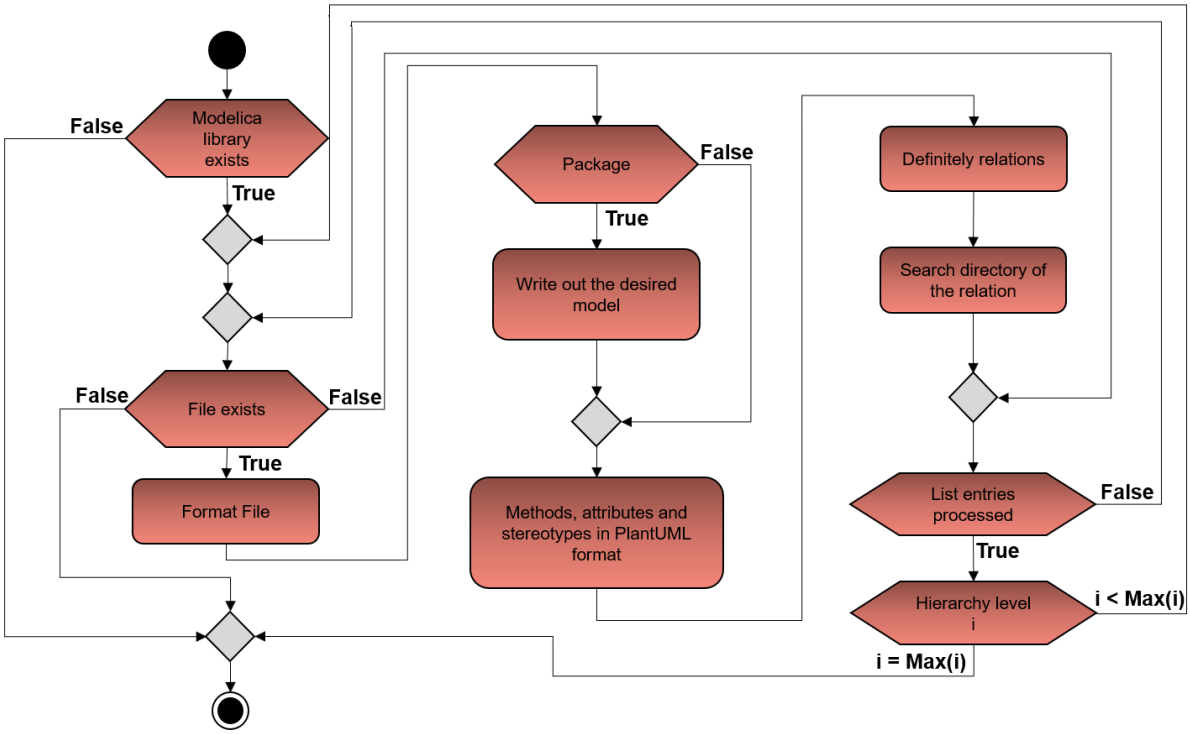


Figure 3: Workflow of the parser translating Modelica code into an UML readable notation.

With the aid of the UML class diagram, the structure of the model can be displayed graphically. For example, inherited models, parameters or exchangeable classes can be displayed. The visual representation reduces the complexity of the model and the relationships and composition of the model can be detected more quickly. Furthermore, simplifying the use of this algorithm and disseminating it, we develop a graphical user interface (GUI) that is presented in the next section by applying UML documentation to the expansion valve simulation model.

4 Use of ADoCSM

Supporting the entry in the world of Modelica, by making the structure of Modelica models more transparent, the parser translates Modelica code into the UML notation. For a user-friendly design, we develop a graphical user interface (GUI).

The GUI of ADoCSM is mainly separated in six parts, which are shown in Figure 4. These are initialization (1), settings (2-5) and console (6).

For “initialization” the input file (1) has to be chosen. With this file the paths of the files as well as libraries are defined for the parser. This example uses the *AixLib* (D. Müller, et al. 2016) and the Modelica standard library in the Model Library directory.

Additionally, the result file path can be redefined.

After that, the “settings” for the parser are selected. The user can choose, which information e.g. parameter, constants or variables (2) of a model should be plotted within the UML representation. In particular for complex models with a large number of parameters, constants or variables, it is helpful to show or hide a specific group of attributes in order to retain the overall overview. In particular, depicting model structures with a high degree of polymorphism, whole packages can be superimposed or hided out to keep overview. Additionally, the user immediately learns, which packages can be replaced by other ones.

