BIOINFORMATICS
Vol. 19 no. 9 2003, pages 1169–1176
DOI: 10.1093/bioinformatics/btg128

Modular modeling of cellular systems with ProMoT/Diva

M. Ginkel, A. Kremling*, T. Nutsch, R. Rehner and E. D. Gilles

Max-Planck-Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, D-39106 Magdeburg, Germany

Received on April 9, 2002; revised on August 22, 2002; November 18, 2002; accepted on November 29, 2002

ABSTRACT

Motivation: Need for software to setup and analyze complex mathematical models for cellular systems in a modular way, that also integrates the experimental environment of the cells.

Results: A computer framework is described which allows the building of modularly structured models using an abstract, modular and general modeling methodology. With this methodology, reusable modeling entities are introduced which lead to the development of a modeling library within the modeling tool ProMot. The simulation environment Diva is used for numerical analysis and parameter identification of the models. The simulation environment provides a number of tools and algorithms to simulate and analyze complex biochemical networks. The described tools are the first steps towards an integrated computer-based modeling, simulation and visualization environment.

Availability: Available on request to the authors. The software itself is free for scientific purposes but requires commercial libraries.

Contact: ginkel@mpi-magdeburg.mpg.de

Supplementary information: http://www.mpi-magdeburg.mpg.de/projects/promot

INTRODUCTION

The success in modern biology in analyzing the genetic structure of many organisms has allowed bioinformatics to become a very popular science. Consequentially, a number of database systems have been developed to organize the large amount of data occurring during research (e.g. Kanehisa and Goto, 2000; Salgado *et al.*, 2001). New measurement techniques like cDNA microarrays and 2D-gelelectrophoreses have been also established and are used to obtain insight into the overall cellular state. Moreover, new measurement techniques which allow samples to be taken within a time window of 2/100 seconds are possible now and are used to analyze the intracellular dynamics of small metabolites if the system is shifted from one steady state to another (Schaefer *et al.*, 1999). These facts—availability of knowledge of the genetic structure and new measurement techniques—smooth the transition of biology from a qualitative to a quantitative science. However, to analyze and possibly predict cellular behavior based on the the increasing quantity of knowledge and therefore more complex cellular system models the application of mathematical modeling is necessary.

For dynamical systems, we previously introduced a suitable modeling framework (Kremling *et al.*, 2000) based on the definition of submodels called *modeling objects*. These modeling objects cover a broad range from single enzymatic reaction steps to rather complex structures, which are called *operons* and *modulons* in bacterial genetics (Neidhardt *et al.*, 1990). This paper deals with two computational aspects in modeling cellular systems: (i) the modular assembly of dynamic model equations; and (ii) model validation based on parameter identification from available measurements. Although different other modeling and simulation tools like GEPASI (Mendes, 1997), Jarnac (Sauro, 2000), VCell (Schaff *et al.*, 1997), DBSolve (Goryanin *et al.*, 1999), E-Cell (Tomita *et al.*, 1999) and others also solve systems of differential equations, they don't provide a modular approach for model setup.

A problem not discussed here in detail is the exchange of models between different modeling and simulation tools. We take part in an international initative of simulation tool developers to define a practical standard for mathematical models of cells that is called systems biology markup language (SBML Hucka *et al.*, 2000). It is planned to import and export SBML in ProMot.

After introducing the modeling concept, a software environment, combining two tools, namely ProMoT and Diva will be presented. The Process Modeling Tool ProMoT (Tränkle *et al.*, 2000) was originally designed for the computer-aided modeling of chemical processes as well as for the implementation of libraries that contain reusable modeling entities. The differential-algebraic

*To whom correspondence should be addressed.

models created with ProMoT are added to the model library of the simulation environment Diva (Mohl *et al.*, 1997). Differential Algebraic Equations—DAE, sometimes also called ODE–NAE models are a combination of ODE that are simultaneously solved with algebraic constraints. The numerical methods provided by Diva are applied to the numerical analysis, dynamic and steady state simulation and identification of model parameters.

SYSTEM AND METHODS

Systems biology seeks to combine experimental and theoretical work for a better understanding of the overall behavior of cellular systems. This implies that not only the cellular interior has to be modeled but also the environment, e.g. the fluxes into and out of a bioreactor which allow exposure of the organism to defined and reproducible conditions. When analyzing complex systems with a high number of elements and several interconnected levels, e.g. a fermentation plant with a bioreactor containing liquid and biophase, where the biophase again is decomposed in metabolic units, a common base is required which is applicable to all levels. Therefore, network theory was proposed for analysis and synthesis problems in chemical and biochemical engineering.

