Physiological Model Creation and Sharing

Jan Šilar[1,2], **Jiří Kofránek**[1]

[1] Institute of Pathological Physiology, First Faculty of Medicine, Charles University in Prague, Czech Republic

[2] 3rd Medical Department, First Faculty of Medicine, Charles University in Prague, Czech Republic

Abstract

Background: Mathematical modelling in medicine (physiology) takes place in research, clinical practice and learning. There are several suitable modelling languages and tools for physiological model implementation. Graphical simulators are created in order to make models used in teaching more illustrative.
Objectives: It is important to make model developers' cooperation easier. For this reason it should be possible to interconnect models written in different languages. E-learning simulators must be well available for students and teachers. Necessity to install the application can be an obstruction. Methods for sharing models written in Modelica language should be enhanced.
Methods: Standardised interface FMI allows interconnection of models written in different languages. E-learning models are developed on .NET platform. It enables them to run in web browser. The editor for models written in Modelica is being developed. It will run in browser and will be interconnected with web based model repository.
Results: Simulation runtime, solver and simulation centre was implemented in .NET languages. They are used in web e-learning applications. They will be also used together with web-based model repository, that is being developed now.

Ing. Jan Šilar

Conclusions: New tools will simplify model creation and interconnection. They will also improve collaboration of model developers.

Keywords

Model, simulation, physiology, differential equations, solver, domain specific language, block oriented / acausal language, simulator, .NET, teaching, web-based model repository

Correspondence to:

Ing. Jan Šilar
Institute of Pathological Physiology, First Faculty of Medicine,
Charles University in Prague, Czech Republic
Address: U Nemocnice 5, 128 53 Prague 2, Czech Republic
E–mail: jansilar@jansilar.cz

EJBI 2011; 7(1):55–58
recieved: September 15, 2011
accepted: November 8, 2011
published: November 20, 2011

1 Introduction

Many scientists and physicians in the whole world deal with physiological modelling. Modelling is used particularly in areas like genetics, proteomics, pharmacokinetics, physiology of organ systems etc.

It is important for scientists to be able to cooperate well and share results of their work. Sharing models by lengthy descriptions in journal articles is not always sufficient. It is impossible to describe the complex model with all details in the article. Sharing implemented models ready to be simulated on a computer is better. It is important to choose a language that can describe the model's mathematical relations in an accurate as well as understandable manner. The goal is to make even a complex and complicated model easy to understand.

2 Languages and Tools

From the mathematical point of view, models can consist of algebraic equations, ordinary differential equations (ODE) and partial differential equations (PDE). There are also stochastic models, agent models, neuron networks, etc. All listed options can also be combined. In this arti-

cle we concentrate on models based on systems of ordinary differential equations and algebraic equations (DAE). Several convenient languages exist to implement these models.

Implementing simulation runtime in some general programming languages (usually FORTRAN or C), that combines the model and a numerical solver is an old approach but it is sometimes still utilised. This makes the creation of a solver tailored for the particular model possible. Various numerical methods, different time steps etc. may be used for each equation of the model to make simulation faster and more precise. This approach was convenient in times when computer performance was significantly lower than today. This was actually the only possibility when no specialised simulating tools and languages were yet developed. The disadvantage is that the model and the solver are merged in one code. It is necessary to implement numerical methods to solve equations with each new model. This is very labour-intensive and demands knowledge of numerical mathematics. The main disadvantage is a lack of clarity. The model equations are being lost in the clutter of the code.

It is more suitable to use one of the domain specific languages (i.e. a language focused on the specific issue) for the DAE model description. These languages use either a block oriented or an acausal approach or their combination [2, 5].

Block oriented languages (also called signal or causal languages) describe a model using functional blocks. These blocks have inputs and outputs that are used for their interconnection. Individual blocks may for example represent common mathematical operations such as addition, multiplication, integration or many other mathematical functions. The signal passes through these blocks, where it is being modified. Like this the equations are already clearer. The functional blocks stand for equations and the connections between them stand for the variables.

