Cell Modeling with Reusable Agent-based Formalisms

Ken Webb

Tony White*

School of Computer Science
Carleton University, Canada

arpwhite@scs.carleton.ca

Abstract

Biologists are building increasingly complex models and simulations of cells and other biological entities, and are looking at alternatives to traditional representations. Making use of the object-oriented (OO) paradigm, the Unified Modeling Language (UML) and Real-time Object-Oriented Modeling (ROOM) visual formalisms, and the Rational Rose RealTime (RRT) visual modeling tool, we summarize a previously-described multi-step process for constructing top-down models of cells. We first construct a simple model of a cell using an architecture in which all objects are containers, agents, or passive objects. We then reuse these architectural principles and components to extend our simple cell model into a more complex cell, the goal being to demonstrate that encapsulation familiar to artificial intelligence researchers can be employed by systems biologists in their models. A red blood cell is embedded in a straight-forward manner within a larger system, which is in turn iteratively embedded within still larger systems, including a blood vessel, a circulatory system, a human being, and a simple ecology. Each complexity increment reuses the same architectural principles, including the use of agents, each of which continuously either moves passive small molecules between containers, or transforms these passive objects from one type into another. We show how it is possible to start with a direct diagrammatic representation of a biological structure such as a cell, using terminology familiar to biologists, and by following a process of gradually adding more and more detail, arrive at a system with structure and behavior of arbitrary complexity that can run and be observed on a computer.

Keywords: agent-based modeling, cell simulation, architectural reuse

---

* Corresponding author. Tel.: 613-520-2600 x2208 E-mail address: arpwhite@scs.carleton.ca

1. Introduction

Bioinformatics and systems biology researchers increasingly use computer models and simulation to explore complex inter- and intra-cellular processes. The principles of object-oriented (OO) analysis, design, and implementation, as standardized in the Unified Modeling Language (UML), can be directly applied to top-down modeling and simulation of cells and other biological entities. Intelligent systems researchers similarly strive to encapsulate and reuse knowledge captured at increasing levels of abstraction. This paper describes the process of how an abstracted cell, consisting of membrane-bounded compartments with chemical reactions and internal organelles, can be modeled using tools such as Rational Rose RealTime (RRT), a UML-based software development tool. The resulting approach, embodied in CellAK (for Cell Assembly Kit), produces models that are similar in structure and functionality to those that can be specified using the Systems Biology Markup Language (SBML) [1, 2], and CellML [3], and implemented using E-CELL [5], Gepasi [6, 7], Jarnac [8], StochSim [9], Virtual Cell [10, 11, 12], and other tools currently available to the biology community. We claim that this approach offers greater potential modeling flexibility and power because of its use of OO, UML, ROOM, and RRT. The OO paradigm, UML methodology, and RRT tool, together represent an accumulation of best practices of the software development community, a community constantly expected to build more and more complex systems, a level of complexity that is starting to approach that of systems found in biology.

We do not use ordinary differential equations (ODE) to determine the time evolution of cellular behavior, as is the case with most cell modeling systems. Differential equations find it difficult to model directed or local diffusion processes and sub cellular compartmentalization [13] and they lack the ability to deal with non-equilibrium solutions. Further, differential equation-based models are difficult to reuse when new details are added to the model. CellAK more closely resembles Cellulat [14] in which a collection of autonomous agents (enzymes, transport proteins,

lipid bilayers, and others) act in parallel on elements of a set of shared data structures called blackboards (our containers with passive small molecule objects). Agent-based modeling of cells is becoming an area of increasing research interest [13, 14] owing in no small measure to the desire to understand cellular processes at an increasing level of detail.

This paper describes a process that starts with the identification of biological entities and their relationships with each other, progresses through the gradual addition of details, and ends with an executable program that simulates biochemical pathways. This relatively simple process can be used to model any chemical-like system that involves agents transforming and moving passive small molecules. These systems include simple single cells, or arbitrarily complex systems such as a circulatory system, neural circuits, or organisms. We believe this process to be superior to other modeling approaches owing to its use of standard techniques from software engineering, the visual nature of the modeling process, and the significant potential for reuse of the model components and architectural principles.

The remainder of the paper is organized as follows. Section 2 describes the main concepts behind the Real Time Object Oriented Methodology (ROOM). Having described ROOM and the Rose RealTime tool, section 3 summarizes the process used in CellAK for cell modeling. Section 4 demonstrates reuse of the architectural principles and sample components developed in section 3, to produce arbitrarily large and complex systems.

2. The ROOM formalism and the Rational Rose RealTime tool

David Harel, originator of the hierarchical state diagram (statecharts) formalism used today in UML [15], and an early proponent of visual formalisms in software analysis and design [16], has argued that biological cells and multi-cellular organisms can be modeled as reactive systems using real-time software development tools [17, 18]. Two such commercially-available tools are I-Logix Rhapsody [19], and IBM Rational Rose RealTime [20], the latter being used to implement the system described in this paper.

IBM Rational Rose RealTime (RRT, also called IBM Rational Rose Technical Developer) is a visual design and implementation tool for the production of telecommunication systems, embedded software, and other highly-concurrent real-time systems. It combines the object-oriented (OO) features of the Unified Modeling Language (UML) [21] with the real-time specific features and visual notation of the Real-time Object-Oriented Modeling (ROOM) [22]. A RRT application's main function is to react to events in the environment, and to internally-generated timeout events, in real-time.

RRT software developers decompose a system into an inheritance hierarchy of classes and a containment hierarchy of objects, using UML class diagrams. Each architectural object, or capsule as they are called in RRT, contains a UML state diagram that is visually designed and programmed to react to externally-generated incoming messages, and to internally-generated timeouts. Messages are exchanged through ports defined for each capsule. Ports are instances of protocols, which are interfaces that define sets of related messages. All C++, C, or Java code in the system is executed within objects' state diagrams, along transitions from one state to another (which may be a self-transition to the same state). An executing RRT system is therefore an organized collection of communicating finite state machines. The RRT run-time scheduler guarantees correct concurrent behavior by making sure that each transition runs all of its code to completion before any other message is processed. RRT generates all required code from the diagrams, and produces an executable program.

CellAK uses ROOM/RRT ports, message passing and state diagrams to initially configure systems, and to bring the run-time agents and passive objects into direct contact with each other.

