In virtuo modelling for biologists: How to design graphical interface for the RéISCOP simulation platform?

Guillaume Longelin Péron, Gireg Desmeulles

In virtuo modelling for biologists: How to design graphical interface for the RéISCOP simulation platform?

Guillaume Longelin Péron[1], Gireg Desmeulles[1]

[1]Lab-STICC, UMR 6285 CNRS, UEB/ENIB/CERV, France

Abstract

This article presents a future component of the RéISCOP simulation platform. RéISCOP provides an interaction-based meta-model to build models for simulations. Currently, model construction needs to be done by a programmer. This is a problem for the biologist who cannot construct models by himself. We propose to include a graphical interactive interface to allow biologists to build models in RéISCOP with help of *in virtuo* experiments.

1 Introduction

In 2013, the Royal Swedish Academy of Sciences decided to award the Nobel Prize in Chemistry to Martin Karplus, Michael Levitt and Arieh Warshel for the development of multiscale models for complex chemical systems. These works stress the point that mathematical modelling and computer simulation of complex phenomena are more and more central to the field of fundamental research applied to living systems.

In this context, Virtual Reality (VR) may become essential to study complex systems such as biological systems. VR places the user at the heart of a virtual laboratory, so that he can use tools which share similarities with experimental science methods: the user (*i.e.* the biologist) can therefore investigate the virtual biological world using various methods such as 3D visualisation and interactions, numerical methods, etc. We usually call this kind of investigations "*in virtuo* experiments" for its similarities with the expressions *in vivo* and *in vitro* [1]. *In virtuo* experiment is a subset of computer simulations studies called *in silico*. It replaces user on experiment context by immerse him in simulation environment, for example a biology laboratory (Figure 1). He can use VR elements to interact with environment. This places *in virtuo* experiments at the intersection of biology and VR. Humans are then directly involved in the *in virtuo* experimentations of the numerical models within the virtual environment. RéISCOP is both, a framework and a meta model that have been designed to enable the *in virtuo* experiments. Recently, it has been rewritten into a version 2.0 that includes meta-model, interactive simulation engine and reaction/diffusion and bacteria models. The GUI(Graphical User

Interface) that is necessary to enable the *in virtuo* experiments has not been developed yet. This position paper addresses our work on the design and implementation of an efficient graphical user interface. First, we present RéISCOP 2.0. Then we review the various multi-agent platforms by focusing on the proposed interfaces; Finally we conclude about the scientific challenges that we must overcome to imagine an relevant graphical user interface for *in virtuo* experience.

Figure 1: Virtual biology laboratory on RéICOP simulation. Bacteria development simulation take place on this environment. We can see it by zooming in on microscope (Figure 6)

2 RéISCOP 2.0

RéISCOP is a meta-model and a C# written simulation platform using Unity3D as basement for graphical user interface. Its name is an acronym of following sentence that gets main concepts of meta-model together – **Re**ification of **I**nteractions, **S**ystems, **C**onstituents, **O**rganizations and **P**henomenons. We can refer to [2] for a description of the former 1.0 version to understand all meta-model notions. The 2.0 version has not been published yet but we can present some points[1].

[1]Documentation can be found at: http://www.cerv.fr/ReISCOP/doxygene/index.html

2.1 Meta-model

The RéSCOP meta-model provides a means to create an interaction-based simulation that can be seen as a dual method for individual based model(IBM) simulation (Figure 2).

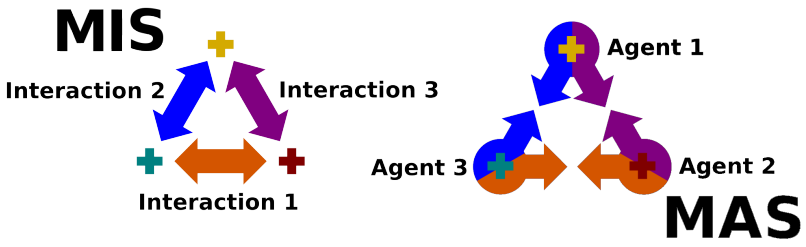


Figure 2: In multi-agent systems(MAS), individuals are explicit and interactions implicit whereas in multi-interaction systems(MIS), interactions are reified and individuals become implicit.

