MOSAIC, AN INTERNET TOOL FOR CO-OPERATIVE MODELING IN THE DOCUMENTATION LEVEL AND CODE GENERATION

Stefan Kuntsche, Robert Kraus, Harvey Arellano-Garcia, Günter Wozny

Chair of Process Dynamics and Operation, Technical University of Berlin
KWT-9, Straße des 17. Juni 135, 10623 Berlin

Apart from using ready-made models that are integrated in software tools like Aspen Plus or Chemcad it is often necessary to create customized models. Custom modeling in CAPE is at a great extend concerned with assembling a suitable set of model equations and implementing them in a computer language. There are many software tools and different languages that allow for such an implementation. The computational description of the equation system differs in detail from language to language, and also the capabilities of the respective underlying numerical environment are different. The starting point of the formulation in a computer language, however, is always a well formulated equation system that can and should be laid down inside a documentation during the modeling process. In the long term, the abstract formulation in the documentation is more valuable for the engineer than its implemented pendant – e.g. when numerical software tools are updated and old models lose their compatibility after a certain time, which may even lead to a reimplementation of the model equations in a new version of the same software tool (cf. Eggersmann et al. (2004)). If the documentation-level formulation is indeed the real asset (cf. Bausa and Dünnebier (2005)) than there should be a modeling environment that supports modeling with such formulations, i.e. a modeling environment that works with symbolic mathematic formulations instead of program code. Therefore, a new modeling environment for modeling in the documentation level has been developed and is presented in this contribution. Its modular modeling approach and reuse capacities are briefly described. The new environment is named MOSAIC and allows flexible and user-controllable code generation in many programming and documentation languages. Furthermore MOSAIC is implemented as a web application, which lowers maintenance efforts and makes the tool well suited for cooperation over the Internet.

1. INTRODUCTION

In both the research and development sector and the educational sector, cooperation on the internet becomes increasingly important. The personal workforce is locally decentralized and information technology is used to coordinate the collaboration and to merge the respective results into large research projects. One key issue in this process is the centralized access of cooperative results in knowledge databases. This allows for the use and the reuse of information created by collaborating researchers. The used set of mathematical equations describing process systems engineering related phenomena and their assembly into models for different applications is an important part of this sharable knowledge. This kind of knowledge is traditionally shared via public or internal publications i.e. the mathematical formulations are exchanged in the documentation level. In the documentation level, model equations and functions are laid down in two-dimensional symbolic mathematic language as defined further below. The implementation of such custom models on one of the many available tools in a numerical model that allow the solving of a related modeling problem by a computer. For this implementation it is necessary, however, to translate the model given in some kind of documentation (e.g. hand-written formulations, computerized documentation in Word or Latex, equations given in an article or in a book) into the programming language of the respective tool. This translation step takes resources from the engineer and it is

prone to errors. Furthermore, the resulting numerical model cannot be exchanged unless the cooperating workers use the same numerical environment, which represents a considerable limitation for collaboration.
In this work, a new modeling environment for the creation, administration and translation of custom models is presented that addresses the above (cf. Kuntsche et al. (2010)). The proposed modeling environment is named MOSAIC and it is provided as a web application. The tool consists of a modeling and evaluation environment that is implemented as a Java Applet and a modeling server that allows for collaborative storage and controlled access of the entered model equations, equation systems and case studies. The environment can be integrated in other Cloud Computing applications.
The mathematical and structural content of the models is stored in XML and MathML. To further the reuse of models, the new modeling environment provides a very high extent of modularity. Equations are defined in separate modular structures that are independent from equation systems. An equation system is built from modular equations or from other equation systems. In subsequent steps the equation systems are used to fully define numerical problems, for which program code is generated, see figure (1).
There are two ways of generating program code. One is using the built-in language specifications, which allows for the generation of code for the BzzMath Library (C++), see Buzzi-Ferris and Manenti (2010), MATLAB, gPROMS see Oh and Pantelides (1996), Microsoft Excel, and others. A second way is the specification of a goal language by the user. The translation of all aspects of a MOSAIC XML/MathML model can be user-defined in a model element called Language Specificator. Such Language Specificators can be shared and adapted according to each user's needs.