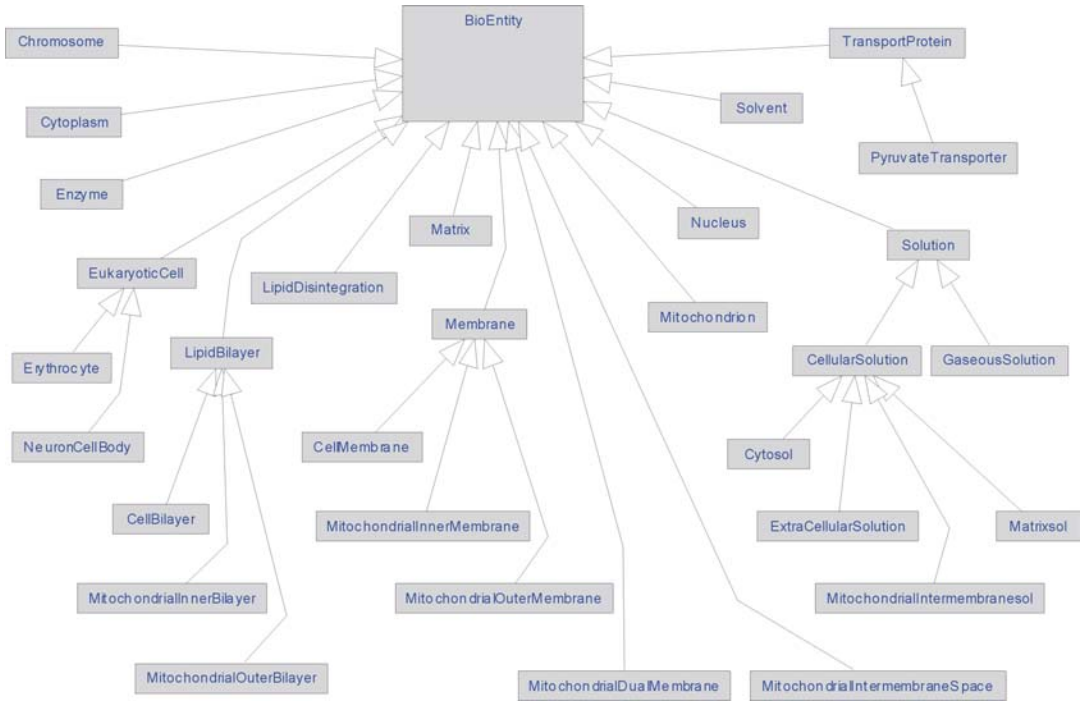

Figure 1 - Set of entities organized into an inheritance UML class diagram.

3. Process

This paper will first summarize a simple process [23] that has been used to develop cellular models. The process has four iterated steps. Each step has a number of sub-steps. These steps and the principles behind them have been adapted loosely from a number of computer industry sources [24, 25].

A cell consists of various compartmental containers, each containing active agents that act chemically on various types and quantities of passive small molecule objects. An agent is defined here as a RRT capsule that acts in a biologically-plausible manner on some substrate molecule or set of substrates, possibly located across multiple containers. Each container may contain other containers to any arbitrary depth.

The four-step CellAK process described here has been successfully used to develop models and executing simulations of cells and of cell aggregates. A simple cell model is used to motivate the discussion for each step.

Step 1: Identify entities, inheritance and containment hierarchies

The first step can be divided into three sub-steps:

1. Identify entities in the problem domain (biology),
2. Identify inheritance relations between these entities, and
3. Identify containment relations between them.

Figure 1 shows a set of candidate entities organized into an inheritance hierarchy, drawn as a UML class diagram using RRT. The lines with a triangle at one end are standard UML notation for inheritance. Erythrocyte (red blood cell) and NeuronCellBody are particular specializations of the more generic EukaryoticCell type. All entity classes are subclasses of BioEntity.

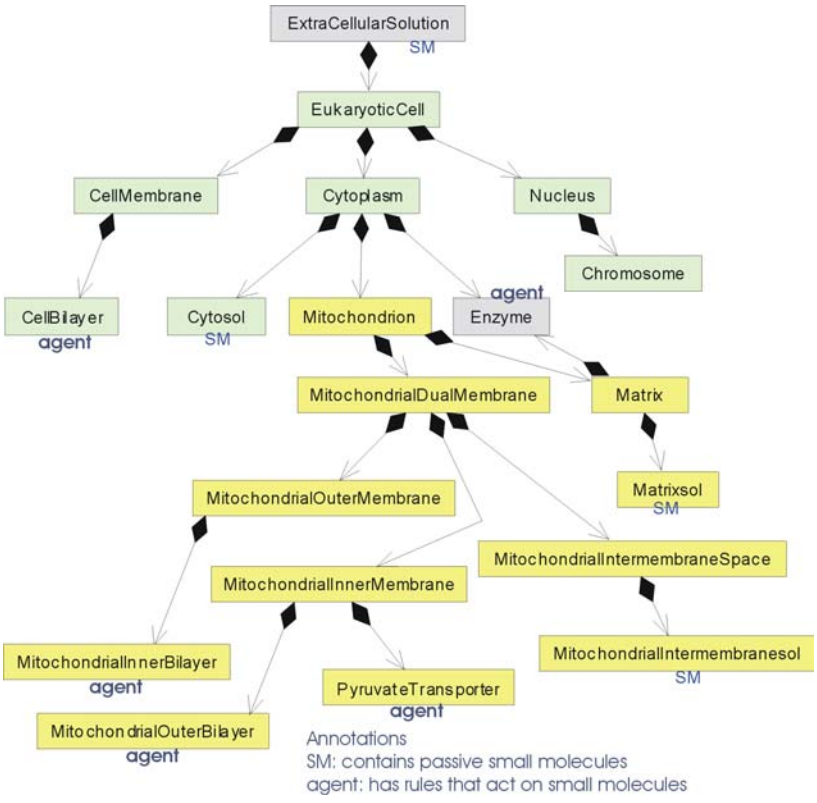

Figure 2 shows a different hierarchy, that of containment. In this UML class diagram, a EukaryoticCell is contained within an ExtraCellularSolution. The EukaryoticCell in turn contains a CellMembrane, Cytoplasm, and a Nucleus. This reductionist

Figure 2 - The containment hierarchy as a UML class diagram.

decomposition continues for several more levels. It includes the dual membrane structure of a Mitochondrion along with its MitochondrialInterMembraneSpace and solution, and its Matrix (the internal space within a Mitochondrion) and Matrixsol solution. Each subclass of Membrane contains a specific subclass of LipidBilayer. For example, CellMembrane is a subclass of Membrane, CellBilayer is a subclass of LipidBilayer, and CellMembrane contains a CellBilayer.

