iSimBioSys: A Discrete Event Simulation Platform for 'in silico' study of biological systems

Samik Ghosh, Preetam Ghosh, Kalyan Basu, Sajal K. Das and Simon Daefler[2]
The University of Texas at Arlington, Texas 76013, USA
[2]Division of Infectious Diseases, Mount Sinai School of Medicine, New York USA
E-mail: {sghosh, ghosh,basu,das}@cse.uta.edu, [2] simon.daefler@mssm.edu

Abstract

With the availability of huge databases cataloguing the various molecular "parts" of complex biological systems, researchers from multiple disciplines have focused on developing modeling and simulation tools for studying the variability of cellular behavior at a system level - encompassing the dynamics arising from many species of interacting molecules. In this work, we present a system engineering approach to model biological processes. In this approach, a biological process is modeled as a collection of interacting functions driven in time by a set of discrete events. We focus on the discrete event simulation platform, called "iSimBioSys", which we have developed for studying the dynamics of cellular processes in silico. As a test-bed for studying our approach we model the two component PhoPQ system, responsible for the expression of several virulence genes in Salmonella Typhimurium. We analyzed the effect of extra cellular magnesium on the behavioral dynamics of this pathway using our framework and compared the results with an experimental system. We also analyze the performance of iSimBioSys, based on the model biological system, in terms of system usage and response.

1. Introduction

Traditionally, the key focus of biology has been on detailed understanding of single genes, molecules or processes involved in particular phenotypic manifestations. This powerful approach has resulted in a significant understanding of the structure and function of individual genes and proteins. In the recent past, with the development of high throughput micro array experiments and bio chips, an explosive amount of empirical data on the molecular foundations of biological structures and functions [1] have been opened up to researchers. However, as more and more data become available, biologists are now looking beyond assigning functions to individual genes – focusing on dynamic processes, interdependent regulatory controls, and the operation of multiple interacting components.

The fundamental challenge in a "wholistic" understanding of biological processes is the complexity involved in the interaction of molecules. The complexity increases manifold as we move into higher scales such as interaction of large ensemble of cells in a tissue, or interaction of tissues in continuum for rhythmic pumping of the heart. The challenge [1] is to develop a comprehensive model integrating molecular and genetic data together with the pathway intelligence of biological systems, for a quantitative understanding of physiology and behavior of biological processes at multiple scales.

In recent years, researchers from diverse backgrounds of physical sciences, mathematics, biological and computational sciences have collaborated on developing models which capture the dynamics of biological processes. Continuous system models, which employ differential equations to simulate cellular dynamics, have been extensively used in tools like GEPASI [2] and JARNAC [4]. Stochastic discrete time models, like StochSim [3], have been developed for capturing the stochastic nature of molecular interactions based on intracellular biochemical reactions.

More integrative tools at the whole cell level have also been developed, like agent based simulation [11] and Functional Unit Representation Model (FURM) [12], which try to model cellular mechanisms and present visual representation of their functionality.

In this paper, we present a system engineering approach to model and simulate a biological process. The central theme revolves around abstracting a

0-7695-2559-8/06 $20.00 © 2006 IEEE
IEEE COMPUTER SOCIETY

complex process as a collection of interacting functions driven in time by a set of discrete events. We present the details of our approach, develop the simulation framework *iSimBioSys*, and the system architecture for the same. We quantify the performance of the framework against a test-bed process of virulence gene regulatory networks in Salmonella Typhimurium. We report results on the dynamics of the PhoPQ system corroborating similar behavior reported by experimental systems. Our results show that a discrete event based simulation platform, like 'iSimBioSys' can provide an in silico environment for studying biological systems and hypothesis testing before wet-lab experiments.

2. Modeling and Visualization

As mentioned in the previous section, use of mathematical modeling for analysis of complex biological systems has found recent attention from various disciplines of research. In particular, mathematical models are being extensively used for intracellular molecular networks like kinase cascades and metabolic pathways, gene regulatory networks and protein interaction networks [3], [4], [5]. The increased availability of data, coupled with enhanced computational powers, have also led to the development of integrative tools which model multi-cellular networks with multiple interactions and involving multiple anatomical scales.

