

## Journal Information

Journal ID (publisher-id): vb  
Title: Virtual Biology  
ISSN (electronic): 2306-8140

## Article Information

Publication date (electronic): 15 March 2013  
Electronic Location Identifier: e11  
DOI: 10.12704/vb/e11

---

# Modular Modeling of Biological Systems

Ilya Kiselev<sup>1, 2</sup>, Fedor Kolpakov<sup>1, 2</sup>

[1] Institute of Systems Biology, Ltd, Novosibirsk, Russia

[2] Design Technological Institute of Digital Techniques SB RAS, Novosibirsk, Russia

---

## Abstract

*Motivation:* Modeling of complex biological systems such as cells, organs or even whole organisms is not a trivial task because of their intricate structure. Although modern computers allow simulation for quite complex models, such models are difficult to support and work with. Modular approach facilitates the creation of complex models by representing them as combinations of submodels. On the other hand, there are a large number of models describing particular subsystems created by different authors using different formalisms and scales. These models may be reused as “bricks” in the creation of comprehensive overall models.

*Results:* We have developed a modular approach to the modeling of complex biological systems. It includes a formal definition of graphical notation for modular models, an algorithm for the transformation of a modular model into a non-modular model which is appropriate for simulation using standard methods for solving ordinary differential equations (ODE), and an algorithm for simulating modular model based on the agent-model principles. The approach was implemented in software plug-in for BioUML platform.

*Availability:* The developed software and the source code are freely available as a part of BioUML in both standalone and web versions at [www.biouml.org](http://www.biouml.org).

## Keywords

Modularity, Visual modeling, Flattening

---

## Introduction

The modular approach involves system description as a set of interconnected subsystems. A model of a system is represented as a combination of subsystem models (modules). A direct benefit is that modules can be created, validated, and improved independently by different authors. Modules may utilize different mathematical formalisms, time and spatial scales, and levels of detail. Modularity, by its nature, facilitates reusability; moreover, it separates modules' interfaces and inner realization. Therefore modules may be used as replaceable parts: they can be easily modified and improved independently from other modules. Another advantage is the more explicit structure of the complex models, which facilitates their understanding and further support.

The modular approach is widely used in modern engineering, and in the past decade, its importance in modeling biological systems has become evident as well. One reason is that this approach well matches the structure of biological systems on multiple levels: from cells [Hartwell et al, 1999] to organs and whole organisms [Cooling et al, 2010]. Snoep et al. [Snoep et al, 2006] describe their vision of the construction of a comprehensive model describing a complete cellular system at the reaction level (i.e. silicon cell). Such a model should be constructed on the basis of many modules developed by community and stored in one repository. An example of a manually created model by combining three other models is provided.

Extensive research has also been devoted to the creation of a comprehensive global model of the human organism (virtual physiological human). It is supported by International Union of Physiological Sciences Physiome Project [Hunter et al, 2002] and EuroPhysiome initiative [Fenner et al, 2008] coordinating the efforts of many research groups around the world. One of its major challenges is models integration and coupling [Fenner et al, 2008].

Such ambitious goals (creation of virtual cell and virtual human models) require widely accepted standards for model description, which is crucial for model exchange and reuse and the development of approaches to composing models and specialized tools for creation, simulation, and analysis of modular models. As a model becomes more and more complicated, it would also greatly benefit from convenient visual representation and editing.

There are two general types of modular approach [Hernandez et al, 2009], [Vangheluwe, 2000]:

- **Formalism-transformation.** A modular model is transformed into a flat model, which can be simulated using standard methods (ODE, stochastic, etc.). This process needs a formal definition of the transformation of different formalisms into each other, which is not a trivial task and may place strict limitations on modules. The

degenerate case is when all modules use the same formalisms and are even described in the same formal language (for example SBML).

- **Co-simulation.** This approach implies that each module should be simulated separately using its own simulation engine. This process should be controlled by some meta-engine which should handle interactions between modules during the simulation.

Both approaches have advantages and disadvantages. In the first case, we have strict mathematical foundations but also strict limitations on the module formalisms. In the latter case, we have great flexibility in the inner implementation of models and no need for formalisms transformation algorithms. However, the co-simulation approach lacks mathematical foundations. Even the questions about solution existence and uniqueness are open.

The key question for both approaches is how interactions between modules are described in the frames of the modular model. Usually, there are some additional elements in the modular model which are used to integrate modules.

The way how the model addresses the modules is another important question. It may directly address the module elements (equations, chemical reactions, variables, etc) or the module elements can be accessed only through predefined ports. Module ports define the interface through which other modules may be connected with it. The first approach is more flexible and the latter approach provides more controllable and well-defined modular models.

If models of subsystems are initially designed to be modules, they may predefine their interfaces. For the modeler to create a modular model, it takes only to pick up the needed modules and establish appropriate connections between their interfaces.

Approaches to handle composite models are discussed in [Randhawa et al, 2009], [Randhawa, 2010]:

- **Fusion** is simply manual creation of a larger model using submodels elements.
- **Composition** means that a complex model includes submodels in an implicit manner. Connections are established between their inner elements.
- **Aggregation** differs from the composition in the way which submodel elements are available for the modular model creator. Submodels predefine which elements are accessible from other submodels. This implies that submodels are initially created to be parts of larger models.
- **Model flattening** is the process of automatic generation of a fused model on the basis of the complex model created using a composition or aggregation approach. This procedure is a degenerate variant of the formalism-transformation approach described previously.

Most widely used standards for model description are markup languages: SBML (Systems Biology Markup Language) [Hucka et al, 2003] and CellML [Garny et al, 2008]. Both of them are evolving toward the modularity concept. Discussion about including instruments for describing modular models continued in the SBML community since 2000, and a number of proposals were made by different authors. Finally, in 2012 a specification for a composite hierarchic extension to SBML and a set of tests were released. The extension allows SBML models refer to another SBML models. This is provided by an **instance** object which contains a reference to a submodel. Submodels can be defined either in the same file as the **submodel** object or somewhere else (they can be referred by URI). “Glue” elements which bind modules are **replacements** which work in two directions. An element from a submodel can be replaced by an element of the modular model or vice-versa. Two elements from different modules can be connected by replacing both of them with the same element. Elements may be addressed either directly or through predefined **ports**, which correspond to the composite approach.

CellML was initially designed to directly support modularity. Models are composed of interconnected components. However, only version 1.1 which was finalized in 2006 [Garny et al, 2008] supports import of external models and their reuse as parts of another models. Unlike SBML, in CellML modules may be connected only by replacing variable of one module with the variable of another module. Modules should predefine **ports** for variables which may be of two types – **in** and **out**. **Connection** between two modules defines mappings between “out” variable of one module and “in” variable of another.

There are a lot of modeling and simulation tools for systems biology. For example, the most complete list of tools supporting SBML is available at [www.sbml.org](http://www.sbml.org) and includes more than 250 tools. However, not many of them support modular design. The reason may be that the SBML composite package was released only recently.

We will give a brief overview of tools for systems biology which support the modularity concept. Summary tables are presented at the end of the current paper in the conclusion section.

Probably, the first modeling tool aiming to support modular design was ProMoT [Ginkel et al, 2003]. It presents an object-oriented language able to describe modular models with DAE formalism along with a visual representation. Unfortunately, at the moment, it supports only SBML l2v1 without events, which is quite an old version of SBML.

JigCell [Vass et al, 2004] is a set of tools supporting SBML l2v1 and allowing SBML models aggregating [Randhawa et al, 2009] into modular models. Modules are edited using tables of elements. It also provides simulation and analysis tools.

iBioSim [Myers et al, 2009] allows visual editing of SBML models and modular models; in addition it supports the SBML composite hierarchic package.

There are also several pure simulation tools supporting SBML composite models such as RoadRunner (<http://roadrunner.sf.net>) and SB Simulation Core Library (<http://simulation-core.sf.net>).

TinkerCell [Chandran et al, 2009] like iBioSim supports both visual and modular concepts. It is plugin-based and

provides simulation and analysis using COPASI [Hoops et al, 2006]. However, to our knowledge, it supports only export to SBML format.

M2SL [Hernandez et al, 2009] bridges the gap in supporting multiformalism modeling utilizing the co-simulation approach, but, at the moment, it lacks standards support and visual representation of models.