Figure 2 has been annotated with text to show which entities (the four Solution subclasses identified in the inheritance hierarchy) will contain small molecules (SM) such as glucose, pyruvate, and the other substrates and products of the metabolic pathways that are part of this simulation. Small molecules are captured in the model as a pair of passive (non-capsule) classes (SmallMolecules containing multiple instances of SmallMolecule, one instance for each type such as glucose or pyruvate). The three entity types identified as active objects (agents Enzyme, PyruvateTransporter, LipidBilayer) will act on the small molecules to create a dynamic metabolism.

Step 2: Establish relationships between entities

The second step involves several sub-steps, which would typically be done in parallel:

1. Identify adjacency and other relationships between capsules (relationships not identified in step 1),
2. Identify and specify protocols (interaction types between entities),
3. Create ports (instances of protocols) on capsules, and
4. Connect ports using connectors.

This step establishes the adjacency structure of the biological and chemical entities in the system, and their potential for interaction. In a EukaryoticCell, CellMembrane is adjacent to Cytoplasm. By specifying message protocols, by creating ports, and by connecting these ports using connectors, CellMembrane, Cytoplasm, and the agents and passive objects contained within these, will be able to interact at run-time.

Step 3: Define external and internal behavior patterns

The third step adds behavior to the existing structure.

1. Define the desired behavior of the system by specifying patterns of message exchange between capsules, and
2. Define the detailed behavior of each capsule using state diagrams, the combined effect of which will produce this desired overall pattern of message exchange.

State diagrams are added to each agent to implement its configuration and run-time behavior. Agents, implemented as state machines, initially exchange messages to learn which passive objects are adjacent to them, and subsequently perform actions at each timestep.

The run-time structure for the entire system, at the end of steps 1, 2 and 3, is shown in **Figure 3**. LipidBilayer and PyruvateTransporter both point to the passive small molecules within both MitochondrialIntermembranesol and Matrixsol. Because there are multiple instances of EukaryoticCell, Mitochondrion, and Enzyme, some of the pointers have double arrow heads to graphically represent this multiplicity. The instances of LipidBilayer also point to internal small molecules that contain the lipids that they are composed of, allowing for lipid creation in the Cytoplasm, lipid transport, and disintegration to be modeled.

In the sample model, the glycolytic pathway is implemented through the multiple enzymes within Cytoplasm, all acting concurrently on the same set of small molecules within Cytosol. The TCA metabolic pathway is similarly implemented by the concurrent actions of the multiple enzymes within Matrix acting on the small molecules of the Matrixsol. Movement of small molecules across membranes is implemented by the various lipid bilayers. For example, LipidBilayer within MitochondrialOuterMembrane transports pyruvate from the Cytosol to the MitochondrialIntermembranesol, and PyruvateTransporter within MitochondrialInnerMembrane transports pyruvate across this second membrane into the Matrixsol.

Figure 3 also shows the influence of the genes. Each enzyme, transporter, and other protein, is configured to reference a detailed description of its functionality. The description is contained within a table of gene data residing within the Chromosome object inside the Nucleus.

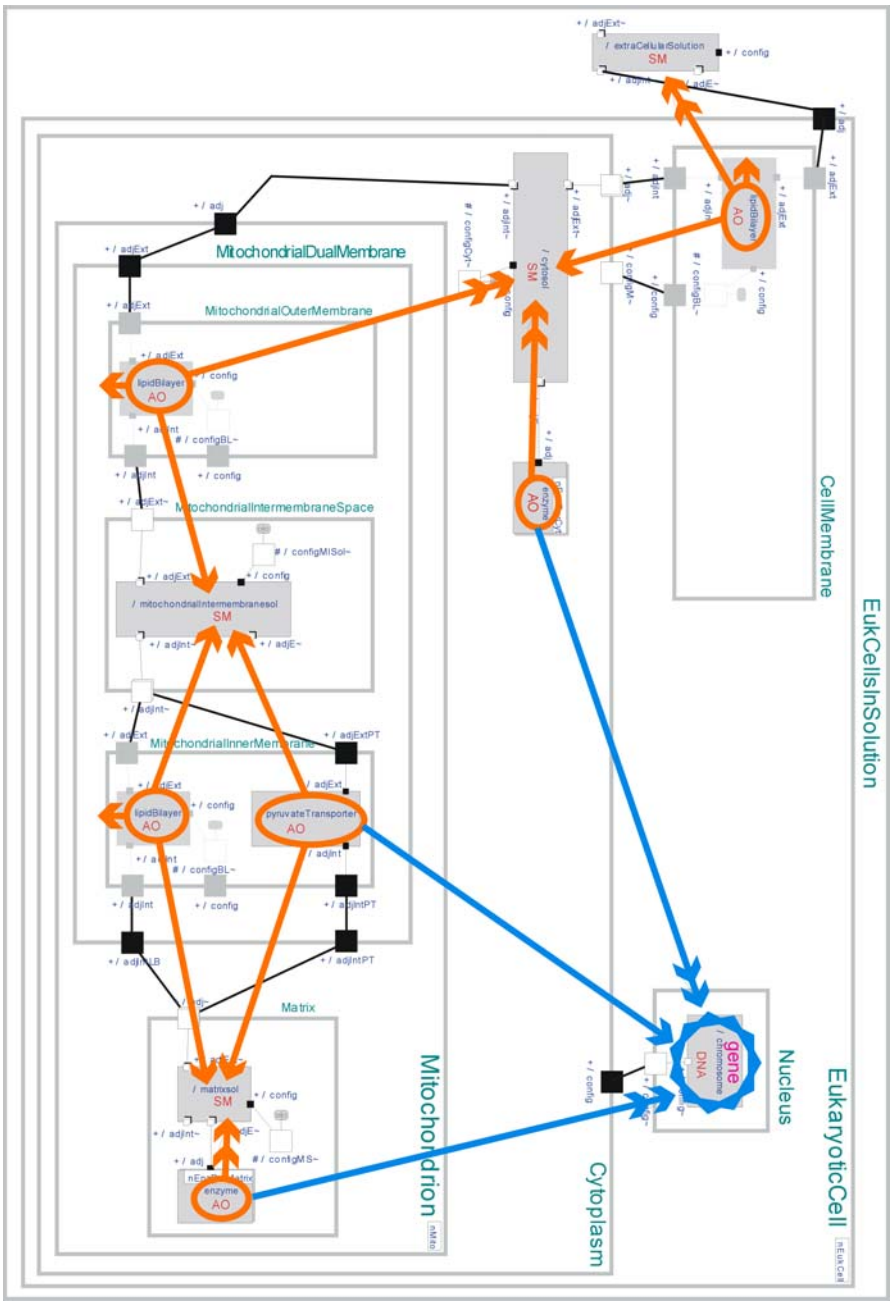

Figure 3 - Structure of the entire system with references from agents (active objects, AO) to containers that include passive small molecules (SM), and to the genes that reside at the Chromosome.

