MeCoSim: A General Purpose Software Tool for Simulating Biological Phenomena by Means of P Systems

I. Pérez-Hurtado [#1], L. Valencia-Cabrera [#2], M.J. Pérez-Jiménez [#3], M.A. Colomer [*4] , A. Riscos-Núñez [#5]

[#] *Research Group on Natural Computing*
Dpt. of Computer Science and Artificial Intelligence, University of Sevilla
Av. Reina Mercedes s/n, 41012 Sevilla, Spain
[1] perezh@us.es
[2] luivalcab@alum.us.es
[3] marper@us.es
[5] ariscosn@us.es

[*] *Dpt. of Mathematics, University of Lleida*
Av. Alcalde Rovira Roure, 191. 25198 Lleida, Spain
[4] colomer@matematica.udl.es

***Abstract*—In recent years, the increasing importance of the computational systems biology is leading to an impressive growth of the knowledge of several real-life phenomena. In this framework, *membrane computing* is an emergent branch within natural computing that has been succesfully used to model biological phenomena. The study of these phenomena usually requires the execution of virtual experiments using mechanisms of simulation, implying the development of ad-hoc tools to simulate. However, the advance of the research is demanding general solutions to avoid the necessity of custom software developments for each matter of study, when there are some common problems to resolve. MeCoSim (Membrane Computing Simulator) is a first step in this direction providing the users a customizable application to generate custom simulators based on membrane computing by simply writing a configuration file.**

I. Introduction

Membrane computing is an emergent branch within natural computing that was initiated by Gh. Păun in 1998 [7]. The main idea is to consider biochemical processes taking place inside living cells from a computational point of view, in a way that give us a new non-deterministic model of computation. In this sense, the computational devices in membrane computing are called P systems [12].

P systems have several *syntactic* ingredients: a *membrane structure* consisting of a hierarchical arrangement of membranes embedded in a *skin* membrane, and delimiting regions or compartments where sets of objects and sets of evolution rules are placed. P systems also have two main semantics ingredients: their inherent *parallelism* and *non-determinism*.

The objects inside the membranes can evolve according to given rules in a synchronous (in the sense that a global clock is assumed), parallel, and non-deterministic way. Computation of P system is a finite or infinite sequence of instantaneous configurations, as shown in Figure 1. Computation always starts with an initial configuration of the system, where the input data is encoded.

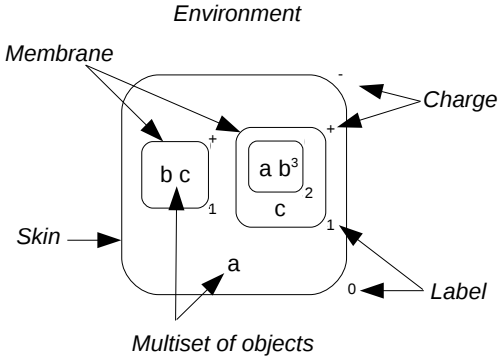

Fig. 1. Structure and main components of a P system

Although most researches in P systems concentrates on the computational power and efficiency of the devices involved, lately they have been used to model biological phenomena within the framework of Computational Systems Biology. P systems provide a modelling approach to biological systems fulfilling the requirements of a good modelling framework: relevance, understandability, extensibility and computational/mathematical tractability. In this case, P systems are used as a formalism for describing the behaviour of the system to be modelled. Several P systems models have been introduced to define oscillatory systems [5], signal transduction [3], [9], gene regulation control [10], quorum sensing [11], metapopulations [8] and population dynamics (biological ecosystems) [2], [1].

In [1], a P system based general framework for modelling ecosystem dynamics by means of P systems was presented. In contrast to differential equations, P systems explicitly represent the discrete character of the quantity of components of

an ecosystem by using rewriting rules on multisets of objects which represent the variables of the system. The inherent stochasticity, external noise and uncertainty in cellular systems is captured by using probabilistic strategies.