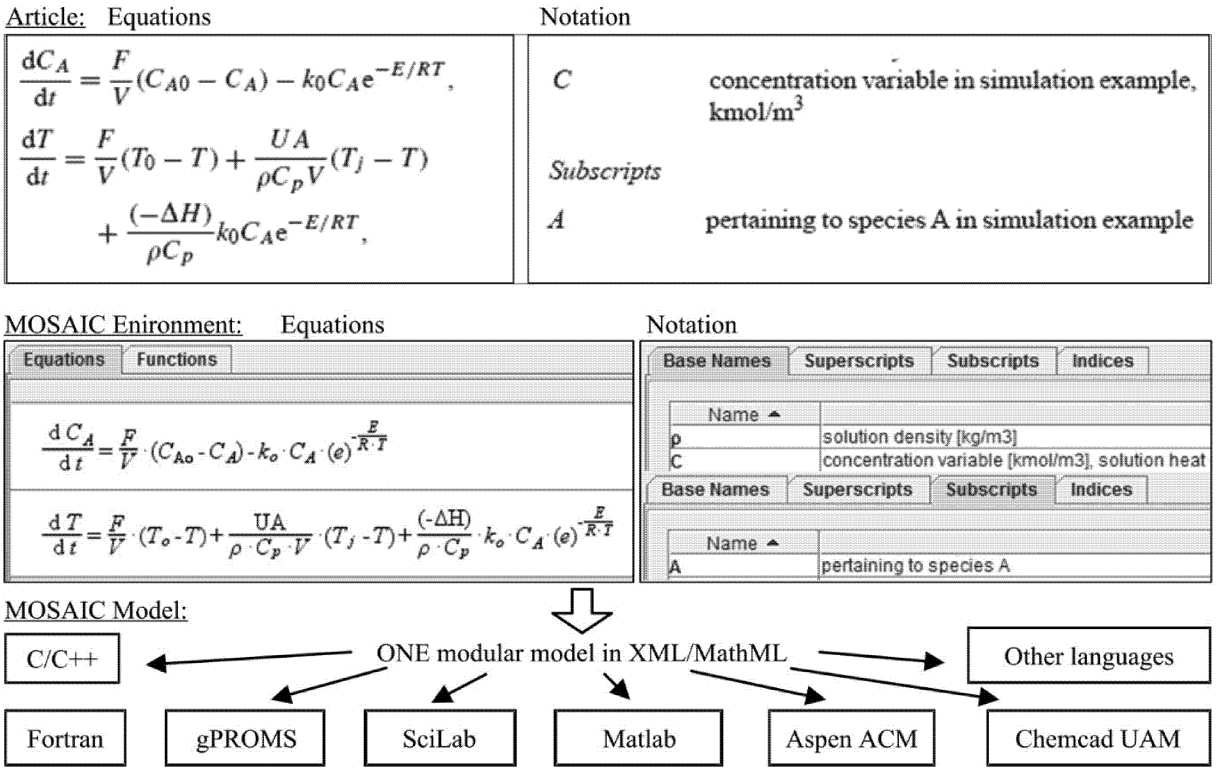

Fig. 1: Top and Middle: Presentation of model equations in the literature and in MOSAIC. In both representations a notation provides information on the meaning of the variables. Middle and Bottom: The mathematical model from the documentation level is translated into program code for different modeling languages.

The proposed modeling environment MOSAIC tries to narrow the gap between the model description in the documentation level and the numerical model. This is done by using two-dimensional symbolic expressions for the mathematical content and by considering the notation as separate and mandatory model element belonging to equations and equation systems. The notation carries descriptions for all identifiers that are present in the equations. More importantly, the notation will carry information about engineering units. Applicable data structures allow for the translation of a model's variable name and engineering units from one notation to another.

