From Art to Industry: Development of Biomedical Simulators

Kofranek, Jiri; Andrlik, Michal; Kripner, Tomas; and Stodulka, Petr

Abstract — *The authors' methodology of creating e-learning content for the students of physiology, pathophysiology and biomedicine at the Faculty of Medicine is presented here. The design process from a formalized description of physiological reality to interactive educational software is described. Various tools are used during the design – starting with the numerical simulation software Matlab/Simulink, through Macromedia Flash for interactive animations, Control Web or MS Visual Studio for user interface design to web publishing tools including the Macromedia Breeze learning management system. Also, various professions are involved – teachers, physicians, simulation/modeling experts, graphic designers and programmers. The aim is to provide students with software that helps them understand the complex dynamics of physiological systems.*

Index Terms — *Computer aided learning, e-learning, medicine, physiology, simulation models*

1. *Introduction*

THE process of medical e-learning content creation is a complex one. Interactive learning material promises to be more efficient than hypertext versions of medical textbooks with passive images, animations and video sequences. To provide interactivity, simulation models of real physiological systems are used to drive the animations based on the user's actions. The Department of Biocybernetics and Computer Aided Teaching possesses a multidisciplinary team and software development tools that cover all stages of the design process of educational simulators in medicine. The connection between multimedia presentation and physiological simulation models facilitates comprehensive demonstration of complex regulatory processes and their disorders to medical students and physicians alike.

Physiological simulations have a long tradition. Formalized description of the physiological reality starts with the pioneering work of Grodins et al. [1] in the end of 1960's. A classical milestone was a description of circulation by Guyton et al. [2] in the first half of 70's, the first large-scale model, which attempted to cover in broader perspective the physiological interconnections between circulation, breathing and kidneys. Since then, the number of studies that use computer models (mainly for evaluation and interpretation of experimental data) is steadily increasing. Likewise, simulation models can be used in clinical applications and medical equipment control (so called minimal models with a simple structure and a small number of parameters that can be easily identified from a particular patient's data) – e.g. a model of glucose metabolism [3].

Various approaches have been taken in the field of educational simulators. The aim of this paper is not to analyze the simulation models behind the simulators but rather the methods of their formalization (according to the Physiome project – http://physiome.org) and incorporation into educational software. Special attention is paid to the creation of web accessible graphical user interfaces – Microsoft .NET platform in combination with Macromedia Flash animations is provided as an alternative to commonly used Java applets or pure Flash web-based simulators [4]. The paper summarizes the authors' experience and achievements in this field.

2. *Educational Software Design*

Just as the reception of a text-book by students depends on the author's ability to explain complex material in an illustrative and comprehensive way, the key to success of multimedia educational software is a good scenario. Thus the *design cycle* of our programs (see **Fig. 1**) begins with the creation of scenarios. A scenario comprises not only of textual material, but also of the cartoon strips of the "storyboard" which will later help the graphic designers create graphics and animations.

There are more professions involved throughout the development cycle besides the already mentioned graphic designers and physiology teachers (responsible for the scenario). On the level of constructing simulation models, cooperation has to be established among system engineers (skilled in mathematical modeling), physiologists and physicians eventually, if the model is supposed to be applied in clinical medicine.

Educational software creation is concluded by application programmers and web designers,

Revised manuscript received May 25, 2005. This work was supported by the Research and Development Initiative of the Czech Ministry of Education MSM 111100008 (physiome.cz).

Authors are with the Institute of Pathophysiology, 1st Faculty of Medicine, Charles University in Prague, Czech Republic

Contact person: dr. Jiri Kofranek, e-mail: kofranek@cesnet.cz

responsible for wrapping the simulation models with graphical user interfaces according to the scenario. The software then enters the testing phase in education and further refinements are made, if required by the teachers.

More on the respective phases of development will be given in the following paragraphs.

