

BIOUML – OPEN SOURCE EXTENSIBLE WORKBENCH FOR SYSTEMS BIOLOGY

Fedor A. Kolpakov

Biosoft.Ru/DevelopmentOnTheEdge.com, Novosibirsk, Russia;

Design Technological Institute of Digital Techniques SB RAS, Novosibirsk, Russia.

e-mail: fedor@biosoft.ru

Keywords: systems biology, database, visual modeling, plug-in based architecture, Java, Eclipse

Summary

Motivation: With the completion of several genomics initiatives, including the Human Genome Project, researchers are poised to begin the next phase of elucidating how living systems function. This step requires integrated software environment that spans the comprehensive range of capabilities.

Results: BioUML – Biological Universal Modeling Language – is open source extensible Java workbench for systems biology. BioUML's core is a meta model that provides an abstract layer for comprehensive formal description of wide range of biological and other complex systems. Content of databases on biological pathways (TRANSPATH, KEGG/pathways, GeneNet, GeneOntology) as well as SBML and CellML models can be expressed in terms of the meta model and used by BioUML workbench. Plug-in based architecture provides the workbench extensibility and possibility of seamless integration with other tools for systems biology. The workbench consists from Eclipse platform runtime that supports plug-ins registry and a set of plug-ins for database access, diagram editing, biological systems simulation and for integration with MATLAB and SBW/SBML.

Availability: <http://www.biouml.org>

Introduction

Sydney Brenner, 2002 Nobel Prize winner said (Bradford, 2003):

"We now have unprecedented ability to collect data about nature but there is now a crisis developing in biology, in that completely unstructured information does not enhance understanding. We need a framework to put all of this knowledge and data into — that is going to be the problem in biology. We've reached the stage where we can't talk to each other — we've all become highly specialized. We need a framework, a framework where people can come back to us and say, 'Yes, I understand.' Driving toward that framework is really the big challenge."

We believe that BioUML – Biological Universal Modeling Language – is a step in this direction. It is imagined as a language to write a “book of life”. From the user's perspective BioUML workbench (fig. 1) is integrated environment that spans the comprehensive range of capabilities including access to databases with experimental data, tools for formalized description of biological systems structure and functioning, as well as tools for their visualization and simulations.

