Parsing and Model Generation for Biological Processes

Laouen Belloli
Computer Science Dept.
FCEyN UBA
Ciudad Universitaria, Buenos Aires, Argentina. 1428
laouen.belloli@gmail.com

Gabriel Wainer
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, ON, Canada K1S 5B6
gwainer@sce.carleton.ca

Rafael Najmanovich
Dept. of Biochemistry
University of Sherbrooke
Sherbrooke, Canada
rafael.najmanovich@usherbrooke.ca

ABSTRACT

Modeling allows us to focus on the important components of a system under study, leaving aside the non-meaningful information. To study metabolic networks, we need to create a new model of the phenomena in order to start the simulations. Here, we propose a method for automating modeling and simulation of biological cell processes using DEVS. We have built a parser and a model-generator in order to read SBML files and generate the model from the data. We also defined a generic model container for a biological cell including essential attributes of a cell, which can be instantiated using SBML to generate individual models.

INTRODUCTION

In the past decades, the need to understand biological processes has increased. In medicine, better understanding of these processes, allows improving drugs and treatments. In biology and biochemistry, biological cells play a fundamental roll and their study is essential. Studies of biological processes are expensive and time-consuming. Often, it is not possible to conduct experiments in the natural environment. Those in-vitro studies can also be expensive and take time. In those cases, Modeling and simulation (M&S) becomes a powerful tool. M&S allows abstracting the phenomena, discarding the non-relevant parts.

We are interested in studying different biological cells. They are divided by the prokaryotic and eukaryotic cells. Differences exist between the large amounts of existing cells, for example, some cells have organelles, and some others have not. Furthermore, several biological processes do not follow the rules that the majority of them do. Taking in consideration these obstacles, it is hard to consider a single model structure. We aim to achieve a flexible structure model that helps modelers by offering a framework that can easily be improved and adapted. This would allow modelers to integrate different models (i.e., a signaling path model with a metabolic network simulated together in order to study their interaction). To do so, we define a mechanism to obtain biological information from SBML files, and use them to generate instances and automate model generation.

Many existing biological processes were traditionally modeled using rigid structures. Making changes in their structure implies restructuring the whole model and new validation process. These models are not ready to be reused in other similar phenomena. Some examples of this are shown in [7,31]. Instead, we merged the M&S process with a DEVS [4,34] generic model in order to obtain an automated procedure where multiple models can be integrated easily.

We propose a DEVS structure of a biological cell that allows the integration of different models even if they have different levels. The proposed model can be seen as a model container. Once the model is added to the structure, we can use SBML files to get different scenarios. The model structure is defined hierarchically so any modification can be done in a particular sub-component of the structure without affecting the rest. This hierarchical structure also allows adding new components to the model with the only need of linking them to the rest of the structure. The parser can be extended in order to automate the instantiation process of new models that were included.

In order to show a simple example of application, we propose a theoretical case of a metabolic network in a cell with no organelles. In this example, we want to show how the M&S process structure works with SBML files. To do so, we have considered a basic metabolic network where we know the expected model and simulation results.

RELATED WORK

There are many different computational approaches for modeling biological systems. A review of computational versus mathematical approaches is shown in [9]. Nevertheless, complex biological processes can be better explained combining different approaches as each sub-processes can be better describer with a different technique. Recently, XML-based technologies have made possible exchanging remote messages on the Internet [10] and it also improved the way that models are saved and reconstructed. For instance, in [20], the authors showed how to use XML to save and reconstruct PDEVS models automatically.

As shown in [26,28] combining macro and micro levels has advantages. The abstraction at different levels allows the models to describe each part in a more customized and accurate way [24]. Hierarchical modeling and composition in system biology is presented in [17] and the idea of model re-usability is defined in [27]. Also, since spatial modeling in biological cells play an important role and different approaches are used for this purpose [2]. Models that are more realistic can be achieved integrating these spatial models.

Regarding M&S in biological systems, some new methods use computational techniques to automate some parts of the work. In [21] a stochastic simulation environment that focus in the spatial aspect of biological processes was proposed. In [35] a methodology for automatically detecting signaling pathways from a source of data allowed modelers to find pathways on large reaction networks. JAMES II [6] allows different formalism and plug-ins to be included in order to find more accurate tools for each model. A review of different computational tools can be found in [5].

