AIDIC *Conference Series*, Vol. 10, 2011
Editor Sauro Pierucci
DOI: 10.3303/ACOS1110023

ISBN 978-88-95608-58-7 ISSN 2036-5969

MODELLING IN THE DOCUMENTATION LEVEL USING MOSAIC AND NUMERICAL LIBRARIES

Robert Kraus[1], Victor Alejandro Merchan[1], Stefan Kuntsche[1], Flavio Manenti[2], Guido Buzzi-Ferraris[2], Günter Wozny[1]

[1] Technische Universität Berlin, Dynamics and Operation of Chemical Plants
Sekr. KWT-9, Str. d. 17. Juni 135, 10623 Berlin, Germany

[2] Politecnico di Milano, CMIC dept. "Giulio Natta"
Piazza Leonardo da Vinci 32, 20133 Milano, Italy

In the today's constellation of globally operating enterprises, the workforce is locally decentralized. Engineering tasks are increasingly divided up under several work groups that are situated at different premises. Information technology is used to coordinate the collaboration and to merge the results. Due to its versatility and adaptability, the cooperation on the Internet becomes increasingly important. One key issue in this development is the placement of applications in the Internet (cloud computing). Another important issue is the centralized access of cooperative results. Such centralized information may be stored in internet-based knowledge databases. With regard to the above developments, it seems to make sense to have a modelling environment working as a web application connected to a model database. Such a modelling environment should be linked to a strong numeric library, which should contain reliable and very performing algorithms for solving ordinary differential equation (ODE) systems and differential and algebraic equation (DAE) systems. Structural informations like the sparsity pattern and the structure of the Jacobians should be automatically recognized to automatically select the best set of algorithms to integrate the differential system. By the help of these measures the computational time should be reduced significantly. In this contribution the MOSAIC modelling environment and the BzzMath library are presented and some main characteristics are described.

1. INTRODUCTION

Usually both the modelling software and the software to access company-wide shared databases are installed on the local computer. In the modelling environment MOSAIC (Kuntsche et al., 2009) both these software parts are integrated in one web application. Web applications and cloud computing have several organizational and economic advantages, cf. Linthicum (2009). With respect to modelling, the focus of MOSAIC lies on the creation and shared (re)use of customized models for process systems engineering. The modelling is done in the documentation level and code generation is used to create program code that is suitable for the environment the modeller needs. A considerable improvement of the workflow could be achieved by using the BzzMath numeric library (Buzzi-Ferraris, 2010a) to solve the models on the server. The BzzMath Library is a comprehensive tool covering several fields of numerical analysis. For the specific case of interest, it includes a set of algorithms to integrate differential and differential-algebraic systems particularly efficient according to their stiffness, dimension, sparsity, and Jacobian structure, if any.

1.1 The literature aspect of documentation-level-modelling

One aspect of modelling in the documentation level is the orientation to the representation of equations in papers or documents for reference and communication. In such documents the equations are very often represented in two-dimensional symbolic form and the variable names follow a certain systematic, e.g. a letter 'x' on the base line might stand for molar fraction and the superscript letters 'L' and 'V' adjoined to a given 'x' specify if the

Please cite this article as: Kraus R., Merchan V.A., Kuntsche S., Manenti F., Buzzi-Ferraris G. and Wozny G., (2011), Modelling in the documentation level using mosaic and numerical libraries, AIDIC Conference Series, 10, 207-214
DOI: 10.3303/ACOS1110023

variable is a liquid or a vapour molar fraction. This technique is widely used in the documentation of models since it allows for short formulations and clearly understandable variable names. Usually the meaning of the symbols is explained in a separate section that can be called 'notation' or 'nomenclature'. This 'notation' section is especially important since different naming conventions exist, e.g. with 'x' and 'y' for liquid and vapour molar fraction respectively. In MOSAIC the modelling is done in the spirit of this situation by three peculiarities: First the equations are entered in two-dimensional symbolic form. Second the variable names may contain superscripted and subscripted symbols (countable subscripted symbols represent indices). Third the 'notation' is introduced as a mandatory model element that works as distinguishing property during reuse, gives guidance during modelling, and complements the documentation output. The first of the above items is well established (cf. Kajler and Soiffer, 1998) especially but not only in algebra tools. The second item is rarely provided and the third item is not at all considered in the current standards. The definition of equation systems using the above documentation-based approaches has the advantage that it is both computer-processable and human-readable with a very small visual and conceptual offset between published and implemented model equations. This does not only reduce the introduction of errors and the effort of error finding during the implementation. It also allows the generation of documentation in no time, since all necessary information is already contained in the model.