Network theory

Network theory (Gilles, 1998) gives a fundamental way to decompose various processes into hierarchical units in a systematic manner. The hierarchical structure of the process is represented in several levels (see Fig. 1). All levels consist of two basic types of elements, namely *components*, representing the holdup of different physical quantities (drawn as circles in the figure), and *coupling elements* describing the interactions and transports between the different components (rectangles in the figure). The top level can be, for example a device level consisting of components like reactors and other devices and coupling elements like valves and pumps etc. The devices again consist of phases that are coupled by phase-boundaries or membranes and finally the phases consist of *storages* that are coupled via reactions or diffusive and convective relations. Network theory integrates all these levels into the same theoretical concept in a modular way with well defined interfaces.

How are these models computed? There is a division of tasks between the basic elements. Components provide information about their *potentials*, i.e. their concentration and require information about the fluxes coming in and leading out of them. Essentially they balance the potentials with regard to the fluxes. Coupling elements calculate the *fluxes*, i.e. the reaction rates, depending on the potentials and provide the flux information to the components. This two-directional information exchange of potentials and fluxes forms a *potential-flux vector*. If a potential-flux vector is passed across phase or device borders it must consist of extensive quantities only, like mass flux, molar flux or volume flux, otherwise it is not possible to achieve modularity.

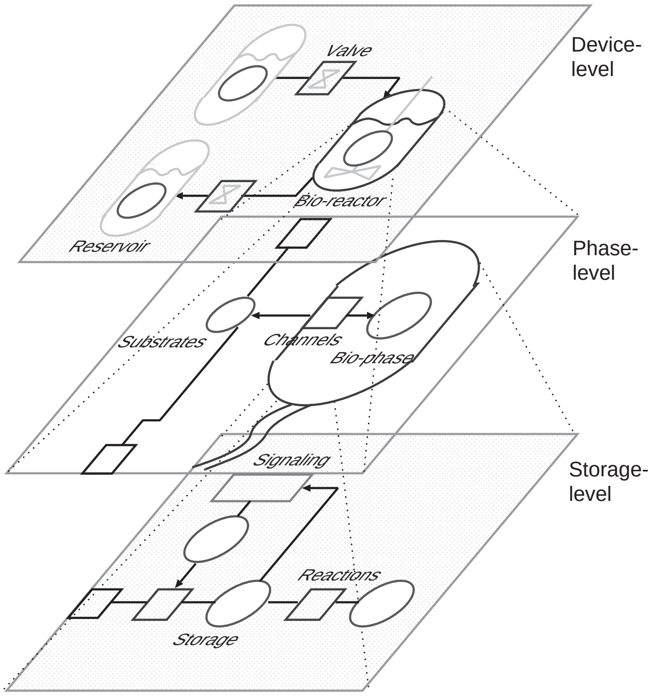

Fig. 1. Representation of different detail levels of a bioreactor model by the means of network theory. The different layers are all represented, based on components and coupling elements. These elements provide a modular structure and interfaces for the more detailed levels.

Modeling concept for biochemical plants

A continuously stirred tank reactor with several substrate feeds and an outflow is considered as an example for a biochemical plant and is depicted in Figure 1. The plant is composed of process devices namely the reservoirs and the bioreactor and their coupling elements that are valves. The bioreactor is modeled with two phases: the liquid-phase and the biophase. The liquid-phase model comprises its volume as an extensive reference quantity V_l (unit $[l]$) and storages for the substrates (concentrations $\underline{c_l}$ (unit $[g/l]$)). The biophase contains storages for intracellular metabolites, and the biochemical reactions as coupling elements. As an extensive reference quantity, e.g. biomass (m_{bio}) or total volume of the cells should be used.

For an exact formulation of the cell growth, all existing exchange-fluxes through transporters between the two phases have to be summed up. But due to the fact that biological models can probably never comprise every single transport pathway connecting the cell to the liquid phase, a slightly different approach has to be chosen. We suggest balancing the exchange-fluxes by using yield coefficients $\underline{Y}$ (unit $[g_{DW}/g]$) for the substrates taken into account in the model. The balance of biomass then results in:

$$\dot{m_{bio}} = \underbrace{\underline{Y}^{\mathrm{T}}\, M\, \underline{j_{ex}}[c_l]}_{\mu}\, m_{bio} - m_{bio}\, \frac{J_{out}[V]}{V_l}\ , \qquad (1)$$

where M is a diagonal matrix of the molar weights of the substrates, $\underline{j_{ex}}[c_l]$ the substrate exchange between biomass and liquid phase, $J_{out}[V]$ the liquid flow out of the reactor and μ the growth rate of the cells. This allows for modeling the cell growth realistically while representing only the main substrate transport pathway (called channel) in the model.