The System Biology Markup Language (SBML) is now the standard for biology representations. It allows saving and exchanging models, and it is useful for visualization and validation [11]. There has been some recent work on the automation for biological systems using SBML. For instance, in [33], SBML-DEVS introduced a framework for modeling reaction kinetics of biological systems using SBML for the model's data. They used LibSBML [3] to translate the chemical reactions into differential equations.

The research above focuses on the particular systems and levels, but integration of different models using those methods is complex. In order to allow integration of different biological processes in cells to be modeled together, we propose an integrative DEVS structure where multiple models can be added. Model integration allows to better model different parts of a system using different approaches. As shown in [8] combining state-based and Scenario-Based approaches can help to facilitate the task of modeling and thus, checking the model against the real observations.

Instead, Discrete Events system Specification (DEVS) [4,34] provides a hierarchical and modular formalism for modeling these Systems. The hierarchical and modular structure of DEVS allows defining multiple models that are coupled to work together in a single and model by connecting their input and output through messages. In the same way, the resulting model can also be coupled with others models defining multiple layers in the hierarchical structure. Coupled models provide modularity to the structure; as their behavior is described by the composition of the sub-components and their connections, a sub-component does not change its behavior depending on which coupled model is using it. The behavior of any DEVS model is independent of the rest of the models. This modularity is helpful for biological systems modeling. For example, a component can be modeled using ODEs with QSS [15] and another one in a genome-scale with common DEVS models. Then final coupled model will use both as sub-components and will connect them through the IC set.

In this project we used a version of the CD++ toolkit [30] called CDBoost [18,29], a DEVS simulator implemented in C++11. The simulator only uses the C++ Boost library [12]. It is a cross platform, and a good tool to allow the model generation to be portable, cross platform and avoid possible overhead added by the simulator [18].

In order to read the SBML files, we use the C++ TinyXML library [37], which offers a simple method to parse an XML file and store its data into a DOM (Document Object Model) (a markup language that can be translated in a in order to store the information from a XML file in the program memory). We use a DOM structure to store the information provided by the SBML file into our program.

PROCESS ARCHITECTURE

The three stages for M&S proposed in this work are parsing, model generation and simulation. These serve to automate the model instantiation using SBML. The model generation stage uses the results of the parsing stage to obtain the data and generate the model. Since the parser is used by the model generation stage, it is valid until the model is ready and the simulation stage starts. The model generation interacts with the parser without knowing the parser implementation. Figure 1 shows how the classes are interconnected. When the generation stage is complete, and the models are ready, *PDEVSSimulator* runs the simulations.

The parser obtains information from the multiple SBML structure lists, which store information about the reactions, the compartments, the species and the units. When the SBML information is read, its data is stored in a DOM structure, and iterators are created pointing this structure to allow fast access to the data, which is processed on demand. Each time the processed data is required, the original data is copied and the processing is done over the copied data. This allows maintaining the original data and it makes it easy to extend the parser class for new functionalities.

We use the hierarchical information of the SBML file to construct the model. For example, in SBML, the species are assigned to compartments, and reactions have a list of reactants. We use the relation between the species, compartments and reactants to infer the hierarchical structure of the compartments. The parser uses the TinyXML library to handle the data. The parser separates the species by compartments, generates the compartments relations and creates a membrane space for each compartment. It also separates the biomass reaction from the rest. Some model parameters cannot be described in SBML. Normally these are part of the specific models, and they do not come from the nature of biological processes. Depending in the model implementation, we can need extra parameters. These extra parameters must be specified as part of the model generator input. More information about SBML models is given in [11].

The parser interface provides the necessary methods to obtain processed data for the model generation. This interface is available after the parser initialization, and it works as a connector between *modelGenerator* and the parser. Whenever the parser is modified to add new functionalities, the existing interface should be extended. The *modelGenerator* then uses the old interface to generate the model structure. The model generator creates the *Parser* in order to use it. This is why when modelGenerator is constructed automatically, the parser is created and its interface is initialized.