As regards CellML, the number of tools supporting it is significantly less than that for SBML (list available at [www.cellml.org](http://www.cellml.org)). Moreover, although the language is modular by its nature, many tools supporting it do not support modularity and only provide import into flat models (JSim, <http://www.physiome.org/jsim/>, VirtualCell, <http://vcell.org/>). Tools that support modular design (OpenCell, <http://www.cellml.org/tools/opencell/>, COR, <http://cor.physiol.ox.ac.uk/>) lacks visual representation of models. The only tools presented at the official site utilizing a visual approach are the GUICellML (see [www.cellml.org](http://www.cellml.org)), which is quite simple and allows modules editing only as CellML text and CellModelViewer [Wimalaratne et al, 2009] and does not allow editing or creation of models.

SBML and CellML are not human readable, especially when dealing with complex modular models, and needs tools for interpretation into a user friendly interface. There are several research efforts aimed at describing modular models of biological systems using human readable languages, e.g., Antimony [Myers et al, 2009] (supports SBML composite and CellML), PySB [Lopez et al, 2013] (uses Python programming language to describe models, submodels are treated as callable subroutines) and little-B [Mallavarapu et al, 2009] (based on Lisp). However, we believe that the creation and support of complicated modular models strongly requires a visual representation and editing (human readable language, though, may be a good add-on to it). A common standard for graphical notation in systems biology is SBGN [Novère et al, 2009]. However, it does not provide notation for modular models. Therefore, our aim is not only to implement a modular approach, but also to develop a graphical notation and a convenient tool supporting both common standards (SBML, CellML, SBGN) and multiple formalisms.

## Modular model definition

In our approach (which is more similar to CellML then to SBML), modules are black boxes which receive and send signals. They may be combined using connections – special elements representing signal transfer from one module to another. Here a signal is a value of a specified variable of the module. Hence, the overall model is described in terms of signal transmission between the functional modules. This approach is well compatible with co-simulation: modules may be simulated independently, and connections determine the exchange of variable values between them. However, it may also be used for model transformation into a flat model. We have implemented an algorithm of such transformation for the case where all modules have certain limitations on the mathematical formalism: differential algebraic equations (DAE) with discrete events and chemical reactions which can be interpreted as ODEs or stochastic processes. It provides flexibility in modular model simulation: if all modules satisfy certain conditions, then the modular model will be simulated using a transformation approach. If some modules do not satisfy these conditions, a generalized co-simulation approach is applied. Visual representation of models in both cases can be the same, and only the inner implementation of the modules is changed.

Note that in the present paper, by DAE we will mean first order explicit differential equations supplemented with pure algebraic equations which usually model conservation laws. Equations of this type are most common in systems biology and can be described in SBML and CellML:

$$\begin{cases} \frac{dx}{dt} = f(x, y, t) \\ 0 = g(x, y, t) \end{cases}$$

Essentially, a **module** is a black box with a defined interface through which it can be connected with other modules fig. 1. Formally, we define it as a quadruple:

$$M = (X, I, O, C),$$

where:

$X = \{x_1, \dots, x_n\}$  – set of module **variables**

$I \subset X$  – set of module **input** variables, whose values should be transferred from the outside of the module.

$O \subset X$  – set of module **output** variables, whose values are calculated inside the module and may be transferred to other modules.

$C \subset X$  – set of module **contact** variables, whose values can be modified both inside the module and outside.

The sets of output input and contact variables define a module **interface** through which the module can be connected with other modules. Graphically they are denoted using **port** elements. It should be noted that, in general, some variables may be both input and output or neither:

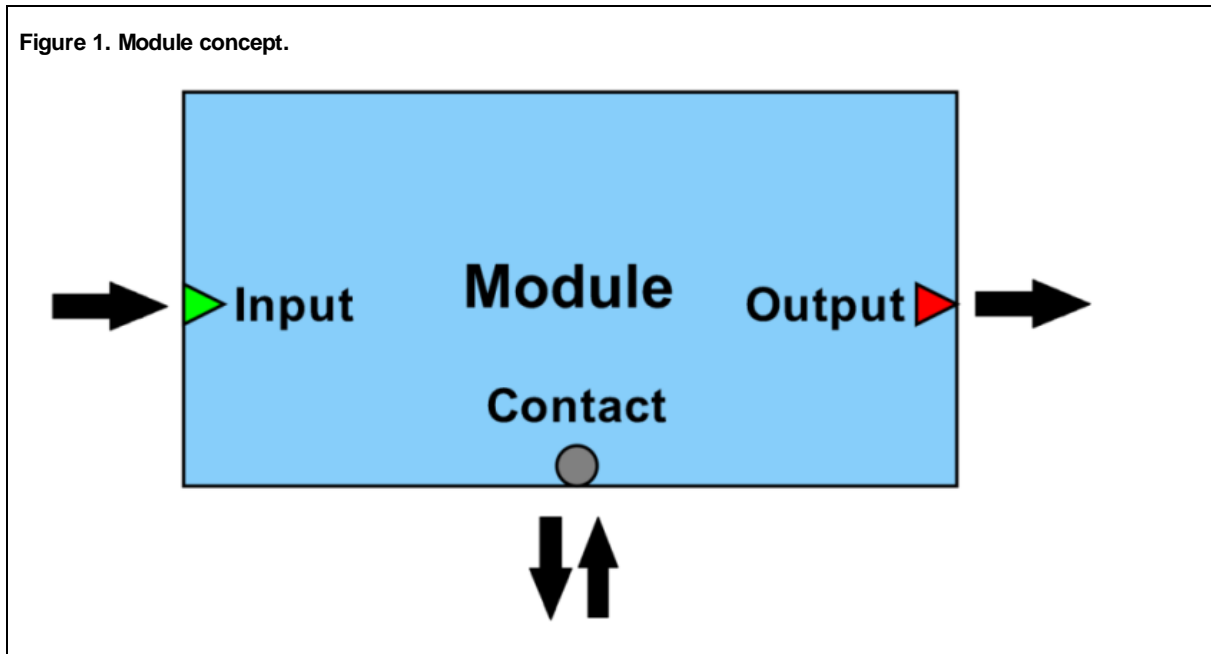
$$I \cap O \cap C \neq \emptyset,$$

$$I \cup O \cup C \neq X.$$

A module can be represented as a functional block which calculates output on the base of input:

$$M(I, C) = (O, C)$$

Figure 1. Module concept.



Suppose we have two modules:

$$M_1 = (X_1, I_1, O_1, C_1), \quad M_2 = (X_2, I_2, O_2, C_2).$$

**Connection** between two models is defined as a correspondence between their variables. If two model variables are interconnected they should be considered as one variable in the frames of modular model. Properties of this new variable are determined by the connection type and interconnected variables.

**Directed connection** means that a new variable value is calculated in one module and then used in another. In that case, one module affects another but not vice versa. Formally, it is a triple:

$$(x, y, p), \quad x \in I_1, y \in O_2, \quad p: \mathbb{R} \rightarrow \mathbb{R}$$

Where  $p$  is an arbitrary, not mandatory function. It can be used, for example, to adjust variable units.

Dynamics of a new variable will be fully defined by the dynamics of  $y$  (and it will inherit its name and properties). All references to  $x$  will be replaced by the expression  $p(y)$ .

Notion:  $y \xrightarrow{p} x$  ( $x \xleftarrow{p} y$ ) or in short:  $y \rightarrow x$  ( $x \leftarrow y$ ).

**Undirected connection** means that a variable value may be changed by both modules (the modules affect each other). Such variables are called shared. Formally, undirected connection is a pair:

$$(x, y), \quad x \in C_1, y \in C_2.$$

Dynamics of the new variable is a composition of two connected variables dynamics. Undirected connection may define properties of the new variable such as name, initial value, etc directly, or by selecting one of the interconnected variables  $x$  or  $y$ . If these properties are not set then one of interconnected variables will be selected randomly.

Notation:  $x \longleftrightarrow y$ . Further in the text, a connection chain  $x_1 \leftarrow x_2, x_2 \leftarrow x_3, \dots, x_{n-1} \leftarrow x_n$ , will be denoted as  $x_1 \leftarrow x_2 \leftarrow \dots \leftarrow x_{n-1} \leftarrow x_n$  and similarly for undirected connections.