The meta-model can be used to implement models or simply to discuss and to design on paper. We briefly describe here the main concepts of the modelling formalism:

Constituent: *Constituents* are variables, parameters or quantities involved in the models. At each moment, they represent the current state of the model. *Constituent* may model a concentration, a 3D shape, a position or a diffusion coefficient. In our graphical notation, constituent is represented by "+".

Interaction: *Interactions* are the processes that modify the states of the *constituents* over time. An *interaction* points towards a constant set of *constituents* with read or write access, at every step of simulation. *Interaction* may model a chemical reaction, a mechanical collision, a chemical diffusion. Interaction are represented by a multi-head arrow. Each arrow head points out a constituant.

Phenomenon: *Phenomena* are in charge of producing *interactions* during the simulation. A *phenomenon* focuses on the states of *constituents* (only those which are known by its *organization*). If it detects that the required conditions are satisfied, then it produces a new *interaction*. In this way, depending on its type, each *interaction* belongs to a *phenomenon*. Link between a phenomenon and its *interactions* is represented by a dotted line.

Organization: An *organization* represents the dynamics of a system part. It is composed of *phenomena* which themselves are composed of *interactions*.

In addition, the *organization* handles a set of *constituents* which represent the static part of the system. The *organization* role is to maintain the consistency of that set of *constituents* over time. Finally, an *organization* can be composed of sub-*organizations* corresponding to sub-systems.

System: *Constituents, interactions, phenomena* and *organizations* are used to model and populate our virtual worlds with autonomous *systems* (Figure 3).

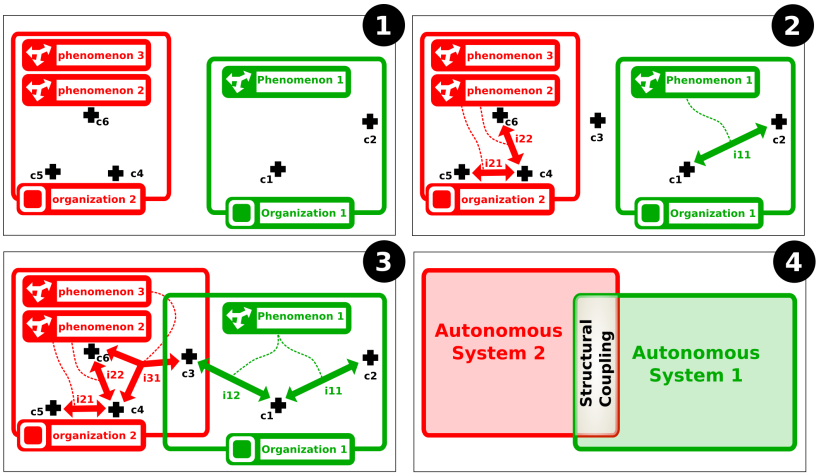


Figure 3: RÉISCOP dynamics. 1: Organizations 1 and 2 have phenomena and know constituents. 2: If conditions are satisfied, phenomena instanciate interactions; a new constituent (c3) is created. 3: Organizations adapt their boundaries according to their internal rules; phenomena detect new conditions for creating interactions; new interactions are created. 4: We have here two autonomous systems coupled by a part of their structure. A concrete model can be seen on figure 5

Constituents represent the static part of *systems*. We call *structure* the whole set of *constituents*. *Interactions, phenomenon and organizations* define the dynamics. By doing so, we focus on the dynamics rather than on the statics. We organise the dynamics instead of structuring the *system* state, assuming that the method is less reductionist and more suited to the study of complex *systems*. Furthermore, multi-interaction *systems* are connected by structural coupling [3] (Figure 4). This differs from the usual component based approach which uses input/output connections between components. Here, *systems* perceive one another through perturbations on their own structure. The autonomous nature of *systems* is thus consolidated.

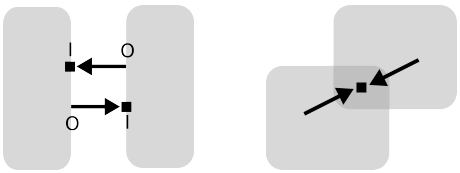


Figure 4: Classical input/output interface between two systems (left) and structural coupling between two systems (right)

2.2 Platform

RéISCOP software implements the meta model and can build and simulate models (Figure 5) described with XML files and C# delegate functions. Currently, simulation engine work properly(Figure 6). We use it to simulate bacterium colonies development in the context food security project however it is not an "*in virtuo*" experimentation. Indeed, Models have to be built by writing XML description file. Model editor is missing from 3D RéISCOP simulator. RéISCOP 1.0 had two distinct interfaces: a 3D simulator and a 2D editor. The solution will be to include model building on simulator. For this, we need to study graphical user interface of actual multi-agent platforms.

