Engineering Self-Modeling Systems: Application to Biology

Carole Bernon[1], Davy Capera[2], Jean-Pierre Mano[1]

[1]Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier, France
[2]UPETEC, 31520 Ramonville, France
{bernon, mano}@irit.fr ; davy.capera@upetec.fr

Abstract. Complexity of today's systems prevents designers from knowing everything about them and makes engineering them a difficult task for which classical engineering approaches are no longer valid. Such a challenge is especially encountered in actual complex systems simulation in which underlying computational model is very tough to design. A prospective solution is to unburden designers as much as possible by letting this computational model self-build. Adaptive multi-agent systems are the foundation of the four-layer agent model proposed here for endowing systems with the ability to self-tune, self-organize and self-assemble. This agent model has been applied to an application (MicroMega) related to computational biology which aim is to model the functional behavior of unicellular yeast Saccharomyces Cerevisiae.

Keywords: complex system, self-organization, cooperation, biological modeling.

1. Introduction

Nowadays systems are becoming more and more complex due to, on the one hand, the huge number of heterogeneous, autonomous and evolving components and, on the other hand, their required features of openness and scalability. A few years ago, referring to information technology systems, IBM underlined that "Even if we could somehow come up with enough skilled people, the complexity is growing beyond human ability to manage it…" and brought out the need of new approaches for dealing with complexity. Namely building "autonomic systems" capable of "running themselves and adjusting to various circumstances…" [21]. This vision may be enforced to every complex system surrounding us, especially biological ones on which this paper focuses, since natural complexity prevents designers from knowing everything about such systems, let alone controlling them.

For example, in computational biology, complexity comes from the huge amount of constantly increasing heterogeneous data that have to be gathered, visualized, exploited or processed. In systems biology, complexity arises from the need of modeling large-scale biological interaction networks for which interactions are not always known; moreover experimental data are not always available or homogeneous [23]. Furthermore integrative biology adds a level of complexity by aiming at

reconstructing “the whole by putting the parts together (once enough parts have been collected and understood)” [29]. In other words, the aim is to assemble several different yet coupled models in order to obtain an upper level one that explains a higher level of functioning. For instance, at the level of a unicellular organism, integrative biology is expected to determine how all genes and their products interact to produce the functioning organism [1].

Due to the lack of satisfactory theories for explaining biological systems, biologists usually rely on modeling and simulations to understand their behavior and different approaches exist to build these models, from mathematical ones to neural networks-based ones. The latter ones offer some interesting results in finding correlations and extracting some explaining variables but like black boxes they prevent any knowledge on the real structure and functioning of the system [14]. Moreover they hardly take into account contingency of phenomena that are a bit delayed.

Many approaches of computational modeling are concerned with biological networks [7]: Boolean networks emphase causal and temporal relationships between activation of different molecules; Petri nets extend Boolean networks with stochastic and non deterministic properties, but both approaches are hard to compose in larger models [24]. Interacting State Machines focus on state and produce models of transition of states. Process Calculi focus on events and enable modeling causality relationships between events. Both offer composition and may work in parallel or be given with hierarchical structures (with microlevels of functioning combined in a more abstract view of the system) [11][14].

To combine any of those approaches in a single (rather not simple) model some hybridization is possible through discrete event techniques. But all those approaches are not able to help scientists to construct models especially when formalism adds constraints in discrepancy with biological reality.

Ideally, biologists would like to understand underlying mechanisms of biological systems without requiring very costly *in vivo* experimentations, or at least would like to have means for focusing on really interesting ones. Most of the time the models they are provided with are static ones; influencing or modifying them in a dynamic way for trying to understand or discover new virtual experiments is usually impossible. Models have then to constantly reflect experimental data and user’s desiderata in order to be useful and thus need adaptation capabilities to stay functional in evolving environments.

This feature adds some more complexity when designing this kind of systems and classical top-down approaches are no longer helpful. New ways of engineering them are therefore mandatory for enabling them to self-build, self-tune and self-assemble.

Adopting a bottom-up approach to enable a model to build itself by giving the basic components and letting them interact in the right way is an imaginable solution. Multi-agent systems (MAS) are suited for this.

For some years now, agent technology is considered as a possible answer to biological domain problems [27][2]. Agent-based models are primarily used to deal with huge quantities of data [27] or for simulating *in virtuo* experiments: protein docking or folding [3][4], modeling signaling pathways [15][22], modeling unicellular organisms alone [25][33] or within a population [12][16]. For example, in CellAK [33] complexity is dealt with software engineering principles: a UML class diagram expresses inheritance relationships between the different components of a

eukaryotic cell (membrane, cytoplasm and nucleus) that are decomposed in turn until a certain level of complexity is reached. If in this case, agents are a way to obtain more easily understandable models compared to models based on differential equations like Gepasi [26], such a top-down approach does not make the model flexible and able to self-build.