**Semantic control.** The following restrictions should be considered to create a semantically correct model:

1. Multiple ingoing directed connections for one variable are restricted (source of signal is ambiguous):  
 $x \rightarrow y \leftarrow z$ .
2. Cyclic directed connections are restricted (source of signal is uncertain):  
 $x \leftarrow y \leftarrow \dots \leftarrow z \leftarrow x$
3. Undirected and ingoing directed connection for the same variable (conflicting signals):  
 $x \rightarrow y \leftrightarrow z$ .

We can now define a **modular model**:

$$MM = (M, E, DC, UC)$$

where:

$M = \{M_1, \dots, M_N\}$  – set of modules.

DC – set of directed connections between modules,

UC – set of undirected connections between modules.

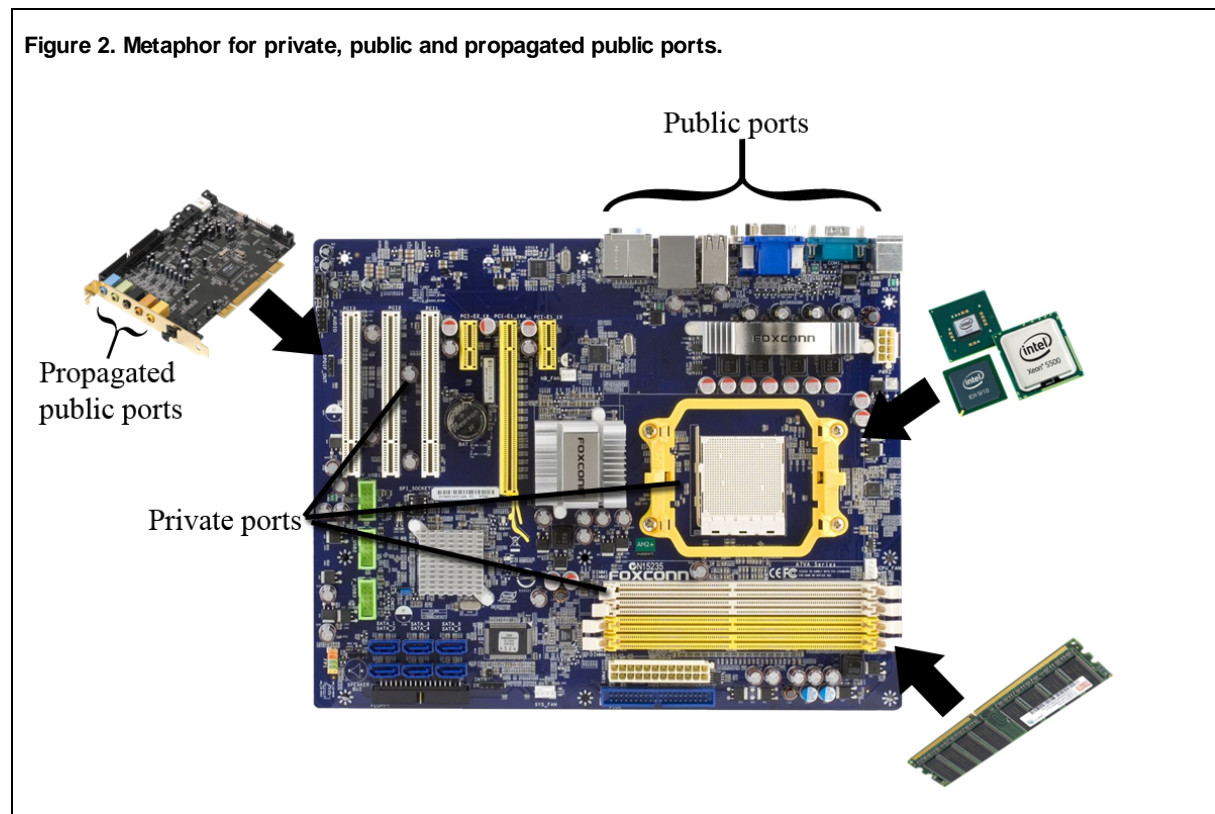
*E* – **external environment** for modules which defines conditions for modules interaction.

Environment defines two types of interface (similar to CellML interface types): **private** interface is used to alter encapsulated modules behavior. **Public** interface is needed when modular model is encapsulated as a module into another modular model forming hierarchical structure. Metaphor for private and public interfaces and using modular as a module is presented on fig. 2.

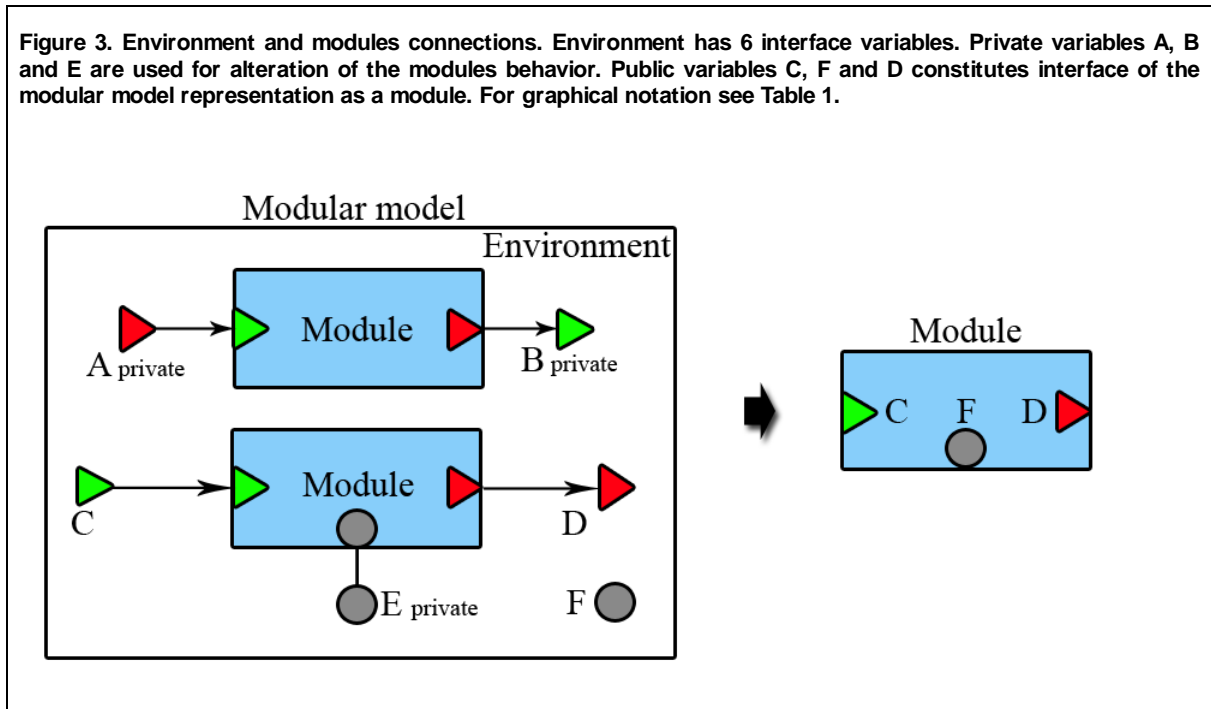
Both interfaces define three sets of variables: input, output and contact. Private interface variables may be used to establish connections with encapsulated modules. In that sense, environment may be simply considered as one of the interacting modules.

Public interface is used when modular model is encapsulated into parent modular model as a module. These variables then may be accessed by other modules. There are two ways to create public interface variable. It may be variable of external environment which value is calculated according to established rules possibly under influence of encapsulated modules. Another way is to extend interface variable of the encapsulated module to the interface of modular model by establishing a connection between interface variable of a module and interface variable (of the same type) of environment. It means that signal from encapsulated module should be propagated to the outside of the modular model (for output variable) or signal incoming to the modular model will be propagated directly to one or more of its modules (for input variables). Possible connections between environment and its modules are depicted on fig. 3.

**Figure 2. Metaphor for private, public and propagated public ports.**



**Figure 3. Environment and modules connections.** Environment has 6 interface variables. Private variables A, B and E are used for alteration of the modules behavior. Public variables C, F and D constitutes interface of the modular model representation as a module. For graphical notation see Table 1.



