A Biological Inspiration to Support Emergent Behavior in Systems-of-Systems Development

Valdemar Vicente Graciano Neto[1,2], **Elisa Yumi Nakagawa**[2]

[1]Instituto de Informática (INF/UFG), Universidade Federal de Goiás, Goiânia, Brazil

[2]ICMC, University of São Paulo, São Carlos, Brazil

valdemarneto@inf.ufg.br, elisa@icmc.usp.br

Abstract. *Systems-of-Systems (SoS) are engineered with pre-existing software called constituents. SoS development requires a new software engineering endeavor to address the SoS' particularities. Computer science has a tradition in looking for inspiration in biological systems to propose solutions to solve complex problems. Remarkable examples include solutions inspired in ant colonies, swarm of bees, and neural networks. Recent results have been communicated regarding a living cell software-based simulation. In that simulation, the entire life cycle of a living cell is simulated as a function of individual capabilities provided by individual software modules. Those modules simulate the behavior of inner structures of a cell. For cells (or simulated cells) and for SoS, independent parts contribute to deliver a more complex behavior. Then, we investigated how the similarities between SoS and cells structure, and results from cell simulation could foster SoS engineering. This paper presents preliminary results of this research, reporting perceptions we realized between those two research topics, as well as some insights on how both areas could cross-fertilize each other.*

1. Introduction

Systems-of-Systems (SoS) are software-intensive systems constructed with pre-existing systems called constituents. Those constituents, when working together, can deliver complex behaviors that they could not exhibit if working in isolate. Despite the existence of successful proposals to integrate such constituents into an operational SoS [Nakagawa et al. 2014], Software Engineering community has investigated best approaches, practices, methods, models, and processes to suitably address a forthcoming class of civil systems [Graciano Neto et al. 2014]. Some examples include Smart Cities, Smart Buildings, Smart Grids, and all sort of smart systems composed by other systems and delivering innovative complex functionalities. On the other hand, bio-inspired solutions have been explored in the last years for software development initiatives purposes, such as ant colonies, swarm of bees, neural networks [Dorigo and Birattari 2010, Karaboga et al. 2014, Schmidhuber 2015], and even software ecosystems [Santos et al. 2014b, Jansen and Cusumano 2012], which are inspired in real ecosystems.

In this direction, recent results reported a successful simulation of an alive cell. Covert et al. have developed a software-based simulation that integrates several independent software modules to emerge a complex emergent functionality: a simulation of a complete life cycle of an alive cell [Karr et al. 2012, Covert 2014, Matsuoka and Shimizu 2015]. Since simulation is a recurrent paradigm in Systems Engineering guides [Graciano Neto et al. 2014], and suitable models are still necessary to represent the whole

dynamics of large complex smart SoS, we envisioned a possibility of migrating those concepts and results presented by Covert et al. to open a new possible research path regarding SoS development: a bio-inspired SoS development. We understand that this research topic can be plausible, since other successful bio-inspired initiatives are widespread in Computer Science.

This paper presents some insights about a cross-fertilization that could be explored regarding a bio-inspired approach to support SoS development. We expect that these preliminary discussions can open a new research branch in SoS development, enabling prominent results from a multidisciplinary research perspective. Remainder of this paper is structured as follows: Section 2 brings the necessary background to start the discussion, Section 3 presents the first insights we gathered, Section 4 discusses our preliminary findings, and Section 5 concludes the paper, presenting remarks and future research.

2. Foundations

Systems-of-Systems (SoS) are an arrangement of software systems called constituents. Such constituents are pre-existing independent software which contribute with their individual functionalities for accomplishing innovative SoS purposes [Boardman and Sauser 2006]. SoS are engineered to articulate their constituents to offer higher-level functionalities [Nakagawa et al. 2014, Santos et al. 2014a]. Those functionalities emerge as a synergistic result of constituents' interoperability. SoS are required to accomplish missions, a higher goal structured as a set of tasks to be performed by the constituent systems. Constituents might be unaware of their cooperation and information exchanging in order to accomplish it. Each constituent system accomplishes its own individual mission and is able to contribute to the accomplishment of the global mission of the SoS [Silva et al. 2014].