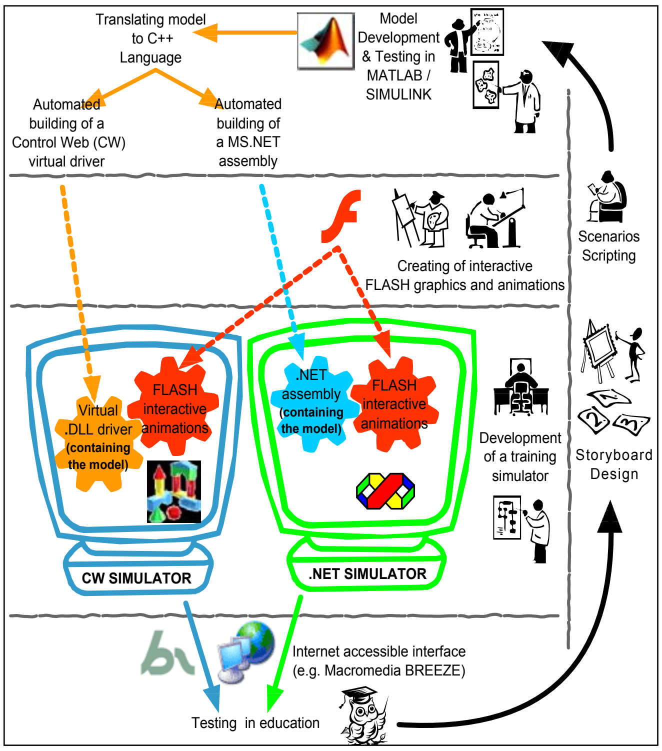
Translating model to C++ Language
Model Development & Testing in MATLAB / SIMULINK
Automated building of a Control Web (CW) virtual driver
Automated building of a MS.NET assembly
Scenarios Scripting
Creating of interactive FLASH graphics and animations
Virtual .DLL driver (containing the model)
FLASH interactive animations
.NET assembly (containing the model)
FLASH interactive animations
Development of a training simulator
Storyboard Design
CW SIMULATOR
.NET SIMULATOR
Internet accessible interface (e.g. Macromedia BREEZE)
Testing in education

Fig. 1 The development cycle of educational software

A. *Simulation models, chips and libraries*

The heart of an e-learning application is a simulation model of a physiological system. Such a model is usually based on a system of differential equations, which have their origin in measurements on humans (either healthy or under pathological conditions). The biological reality is transformed into a formalized description in the language of mathematics.

On the other hand, a purely mathematical description is not suitable for further communication of the system engineer with physiologists or physicians (who are to judge the correctness and quality of the model). Thus MATLAB/Simulink graphical modeling/simulation environment has been adopted and physiological systems have been deconstructed into atomic blocks – *simulation chips*. These blocks represent elementary physiological subsystems; simulation models are formed by networks of these interconnected chips, thus keeping the organized structure of the original physiological system. A chip communicates with other simulation chips through defined inputs and outputs (corresponding to physiological quantities). All the mathematical expressions needed to implement the physiological function are hidden inside the chip. The simulation chips are then the common language of system science and physiology.

Once a simulation chip has been created, it can be of course reused in a number of simulation models. This idea gave birth to a Simulink library of physiological blocks – the *Physiology Blockset* (see **Fig. 2** and [13]).

B. *Containers for simulation models*

The MATLAB/Simulink environment is particularly good for professional creation, tuning and testing of simulation models, but due to its complexity it is not suitable for direct interaction with users. Solutions allowing rapid graphical user interface (GUI) development, and having sufficient computational power, at the same time had to be sought.

Two successful candidates for this post are available, both running on the Windows platform, are *Control Web* (CW), an industrial process control and visualization tool [6] and *Microsoft Visual Studio .NET*, a general purpose developer environment. Both of these tools can host the simulation model in the form of an external *.dll* library, provide rich means for creating GUI's and also allow incorporation of interactive Flash animations.

Fig. 1 reveals that the process of creation of the model library is automated with the help of Simulink Real Time Workshop (RTW), which translates simulation models to C++ language, links them with a proper differential equation solver and builds a *.dll*. The target platform of the *.dll* depends on a RTW template; templates for creating both CW driver and MS .NET assembly were created in our laboratory [7].