1.2 Reuse aspects in MOSAIC

In computer science, the reuse of code is an appropriate way to reduce the implementation time and to take advantage of already invested time in debugging with the hope of a more reliable program. The beneficial effects of reusing model parts are well known and are discussed e.g. by Eggersmann et al. (2004). Modeling errors are most likely reduced and the new model can easily be created by adding new components to the existing system. A more intense reusage of equation systems minimizes the number of multiple and redundant implemented systems. This is inversely proportional to the degree of reusability. If more equation systems and models are reused, the quantity of redundant work is reduced. The reuse of equation systems and the thereby achieved minimization are the basis for a better maintenance of a model. Corrections and important changes which are done in an equation system, are immediately applied to all derived systems and models who make use of it.

Modelling in the documentation level as it is done in MOSAIC has several reuse aspects. The first of these reuse aspects is that equations and equation systems are created only once and then used to produce program code in several languages. Thus, program code can be produced in the language that is needed by the modeller. Creating the models in a meta level (such as the documentation level) in combination with adaptable code generation allows the reuse of equations, and even large equation systems in different languages and for different purposes.

The second aspect of reuse is the modular and repeated use of the created model elements (such as equations, equation systems and notations) in many different contexts. The modelling concept in MOSAIC is based on creating modular model elements that are individually stored in XML files. For example, each equation is stored in a separate XML file. An equation system is also stored in a separate XML file and contains references to the files containing the equations that should be used. Another class of such model elements is the notation. Both equations and equation systems must reference the XML file of the notation that shall be used to explain the

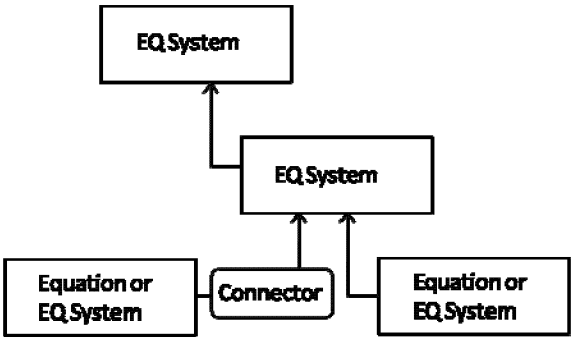

Fig. 1: Modular modelling and reuse in MOSAIC.