Translating the SoS concept into reality requires a different approach despite the one currently used to engineer systems [Boardman and Sauser 2006]. Part of such additional complexity is due to SoS' inherent characteristics. SoS are distinguished from large monolithic systems by a set of key 'dimensions' postulated by Maier [Maier 1998,Nielsen et al. 2013, Fitzgerald et al. 2012, Andrews et al. 2013, Pérez et al. 2013]: Operational independence of constituents and Managerial independence of constituents, emergent behavior, and evolutionary development processes (evolution), besides geographical distribution as an essential feature [Board 2014]. Besides Maier's dimensions, dynamic architecture, and self-management aspects [Nielsen et al. 2013,Fitzgerald et al. 2012,Andrews et al. 2013, Weyns and Andersson 2013, Batista 2013, Romay et al. 2013] have been recognized as inherent and expected characteristics for software-intensive SoS.

When considering SoS engineering, it is important to highlight that constituents might be monolithic systems, or even other smaller SoS themselves, delivering specific functionalities which contribute to the larger SoS' mission. Thus, under this perspective, SoS can be constituted by subsequently smaller parts, also considered constituents themselves.

Under another perspective, any product of engineering is based on models, including SoS. Independently of the various modeling approaches available, a proper modeling is useful in a range of areas. Appropriate models helps to realize the systems' behavior and reveal the underlying principles of the target real element that is being modeled [Matsuoka

and Shimizu 2015]. With such appropriate models, it is possible to simulate real behaviors, predicting effects, and supporting necessary calibrations, adaptations, and changes which can, in an early stage, avoid errors and losses.

One of the frequent models present in computer science are the bio-inspired models. Among recent initiatives, one is remarkable: a living cell simulation [Karr et al. 2012, Freddolino and Tavazoie 2012, Covert 2014, Matsuoka and Shimizu 2015]. Markus W. Covert, a bioengineer, developed a well-succeeded simulation of a living cell. Authors developed a sort of software models and modules representing, each one, a particular cellular inner process, such as DNA processing, RNA transcription, and cell division, originating a new cell [Karr et al. 2012, Covert 2014]. They developed 28 independent modules which, interacting, offered a complete living cell life cycle as an emergent result. They claim that such interactions might help scientists to understand cause-effect relations among organelles into a real cell.

They selected a quite simple living unicellular bacterium called *Mycoplasma genitalium*, which have only 525 genes against the almost 20 thousand genes a human cell has. Duplication period manifested as an emergent property that results from a complex interaction between distinct replication phases. They developed a comprehensive whole-cell model. Simulations, source code, knowledge database, visualization code, and experimental data are available online[1].

3. A bio-inspired solution for SoS's emergent behavior

Biological analogies are recurrent in computer science. Ant colony [Dorigo and Birattari 2010], swarm of bees [Karaboga et al. 2014], neural networks [Schmidhuber 2015], and genetic algorithms [Karakatic and Podgorelec 2015] are examples of computational solutions whose operating principles are based on existing real biological systems. Thus, given SoS inherent characteristics, and living whole-cell simulation results, features, and open issues, we envisioned some points of intersection between both areas that could benefit each other to join efforts toward unified results.

Briefly, a cell can be individually considered as an SoS. The interaction among their organelles displays behaviors. Together, such interactions and individual behaviors culminate in cellular function and dynamics. In this sense, a biological analogy can help to realize how emerging behaviors occur and how this spontaneous phenomenon can be represented and raised as a function of individual constituents and their individual capabilities. Thus, it can be possible to migrate such interaction strategies among constituents to better 1) represent interactions among constituents, 2) realize the impact of constituents' individual influences over the whole behavior, and 3) engineer such emergent behaviors in SoS.

In fact, the whole human body can be seen as a huge and complex SoS, where its constituents are themselves, other complex SoS. Human body SoS is divided in organs. Organs interact among them to deliver an emergent behavior: your life. Organs are themselves other SoS, since they are constituted by specific tissues. Tissues are also SoS, since they are a composition of cells. And cells themselves are minor SoS constituted by organelles, such as Golgi complex, plasma membrane, and mitochondria. Organelles

[1] `https://simtk.org/home/wholecell`

interaction deliver the whole-cell life cycle as an emergent behavior. Such as automotive systems, which are composed by smaller parts which interact to deliver the car operation, biological systems can be perceived under an SoS perspective.