Two of the integrative modeling and visualization environments are E Cell [5] and Virtual Cell [10], which endeavor to capture biological phenomena at multiple timescales, based on different techniques like partial differential equations, gas kinetic theory etc. Specific models, focused on modeling particular biological phenomena have also been proposed [13]. Agent oriented simulation, also known as agent based modeling (ABM) for biological systems have been studied in [6]. In [11], the authors have developed AgentCell, an ABM based digital assay for the study of bacterial chemotaxis. Another modeling technique, Functional Unit Representation Model (FURM) [12] has been proposed for large scale modeling of physiological processes. In Table 1, we present a list of some of the tools being used for modeling and simulation of biological systems.

While these efforts have contributed to a great extent in the study of biological systems, the models have focused on specific components and techniques making them tightly coupled with the system under consideration. In the following section and the next, we elucidate a discrete event based system (DES) engineering approach to the modeling of biological systems and present an object-oriented DES platform *iSimBioSys* for modeling biological networks.

Table 1. Modeling and Simulation software for biological systems

Software Platform	**Application Area**	**Availability**
JARNAC	Biological network layout tool	http://www.sys-bio.org
GENESIS	simulation environment for signaling networks	http://stke.sciencemag.org/
Cellerator	Mathematical package for automatic equation generation for network of cells	http://www.cellerator.info
Copasi	Complex pathway simulator	http://www.copasi.org
E-Cell	Whole cell simulator	http://www.e-cell.org
JigCell	Modeling of biochemical pathways	http://jigcell.biol.vt.edu
MCell	Monte Carlo simulator of cells	http:// www.mcell.psc.edu
AgentCell	Agent based simulation of cell processes	http://www.agentcell.org
FURM	Functional unit representation of biological processes	http://biosystems.ucsf.edu

3. System Engineering Approach

In the system engineering view of complex processes [9], the key notion is to abstract the complexity of the system as a set of discrete time and space variables (random variables), which capture the behavior of the system in time. The entire system is a collection of functional blocks or modules, which are driven by a set of "events", where an "event" is a combined process of a large number of micro level state transitions between a set of state variables accomplished within event execution time. The underlying assumption is that it will be possible to segregate the complete state space into such disjoint sets of independent events which can be executed simultaneously without any interaction. The application of this technique in large complex communication networks has demonstrated the accuracy of the approach for the first and higher order dynamics of the system within the limits of input data and state partitioning algorithms. For example, discrete event based system modeling has been effectively applied for designing routers, the key components responsible for routing traffic through the Internet. Discrete event based simulation techniques have also been used a wide variety of manufacturing processes and system dynamics of complex industrial processes. Our approach is based on identifying and modeling key biological functions at a cell, tissue or organ level and linking them together using the underlying intelligence of the system to create the dynamics in time domain.

We identify a biological process as a system of resources (which can typically be the various molecules, ions, ribosome, chromosome, operons, tissue, organ etc involved in the system) that are periodically changing their state between "busy" (e.g., a molecule is busy in a reaction), 'free"(e.g., a molecule is free to enter a new reaction),'created" (e.g., a molecule is created by a reaction) and "killed" (e.g., a molecule is taken up by a reaction) based on the underlying resource usage algorithms. The state transitions from one state to another are governed by transition flow rates of the functions involved in the process. The estimation of the transition flow rates is governed by modeling of the physical processes involved in the functions, together with the state of the resources participating in the execution of the function. As an example, we consider the fundamental function of *phosphorylation*, which involves the transfer of a phosphate ion from an Adenosine triphosphate (ATP) molecule to another molecule/ion resulting in the phosphorylated molecule/ion and a molecule of Adenosine diphosphate (ADP). In order to capture the dynamics of this basic biological function, we need to account for the state of the resources involved (in this case ATP, another molecule/ion and ADP). Further, the time required to perform this function, which is termed as the "*holding time*", is estimated on the basis of fundamental physical processes like kinetic theory, diffusion models and molecule binding models. At the end of the "holding time", the phosphorylation molecule can trigger an "event" to drive another functional event. As the simulation proceeds at a molecular level, the resource states are determined in terms of the "molecular count" of the individual resources. For example, after the successful completion of the phosphorylation function, the count of ATP in the system is decreased by one while that of ADP is increased by one. In this way, basic biological molecules and their events are identified, modeled and linked together in a discrete - event simulation framework to capture the dynamics of a cellular process.