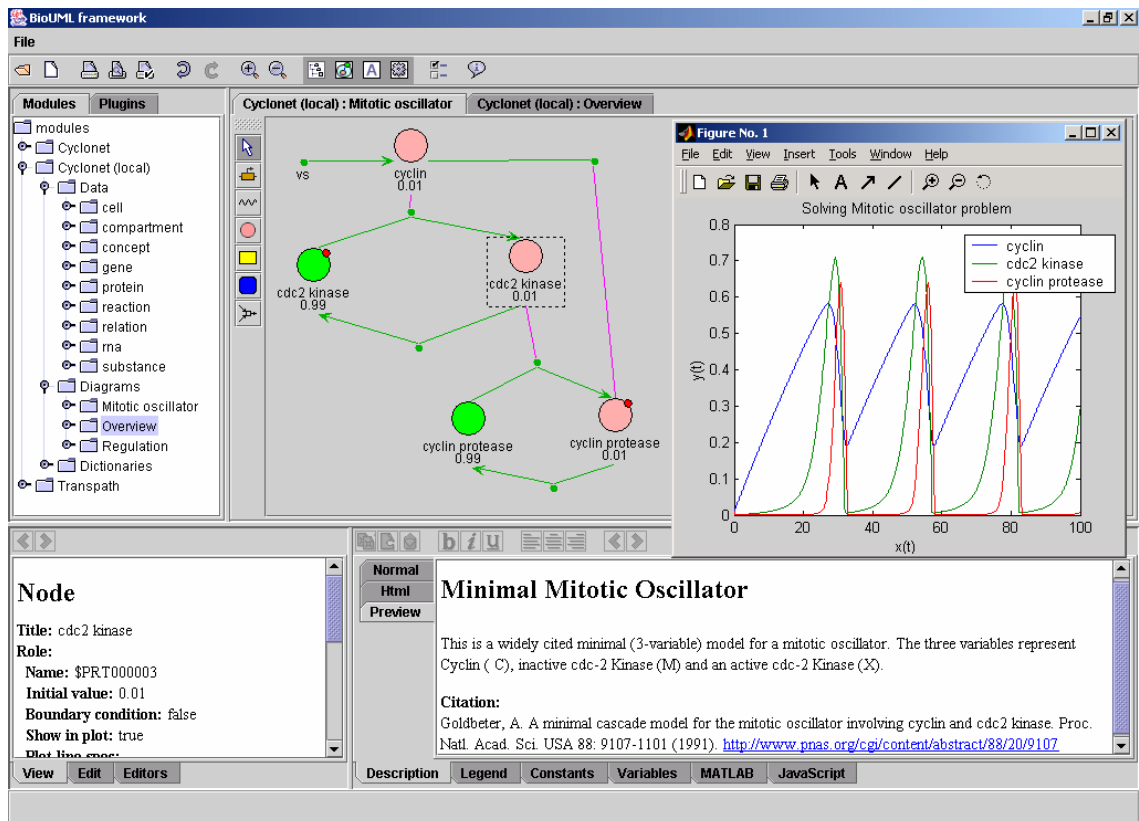


Figure 1. BioUML workbench screenshot. Top left pane – repository pane that shows database modules, here Cyclonet database module; top right pane – diagram editor; bottom left pane – property editor for the selected object; bottom right pane – diagram editor parts, here the diagram description editor; middle right pane – results of the simulation.

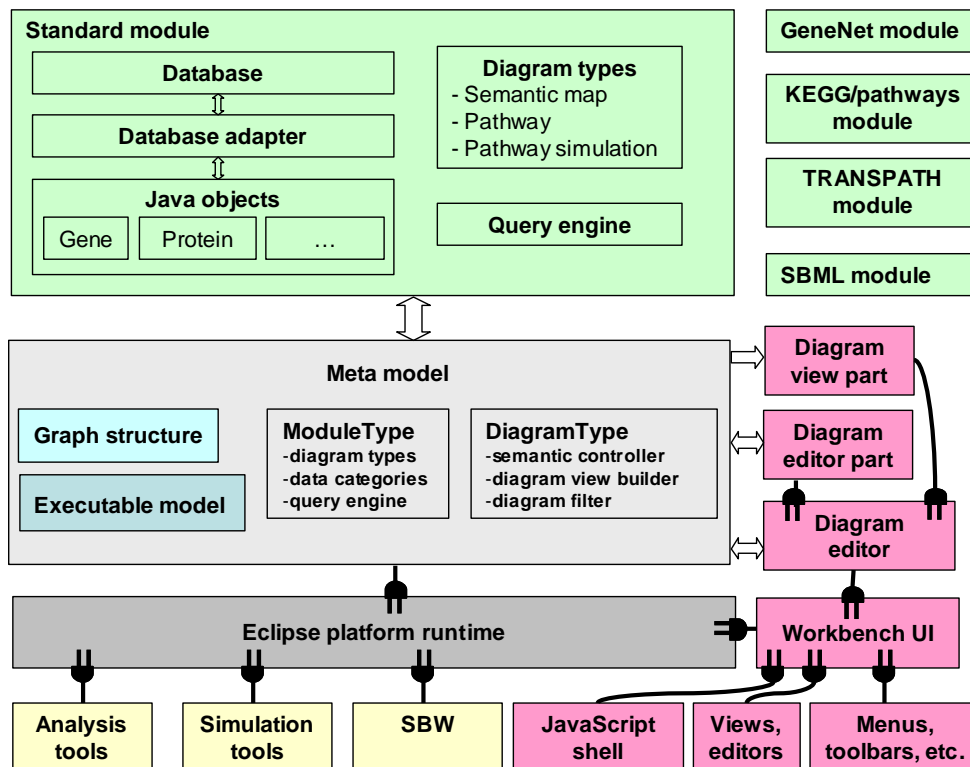


Figure 2. BioUML workbench architecture overview.