If we find a feedback (i.e. the input is being fed by an output dependant on this input) in the block oriented model, we call this situation an *algebraic loop*, see Figure 1.

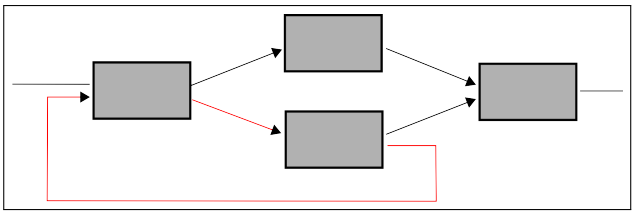

Figure 1: Algebraic loop.

Some algebraic loops can be solved by simulation tools, some represent very complex computational issues and some are insoluble altogether.

The most used and also the most refined block oriented[1] simulation language, also used in physiology [3], is *Simulink* by Mathworks company[2]. The obvious disadvantage is its dependency on the commercial Simulink/Matlab tool. Another example of a commercial signal modelling tool also utilised in the physiology field is *LabView* [7, 4].

Acausal languages work with equations. Unlike in the block oriented approach, there is no need to identify which variable is calculated from which equation. The equations constitute a system of equations, which is being solved all at once. This best represents the mathematical approach. The equations are clearly visible in the implemented model. The following languages are closer to the acausal approach.

There are several open languages directly focusing on biological simulations. The first is the *SBML*[3] [1] (Systems Biology Markup Language), a markup language based on XML. MathML is used for the description of equations. The models being created can be put together from components (submodels), which increases clarity. The next advantage is the possible usage of one component in various models. There are many tools available for model creation and simulation in this language. The SBML language is particularly designed for chemical reaction modelling (genetics, proteomics, enzyme chemical reactions, metabolic pathways). Another very similar language is *CellML*[4] and is designed for cell simulation. It is being developed at the Auckland university. The CellML project contains a storage space, where models by different authors are shared[5] [10].

JSim [9] is a modelling tool, which came into being as a part of the Physiome Project[6]. It uses its own text language *MML* (Mathematical Modeling Language) for the model description. The tool can also cooperate well with SBML and CellML languages. JSim can work with models based on ODE, PDE and discrete[7] equations. JSim is running partially on the client computer and partially on the server. The Physiome project also includes a web-based model repository.

Modelica is an acausal object-oriented language aimed at complex physical system modelling. It was developed by an international organisation Modelica Association. The language can model hybrid systems described with algebraic, differential and discrete equations. Libraries with predefined blocks that model several physical and technical elements (e.g. electrical, mechanical and heat components, mathematical functions etc.) are a part of the language. Well known commercial modelling and simulation environments are Dymola and MathModelica. MathModelica enables direct interconnection with the integrated computational system Mathematica.

[1] Acausal libraries have been arising in Simulink recently, e.g. Simscape.
[2] http://www.mathworks.com/products/simulink/
[3] http://www.sbml.org
[4] http://www.cellml.org
[5] http://models.cellml.org/cellml
[6] http://www.physiome.org
[7] Discrete equations are often used to describe phenomena where values of certain quantities or even whole equations are being altered depending on some conditions (usually inequalities).

OpenModelica is the most widely used open environment. It is developed by Open Source Modelica Consortium in cooperation with Linköping University and other 10 universities and 14 companies. Part of the OpenModelica software package are several tools: OMEdit – graphical tool for creation of models, OMC – compiler of Modelica, OMShell – tool for simulation and other actions with models from command line, OMNotebook – for teaching Modelica (creates similar notebooks as Mathematica.), OMOptim – for model optimisation and others.

It is possible either to write the model textually from scratch or use and connect blocks from Modelica libraries or to create your own blocks. The complete model is translated in C. It is further compiled with runtime and numerical solver into an executable application. Everything is of course done automatically.

OMEdit also contains a simulation centre, that can run models, plot results into graphs etc. OMEdit can even simulate interactively, i.e. it allows a user to change model parameter values during simulation and directly observe effects of these changes.