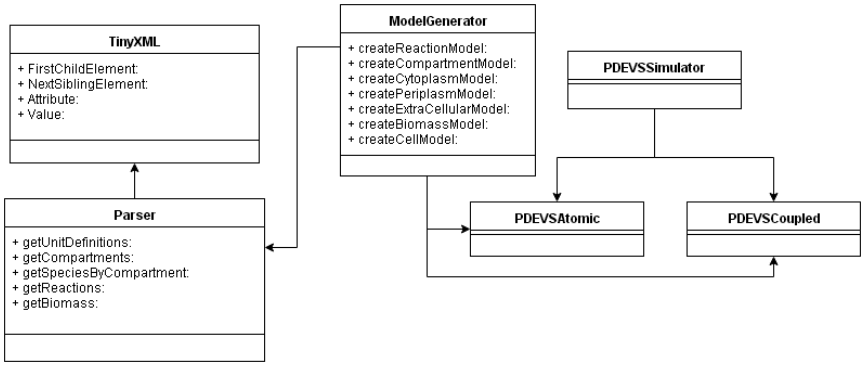

Figure 1. Class diagram of the parsing and model generation processes.

The *modelGenerator* class stores the logic of how to create the components. It also has the model structure embedded in the method *createCellModel*. The process of constructing the model is incremental. At first, all the atomic models are generated. When they are ready, the coupled models are built using their dependencies in the hierarchy. In this process, the model is generated bottom-up from. The *modelGenerator* class creates the parser. The atomic models are created as classes extending *PDEVS Atomic*. These describe the common aspects of atomic models and they become a well-defined model when they are instantiated. To do so, *modelGenerator* knows the structure (for coupled models), and the instantiation parameters (for atomic models). When the model structure needs to be improved and/or new models are added to the structure, the *modelGenerator* methods must be updated to allow these changes.

The *modelGenerator* interface is abstracted from the model structure or any detail that could be changed in the future. This allows integrating different structure models in different application without any additional modifications. This abstraction generates models as an abstract factory without the need to know the model's full extent of the application will not be affected. As an example, Figure 2 shows the algorithm to create the bulk solution models:

Algorithm 1 bulk solution model instantiation algorithm

```
1: procedure GETBULKSOLUTIONMODEL(compartment_name)
2:     enzymes ← this.getEnzymeSetModel(compartment_name)
3:     space ← this.getSpaceModel(compartment_name)
4:     eic ← { space }
5:     ic ← {{space, enzymes}, {enzymes, space}}
6:
7:     eoc ← { space }
8:     model ← coupled_model(eic,ic,eoc)
9: end procedure
```

Figure 2. modelGenerator to create bulk solution models.

We can see how modularity is achieved by using the interface *getEnzymeSetModel* and *getSpaceModel* to get subcomponents. If changes are made in the subcomponents of a bulk solution, the coupled model does not need to change *getBulkSolutionModel* for the new subcomponents.

When the *modelGenerator* stage is complete and the model is ready, *PDEVSSimulator* from CDBoost is used to run simulations. We can generate different scenarios from the SBML file without modifying the program. We can also generate different scenarios by the input data, an external file that is used by a generator model and it normally serves to give metabolites to the extra cellular space. The execution process is shown in Figure 4.

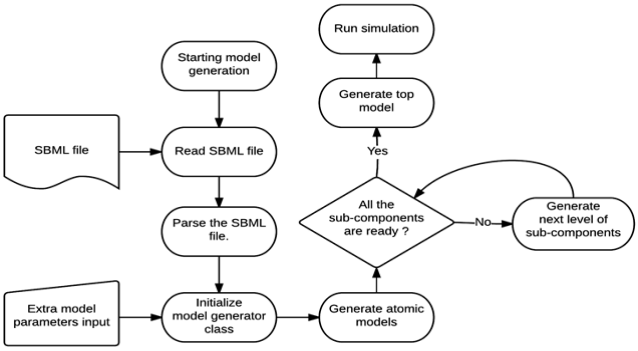

Figure 3. Execution process flow.