Modeling framework for cellular systems

In microbiology, the thinking in functional units (describing a subset of the cellular processes) has become popular and has resulted in the definition of subnetworks that are under control of a common regulator protein (Neidhardt *et al.*, 1990). The combination of these ideas with network theory leads to a modeling framework which was previously introduced (Kremling *et al.*, 2000). At the highest level of resolution, elementary submodels (modeling objects) are defined. Important elementary modeling objects are substance storages and substance transformers for the metabolic network and signal transformers for the regulatory network.

Two or more storages can be connected by a substance transformer that represents a biochemical reaction. Transformers are treated as two complementary aspects: (i) the representation of the stoichiometric structure of the reaction with interfaces for substrates and products; and (ii) the reaction kinetics together with the participating and controlling ligands (activators and inhibitors).

Since the understanding of signal transduction and processing is the key for describing the overall behavior of cellular systems, these processes are described in a separate class named signal transformers. Elementary modeling objects can now be aggregated to describe more complex processes like gene expression or signal transdcution cascades (Kremling and Gilles, 2001).

Modular model representation

ProMoT enables the use of object-oriented modeling techniques including encapsulation, aggregation, and inheritance. In ProMoT, dynamic models are built by aggregating *structural* and *behavioral modeling entities*. Structural modeling subdivides a model into *modules*. Examples for modules in systems biology are process units (e.g. fermentation reactors), balanced volumes (e.g. phases), functional units of the metabolism (e.g. glycolysis) and elementary entities (e.g. reactions). In general they represent components and coupling elements of network theory on different hierachy levels. Modules are encapsulated and therefore separated from their environment whereas their interfaces are defined by *terminals*. The behavior of a module is characterized by aggregated variables and equations in a module-local DAE.

To establish connections in a modularized model, groups of variables are assigned to terminals. When terminals are linked, additional linking equations connect the different behavioral subsystems. Terminals are not required to have a specified direction (e.g. input or output). In case of substance flows in biological reaction networks they represent a bidirectional information exchange of a concentration and a flow rate in the sense of potential flow vectors. Another important form of terminals in biological systems are cellular signals which represent only a concentration. Modules, terminals and links are structural modeling entities, whereas variables and equations are behavioral modeling entities.

The emphasis on modularity has several advantages in modeling complex biological systems:

- the user works with comprehensible networks of modules rather than with reaction networks with hundreds or thousands of parts. With this feature it is also easier to divide tasks between different modelers working on parts of the same system, which is desirable for large scale biological models;

- the interface of a module can be specified separately from its implementation. This leads to a simplified exchangeability of different module implementations with the same interface. This can be used e.g. for implementations of a module which differ in the detail-level;

- an important and often neglected task in model development is model debugging. With the depicted structure it is easier to debug an individual module with its input-output behavior in a well defined test frame first before the modules are combined to a larger system. Since the couplings are explicit in the modular system, removal of feedback can be easily carried out as simplification to isolate errors in the model.

The modeling entities in ProMoT are organized as an object-oriented class hierarchy with multiple inheritance. This concept from computer science was adopted to allow a better organization of complex modeling libraries and flexible implementation of large scale models. Every entity in this hierarchy inherits all parts and attributes from their respective superclasses. With this method abstraction is possible and more general and reusable entities can be formed.

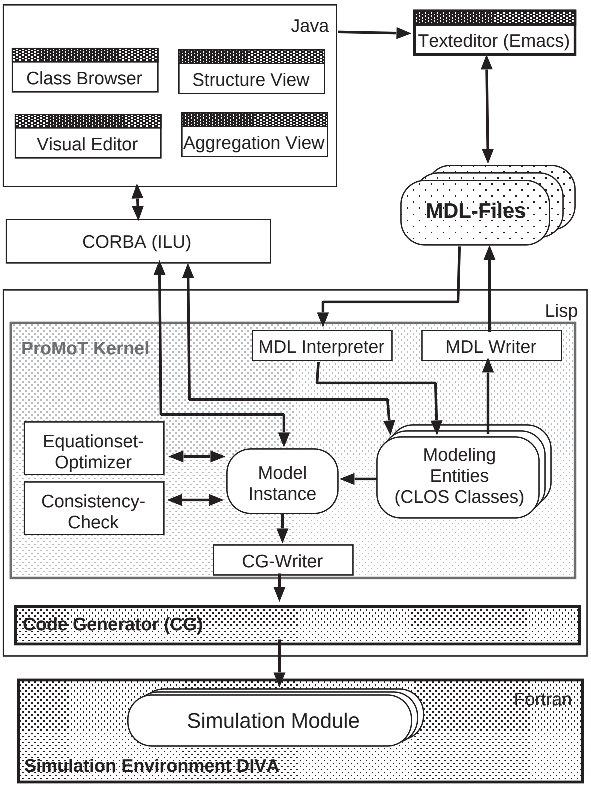

Fig. 2. Software architecture of ProMoT. The kernel provides all model handling including reading and writing modeling language, instantiation, consistency-check and writing of Diva models. The GUI and the kernel interact through a Corba-middleware called ILU.