Simulating biological systems using actual multi-agent systems is still in an early stage of study. MAS are very promising for helping understanding underlying self-organization mechanisms in populations of biological organisms. [31] considers interactions between agents representing cells for modeling tissues and focuses on self-organization phenomena within these tissues. [30] studies a self-organization phenomenon at the molecular level. [10] models the behavior of stem cells within a niche to study the emergent global organization of this population. [5] and [20] are more focusing on modeling intracellular phenomena. An AGR [13] approach is extended in [5] for describing the internal organization of a unicellular organism (E. Coli) and proposing a software environment that enables formulation of dynamic properties within this organization. [20] extends [5] by focusing on the internal states of a cell BDI which are described using temporal relationships between its intentional states. Although this approach may help to understand relations between external and internal levels of a cell, it does not provide biologists with a tool that enables them to discover new phenomena within a cell. Actually, to our knowledge, most of MAS aiming at modeling biological systems consider that laws governing components of these systems are known, or can be inferred. The organization between agents is therefore predefined and static in these models which are unable to evolve and dynamically respond to disruptions. Furthermore these laws are generally not completely known for all the levels that need to be modeled in a unicellular organism (genome, proteome, metabolome) and letting the model learn these rules in order to reflect experimental data would alleviate complexity and then modelers' workload.

Outlining how complex systems modeling can be engineered in order to make the obtained model self-build is the objective of this paper. This is illustrated with an application, MicroMega, related to systems biology which aims at modeling the functional behavior of the unicellular yeast Saccharomyces Cerevisiae.

This article is organized as follows. Concepts adopted for making a model build itself are described in Section 2. These concepts are applied in section 3 for setting up the architecture implementing the MAS related to MicroMega. Some preliminary results obtained by simulating the glycolysis metabolic pathway in Saccharomyces Cerevisiae are discussed in Section 4 before concluding.

2. Towards Self-Building Systems

Few approaches exist for engineering systems with self-organizing or emergent properties. For instance, [8] merges an analysis algorithm with simulation runs in order to tune variables reflecting chosen macroscopic properties. This approach does not completely unburden engineers because the analyzed macroscopic variables have still to be identified and the feedback has to be used in the engineering process. Engineers' role has to be reduced by initially providing the system with existing

expertise and letting it build itself while giving it, if possible and required, some minimal feedback from time to time. Under these conditions, making a complex system build itself is done by letting it autonomously change the organization between its components but also by enabling these latter parts change as well their behavior in an autonomous way. Principles of self-organization are an answer to the former point [18][9]. The second point can be fulfilled by endowing components with abilities of learning what is unknown or incompletely known from experts designing the system. This learning concerns their features (for instance, chemical or physical laws they apply) as well as they ability to appear into or disappear from the system depending on whether they are useful or not. Self-building requires then properties of self-organization at the system level, and self-tuning, self-reorganization and evolution at the component level.

A four-layer model for engineering systems having those properties is detailed in this section and applied in the following sections on an example coming from the biological domain.

2.1. Self-organization by Cooperation

The approach proposed here rests on the Adaptive Multi-Agent Systems (AMAS) theory in which self-organization is led by cooperation which embodies the local criterion that makes agents self-reorganize [6]. When an agent locally detects, at any time during its lifecycle, a situation that may be harmful for its cooperative state, it changes its relationships with others to stay or come back to a cooperative state. Situations that are against the cooperative social attitude of an agent are called Non Cooperative Situations (NCS). Furthermore processing these NCS enables an agent to constantly adapt to changes coming from its environment and therefore provides it with learning abilities.

Since the objective is to simulate the functional activity of a given system, agents of the self-building model represent either elementary domain-related objects or functions which manipulate these objects. Usually, elementary objects are easy to identify by answering the simple question « What elements are making up my targeted system? ». Such a naive approach is usually (and historically) set aside because of the tremendous implied computation load of the simulation and the huge complexity of the required model design and control. In order to deal with these two aspects, we chose to totally rely on the emergence property of Adaptive MAS:

- Simulation computation load is reduced by the fact that MAS are only composed of agents with very light computational capabilities. There is no need of any non local control to ensure consistency of the whole system activity. Moreover, as agents behave according to purely local and limited information, they are more readily to be computed in a distributed way.
- Model design complexity is greatly reduced by locally cooperation-driven self-adaptation. The adaptation process is not based on any global feedback from the system environment toward the whole MAS; no fitness function neither performance measures of the whole MAS are used. On the contrary, this adaptation process fully relies on emergence to ensure consistency and keep advantage of low computation load, readily distribution etc.

2.2. A Four-layer Model of Agent

Within the model, agents are all designed alike and consist of three main modules:

- *RepresentationModule*: this module contains all information an agent deals with: internal parameters, characteristics, knowledge about other agents etc. All other modules have to use the representation module to obtain and store all relevant data;
- *InteractionModule*: it manages all kinds of interaction between an agent and its environment (including other agents). Since communication between agents is often based on message exchange, the default interaction module manages a peer-to-peer communication system;
- *BehaviorModule*: this module defines the agent behavior which consists of two parts: the nominal behavior and the cooperative one which is subdivided into tuning, reorganization and evolution. Figure 1 shows how these behaviors interact with one another and the agent environment (other agents, user, external data).