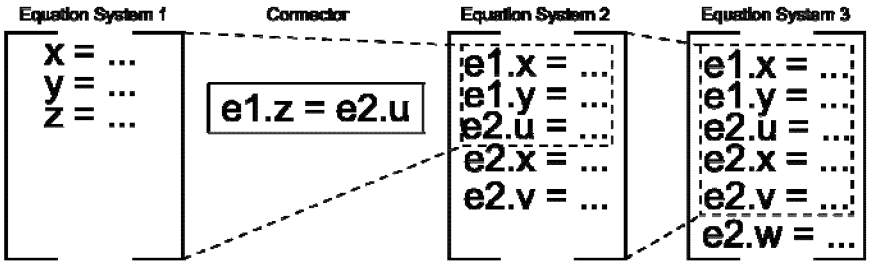

Fig. 2: Reuse and connection of the modelling elements

contained symbols. Each equation or equation system can be (re)used in other equation systems, as shown in Fig. 1. If the notation in one of the connected element is different, a connector is used to translate single, or even all variables. It is important to mention, that the connection is not done with addition equations. Instead, the variables are replaced with the new naming and thereby the equation system does not have to grow unnecessarily. Variables, which are not translated can be kept in their own name space. As can be seen in Fig. 2, in this way, they are strictly isolated from other equation systems where the variable is maybe used with another intention and meaning. The variable x is kept with its own name space e1. It is clearly isolated from the second x with another meaning. The variable z has the same meaning in both equation system and is matched with the aid of the connector.
The equations and equation systems only contain the mathematical structure and the names of the variables used. The values and the classification of the variables into design values and calculated values are of course case-specific and therefore not contained in the equation system. Such information is contained in other modular model elements that are not further covered here. Of course having to create every model element individually takes more time in the short term. However, the resulting high modularity, which notably supports the reuse model elements, has proven to speed up the modelling in the long term.

1.3 Analysis of the model structure in MOSAIC

MOSAIC determines and presents the Jacobians incidence matrix as can be seen in Fig. 3. In contrast to most other simulation programs, the user has an open and direct access to the structure information. With the interactive interface, the relationships and dependencies of the different equations can be explored. The selected point in Fig. 3 for example is the mass holdup of component 2:

$$HU^{m}_{i=2} \tag{1}$$

And is occurring in the equation:

$$HU^{m}_{i=2} = \xi^{L}_{i=2} \cdot HU^{L,m} + \xi^{V}_{i=2} \cdot HU^{V,m} \tag{2}$$

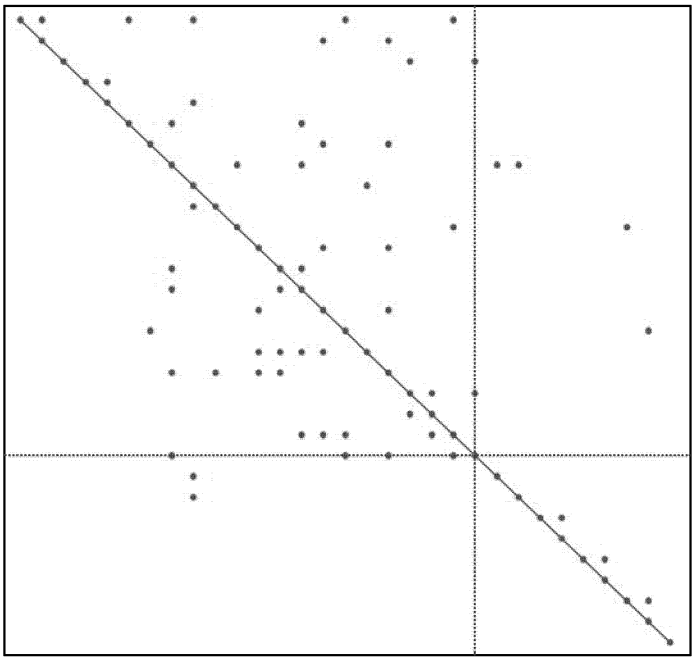

Fig. 3: Incidence matrix in MOSAIC

The incidence matrix is resorted into a block triangular form as described in Duff, 1977 and with this information, it can be possible to decompose the equation systems in various sub problems which can be solved one after another (Barton, 2000). The motivation is to support faster and more robust solution algorithms, which are implemented in the BzzMath library and can be directly called from MOSAIC.

1.4 Derivative generation in the documentation level

By exploiting the tool independency given by modelling in the documentation level, not only the model equation themselves but also their respective derivative information can be generated automatically in a format fulfilling the needs of specific solvers or numerical libraries.

It is well known that derivative information plays a major role in the solution of model equations appearing in engineering (Tolsma and Barton, 1998). Typical problems like the solution of nonlinear equation systems (NLEs), the integration of dynamic systems described by stiff ODEs or DAEs, or the solution of optimization problems do not only require derivative information of first and perhaps higher-order but also benefit from its efficient evaluation. General experiences with numerical computing environments like MATLAB and solvers offering the possibility to incorporate derivative information show that an efficient numerical solution in terms of reduced CPU times and better convergence behavior is often related to an exact and efficient evaluation of derivatives.