IMPLEMENTATION OF ProMoT

ProMoT provides a special modeling language as well as a graphical user interface (GUI) for interactive modeling. The modeling tool, as well as the simulation environment, are developed under different Unix-derived operating systems, however the main platform is Linux. As shown in Figure 2 the kernel of the system is implemented as a modeling server in object-oriented Common Lisp (using the Common Lisp Object System CLOS). Further information about availability and requirements of the software can be found on the web page. Although Lisp is currently not a very popular language, it has certain qualities that are adjuvant for an easy and flexible implementation. ProMoT's modeling entities are classes and use multiple inheritance. Therefore they are internally represented by specialized classes in CLOS, which handles inheritance and creation of instances. The classes represent aggregation and composition of aggregated parts explicitly, which allows construction of complex containment hierarchies and their analysis in the final model. Since Lisp classes themselves can be programmed, conforming to the CLOS Meta Object Protocol (Kiczales *et al.*, 1991), this foundation of ProMoT is implemented as an extension of standard Lisp classes. The classes are dynamic meta-objects in the Lisp runtime environment; that is why it is also possible to edit them at runtime using either the graphical editor or through changing the source code.The possibility to do this is rarely found in programming languages: most languages provide class meta objects only for reflection (i.e. read-only introspection in Java), if they provide any at all. For example, common Lisp and Smalltalk also allow one to change classes (write access), which is one of the main reasons to build the modeling environment in Lisp. The representation of the mathematical model is done in a symbolic way. This makes it possible to manipulate the formulae, e.g. for normalization of the differential equations or during optimization of the final simulation model, which can be easily implemented in Lisp.

The modeling language MDL (Model Description Language) of ProMoT is a declarative, object-oriented language that allows a symbolic implementation of variables and equations rather than the programming of imperative code. ProMoT interprets MDL to create the class representation and can serialize the classes to MDL. Thus the modeling language is used as the storage format for the modeling libraries. Because every aspect of a model can be described within MDL, the modeling language is the most powerful way to model in ProMoT. The GUI is a client that is implemented in Java using the Java Foundation Classes (Swing). It interacts with the kernel in a Model View Controller (MVC) fashion and has the role of a view and controller for the models in the kernel. With the GUI, users can explore and manipulate the modeling entities by their graphical representation. Therefore views of the inheritance hierarchy and the topology of submodules and their connections can be presented. The visual aspect is very important especially for the communication in interdisciplinary teams, to have a common notion of the considered modeling entities. Besides that also graphical editing of the topological structure of modeling entities can be done interactively with flow-chart diagrams. In this way new higher structured modules can be created easily. For changes on behavioral modeling entities the GUI launches a text editor in order to change the MDL source code of a single modeling entity. Thus modeling language and graphical editor can be used alternately to change modeling entities from the user interface.

The internal processes in a typical modeling scenario of ProMoT are as follows: The user loads necessary libraries (the details are introduced in the next subsection) with basic module definitions from MDL files using the class browser of the GUI (Fig. 3). Then he creates or extends a module that should be simulated in Diva (the main model). Therefore he builds the module structure out of predefined

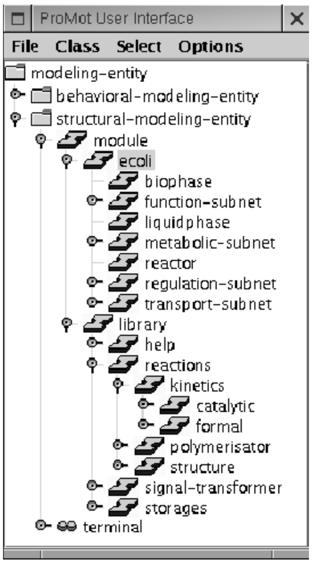

Fig. 3. ProMoT class browser showing the class library discussed in the text.

modules from the library and adds special parameter and initial values for the variables. If the user needs a special module with a behavior not available in the library, he can create this using the modeling language. He should use standardized abstract superclasses for the new module to stay compatible with the rest of the library. Finally the main model is written to Diva. In this process the Lisp class representing the main model is instantiated with all contained submodules. Then a consistency check is performed, that unveils logical errors in the model structure and also structural errors in the underlying equation system.