As is evident from the above illustration, one of the key challenges of a system engineering approach to modeling biological processes is the identification of basic functional modules, the resources involved in them and the key events driving the interaction between the different modules. The wide variability and complexity of modules, resources and events in natural sciences further complicate the problem.

3.1. Tracing temporal dynamics

In a system engineering approach, the complexity of the system is captured in the "time" domain [9]. The dynamics of resource utilizations with progression in time unveil the interaction of the complex processes. In discrete event simulation, "simulation time" is the representation of the "physical time" of the system being modeled: a totally ordered set of values representing time in the physical system being modeled. Each event computation is associated a time-stamp indicating when that event occurs in the physical system being simulated. As mentioned in the previous section, the event time computation at a functional module, also called the "holding time" of the module is governed by the estimation of the parameters involved in the function. The exercise of characterizing the system parameters is performed as follows:

- Identify the list of discrete events that can be included in the model based on the available knowledge of the system.
- Identify the resources of interest for the experiment which are being used by the biological process for each discrete event.
- Compute the time taken to complete this biological discrete event, i.e. the holding time of the discrete event. For this purpose, it is important to identify the parameters which affect the interaction of the resources in a particular biological discrete event process and mapping them into time domain. Unlike in rate based simulation models, where it is assumed that the system state remains the same during the complete reaction of multiple molecules, the time required for completion of a biological discrete event processing is computed as a function of these parameters. In

this method, the system state changes with each molecular reaction.

- Identify the next set of biological discrete events initiated on the completion of an event. If multiple discrete events are generated, it is necessary to find out the probability of the individual next event.

The resource utilization algorithms which determine the holding time of the functional blocks, together with the resources involved and their count in the system, all play a key role in the dynamic behavior of the biological process being simulated. A discrete event based system allows modeling different blocks at different levels of granularity depending on available knowledge. In the next section, we develop the architecture of the object-oriented DES framework we use for our approach, *iSimBioSys*. We start with the overall architecture of the platform, moving on to present the current user interface and simulation visualization plane.

4. *iSimBioSys* System Architecture

The modular nature of the functional blocks involved in a system engineering approach lends itself to an object-oriented computing paradigm [8]. Each functional module is represented as an object, having its own *state* (the resources involved in the module) and its associated *behavior*, which is modeled on the functionality of the module. Another characteristic of a module are its associated *input events*, which drive the functionality of that module and its corresponding *output events* which are inputs to other modules. The central theme of a discrete event simulator revolves around the *event queue*, which is the global data structure responsible for storing time-stamped events for the simulation. A central scheduler is in control of the queue, popping events in a time-ordered manner and sending it to the corresponding modules. The scheduler is also responsible for pushing events generated by a module into the event queue. As is evident from the discussion, the scheduler together with the event queue form the heart of the simulation environment while the module objects and their behaviors are the event handlers of the framework.

Our current framework supports a multi-threaded architecture with the main simulation engine running in one thread while the visualization plane running on another. The basic architecture involves four logical packages, identified in the class diagram presented in Fig.1, which form the framework of the simulation:

- *In Silico Experimental Setup*: These set of classes are responsible for setting up the modeling and system parameters used in the particular simulation block and are generally provided through user interface or plain text files.
- *Discrete Event Process Modules:* These set of classes, derived from a common base class, essentially the resource utilization algorithms for the biological process being simulated and provide methods to compute event holding times.
- *Main Simulation Engine:* This class is responsible for handling the main thread of the discrete event simulator and implements the global event queue used.
- *Visualization and Data Generation:* These set of classes are responsible for data generation of the simulation and tracing the simulation as we describe below.

4.1 User Interface

The user interface of the current implementation involved three parts:

- *User Interface for experiment setup:* The user interface is presented before the start of the simulation for the user to set up system parameters, simulation runtime environments and visualization data. Fig. 2(a) captures snapshot of this interface.
- *Runtime visualization of simulation:* The simulation can be traced in run-time in the visualization plane which runs on a separate thread as discussed earlier. Depending on user inputs, it traces the change in resource concentration of the system and also system signal states.
- *Performance visualization:* These screens trace the various performance metrics of the simulation platform as it is executed. In the current implementation, it is trace of the memory and CPU usage of the system. Fig.3 (a) and Fig3 (b) show the CPU and memory usage system snapshots.