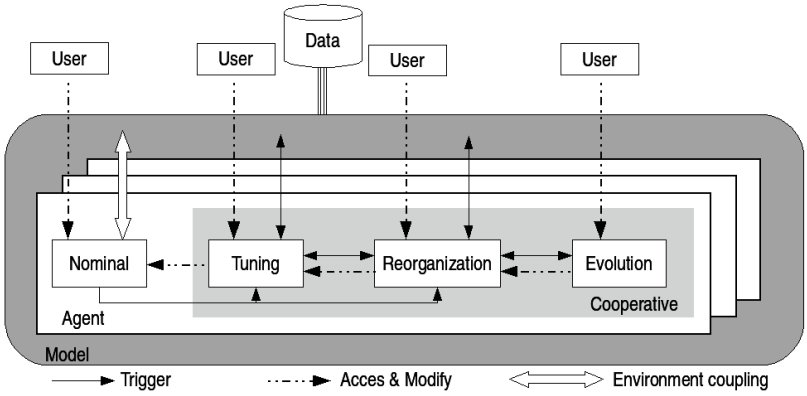

Fig. 1. Four-layer model of a self-adaptive agent

Nominal Behavior. The nominal behavior of an agent is based on the perceive-decide-act lifecycle. Basically, this behavior corresponds to whatever this agent does unless the action performed is a learning action. Learning actions modify the agent itself or its relation to its environment in such a way that, from now on, it will act in a different way than before for at least one situation. In stochastic-driven behaviors (like Monte-Carlo), the action of randomly selecting an action according to a set of parameters is not a learning action. In that context, the randomized selection IS the nominal behavior while the parameters do not change: the fact that the agent acts randomly is neither unpredictable nor random... even if the resulting actions are.

However, nominal behavior can sometimes be so complex (specification of a wide range of situations associated with smart actions) that an agent may appear to be adaptive whereas it only behaves contextually. The limit between nominal adaptation and actual adaptation is often problematic.

In Fig. 1, one can notice nominal behavior obviously interacts with the agent environment (two-sided empty arrow) or a user (dotted arrows) and sometimes it may

directly trigger tuning or reorganization behaviors (solid arrows) if it is unable to execute because of a bad parameter value or a missing link with other agents.

Tuning Behavior. This behavior is the first cooperative layer of adaptive agents: it manages parameters values tweaking for an agent. Basically, agent tuning consists in analyzing the nominal behavior computation to find cooperation failures. These Non Cooperative Situations (NCS) can be either endogenous (exception or error while executing the nominal behavior) or exogenous (messages from other agents, conflict or concurrence of actions etc.). Tuning behavior tries to solve these NCS by modifying the parameters that take an active part in nominal behavior.

Figure 1 shows tuning behavior may modify the nominal one (dotted arrow on the left), trigger reorganization if it fails to solve a problem and send messages into the environment (toward other agents) in order to delegate/propagate non cooperative situations. Tuning activation usually comes from messages sent by tuning behavior of other agents, nominal behavior failure or reorganization behavior modifications. User is free to suspend or resume tuning activity as well as modifying the parameters of the tuning algorithm.

Reorganization Behavior. It consists in modifying the way in which an agent interacts with its environment and other agents. Cooperation failures that require reorganization can occur either during nominal or tuning behavior. Usual NCS that lead to agents reorganization are:

- partial or total uselessness: an agent needs to establish a link with a new partner because it is currently not able to execute its nominal behavior;
- incompetence: an agent is unable to perform a satisfying behavior (other agents always send requests) and the tuning behavior seems unable to cope with the problem.

As illustrated in Fig. 1, reorganization behavior is caused by tuning failure, new agent appearance (evolution) or messages reception (other agents are looking for some help). Sometimes, a nominal critical failure can directly lead to reorganization-related actions: for example an agent whose main behavior consists in adding two numbers provided by two other agents cannot execute this if it is only connected with zero or one agent.

Evolution Behavior. It is the last kind of modification an agent can perform to solve any problem coming from the previous layers and, more precisely, coming from reorganization failures (see Fig. 1). Evolution[1] actions concerns system openness and consist in creating new agents or removing itself. New agents are generally created when the reorganization process is unable to find new agents to solve uselessness. Agent self-removal can only be performed when agents are in total uselessness.

As a matter of fact, the model is continuously evolving according to the data it is perceiving while the cooperative behavior is enabled and non cooperative situations are detected. Nevertheless this adaptation is not transient (purely instantaneous) because cooperative behaviors must remain consistent with past learnt states. So, the

[1] Evolution here is not related to Darwin's theory of evolution (species evolution) but only refers to the dynamics of the system from an openness point of view (add/remove agents).

model is supposed to converge towards a stationary state. Once this state is reached cooperative behaviors can be disabled or even removed, the only remaining behavior being the nominal one.

The next section details how this framework has been applied for modeling a biological complex system.

3. System Architecture in MicroMega

MicroMega[2] application aims at simulating the functional behavior of a yeast cell. The computational model has to integrate phenomena from genomic level (e.g. genes activity) to macroscopic data (e.g. quantity of consumed/produced O_2, CO_2, glucose etc.). Since chemical elements involved into cellular activity are hugely numerous as well as their transformations/interactions, MicroMega has to be designed taking into account the need of autonomous building and tuning of the yeast model according to experimental data and user wishes as well as adaptive capabilities toward user interaction to help or drive this highly complex process.

MicroMega is based on a unified model with a multi-agent system that simulates chemical reactions from RNA production at genomic level to exchanges of substrates with extracellular environment. The system is made up of two main classes of agents: functional agents (elements or reactions) or viewer agents. These different types of agent and the way the simulation is handled to control their computations are presented hereafter.

3.1. Functional Agents

In MicroMega, functional agents represent either physical elements or chemical reactions. Elements and reactions interact: reactions produce or consume elements and elements may act as regulators for reactions.