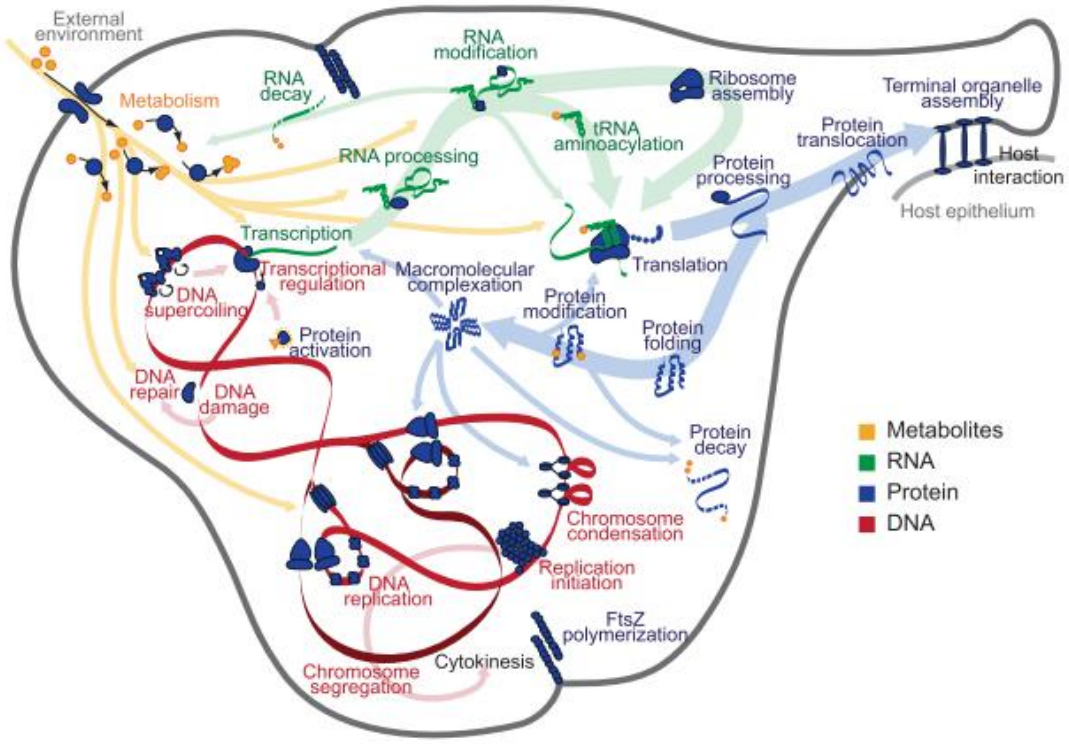

Figure 1. A model of a Cell separated by organelles [Karr et al. 2012].

Figure 1 presents a model of a living cell. That figure helps to realize a cell under an SoS perspective. Under this perspective, many small distinct structures have independent work, such as RNA processing, RNA decay, RNA modification, Ribosome assembly, and Protein processing. Those structures play, each one, a constituent role, delivering individual capabilities which, together, contribute to the whole cell life cycle, until the cell division.

We could establish a comparison between SoS characteristics and a cell (as an alive SoS) characteristics. That comparison is available in Table 1. To establish such comparison, we based it on the criteria available in a conceptual model for SoS available in literature that lists all of the essential characteristics an SoS should exhibit [Benites Goncalves et al. 2014]. Under this perspective, organelles work as constituents of the Cell SoS. The Global Mission could be growing, reproducing, or even the cell entire life cycle. About software dominance, the simulated cell is based on several software modules. Each organelle offers independent behavior, what characterizes operational independence of their constituents. Each organelle is represented by one or more software modules, and those modules are required to evolve to approach their configuration parameters to reliably represent a real cell. Thus, the simulated cell requires evolutionary development. The “execution” can be considered the emergent behavior. Geographic distribution is a relative concept, and could be considered in the following way: if the

SoS	Cell
Constituents	Organelles
Global mission	Growing and reproducing
Software dominance	The simulated cell is based on 28 distinct software modules
Operational independence	Each organelle or inner structure (simulated as a software module) has independent behavior and operation
Evolutionary development	Improvements and calibrations in the parameters which rules the organelles or cell structures operation require constant evolution of modules, featuring evolutionary development
Emergent behavior	Cell's metabolism and life cycle are the emergent behaviors
Geographical distribution	The real cell has a minimum distance among their organelles. Simulation dispenses it.
Connectivity (Interoperability)	Also known as interoperability, organelles interact to deliver emergent behavior.

Table 1. Comparison between SoS and cells under Benites et. al perspective [Benites Goncalves et al. 2014].

structures interoperating are "physically separated", they deliver geographic distribution. Since organelles are physically distinct entities, the real cell offers such distribution, and the software modules simulate it. Finally, such organelles interact and are connected through their inner processes to deliver the emergent behavior. Then, they offer interoperability. And, for the set of characteristics they exhibit, a cell can actually be considered a biological SoS.