## Visual modular modeling

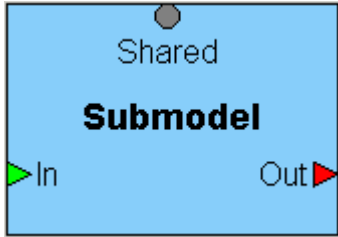
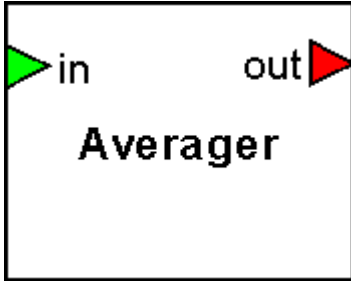
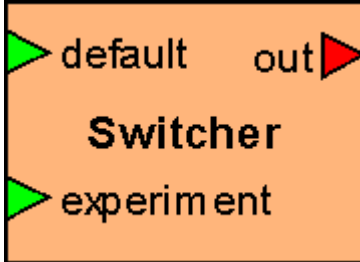
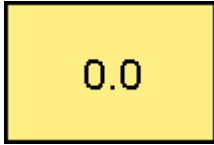



As the base for implementation of the modular approach, we have used the BioUML platform ([www.biouml.org](http://www.biouml.org)) written in Java. Important features of BioUML relevant to the modular model approach are:

- Visual modeling support.
- Supports of different mathematical formalisms: ODE/DAE with discrete events, stochastic, PDE.
- SBML, CellML and SBGN support.
- Tools for model analysis (steady state, sensitivity, flux balance, etc.) and parameter estimation.
- Test suite facility for step by step validation of models during development.
- Modular design – BioUML can be easily extended by adding software plugins.

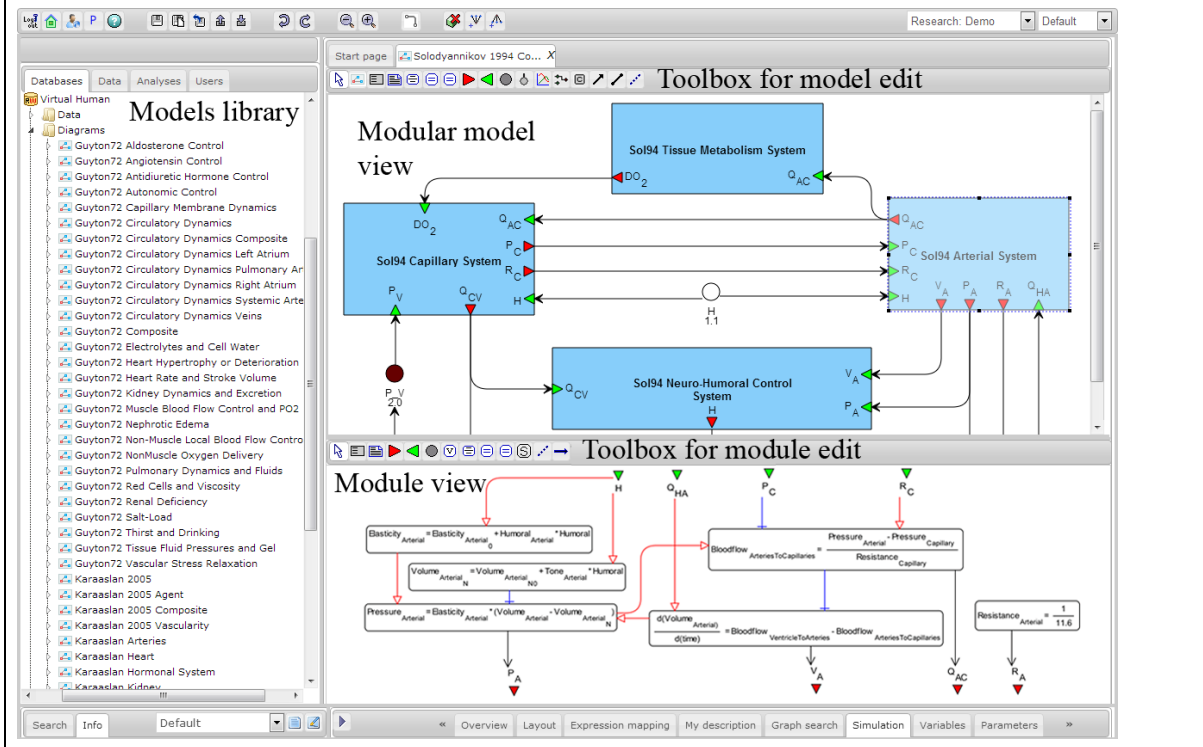
BioUML describes a model as a graph whose nodes and edges denote model elements and interactions between them. Each diagram element may be associated with a database entry and simultaneously with an abstract mathematical entity. Diagrams may be visually created and edited by the user. Visual representation is defined by a formal graphical notation which depends on the diagram type. For example, in importing a SBML model, the user may choose to represent it using the SBGN or BioUML notation. Specific notation data along with graph layout information will be saved as an annotation element. Graph is used to generate Java code for numerical simulations. The generated code depends only on the mathematical properties of the model; therefore, the mathematically equivalent SBML (with or without SBGN) and BioUML models will produce the same code.

In order to implement the modular approach in the BioUML platform, we have developed the graphical notation presented in Table 1. The user interface for modular modeling is presented in Fig. 4. It includes graphical notation for different modules types, interface ports and connections.

Table 1. Graphical notation for modular models

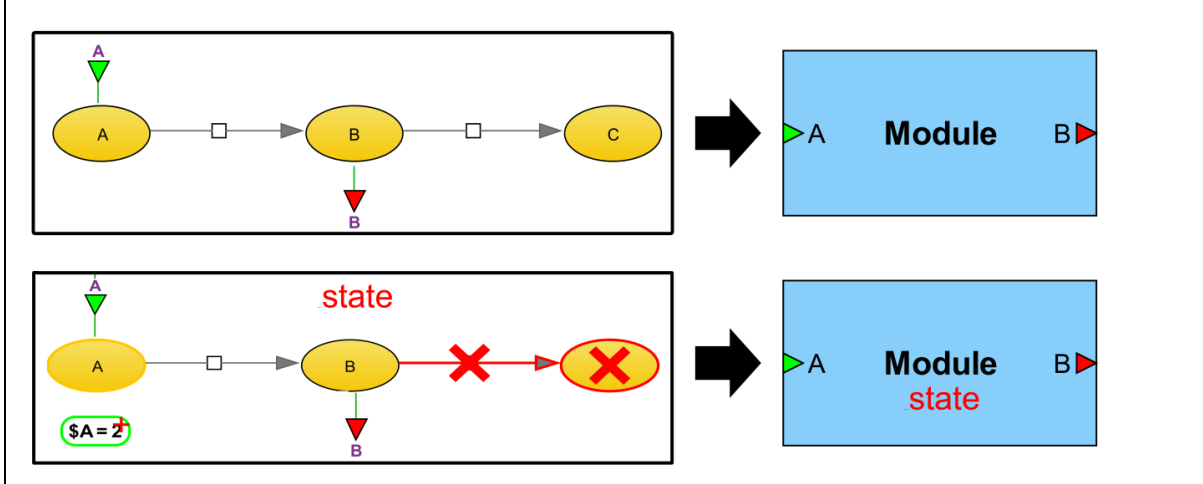
Modules		
Graphical notation	Description	X, I, O, C
	<p><b>Submodel.</b> Module containing a mathematical model. Possible model types:                      Modular model,                      SBML model,                      ODE model with discrete events created in BioUML,                      Model of biological pathways created in BioUML,                      PDE model of the human arterial tree.</p>	<p>Defined by submodel properties.</p>
	<p><b>Averager.</b> Module calculating the moving average of the input signal according to formula:  <math display="block">out(t_k) = \frac{1}{n} \sum_{i=k-n}^k in(t_i)</math>                     n – module parameter – number of data points.</p>	<p>X = {in, out},                      I = {in},                      O = {out},                      C = ∅.</p>
	<p><b>Switcher.</b> Module outputting one of the two input signals:  <math display="block">out = \begin{cases} in1, &amp; h(t) \\ in2, &amp; otherwise \end{cases}</math> <math>h: R \rightarrow \{true, false\}</math></p>	<p>X = {in1, in2, out},                      I = {in1, in2},                      O = {out},                      C = ∅.</p>
	<p><b>Constant.</b> Module containing one variable with a constant value.</p>	<p>X = {x},                      I = ∅,                      O = X,                      C = ∅.</p>
	<p><b>Plot.</b> Module dynamically drawing input signal value on chart.</p>	<p>X is defined by the user                      I = X,                      O = ∅,                      C = ∅.</p>
	<p><b>Bus.</b> Module with one variable which can be input, output or contact. Several buses can contain the same variable, which decreases the edge intersections.</p>	<p>X = {x},                      I = X,                      O = X,                      C = X.</p>
Other elements		
<p>▶ input port</p> <p>output port ▶</p> <p>contact port ●</p>	<p><b>Port</b> (input, output and contact). Refers to interface variables of the module. Each port refers to one variable and only one port for each variable may be declared in the module.</p>	
	<p><b>Connection</b> (directed, undirected). Directed connection can be established between the output port and the input port. Undirected connection between two contact ports.</p>	