Element Agents. These agents are representative of physical items that constitute the cell: RNA, substrates, proteins, protons (H+), water etc. Their core function is to manage the quantity of the element they represent during the simulation. As this quantity is generally not uniformly distributed in the cell, each physical element is actually represented by a group of element agents in which each agent manages the quantity into a given compartment (extracellular, cytoplasm etc.).

However, element agents are quite passive from a functional point of view: they do not actually modify or compute their quantities by themselves because these modifications are computed by reaction agents (see below). Some information like unitary mass or internal energy of an element can also be managed by element agents.

Element agent tuning behavior is able to handle incompetence: the agent receives a message from either a viewer agent or a reaction agent it regulates. This message requests a different quantity value. The agent modifies its current quantity value according to the requested one and sends requests to the reaction agents it is linked

[2] Project funded by ANR (National Agency for Research) under the number 05-BLAN-0202-05.

with to increase or decrease their reaction speed (and therefore be more or less consumed or produced for tuning their quantity).

Reorganization behavior handles:

- uselessness: an element agent with no reaction agent is useless: it has to broadcast messages to inform reaction agents that it is available;
- incompetence: an element agent receives messages related to its quantity value from either viewers (value not matching experimental data) or reactions whose this element is a regulator (the reaction agent wants to change its context – see below). This element agent has tried to change its quantity during its tuning behavior and this is not sufficient. Therefore the confidence it may have toward its current partners is going to decrease and it may search for new partners in order to regulate other reactions.

1 Fructose1,6DP + 2 ADP + 2 NAD+ → 2 Pyruvates + 2 ATP + 2 NADH,H+ (1)

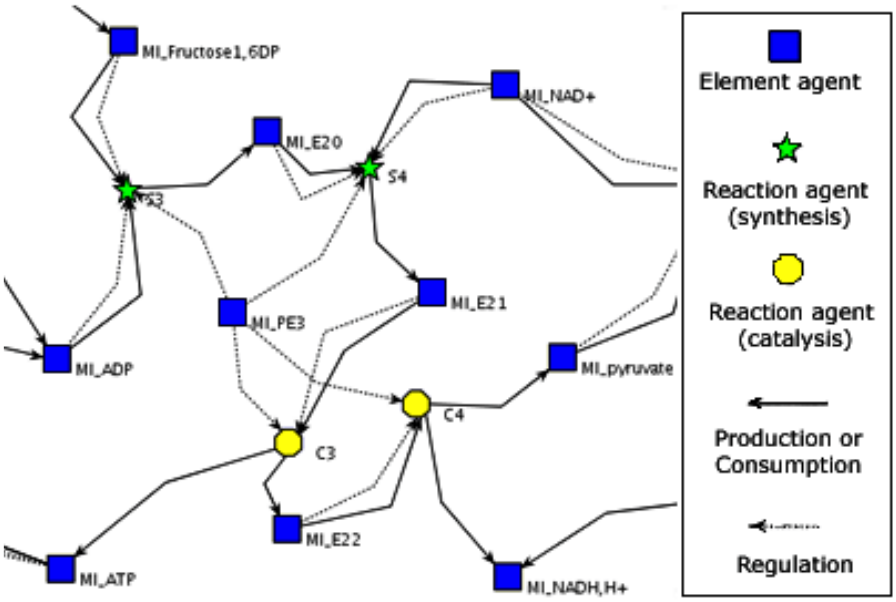

Fig. 2. Agentification of the complex reaction (1) catalyzed by one protein. Names of elements are prefixed by MI_ because this reaction takes place in the mitochondrial compartment

Reaction Agents. They manage all transformation and transportation of elements within the cell. Chemical reactions are reduced to two elementary subtypes of reaction:

- Synthesis: a given element is broken down into two parts. An example of basic synthesis reaction is the electrolysis of water to give hydrogen and oxygen: 2 H_2O → 2 H_2 + O_2.
- Catalysis: two elements are docked together to build a new one. Hydrogen combustion (2 H_2+ O_2 → 2 H_2O) is a well-known example of such a reaction.

Each synthesis or catalysis agent is also characterized by its stoichiometry, that is to say the quantitative relationship between the reactants and products of the chemical reaction, or -from a practical point of view- the weights of inputs and outputs of reaction agents. By combining these two subtypes of reaction and using intermediate elements, more complex reactions can be built. Figure 2 shows a combination of 2 synthesis (stars), 2 catalysis (octagons) and 3 intermediate elements (E20, E21, E22 – squares) to achieve transformation of 1,6 DP to pyruvate with one protein (PE3) as the reaction speed regulator (dotted lines represent regulation influence).

Two types of reaction agents manage transport:

- Passive carriers which produce/consume ambient energy (temperature, pressure). This energy is usually dissipated or absorbed around the reactive site;
- Active carriers which need to consume high energy (from ATP for instance) to achieve transportation.

Carrier agents are only used for intercompartment transport because we assume elements are uniformly distributed in each compartment.

Moreover genes are modeled as reaction agents: they consume several metabolites and produce one RNA and one protein. Genes design is peculiar because they must be paired with an RNA element. This RNA is produced by a gene but can also be consumed by this gene and is involved as a regulator of its activity.

During the simulation, each reaction agent consumes and produces elements this agent is linked with according to its stoichiometry and a contextual reaction speed. This speed determines how fast the reaction is running and as a consequence how much of the quantities of elements the reaction consumes and produces. Speed is contextual because reactions are regulated by their environment: some elements can speed up the reaction rate or slow it down. These elements define the reaction context as the vector of their current quantities. Thus one speed is associated with each possible vector in order to define the discrete function which enables speed computation.