4. Discussion

We glimpsed both research areas as potential to benefit each other. SoS community can benefit from cell simulation, and cell simulation can benefit from SoS approaches. Cell simulation approach delivers techniques and source code to simulate a living cell. It can be adapted to engineer suitable simulation environments for the forthcoming smart SoS, using such prominent results regarding emergent behaviors as a composition of individual capabilities to an effective emergent behavior in SoS (a still open issue). Conversely, an SoS development approach could benefit whole-cell simulation, providing a lighter approach of simulation for it. Well-succeeded SoS simulation approaches, such as Agent-based SoS simulation approaches [Pavon et al. 2011], can be migrated for cell simulation, improving those results and visualization. This kind of adaptation can bring advantages for the simulator maintainability and development, since it uses an approach already well-succeeded for SoS issues.

Simulation is, in fact, a recurrent and traditional paradigm and a genuine step in Systems and SoS Engineering approaches [Graciano Neto et al. 2014]. It brings important advantages, such as an early perception of errors, defects, and problems that need to be corrected in specification level before integration step being conducted; and better consistency, verification and validation. Simulation may early evidence phenomena if

simulation has been faithfully modeled, providing data not previously obtained.

Simulation make intensive use of models. Conversely, models give support for verification and validation processes, through the use of simulation or other automated techniques. Some types of models are called *runtime models*, enables simulation of the SoS operation via model execution. Agents are frequently mentioned as a complete and mature technology to perform SoS simulation, and can be considered the state-of-art for SoS simulation. SiCoSSyS approach is a sample approach to engineer SoS based on agents simulation [Pavon et al. 2011]. In fact, agents could be used as a lightweight approach to represent each of the modules listed by Covert et al., inovating in cell simulation, and providing the cellular structures capabilities individually, and the whole life cycle as an emergent behavior.

On the other hand, the realization of organelles interaction to deliver its life cycle as an emergent behavior could benefit SoS community. Cell structure interaction patterns could be investigated and reproduced in SoS development, achieving an important requirement imposed by SoS engineering initiatives: addressing of emergent behavior. In cells, emergent behavior really *emerge* as a result of the synergy between the parts which interact. However, in SoS, the emergence is deliberately and intentionally designed [Boardman and Sauser 2006]. Boardmand and Sauser mention that emergent behavior dare not be restricted to what can be foreseen or deliberately designed. They claim that an SoS must be richer in emergence, and that a challenge for the SoS designer is to know, or learn how, as the SoS progresses through its series of stable states, to create a climate in which emergence can flourish, and an agility to quickly detect and destroy unintended behaviors, much like the human body deals with unwanted invasions [Boardman and Sauser 2006]. Following this trend, investigating such recent results in cell simulation could benefit SoS engineering to provide and accomplish some highlighted desired challenging characteristic.

Regarding related work, Boardman and Sauser [Boardman and Sauser 2006] also establish some parallels between the concept of software systems and biological systems, such as the similarities between the human brain and their neurons as constituents, and the colonial behavior shared by ants. However, authors do not discuss how those similarities could be explored to benefit both areas, SoS engineering/simulation and biological simulations.

Indeed, e-Science community[2] has approached some efforts to connect biology, astronomy, and other science efforts to computational solutions. e-Science consists of science that is carried out in highly distributed environment, using computationally intensive solutions [Taylor et al. 2014]. We did not find any evidences of an SoS approach for cell simulation or a cell simulation approach for SoS conception being communicated, as in SoS community, as in e-Science community.

5. Final Remarks

This paper presented some insights for a cross-fertilization between living cell simulation and SoS development areas. We offer this initial results as an starting point of investigation for both e-Science and SoS communities. We expect that this first effort can serve

[2] `http://www.nesc.ac.uk/`

as an enlightening result, which effectively establish an existent parallel among those research areas. Bio-inspired solutions are recurrent in Computer Science. We expect that our envisioned approach, despite the glimpsed possibilities for cell simulation advances itself, could contribute to improve techniques for effectively providing emergent behavior for SoS. Such SoS intrinsic feature is still an open issue. We wish to provide it for SoS not as a mechanical and nondynamic property, but as a genuine and adaptive condition.

6. Acknowledgements

Authors thank Goiás Research Foundation (FAPEG) and FAPESP by financial support under grant number 09/2013, ID 2013.009.97100854, and grant number 2014/02244-7, respectively.

References

Andrews, Z., Payne, R., Romanovsky, A., Didier, A., and Mota, A. (2013). Model-based development of fault tolerant systems of systems. In *SysCon*, pages 356–363.

Batista, T. (2013). Challenges for SoS Architecture Description. In *1st SESoS*, SESoS '13, pages 35–37, New York, NY, USA. ACM.