Step 4: Implement detailed behavior

The fourth step involves adding C++, C or Java programming language code to the state diagrams. At runtime each active object repeatedly times out at discrete intervals to perform its simple processing.

Enzyme reactions can take various forms. This paper considers the simplest case, in which an enzyme irreversibly converts a single substrate molecule into a different product molecule. Irreversible enzyme reactions cannot also convert the product into the substrate. More complex reactions include combining two substrates into one resulting product, splitting a single substrate into two products, and making use of activators, inhibitors, and coenzymes.

In the C++ code in Figure 4, which implements irreversible Michaelis-Menten kinetics [26, p.148+; 27] `sm->` is a reference to the SmallMolecule data structure that in this case is located in Cytosol, while `gene->` refers to a specific gene in the Chromosome. All processing by agents makes use of these two types of data, data that they know about through message exchanges that occur during initial configuration.

```
1 // Irreversible, 1 Substrate, 1 Product, 0 Activator, 0 Inhibitor, 0 Coenzyme
2 case Irr_Sb1_Pr1_Ac0_In0_Co0:
3   s = sm->molecule[gene->substrateId[0]].get();
4   nTimes = enzymeLevel * ((gene->substrateV * s) / (gene->substrateK + s));
5   sm->molecule[gene->substrateId[0]].dec( nTimes );
6   sm->molecule[gene->productId[0]].inc( nTimes );
7   break;
```

Figure 4 - One of many possible Enzyme reaction types, as implemented in C++.

The gene in CellAK is encoded as a set of features that includes protein kinetic constants. For example, in Figure 4, `gene->substrateV` refers to V the upper limit of the rate of reaction, and `gene->substrateK` is the Michaelis constant K_m that gives the concentration of the substrate molecule s at which the reaction will proceed at one-half of its maximum velocity.

The Gepasi software package [7] performs the same processing using ODEs. Gepasi implements irreversible Michaelis-Menton kinetics by operating on the following formula [27]:

$$v = \frac{V * S}{K_m + S}$$

where v is the amount of change in the quantity of the substrate and product, and S is the initial quantity of substrate. This formula is implemented on line 4 of Figure 4.

The reaction rules could be as simple or as complex as needed. Because RRT can incorporate existing C, C++ or Java code, the reaction rules could make use of existing code from other biochemistry modeling tools. The combined action of multiple reaction rules over time results in the two metabolic pathways that are part of the simple example model, the glycolytic pathway and the TCA cycle.

Validation

The main focus in CellAK has been on qualitative modeling, but it can also provide quantitative results which very closely approximate those computed using Gepasi, a tool that does claim to produce accurate quantitative results. In addition to the practical value of having CellAK generate accurate results, these expected results also help to validate the design and implementation of a specific application.

A simplified Glycolytic Pathway model was run in parallel using CellAK and Gepasi. The model includes the ten standard enzymes of glycolysis, and the eleven standard substrate and product metabolites [26, p. 308], all of which are shown in Table 1. The results of this experiment [23] are that the CellAK model tracks the Gepasi results very closely over 1000 timesteps. The difference between the CellAK and Gepasi results is never greater than 0.005%.

Table 1 – The Glycolytic Pathway (simplified). Each enzyme continuously converts its substrate small molecule (SM) into its product SM. For example, PyruvateKinase converts PhosphoEnolPyruvate into Pyruvate. All of these enzymes and small molecules are found within the Cytoplasm and Cytosol.

Enzyme	Substrate SM	Product SM
Hexokinase	Glucose	Glucose-6-Phosphate
PhosphoGlucoIsomerase	Glucose-6-Phosphate	Fructose-6-Phosphate
PhosphoFructokinase	Fructose-6-Phosphate	Fructose-1,6-Biphosphate
Aldolase	Fructose-1,6-Biphosphate	DihydroxyacetonePhosphate
TriosePhosphateIsomerase	DihydroxyacetonePhosphate	Glyceraldehyde-3-Phosphate
Glyceraldehyde-3-phosphateDehydrogenase	Glyceraldehyde-3-Phosphate	1,3-BisphosphoGlycerate
PhosphoGlycerokinase	1,3-BisphosphoGlycerate	3-PhosphoGlycerate
PhosphoGlyceromutase	3-PhosphoGlycerate	2-PhosphoGlycerate
Enolase	2-PhosphoGlycerate	PhosphoEnolPyruvate
PyruvateKinase	PhosphoEnolPyruvate	Pyruvate

4. Architectural and Component Reuse

In the systems described in this paper, all objects are pure containers, pure active objects, or pure passive objects. In other CellAK systems [28], a single object may take on two or all three of these roles. An active object is an agent that acts on passive objects. A passive object performs no actions of its own. A container contains some combination of other containers, active objects and passive objects.

Figure 5 presents an abstracted view of the simple system (**Figure 3**) described earlier. In this and subsequent similar diagrams, containers, agents and passive objects can be distinguished from each other by the existence and directionality of the connecting arrows. A container is never the source or target of an arrow. An arrow originates at an agent, and points to a passive object. The network of arrows form a graph superimposed on the containment hierarchy tree.

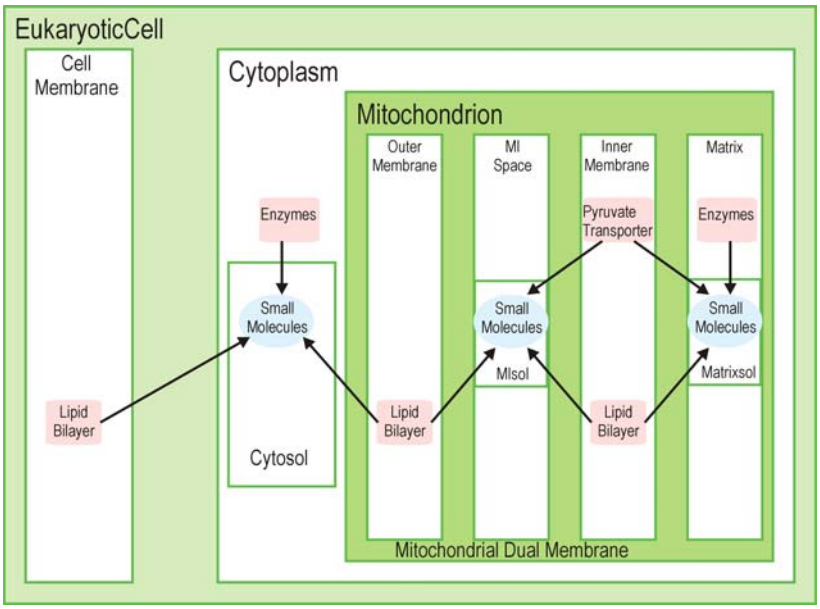

Figure 5 – An abstracted view of the original simple model of a EukaryoticCell. Compare with Figure 3.