Reaction agent tuning behavior handles the following NCS:

- Unproductiveness: the current context is unknown so the agent has to use a default context (with a speed value of 0). The agent tries to solve this NCS by adding a new context which corresponds to the current situation.
- Incompetence: the agent receives requests from neighbor element agents to change a previous consumption or production. The agent tries to adjust either its stoichiometry values or the speed. Speed adjustment can be performed by modifying the speed associated with the context used or by influencing the selection of contexts. The selection can be changed either by adjusting the ranges of regulators of the existing contexts or by requesting different values from the regulators themselves to switch situation for one that will allow to select a better context.

Reorganization behavior handles uselessness (missing consumption/production link) and incompetence. For instance, if a reaction agent is unable to tweak the speed of a given context (because two different speed values are alternately requested in the same context); it may find a new regulator to help distinguishing the two situations.

3.2. Viewer Agents

Viewer agents enable interactions between functional agents and the system user by extracting comprehensible data (for analysis and display purposes) and injecting experimental data and constraints specified by this user. This is a way to inject experts' knowledge into the model. Two types of viewers are currently used in MicroMega:

- *ElementViewerAgent*: this basic type of viewer agent is dedicated to gather quantities of a given list of element agents. It is possible to specify experimental values associated with these quantities to dynamically compare them with simulation results;
- *ElementSetterAgent*: this is a viewer agent which remotely controls the quantity value of a given list of element agents. Whatever the computed quantity value of the element an *ElementSetterAgent* will nominally set the quantity value according to its database. These viewer agents are used to control the activity of specific elements during simulation (some regulators, genes etc.).

  However, other kinds of viewer agents could be defined:
- *BioMassCheckerAgent*: this viewer agent evaluates the whole mass of the cell by summing the mass of all elements of the system;
- *CompartmentsAgent*: this viewer agent analyses the system focusing on carrier agents to identify different compartments within the cell. This information could be useful if the system is able to self-reconfigure and the user might wish to control such a critical parameter in order to limit changes at the global cell structure level.

The nominal behavior of viewer agents consists in accessing data of functional agents and storing the gathered values. Viewer agents which compare the gathered data to experimental ones can also compute errors. These errors will be used during the tuning part which usually deals with conflicts. If the error computed during nominal behavior differs from 0, the viewer agent sends a positive reinforcement message to the involved element agents; otherwise it sends them quantity error messages.

3.3. Simulation Control and Computation

MicroMega simulation process is handled by a single thread and decomposed into elementary time steps. Each step represents an arbitrary discrete item of time and is decomposed into two phases (each one is handled by a scheduler): the nominal computation and the cooperative computation which is itself broken down into tuning, reorganization and evolution (according to the model given in section 2.2).

Nominal computation corresponds to simulation computation. Within each step, the nominal scheduler notifies each agent of the system to execute its nominal behavior. These behaviors actually manage yeast simulation through chemical elements quantity updates and trigger data gathering and injection in the case of viewer agents. As a matter of fact, nominal scheduling is decomposed as follows for each step t:

1. Activation of each reaction agent to compute quantity variations of elements;

2. Activation of each element agent to update its quantity according to the quantity variations computed by reaction agents;
3. Activation of all viewer agents to update data from agents and/or check whether MAS organization and parameters fit with the user data (experimental data, for example).

Cooperative computation corresponds to yeast model adaptation according to both internal problems (for example, if model state becomes inconsistent by computing negative element quantities) and external data (for example, if collected data no longer fit experimentally probed quantities or well-known properties of the cell). Each agent sequentially executes parameters tuning (adjustments of quantities, of reaction speed etc.), reorganization (addition/removal of reaction regulations or producer/consumer link between reactions and elements etc.) and evolution (addition/removal of new elements or reaction agents). During each step t, the cooperative scheduler activates agents following the reverse order of the nominal phase:

1. Activation of each viewer agent to notify functional agents if they detect any problem for the parameters computed during step t of nominal computation;
2. Activation of each reaction agent to check if they have received any message (under- or overestimated speed) or have detected any computation problem (like an unknown contextual value or speed, missing element agents) while performing their production/consumption during step t-1;
3. Activation of each element agent to check inconsistent parameters (negative values) or bad/good quantity values (from step t-1) notified by quantity-related messages from either viewer agents or reaction ones.

As one can acknowledge, all the agents of each group (element, reaction, and viewer) can be readily triggered simultaneously because no direct interaction occurs between agents that belong to the same type.

4. Simulation Results

To test MicroMega we have first addressed glycolysis modeling as an example of biochemical pathway. Glycolysis allows cells to transform absorbed glucose into energy or smaller metabolites like pyruvate that roughly are used in biomass production. Mathematical models in cell biology are overcome by the complexity of described systems and are limited to linear dynamic systems in steady state. Model reduction is of major use to produce more simple and stable models that balance between approximation accuracy and numerical efficiency [19].

Biochemical network models transpose metabolic systems into differential equation systems like: $s(t) = Nv(s(t); p)$ where s is the vector of metabolite concentrations and v is the vector of reaction velocities. The vector p contains the kinetic parameters, and the stoichiometric matrix N contains coefficients for corresponding reactions [17].