It may be mentioned here that the current implementation of *iSimBioSys* is based on Java 1.5 SDK and runs on a windows XP service pack 2 (enterprise edition) based Dell XPS Dimension system (Intel Pentium 4 processor with HT technology running at 3.4 GHZ), 2GB DDR2 SDRAM at 533 MHz and 250MB ATI Radeon X850 XT PE video card.

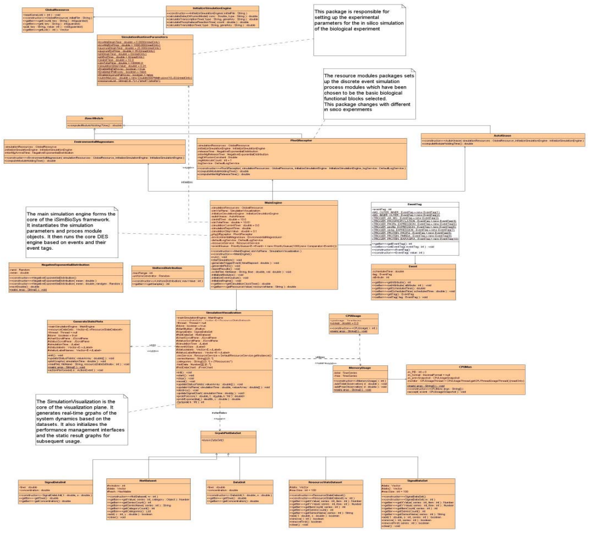
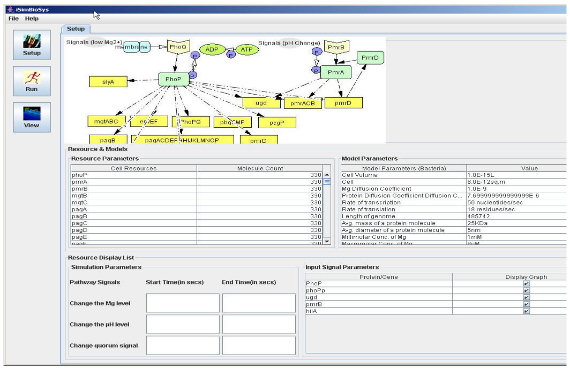

This package is responsible for setting up the experimental parameters for the in silico simulation of the biological experiment
The resource modules packages sets up the discrete event simulation process modules which have been chosen to be the basic biological functional blocks selected. This package changes with different in seco experiments
The main simulation engine forms the core of the iSimBioSys framework. It instantiates the simulation parameters and process module objects. It then runs the core DES engine based on events and their event tags.
The SimulationVisualization is the core of the visualization plane. It generates real-time grpahs of the system dynamics based on the datasets. It also initializes the performance management interfaces and the static result graphs for subsequent usage.

Figure 1. Class Diagram of *iSimBioSys*



Resource & Models

Resource Parameters

Cell Resources	Molecule Count
phoP	330
pmrA	330
pmrB	330
mgtB	330
mgtC	330
pagA	330
pagB	330
pagC	330
pagD	330
pagE	330
pagF	330

Model Parameters

Model Parameters (Bacteria)	Value
Cell Volume	1.0E-15L
Cell	6.0E-12sq.m
Mg Diffusion Coefficient	1.0E-9
Protein Diffusion Coefficient Diffusion C...	7.699999999999999E-6
Rate of transcription	50 nucleotides/sec
Rate of translation	18 residues/sec
Length of genome	485742
Avg. mass of a protein molecule	25KDa
Avg. diameter of a protein molecule	5nm
Millimolar Conc. of Mg	1mM
Macromolar Conc. of Mg	8uM

Resource Display List

Simulation Parameters

Pathway Signals	Start Time(in secs)	End Time(in secs)
Change the Mg level		
Change the pH level		
Change quorum signal		