The mentioned framework allows us to study the simultaneous evolution of a high number of species. Furthermore, its modularity enables the addition of new ingredients in a quite simple way. In this sense, ecosystems are specified by means of multienvironment P systems composed by finite number of environments, each one containing the same membrane structure with active membranes (where membranes play a relevant role for the application of rules). The semantics is probabilistic-type and it is implemented by assigning each rule a probabilistic constant which depends on the environment and the run time.

A simulator was presented in [1] accompanying the modelling framework. It provides a flexible way to check, experimentally validate and improve computational models of ecosystems based on P systems. Furthermore, it is possible to change the initial parameters of the modelled ecosystems in order to make the virtual experiments suggested by experts. These experiments will provide results that can be interpreted in terms of hypotheses. Finally, some of these hypotheses will be selected by the experts in order to be checked in real experiments.

The simulator has been developed in Java programming language [15] by using the simulation core based on *P-Lingua* [6], [13] and *pLinguaCore* [6], [14]. P-Lingua is a programming language to define P systems in an easy-to-learn, parametric and modular way, and pLinguaCore is a Java library under GNU GPL license [19] which implements several simulation algorithms for P systems. Moreover, the simulator presented in [1] allows two different types of users: the first one is the *designer user*, who is the responsible for defining, debugging and validating the model; and the second one is the *end-user*, who uses the software in order to develop virtual experiments over the ecosystem.

One of the main problems for the mentioned simulator is the need to design, develop and maintain several different graphic user interfaces (GUIs), each one developed *ad hoc* for each modelled ecosystem. At this moment, there are two subversions focused on two real and relevant ecosystems, the first one is related to an endangered species (the bearded vulture) and the second one is related to an exotic and invasive species (the zebra mussel). Both variants have different GUIs but the same simulation core.

In this paper, it is presented *MeCoSim*, a new software application which allows the same functionality of its predecessor but enables the *designer user* for creating specific GUIs through the database definition. Thus, it has been provided an easy-to-use method for creating new *ad hoc* GUIs for specific ecosystem models. In this sense, the development of the GUI in a programming language has been avoided, delegating this process on the designer user.

This paper is structured as follows: the protocols for experimental validation of computational models and virtual experimentation by using software tools are discussed in Section II. In the next section, the features of a first software tool presented in [1] in order to simulate computational models of ecosystems based on P systems are enumerated, as well as the different user roles. In Section IV, MeCoSim, a general purpose software tool for simulating biological phenomena by means of P systems is presented, showing its main features. A case study about a model of a ecosystem related to the Pyreneen Chamois is presented in the next section, focusing on the configuration of MeCoSim in order to generate a custom simulator. Finally, in Section VI, some conclusions and future work are discussed.

II. Experimental validation and virtual experimentation

The inherent randomness in the dynamics of ecosystems makes it unfeasible the formal validation of models that attempt to reproduce their behaviour. It is therefore necessary an experimental validation by comparison of results generated by simulation tools with experimental data obtained directly from the real ecosystem.

The experimental validation protocol is extensible to any computational model of real-life processes and includes incremental debugging of the model, as shown in Figure 2.

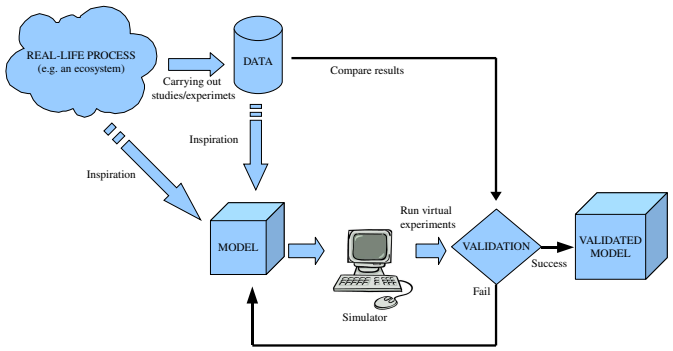

Fig. 2. Experimental validation protocol

The protocol is as follows:

1) The first step consists of stating the specific part of the system one wants to model, the objectives to achieve, the questions to be addressed and how the model will be analysed.
2) Extraction of quantitave and qualitative data of the real-life process to study.
3) Design of a model based on probabilistic P systems [1] through information provided by experts and after a process of interaction with them.
4) Development of a simulation tool to reproduce the behaviour of the process over a period of time that has

been previously studied and which have experimental data.

5) Comparison of the results obtained by the simulator with the data obtained experimentally. If the margin of error is acceptable, then it is considered that the model has been validated experimentally. Otherwise, it will return to step 2 in an iterative process.

The development of a software tool in order to simulate the behaviour of the model is fundamental for validating and debugging the model. Moreover, once a model is validated, it is possible to use the software tool to analyze the dynamics of the real-life process for different virtual scenarios that could be interesting for the experts in order to formulate plausible hypotheses, as shown in Figure 3.

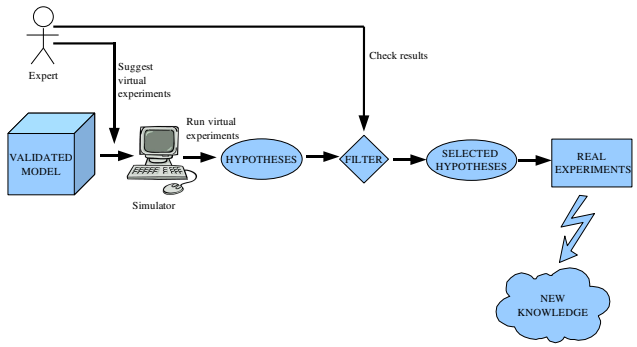

Fig. 3. Virtual experimentation protocol

The protocol for virtual experimentation is as follows:

1) The experts on the modelled real-life process suggest some initial scenarios in order to study the evolution of the system.
2) The initial parameters that determine the hypothetical scenario for each virtual experiment are configured in the software tool through the GUI and the corresponding simulations are performed through the execution of simulation algorithms. The results provide plausible hypotheses about the evolution of the real-life process for some specific initial conditions.
3) The experts filter the obtained hypotheses, discarding those considered less plausible.
4) In some cases where possible, the selected hypotheses will be confirmed through real experiments.

III. A SOFTWARE TOOL FOR EXPERIMENTAL VALIDATION AND VIRTUAL EXPERIMENTATION

In [1] a software tool was presented that simulates the modelled ecosystems. It provides two operating modes, each of them addressed to a specific user type: *the end-user* and *the designer user*. The program allows several types of actions for each user category.

On the one hand, the end-user does not need to know anything about membrane computing, and the program behaves as a black box for him. The goal of the end-user is to develop virtual experiments on the ecosystem and, for this purpose, the program allows the next actions:

- Initialization of parameter values of the ecosystem.
- Selection of the number of years to simulate.
- Selection of the total number of simulations per year.
- Saving/loading the values of initial parameters to/from files.
- Execution of simulations.
- Generation of statistical results by means of graphics.
- Saving the results to files.

Note that the name of parameters are specified by the designer user (through a spreadsheet, as it will be mentioned in section IV), but the values of such parameters need to be provided by the end-user, either directly through the interface or by loading a file.

With these actions, the end-user can modify the initial conditions of the ecosystem and run simulations, and then collect the results interactively in graphics files. For each simulated year, a number of simulations are produced and the average values as well as the standard deviation are saved. The graphics are saved in *png* format and the initial parameters of the ecosystem are saved in binary files with *.ec2* extension. In Figure 4, the graphics user interface is shown.