When MicroMega is provided with a set of reactions (see left part of Fig. 3) under the shape of a stoichiometric matrix, it will produce a multi-agent system (see right part of Fig. 3), containing quantitative elements (including reaction intermediates) and

functional elements. A second matrix containing data of regulation can be loaded to help the system during the learning of interactions.

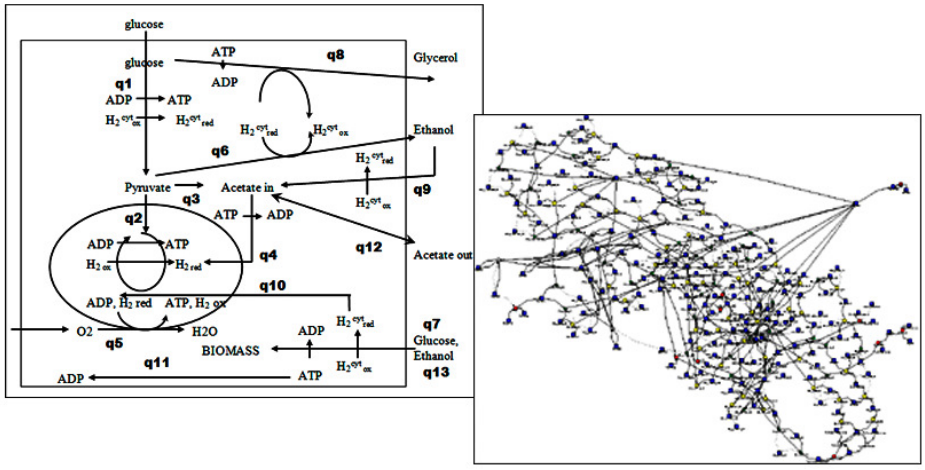

Fig. 3. Simplified yeast glycolysis metabolic pathway (on left) and MicroMega graph visualization associated (on right, legend is the same as in Fig. 2)

For a case study, a simpler model of glycolysis has been produced and implemented in MicroMega. This toy model has no biological validity, since it has been given with arbitrary kinetic values, but shows MicroMega platform about context selection functioning. Figure 4 presents some curves of this virtual experiment with 2 pulses of glucose.

It is of interest to notice that curve profiles are different after first and second pulse. Differences of kinetics are based on different intracellular concentrations of some metabolites. Therefore previous states of the system influence its current behavior.

Another interesting consequence of MicroMega simulation process is visible on Fig. 4 (zoomed area) and concerns emergence of combined kinetics based on oscillation between different states. As it can be seen in A and B, a mean speed results of quick oscillation between two or more very different speeds.

Behavior of the system is the result of given or learned kinetic parameters and of previous states of the system; under these conditions MicroMega goes beyond classical steady state description of metabolic reactions. Each reaction agent locally observes its context of functioning and consequently determines a kinetic. By this way any reaction agent can become aware of transient states triggered by some fluctuating values of element agents. A consequence is visible in Fig. 4 (zoomed box), the appearance of high frequency oscillations in A and B enlightens us of at least two underlying concomitant states. Steady state models could only express such a phenomenon with a mean kinetic.

As it can be seen on the curves related to products of glycolysis CO_2 and acetate (red and yellow ones, Fig. 4) after the second pulse of glucose, the glycolytic activity of the model is more important than after the first glucose pulse. This difference is

due to specific repartition of intracellular metabolites that sets up during the first part of the simulation.

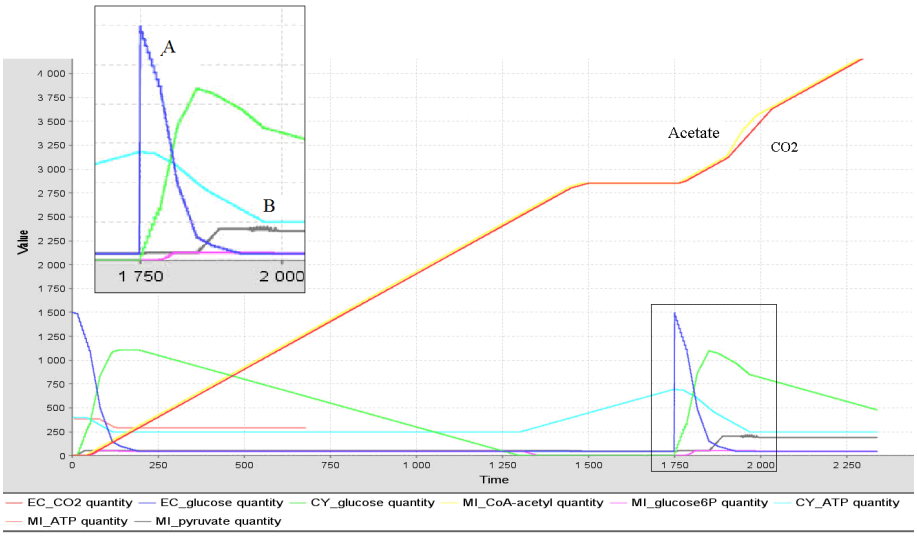

Fig. 4. Simulation results of a simplified model of glycolysis with two glucose pulses (1500 units), first at t=0 and second at t=1750. Extra-cellular glucose is consumed, in a first stage turned into intra-cellular glucose, then metabolized into acetate and CO_2 that are stored up in this closed model. Emergence of some quick kinetic oscillations in A (glucose consumption) and B (mi_pyruvate) after a second pulse of substrate can be seen in the zoomed area

These results show that simulated system even though in strict nominal functioning, has a contextual nominal adaptive behavior.