2. MODELING IN THE DOCUMENTATION LEVEL

Modeling in the documentation level can mean several things. In the case of MOSAIC the term is used due to technical and structural reasons. The technical reasons are the use of documentation standards instead of calculation standards. In MOSAIC, MathML in the Presentation Markup (not in the calculation-oriented Content Markup, see also www.w3.org/TR/MathML2) is used for storage and as a meta description for code generation. Latex is used to enter the mathematical content more easily than creating MathML directly. It is also possible to import from Microsoft Word equations. Presentation MathML and Latex (cf. Knuth (1986)) are clearly documentation standards. The structural reason is first the fact that variable names in MOSAIC may consist of several character symbols that may also be superscripted or subscripted (see below). All of such symbols must be described in a nomenclature element that is called notation. This feature closely reflects the model description in papers or books. Creating models in this way can be considered as modeling in the documentation level since it uses the concepts two-dimensional variable names and notations as description for the symbols used therein.
However, we do not try in this project to define whole models in files that are by their format really intended for documentation purposes. The premises of offering a high modularity of the model parts and a good reuse functionality are in conflict with storing the models in such documents. Therefore the model parts are stored in XML and Presentation MathML, which can be considered as a meta format using documentation techniques (see above). This meta format, however, contains all documentation-intended, mathematical and numerical information at the same time and allows the fast and flexible creation of program code for evaluation (such as gPROMS) on one hand and code in the desired documentation format (such as Latex) for the full model on the other hand.

2.1 Mathematic expressions in two-dimensional symbolic form

As mentioned above, the modeling strategy used in MOSAIC focuses on the symbolic representation of the mathematical content. The mathematical expressions for calculation are given in two-dimensional symbolic form and are rendered and presented as they are in a paper, e.g.

$$x_{i=1} \cdot \gamma_{i=1} \cdot p_{o,i=1}^{L,V} = y_{i=1} \cdot p_{sys} \tag{1}$$

Whereas, a large class of modeling languages and respective environments use one-dimensional textual expressions with a restricted character set for the variable names and the operations (mostly a subset of ASCII), e.g.

```
x1*gamma1*pLiqVap1=y1*psys
```
(2)

The above equations represent the same information in two forms of display. Usually the one-dimensional expression is created when the two-dimensional expression found in an article or book is programmed in a language of the modeler's choice or availability. However, this programming step has some disadvantages: It is time consuming, not very demanding, and yet error-prone. Most modelers will agree at this point that many problems that are encountered during the numerical evaluation of equation systems are caused by typing errors, small mistakes like interchanged indices in loops, etc.

The MOSAIC modeling environment generates the full program code automatically. The user can influence many aspects of the code generation so that even languages can be supported that are not yet developed. The advantage of modeling in the documentation level with subsequent user-defined code generation is that the same symbolic model can be used in different numerical environments without additional programming effort.

2.2 MOSAIC vs. other programming tools using two-dimensional symbolic expressions

There are several programs for mathematical modeling that allow the use of two-dimensional symbolic expressions, among them MapleSim and MathCad, but there are some differences in the way symbolic expressions are supported by MOSAIC: First, MOSAIC is restricted in its scope to algebraic and ordinary differential equations (including systems built from such equations). It is also possible to define and use functions in MOSAIC, which is useful to reduce the size of the equation systems. Matrices and integrals are not supported. Second, the variable names in MOSAIC are not restricted to a character string; they may have superscript and subscript elements, as is common in the literature. Third, the meaning of the variable names is explained by a Nomenclature (in the following: Notation) that is an additional mandatory (and modular) model element.

The consideration of superscripts and subscripts in combination with a Notation specification as documentation element does not exist in other current tools for symbolic modeling. To explain that point a bit further, an example is given: A variable name in MOSAIC could be

$$p_{o,i=1}^{L,V} \tag{3}$$

In MOSAIC every equation needs a specified Notation that must be used to describe the variable names. An equation that contains the name given above would have to use a Notation that contains at least the following information (see table 1):

Table 1: Notation information for variable naming shown in (3)

Base names	Superscripts	Subscripts	Indices
p : pressure	L : liquid phase V : vapor phase	o : reference condition	i : component index

In MOSAIC this Notation information is available to the modeler at every modeling step, i.e. whenever the variable bearing that name is used. It can also be included as comment in the generated program code to improve the comprehension.

3. MODULARITY IN MODELING

Every advanced modeling approach supports a modularization of the models. The simplest way to think of this is the use of subroutines in pure programming languages such as FORTRAN. However, equation oriented programming environments like gPROMS also offer powerful ways to create small model elements and connect them in different ways. Object orientation is now supported by the advanced modeling languages such as Modelica and gPROMS. In general, the modules contain model parts or calculations that are likely to be reused or that belong to a context that the modeler wants to separate from the rest of the model. This reuse is an intention of the modularization, which aims to save time, reduce tedious reprogramming efforts, and minimize modeling errors introduced during the programming. The second intention, the separation of contextually different model parts, is aimed at structuring the models and making them more accessible for maintenance or extension by the creator of the model or by colleagues.