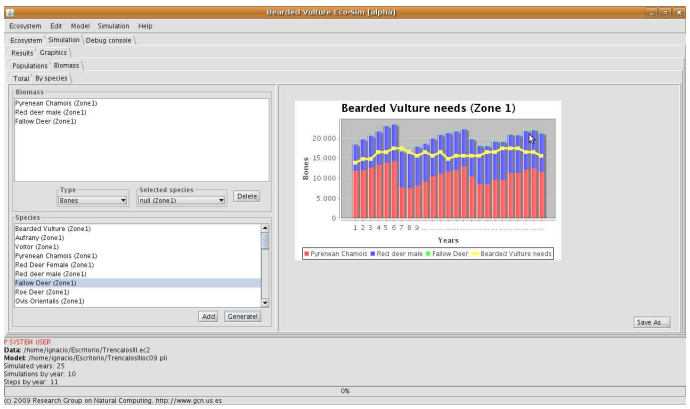

Fig. 4. A simulation tool

Transparently to the end-user, the program instantiates in each simulation a P system exhibiting the behavior of the ecosystem with the initial parameters entered for each simulation.

The designer user is responsible for specifying, debugging and validating the family of P systems used by the program. We can say that validation is performed experimentally, by comparing simulation results with actual data obtained from the ecosystem. Each simulated year corresponds to a number of computational steps of the P system, with this information also entered by the designer user.

The specification of the family of P systems in order to model a specific ecosystem is written in a P-Lingua file. The

files codifying computational models in P-Lingua have *.pli* extension.

The program offers the designer the same actions as it does to the end-user, plus some additional actions to facilitate the debugging process:

- Edition of P-Lingua files.
- Compilation of P-Lingua files.
- Simulation step by step.
- Selection of the number of computational steps per year.

The application is cross-platform and it has been developed in Java programming language along with the Swing graphical environment [20] by using the following libraries:

- pLinguaCore [14], which is responsible for processing P-Lingua files and performing simulations.
- Colt [16], which is responsible for generating random numbers.
- JDom [17] for processing XML files.
- JFreeChart [18], which is responsible for generating the graphics after each simulation.

The pLinguaCore library implements different simulation algorithms for multienvironment funcional-extended probabilistic P systems with active membranes, providing a good efficiency.

IV. MeCoSim: A general purpose application to simulate virtual experiments based on membrane computing

In section II we explained the necessity of making virtual experiments to model and analyze the dynamics of ecosystems and, as shown in Figure 2, there is a protocol to experimentally validate a model. The third step in this protocol required the development of a simulation tool to make this validation.

As justified in Section III, in order to make the referred validation, in [1] it was presented a software tool that has been successfully used to simulate the following ecosystems:

1) Scavenger Birds in the Catalan Pyrenees (Spain).
2) Zebra mussel in the reservoir of Ribarroja (Spain).

Both models of ecosystems implied the development of ad-hoc specific purpose applications, based in the structure introduced in Section III. If we would need to analyze other ecosystem, the previous protocol establish the necessity of developing another software tool to cover the specific necessities of the problem domain, with different factors, input data, etc.

However, the referred specific tools share the same simulation core (pLinguaCore) to simulate the dynamics of the ecosystems and the same method to define P systems (P-Lingua). And what is more, this core provides algorithms that are not exclusive for ecosystems, including simulation algorithms for probabilistic P systems with active membranes, stochastic P systems and others.

These reasons introduce the necessity of providing a general solution to make virtual experiments based on models of different ecosystems and other real-life biological phenomena within the framework of membrane computing. The increasing importance of Computational and Systems Biology is demanding general solutions to avoid the necessity of developing new ad-hoc software tools for each new phenomena or problem to study.

MeCoSim (Membrane Computing Simulator) try to make a move towards this direction, introducing a general purpose application to make virtual experiments inside the framework of membrane computing, avoiding ad-hoc developments of specific tools for each particular scenario or biological phenomena/problem to study.