3 Agent-based simulation platform

We have reviewed actual multi-agent simulation platforms to study their graphical user interfaces for models construction. We have reviewed this platforms because their models paradigms (multi-agent system) are close to RéISCOP meta-model paradigm (multi-interaction system). We focus on three factors:
Simplicity: How easy is it to build model? Can a biologist build model alone?
Expressiveness: Is it possible to build complex model?
Interactivity: Do model and simulation are easy to manipulate?
This review shows us different examples of agent-based simulation building. We have examined three platforms that are frequently used for simulation modelling with multi-agent systems: Repast, NetLogo and GAMA; and finally, two platforms that use graphical elements to construct model for simulation: AgentSheet, SeSam.

Repast: Repast is a Java framework for agent-based simulation[5]. It allows the creation of an agent-base simulation using the Java and it includes a library of object to create, run, display and collect data from agent-based simulation. Repast models may only be build with Java programming language. This is an obstacle for those who want to build their models alone but who are unable to program (as is the case for many biologists). Although Repast framework

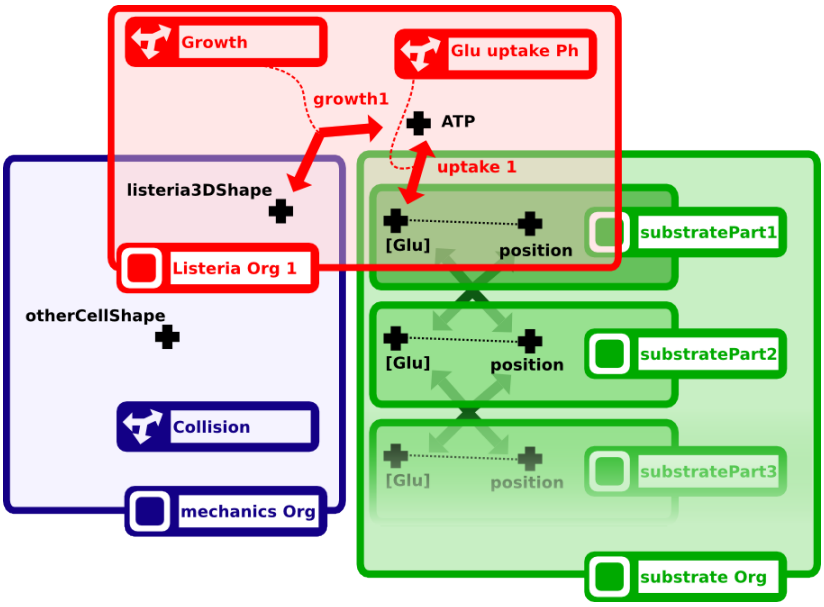


Figure 5: This figure shows a RéISCOP model containing 3 sub-systems : (red) a Listeria that is coupled with (green) a part of discretized chemical substrate and (bleu) a mechanical organization handling the possible interactions between the different cell shapes. For example, *Listeria Org 1* is an organization that is responsible for the cell is coupled with correct constituent *Glu* and cell *position* correspond with its shape location. If it is necessary, *Collision* phenomenon instantiate collision interactions between shapes. *Glu* constituent is a real number that represents glucose concentration in a mesh of discretized substrate. Every simulation step, *growth1* interaction decreases *ATP* value and increases shape size.

allow to write varied and complex model with the help of Java programming language and framework tools.

Netlogo: NetLogo is a multi-agent programming language and modelling environment[6]. This platform is designed for research and education for a large range of disciplines. The NetLogo language is a logo language variation. With this language, *turtles* represent agents during the simulation. They are located agents that move on spacial agents: *patches*. It hopes teach multi-agent simulation programming. Build a model on NetLogo is more simple than on Repast but models are more limited. The platform uses a programming language that may be a problem to model building for non-programmer. The platform offers parametric simulation interface to edit simulation parameters.