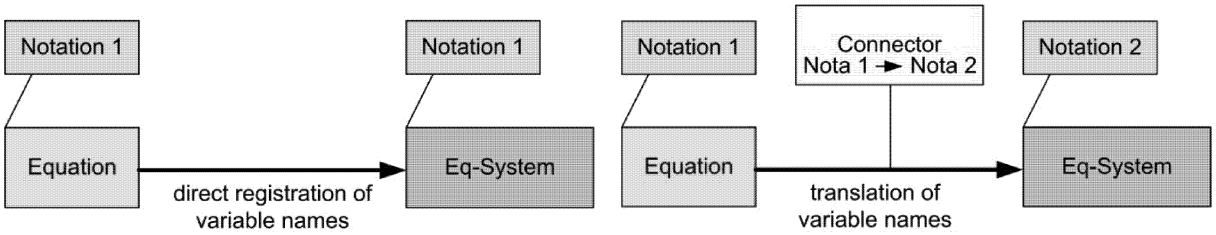

Fig. 2: Connecting an equation to an equation system. Left: both elements (equation and equation system) use the same notation, therefore the variables can be registered directly. Right: The elements use different notations, therefore a connector is used to translate between the two naming conventions.

The modularity of MOSAIC is very high. The standard basic model element is the equation. Each equation is stored separately and can thus be used in different contexts. The modeling technique of most existing modeling languages or environments of typing a set of equations and variable specifications into one file is not possible in MOSAIC. More specifically, in MOSAIC an equation system is created by combining several modular equations or other equation systems. In this connection process the name of the variables and the respective notations are considered. There are several ways to connect equations to equation systems. In the simplest connection strategy, if the equation system and the connected equation use the same notation, the variables bearing the same name are considered as one variable. If the connected equation has a different notation, the model element Connector describes the correlation of the variable names, see figure (2). With the MOSAIC modeling strategy based on modular equations, the focus is put on the content and correctness of every single equation. At the same time the user creates a personal reusable data base of equations and equation systems. The reuse concept of the modular model elements is motivated in figure (3)

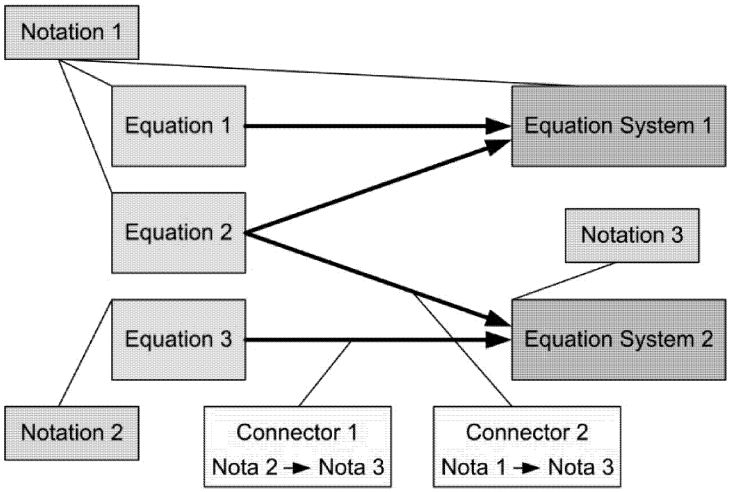

Fig. 3: A more complex scenario demonstrating the reuse of modular equations and equation systems that are created by different users. The use of notations provides coarse grained conceptual meaning to the variables. The translation between two different notations is done by the help of connectors.