Input Signal Parameters

Protein/Gene	Display Graph
PhoP	☑
phoPp	☑
ugd	☑
pmrB	☑
hilA	☑

Figure 2(a). User input interface of *iSimBioSys*

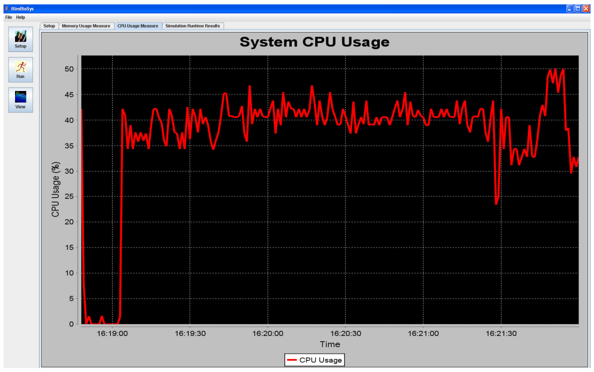


Figure 3(a). System CPU usage during simulation

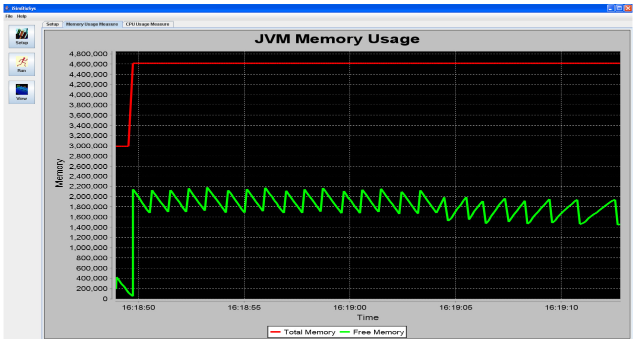


Figure 3(a). JVM Memory usage during simulation

5. Modeling Validation and Performance Measurement

In this section, we employ the discrete simulation framework, described in the earlier section to run a system level simulation of a biological process and verify results with actual wet-lab experiments

Also, we quantify the performance metrics of the simulation for this test-bed system and study how the framework behaves in terms of memory usage, CPU usage, event queue size and method call graphs.

5.1 Modeling virulence gene expression in Salmonella

Bacterial pathogenesis in Salmonella Typhimurium involves the complex interaction of regulatory pathways which play different roles in various stages of infection [7]. As mentioned earlier, we focus on the two component *phoPQ* regulatory system and its role in accomplishing parasitism of the host. [7] elucidates the role of extracellular Magnesium (Mg^{+2}) concentration as a primary signal of this pathway which acts as a global regulator of Salmonella virulence and helps in the survival and replication of the bacteria in the macrophages. Low extracellular Mg^{+2} (micro Molar concentrations) was shown to cause an increase in the expression of certain phoPQ activated genes, while high Mg^{+2} concentrations (mill Molar) caused an immediate "switch off" of these genes. We focus on this system as a test-bed for developing our approach and building a simulation framework. Information about the various gene regulatory pathways involved in salmonella pathogenesis is available in biological literature. Specific *in vivo* and *in vitro* results for virulence gene regulation in Salmonella are also available [7].

Based on knowledge extraction from literature, we have constructed the gene regulatory pathways for the phoPQ network, identifying the common intersection of the pathways (gene and gene products which are regulated by this system at various stages) as depicted in Fig. 4. (the pathways have been constructed using the Cell Designer 2.2 software which presents a structured (Extensible Markup Language (XML)) format data which can be easily rendered into the discrete event simulation framework. The gene regulatory pathway facilitates the first stage of our system modeling approach. It is the intelligence that is driving the complex dynamics of the biological process of PhoPQ system. It leads to the identification of the various "resources" as described in the previous section, involved in the biological process. Once the resource identification is completed, the next step is to abstract the functional modules which are involved in these systems.

As described earlier, the identification of the functional modules for the biological process, in this case, the expression of virulence genes on the signal of extra-cellular magnesium, is central to the development of the simulation. Each functional module is associated with the resources involved, the resource utilization algorithms and the corresponding holding time and identification of the discrete events associated with each module. Fig.5 shows the functional modules and their interactions for the test-bed system. The functional modules trace the temporal flow of the simulation, i.e. the simulation progresses by events which move from one functional block to other.