**Figure 4. BioUML web interface for the creation of modular models on the example of the short-term CVS model by professor Solodyannikov [Solodyannikov, 1994].** The interface consists of a database tree on the left, which provides a library for reusable models, a Panel for editing the selected modular model at the center, and a panel for editing the selected module below. It also contains various tabs (at the bottom) for model simulation, layout, overview, etc.



The modular model definition described above permits modules integration only by connecting their variables. However, the user probably would like to tune modules before integrating them into a modular model. This process is facilitated by the **state** concept in BioUML (fig. 5). A state comprises a list of changes which are applied to the model. This can be almost any change that can be done to the model: element deletion, addition, editing. The model can be even fully rewritten, though this is not the best practice and usually is not expected. After adding the module into the modular model, the user may specify which state will be used as default in the frames of the modular model or create a new state for module.

**Figure 5. State concept. Different inner realizations of the same module. Model elements can be added (green border), removed (red border), and edited (yellow border). In the current example species A is set to boundary condition, new equation is added which sets A to new constant value, and species C and reaction  $B \rightarrow C$  is deleted.**



## Model flattening algorithm

In this section, we will describe an algorithm which transforms the modular diagram defined in previous sections into a "flat" one. The input of the algorithm is a modular diagram with submodels that can be:

1. SBML model (in SBGN or BioUML notation)
2. ODE with discrete events model created in BioUML
3. Model describing biological pathways created in BioUML



4. Modular model whose modules also satisfy conditions 1-4.

Technical modules – Switchers, Constants, Plots and buses – are also allowed. Only Averager module is restricted as it cannot be efficiently transformed into an algebraic or ODE equation. Another natural restriction is that modular model cannot contain itself as a module.

The result of the algorithm work is a diagram described in the same formalism. It can be interpreted either as a set of DAE with events simulated by standard methods (Euler, Dormand-Prince, ported to Java CVODE [Hindmarsh et al, 2005]) or alternatively as a stochastic model simulated by stochastic algorithms [Gillespie, 2007].

The algorithm can be applied in two cases:

1. Automatically, when the user decides to simulate modular model. The transformed model is passed to the solver, which generates the simulation result. The flat model is intermediate and is not stored in the repository or shown to the user.
2. If the user wants to convert a modular model into flat one and save it for further work. In this case the result is the ODE model which is stored in the BioUML repository.

The algorithm goes as follows:

**Step 1.** If some modules are modular models themselves, they are transformed into flat ODE modules, – the algorithm is applied recursively.

**Step 2.** Each variable of each module is given a new name, unique in the frames of the modular model. The set of unique identifiers will be denoted as  $N_X$ . Created mapping:  $N: X_{MM} \rightarrow N_X$ .

**Step 3.** Processing connections which are established in the modular model, **replacement rules** are generated. Each variable  $x \in X_{MM}$  is associated with some mathematical expression  $S(x)$ . Set of replacements will be denoted as  $S$ .

1. If there are no ingoing directed or undirected connection for  $x$  ( $\nexists y \in X_{MM}: x \leftarrow y \vee x \leftrightarrow y$ ), then we set  $S(x) = N(x)$ . Thus, if there are no connections in the model, then  $S$  is equal to  $N$  and each variable will be replaced by its new name only. Moreover, if variable names in different modules do not coincide, then all names will be preserved and no replacements will be done at all.
2.  $\exists y \in X_{MM}: x \leftrightarrow y$ . Let us construct a chain (possibly cyclic) of undirected connection:

$$\dots \leftrightarrow x \leftrightarrow y \leftrightarrow \dots \leftrightarrow z$$

In this chain we choose one variable which will be called **main**, we will denote it as  $Main(x)$ . Then for each element of chain we set:

$$S(x) = S(Main(x)) = N(Main(x))$$

Choice of main variable:

- a. If connection chain includes bus then bus variable is taken as main. Only one bus may be included into connection chain.
  - b. Otherwise if connections define their own variables, one of them will be chosen. Modeler may define which connection should have priority. If priorities are not set then choice between connection variables is random.
  - c. At last, if connections do not define new variables then variable is chosen between connected variables.
3.  $\exists y \in X_{MM}: x \xrightarrow{p} y$ . We set  $S(x) = p(S(y))$ . Let us show that this definition is correct: if  $y$  has no other connections or has undirected connection, then  $S(y) = N(Main(y))$  – is already defined. If  $y$  has ingoing directed connection  $y \xleftarrow{q} z$ , then applying the same rule to  $y$ :

$$S(x) = p(S(y)) = p(q(S(z))).$$

This process is guaranteed to converge because there is a limited number of variables in the model and the semantic rules restrict the cyclic and multiple directed connections. Finally we obtain the mapping:

$$S(x) = p(q(\dots r(N(Main(v)))) = p(q(\dots r(N(u)))).$$

Here  $u = Main(v)$  and  $u$  has no ingoing directed connections.

Thus, we may define a mapping which associates each variable from  $X_{MM}$  with an expression:

$$S(x) = \begin{cases} S(Main(x)), & \exists y \in X_{MM}: x \leftrightarrow y \\ p(S(y)), & \exists y \in X_{MM}: x \xrightarrow{p} y \\ N(x), & \text{otherwise} \end{cases}$$

**Step 4.** The procedure is applied to the modular model. Each element is copied to the plain model. If the element is a compartment, then this procedure is applied to all its inner elements recursively. This process takes into account the replacement rules generated on the previous step. Namely, all references to variable  $x$  are replaced by references to expression  $S(x)$ .

Additionally, a number of special rules are applied during this process:

**Equations.** Assignments (including the initial assignments and event assignments) and differential equations which define the variables  $x$  such that  $S(x) \neq N(x)$  are ignored. Those variables are either replaced via directed connection and therefore their dynamic is defined by another variable, or interconnected via undirected connections and were not considered as **main**.

**Differential equations.** If several variables are defined by differential equations and there is an undirected connection between them,

$$x_1 \leftrightarrow x_2 \leftrightarrow \dots \leftrightarrow x_m, \quad S(x_i) = x$$

$$\frac{dx_i}{dt} = f_i$$

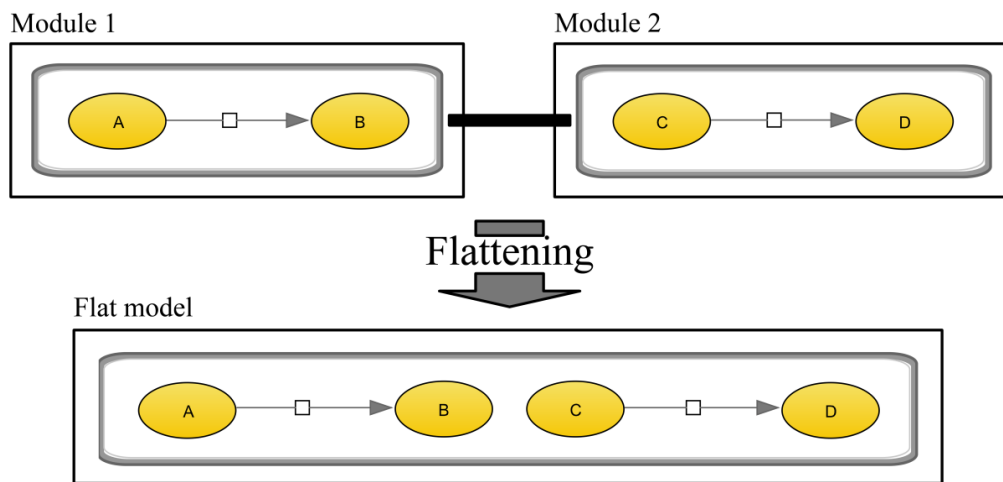
then a new differential equation is generated:

$$\frac{dS(x)}{dt} = \sum_{i=1}^m f_i$$

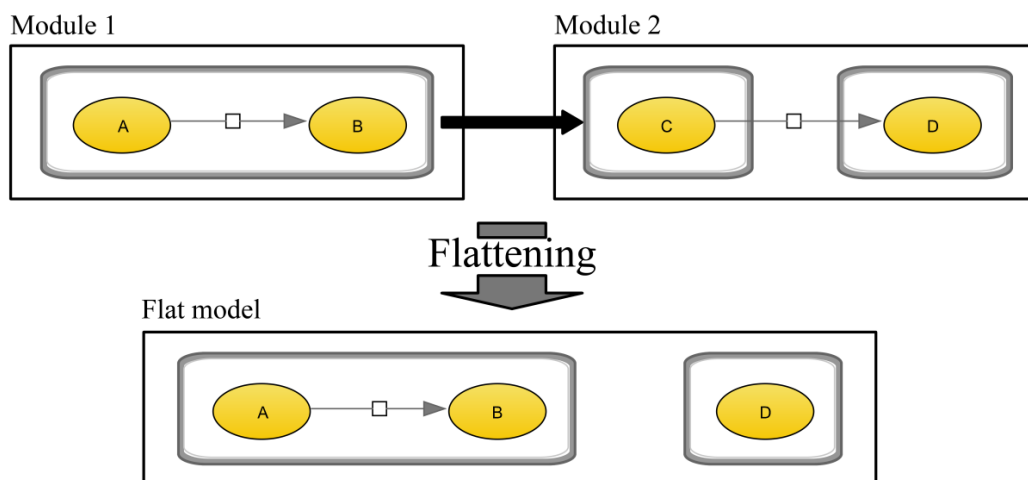
Thereby, the chain of connected variables is transformed into one variable, whose dynamic is the sum of the dynamics of the chain variables. It should be noted that this agrees with stitching reactions for interconnected species (see fig. 8).

**Compartments.** If two compartments are connected by an undirected connection, they merge into one compartment containing elements from both of the initial compartments (see fig. 6). If two compartments are connected by a directed connection, then one of them is replaced by the other, all its elements are removed from the model (see fig. 7).

**Figure 6. Compartment merge. Two compartments merge into one. New compartment contains all content from the initial compartments.**



**Figure 7. Compartment replacement. Compartment containing species C is completely replaced with all its content. If any species inside it participate in reactions, those reactions are also deleted.**



**Parameters and species.** From a mathematical point of view, species and parameters are almost equal. They both are associated with model variables whose dynamics may be defined by assignments, events, differential and algebraic equations and both may be subject of module interface ports so we may establish connections between the parameter and species. However, they are differently represented in the diagram; therefore we should process such connections in a

special way. Suppose, we have parameter  $p$  in one module and species  $S$  in another. We have three possible connections between them:

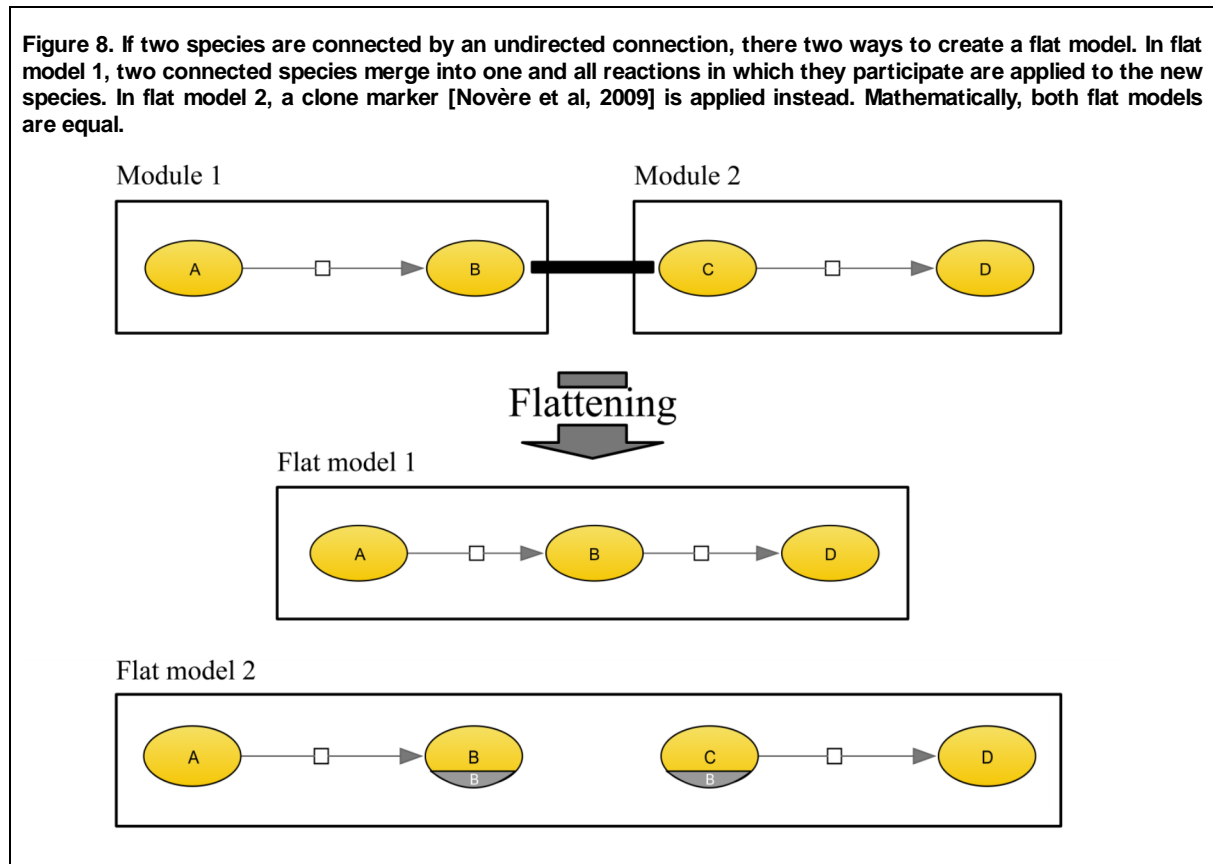
$p \leftrightarrow S$ . In this case, species  $S$  will be selected as the main species and therefore will substitute  $p$  in all equations and assignments.

$p \xrightarrow{f} S$ . Species  $S$  will not be entirely substituted by the parameter. Instead, a new equation will be generated to ensure signal transmission:

$$" S = f(p) "$$

$p \xleftarrow{f} S$ . Just as in the simple case with two connected parameters: the parameter  $p$  will be substituted by  $S$  and all equations which affect its dynamics will be eliminated (so that its dynamics will be entirely defined by the dynamics of species  $S$ ).