These results show that the simulated system even though in strict nominal functioning, has a contextual nominal adaptive behavior. We could qualified this observed adaptation as “weak” in the sense that neither the system nor one of its part does actually learn something but just acts contextually and according to its changing internal states. Concentrations of many intracellular metabolites and enzyme cofactors provide the model with a kind of memory of previous states that will modify future behavior related to environmental changes.

5. Conclusion

Due to the complexity of nowadays systems, engineers are no longer able to know everything in order to implement them or to fully control them. Although multi-agent systems are a recognized paradigm for implementing complex systems, engineering them is also a complex task. New ways of engineering complexity are thus required and the aim of this paper was to present a model for making such nonlinear complex systems self-build in order to lighten engineers’ workload. Self-organization driven by cooperation was chosen to enable not only an autonomous evolution of the system

organization but also an autonomous adjustment of the behavior of its agents. The four-layer agent model proposed separates the "basic" behavior of agents composing an adaptive MAS from the behavior that enables them to self-tune, self-reorganize and evolve. Application of this agent model to an application related to biology is then detailed and some preliminary results are given.

Focusing on a specific domain such as biology, biologists rely on models to understand natural phenomena or to discover new laws through less expensive *in silico* experiments. Openness and adaptation are two required features in the modeling process, with their inherent potential MAS are becoming a promising answer toward automatic modeling. In this sense, we applied the proposed framework for modeling the intracellular functioning of S. Cerevisiae yeast. Although the model obtained is still incomplete, preliminary results show that it exhibits adaptation abilities when disrupted. Among the few multi-agent approaches that exist for modeling biological problems, fewer try to deal with dynamical phenomena and disruptions. For example, [28] uses features of oRis, a dynamic language, for disturbing the model of the MAPK while the simulation is running. Dynamic aspects are then handled however the model does not self-build. In [32], a combination of top-down and bottom-up approaches for modeling biological networks rests on holonic MAS. Reactions (transformation, transport or binding process) are viewed as interactions between two holons and several rules. Each holon can manage its rate and stoichiometry by inferring from the rule engine. However dynamic configuration is possible at run-time to get closer to real situations: user can add or remove substrates or products, change rules or regulate the best rate for a reaction and this latter can also be learnt using neural networks or genetic algorithms. So, dynamic changes in the model are endured, however concluding that the model self-builds is difficult due to the lack of details about how learning is done at the level of holons.

From a pure modeling point of view, MicroMega approach of pathway modeling is generic and not restricted to glycolysis: any cellular process that can be described as regulated production/consumption mechanisms is potentially transposable into a set of element and reaction agents. Nevertheless, offering multi-state and transient possibilities using contextual functioning has an important drawback: the list of contexts a reaction agent deals with is rather more difficult to visualize and analyze than global differential equations.

Finally this article wants to claim that this self-building ability is the only answer for dealing with complexity, notably in biological modeling, and MicroMega aims at sustaining this argument. It has been designed in such a way that building and refining complex models will be greatly facilitated by a cooperative behavior. Even if some phases of the model are still under development (such as the evolution behaviors of agents), this approach supports the claim that only an emergent approach has chances to permit complex systems to self-build.

6. References

[1] Alon U.: An Introduction to Systems Biology: Design Principles of Biological Circuits, Chapman & Hall (2006)

[2] Amigoni F., Schiaffonati V.: Multiagent-Based Simulation in Biology: A Critical Analysis, In: Model-Based Reasoning in Science, Technology, and Medicine, Springer Verlag, Studies in Computational Biology, 64, Lorenzo Magnani and Ping Li (Eds), 179-191 (2007)

[3] Armano G., Mancosu G., Orro A., Vargiu E.: A Multi-agent System for Protein Secondary Structure Prediction, In: Transactions on Computational Systems Biology III, LNCS 3737, Springer, 14-32 (2005)

[4] Bortolussi L., Dovier A., Fogolari F.: Multi-Agent Simulation of Protein Folding, In: Proc. of the First Workshop on Multi-Agent Systems for Medecine, Computational Biology, and Bioinformatics (MAS*BIOMED'05@AAMAS'05), 91-106 (2005)

[5] Bosse T., Jonker C., Treur J.: Simulation and Analysis of Complex Biological Processes: an Organisation Modelling Perspective, In: 39th Annual Simulation Symposium (2006)

[6] Capera D., Georgé J-P., Gleizes M-P., Glize P.: The AMAS Theory for Complex Problem Solving Based on Self-organizing Cooperative Agents, In: 12th IEEE International Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WETICE), 9-11 June 2003, Linz, IEEE Computer Society, 383-388 (2003)

[7] De Jong H.: Modelling and Regulation of Genetic Regulatory Systems: a Literature Review, In: Journal of Computational Biology, 9, 67-103 (2002)

[8] De Wolf T., Samaey G., Holvoet T.: Engineering Self-Organising Emergent Systems With Simulation-Based Scientific Analysis, In: Proceedings of the Third International Workshop on Engineering Self-Organising Applications, Brueckner S., Di Marzo Serugendo G., Hales D., Zambonelli F. (Eds), 146-160, Utrecht (2005)