EukaryoticCell contains two other containers. CellMembrane in turn contains an instance of a LipidBilayer agent. The Cytoplasm container includes Enzyme agents that act on various passive SmallMolecule objects that exist within the Cytosol container. The regular interaction of these Enzyme agents and SmallMolecules results in the cellular network known as the Glycolytic Pathway [26, p.308], a multistep pathway that converts Glucose into Pyruvate. Each timestep each enzyme converts a small amount of one type of small molecule (e.g. Glucose) into a small amount of another type (e.g. Glucose-6-Phosphate).

The Mitochondrion is a separate organelle contained within Cytoplasm. Each timestep the LipidBilayer within OuterMembrane moves SmallMolecules of specific types (e.g. Pyruvate and Water) between the Cytosol and the Mitochondrial IntermembraneSpace (MISpace). The LipidBilayer and PyruvateTransporter within InnerMembrane move SmallMolecules between the MISpace and the Matrix. Each Enzyme agent within the Matrix continuously transforms SmallMolecules of one type into another type. These Matrix Enzymes implement the TCA Cycle [26, p.335].

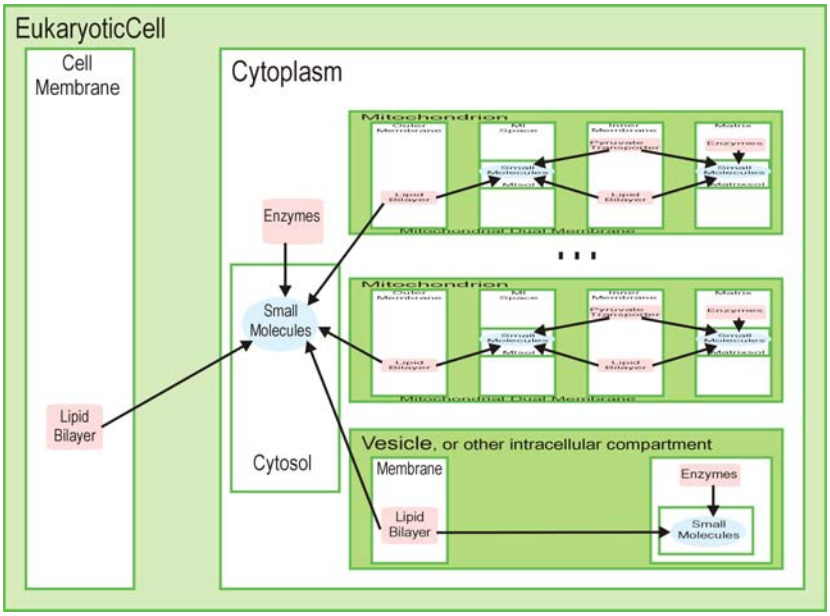

Figure 6 – Reuse of organelles, each with its containers, agents and passive objects.

We will now explore how well these architectural principles can be reused and how well they scale, by constructing simulations of iteratively more complex natural systems. One way that this architecture can be reused is by adding additional internal containers, as shown in Figure 6. Any number of additional Mitochondria (a typical EukaryoticCell contains hundreds of these) can be added by simply specifying a higher multiplicity value in the Rose RealTime model. When it starts up, each instance of the LipidBilayer agent automatically checks to see if there are passive SmallMolecule objects outside the OuterMembrane (as well as inside) that it can interact with. A cell's Cytoplasm usually contains many different types of membrane-bounded organelles, all of which can be readily added to the model because they all use the same reusable interaction mechanism. Figure 6 shows an instance of a simple Vesicle with the same types of LipidBilayer and Enzyme agents. Each type of organelle in a cell is specialized to perform certain functions, dependent on the types of agents and passive objects it contains.

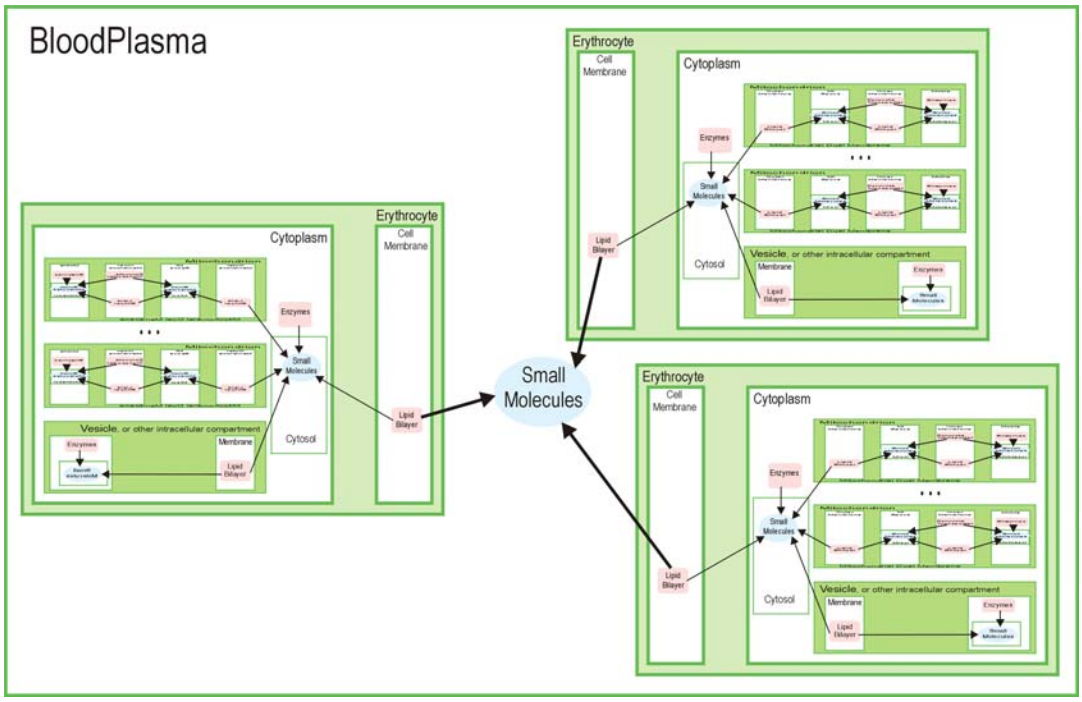

Figure 7 – Reuse of EukaryoticCell to produce multiple Erythrocytes (red blood cells) inside a unit of BloodPlasma.