If these checks can be passed successfully, a compact DAE is generated from the structured representation within ProMoT by aggregating all equations together with coupling relations. The modular structure is only used during modeling: since the Diva simulator does not allow changes to the structure at runtime, it is not needed during simulation. The structured way of modeling in ProMoT and the use of modeling libraries often introduce unnecessary algebraic equations in the resulting model for couplings in links and calculations of variables to achieve flexible modules. The resulting DAE can be divided into a differential part (2) and a purely algebraic part (3):

$$\boldsymbol{B_1}(\boldsymbol{x}, \boldsymbol{p}, t)\dot{\boldsymbol{x}} = \boldsymbol{f_1}(\boldsymbol{x}, \boldsymbol{p}, t) \tag{2}$$

$$0 = \boldsymbol{f_2}(\boldsymbol{x}, \boldsymbol{p}, t), \tag{3}$$

where $\boldsymbol{x}$ is the vector of states, $\boldsymbol{p}$ the vector of parameters, $\boldsymbol{u}$ the input vector and $\boldsymbol{B_1}$ is the descriptor matrix. The modeling system analyzes the algebraic part $\boldsymbol{f_2}$ and identifies implicit algebraic equations ($\boldsymbol{f_2}'$), that have to be calculated simultanously with the differential equations.

$$\boldsymbol{h} := \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{p}, t) \tag{4}$$

$$\boldsymbol{B_1}(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{p}, t)\dot{\boldsymbol{x}} = \boldsymbol{f_1}(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{p}, t) \tag{5}$$

$$0 = \boldsymbol{f_2}'(\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{p}, t). \tag{6}$$

Explicit equations in the algebraic part are sorted according to their dependencies and are directly calculated as assignments to intermediate variables $\boldsymbol{h}$ by the functions $\boldsymbol{g}$. Additionally constant expressions and unnecessary variables are identified and eliminated through symbolic transformations. This produces a more compact and performant implementation of the model that also avoids numerical problems with inconsistent initial conditions of the DAE.

Finally ProMoT generates Fortran source code that can be used within the simulation environment Diva. Therefore the Code Generator (Köhler *et al.*, 1997) is invoked which translates the symbolic representation of ProMoT to Fortran subroutines and prepares the initialization of the sparse matrix numerics of Diva.

Library for metabolic models

All modeling entities are held in a knowledge base that comprises elementary modeling objects like terminals, storages, transformers and channels as well as predefined higher structured modules, e.g. for gene-expression. The user-defined models can be based on the predefined modules and are added also to the knowledge base. In this way the setup of new models is simplified and sped up considerably. It is less error-prone and in addition the models become standardized, what enables exchange and reusability of models. The library for modeling biological systems in ProMoT contains several categories which are presented to the user in the tree-structure of the class browser (see Fig. 3).

For the representation of basic modules like storages and substance transformers basic terminals are defined, e.g. `term-reaction-flux` and `term-storage-flux` for connections of intracellular reactions with intracellular storages. These terminals define a potential variable `c` for a concentration and a flux variable `r` for the reaction rate. Subclasses of these basic terminal types add additional attributes, e.g. in the terminal `term-liq-storage-flux` a variable for the molar weight is aggregated which is necessary to convert cell-external concentrations at the border of the biophase.

The predefined elementary modules are represented beneath `module/library` with the `storages` as an important subgroup. There are storages defined for the liquidphase and for the biophase with different kinds of terminals. For example `storage-intra_x` is a storage with a `term-storage-flux` terminal. It contains a differential equation for a substance storage that automatically takes

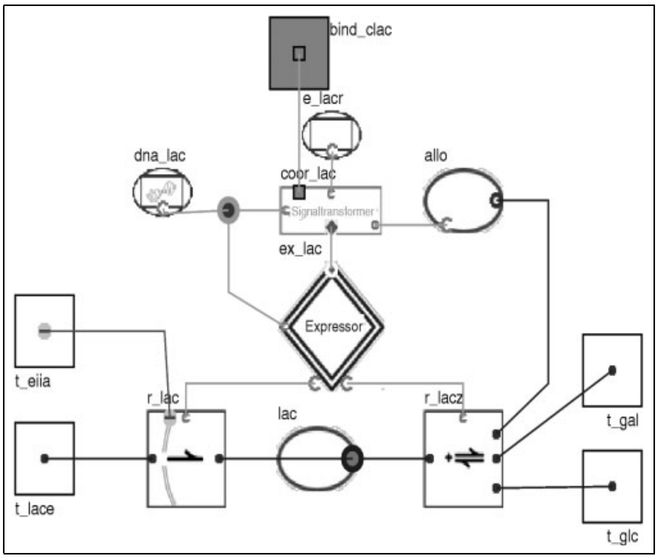

Fig. 4. Setup of a model for lactose transport as it appears in the visual editor of ProMoT. The whole drawing is the interior of a module. The white and gray boxes at the outer edges are external terminals (in this case referring to terminals of the submodules). The elements inside the drawing are aggregated submodules, lines represent links.

the dilution by the cell-growth μ into account. In Figure 4 the modules `lac` and `allo` are instances of this class. The module `storage-const-enz_c` like `e_lacr` in the figure is used for defining the constant enzyme-concentration as an input into a substance or signal transformer.