In this sense, experimental validation will suffer a change, replacing the development of a simulation tool for the simple configuration of a file with the particular necessities of the specific scenario. This significantly reduces the *time-to-market* to validate a computational model based on P systems. Consequently, this allows us to advance faster the analysis of biological phenomena by means of P Systems.

To make this process possible, MeCoSim provides an application that offers the user the capabilities of the mentioned tools of Section III (designer and end-user capabilities) for different modelling scenarios. Thus, it delegates to the designer the task of configuring/customizing the application for each specific scenario.

To summarize, MeCoSim offers to the users (designer and end-user) a highly customizable simulators generator to apply simulation algorithms for P systems modelling several scenarios object to study. Thus, MeCoSim is a final product that avoid the necessity of ad-hoc GUI development per each scenario, introducing enough flexibility to permit the designer user to generate a simulator adapted to the scope of the domain of study of the end-user, with the inputs, parameters and outputs he need.

The process to adapt MeCoSim to each scenario requires the definition of a configuration file. The structure of the file is provided to the designer user in order to configure the custom simulator he wants to generate. After that, the file is processed by MeCoSim, that loads it in an embed database and generates the custom simulator that complies with the information introduced by the designer user. With this simple task done by the designer and without any software development, the end-user will get a custom simulator for his specific domain problem or case study. A change in the original model structure (desired structure of inputs, outputs or parameters) will be reflected in the simulator with the simple change of the configuration file and its reload in MeCoSim.

The configuration file permits the introduction of the following information:

- General data about the simulator (custom simulator name, number of years to simulate, paths to .ec2 and .pli files, among others).
- Tabs hierarchy in the main window (visual organizative elements to structure the inputs and outputs in the simulator).
- Input tables configuration (information about the input tables to enter the information of the studied phenomena).

- Parameters configuration (information about parameters to use during the simulation).
- Output elements configuration (information about tables/graph to show from the output of the simulation).

The designer user will fill the required information in this file through the use of a spreadsheet (OpenOffice.org Calc [22], Microsoft Excel [21], etc.), and the resulting file will be introduced in MeCoSim. The simulator generator will read the information in the configuration file through the use of the Jakarta POI Library [23] (this library permit reading/writing speadsheet files, including formula evaluation support), that will load this info into a SQLite embed database [24]. SQLite offers an Application Programming Interface (API) with the advantages of a relational database, permitting the use of standard SQL queries, but with an inner engine that integrates with the calling application, increasing the efficiency, and a simple file that contains the definition and inner data of the tables it use. The access to this database is done by an implementation [25] of the standard persistence engine JPA 2.0 of Java.

V. A case study: Modelling and simulating the ecosystem of Pyrenean Chamois by means of P systems using MeCoSim

The modelling of a biological system is usually complicated, taking into account the most important factors involved and the relationships between them. Computational models based on P system offers significant advantages: modularity, parallelism, and no limitation on the number of interrelated variables that evolve in parallel. These properties make them very attractive for modelling complex ecosystems. There are some aspects common to most ecosystems such as:

- They are formed by a large number of species.
- The life cycle of the organisms includes some basic processes such as: feeding, growth, reproduction and death.
- The cycles can be considered annually.
- The evolution often depends on the environment factors, such as climatic factors, soil factors, etc.
- The natural dynamics can be modified by means of human activities.

These common features to all ecosystems have led to the definition of an appropriate modelling framework based on P systems and it has been applied to two relevant real ecosystems [1].

Following, a case study is discussed: the dynamics of the Pyrenean Chamois in the Catalan Pyrenees (Spain). It is a small ungulate living in the Catalan Pyrenees, it attracts a great interest, not only from a hunting standpoint, but also naturalistic and touristic. At present the existing population in the Pyrenees is estimated at about 53,000 individuals. The status of the species has not always been so favorable, in the late 60s the population decreased down to the edge of extinction due to indiscriminate hunting. Fortunately National Hunting Reserves managed by the regional administration were created in order to save the species.