Until now the modular creation of equation systems out of equations and other equation systems was described. However, the equation system does not contain enough information to constitute a modeling problem: at least values for the design and iteration variables have to be designed and a numerical strategy has to be assigned. In MOSAIC the equation systems contain as view information as possible: First, the maximum values for the variable indices (e.g. 'NC=3' when index i=1..NC) are not stored in the equation system. This allows for the scaling of the equation system in a later step and thus for a flexible use in different applications. Second, it is not

specified in the equation system which variables are design and which are state or iteration variables. Such information is provided in the evaluation step. Once all necessary information is assembled to define a problem, the code generation software within MOSAIC creates program code according to a language specification that has been chosen by the user, see figure (4). It is important to note that the language specification is also a modular model element, which can be created, modified and shared by the users.

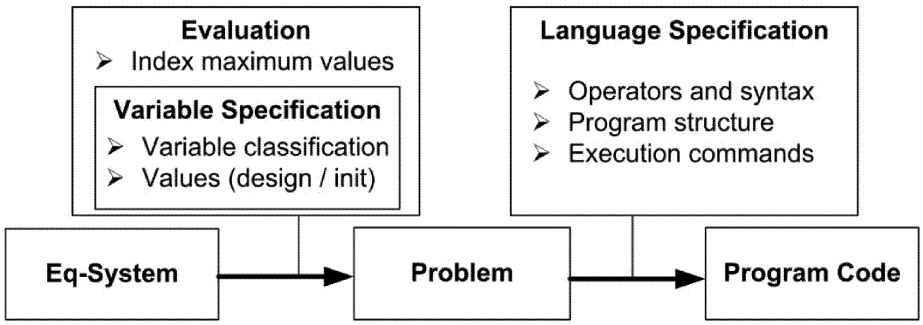

Fig. 4: Modular model elements that define a numerical problem and govern code generation in MOSAIC. The equation system can be used in several evaluations (different scaling of indexed variables, different variable values for problem specification). The resulting problem can be translated into different forms of program code according to the applied language specification.

4. USER-DEFINED CODE GENERATION

One important aspect of this project is the modeling in a meta level to allow cooperation of engineers that use different numerical environments. To achieve this aim, MOSAIC is designed as a domain specific code generator. The model information that is stored in XML/MathML is transferred into program code expressions. The translation process is controlled by the help of an additional model element, the Language Specificator. The Language Specificator contains the information about how to translate the structural aspects of a NLE or DAE problem given by the concept described above. This specification contains basic parts like the translation of mathematic operations like addition, subtraction, multiplication, etc., but also instructions on how the entire code should be organized. It is also possible to produce code that consists of several communicating methods. The Language Specificator itself is stored as an XML document and the MOSAIC modeling environment provides a user interface to create or modify this model element. The user definition of the Language Specificator works without fixed keywords to avoid clashing with keywords of output languages.
By the help of this versatile, user-defined code generation, the models created in MOSAIC can be used in different numerical environments. Thus, the cooperation of different research groups is less dependent on the modeling software used by the different parties.

5. EXAMPLE OF USE

To show the usage of MOSAIC, a semi-continuous fermentation is implemented as published in Asprey et al. (2002). The equation system is presented in Figure 5. The equations and the notation have been created after the implementation of the model by the help of the documentation function of MOSAIC. This shows how close the modeling procedure is to the documentation level. Rendered variables and equations are used, which can always be identified through the compulsory notation.