MODEL ARCHITECTURE

Biological cells have common structures and behaviors that can be abstracted in to create a generic model structure that describes the common properties of the biological cell models and can be instantiated with particular parameters. For example, the formula in **Error! Reference source not found.** describes the common properties of a non-reversible reaction.

$$\sum_{i=0}^{n} a_i s_i \rightarrow \sum_{j=0}^{m} b_i p_i \quad \text{Equation 1}$$

- $a_i \in subtract\ stoichiometry.$
- $s_i \in subtracts.$
- $b_i \in product\ stoichiometry.$
- $p_i \in products$
- $n, m \in \mathbb{N}$

On the other hand, the mathematical formula shown in Equation 2 is a particular instance of the general formula in **Error! Reference source not found.**.

$$8Fe + S_8 \rightarrow 8FeS \quad \text{Equation 2}$$

A biological cell has a hierarchical structure where its components can have sub-components. Each interacts with others at the same level and with components from nearby lev-

els (the parent level and the child level). Each component of the cell can be seen as DEVS model, and the mapping between the cellular structure and the model of the structure is simple and clear. The mapping between a generic biological cell and the proposed DEVS structure is shown in **Error! Reference source not found.**.

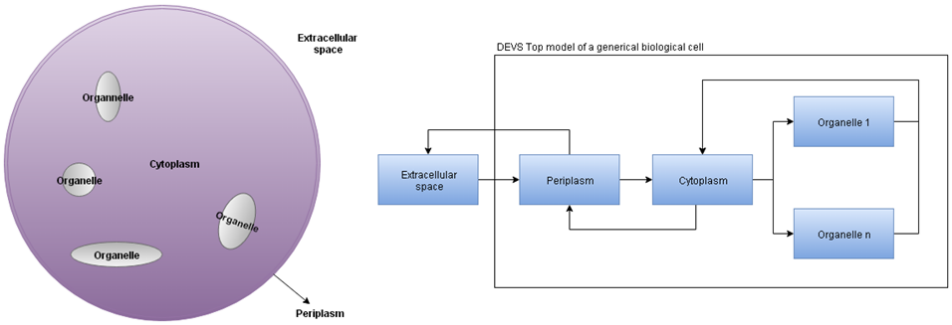

Figure 4. Mapping between a general biological cell and the model structure.

The idea of the general structure shown in Figure 4 is not to describe every possible cell, but to be flexible enough in order to allow modeling any of them by extending the structure and modifying the sub-component. Each atomic model can be replaced by a coupled model adding more sub-component achieving models that are more complex.

In the proposed model, we only consider the structural part of a cell as the model components. These components are in charge of handling the interaction between compartments and the movement of the molecules through the space and bulk solutions. The molecules used as substrate and product are not represented by atomic models, and they are in the model messages. On the other hand, the metabolites are not static components of a biological cell. They can be obtained from the extracellular space or from reactions. At the same time, metabolites do not play an active role introducing any behavior. Instead, they are only used as the reaction's subtract and product. It is important to notice that in most of the cellular reaction, enzymes are in charge of handling those reactions. For this reason, the metabolites are used as the input/output of the components, and they only exist in the messages and in the component states. Both messages and component state are not static; they are constructed and destructed when needed. Without modeling metabolites, we can handle the dynamic of the cellular reaction.

In biological cells, there are thousands of proteins and reactions. This could be a problem if we wanted to model each protein as a component. We use multistate models to have sets of proteins grouped by their states. All the proteins with the same state are seen as a single one until their change and are regrouped by their new states.

As mentioned in [14] whenever a metabolic network is modeled we need to deal with the exponential explosion. We have modeled the space as a stochastic process that determines when collisions between proteins and metabolites happen, and send the information to the component in charge. The space models store the available amount of metabolite and protein in their states. Whenever a new reaction takes place, the component in charge sends the product back to the space for the amount of metabolites and proteins to be updated in that corresponding space. These models must be defined abstracting the particular values. On the other hand, they also need to be flexible to use SBML to instantiate those values. The classes defined in CDBoost are used to implement the general models as classes.