Architecture overview

BioUML workbench is a plugin-based application framework (Fig. 2) based on Eclipse platform runtime (IBM, 2003). It consists from a core runtime that supports 'plug-ins' and a set of plug-ins that support database access, diagram editing, simulation, integration with MATLAB and SBW/SBML (Hucka et al., 2002, 2003), etc.

A plug-in is the smallest unit of BioUML workbench function that can be developed and delivered separately into BioUML workbench. Extension points are well-defined function points in the system where other plug-ins can contribute functionality. An extension is a specific contribution to an extension point. Plug-ins can define their own extension points, so that other plug-ins can integrate tightly with them.

Plug-ins are coded in Java. A typical plug-in consists of Java code in a JAR library, some read-only files, and other resources such as images, native code libraries, etc. BioUML workbench installation includes a plug-ins folder where individual plug-ins are deployed. Each plug-in is installed in its own folder. A plug-in is described in an XML manifest file, called plugin.xml, residing in the plug-in's folder. The parsed contents of plug-in manifest files are made available programmatically through a plug-in registry API provided by Eclipse runtime.

The table below lists plug-ins that are included in default installation of BioUML workbench and provides their brief description.

Table 1. Main BioUML workbench plug-ins

Plug-in name	Short description
org.eclipse.core.runtime org.eclipse.osgi.services org.eclipse.osgi.util org.eclipse.osgi org.eclipse.update.configur ator	Eclipse platform runtime. It is used to start the workbench and manage the plug-ins registry.
com.beanexplorer	Rapid UI development framework, extension of JavaBeans technology, http://www.beabexplorer.com
ru.biosoft.access	Heterogeneous biological databases access and management.
ru.biosoft.access.search	Universal database search engines.
ru.biosoft.graphics	Vector graphics library. It is used for diagram painting and editing.
ru.biosoft.workbench	Workbench user interface issues.
biouml.workbench	BioUML meta model, standard data and diagram types workbench user interface issues.
<i>Query engine</i>	
ru.biosoft.graph	Library of graph layout algorithms: orthogonal, hierarchical and force-directed algorithms are provided.
biouml.workbench.graph	Query engine allows user to find interacting components of the system and show results as an editable graph.
<i>Simulation</i>	
ru.biosoft.math	mathematical formulas processing. Formulas are stored in the form of abstract syntax tree. There are a set of parsers and formatters that parse formula text from different formats (text or MathML content markup) and generate corresponding code for MathML, MATLAB or Java. Formula editor.
ru.biosoft.ode	Ordinary differential equations solvers.
org.jfree.chart	Plot visualization library.
biouml.plugins.simulation	Common definition and routines for simulation engine, simulation results and plot visualizations. Also includes own Java simulation engine that generates model

	as Java file, compiles it and simulates the model behavior using ODE solvers that were ported from odeToJava library.
biouml.plugins.matlab	Generates MATLAB files for models simulation, provides integration with MATLAB engine.
<i>JavaScript support</i>	
org.mozilla.javascript	JavaScript/ECMA interpreter.
biouml.plugins.javascript	Scripting support. BioUML workbench provides JavaScript shell where user can start his script for data analysis or simulation. Alternatively BioUML workbench can be started in headless mode that allows user to execute commands or script from command line.
<i>SBML/SBW support</i>	
biouml.plugins.sbml	SBML (Hucka et al., 2003) - Systems Biology Markup Language support.
edu.caltech.sbw	SBW (Hucka et al., 2002) - Systems Biology Workbench.
biouml.plugins.sbw	Provides integration of SBW into BioUML workbench.
<i>CellML support</i>	
biouml.plugins.cellml	CellML (Physiome Sciences, 2001) support.

Meta model

The core of BioUML workbench is meta model 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 CellML and SBML models can be expressed in terms of meta model and then can be visualized and edited as diagram by BioUML diagram editor, simulated using MATLAB or BioUML simulation engine, etc.