3 FMI

It is obvious that many different simulation languages and tools are used. It is often purposeful, because to describe certain problems, some languages are more suitable than others. Standardised interface *FMI*[8] (Functional Mock-up Interface) was designed in order to make cooperation of different simulating tools and languages possible. The interface enables creating a model in one tool and running it in another. It also enables co-simulation, i.e. creating a model consisting of several interconnected submodels written in different simulation languages.

FMI defines interface in C language to be implemented by functions that represent specific model equations. Interface implemented with a particular model is called *FMU* (Functional Mock-up Unit). Functions of FMU are called by the numerical solver in every time step of the calculation. Models are distributed in zip archive, that contains in addition to FMU files also standardised XML file with description of the model. The FMU files in the archive might be either in source code in C or translated into DLL.

Many simulation tools support FMI. The most common are e.g. Dymola, Matlab/Simulink, JModelica and CATIA. FMI support in OpenModelica is in development.

4 Models on Web

Creation of e-learning applications for medicine [6] based on physiological models is one of the main goals of the Laboratory of Biocybernetics and Computer Aided Teaching at First Faculty of Medicine, Charles University in Prague. These applications are intended mostly for students of medicine, but applications for biology teaching at elementary and high schools will be developed, too.

Representation of the model should be as instructive as possible. Description with equations and visualisation of results in graphs is not suitable. The model is therefore supplemented with graphical user interface, that represents individual components of the model (usually organs), their mutual interaction and current state, see Figure 2. Students can understand the physiological meaning of the model and the principle of the demonstrated effects far better this way. The interconnected model and graphical interface is called *simulator*.

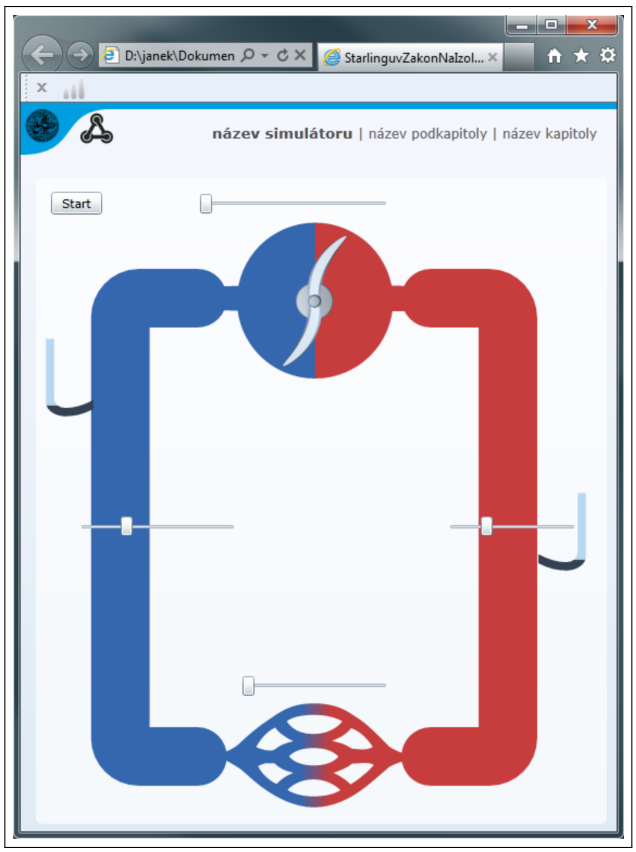

Figure 2: A simple systemic circulation simulator.

Modelica language is used to implement models in this laboratory. In order to remove barriers in simulator utilisation for students and teachers, they are being developed on the *.NET* platform. This enables for them to be run directly in the web browser [8], using plugin *Silverlight*, which is already a part of Windows[9].

To make this possible, the OMC (Open Modelica Compiler) has been expanded in such a way, that it enables model translation also into the C# language and the runtime and numerical solver have been implemented in the F# language. C# is directly supported by Silverlight. For F# support the F# runtime needs to be installed.