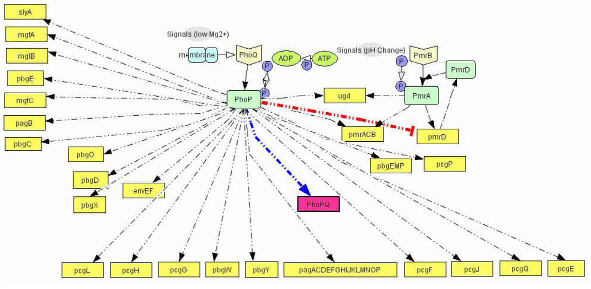


Figure 4. PhoPQ gene regulatory network

As mentioned earlier, the discrete event based simulation framework allows modeling of the various blocks depending on biological knowledge or focus of an in silico experiment.

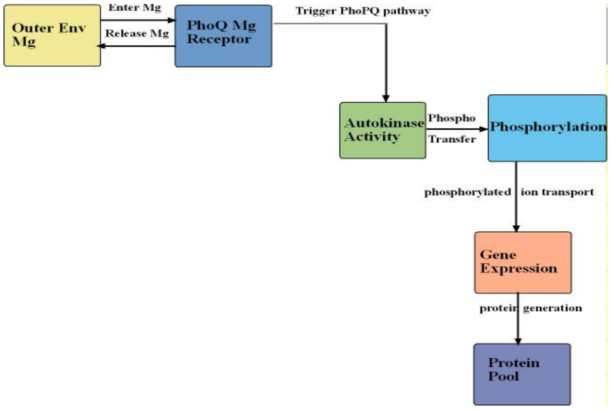


Figure 5. Functional modules of the phoPQ regulatory network

.

The experimental setup, explained in details in [7], consists of reporting the system output of the phoPQ system on bacterial cells. As reported in [7], fluorescence measure of expression of destabilized green fluorescence protein (dEGFP) under the control of a PhoPp (phosphorylated PhoP) responsive promoter was used as the reporter system. Thus, the system measure of the dEGFP was in essence an indication of the PhoPp concentration in the system. In the "in silico" simulation, results focused primarily on PhoPp as the main resource whose dynamic temporal behavior was observed in the simulation.

In the experimental system, low Mg^{2+} was maintained for a period of 60 mins, during which the system output increased, after which the signal was toggled to high $Mg^{2+.}$ The measurements of the fluorometer were taken every 15 mins for the positive activation state and Fig. 6 shows the GFP reading from the experiment. In order to simulate similar conditions "in silico", the simulation was configured to run with low Mg^{2+} for 60mins, during which no resource conflicts or starvation were assumed (i.e. the simulation would not stop due to lack of any resource). As seen in Fig. 7, the simulation responds with continuous growth in PhoPP concentration, implying increasing dEGFP fluorescence.

In another in silico setup, the system was started with high Mg^{2+} which was switched to low Mg^{2+} at 20mins which was kept low for 30 mins and toggled back to high. Fig.8 (a) and 8(b) plots the system response of PhoPp and ATP under the above conditions. In this case, as can be seen, the on-off nature of phoPp response is captured. The in silico framework also allows the study of an important resource of the gene regulatory system, ATP which is responsible of providing energy to the processes.

As seen from Fig.8 (b), as soon as up regulation of genes start, ATP is used and decreases steadily unless the signal is stopped and the system stabilize. (Note that although the signal is stopped at time 50 mins, the system produces phoPp and thus consumes ATP till around 100mins).