As in real biological cells, the metabolites of the extracellular space cannot go directly to the cytoplasm without passing through the periplasm. We use the same structure. The periplasm model has three membranes, the space and the inner. It has a space that allows reactions to happen. This component is an intermediary between the extracellular space and the cytoplasm. The *space* component in Error! Reference source not found. is a model of the metabolites and enzymes in the bulk solution and their collisions. The internal transition function calculates collisions using a stochastic process, and it sends the metabolites that have collided to the Inner where reactions will take place and the products are returned to the *Space*. The membrane components play a similar roll that the real ones do: they store transport proteins and they bring the needed metabolites to the inner of the periplasm. A metabolite has two options in order to pass through the periplasm. The first option is to go inside the periplasm by one of the two inner or outer membranes and go outside again. The second option is to go directly by the trans membrane. This membrane communicates the extracellular, the periplasm space and the cytoplasm. The Organelle model is similar to the periplasm, but with only one membrane that handles transport reactions between the organelle and other compartments. There is also the organelle *Inner* that handles all reactions inside the organelle.

In order to add new models to the structure or to improve the existing model, we must decide if it is a structural or behavioral modification. The first one will cause the model to represent a new organization of the real cell, while the second one will represent a different interaction between its components. The structural modification uses modularity to modify the structure in order to add and/or remove components in any level. This allows us to add new models in the

structure. For example, we can add a component that models the signaling network and then we will have a more complex model of a biological cell with the metabolic and signaling network modeled together. The second possibility is to modify the behavior of the components. This is useful to validate new models: if we have a tested model for the metabolic networks using a component to model the space, we could replace it by a new one and check the results.

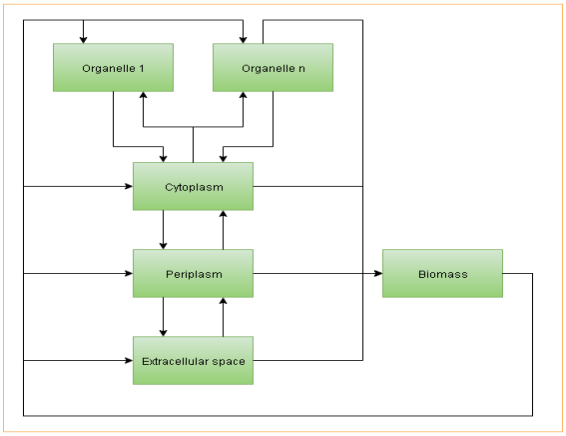

Figure 5. Cell coupled model structure.

The proposed model structure shown in Figure 5 defines the structure of a biological cell and its metabolic network. However, the main benefit is its general definition that allows easy instantiation and integration of different models. With this purpose, we have designed the model to work as a framework that modelers can use.

CASE STUDY

In order to show how the proposed process works, we have designed a theoretical model. This model has a transport protein from the extracellular space to the periplasm and from the periplasm to the cytoplasm. The transport proteins carry metabolites of A from the outside to the inside of the cell. In the cytoplasm, there are three proteins. The first one takes A and returns 2A, the second one takes 100A and returns 200A and the third takes 1000A and returns 3000A. The biomass model is triggered when the amount of A is over a specific number of molecules.

Transport proteins placed in membranes carry metabolites to the cell; if this does not happen, there will never be metabolites in the cytoplasm. At the same time, the Space models and the Inner models interaction is validated through the increase of metabolite A in the cytoplasm until the biomass drop the surplus. In this example we have used three different values for the Kon and Koff constant. In order to validate the stochastic behavior using Kon and Koff constant, we run the examples using for these constants a value of 0.8. This value is high enough to predict an almost constant increase that is not completely regular.

For the non-reversible example, Figure 6 shows the amount of metabolite A over the time. The amount increases constantly because there are neither reversible reactions nor reactions that help to decrease the concentration of A, which is increased by the reaction factors, which are linear.

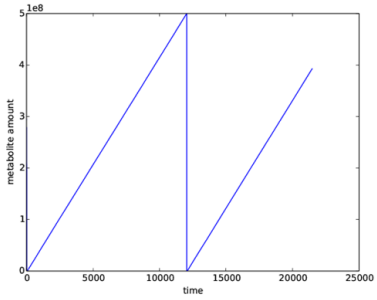

Figure 6. Metabolite A; non-reversible Biomass is 500000000.

Error! Reference source not found. shows a reversible example. The amount of metabolite is not constant because metabolite A is not only increased by the reactions but also decreased when the reactions occur in the opposite direction. Nevertheless, the concentration of A increases because the stoichiometry factors need less concentration to produce than to remove.