[9] Di Marzo Serugendo G., Gleizes M-P., Karageorgos A.: Self-Organization in Multi-Agent Systems, In: The Knowledge Engineering Review, Cambridge University Press, Simon Parsons (Eds), 20(2), 165-189 (2005)

[10] D'Inverno M., Saunders R.: Agent-based Modelling of Stem Cell Organisation in a Niche, In: Engineering Self-Organising Systems: Methodologies and Applications, Springer-Verlag, Brueckner S., Di Marzo Serugendo G., Karageorgos A., Nagpal R. (Eds), LNCS 3464, Springer, 52-68 (2005)

[11] Efroni S., Harel D., Cohen I. R.: Toward Rigorous Comprehension of Biological Complexity: Modeling, Execution, and Visualization of Thymic T-Cell Maturation, In: Genome Research, 13, 2485-2497 (2003)

[12] Emonet T., Macal C., North M., Wickersham C., Cluzel P.: AgentCell: a Digital Single-cell Assay for Bacterial Chemotaxis, Bioinformatics Advance Access, In: Bioinformatics, 21, 2714-2721 (2005)

[13] Ferber F., Gutknecht O.: A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems, In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98), 128-135, Paris, France (1998)

[14] Fisher J., Henziger T.: Executable Cell Biology, In: Nature Biotechnology, 25(11), November, 1239-1249 (2007)

[15] González P., Cárdenas M., Camacho D., Franyuti A., Rosas O., Lagúnez-Otero J.: Cellulat: an Agent-based Intracellular Signalling Model, In: Biosystems, 68(2-3), 171-185 (2003)

[16] Guo D., Santos E., Singhal A., Santos E., Zhao Q.: Adaptivity Modeling for Complex Adaptive Systems with Application to Biology, In: IEEE International Conference on Systems, Man and Cybernetics, 272-277 (2007)

[17] Heinrich R., Schuster S.: The Regulation of Cellular Systems, Chapman & Hall (1996)

[18] Heylighen F.: The Science of Self-organization and Adaptivity, In: The Encyclopedia of Life Support Systems, EOLSS Publishers Co. Ltd (2001)
[19] Hynne F., Dano S., Sorensen P. G.: Full-scale Model of Glycolysis in Saccharomyces Cerevisiae, In: Biophys. Chem., 94:121-163 (2001)
[20] Jonker C. M., Snoep J. L., Treur J., Westerhoff H. V., Wijngaards W. C. A.: BDI-modelling of Complex Intracellular Dynamics, In: Journal of Theoretical Biology, 251, 1-23 (2008)
[21] Kephart J., Chess D.: The Vision of Autonomic Computing, In: Computer, 36(1), 41-50 (2003)
[22] Khan S., Makkena R., Gillis W., Schmidt C.: A Multi-agent System for the Quantitative Simulation of Biological Networks, In: Proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS'03), Melbourne, ACM, 385-392 (2003)
[23] Kitano, H.: Systems Biology: A Brief Overview, In: Science, 295, 1662-1664 (2002)
[24] Li C., Ge Q.W., Nakata M., Matsuno H., Miyano S.: Modelling and Simulation of Signal Transductions in an Apoptosis Pathway by using Timed Petri Nets, In: J. Biosci. 32,113–127 (2007)
[25] Lints T.: Multiagent Modelling of a Bacterial Cell, a DnaA Titration Model Based Agent Model as an Example, In: Proceedings of the Ninth Symposium on Programming Languages and Software Tools, Tartu, Estonia, Vene V., Meriste M. (Eds.), 82-96 (2005)
[26] Mendes P.: GEPASI: A Software Package for Modelling the Dynamics, Steady States and Control of Biochemical and other Systems, In: Computer Applications in the Biosciences, 9(5), 563-571 (1993)
[27] Merelli E., Armano G., Cannata N., Corradini F., d'Inverno M., Doms A., Lord P., Martin A., Milanesi L., Moller S., Schroeder M., Luck M.: Agents in Bioinformatics, Computational and Systems Biology, In: Briefings in Bioinformatics, 8(1), 45-59 (2006)
[28] Querrec G., Rodin V., Abgrall J.F., Kerdelo S., Tisseau J.: Uses of Multiagents Systems for Simulation of MAPK Pathway, In: Proceedings of the Third IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03), 421-425 (2003)
[29] Ripoll C., Guespin-Michel J., Norris V., Thellier M.: Defining Integrative Biology, In: Complexity, 4 (2), 19-20 (1998)
[30] Troisi A., Wong V., Ratner M.: An Agent-based Approach for Modeling Molecular Self-organization, In: Proceedings of the National Academy of Sciences of the USA (PNAS), 102(2), 255-260 (2005)
[31] Santos E., Guo D., Santos Jr. E., Onesty W.: A Multi-Agent System Environment for Modelling Cell and Tissue Biology, In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, CSREA Press, Arabnia H. R. (Eds), 3-9 (2004)
[32] Shafaei S., Aghaee N.: Biological Network Simulation Using Holonic Multiagent Systems, In: Tenth International Conference on Computer Modeling and Simulation (UKSIM'08), 1-3 April, 617-622 (2008)
[33] Webb K., White T.: Cell Modeling with Reusable Agent-based Formalisms, In: Applied Intelligence, 24(2), 169-181 (2006)