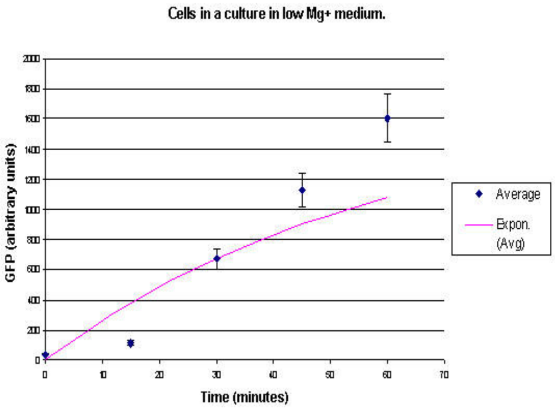


Figure 6. Response of wet-lab test for 60mins of low extracellular Mg^{+2}

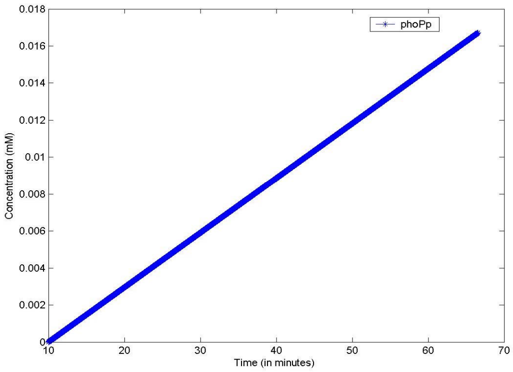


Figure 7. Response of in silico test for 60mins of low extracellular Mg^{+2}

5.1 Performance metric of the simulation platform

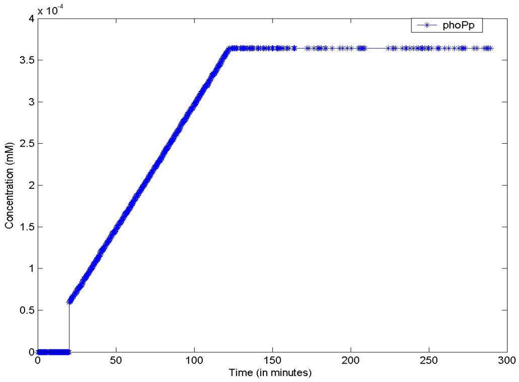


Figure 8(a). PhoPp Response 'in silico' for 30min low Mg.

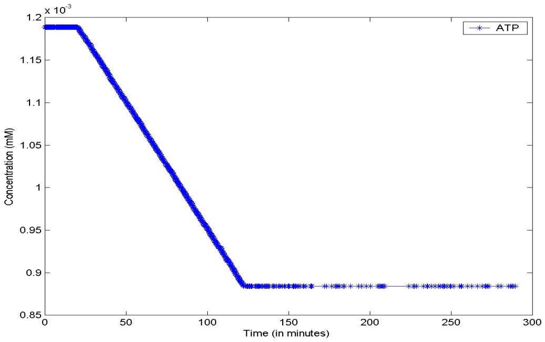


Figure 8(b). ATP Response 'in silico' for 30min low Mg.

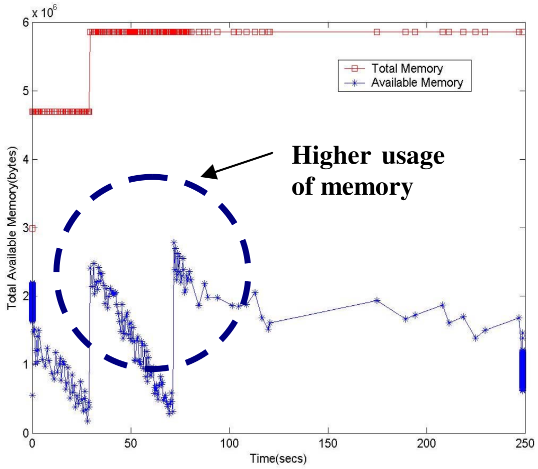


Figure 10. Virtual machine memory usage under execution scenarios of 60min

In order to measure the behavior of the simulation software framework, we have developed a memory and CPU usage monitor tool which runs in a parallel thread with the simulation. Considering the scenarios of the in silico experimental setup running for 60min of low magnesium, i.e. when the virulence genes are unregulated, we consider the response of the platform in terms of Java Virtual Machine (JVM) machine usage (total memory and free memory) and CPU usage of the particular process. Fig. 10 and 11 show the response to the above mentioned parameters. As can be seen in Fig 10, the simulation engine leads to an increasing usage of memory (decrease in size of free heap memory) as the event queue size increased with more gene expression events generated in the system. Also, it can be noted that as the size of free memory falls below a threshold, the garbage collector agent (gc) of the JVM is invoked which increased allocated memory and increases the size of free heap temporarily before starting to decrease again (the saw-tooth nature of the encircled region in Fig.10). On the same lines, the CPU usage of the process increases around the same region (Fig. 11). A spline invariant curve has been fitted to the CPU usage data to present a handle on the overall nature of usage by the process.