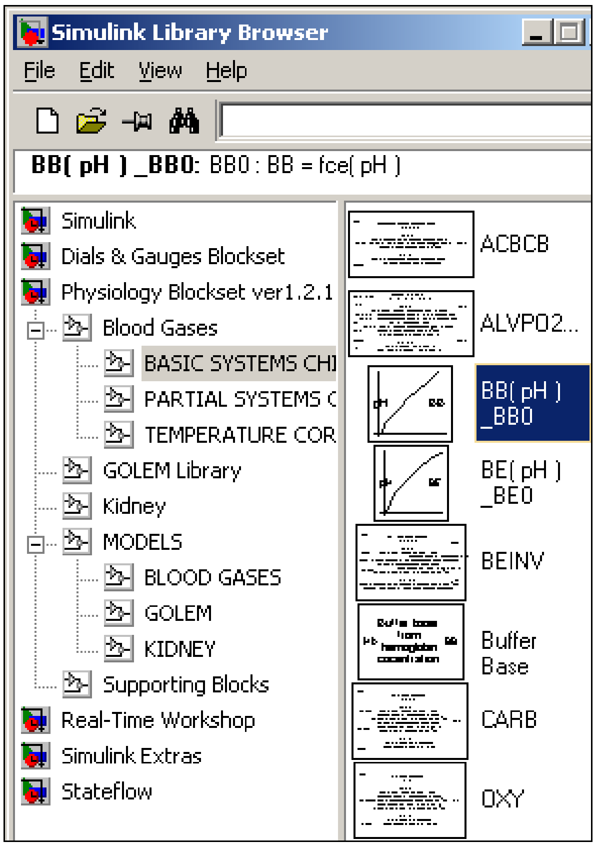

Fig. 2 Simulation chips, the bricks of simulation models, organized in a MATLAB / Simulink library

C. Creating graphical user interfaces

Besides the standard GUI elements (buttons, sliders, text boxes, data grids...), there is a need for special controls in our e-learning applications like knobs and meters for manipulating the simulation model inputs and outputs. This explains the choice of Control Web as one of the simulation front-ends, as it offers a wide range of these "industrial level" controls.

Some custom components also had to be designed from scratch, e.g. intelligent graphs and controls for manipulating vector inputs. When the graphical capabilities of the above mentioned integrated development environments are not sufficient, Macromedia Flash can be used to create interactive animations that can be inserted in both CW and MS .NET application, as well as in web pages. The Flash animations can communicate through the ActiveX interface, which means that they can be controlled from their host application.

An interesting example of a graphical control created in Flash can be seen in the simulator of mechanical properties of a skeletal muscle [8], where the control, in the form of a muscle fixed to an experimental stand, serves as an input and an output at the same time. The user can drag the muscle to a desired length using a mouse and during the electrical stimulation; the same graphic visualizes the muscle contraction.

In most cases, interactive graphics created with Flash are used to visualize the outputs of simulation models, e.g. a dilating/contracting blood vessel, a change in the respiratory volume and the breathing frequency of an alveolus.

D. UCM architecture and state management

As the structure of the simulation model gets more complicated and the educational scenario grows more complex, it is not efficient to connect the model directly to the presentation layer. *UCM architecture* and *State charts* offer an elegant solution to this problem.

UCM [4] stands for *User interface / Control object / Model layers*. The UCM architecture dictates a separation between functional layers, such as the user interface and simulation models, that are interconnected through the control object layer. The control object receives and handles events and messages from the interface elements (e.g. pressing the buttons, turning the knobs etc.) The control object may also receive events and messages from the model layer. Only the control object is allowed to communicate with the model layer. The control and model layer objects communicate by sending messages to each other. This centralizes access to the model layer and makes it perfectly clear what information the interface needs, when (in which context) the information is sought and how the internal functions can communicate with the interface.

The structure of the control object can be based on the *state engine* data structure combining a mechanism for managing context (the *state network*) with a mechanism for handling messages (see **Fig. 3**). The state network is a mechanism for keeping track of the current learning context of the user and of the current context of the simulated objects. The context can be changed as the result of messages or events from the user interface or from the model layer. As the current context changes the state network can be programmed with the actions to be executed – such as sending messages to the user interface layers (and consequently change some visual objects) or sending messages to the model layers (and subsequently change some model inputs, or request some model outputs).

For the visual description of the behavior of state networks we use a visual design methodology and notation called *statecharts* [5]. It is an extension of deterministic finite state automata (or state machines). For the design of the state network we use Stateflow, an extension of Simulink. Using Stateflow we can graphically describe and test the behavior of state networks for the tracking of simulated objects and learning context.

We have designed a special wizard that automatically converts the Stateflow statecharts into Microsoft .NET assemblies. These .NET components can communicate with the user interface layer (created with Microsoft .NET Visual Studio and with Macromedia Flash components incorporated in the Microsoft .NET environment) and with the model layer (containing simulation models developed using Simulink and converted into .NET components).