GAMA: GAMA is a modelling and simulation environment for building spatially agent-based simulation [7]. It is originally used for simulate geographic

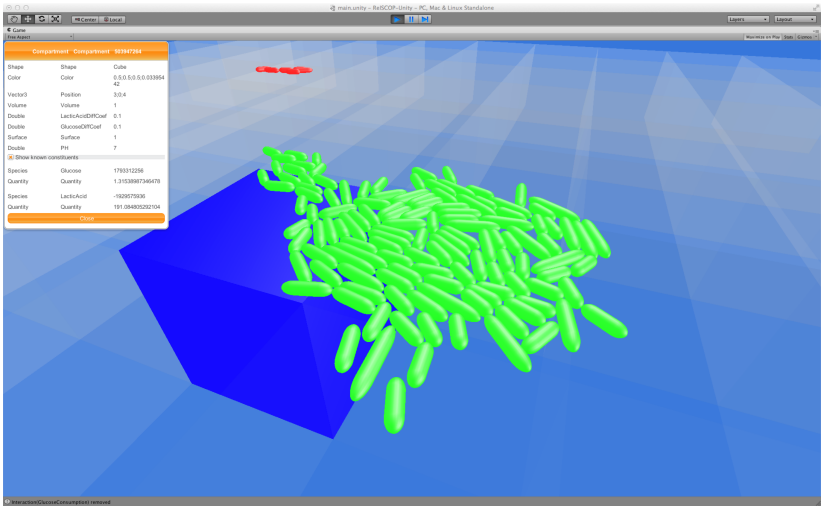
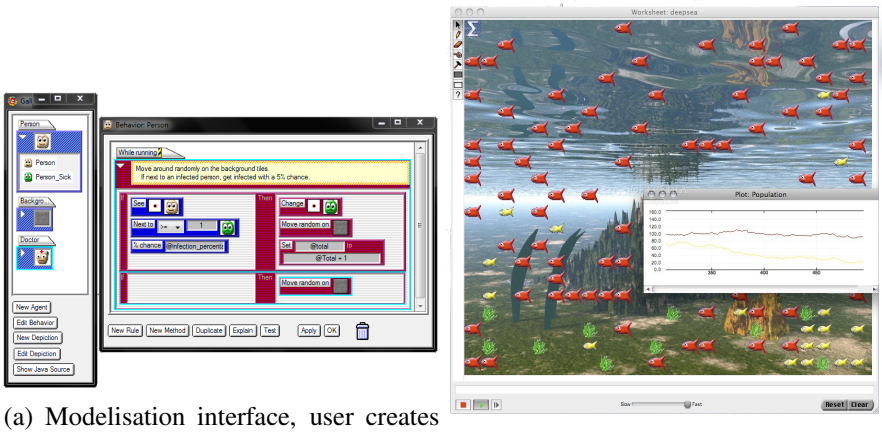


Figure 6: RÉISCOP simulation interface. We see two bacteria colonies: green bacteria in foreground are listeria and red bacteria in background are carnobacterium. Bacterial colonies live on an agar-based growth medium on Petri dish.)

information system but must be extend to other domains. GAMA makes available an agent-oriented language GAML(GAma modelling Language). It is more simple to build a model with this language than Java because it is structured with multi-agent system elements like agents, agent actions, agent reflex, etc. GAMA supplies a graphical editor to simply build simulation with minimum code writing. But this interface is only used during first part of a project to structure simulation elements (agents, theirs actions, theirs reflexes) after this, it must be exported to GAML and user writes agent actions code. GAMA is easier to use than Repast because it has its own, specifically developed, simulation language. But platform has important coding part just as Repast and NetLogo.

AgentSheets: AgentSheets is a platform that provides graphical tools to construct simulation model[8]. Non-programmer can simply use AgentSheets to build model. AgentSheets consists of two main interfaces, a interface to build models and an other to launch simulations. In modelling interface, user can add new agents and specify their behaviours. Behaviours are formed by condition-action statements (Figure 7a). In simulation interface, user can instantiate agent in environment called *agentsheets* (Figure 7b). An AgentSheets function allows user to select an agent in simulation interface and see verified conditions and realised actions in modelling interface. This functionality can help user to understand relationship between an agent in modelling interface and agents



(a) Modelisation interface, user creates agents and specifies theirs behaviors with condition/action components.

(b) Simulation interface, agents can be add in environment.

Figure 7: AgentSheets interfaces

with the same behaviour in simulation interface. This platform is limited for complex system building. Simulations look like cellular automaton. This simulation platform can not be used in biologic simulation because of limited possibilities.