The efficiency of derivative evaluation depends strongly on the methods applied, which are normally either finite differences (FD), symbolic derivatives (SD) or automatic differentiation (AD). Although in principle, the code generation capabilities of MOSAIC can be used to generate derivatives with FD and AD compatible code for some specific packages, for the sake of generality, the tool independent XML representation of the symbolic expressions is exploited to generate tool independent derivative information. This is done by the appliance of SD. As described in the literature (Pantelides, 1988) symbolic derivatives are obtained by applying the differentiation rules to a tree representation of an equation. Within MOSAIC this tree results from the analysis of the XML representation of an equation. For instance, the tree representation of the equation $(2)^{2x} - (4 + \sin(x))$ is given by Fig. 4. Since the application of symbolic derivatives to the XML meta-level expressions results in a further XML meta-level expression of the derivative, this can not only be used directly for code generation but also be further analyzed in order to calculate higher-order derivatives.

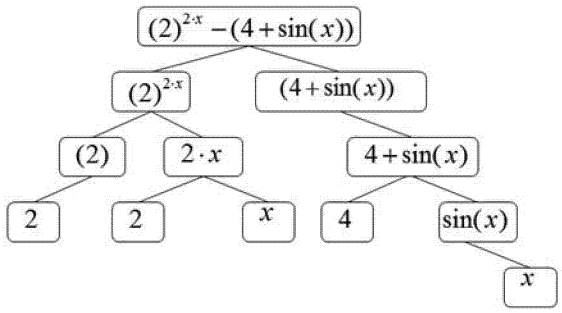

Fig. 4: MOSAIC's internal tree representation of equation

Currently the generation of symbolic derivatives is implemented for the generation of full Jacobians required for the solution of NLEs with the GNU Scientific Library and for full or sparse Jacobians applied for the solution of NLEs and DAEs with MATLAB. By putting together the derivative information along with the model structure information obtained by MOSAIC and the structure dependent algorithms available in the BzzMath Library a higher solution efficiency on the server could be achieved.

2. THE BZZMATH NUMERIC LIBRARY

The BzzMath numerical library is currently used in MOSAIC to solve nonlinear algebraic equation systems and differential algebraic equation systems. Comprehensible output is generated that allows assessing the convergence quality and the reliability of the results and the library has proven to be robust and to provide reliable results not only on these topics. It provides algorithms and methods to tackle problems of different nature. To do so, the library has a very performing kernel for linear algebra (Buzzi-Ferraris, 2010b) that "cannot be overstated in modern scientific computing" (Gill et al., 1991). The library is not only predisposed to solve linear systems, but even to automatically adopt the most performing algorithm in accordance with the type of system to be solved. Being thought in an object-oriented way, objects belonging to BzzMath can easily identify and, if possible, exploit matrix sparsity and structure of linear systems and openMP directives for shared memory machines, which results in a considerable gain of performance (Buzzi-Ferraris and Manenti, 2010a). The library includes specific algorithms and criteria for parameter estimations, outlier detections, rival model discriminations, and design of experiments based on very robust algorithms to solve linear and nonlinear regression problems (Buzzi-Ferraris and Manenti, 2010b). The same algorithms allow identifying with a certain accuracy the existence of common problems in the regression field such as masking/swamping effects, heteroscedasticity, multicollinearity, parameter correlations, gross errors, and influential observation. A methodology to discriminate among rival models and, at the same time, to define the optimal design of experiment is implemented in BzzMath, even accounting for the correct meanings of statistical tests and confidence regions (Buzzi-Ferraris and Manenti, 2009, 2010c). Numerically robust and efficient algorithms are implemented to tackle nonlinear systems and optimization problems. Optimizers are validated by the field in many industrial applications such as the online plant optimization of an oil refinery sulphur recovery unit (Signor et al., 2010) and the real-time dynamic optimization of a large-scale polymer plant (Manenti and Rovaglio, 2008). In addition, certain ad hoc solvers to tackle partially structured DAE systems, which are typical for process control and process systems engineering, have been recently introduced in the library and their superior performances have been validated on industrial cases (Manenti et al., 2009). In this context and looking forward to future developments, we plan to extend MOSAIC so that it is possible to describe optimization and parameter estimation problems. The BzzMath library offers the corresponding numerical algorithms and is thus predestined as standard background environment for calculations on the modelling server.