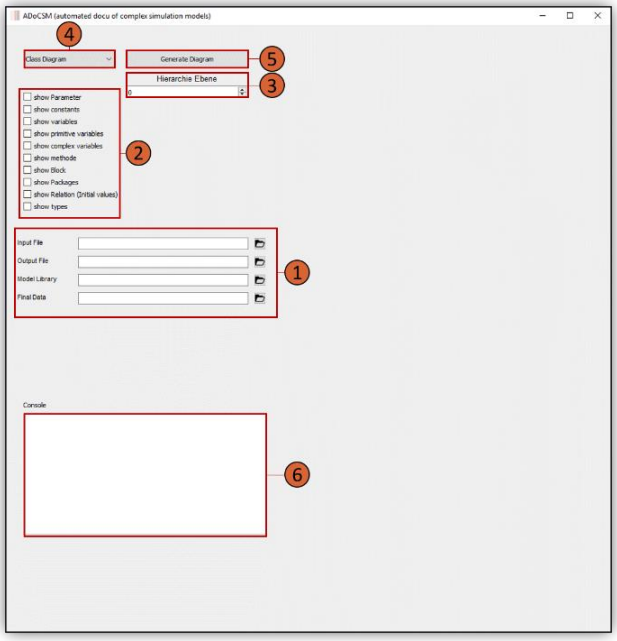


Figure 4: Graphical User Interface (GUI) to simplify the use of ADoCSM.

DOI 10.3384/ecp19157709

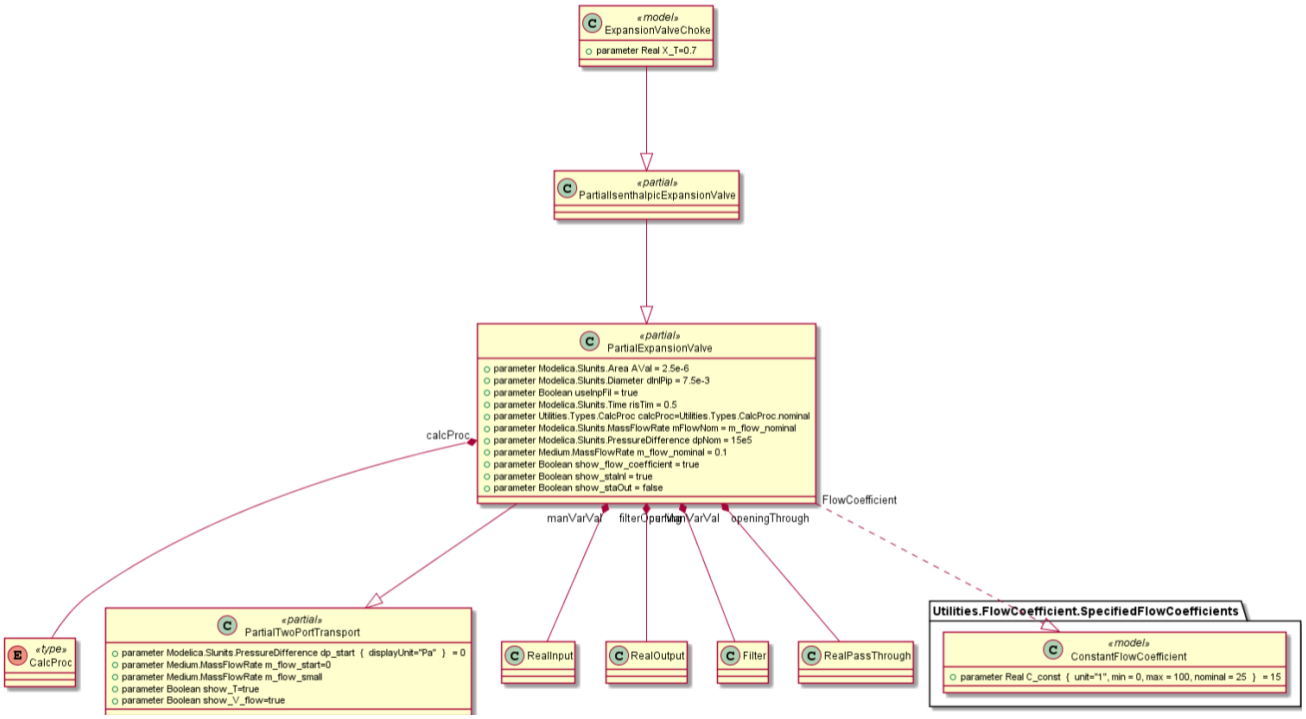


Figure 5: Representation of an expansion valve in an UML class diagram that was translated from Modelica code showing the model structure and the relations.

Regarding a structured representation, the number of investigation layers (3) is set and UML diagram type (4) can be chosen.

By default, a class diagram is predefined. For further work, it should be considered whether additional diagram types should be added. Pushing the button (5) executes the generation of the UML structure regarding defined paths and settings. Using the expansion valve as example, in Figure 5 the referring UML class diagram is depicted with all relations and model parameter. The "console" (6) shows in the end, if a workflow was successfully executed or if errors occurred. A successful execution is then shown in a separate window.

In Figure 4 all structure investigation layers are depicted. On top, in the first layer there is the first box of the expansion valve model with respect to choke effects. The box is regarding UML class diagram requirements divided into three parts. In the upper one the indication is shown as a "model" named "ExpansionValveChoke". In the lower one the user can see the corresponding parameter X_T. The third box is not required in this case, because no operators are defined within this model. It inherits its properties from the next layer, where the partial model "PartialIsenthalpicExpansionValve" is shown.