As a concluding remark for this section, it may be mentioned that the performance metrics are all for the computer system configuration mentioned in the earlier section and with the JVM executing the simulation process only, while no other user space process is running on the computer system.

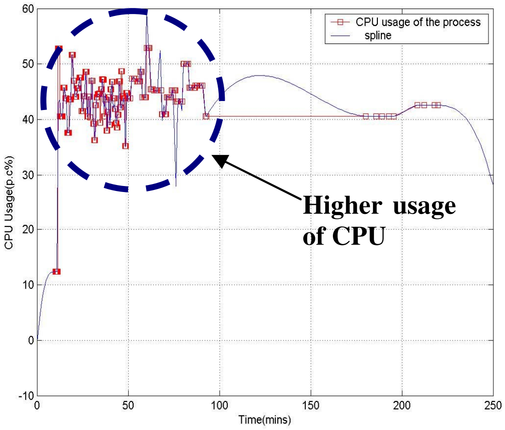


Figure 11. CPU under execution scenarios of 60mins

6. Conclusion and Future Work

We have presented a system engineering approach that uses stochastic discrete event simulation to the study of complex biological systems. We have developed an object-oriented simulation framework which can be leveraged to study interaction of such systems *in silico* and validated the concept with a test-bed system. We believe that such a system simulation approach can provide an in silico test-bed for a system level study of biological processes.

7. Additional Materials

The biological experiment is elucidated in [7]. The system parameters, resource list and models for the biological systems [14] on modeling virulence gene regulation in Salmonella are available online at http://crewman.uta.edu/dynamic/bone/projects.htm.

10. References

[1] Andrew D. McCulloch and Gary Huber, "Integrative biological modeling in silico", 'In Silico' Simulation of Biological Processes, Novartis Foundation Symposium 247, 2002.

[2] P.Mendes, "GEPASI: A software package for modeling the dynamics, steady states and control of biochemical and other systems", Comput. Applic. Biosci. 9, 563571, 1993.

[3] N. Le Novre & T.S Shimizu, "StochSim: modeling of stochastic biomolecular processes", Bioinformatics 17, 575-576, 2000.

[4] H.M. Sauro, "Jarnac: a system for interactive metabolic analysis. Animating the Cellular Map", 9th International BioThermoKinetics Meeting, Stellenbosch University Press, Ch. 33, pp.221228, 2000.

[5] M. Tomita, et.al, "ECell: Software environment for whole cell simulation", Bioinformatics, 15(1), 1999, 7284.

[6] A.M. Uhrmacher, "Concepts of Object and Agent Oriented Simulation", Transactions on SCS, 14(2), 1997, 59-67.

[7] Vamsi K. Rangavajhala, "Modeling the Salmonella PHOPQ Two Component Regulatory System", Master's Thesis, UTA, Dec 2003.

[8] D.Degenring, et.al, "Discrete Event Simulation for a Better Understanding of Metabolite Channeling", In Proc. Of the International Workshop on Computational Methods in Systems Biology, 2426 February 2003.

[9] B. Gaines, "General Systems Research: Quo vadis?", General Systems Yearbook, 24, 1979, 19.

[10] Virtual Cell Project, http://www.nrcam.uchc.edu/.

[11] T. Emonet,et. al, "AgentCell: A Digital Single-Cell Assay for Bacterial Chemotaxis", Bioinformatics, Vol. 21, No. 11, pp. 2714-2721, Oxford University Press, Oxford, UK (March 17, 2005).

[12] Sunwoo Park, C. Anthony Hunt, Glen E.P Ropella, "PISL: A Large-Scale In Silico Experimental Framework for Agent-Directed Physiological Models", 2005 Spring Simulation Multiconference (SpringSim'05), The Society for Modeling and Simulation International, San Diego, CA, April 2-8, 2005.

[13] D. Noble, "Modeling the Heart", proceedings of Physiology 19, pp. 191197, 2004.

[14] Preetam Ghosh, Samik Ghosh, Kalyan Basu and Sajal Das, "A Diffusion Model to Estimate the Interarrival Time of Charged Molecules in Stochastic Event based Modeling of Complex Biological Networks", poster paper at CSB 2005.