[8] http://www.modelisar.com/

[9] For Linux and Mac OS there is an alternative opensource plugin *Moonlight*.

Subsequently, all models run directly in the web browser without a need for further installations.

A long term goal of the laboratory is to build up a sophisticated web based repository of physiological models. A simulation centre was implemented to make possible solving any model of the repository directly in a web browser. This application allows user to set parameters and initial conditions of model and run the computation.

Selected quantities are plotted into a graph during computation. It is even possible to change parameters of the model on the fly. Creation of the web based Modelica editor has been proposed. Implementation of the new Modelica compiler in .NET languages would be too labour-intensive, therefore the translation will be realised on the server side.

This web-based simulation tool will be directly interconnected with the model repository. This will enable anybody to run the models as well as modify them or create new ones. By doing that the repository will continually be growing.

5 Conclusion

There are many languages suitable for physiological model description. Some of them are directly aimed at biological and medical simulations. Some languages and simulation tools are compatible and can cooperate. Increasingly more modelling tools support the FMI interface, which enables such compatibility. In order to make simulation tools and models even more widely available, a creation of an innovated model editor in Modelica language has been proposed. This editor will run in a web browser and will have a direct access to the web model repository. For this purpose, Modelica compiler was already extended and new simulation runtime, solver[10]and simulation centre was implemented.

Acknowledgments

The paper was supported by the project SVV-2011-262514 of Charles University in Prague.

[10]Author participate on implementation of the simulation runtime and solver.

References

[1] Anrew Finney, Michael Hucka, Benjamin J. Bornstein, Sarah M. Keating, Bruce M. Shapiro, Joanne Matthews, Benjamin K. Kovitz, Maria J. Schilstra, Akira Funahashi, John Doyle, and Hiroaki Kitano. System Modeling in Cellular Biology, From Concepts to Nuts and Bolts. The MIT Press, 2006.

[2] P.A. Fritzson. Principles of object-oriented modeling and simulation with Modelica 2.1. IEEE Press, 2004.

[3] Kofránek J. and Velan T. Components for Golem. Simuling and Control Web utilisation for creation of physiological functions simulator. (Komponenty pro Golema. Využívání simulinku a control webu při tvorbě simulátoru fyziologických funkcí. In Objekty 1999, pages 189–204, Praha, 1999. Česká zemědělská univerzita.

[4] M. E. Jackson and J. W. Gnadt. Numerical simulation of nonlinear feedback model of saccade generation circuit implemented in the labview graphical programming language. Journal of Neuroscience Methods, 87(2):137–145, 1999.

[5] J. Kofránek, M. Mateják, and Privitzer P. Causal or acausal modeling: labour for humans or labour for machines. In Technical Conmputing Prague, pages 1–16, 2008.

[6] J. Kofránek, S. Matoušek, J. Rusz, P. Stodulka, P. Privitzer, M. Matěják, and Tribula M. The atlas of physiology and pathophysiology: web-based multimedia enabled interactive simulations. In Computer Methods and Programs in Biomedicine, volume 104(2), pages 143–153, 2011.

[7] G. Lipovszki and P. Aradi. Simulating complex system and processes in labview. Journal of mathematical Sciences, 132:629–636, 2006.

[8] P. Privitzer, J. Šilar, M. Tribula, and J. Kofránek. From model to simulator in a web browser (Od modelu k simulátoru v internetovém prohlížeči). In MEDSOFT, pages 149–169, 2010.

[9] G. M. Raymond, E. A. Butterworth, and J. B. Bassingthwaighte. Jsim: Mathematical modeling for organ systems, tissues, and cells. The FASEB Journal, 21(6):827, 2007.

[10] Tommy Yu, Catherine M. Lloyd, David P. Nickerson, Michael T. Cooling, Andrew K. Miller, Alan Garny, Jonna R. Terkildsen, James Lawson, Randall D. Britten, Peter J. Hunter, and Poul M. F. Nielsen. The physiome model repository 2. Bioinformatics, 27(5):743–744, 2011.

