**BioUML: VISUAL MODELING, AUTOMATED CODE GENERATION AND SIMULATION OF BIOLOGICAL SYSTEMS**

**Kolpakov F.**, 1, **Puzanov M.**, 1, 2, **Koshukov A.**, 1, 2

1 Institute of Systems Biology OOO, Novosibirsk, 630090, Russia; 2 Design Technological Institute of Digital Techniques, SB RAS, Novosibirsk, Russia

* Corresponding author: e-mail: fedor@biouml.org

**Key words:** metamodel, simulation engine, Java, MATLAB, SBML, biological systems

**SUMMARY**

**Motivation:** Reconstruction of complex biological systems from a huge amount of experimental data requires a formal language that is suitable both for human and computer.

**Results:** BioUML is integrated extensible open source Java workbench that adopts visual modeling approach for formal description and simulation of complex biological systems. Its core is metamodel that provides an abstract layer for comprehensive formal description of wide range of biological and other complex systems. Content of databases on biological pathways as well as SBML and CellML models can be expressed in terms of the metamodel. This formal description can be used both for visual depiction and editing of biological system structure and for automated code generation to simulate a model behavior. BioUML workbench provides two alternative simulation engines using Java and MATLAB: 1) Java simulation engine – workbench automatically generates and compiles Java code on the base of visual model (diagram) of a biological system. 2) MATLAB simulation engine – workbench automatically generates code for MATLAB and invokes MATLAB engine to simulate a model behavior. Both simulations engines passed 100 % SBML semantic test suite.

**Availability:** http://www.biouml.org.

**INTRODUCTION**

Reconstruction of complex biological systems from a huge amount of experimental data requires a formal language that can be easily understood both by human and computer. It is known that graphical depiction of complex system is the most suitable way of understanding of its structure by human. Graphical notation allows human to completely and formally specify model so computer programs can analyze the model and simulate its behavior (Lee, 2001). Thus the problem of modeling and simulating of complex systems can be significantly simplified for researchers by using computer systems providing visual modeling. This approach is widely used in engineering and computer science. Examples are: MATLAB/Simulink (http://www.mathworks.com), AnyLogic (http://www.xjtek.com), UML (http://www.omg.org/uml/).

BioUML (http://www.biouml.org) – Biological Universal Modeling Language – is integrated extensible open source Java workbench that adopts visual modeling approach for formal description and simulation of complex biological systems (Fig. 1). Another distinctive feature of BioUML workbench is tight integration with databases on biological
pathways, query engine allows user to find interacting components of the system and show results as an editable graph (Kolpakov, 2004).

Figure 1. Data flow in BioUML workbench – left; parsing and conversions of mathematical expressions – right.

METAMODEL

The core of BioUML is a metamodel that provides an abstract layer (compartmentalized attributed graph) for comprehensive formal description of wide range of biological and other complex systems. Content of databases on biological pathways as well as SBML (Hucka et al., 2003) and CellML (Lloyd et al., 2004) models can be expressed in terms of the metamodel. This formal description can be used both for visual depiction and editing of biological system structure and for automated code generation to simulate a model behavior. Metamodel is problem domain neutral and splits the system description into three interconnected levels:

- graph structure – the system structure is described as compartmentalized graph;
- database level – each graph element can contain reference to some database object;
- mathematical model – any graph element can be element of mathematical model.

Currently BioUML supports following mathematical elements: variable, formula, equation, event, state and transition. Fig. 2 demonstrates graphical depiction of these elements.

Special BioUML diagrams markup language (DML) is developed to store BioUML metamodel instance in XML format. Diagram structure description is divided into two parts: 1) diagram structure model – it describes the graph structure, location of diagram elements and contains references to associated with them database objects; 2) executable model – stores mathematical model associated with graph. Detailed description of DML format is available at http://www.biouml.org/dml.shtml.

SIMULATION ENGINE

Main parts of simulation engine are: code generator, formulas processor, algebraic equations solver and results writer. BioUML provides powerful formula processor that parses text and MathML expressions, result is presented as syntax tree and used by formatters to generate corresponding Java or Matlab code (Fig. 1, right).
Currently BioUML workbench provides two alternative simulation engines:

1) Java simulation engine – workbench automatically generates and compiles Java code on the base of visual model (diagram) of a biological system. For simulation we have adopted odeToJava library (Patterson, Spiteri, 2003) that provides methods for numerical solutions both stiff and non-stiff systems of ODEs. For solving algebraic equations Newton solver is used.

2) MATLAB simulation engine – workbench automatically generates code for MATLAB and invokes MATLAB engine to simulate a model behaviour using JMatlink library (http://www.held-mueller.de/JMatLink/).

Both simulations engines pass 100 % SBML semantic test suite that provides a set of valid SBML models with a simulated time course data (Finney, 2004). Test details are available at: http://www.biouml.org/sbml_tests/overview.html.

DISCUSSION AND FURTHER DEVELOPMENT

Several XML dialects like CellML, SBML are being developed for formal description and simulation of biological pathways. However they do not address problems of graphical notation for pathways visualization and tight integration with existing databases. Other approaches, for example BioPax (http://www.biopax.org), try to map information from different databases on biological pathways into common format; however BioPax format is not suitable for storage of mathematical models and simulation. The suggested approach should fill this gap – from one hand majority of models that can be expressed on SBML or CellML can be mapped into corresponding BioUML models, from the other hand information from different databases on biological pathways can be queried and presented as a set of diagrams.

Figure 2. Simulation of toy hybrid model of cell cycle using BioUML workbench. Main cell cycle phases (G1, S, G2, M) are described as states. There are 4 transitions between these states: 3 transitions with time delay and 1 conditional transition – cell enters into S phase only when cytoplasm volume exceeds the specified threshold. Cell growth is described by simple ODE as rate of growth of cytoplasm volume, for different states growth rate is different.
Initial BioUML graphical notation and simulation engines have supported only ordinary differential equations for simulation of biological pathways (Kolpakov, 2004). Extended graphical notation (Fig. 2) and new version of simulation engines support piecewise functions, time delays, algebraic equations, events, states and transitions that allows user simulate wide range of biological systems. Nevertheless there are some limitations – mainly these are spatial models, for example models with diffusion and biomechanical models. We hope to overcome these limitations during further BioUML workbench development – now we are developing new plug-in for one-dimensional modeling of a vascular network in space-time variables. This task requires solving of one-dimensional partial differential equations (PDE).

Other direction of BioUML workbench development is connecting two worlds: world of cis-regulatory signals in DNA and world of molecular interaction networks of cells. For this purpose we are developing a set of plug-ins for analyses of nucleotide sequences and their visualization.

ACKNOWLEDGEMENTS

This work was supported by INTAS grant No. 03-51-5218, RFBR grant No. 04-04-49826-a and Siberian Branch of Russian Academy of Sciences (interdisciplinary projects No. 46).

REFERENCES