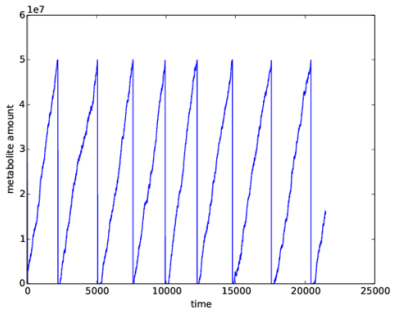

Figure 7. Reversible reaction. Biomass 500000000. Kon/Koff 0.8.

These examples shows the interaction between reactions and bulk solutions and the transport. The increases of metabolite A would never be able to start if the transport protein will not carry them to the cytoplasm. At the same time, if the interaction between the bulk solution and the inner was not well defined, reactions will not happen. The reversible reaction help to validate the not-determinist behavior of the bulk solution and binding process but is not a complete validation. The complete validation is in progress using for this a real study case of the E. coli.

CONCLUSIONS

We showed how to combine automatic instantiation and integrative modeling. Whenever the model structure or the automation process is not useful, we need to adapt it. The modular architecture of the tool and the model allows the modeler to improve what needs to be improved without affecting the rest. These improvements can also be integrated in other models if needed.

The combination of automation and model integration allows not only reusing models, but also improving them. In this work we have also proposed a strategy to achieve com-

plex biological models allowing backtracking in the construction process.

We aim to improve the general structure to optimize computational complexity and be able to use clusters to run simulations. For a best communication and interaction between models, we also aim to integrate this tool with multiple web services as shown in [32] and create a web interface so modelers can improve the communication process between collaborators and with the simulation server.

REFERENCES

[1] S. Biermann, A.M. Uhrmacher, H. Schumann, Supporting Multi-Level Models in Systems Biology by Visual Methods, in: Proceedings of European Multi-Simulation Conference, 2004.

[2] A.T. Bittig, A.M. Uhrmacher, Spatial modeling in cell biology at multiple levels, in: Proceedings of the 2010 Winter Simulation Conference, IEEE, 2010: pp. 608–619.

[3] B.J. Bornstein, S.M. Keating, A. Jouraku, M. Hucka, LibSBML: an API Library for SBML, Bioinformatics. 24 (2008) 880–881.

[4] A..-H. Chow, Parallel DEVS: a Parallel, hierarchical, modular modeling formalism and it distributed simulator, in: WSC '94 Proceedings of the 26th Conference on Winter Simulation, 1996: pp. 716–722.

[5] W.B. Copeland, B. a. Bartley, D. Chandran, M. Galdzicki, K.H. Kim, S.C. Sleight, et al., Computational tools for metabolic engineering, Metabolic Engineering. 14 (2012) 270–280.

[6] R. Ewald, J. Himmelspach, M. Jeschke, S. Leye, A.M. Uhrmacher, Flexible experimentation in the modeling and simulation framework JAMES II--implications for computational systems biology., Briefings in Bioinformatics. 11 (2010) 290–300.

[7] A.M. Feist, C.S. Henry, J.L. Reed, M. Krummenacker, A.R. Joyce, P.D. Karp, et al., A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information, Molecular Systems Biology. 3 (2007) 1–18.

[8] J. Fisher, D. Harel, E. Hubbard, Combining state-based and scenario-based approaches in modeling biological systems, Computational Methods in Systems Biology. (2004) 236–241.

[9] J. Fisher, T. Henzinger, Executable cell biology, Nature Biotechnology. (2007) 1239 – 1249.

[10] Y. Harzallah, V. Michel, Q. Liu, G. Wainer, Distributed simulation and web map mash-up for forest fire spread, 2008 IEEE Congress on Services, SERVICES 2008, July 6, 2008 - July 11, 2008. PART 1 (2008) 176–183.

[11] M. Hucka, a. Finney, H.M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, et al., The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models, Bioinformatics. 19 (2003) 524–531.

[12] B. Karlsson, Beyond the C++ standard library: an introduction to boost, 2005.

[13] D.B. Kell, Systems biology, metabolic modelling and metabolomics in drug discovery and development, Drug Discovery Today. 11 (2006) 1085–1092.

[14] S. Klamt, J. Stelling, Combinatorial complexity of pathway analysis in metabolic networks., Molecular Biology Reports. 29 (2002) 233–236.