Another important subgroup of the library is `reactions`, where transformers and channels are defined. Their attributes are subdivided into kinetic and structural properties. Beneath the subclasses `kinetics` there are various predefined modules, e.g. `mm`: Michaelis–Menten, `mmea`: Michaelis–Menten with essential activation, `pp`: ping-pong mechanism, `sr2`: sequential random 2 substrate mechanism and of course simple formal first- and second-order reactions. In these modules the appropriate variables, equations and terminals for the calculation of the reaction kinetics are defined. Kinetics define terminals as well for connecting storages that actually do not take part in the reaction as substrate or product but affect the rate as e.g. enzyme, activator or inhibitor. For the structural (stoichiometric) part of the reactions there is a separate group of modules below `structure`. These classes contain terminals and stoichiometric parameters for subtrates and products and most of the the geometry-information (position of substrate and product terminals, iconic representation). Finally the complete transformer or channel inherits the relevant kinetic and structural properties from one element of both groups respectively that form the aforementioned complementary aspects. This enables a selection of the required transformer either trough the branch `kinetics` or `structure` from the browser tree. For example, the module `trans2a-mm_l` is a transformer that connects two storages with a Michaelis–Menten-reaction and a graphical representation to the left, `channel2a-mmui` is a channel with a Michaelis–Menten reaction and uncompetitive inhibition, where the first connected storage has to be extracellular and the product is intracellular. This channel class is instantiated as submodule `r_lac` in Figure 4.

The user-defined modules are also held in the knowledge base. There is no formal difference between user-defined and library modules. Library modules are just designed having more generality in mind. As an example a model of the carbohydrate uptake of *Escherichia coli* (Kremling and Gilles, 2001; Kremling *et al.*, 2001) has been implemented in ProMoT. The top module of the *E. coli* model is the class `reactor` that consists of the biophase and the liquid-phase (as shown in Fig. 1). The class `biophase` itself is highly structured and comprises some central parts of the catabolism and different transport pathways with their respective interacting regulation networks. One of these pathways is the uptake of lactose as shown in Figure 4.

Lactose is taken up, coming in with the `t_lace` terminal on the left, through the `r_lac` channel on the left and cell-internal lactose `lac`. It is further degraded up by the transformer `r_lacz` into glucose and galactose leaving the module through the terminals on the right hand side. A by-product of this reaction is allolactose (`allo`), which is important for the control of transcription of the enzymes LacY and LacZ catalyzing `r_lac` and `r_lacz` in `coor_lac`. This provides, that the enzymes only get expressed, if lactose is present in the medium. The signal-transformer `coor_lac` models the interactions at the promoter binding site of the DNA-sequence for the enzymes and also integrates the signal of a global activator that is included through the terminal `bind_clac`. The diamond-shaped expressor contains a model for the translation and the degradation of the two enzymes. The reaction `r_lac` interacts with other transport pathways for glucose (namely the phosphotransferase system) via the inhibitor EIIA, which enters the model as a concentration signal through the terminal `t_eiia`. As long as the glucose transport is active the resulting high concentration of EIIA inhibits the uptake of lactose.

NUMERICAL MODEL ANALYSIS WITH DIVA

The numerical analysis of the models is done with the simulation environment Diva (Mohl *et al.*, 1997). Within Diva many different numerical computations are possible, based on facilities to calculate the steady state and dynamic behavior of the model using non-linear equation solvers and integrators. For metabolic models two methods are of special interest: (i) parameter analysis with respect to experimental data; and (ii) identification of parameters and model accuracy.

Parameter analysis and sensitivities

Sensitivity analysis of cellular models is often associated with the calculation of flux and concentration control coefficients from 'Metabolic Control Analysis' (Heinrich and Schuster, 1996). Diva uses parameter sensitivities $w_{ij} = \partial x_i / \partial p_j$ for another purpose.

The aim of the parameter analysis is to find a parameter vector $\boldsymbol{p}$, that is a subset of all parameters contained in the model. It should be possible to estimate $\boldsymbol{p}$ with a given variance γ. The choice of γ depends on the accuracy of the measurement data and the demands on the model. To identify the elements of $\boldsymbol{p}$, given some initial vector $\boldsymbol{p}_0$ the user is interested in, the following approach is used: The model and the measured data are analyzed with the Fisher information matrix (Ljung, 1999). The Fisher information matrix is defined by:

$$\boldsymbol{F} = \sum_{k=1}^{N} [\boldsymbol{W}(t_k)^{\mathrm{T}} \boldsymbol{C}(t_k)^{-1} \boldsymbol{W}(t_k)], \tag{7}$$

with the matrix of the sensitivities $\boldsymbol{W}(t_k) = \partial \boldsymbol{x} / \partial \boldsymbol{p}$ and the covariance matrix $\boldsymbol{C}(t_k)$. The covariance matrix $\boldsymbol{C}$ is assumed as a diagonal matrix with the variance of the states σ_i as elements. It is assumed that σ_i do not depend from time point t_k while it is taken as a constant.