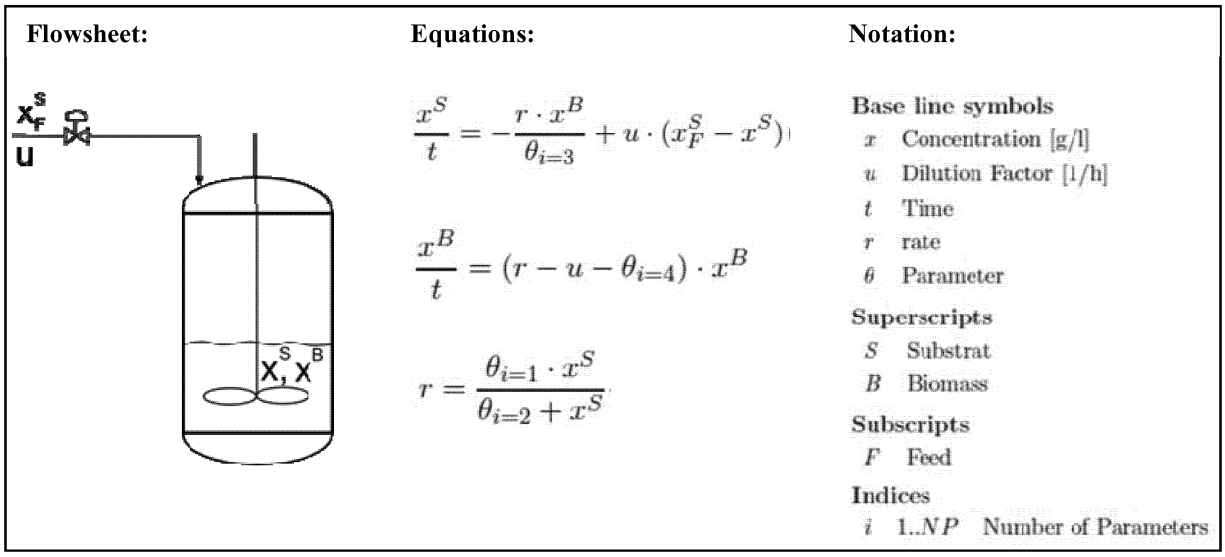

Fig. 5: A fed-batch reactor

The workflow to implement the DAE's of the system in MOSAIC is as follows

1. Definition of the notation – all naming elements, such as base names, indices, super- and subscripts, have to be specified
2. Entry of the equations – based on the naming elements in the notation, equations are created
3. Combination of the equations to an equation system – the single differential and algebraic equations are connected to a complete equation system
4. Specification of the indices, the design and initial values – see Figure 6

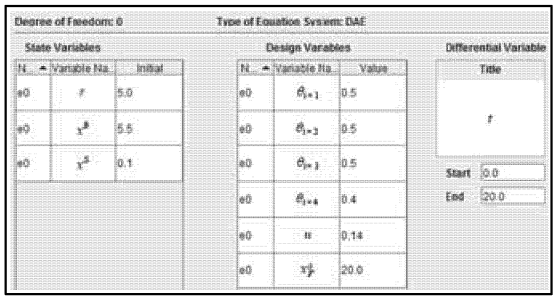

Fig. 6: Variable specification in MOSAIC

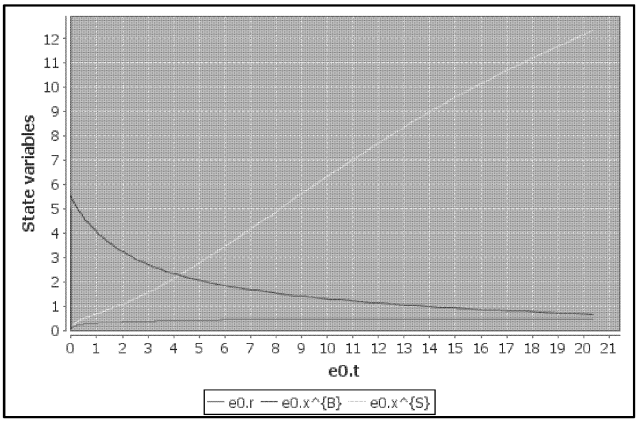

Fig. 7: Solution in MOSAIC by using the BzzMathDAE solver

After the model is fully specified, the user has two options. He can solve the equation system directly in MOSAIC, or he can create code for the local execution. The result of a solution in MOSAIC is shown in Figure

7 where a DAE solver from the BzzMath numerical package was used. If the solution of the system is not enough, the validated model can be exported to other environments, e.g. to gPROMS where it can be used to create a process flow sheet. A brief list of supported programming languages is shown in Figure 1. As already described in section 5, user-defined code can be created by the user. Thus it is possible to translate models into programming languages that are not directly supported by MOSAIC.