Benites Goncalves, M., Cavalcante, E., Batista, T., Oquendo, F., and Yumi Nakagawa, E. (2014). Towards a conceptual model for software-intensive system-of-systems. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 1605–1610.

Board, B. E. (2014). The guide to the systems engineering body of knowledge (sebok). Technical report.

Boardman, J. and Sauser, B. (2006). System of systems - the meaning of of. In *System of Systems Engineering, 2006 IEEE/SMC International Conference on*, pages 6 pp.–.

Covert, M. W. (2014). Simulating a Living Cell. *Scientific American*, (310):44–51.

Dorigo, M. and Birattari, M. (2010). Ant colony optimization. In Sammut, C. and Webb, G., editors, *Encyclopedia of Machine Learning*, pages 36–39. Springer US.

Fitzgerald, J., Bryans, J., and Payne, R. (2012). A formal model-based approach to engineering systems-of-systems. In Camarinha-Matos, L. M., Xu, L., and Afsarmanesh, H., editors, *Collaborative Networks in the Internet of Services*, volume 380 of *IFIP AICT*, pages 53–62. Springer Berlin Heidelberg.

Freddolino, P. and Tavazoie, S. (2012). The dawn of virtual cell biology. *Cell*, 150(2):248 – 250.

Graciano Neto, V. V., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. In *SESoS*, ECSAW '14, pages 22:1–22:8, Vienna, Austria. ACM.

Jansen, S. and Cusumano, M. (2012). M.: Defining software ecosystems: A survey of software platforms and business network governance. In *Proceedings of the international Workshop on Software Ecosystems*, pages 41–58.

Karaboga, D., Gorkemli, B., Ozturk, C., and Karaboga, N. (2014). A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57.

Karakatic, S. and Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, 27(0):519 – 532.

Karr, J. R., Sanghvi, J. C., Macklin, D. N., Gutschow, M. V., Jacobs, J. M., Bolival, B., Assad-Garcia, N., Glass, J. I., and Covert, M. W. (2012). A Whole-Cell Computational Model Predicts Phenotype from Genotype. *Cell*, 150(2):389–401.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Matsuoka, Y. and Shimizu, K. (2015). Current status and future perspectives of kinetic modeling for the cell metabolism with incorporation of the metabolic regulation mechanism. *Bioresources and Bioprocessing*, 2(1):4.

Nakagawa, E. Y., Capilla, R., Díaz, F. J., and Oquendo, F. (2014). Towards the dynamic evolution of context-based systems-of-systems. In *WDES 2014*, pages 45–52, Maceió, Brazil.

Nielsen, C. B., Larsen, P. G., Fitzgerald, J., Woodcock, J., and Peleska, J. (2013). Model-based engineering of systems of systems. Technical report. Available from http://www.compass-research.eu/resources/sos.pdf.

Pavon, J., Gomez-Sanz, J., and Paredes, A. (2011). The sicossys approach to sos engineering. In *SoSE 2011*, pages 179–184, Irvine, CA, USA.

Pérez, J., Díaz, J., Garbajosa, J., Yagüe, A., Gonzalez, E., and Lopez-Perea, M. (2013). Large-scale smart grids as system of systems. In *SESoS 2013*, pages 38–42, Montpellier, France.

Romay, M. P., Cuesta, C. E., and Fernández-Sanz, L. (2013). On self-adaptation in systems-of-systems. In *1st SESoS*, SESoS '13, pages 29–34, New York, NY, USA. ACM.

Santos, D. S., Oliveira, B., Guessi, M., Oquendo, F., Delamaro, M., and Nakagawa, E. Y. (2014a). Towards the evaluation of system-of-systems software architectures. In *WDES 2014*, pages 53–57, Maceió, Brazil.

Santos, R., Gonçalves, M., Nakagawa, E. Y., and Werner, C. (2014b). On the relations between systems-of-systems and software ecosystems. In *WDES 2014*, pages 58–62, Maceió, Brazil.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61(0):85 – 117.

Silva, E., Cavalcante, E., Batista, T., Oquendo, F., Delicato, F. C., and Pires, P. F. (2014). On the characterization of missions of systems-of-systems. In *Proc. of the ECSAW*, pages 26:1–26:8, Vienna, Austria. ACM.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. (2014). *Workflows for e-Science: Scientific Workflows for Grids*. Springer Publishing Company, Incorporated.

Weyns, D. and Andersson, J. (2013). On the challenges of self-adaptation in systems of systems. In *1st SESoS*, SESoS '13, pages 47–51, New York, NY, USA. ACM.