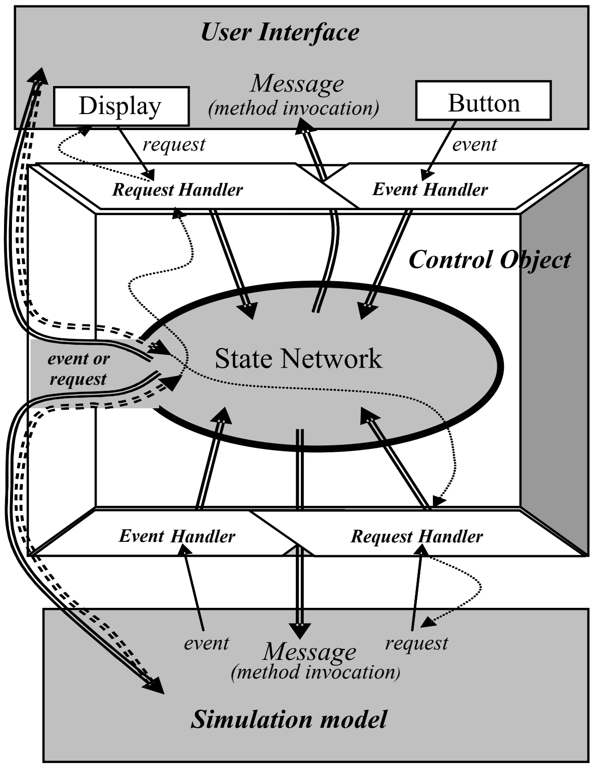

Fig. 3 UCM architecture and the state management

E. Putting it all together

The preceding paragraphs covered the characteristic phases of a medical simulator development – writing a scenario, simulation model design, converting the model to a linkable library, creating GUI elements and eventually a statechart. All these are put together in a standalone application – a simulator.

Model inputs and outputs (available through the linked library interface) are connected to respective control and visualization elements of the user interface (possibly through the Stateflow "control object") and the simulator enters the final stage of its development process – testing and evaluation in education. This might, of course, result in changes to the simulator or even raise demands for new simulation models. For an example of a user interfaces combining custom .NET controls and Flash animations, see **Fig. 4**.

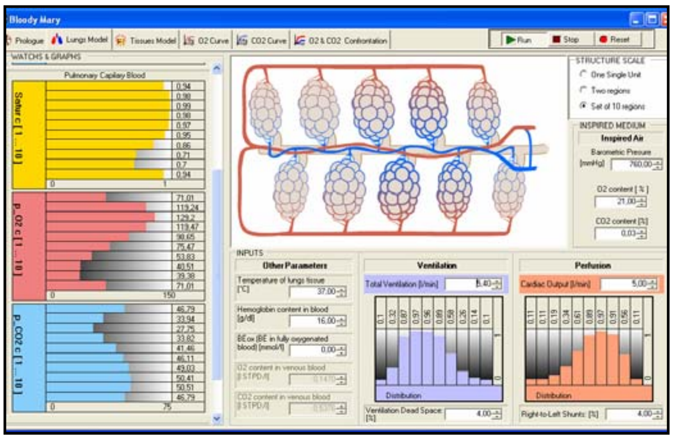

Fig. 4 An example of a simulator user interface combining custom MS .NET controls (bar graphs) with Flash animations (alveoli)

3. DEPLOYMENT OF SIMULATORS IN EDUCATION

There are two new interesting features brought to e-learning by simulation models – one is the scalability of the models and the other is the possibility of breaking the control loops of the model.

As for the *scalability* (both on the structural and time scale), one has to keep in mind that one level of detail of a model is not suitable in all situations. Consider a model of respiration – when explaining the disorders of the breathing mechanics, the time-course of every breath counts, while when tracking the effects of acid-base balance disorders on respiration, the time constant of the response is in the order of hours. The detailed model should be replaced with a simplified model with lumped parameters.

Another observation from the teaching experience with the simulation models is that better understanding of physiological functions is achieved through the *possibility of breaking the control loops of physiological regulations*. It means that just one input can be changed, while keeping the other inputs constant, revealing the dynamics of the subsystem (so called "ceteris paribus" principle).That's why our models allow students to switch various physiological quantities to manual control, thus avoiding the compensatory effect of other control loops.

A good example of a large scale simulator of the body-fluid balance, respiration, circulation and renal function is Golem [10], [11], **Fig. 5**. The simulator allows the user to set specific pathological situations and practice therapeutic interventions by changing input values. This "virtual therapy" promotes better understanding of the physiological and pathophysiological mechanisms without putting patient's health at stakes.