Applying a method introduced by Posten and Munack (1990), $\boldsymbol{p}$ can be determined by analyzing the eigenvalues λ and eigenvectors of $\boldsymbol{F}$: The parameter p_i out of $\boldsymbol{p}$, most contributing to the eigenvector that corresponds to the smallest eigenvalue λ_{min}, is removed successively from p until $\gamma \geqslant \sqrt{\frac{1}{\lambda_{min}}}$. Although this provides only a local estimate of the lower bound for the variance of parameter estimation, the method was applied successfully in optimal experimental design for a biotechnological process (Baltes *et al.*, 1994).

Parameter estimation

Identification with Diva is restricted to the estimation of parameters in a fixed model structure. Measurement data (z_{ik}) is available for a subset of the states at time point t_k. The aim of the estimation method is to minimize the objective function:

$$\Phi(\boldsymbol{p}) = \sum_{k=1}^{N} \sum_{i=1}^{n} w_{ik}{}^2 \left(\frac{x_i(\boldsymbol{x}_o, \boldsymbol{u}, \boldsymbol{p}, t_k) - z_{ik}}{z_i^{\max}} \right)^2 \tag{8}$$

where w_{ik}, $z_i^{\max}$ are scaling factors of the individual measurement data and for each experiment respectively. For the optimization a SQP (**S**equential **Q**uadratic **P**rogramming) method from the NAG library (Moré and Wright, 1993) is used.

Numerical analysis of the example

A mathematical model for catabolite repression was introduced previously (Kremling and Gilles, 2001; Kremling *et al.*, 2001). The model describes glucose and lactose uptake as well as the control of gene expression of the respective enzymes. The model comprises 22 ODE's and seven algebraic equations. According to the underlying modeling concept, the equations are assigned to modeling objects. One of these objects, Lac transport has been introduced above.

Based on the available measurements in first run—biomass, extracellular glucose, lactose and intracellular LacZ—and the experiments performed it is expected that parameters which could be estimated are closely related to glucose and lactose transport kinetics and to LacZ synthesis. The result of the analysis shows that 16 parameters could be estimated In a second run three additional measurements—intracellular concentration of protein EIIA which is involved in glucose uptake and intracellular and extracellular cAMP concentration—are included in the analysis (the time course of these experiments are not yet published). With these measurements available, 20 parameters could be estimated.

DISCUSSION

An overview was given for a workbench of software tools that supports modeling and numerical analysis of cellular systems. The modeling tool ProMoT provides an approved methodology and the possibility to use ready made modeling entities out of knowledge-bases. Efficient modeling is supported by the use of a graphical user interface and a modeling language. Sophisticated methods for the numerical analysis of the resulting models are provided by the simulation environment Diva. Besides dynamic simulation the identification of parameters and the analysis of sensitivities are possible. The workbench is different from other tools like Gepasi and Jarnac because it deals with modular models and can handle DAE like VCell and DBSolve. It is well suited for larger simulation and parameter estimation problems with up to 10 000 differential equations, because of the advantages of modular model development and the efficient numerical routines of the simulator. The main advantage of our software is the modular modeling approach which is to our knowledge not provided by any other aforementioned tool. Since the development cycle for the models includes a compilation step, the advantage of the efficient computation in Diva is bought with a longer preparation time, which does not scale well for rapid prototyping of small text-book models.

For the modeling of cellular signaling the current approach is to simplify the interactions of proteins, DNA binding sites and other substances to form signal transformers. This simplification is often not easily possible, due to a deficiency of quantitative knowledge. Therefore the only correct way is to model all possible interactions with explicit reactions and to explore the behavior of the system interactively in simulation and experiment. Since in complex regulatory networks the number of reactions increases exponentially due to combinatoric effects of binding sites in complex macromolecules, the models become very complicated. For a solution of this problem current research aims to describe the basic interactions of binding sites and the compound structure of the molecules and to let the modeling system generate the complete reaction network automatically. It is planned to implement this approach as a specialized class of modules in ProMoT.

The goal of the described software tools is a virtual laboratory containing facilities for modeling, simulating and visualizing parts of intracellular metabolisms. Therefore other computer aided methods have to be integrated. For rising amounts of knowledge the use of databases is necessary which allow the sharing of discoveries between workgroups at different locations connected by the internet. For effective development of complex metabolic calculation models the use of standardized modeling entities is only one aspect. Others are facilities which allow the visualization of different aspects of the resulting special models and support especially the debugging of models under construction.