Another way that the principles and some of the specific components of this architecture can be reused is by combining multiple instances of EukaryoticCell, all simultaneously interacting with SmallMolecules in the external world. **Figure 7** shows multiple instances of Erythrocyte. The LipidBilayer of each instance is continuously moving SmallMolecules between its Cytoplasm, and the BloodPlasma that all three cells are contained within. An Erythrocyte is a red blood cell. The SmallMolecules being moved bidirectionally across the CellMembranes include Glucose, Oxygen, CarbonDioxide and Water [26, p.202].

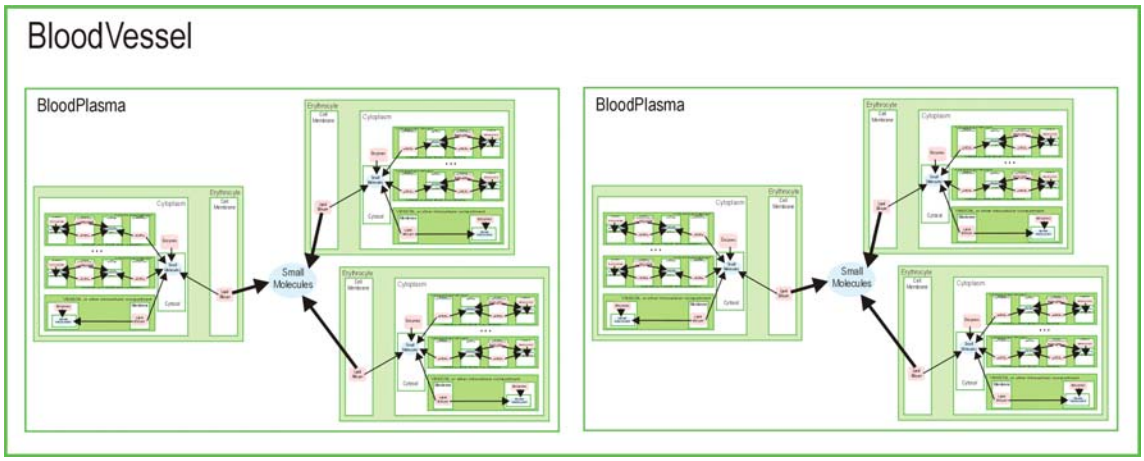

Figure 8 – Reuse of the BloodPlasma container, multiple instances of which function as the contents of BloodVessels.

Reuse can extend outwards any number of additional levels. **Figure 8** shows multiple units of BloodPlasma within a BloodVessel (Artery, Vein or Capillary).

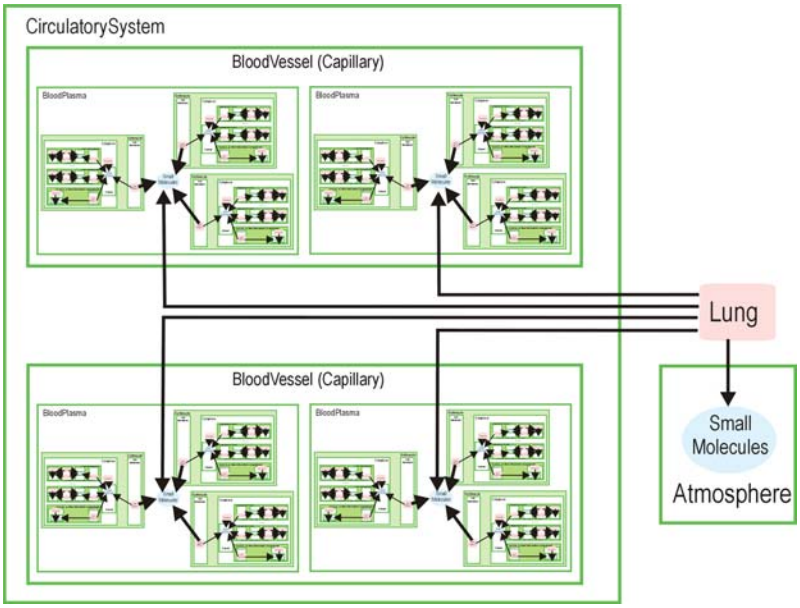

Figure 9 – A simple CirculatorySystem exchanging gases with the Atmosphere, through the actions of the Lung agent.

Figure 9 shows multiple Capillary BloodVessels contained within a CirculatorySystem. A Lung agent continuously moves SmallMolecules (Oxygen and CarbonDioxide) between each unit of BloodPlasma and the external Atmosphere. The active Lung and passive Atmosphere objects are no longer at the microscopic cellular level, but they can still be modeled using the same principles of container, agent and passive object.

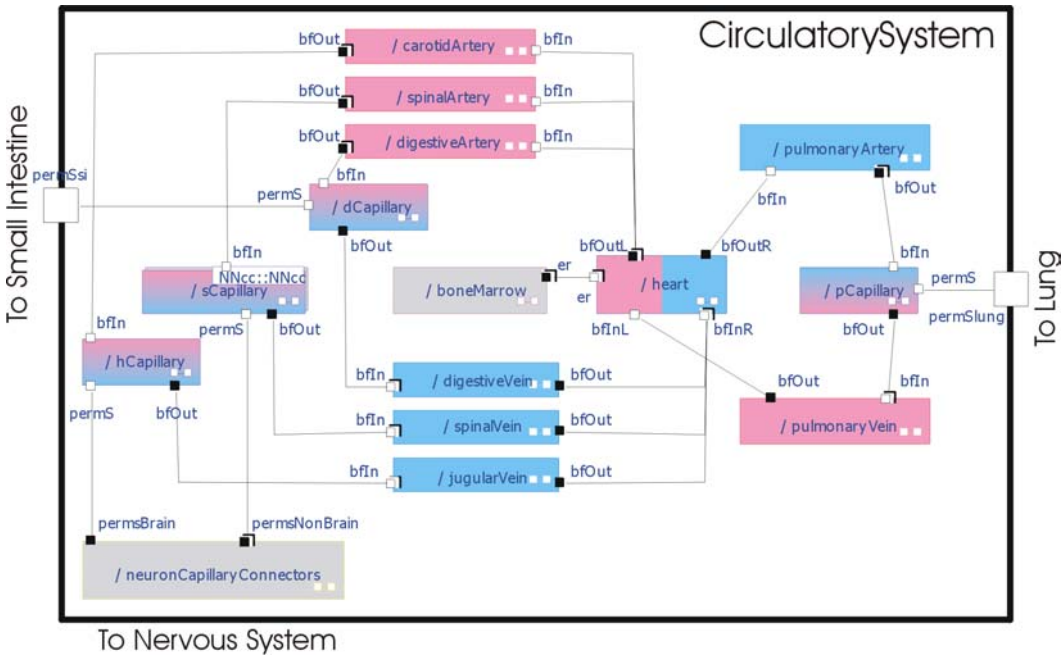

Figure 10 – A CirculatorySystem as implemented using Rose RealTime.