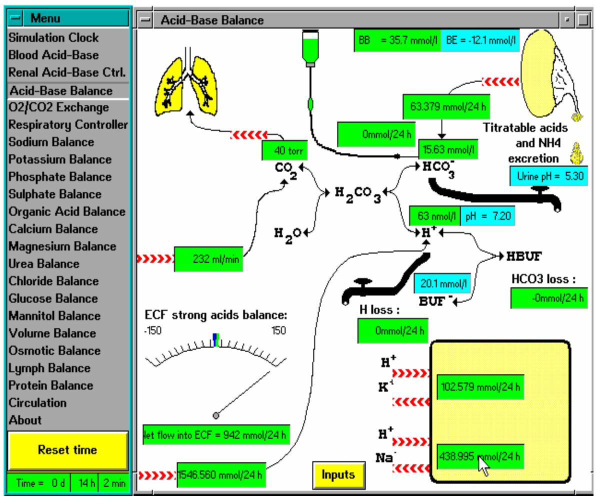

Fig. 5 Acid-base balance model in the Golem simulator, implemented in the Control Web environment

4. Web Accessible Simulators

The last problem to solve is making the simulators, as a part of the e-learning content, easily accessible to a large number of students. These efforts resulted in several approaches to web-accessibility of the models.

One of the possibilities is running the model on a server and exchanging only the input and output data with a client. The drawbacks here are obvious – for every connected client, there has to be a separate instance of the model on the server and the amount of data transferred between the server and a client per unit time is limited.

A better solution is *running the simulations on a client* while browsing the e-learning content on the internet. A direct method would be using a purely Flash simulator interface, because Flash applications can execute in the plug-in of a web browser. This method is feasible for simple models only (e.g. simulating the muscle contractions [8]), as Flash is not able to utilize the model in the form of the *.dll* library. The model equations have to be implemented in the native language of Flash – ActionScript, which is very slow due to its interpreted nature.

Currently the most promising architecture for delivering the e-learning content with simulation models over the internet is displayed in **Fig. 6**. The educational material is conceived as a web page containing text, graphics and Flash animations (possibly a Macromedia Breeze presentation) with hyperlinks to simulation models. The proper downloading and execution of the simulation models, as well as managing their context within the web presentation, is ensured by the model dispatcher, a component that is downloaded and installed prior to running the first simulation model [12].

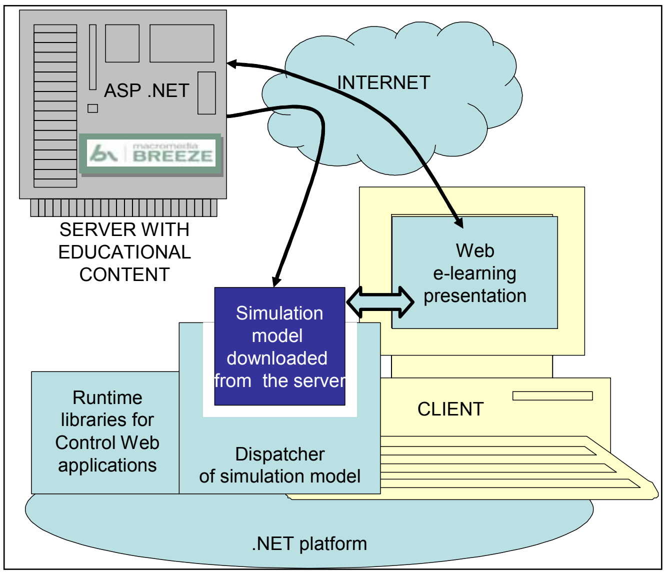
ASP .NET
BREEZE
SERVER WITH EDUCATIONAL CONTENT
INTERNET
Web e-learning presentation
Simulation model downloaded from the server
CLIENT
Runtime libraries for Control Web applications
Dispatcher of simulation model
.NET platform

Fig. 6 Proposed structure of a modern e-learning system combining web accessibility with locally running .NET (or CW) simulators

The *Macromedia Breeze* learning management system offers a couple of beneficial features for e-learning content development – rapid creation of web accessible presentations from Microsoft PowerPoint slides accompanied by a spoken commentary and the possibility of preparing on-line test (with score reporting for individual students), to name just two.

5. Conclusion

The creation of modern educational software represents a challenging and complicated project requiring the team cooperation of various professionals: *Skilled teacher / physiologist* - who prepares the scenario (including the basic design of pictures and interactive animations) and tests the final products as a teaching aid. *System analyst* – an expert that designs, formalizes and tunes the simulation models in cooperation with a physiologist. The means of their mutual communication are the simulation chips in the Matlab/Simulink environment. *Graphic designer* - designs and constructs graphical components for simulator user interfaces and produces interactive animations in Macromedia Flash. *Application programmer / Web designer* - utilizes Microsoft .NET or Control Web as a container for the simulation model. He connects it with the interactive animations, and other multimedia features, and programs the actual educational application. And last but not least – the *student*, for whom the whole product is intended and whose comments and opinions after testing the program should be of high interest to the teacher and the developers.