On the next deeper layer, "PartialExpansionValve" with all typical properties of an expansion valve is illustrated. Different parameter such as "Area" or "Diameter" are defined with a value and the corresponding SI unit. Additionally, parameter of the type Boolean like "useInpFil" are shown.

In the last layer, the inheritances, compositions and a polymorphism are pointed out. The only inheritance of a "PartialTwoPortTransport" model is revealed. A combination of "CalcProc", "RealInput", "RealOutpt", "Filter" and "RealPassThrough" constitutes all model compositions. Furthermore, a polymorphism "ConstantFlowCoefficient" is shown. As consequence, the user immediately knows, that this is a replaceable model within the "ExpansionValveChoke" model.

Compared to typical Modelica tree structures, the UML class diagram easily illustrates the expansion valve simulation model structure. That supports the understanding of inheritances and compositions as well as polymorphism. Additionally, parameter and initial values are revealed by the graphical representation of the model.

All these points simplify the understanding of complex simulation models. As consequence, training times of model structures can be reduced and simultaneously the code quality is increased by better documentation.

The algorithm is freely available on GitHub. We kindly invite external users to use and improve the functionalities of ADoCSM by online cooperation using this link:

https://github.com/RWTH-EBC/ADoCSM

Improving the quality of the algorithm and doing systematic troubleshooting, we will apply ADoCSM to the AixLib (Müller, 2016 - 2). Thus, on the one hand, the model quality increases due to documentation improvements. On the other hand, the algorithm is tested and error rate of application will be reduced.

5 Conclusion

Within this work we show that Modelica code can be translated into UML code via the presented tool that is freely available on GitHub:
https://github.com/RWTH-EBC/ADoCSM.
It allows to visualize the model structure and the parameter interdependencies, which is a powerful tool increasing the understanding of Modelica.
Using the recent version, we show for an expansion valve example that it is possible to create UML diagrams considering both important key words and relations of the model.
Our presented expansion valv simulation model for refrigerant circuits is freely available on GitHub:
https://github.com/RWTH-EBC/AixLib/tree/issue590_ExpansionValve
The model considers idealized expansion valve functionality as well as chocked mass flows for different pressure to pressure drop ratios.
All in all, we show functionality of both the modelling approach and of the parsers algorithm. The next steps will consider the investigation of further models to start systematic troubleshooting in order to increase the quality of our algorithm. Therefore, AixLib will be the first large use-case.

References

European Environment Agency (EEA). "Trends and projections in Europe 2016", 2016.

A. Jahnke. Webasto Schulungs-Handbuch: *Kälte-Klima*, 2000.

X. Cao, Z.-Y. Li, L.-L. Shao und C.-L. Zhang. Refrigerant flow through electronic expansion valve: Experiment and neural network modeling. *Applied Thermal Engineering*, 92:210–218, 2016.

D. Müller. Simulationsmodelle für die Heiz- und Raumlufttechnik: Heizflächen. *Vorlesungsvortrag, RWTH Aachen University, Aachen*, 2016.

M. Loeffler, Michaela Huhn, Christoph Richter, Roland Kossel. Modelica CDV A Tool for Visualizing the Structure of Modelica Libraries, *In The 5 International Modelica Conference*, pp. 55-62., 2006.

T. Weilkiens. Sysml: Ein neuer Standard der omg. omg-Kolumne, pages 12–15., 2006.

K. Huchtemann und D. Müller. Advanced simulation methods for heat pump systems. *In The 7 International Modelica Conference*, 798–803. Linköping University Electronic Press, 2009.

T. Storek et al. A modular modelling approach for thermodynamic systems applied to heat pumps. *In 31st ECOS Conference*, 2018.

C. Vering et al. Transiente Modellierung eines Verdichters zum Vergleich von niedrig GWP-Kältemitteln für Kompressionswärmepumpen. In *BauSIM* 2018.

M. Huo. A study in the characteristics of the flow inside a thermostatic expansion valve, Master thesis, University of Illinois at Urbana-Champaign, 2010.

J. R. Simões-Moreira, C. W. Bullard. Pressure drop and flashing mechanisms in refrigerant expansion devices, *Int. J. Refrig*., 26:840–848, 2003.

A. Davies, T.C. Daniels. Single and two-phase flow of dichlorodifluoromethane, R12 through sharp-edged orifices, *ASHRAE Transactions 79 (Part 1) (1973)* 109-123., 1973.

W. Li. Simplified modeling analysis of mass flow characteristics in electronic expansion valve, *Applied Thermal Engineering*, 54:8-12, 2013.

Drawing UML with PlantUML: Language Reference Guide (Version 1.2018.2)., http://plantuml.com/PlantUML_Language_Reference_Guide.pdf, 2018

D. Müller, et al. AixLib – An open-source Modelica library within the IEA-EBC Annex 60. In *BauSIM*, 2016: 3-9, 2016.

DOI 10.3384/ecp19157709