The Pestivirus disease has a very important impact at the social and economic scale in the Pyrenees. The media have publicised the different epidemics extensively as they occur, reflecting the concerns of local communities, the Government of Catalonia, ranchers, hiking groups, conservationists and hunters. The suspension of Pyrenean chamois hunting in the affected areas has led to major loss of economic income. This loss is due not only to the lack of direct income through payment of hunting licences, but also by the disappearance of the indirect income (ecoturism) that hunters and their guests bring out. Last but not least, we must highlight the considerable ecological impact of the sudden disappearance of this herbivore in the affected areas. Despite the detailed studies currently being carried out, the resulting consequences in the ecosystem are still unclear.

The computational model based on P systems for the dynamics of the Pyreneen Chamois in the Catalan Pyrenees is presented in [4]. In this section, we discuss the process of editing a MeCoSim configuration file in order to generate a custom simulator for the mentioned model, such a software tool has been used in [4] in order to simulate and debug the model.

To fill the cited configuration file, the designer uses a spreadsheet, introducing the following information:

- General Data.
  The designer user introduces an application identifier and an application name to define the ecosystem object to study, as well as the paths to the .ec2 and .pli files, the number of years to simulate, the total number of simulations to generate and the computational steps per year. He also introduces the mode to use in the program (designer or end-user).
- Tabs Hierarchy (Figure 5).

Tab Id	Tab Name	Parent Tab Id
1	Pyrinean Chamois	0
2	Input	1
3	Snow	2
4	Population	2
5	Max density population	2
6	Biological parameters	2
7	Constants	6
8	Variables	6
9	Antropical parameters	2
10	Disease parameters	2
11	Values	3
12	Ranges	3
13	Grass	2
14	Output	1

Fig. 5. Tabs Hierarchy

  The designer establishes the desired structure of tabs to contain the input and output views of the system to simulate.
- Input tables (Figure 6).
  The designer lists the tables to include in the application to contain the data of the ecosystem, specifying the identifiers of the tabs in which the tables will be put,

Table Id	Table Name	Tab Id	Columns	Init Rows	Save To File
1	Values	11	3	8	TRUE
2	Population	4	22	8	TRUE
3	Max density population	5	4	8	TRUE
4	Constants	7	9	3	TRUE
5	Variables	8	4	3	TRUE
6	Antropical parameters	9	5	8	TRUE
7	Disease parameters	10	4	8	TRUE
8	Ranges	12	2	1	TRUE
9	Grass	13	4	56	TRUE

Fig. 6. Input tables

the number of columns, the initial rows and an indicative meaning if the content of the table must be saved into the data file.

- Table columns configuration (Figure 7)

Column Id	Column Name	Default Value	Editable	Tooltip
1	Snow thickness		TRUE	Snow thickness
2	Specie		TRUE	Specie
3	Age adults		TRUE	Age adults
4	Age start fertile		TRUE	Age start fertile
5	Age leave fertile		TRUE	Age leave fertile
6	Life spectancy		TRUE	Life spectancy
7	Proportion females		TRUE	Proportion females
8	Fertility ratio		TRUE	Fertility ratio
9	Number of descendents		TRUE	Number of descendents

Fig. 7. Table columns configuration

For each table, there must be listed the set of columns, including an identifier, a name, a tool-tip to show when the user pass the mouse over each column of the table and a boolean value indicating if the data of the column is editable or not.