It is clear that convenient developer tools and design methodology save time and money – particularly using appropriate specialized tools for the different types of tasks. We use MATLAB / Simulink (from MathWorks) for the development of simulation models, Stateflow (a MATLAB toolbox) for the visual description of the interactive scenario as statecharts, Macromedia Flash for the graphical design and scripting of the interactive animation components and Microsoft Visual Studio .NET or Control Web for the development of the final form of the educational program. For the interconnection of simulation models with the interactive multimedia user interface we use UCM architecture. The architecture describes the separation between the user interface and the underlying model behavior layers. When applied to the creation of interfaces, the architecture promotes good design practice because it centralizes the coordination of the user interface, making the interface behavior easier to understand, design and validate.

REFERENCES

[1] Grodins F.S., Buell J., Bart A.J.: "Mathematical analysis and digital simulation of the respiratory control system," J Appl Physiol., vol. 22, 1967: pp. 260-276.

[2] Guyton A.C., Coleman T.A. and Grander H.J.: "Circulation: Overall regulation," Ann. Rev. Physiol., 1972, pp. 13-41

[3] Cobelli C., Bettini F., Caumo A.and Quon M.J.: "Overestimation of minimal model glucose effectiveness in presence of insulin response is due to undermodeling," Am. J. Physiol. vol. 275, (Endocrinol. Metab. vol. 38), 1998, pp. E1031-E1036

[4] Kayne J., Castillo D.: "Flash MX for interactive simulation," Thomson Delmar Learning, 2003. ISBN 1-4018-1291-0.

[5] Harel D.: "Statecharts: a visual formalism for complex systems," Science of Computer Programing, 8, 1987, pp. 231-274.

[6] Bily, R., Cagas P. et al.: "Control Web 2000 - Object Oriented Industrial Information and Control System,". Computer Press, *Prague*, 1999 (in Czech). ISBN 8072262520.

[7] Kofranek J., Andrlik M., Kripner T., Masek J.: "From Simulation chips to biomedical simulator," in Amborski K, Meuth H, (eds.): Proc. of 16th European Simulation Multiconference, *Darmstadt, Germany*, 2002. ISBN 90-77039-07-4, pp. 431-435.

[8] Wünsch Z., Kripner, T., Kofranek, J.: "Mechanical Properties of a Skeletal Muscle – Multimedia Simulation Educational Software," Proc. of 5th EUROSIM Congress on Modeling and Simulation. (eds: G. Attiya and Y. Hamam), *Paris*, 2004

[9] Smutek D., Kofranek J., Maruna P.: "Mathematical Model of Cytokines and Acute Phase Proteins Induction after Polytrauma and after Infectious Stimulus. Potential Diagnostic context," Proc. of Intl. Trauma Congress, 11-14 June 2003, *Durban*, South Africa, p 51.

[10] Kofranek J., Anh Vu LD, Snaselova H., Kerekes R., Velan T.: "GOLEM – Multimedia simulator for medical education," in: Studies in Health Technology and Informatics, vol. 84, Proc. of the 10th World Congress on Medical Informatics MEDINFO (V.L. Patel, R. Rogers, R. Haux (Eds.)), IOS Press, *Amsterdam, Berlin, Oxford, Washington DC*, 2001.

[11] Kofranek J., Andrlik M., Kripner T., Masek J., Velan T.: "Simulation chips for GOLEM – multimedia simulator of physiological functions," in Simulation in the Health and Medical Sciences (James G. Anderson, M. Katzper (Eds.)). Society for Computer Simulation International, Simulation Councils, *San Diego*, 2002, pp. 159-163

[12] Kofranek J., Andrlik M., Hlavacek J., Stodulka P.: "E-learning applications with simulation models in Macromedia Breeze" (in Czech). Proc. of Medsoft conference, Action M, *Prague*, 2005, *in press*.

[13] Andrlik M., Kofranek J., Kripner T.: "Physiology Blockset for Matlab/Simulink", open source software library http://patf-biokyb.lf1.cuni.cz, 1st Faculty of Medicine, Charles University, *Prague*, 2004