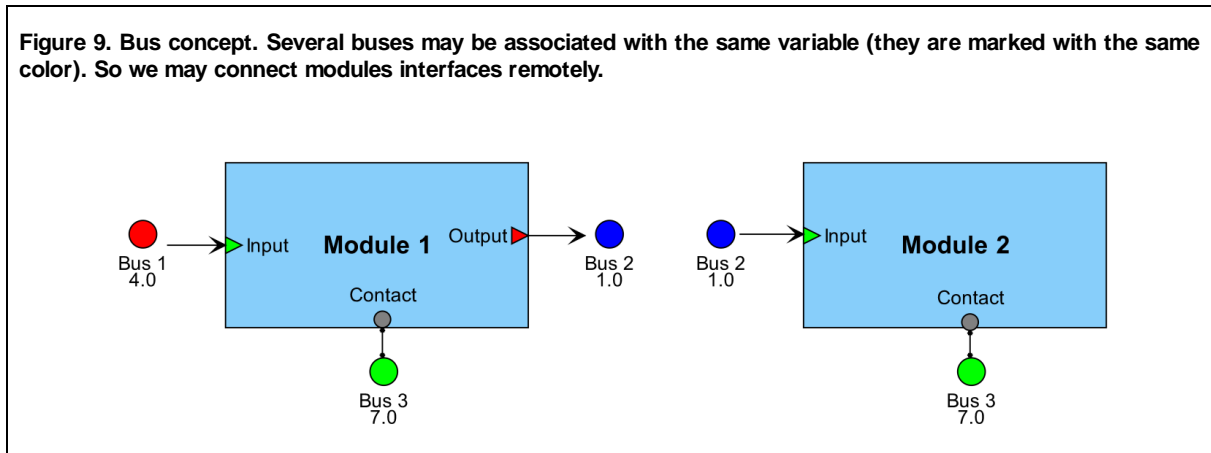
**Species graphical elements.** If two species are connected by an undirected connection, they should now be considered as one entity. A simple way is to create new species which will replace both original species whenever they are referenced in the model. In visual representation, all reactions in which the original species participate will now be applied to the new species. However, if models contain a large number of reactions, they are hard to trace. Another option is to apply a clone attribute (see [Novère et al, 2009]) indicating that two objects in the Diagram are associated with the same entity and hence with the same variable. Examples are depicted on fig. 8.



Let us consider a situation where we have two species connected by a directed connection:  $A \rightarrow B$ . The directed connection supposes that  $B$  will not be changed by its module. So all equations possibly changing its amount should be eliminated. However, we still want it to participate in the reactions. We cannot simply replace it by  $A$  as in the case of undirected connection, because then  $A$  will be affected by these reactions. Therefore, we add both species  $A$  and  $B$  into plain model add new assignment equation " $B = A$ " to provide directed connection effect. Besides, we specify boundary conditions for species  $B$  to avoid its amount changing by reactions.

**Bus.** A bus is special element in the modular diagram representing variables. It serves as private interface ports for modular model environment. Type is defined by established connections. If undirected connection is established with bus then no directed connections may be established and vice versa. It also cannot have multiple incoming directed connections. Several buses may be associated with the same, helping to avoid intersections between connection edges. Bus connected by undirected connections also allow user to set variable properties that will be used for new variable corresponding to the chain of undirected connections. Example of buses using is presented on fig. 9.

**Figure 9. Bus concept. Several buses may be associated with the same variable (they are marked with the same color). So we may connect modules interfaces remotely.**



## Results and discussion

This paper describes a modular approach implemented as a software plugin for BioUML. It is similar to CellML in the sense that it allows modules integrating using connections between variables. However we present two different types of connections with different properties and utilize state concept which allow more extensive altering of modules in the frames of modular model and makes possible to express semantic similar to SBML composite models. The approach also provides a graphical notation which represents a model in terms of signal transmission between modules. This representation is used for both model transformation and general co-simulation approaches. It separates the model visual creation process from the actual inner implementation of modules and type of simulation.

The developed plugin includes:

1. Graphical notation for the visual creation of the modular model.
2. Algorithm for generating a flat model on the basis of modular model in the case of certain limitations on the modules formalisms (DAE with events).
3. Algorithm for an agent-based approach to co-simulation in the case of arbitrary modules formalisms.

A summary table including BioUML with the developed plugin is presented in tables 2 and 3.

The plugin was already used for the development of several modular models:

- Integrated apoptosis model [Kutumova et al, 2012] comprising 13 submodels which were derived from different sources.
- Reconstruction in BioUML of the classic overall circulation model by professor Guyton [Guyton et al, 1972] comprising 18 modules according to the original model scheme.
- Model of the human cardiovascular system [Kiselev et al, 2012] incorporating modules from three different models utilizing different formalisms: a model with heart pulsating [Solodyannikov, 1994] (ODE), a model with long-term human regulation (ODE) [Karaaslan et al, 2005], and a model of blood flow across 55 largest human arteries (PDE) [Biberdorf et al, 2012].

Software is in active development. Currently, we continue our research in the following directions:

- Full support of SBML composite package. We believe that SBML composite models may be effectively described in our terms of connections and states.
- Support for CellML 1.1.
- Extending the the library of atomic modules.
- Visual models creation and editing improvements: more extensive support of multilevel hierarchical models visual representation. Automatic layout algorithms improving.

The developed plugin (as well as the source code) is available both in web and standalone BioUML versions.

**Table 2. Modeling features.**

	Visual modeling	Visual modeling modular	Simulation	Analysis	Parameter estimation
QTAntimony [Smith et al, ]	-	-	-	-	-
BioUML	+	+	+	+	+
iBioSim [Myers et al, 2009]	+	+	+		
JigCell [Vass et al, 2004]	-	+	+	+	+
M2SL [Hernandez et al, 2009]	-	-	+	+	+
OpenCell	-	-	+	-	-
ProMoT [Ginkel et al, 2003]	+	+	Using DIVA/DIANA	+	+
RoadRunner	-	-	+	+	
Simulation Core Library	-	-	+	-	-
TinkerCell [Chandran et al, 2009]	+	+	+	+	+
COR	-	-	+	-	-
Little b [Mallavarapu et al, 2009]	-	-	-	-	-

**Table 3. Formats and formalisms support.**

	SBML core	SBML composite package	CellML	Supported formalisms	Multi-formalism models
QTAntimony [Smith et al, ]	L3v1	+	+	DAE, events	-
BioUML	L3v1	-	import 1.0	DAE, events, PDE, stochastic	+
iBioSim [Myers et al, 2009]	L3v1	+	-	DAE, events, stochastic	-
JigCell [Vass et al, 2004]	L2v4	-		ODE, events, stochastic	-
M2SL [Hernandez et al, 2009]	-	-	-	Bond-Graph, DAE, events, cellular automata.	+
OpenCell	-	-	+	DAE	-
ProMoT [Ginkel et al, 2003]	L2v1, except events	-	-	DAE, logical	-
RoadRunner	L3v1	+	-	DAE, events	-
Simulation Core Library	L3v1	+	-	DAE, events	-
TinkerCell [Chandran et al, 2009]	export	-	-	DAE, stochastic, events	-
COR	-	-	+	DAE	-
Little b [Mallavarapu et al, 2009]	-	-	-	ODE	-

## References

- Bassingthwaighe J B, Qian H, Li Z, authors. **The Cardiome Project. An integrated view of cardiac metabolism and regional mechanical function.** Adv Exp Med Biol. 1999;471:541–553. [PMID:10659188]
- Biberdorf E A, Blokhin A M, Trakhinin Y L, authors; Ivanova L N, Markel A L, Blokhin A M, Mishchenko E V, editors. **Global modeling of the human arterial system.** Circulatory System and Arterial Hypertension: Experimental Investigation, Mathematical and Computer Simulation. 2012. p. 115–142. Nova Science Publishers, Inc;
- Chandran Deepak, Bergmann Frank T, Sauro Herbert M, authors. **TinkerCell: modular CAD tool for synthetic biology.** J Biol Eng. 2009;(1)3:19 ISSN: 1754-1611 DOI:10.1186/1754-1611-3-19
- Cooling M T, Rouilly V, Misirli G, Lawson J, Yu T, Hallinan J, Wipat A, authors. **Standard virtual biological parts: a repository of modular modeling components for synthetic biology.** Bioinformatics. 16–2;2010;(7)26:925–931. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btq063 [PMID:20160009]
- Fenner J W, Brook B, Clapworthy G, Coveney P V, Feipel V, Gregersen H, Hose D R, Kohl P, Lawford P, McCormack K M, Pinney D, Thomas S R, Van Sint Jan, S,, Waters S, Viceconti M, authors. **The EuroPhysiome, STEP and a roadmap for the virtual physiological human.** Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 13–9;2008;(1878)366:2979–2999. ISSN: 1364-503X DOI:10.1098/rsta.2008.0089
- Garny A, Nickerson D P, Cooper J, Santos, R. W. d,, Miller A K, McKeever S, Nielsen, P. M.F., Hunter P J, authors.