REFERENCES

Baltes,M., Schneider,R., Sturm,C. and Reuss,M. (1994) Optimal experimental design for parameter estimation in unstructured growth models. *Biotechn. Progr.*, **10**, 480–488.

Gilles,E.D. (1998) Network theory for chemical processes. *Chemical Engineering and Technology*, **21**, 121–132.

Goryanin,I., Hodgman,T. and Selkov,E. (1999) Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, **15**, 749–758.

Heinrich,R. and Schuster,S. (1996) *The Regulation of Cellular Processes*. Chapman and Hall, London.

Hucka,M., Finney,A., Sauro,H.M., Bolouri,H., Doyle,J.C., Kitano,H. and the rest of the SBML forum, (2003) The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**, 524–531.

Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, **28**, 27–30.

Kiczales,G., des Rivières,J. and Bobrow,.D.G. (1991) *The Art of the Metaobject Protocol*. MIT Press, Cambridge, MA.

Köhler,R., Räumschüssel,S. and Zeitz,M. (1997) Code generator for implementing differential algebraic models used in the process simulation tool DIVA. In *Proceedings of 15th IMACS World Congress*. Berlin, pp. 621–626.

Kremling,A. and Gilles,E.D. (2001) The organization of metabolic reaction networks: II. Signal processing in hierarchical structured functional units. *Metab. Eng.*, **3**, 138–150.

Kremling,A., Jahreis,K., Lengeler,J.W. and Gilles,E.D. (2000) The organization of metabolic reaction networks: a signal-oriented approach to cellular models. *Metab. Eng.*, **2**, 190–200.

Kremling,A., Bettenbrock,K., Laube,B., Jahreis,K., Lengeler,J.W. and Gilles,E.D. (2001) The organization of metabolic reaction networks: III. Application for diauxic growth on glucose and lactose. *Metab. Eng.*, **3**, 362–379.

Ljung,L. (1999) *System Identification—Theory for the User*, Second edn, Prentice Hall PTR, Upper Saddle River, NJ.

Mendes,P. (September 1997) Biochemistry by numbers: simulation of biochemical pathways with GEPASI 3. *Trends Biochem. Sci.*, **22**, 361–363.

Mohl,K.D., Spieker,A., Köhler,R., Gilles,E.D. and Zeitz,M. (1997) DIVA—A simulation environment for chemical engineering applications. *ICCS Collect. Vol. Sci. Pap*. Donetsk State Techn. University, Ukraine, pp. 8–15.

Moré,J.J. and Wright,S.J. (1993) *Optimization Software Guide*. SIAM, Philadelphia, PA.

Neidhardt,F.C., Ingraham,J.L. and Schaechter,M. (1990) *Physiology of the Bacterial Cell: a Molecular Approach*. Sinauer, Sunderland, MA.

Posten,C. and Munack,A. (1990) On-line application of parameter estimation accuracy to biotechnical processes. In *Proceedings of ACC*. **3**, pp. 2181–2186.

Salgado,H., Santos,A., Gama-Castro,S., Millan-Zarate,D., Diaz-Peredo,E., Sanchez-Solano,F., Perez-Rueda,E., Bonavides-Martinez,C. and Collado-Vides,J. (2001) RegulonDB (version 3.2): transcriptional regulation and operon organization in *Escherichia coli* k-12. *Nucleic Acids Res.*, 72–74.

Sauro,H.M. (2000) Jarnac: a system for interactive metabolic analysis. In Hofmeyr,J.-H.S., Rohwer,J.M. and Snoep,J.L. (eds), *Animating the Cellular Map*. Stellenbosch University Press, South Africa, pp. 221–228.

Schaefer,U., Boos,W., Takors,R. and Weuster-Botz,D. (1999) Automated sampling device for monitoring ntracellular metabolite ynamics. *Analyt. Biochem.*, **270**, 88–96.

Schaff,J., Fink,C.C., Slepchenko,B., Carson,J.H. and Loew,L.M. (1997) A general computational framework for modeling cellular structure and function. *Biophys. J.*.

Tomita,M., Hashimoto,K., Takahashi,K., Shimizu,T.S., Matsuzaki,Y., Miyoshi,F., Saito,K., Tanida,S., Yugi,K., Venter,J.G. and Hutchison III,C.A. (1999) E-CELL: software environment for whole-cell simulation. *Bioinformatis*, **15**, 72–84.

Tränkle,F., M.,Zeitz, Ginkel,M. and Gilles,E.D. (2000) Promot: a modeling tool for chemical processes. *MCMDS*, **6**, 283–307.