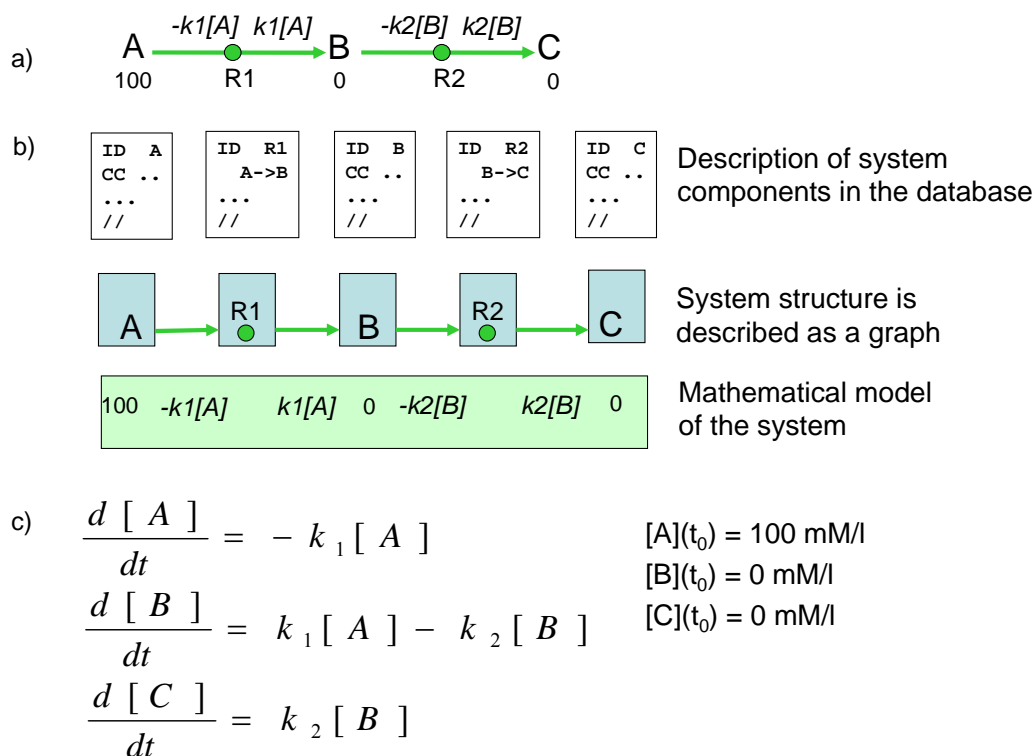


Figure 3. System from two consecutive chemical reactions (a), its formal description using three meta model levels (b), and corresponding mathematical model (c), that can be generated automatically for system simulations.

Meta model is problem domain neutral and splits the system description into three interconnected levels:

1. graph structure - the system structure is described as compartmentalized graph;
2. database level - each graph element can contain reference to some database object;
3. mathematical model - any graph element can be element of executable (mathematical) model.

Figure 3 demonstrates how this approach is applied for modeling system consisting from two consecutive chemical reactions. Here graph nodes representing chemical substances are considered as variables and corresponding graph edges contain right parts of corresponding differential equations. Using this information BioUML workbench can generate MATLAB or Java code for model simulation.

Special BioUML diagrams markup language (DML) is developed to store BioUML meta model 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>

Diagram type concept

To take into account different diagram types and problem domain specificity we have introduced the diagram type concept. Diagram type defines:

- what types of nodes and edge can be used to present on the diagram different components of the system;
- diagram view builder – generates view (image) for each graph element taking into account problem domain peculiarities. For example GeneNet diagram view builder (fig. 4) generates circles as a protein view, rectangle as a gene view and squares for chemical substances;
- semantic controller - provides semantic integrity of the diagram during its editing. It takes into account problem domain constraints, for example if some substance is removed on biological pathway diagram, all related reactions should be removed too;
- filters – hide or highlight diagram elements according to some selection criteria, for example to according gene expression specificity or expression level.

Module concept

The module concept allows developer to incorporate databases on biological pathways into BioUML framework taking into account database peculiarities. Module defines:

- data types that are used for object oriented presentation of the database content;
- mapping of the database content into Java objects using defined data types and vice versa;
- diagram types that can be used to present the database content as a set of diagrams;
- query engine that is used by BioUML workbench to find interacting components of the system. Search results can be shown as graph and edited by user (fig. 4).