[15] E. Kofman, S. Junco, Quantized-state systems: a DEVS Approach for continuous system simulation, Transactions of the Society for Modeling and Simulation International. 18 (2001) 123–132.

[16] Y. Matsuoka, S. Ghosh, N. Kikuchi, H. Kitano, Payao: a community platform for SBML pathway model curation., Bioinformatics (Oxford, England). 26 (2010) 1381–3.

[17] C. Maus, M. John, M. Röhl, A.M. Uhrmacher, Hierarchical modeling for computational biology, Formal Methods for Computational Systems Biology. 5016 (2008) 81–124.

[18] D. Niyonkuru, G. Wainer, O. Dalle, Sequential PDEVS Architecture, in: DEVS '15 Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium, 2015: pp. 165–172.

[19] N. Le Novère, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, et al., BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems., Nucleic Acids Research. 34 (2006) D689–91.

[20] M. Röhl, A.M. Uhrmacher, Flexible integration of XML into modeling and simulation systems, Proceedings - Winter Simulation Conference. 2005 (2005) 1813–1820.

[21] C. Sanford, M.L.K. Yip, C. White, J. Parkinson, Cell++--simulating biochemical pathways, Bioinformatics. 22 (2006) 2918–2925.

[22] H. Schmidt, M. Jirstrand, Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology, Bioinformatics. 22 (2006) 514–515.

[23] B.E. Shapiro, M. Hucka, A. Finney, J. Doyle, MathSBML: a package for manipulating SBML-based biological models., Bioinformatics (Oxford, England). 20 (2004) 2829–31.

[24] B. Soediono, A.M. Uhrmacher, M. Hucka, D. Harel, M. Kwiatkowska, P. Mendes, et al., CHALLENGES FOR MODELING AND SIMULATION METHODS IN SYSTEMS BIOLOGY, Journal of Chemical Information and Modeling. 53 (1989) 160.

[25] C.-F. Tiger, F. Krause, G. Cedersund, R. Palmér, E. Klipp, S. Hohmann, et al., A framework for mapping, visualisation and automatic model creation of signal-transduction networks, Molecular Systems Biology. 8 (2012) 1–20.

[26] A. Uhrmacher, D. Degenring, B. Zeigler, Discrete event multi-level models for systems biology, Transactions on Computational Systems Biology. 1 (2005) 66–89.

[27] A.M. Uhrmacher, D. Degenring, J. Lemcke, Mario Krahmer, Towards reusing model components in systems biology, Computational Methods in Systems Biology. 3082 of th (2005) 192–206.

[28] A.M. Uhrmacher, R. Ewald, M. John, C. Maus, M. Jeschke, B. Sussane, COMBINING MICRO AND MACRO-MODELING IN DEVS FOR COMPUTATIONAL BIOLOGY, Journal of Chemical Information and Modeling. 53 (1989) 160.

[29] D. Vicino, CDBoost simulator, (n.d.).

[30] G. Wainer, CD++: a toolkit to develop DEVS models, Software: Practice and Experience. 32 (2002) 1261–1306.

[31] G. Wainer, R. Djafarzadeh, DEVS modelling and simulation of the cellular metabolism by mitochondria, Molecular Simulation. 36 (2010) 907–928.

[32] S. Wang, G. Wainer, A Mashup Architecture with Modeling and Simulation as a Service, Web Information

Systems Engineering–WISE 2015. (2015) 247–261.
[33] Z. Wang, CELL BIOLOGY SIMULATION USING DEVS COMBINED WITH SBML, 2009.
[34] B. Zeigler, S. Vahie, DEVS formalism and methodology: unity of conception/diversity of application, in: WSC '93 Proceedings of the 25th Conference on Winter Simulation, 1993: pp. 573–579.
[35] X.-M. Zhao, R.-S. Wang, L. Chen, K. Aihara, Automatic Modeling of Signal Pathways From Protein-Protein Interaction Networks, Proceedings of the 6th Asia-Pacific Bioinformatics Conference. (2007) 287–296.
[36] Z. Zi, E. Klipp, SBML-PET: a Systems Biology Markup Language-based parameter estimation tool., Bioinformatics (Oxford, England). 22 (2006) 2704–5.
[37] TinyXML, (n.d.).