- CellML and associated tools and techniques.** Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 13–9;2008;(1878)366:3017–3043. ISSN: 1364-503X DOI:10.1098/rsta.2008.0094
- Gillespie Daniel T, author. **Stochastic Simulation of Chemical Kinetics.** Annu. Rev. Phys. Chem. 2007;(1)58:35–55. ISSN: 0066-426X DOI:10.1146/annurev.physchem.58.032806.104637
- Ginkel M, Kremling A, Nutsch T, Rehner R, Gilles E D, authors. **Modular modeling of cellular systems with ProMoT/Diva.** Bioinformatics. 12–6;2003;(9)19:1169–1176. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btg128 [PMID:12801880]
- Guyton A C, Coleman T G, Granger H J, authors. **Circulation: Overall Regulation.** Annu. Rev. Physiol. 1972;(1)34:13–44. ISSN: 0066-4278 DOI:10.1146/annurev.ph.34.030172.000305
- Hartwell Leland H, Hopfield John J, Leibler Stanislas, Murray Andrew W, authors. Nature. 1999;(supp)402:47–52. ISSN: 00280836 DOI:10.1038/35011540
- Hernandez A I, Rolle, V. Le, Defontaine A, Carrault G, authors. **A multiformalism and multiresolution modelling environment: application to the cardiovascular system and its regulation.** Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 13–12;2009;(1908)367:4923–4940. ISSN: 1364-503X DOI:10.1098/rsta.2009.0163 [PMID:19884187]
- Hindmarsh Alan C, Brown Peter N, Grant Keith E, Lee Steven L, Serban Radu, Shumaker Dan E, Woodward Carol S, authors. **SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers.** ACM Trans. Math. Softw. 1–9;2005;(3)31:363–396. ISSN: 00983500 DOI:10.1145/1089014.1089020
- Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U, authors. **COPASI — a Complex Pathway Simulator.** Bioinformatics. 10–10;2006;(24)22:3067–3074. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btl485 [PMID:17032683]
- Hucka M, Finney A, Sauro H M, Bolouri H, Doyle J C, Kitano H, Forum, t.r.o.t.S.B.M.L. (and), Arkin A P, Bornstein B J, Bray D, Cornish-Bowden A, Cuellar A A, Dronov S, Gilles E D, Ginkel M, Gor V, Goryanin I I, Hedley W J, Hodgman T C, Hofmeyr J H, Hunter P J, Juty N S, Kasberger J L, Kremling A, Kummer U, Novere, N. Le., Loew L M, Lucio D, Mendes P, Minch E, Mjolsness E D, Nakayama Y, Nelson M R, Nielsen P F, Sakurada T, Schaff J C, Shapiro B E, Shimizu T S, Spence H D, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J, authors. **The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models.** Bioinformatics. 1–3;2003;(4)19:524–531. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btg015
- Hunter Peter, Robbins Peter, Noble Denis, authors. **The IUPS human physiome project.** Pfluegers Archiv European Journal of Physiology. 2002;(1)445:1–9. ISSN: 0031-6768 DOI:10.1007/s00424-002-0890-1
- Karaaslan Fatih, Denizhan Yagmur, Kayserilioglu Abidin, Gulcur, H. Ozcan, authors. **Long-Term Mathematical Model Involving Renal Sympathetic Nerve Activity, Arterial Pressure, and Sodium Excretion.** Ann Biomed Eng. 2005;(11)33:1607–1630. ISSN: 0090-6964 DOI:10.1007/s10439-005-5976-4
- Kiselev I N, Semisalov B V, Sharipov R N, Kolpakov F A, authors; Ivanova L N, Markel A L, Blokhin A M, Mishchenko E V, editors. **Modular modeling of the human cardiovascular system.** Circulatory System and Arterial Hypertension: Experimental Investigation, Mathematical and Computer Simulation. 2012. p. 155–206. Nova Science Publishers, Inc.;
- Kutumova E O, Kiselev I N, Sharipov R N, Lavrik I N, Kolpakov F A, authors. **A modular model of the apoptosis machinery.** Adv Exp Med Biol. 2012;736:235–45. (New York, NY). Springer New York. DOI:10.1007/978-1-4419-7210-1\_13
- Lopez Carlos F, Muhlich Jeremy L, Bachman John A, Sorger Peter K, authors. **Programming biological models in Python using PySB.** Mol Syst Biol. 2013;9: DOI:10.1038/msb.2013.1
- Mallavarapu A, Thomson M, Ullian B, Gunawardena J, authors. **Programming with models: modularity and abstraction provide powerful capabilities for systems biology.** Journal of The Royal Society Interface. 6–3;2009;(32)6:257–270. ISSN: 1742-5689 DOI:10.1098/rsif.2008.0205
- Mirschel S, Steinmetz K, Rempel M, Ginkel M, Gilles E D, authors. **PROMOT: modular modeling for systems biology.** Bioinformatics. (5)25:687–689. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btp029
- Myers C J, Barker N, Jones K, Kuwahara H, Madsen C, Nguyen, N.-P. D, authors. **iBioSim: a tool for the analysis and design of genetic circuits.** Bioinformatics. 23–7;2009;(21)25:2848–2849. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btp457 [PMID:19628507]
- Novère Nicolas Le, Hucka Michael, Mi Huaiyu, Moodie Stuart, Schreiber Falk, Sorokin Anatoly, Demir Emek, Wegner Katja, Aladjem Mirit I, Wimalaratne Sarala M, Bergman Frank T, Gauges Ralph, Ghazal Peter, Kawaji Hideya, Li Lu, Matsuoka Yukiko, Villéger Alice, Boyd Sarah E, Calzone Laurence, Courtot Melanie, Dogrusoz Ugur, Freeman Tom C, Funahashi Akira, Ghosh Samik, Jouraku Akiya, Kim Sohyoung, Kolpakov Fedor, Luna Augustin, Sahle Sven, Schmidt Esther, Watterson Steven, Wu Guanming, Goryanin Igor, Kell Douglas B, Sander Chris, Sauro Herbert, Snoep Jacky L, Kohn Kurt, Kitano Hiroaki, authors. **The Systems Biology Graphical Notation.** Nat Biotechnol. 7–8;2009;(8)27:735–741. ISSN: 1087-0156 DOI:10.1038/nbt.1558 [PMID:19668183]
- Randhawa R, Shaffer C A, Tyson J J, authors. **Model aggregation: a building-block approach to creating large macromolecular regulatory networks.** Bioinformatics. 29–10;2009;(24)25:3289–3295. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btp581 [PMID:19880372]

- Randhawa R, author. **Shaffer. C.A., Tyson J.J. Model Composition for Macromolecular Regulatory Networks.** **IEEE/ACM Trans. Comput. Biol. Bioinform.** 2010;(2)7:278–287. ISSN: 1545-5963 DOI:10.1109/TCBB.2008.64
- Smith L P, Bergmann F T, Chandran D, Sauro H M, authors. **Antimony: a modular model definition language.** **Bioinformatics.** (18)25:2452–2454. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btp401
- Snoep Jacky L, Bruggeman Frank, Olivier Brett G, Westerhoff Hans V, authors. **Towards building the silicon cell: A modular approach.** **Biosystems.** 2006;(2-3)83:207–216. ISSN: 03032647 DOI:10.1016/j.biosystems.2005.07.006
- Solodyannikov Y V, author. **The elements of mathematical modeling and identification of blood circulation system.** 1994. Samara: The university of Samara;
- Vangheluwe H, author. **Multi-formalism modelling and simulation.** 2000
- Vass M, Allen N, Shaffer C A, Ramakrishnan N, Watson L T, Tyson J J, authors. **The JigCell Model Builder and Run Manager.** **Bioinformatics.** 23–7;2004;(18)20:3680–3681. ISSN: 1367-4803 DOI:10.1093/bioinformatics/bth422 [PMID:15273159]
- Wimalaratne S M, Halstead M D B, Lloyd C M, Cooling M T, Crampin E J, Nielsen P F, authors. **A method for visualizing CellML models.** **Bioinformatics.** 24–8;2009;(22)25:3012–3019. ISSN: 1367-4803 DOI:10.1093/bioinformatics/btp495 [PMID:19703920]