Apart from that it would fit well into the cloud computing concept of MOSAIC that the used numerical environment is capable of parallel computing. The perspectives of parallel programming have been motivated by Buzzi-Ferraris (2010b). With respect to dividing up computational burden on network computers, such approaches are very promising and the web-based MOSAIC could be one of the first environments that constantly makes use of such techniques.

3. EXAMPLE

The model of a distillation column has been created in MOSAIC. The equation system that was assembled from single reusable equations is presented in the MOSAIC user interface as shown in Fig. *5*. The proximity to the presentation in the literature is obvious.

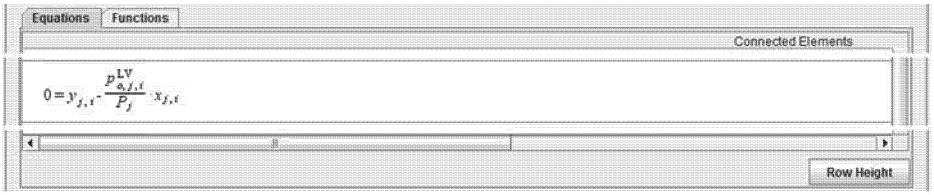

Fig. 5: Equation as it is represented in MOSAIC

To further demonstrate the aspects of program code generation and documentation output, the equation of the phase equilibrium (3) is included as an equation object below. Usually MathType (www.dessci.com/mathtype) would have been used to create the formula within the Microsoft Word document. However, equation (3) has just been copied as a latex string out of MOSAIC and directly pasted into MathType, which resulted without further modifications into the expression displayed here.

$$0 = y_{j,i} - \frac{p_{o,j,i}^{LV}}{P_j} \cdot x_{j,i} \tag{3}$$

The inverse process would also work. More precisely it would be possible to change the formula in MathType, obtain a latex string and hand it over to MOSAIC. The formula expression can be used directly if all contained symbols are described be the specified notation and if the expression follows the MOSAIC convention for mathematic expressions. Due to the indices ‘i’ and ‘j’ contained in (3) the equation is translated into many instances. The C++ code for the instance ‘i=1’ and ‘j=1’ is shown here:

```
f[15] = 0.0-(e0_y_j1_i1-(e0_p_LV_o_j1_i1)/(e0_P_j1)*e0_x_j1_i1);
```
(4)

The full generated code contains solver calls and file output specifications. It can be executed on the server upon a button click in the MOSAIC user interface. The results for nonlinear algebraic system are presented in a table.

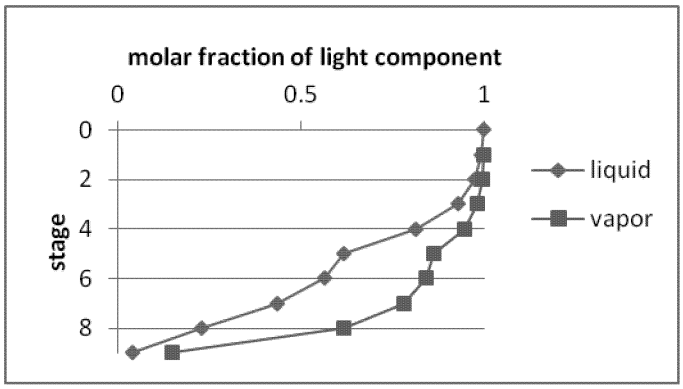

Fig. 6: Plot of results obtained by BzzMath Library with code generated by MOSAIC

The plot of the results obtained by the BzzMath library for the above example is given in Fig. 6.