- P-Lingua parameters (Figure 8)

Param Name	Param Value	Index 1	Index 2	Index 3	Index 4
medw	<1,1,2>	[1..8]			
desw	<1,1,3>	[1..8]			
N1	<8,1,1>				
N2	<8,1,2>				
p1	<@ncdf,medw{1},desw{1},N1>				
p3	1-<@ncdf,medw{1},desw{1},N2>				
p2	1-p1-p3				
q	<2,1,2+2>	[1..<@r,2>]	[1..20]		
d1	<3,1,3>	[1..8]			
d2	<3,1,4>	[1..8]			
g3	<4,1,3>				
g4	<4,1,4>				
g5	<4,1,5>				
g6	<4,1,6>				
k1	<4,1,7>				
k2	<4,1,8>	[1..3]			
k3	<4,1,9>				
m1	<5,1,3>	[1..3]			
m2	<5,1,4>	[1..3]			
m3	<6,1,3>	[1..8]			
h1	<6,1,4>	[1..8]			
h2	<6,1,5>	[1..8]			
ms	<7,1,3>	[1..8]			
md	<7,1,4>	[1..8]			
f	<9,(1-1)*7+2-3,3>	[1..8]	[4..10]		
b	<9,1,4>	[4..10]			

Fig. 8. P-Lingua parameters

In the parameters tab, the designer lists the sets of parameters to use in the simulation of the model, with their name, value and up to 4 indexes for each parameter to produce final parameters to serve as part of the input of the simulation.

When the designer user of the P system has introduced the required information in the configuration file, it can be loaded into the general purpose application, generating the custom simulator that matches the specific necessities of the designer and end-user.

The custom simulator will show the input tables which permit the end-user to introduce the data for the development of virtual experiments as shown in Figure 9.

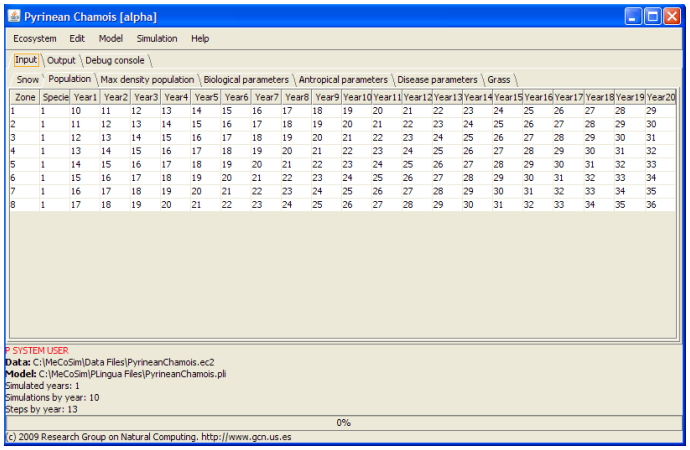

Fig. 9. MeCoSim: A custom simulator

From the input data and the parameters introduced in the configuration file, the custom application will simulate the virtual experiments and will show the outputs of the simulation.

VI. Conclusions and Future Work

In this paper, we have presented a general purpose application that experimentally validates computational models of biological phenomena based on P systems and to simulate virtual experiments over the models. The last few years have witnessed an impressive growth of several disciplines associated with Systems Biology. The understanding of biological phenomena frequently requires computational methods to complement the experimental research, increasing the knowledge of the phenomena under study. This leads to researchers developing ad-hoc simulators to run virtual experiments. However, for research to keep advancing apace we require a general simulation solution, this will avoid the situation where the same problems are solved again and again for the development of each custom software.

MeCoSim makes the first steps in this direction by providing users with a customizable application to generate custom simulators of computational models by means of P systems. To to generate a new custom simulator thee user simply writes a configuration file.

We have designed a mechanism that avoids having a software developer write software tools to execute virtual experiments complying with the users requirements. Thus the designer can concentrate on the task of writing the configuration file to customize the general purpose application in order to generate a custom/specific simulator. The next step will be to facilitate the task of the designer by providing mechanisms to apply many combinations of different formula over the input data he writes. This will enrich the possible uses of

the simulator and make the task of the designer significantly easier.

The focus must be in the automation of effort thus permitting the designer to focus on producing the correct P system design to suit real-life process the end-user wishes to study.