6. RELATED WORK

MOSAIC represents a combination of several concepts that have been focused upon by other groups. One of the most important characteristics in the project MOSAIC is combining the concepts in an Internet tool that is practical to use. Currently there are about 30 users that learned the work with MOSAIC mainly by the help of the user's guides.
A closely related topic is the symbolic programming of computers, which has a long tradition, see e.g. Klerer and May (1965). Another topic is code generation from symbolic expressions, cf. e.g. Klerer (1992) and Kajler and Soiffer (1998). Modeling in the meta level in combination with code generation has been done in many projects. Bogusch et al. (2001) presented a non-internet modeling environment that provided many modeling assistance features and allowed code generation for the two tools gPROMS and Speedup. Westerweele and Laurens (2008) presented a modeling tool that provides code generation into many languages but is not internet based and does not allow the user to create own model equations. Another interesting approach for documentation-level modeling, where the model information is stored directly in Microsoft Office documents, is presented by Alloula et al. (2010).
Great efforts are taken in the field of reuse, cf. Yang et al. (2008). The main idea is to reuse ready-made and implemented models and combine them via CAPE-OPEN (cf. e.g. Belaud and Pons (2002), www.colan.org). Suitable ontologies as proposed by Morbach et al. (2009) are used to reflect the conceptual aspects of CAPE models, their creation and use into a computerized form. A model reference database that is placed in the Internet provides information on implemented models including their respective URL for download. The combination of ontologies, cape open, and a well maintained database should allows to access these aspect by information technology to efficiently find available models.

7. ACKNOWLEDGEMENTS

This work is supported by the Cluster of Excellence 'Unifying Concepts in Catalysis' coordinated by the TU Berlin and funded by DFG (Deutsche Forschungsgemeinschaft).
This work is part of the Collaborative Research Centre "Integrated Chemical Processes in Liquid Multiphase Systems" coordinated by the Technische Universität Berlin. Financial support by the Deutsche Forschungsgemeinschaft (DFG) is gratefully acknowledged (TRR 63).

8. REFERENCES

Alloula, K., Belaud, J.-P.& Le Lann, J.-M., 2010, Solving CAPE models from Microsoft Office applications, Comp Aided Chem Eng, 28, 679-684

Asprey, S. P. & Macchietto, S., 2002, Designing robust optimal dynamic experiments, Journal of Process Control, vol. 12, iss. 4, 545–556

Bausa, J. & Dünnebier, G., 2005, Durchgängiger Einsatz von Modellen in der Prozessführung, Chem Ing Tech, vol. 77, 12, 1873–1884

Belaud, J.-P. & Pons. M. 2002, Open software architecture for process simulation: The current status of CAPE-OPEN standard, Comp Aided Chem Eng, vol. 12, 847–852

Bogusch, R., Lohmann, B. & Marquardt, W., 2001, Computer-aided process modeling with MODKIT, Comp Chem Eng, vol. 25, 963–995

Buzzi-Ferraris, G. & Manenti, F., 2010, Fundamentals and Linear Algebra for the Chemical Engineer: Solving Numerical Problems, Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany

Eggersmann, M., Wedel, L. v. & Marquardt, W., 2004, Management and reuse of mathematical models in the industrial design process, Chem Eng Tech, vol. 27, 1, 13–22

Kajler, N. & Soiffer, N., 1998, A Survey of User Interfaces for Computer Algebra, J Symb Comp, 25, 127-159

Klerer, R. J., Klerer, M. & Grossman, F., 1992, A language for automated programming of mathematical applications., Computer Languages, 19, 3, 169–184

Klerer, M. & May, J.,1965, Two-dimensional programming, Proceedings of the 1965, fall joint computer conference, part I, 63-75

Knuth, D. E., 1986, The TEXbook. Computers and typesetting, Addison Weslay.

Kuntsche, S., Arellano-Garica, H. & Wozny, G., 2010, A New Modeling Environment Based on Internet-standards XML and MathML, Comp Aided Chem Eng, 28, 673-678

Morbach, J., Wiesner, A. & Marquardt, W., 2009, OntoCAPE--A (re)usable ontology for computer-aided process engineering, Comp Chem Eng, vol. 33, 1546–1556

Oh, M. & Pantelides, C. C., 1996, A modelling and simulation language for combined lumped and distributed parameter systems, Comp Chem Eng, 20, 6/7, 611–633

Westerweele, M. & Laurens, J., 2008, Mobatec Modeller - A flexible and transparent tool for building dynamic process models, Comp Aided Chem Eng, vol. 25, 1045–1050

Yang, A., Braunschweig, B., Fraga, E. S., Guessoum, Z., Marquardt, W., Nadjemi, O. et al., 2008 A multi-agent system to facilitate component-based process modeling and design, Comp Chem Eng, 32, 2290–2305