SeSam: SeSam is a platform for modelling and experiment with agent-based simulation[9]. SeSam allows to build agent behaviour with UML-like activity diagrams. Next, user can use a menu to select actions that will be realised during states and conditions between states. User have functions list that can be used for an action. He can specify function parameters with other functions and so on. Edition menus are not user friendly and it is difficult with first look to understand action building. An other difficulty is to edit existing models.

On one hand, there are three first platforms frequently use to build multi-agent simulations that are expressive enough to be used for biological simulation. But building models with this platforms is not easy for non-programmers. On the other hand, two other platforms allow user to construct model with graphical tools. First, SeSAm simplifies organization agent behaviours but edit menu makes action specifications harder. Next, AgentSheets makes successfully easier model building with graphical elements. Any platform has a clear and interactive model representation and allows to create complex model (Figure 8).

	Repast	NetLogo	GAMA	AgentSheets	SeSam
Simplicity	- - -	- -	- -	++	-
Expressiveness	++	+	++	- -	+
Interactivity	- -	-	-	+	+

Figure 8: Summary of platforms studied factors.

4 Graphical user interface to design dynamical system

We have not found an existing method to manipulate complex meta-model with graphical interface. There is only AgentSheets that includes a helpful but limited graphical user interface to build model. At present, usage of RéISCOP meta-model graphical representation is only theoretical. We already use it to describe model with XML files, but we would like to employ it through *in virtuo* experiments. RéISCOP meta model implements concepts like object and class although a biologist does not know oriented object concepts. We would like to clarify concept of instantiation for non-programmer. For example, a phenomenon can apply a physical collision with instantiation of interaction between two constituents. Currently, static model handle dynamic system design. We want to give the possibility to construct model dynamically. We would like to provide non-programmers with Unity3d modelling interface that will be included in RéISCOP simulation platform.

5 Conclusion

Our objective is to include a graphical interface for model building in RéISCOP simulator. User must easily interact with it. Concepts of abstraction need to be clearly understood by model builder. Reviewed platforms do not give us tools to manipulate models dynamically. That's why we want add to RéISCOP an interactive models building interface. We need to create interaction metaphors to build models. This metaphors could be clear for non-programmers. But we have to keep a balance between simplicity to build models and complexity of made models. Model and simulation representations are not in same graphical and concept spaces. The main challenge will be that simulations and constructions spaces work together. In order to do this, a method will be to display graphical connections between elements on simulation interface and items on modelling interface.
In the *in virtuo* experiment context, we plan to interface RéISCOP platform with virtual reality equipment to increase user immersion. It might help to design models.

References

[1] J. Tisseau, "Réalité virtuelle: autonomie in virtuo," *Habilitation á Diriger des Recherches*, 2001.

[2] G. Desmeulles, S. Bonneaud, P. Redou, V. Rodin, and J. Tisseau, "In virtuo experiments based on the multi-interaction system framework: The réiscop meta-model," *Computer Modeling in Engineering and Sciences (CMES)*, vol. 47, no. 3, pp. 299–329, 2009.

[3] P. Dumouchel, P. Bourgine, and F. J. VARELA, *Autonomie et connaissance*. Seuil, 1989.

[4] S. F. Railsback, S. L. Lytinen, and S. K. Jackson, "Agent-based simulation platforms: Review and development recommendations," *Simulation*, vol. 82, pp. 609–623, 2006.

[5] N. Collier, "Repast : An extensible framework for agent simulation," *The University of Chicago's Social Science Research*, vol. 36, pp. 371–375, 2003.

[6] S. Tisue and U. Wilensky, "Netlogo: A simple environment for modeling complexity," in *International Conference on Complex Systems*, pp. 16–21, 2004.

[7] A. Grignard, P. Taillandier, B. Gaudou, D. A. Vo, Q. Huynh, and A. Drogoul, "Gama 1 . 6 : Advancing the art of complex agent-based modeling and simulation," in *PRIMA 2013: Principles and Practice of Multi-Agent Systems*, pp. 117–131, 2014.

[8] A. Repenning and T. Sumner, "Agentsheets: A medium for creating domain-oriented visual languages," *Computer*, vol. 28, no. March, pp. 17–25, 1995.

[9] F. Klügl, R. Herrler, and C. Oechslein, "From simulated to real environments: How to use sesam for software development," in *Multiagent System Technologies*, pp. 13–24, 2003.