Keeping in mind the previous idea, we will attempt to integrate MeCoSim with other formats and applications like SBML, CellDesigner and other interesting tools. This is applying the principle that underlies Systems Biology: *the whole is greater than the sum of the parts* to software development by integrating different tools and efforts. By establishing good interfaces between them we will produce more powerful tools to provide the community better mechanisms to increase the global knowledge.

ACKNOWLEDGEMENT

The authors acknowledge the support of the project TIN2009–13192 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the "Proyecto de Excelencia con Investigador de Reconocida Valía" of the Junta de Andalucía under grant P08-TIC04200.

REFERENCES

[1] M. Cardona, M.A. Colomer, A. Margalida, A. Palau, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy. A Computational Modeling for Ecosystems Based on P Systems. *Natural Computing*, online version (http://dx.doi.org/10.1007/s11047-010-9191-3).

[2] M. Cardona, M.A. Colomer, A. Margalida, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy. A P system based model of an ecosystem of some scavenger birds. *Lecture Notes in Computer Science*, 5957 (2010), 182-195.

[3] S. Cheruku, A. Păun, F.J. Romero-Campero, M.J. Pérez-Jiménez, O.H. Ibarra. Simulating FAS-induced apoptosis by using P systems. *Progress Natural Science*, **4** 17 (2007), 424-431.

[4] M.A. Colomer, S. Lavín, I. Marco, A. Margalida, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy, E. Serrano, L. Valencia-Cabrera. Studying the evolution of Pyrenean Chamois (Rupicapra pyrenaica pyrenaica) by using P systems. Accepted paper at 11^{th} *Conference on Membrane Computing*.

[5] F. Fontana, L. Bianco, V. Manca. P systems and the modeling of biochemical oscillations. *Lecture Notes in Computer Science*, 3850 (2006), 199-208.

[6] M. García-Quismondo, R. Gutiérrez-Escudero, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez. An overview of P-Lingua 2.0. *Lecture Notes in Computer Science*, 5957 (2010), 264-288.

[7] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), pp. 108–143, and Turku Center for Computer Science-TUCS Report No 208.

[8] D. Pescini, D. Besozzi, G. Mauri, C. Zandron. Dynamical probabilistic P systems. *International Journal of Foundations of Computer Science*, **1** 17 (2006), 183-195.

[9] M.J. Pérez-Jiménez, F.J. Romero-Campero. P systems, a new computational modelling tool for systems biology.*Transactions on computational systems biology VI*. Lecture Notes in Bioinformatics 42220 (2006), 176–197.

[10] M.J. Pérez-Jiménez, F.J. Romero-Campero. Modelling gene expression control using P systems: The lac operon, a case study. *Biosystems* **3** 91 (2008), 438-457.

[11] F.J. Romero-Campero, M.J. Pérez-Jiménez. A model of the Quorum Sensing system in Vibrio Fischeri using P systems. *Artificial Life*, **14** 1 (2008), 95–109.

[12] P systems web page http://ppage.psystems.eu

[13] The P-Lingua website http://www.p-lingua.org/

[14] The pLinguaCore library website http://www.p-lingua.org/wiki/index.php/PLinguaCore

[15] The Java web page http://www.java.com/

[16] The Colt library website http://acs.lbl.gov/ hoschek/colt/

[17] The JDom library website http://www.jdom.org/

[18] The JFreeChart library website http://www.jfree.org/jfreechart/

[19] GNU GPL License http://www.gnu.org/licenses/gpl.html

[20] The Java Swing classes http://java.sun.com/javase/6/docs/technotes/guides/swing/

[21] Microsoft Excel http://office.microsoft.com/

[22] OpenOffice.org Calc http://www.openoffice.org/

[23] Jakarta POI library http://poi.apache.org/

[24] SQLite database http://www.sqlite.org/

[25] The Eclipse Integrated Development Environment http://www.eclipse.org/