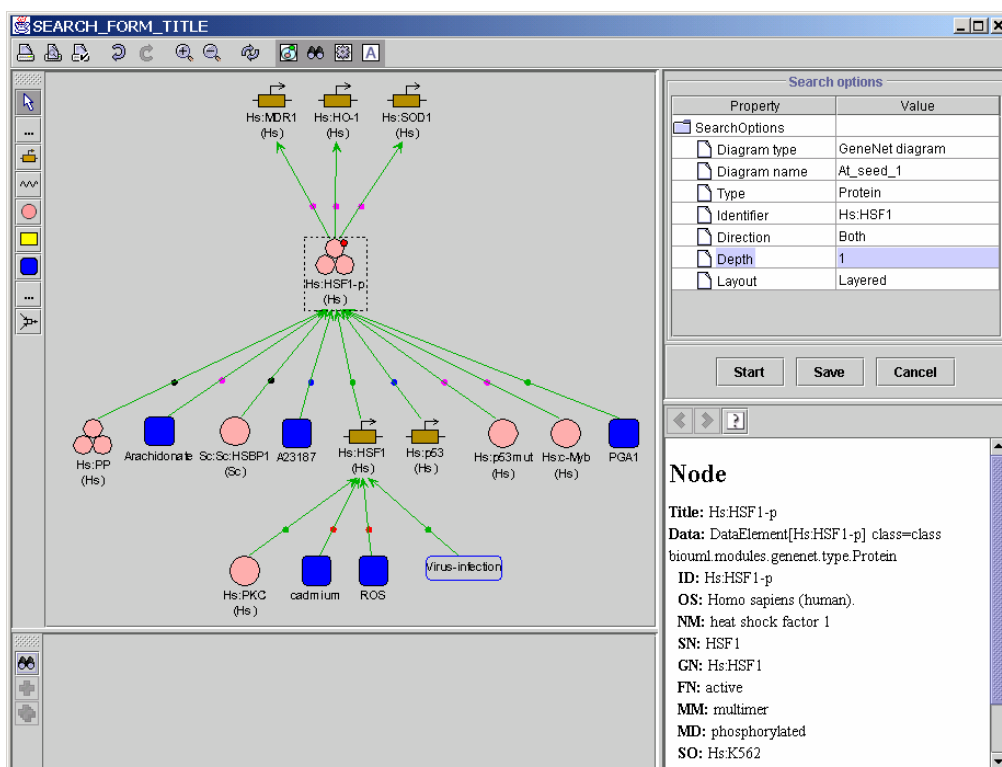


Figure 4. Example of search using GeneNet module query engine and results visualization by BioUML workbench.

BioUML workbench provides standard module for modeling different biological pathways including metabolic pathways, signal transduction pathways and gene networks. The module defines most common biological data types (gene, protein, RNA, substance, reaction, etc.), their mapping into simple text database, default query engine, and three diagram types for description of biological pathways on three semantic levels:

1. Semantic network (ontology) – describes semantic relationships between system components, system states, and related problem domain concepts.
2. Pathway structure – description of biological pathway structure.
3. Pathway simulation – is extension of pathway structure diagram where different mathematical elements are associated with graph elements as described above.

We have also developed modules for GeneNet (Kolpakov et al., 1998), KEGG/Pathways (Kanehisa et al., 2002) and TRANSPATH (Schacherer et al., 2001) databases, as well as modules for processing models in SBML (Hucka et al., 2003) and CellML (Physiome Sciences, 2001) formats. There are number of new databases (modules) that are being developed using BioUML technology, for example Cyclonet database.

SBML support

SBML plug-in that is part of standard distributive of BioUML workbench allows customer to use models in SBML format from BioUML workbench. The detailed plug-in description and implementation details are described at http://www.biouml.org/plugin_sbml.shtml?overview.