4. CONCLUSION AND FUTURE RESEARCH

It is today almost impossible to write about modeling in CAPE without mentioning the great efforts and results for standardization and global reuse of simulation software (Yang et al. 2008) and the corresponding fundaments, the CAPE-OPEN standard (www.colan.org) and ontologies.
Modeling in the documentation level has the essential premises (i) to contain all information and (ii) to present the information in a form that is well readable for humans. Accordingly it seems to be obvious that the use of CAPE-OPEN interfaces should not happen implicitly but should be clearly visible in the documentation level. Thus, all aspects of CAPE-OPEN must be reflected as standard elements in the used documentation language or data format. For MOSAIC this means the reflection into the XML model specification, the user interface and the code generation module.
MOSAIC is a domain specific code generator and thus, its main output is modeling programs, which can be used as independent and executable software parts. If the latter also implement the CAPE-OPEN interface (see above) then they can be classified, registered and reused as agent within multi-agent projects. Another interesting potential would be the export of the pure equation systems (Gruber and Olsen 1994). However, up to the current state of development MOSAIC is left intentionally independent of ontological concepts. The introduction of such concepts is desirable in the long term.

5. RELATED WORK AND ACKNOWLEDGEMENTS

Symbolic programming of computers has a long tradition, see e.g. Klerer and May (1965). An interesting approach for documentation-level modelling in Microsoft Office documents is presented by Alloula et al. (2010). Code generation from symbolic expressions has been considered by e.g. Klerer (1992), see also Kajler and Soiffer (1998). Several projects have been dedicated to modelling in the meta level in combination with code generation. Bogusch et al. (2001) presented a non-internet modelling environment that provided many modelling assistance features and allowed code generation for the two tools gPROMS and Speedup. Westerweele and Laurens (2008) presented a non-internet modelling tool that provides code generation into many languages.
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).

6. REFERENCES

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

Barton, P., 2000, The Equation Oriented Strategy for Process Flowsheeting. Massachusetts Institute of Technology

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

Buzzi-Ferraris, G. 2010a. BzzMath: Numerical library in C++. Politecnico di Milano, http://chem.polimi.it/homes/gbuzzi

Buzzi-Ferraris, G., 2010b, New trends in building numerical programs. Computers and Chemical Engineering doi:10.1016/j.compchemeng.2010.07.004

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

Buzzi-Ferraris, G., & Manenti, F., 2010b, Interpolation and Regression Models for the Chemical Engineer: Solving Numerical Problems. Wiley-VCH, Weinheim

Buzzi-Ferraris, G., & Manenti, F., 2009, Kinetic models analysis. Chem Eng Science 64(5), 1061-1074

Buzzi-Ferraris, G., & Manenti, F., 2010c, Better Reformulation of Kinetic Models. Computers & Chemical Engineering 34(11), 1904-1906

Duff, I. S., 2004, On permutations to block triangular form. IMA Journal of Applied Mathematics 19(3), 339–342

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

Gill, P.E., Murray, W. and Wright, M.H. 1991, Numerical Linear Algebra and Optimization. Vol. 1. Addison-Wesley, Redwood City (CA), USA

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

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

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

Kuntsche, S., Arellano-Garica, H. and Wozny, G., 2009, Web-based object oriented modelling environment for the simulation of chemical processes, Chem Eng Trans, 18, 779-784

Linthicum, D. S., 2009, Cloud computing and soa convergence in your enterprise: a step-by-step guide, Addison-Wesley Longman, Amsterdam

Manenti, F., & Rovaglio, M., 2008, Integrated multilevel optimization in large-scale poly(ethylene terephthalate) plants. Ind & Eng Chem Research 47(1), 92-104

Manenti, F., Dones, I., Buzzi-Ferraris, G., & Preisig, H.A., 2009, Efficient Numerical Solver for Partially Structured Differential and Algebraic Equation Systems. Ind & Eng Chem Res 48(22), 9979-9984

Pantelides C.C., 1988, Symbolic and numerical techniques for the solution of large systems of nonlinear algebraic equations, London Imperial College, Phd thesis

Signor, S., Manenti, F., Grottoli, M.G., Fabbri, P., & Pierucci, S., 2010, Sulfur Recovery Units: Adaptive Simulation and Model Validation on an Industrial Plant. Ind & Eng Chem Res 49(12), 5714-5724

Tolsma J.E., Barton P.I., 1998, On computational differentiation. Computers & Chemical Engineering, 22(4/5), 475-490

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