Figure 10 is a screen shot of how the CirculatorySystem has been modeled using Rose RealTime. The main agent visible in this diagram is the Heart. It regularly pushes Erythrocyte-containing BloodPlasma through the cyclical system of Arteries, Veins and Capillaries. As BloodPlasma and Erythrocytes pass through Capillaries, the Lung agent dynamically sets up connections that allow it to transport Oxygen molecules from and CarbonDioxide molecules to the Atmosphere. The opposite transfer takes place at Capillaries adjacent to the agents within the SmallIntestine, which receives Oxygen and gives up CarbonDioxide.

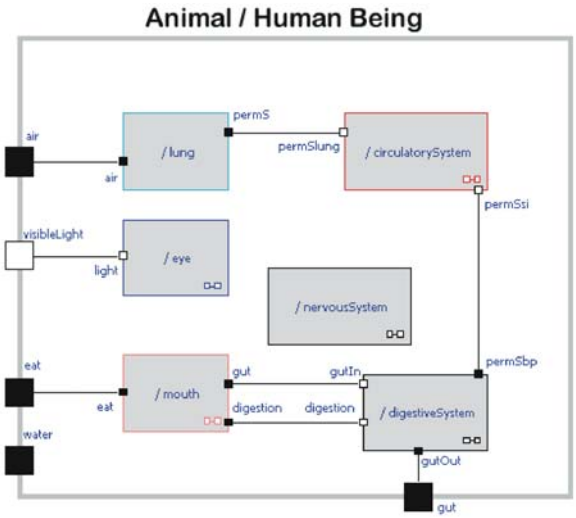

Figure 11 – The CirculatorySystem container interacting with other containers in a HumanBeing or other Animal.

Some of the processing that takes place within the other subsystems of an Animal or HumanBeing (**Figure 11**) have been modeled with CellAK using the architecture principle of containers, agents and passive agents. Interactions in a SimpleEcology (**Figure 12**) again reuse the same architectural principles and components. SolarSystem would be an additional level, in which Sun regularly transfers SolarEnergy into the system, to allow the production of food, and to allow objects to be visible to animals.

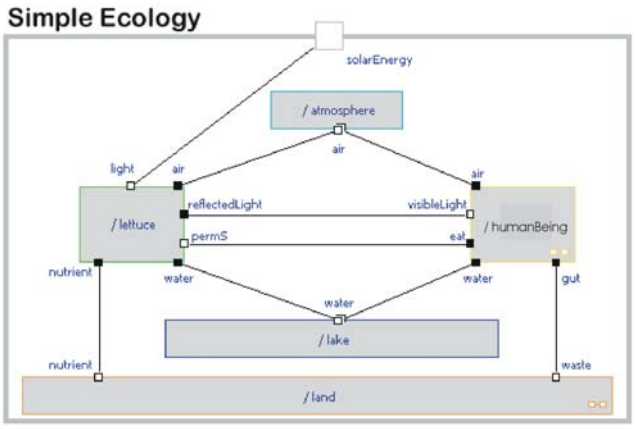

Figure 12 – A SimpleEcology in which the agents and passive objects within the HumanBeing container interact continuously with sources of air, water and food in the external world.

5. Future Work

Work is underway to produce an accessible Java implementation of the architectural principles and components described in this paper. This would replace or supplement the current proprietary Rose RealTime implementation.

6. Conclusion

This paper has demonstrated that systems of arbitrary size and complexity can be produced by iteratively reusing a simple software construction process to combine containers, agents, and passive objects. The resulting software applications can be used to qualitatively simulate aspects of biology in which agents continuously act on passive objects, or to quantitatively model specific biochemical pathways and cycles.

References

1. M. Hucka et al., "Systems Biology Markup Language (SBML) Level 1: structures and facilities for basic model definitions", http://sbml.org/documents/ , 2005.
2. M. Hucka et al., "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models", Bioinformatics, vol. 19, pp. 524-531, 2003.
3. W. Hedley et al., "A short introduction to CellML", Philosophical Transactions - Mathematical Physical and Engineering Sciences, vol. 359, pp.1073-1089, 2001.
4. System Biology Workbench, http://www.sbw-sbml.org , 2005.
5. M. Tomita et al., "E-Cell: software environment for whole-cell simulation", Bioinformatics, vol. 15, pp.72-84, 1999.
6. P. Mendes, "GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems", Comput. Appl. Biosci., vol. 9, pp.563-571, 1993.
7. P. Mendes, "Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3", Trends. Biochem. Sci., vol. 22, pp.361-363, 1997.
8. H. Sauro, "JARNAC: a system for interactive metabolic analysis", http://www.sys-bio.org , 2000.
9. C. Morton-Firth and D. Bray, "Predicting Temporal Fluctuations in an Intracellular Signalling Pathway", Journal of Theoretical Biology, vol. 192, pp.117-128, 1998.
10. J. Schaff et al., "Physiological Modeling with Virtual Cell Framework", Methods in Enzymology, vol. 321, pp.1-22, 2000.
11. L. Loew and J. Schaff, "The Virtual Cell: a software environment for computational cell biology", TRENDS in Biotechnology, vol. 19, pp.401-406, 2000.
12. B. Slepchenko et al., "Computational Cell Biology: Spatiotemporal Simulation of Cellular Events", Annual Review of Biophysics and Biomolecular Structure, vol. 31, pp.423-442, 2002.
13. S. Khan et al., "A Multi-Agent System for the Quantitative Simulation of Biological Networks", in Proceedings of the 2nd International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03), 2003, pp.385-392.
14. P. Gonzalez et al., "Cellulat: an agent-based intracellular signalling model", BioSystems, vol. 68, pp.171-185, 2003.
15. D. Harel, "Statecharts: A Visual Formalism for Complex Systems", Science of Computer Programming, vol. 8, pp.231-274, 1987.
16. D. Harel, "On Visual Formalisms", Communications of the ACM, vol. 31, pp.514-530, 1998.
17. D. Harel, "A Grand Challenge for Computing: Full Reactive Modeling of a Multi-Cellular Animal", in Workshop on Grand Challenges for Computing Research, Edinburgh, Scotland, 2002. http://www.wisdom.weizmann.ac.il/~dharel/papers/GrandChallenge.doc
18. N. Kam, D. Harel, et al., "Formal Modeling of C. elegans Development: A Scenario-Based Approach", in Proceedings of Computational Methods in Systems Biology: First International Workshop (CMSB 2003), LNCS 2602, Rovereto, Italy, 2003, pp.4-20.
19. I-Logix, I-Logix Rhapsody and Statemate. http://www.ilogix.com , 2005.