Special module type is used to store SBML models. Unlike other BioUML modules SBML module type has not 'Data' section. This is due to that all information about species and reactions is defined inside SBML model files.

All models from SBML Model Repository were packed into one BioUML module that can be downloaded from BioUML web site (<http://www.biouml.org/module.shtml?sbml>) and imported into BioUML workbench. Special diagram type is used to show SBML models as diagrams. This diagram type is very similar with BioUML standard pathway simulation diagram and we use the same graphic notation.

This diagram type can be used for model simulation using MATLAB plug-in. The plug-in will generate MATLAB files and starts MATLAB engine for model simulation and results visualization (see plugin user guide for more details).

Alternatively customer can use Java simulation engine that is part of BioUML workbench. Corresponding plug-in will generate Java code, compile it and simulates the model behavior using ODE solvers that were ported from odeToJava library.

While SBML model do not provides information about diagram layout some very simple layout algorithm is used for initial species and reactions layout. The user can customize the layout and store it. For this purpose plug-in defines its own extensions to store diagram layout data using <annotations> element. The detailed description of used approach is available at http://www.biouml.org/plugin_sbml.shtml?extension.

Table 2 summarizes SBML formats supported by BioUML workbench version 0.7.4.

Table 2. SBML formats supported by BioUML workbench version 0.7.4.

SBML format	Reading	Writing
Level 1 version 1	+	+
Level 1 version 2	+	under development
Level 2 version 1	+	under development

Discussion

Formal description and modeling of biological systems require coordinated efforts of different group of researchers:

- 1) programmers - they should provide computer tools for this task;
- 2) problem domain experts - they should specify what and how should be described;
- 3) experimenters and annotators - they should describe corresponding data following to these rules;
- 4) mathematicians - they should provide methods for models analysis and simulations.

I believe that one of the methodological achievements of BioUML workbench is that it separates these tasks so they can be effectively solved by corresponding group of researchers and provides simple contract how these groups (and corresponding software parts) should communicate one with other. Meta-model is the base of this contract for all parties.

Plug-in based architecture provides BioUML workbench extensibility and possibility of seamless integration of other tools for systems biology. Freely available BioUML workbench source code allows customers to develop their own plug-ins and database modules to extend BioUML workbench for their needs. For this purpose there is special BioUML development kit that includes all source code and all needed third party libraries (<http://www.biouml.org>).

Several XML dialects like SBML and CellML 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. 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

Acknowledgements

The work was supported by the grants INTAS Nr. 03-51-5218 and RFBR Nr. 04-04-49826-a. The author is grateful to Alexander Kel and Sergey Zhatchenko for useful discussions and Puzanov Mikhail, Shaidukov Artem, Koshukov Alexander, Hudyakov Vasily for technical support.

References

1. Bradford R. (2003) A Man, A Worm, and A Nobel. *Salk Signals*, 5(2):12-17
2. Hucka M., Finney A., Sauro H. M., Bolouri H., et al., (2003) The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models. *Bioinformatics*, 19(4):524-531.
3. Hucka M., Finney A., Sauro H.M., Bolouri H., Doyle J., and Kitano H. (2002). The ERATO Systems Biology Workbench: Enabling Interaction and Exchange between Software Tools for Computational Biology. *Proceedings of the Pacific Symposium on Biocomputing 2002*.
4. IBM (2003). Eclipse platform. <http://www.eclipse.org>
5. Kanehisa M., Goto S., Kawashima S., and Nakaya, A. (2002) The KEGG databases at GenomeNet. *Nucleic Acids Res.*, **30**, 42-46.
6. Kolpakov F.A., Ananko E.A., Kolesov G.B. and Kolchanov N.A. (1998) GeneNet: a database for gene networks and its automated visualization. *Bioinformatics*, **14(6)**, 529-537.
7. Physiome Sciences, I. (2001). CellML™ home page. <http://www.cellml.org/>.
8. Schacherer F., Choi C., Gotze U., Krull M., Pistor S., Wingender E. (2001) The TRANSPATH signal transduction database: a knowledge base on signal transduction networks. *Bioinformatics*, **17(11)**, 1053-1057.