20. IBM, IBM Rational Rose RealTime. http://www-306.ibm.com/software/rational/ or http://www-306.ibm.com/software/awdtools/developer/technical/ , 2005.
21. J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley: Reading, MA, 2005.
22. B. Selic, G. Gullekson, and P. Ward, Real-time Object-Oriented Modeling; John Wiley & Sons: New York, 1994.
23. K. Webb and T. White, "UML as a cell and biochemistry modeling language", BioSystems, vol. 80, pp.283–302, 2005.
24. T. Quatrani, Visual Modeling with Rational Rose and UML, Addison-Wesley: Reading, MA, 1998.
25. P. Kruchten, The Rational Unified Process: An Introduction (2nd Edition), Addison-Wesley: Reading, MA, 2000.
26. W. Becker, J. Reece, and M. Poenie, The World of the Cell, 3rd ed, Benjamin/Cummings: Menlo Park, CA, 1996.
27. P. Mendes, Gepasi 3.30. http://www.gepasi.org , 2003.
28. K. Webb and T. White, "Combining Analysis and Synthesis in a Model of a Biological Cell", Symposium on Applied Computing (SAC 2004), Nicosia, Cyprus, 2004, pp. 185-190.

List of Keywords

agent-based modeling

cell simulation

architectural reuse

List of Figures and Tables

List of Footnotes

[*] Affiliation of author: School of Computer Science, Carleton University, Canada

[*] Corresponding author. Tel.: 613-520-2600 x2208 E-mail address: arpwhite@scs.carleton.ca

Contact address

Tony White
School of Computer Science
Carleton University, Canada
Tel.: 613-520-2600 x2208
E-mail address: arpwhite@scs.carleton.ca

Changes made to comply with referee comments

Reviewer 1:

Negative Aspects
The main problem of the paper is that the concept presented at cellular level is not used at all in practice (validation section). The logical way ought to present conceptual model of the glycolytic pathway and then to validate it in the validation section.

Comments for Authors
This paper presents modeling of structure and functional relationship of a biological cell using Unified Modeling Language (UML)and Real-time Object Oriented Modeling (ROOM)visual formalisms, and Rational Rose RealTime (RRT) visual modeling. The concept is well presented. However the technical section (ie. Section 5) needs much more work. The simulation (validation) result is not very impressive. Following are details for improvement.
1. This paper have substantial volume discuss about the entities and objects at the entire cell level, but implement only an time course transition of enzyme behavior (Fig. 10). I would expectation to see functional roles of various objects presented earlier, but they are not available.
- We present functional roles of enzymes in new Table 1.
2. Fig 2. The relationship of "CellBilayer", "MitochondrialInnerBilayer" and "MitochondrialOuterBilayer" with "CellMemberane", "MitochondrialInnerMembrane" and "MitochondrialOuterMembrane" is not well addressed. "Matrix" is not defined.
- We changed Fig 2 to better emphasize the relationship of the 3 bilayers with the 3 membranes, and added a few lines to the text.
- We provide a brief definition of Matrix.
3. you presented the glycolytic pathway in your validation section (Step 5). I would like to see 3a, what objects envolves in this pathway? It might be more effective to present a model for this pathway instead of the cell by using the same concept presented earlier.
- We added Table 1 to show the 10 enzymes and 11 metabolites involved in this pathway.
- We have retained the model of the cell and destressed the pathway discussion because of the paper's theme about reuse of architectural principles and compenents at many levels beyond the cell. The pathway is only a secondary theme, included to argue that accurate quantitative results are also possible.
3b, why only glucose showed much difference between CellAK and Gepasi, but not the other 10 metabolites. Explain it in biological context.
- We removed this entire section from the paper. It draws attention away from the main theme. It suffices to mention it and refer the reader to an earlier paper. The reason for the difference in the glucose levels was explained in the original paper submitted, but the whole discussion of glucose has now been removed.
3c, what are the other 10 metabolites.
- We added Table 1 to show all 11 metabolites (substrates and products).

Reviewer 2:

Negative Aspects
1) The experimental part itself is quite limited. Needs to be expanded.
- We have decided to scale back on the experimental results, and instead concentrate on the reuse theme. We reference a previous paper that describes the experiment.
2) The clarity of the presentation must be also improved
- We have done onsiderable editing of the paper.

Comments for Authors
It is not clear from our presentation how this paper is different from your papers with the similar names:
For example:
Combining Analysis and Synthesis in a Model of a Biological Cell , ACM Symposium on Applied Computing
- We have substantially edited the paper to differentiate it from previous publications. It focuses on the theme of architectural and component reuse, by summarizing previous work and then showing how that can be reused to build successively larger and more complex systems.

Reviewer 3:

Negative Aspects
1. The paper is too long. The authors might want to compress the manuscript by limiting themselves to the main point of this work, which is presentation of object-oriented methodology for the whole-cell modeling. An attempt (while quite legitimate) to generalize this modeling approach even to the next level of biological organization (tissue/organ) would require much more detailed analysis of structures, regulations and dynamics of the processes at the level of whole tissue or organ as well as it's relationship to the cellular level which could be a subject of separate paper perhaps.
- We have reduced the paper size from 37 pages to 21 pages, by focusing on the single theme of reuse.
2. The modeling of dynamics of the processes is implemented using commercial software package (RRT tools), which is not readily available for broad biological community. It would be important to address in a paper an alternative technical means such as open source software packages, which would allow implementing similar approach to simulation of dynamical processes. Any of the event-oriented simulation systems would be one possible example of such alternative. The current implementation of CellAK is done using RRT tools which is not an open source package. This could be considered by many computational biologists as major disadvantage of the proposed modeling system as this is not in accord with major trend to use an open source software for biological modeling.
- We have discussed this in the Future Work section. Work is currently underway to produce an open Java implementation. Our discussion is mostly about abstract aspects of the modeled system that are largely independent of the implementation tool.

Comments for Authors
The idea to benchmark CellAK model against other types of whole cell models sounds quite reasonable. Authors presented a simplified Glycolytic Pathway model, which was run in parallel using CellAK and Gepasi modeling systems. The difference in modeling results is presented on Figure 12. However, the results of such comparison might look better if they would be presented in a different form. As a suggestion - we would recommend to consider presenting CellAK results and Gepasi results versus actual experimental measurements.
- We have tried to destress the experimental part, so we can focus on the main theme of reuse.