A Reverse Engineering Form for Multi Agent Systems

François Gaillard, Yoann Kubera, Philippe Mathieu, Sébastien Picault

Equipe Systèmes Multi-Agents et Comportements
Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille
UMR CNRS USTL 8022
59655 Villeneuve d'Ascq cédex – FRANCE
www2.lifl.fr/SMAC/
mail: firstname.lastname@lifl.fr

Abstract. The usual way to design a simulation of a given phenomenon is to first build a model and then to implement it. The study of the simulation and its outcomes tells if the model is adequate and can explain the phenomenon. In this paper, we reverse this process by building a browser in simulations space: we study an automatically built simulation to understand its underlying model and explain the phenomenon we obtain. This paper deals with automated construction of models and their implementations from an ontology, consisting of generic interactions that can be assigned to families of agents. Thanks to the measurement tools that we define, we can automatically qualify characteristics of our simulations and their underlying models. Finally, we offer tools for processing and simplifying found or existing models: these allow an iterative construction of new models by involving the user in their assessment. This simulation space browser is called LEIA for "LEIA lets you Explore Interactions for your Agents".

1 Introduction

Agent-based simulations have taken a preponderant place in life simulation tools, in domains as different as the movie industry, video games, biology, etc. These simulations establish a link between experts of their domain and experts in computer science[12]: this multidisciplinary aspect gave birth to a whole range of frameworks more or less related to the simulated domains.

Many of these platforms like Swarm[14], Madkit[7] or Magic[16] allow considerable freedom to the designer to create agents, the behaviour of those agents and the environment. All design refinements are possible: reusability, genericity, design patterns, components... Other tools like Netlogo[19] are designed to be used by non computer scientists: they rely on a very simple programming. However, openness and genericity are chosen to the detriment of a clear framework for the design of agent behaviour: there is a risk of mixing the framework code with some specific knowledge to the model in the implementation of agents.

The IODA[1] methodology[13] is based on a clear separation between agents, their behaviour and the actions selecting process. In this methodology, interactions are independently reified from agents which use them. As a result, we can establish libraries of interactions for a particular area and adaptable to different families of agents, increasing the genericity of modelling work.

An implementation achieved through the JEDI engine [12] offers a generic support of the IODA methodology. This genericity is perfectly illustrated within the generator called JEDI Builder[2]: from a model written according to IODA, we can easily obtain a simulation which is executable with the JEDI engine.

We also offer measurement tools in order to describe the characteristics of each model designed with IODA:

– simulation activity;
– characteristics of the environment and its population;
– development of populations;
– evolution of the use of interactions during the simulation.

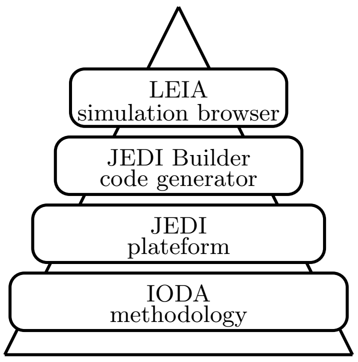

Fig. 1. Hierarchy in the construction of the browser: LEIA is based on automatically generated simulations for the JEDI platform, from a model specified in the methodology IODA

Based on the abstraction provided by IODA and some measurement tools that we set out, we present here a browser in simulations space, i.e. an application which allows the simultaneous display and analysis of several simulations running in parallel. Moreover, to browse the simulations space, we offer tools for processing and simplifying models. These tools enable the iterative design of new models in which the user and measurement tools are the evaluation functions. This browser is called LEIA for "LEIA lets you Explore Interactions for your Agents". LEIA reverses the usual way to design multi agents simulations: we generate a random series of models and their implementation from an already

[1] for : *Interaction Oriented Design of Agent simulations*
[2] www2.lifl.fr/SMAC/projects/ioda/demonstrations/

defined ontology, consisting of generic interactions that we can assign to families of agents. Then, by successive refinements and use of tools, the user will be able to create a model and study its underlying characteristics. The browser lets the user make a reverse engineering work called "design recovery" by Chikofsky and Cross [2]. The user is able to fully understand the implementations and their underlying model and then study the relationship between interactions. Moreover, by the variations of the models, it acts as a "brain stimulator" as Hofstadter explains in "Metamagical Themas"[8]: "Variations on a thema should be considered as the fuel of creativity".

Section 2 presents the IODA methodology and its relevance in the design of the browser. We describe in section 3 some measurement tools analysing the complexity of systems designed in accordance with the IODA methodology. Then, we develop into section 4 the design of LEIA browser. Finally, section 5 presents the opportunities that the browser opens onto the creation of models by using genetic algorithms.

2 The IODA methodology

The IODA methodology for simulation design[13] is focused on interactions: they are independently reified from agents. Agents from $\mathbb{A}$, the set of agents in the environment, have basic primitives:

- perception primitives on the overall state of the simulation (noted $\mathbb{E}$), the environment, its internal state and communication with other agents;
- action primitives which can alter the simulation state: its own state, the state of the environment or the states of other agents.

These primitives are used to define the role of the agent in specific interactions. An agent a also has a perception halo $\mathcal{H}_{\mathcal{F}}(a) \subseteq \mathbb{E}$: i.e. the subset of the overall state of the simulation which the agent a discerns through its perception primitives. The neighborhood $\mathcal{N}(a)$ of an agent a is the set of agents in his perception halo.

In the IODA methodology, the agents have a simple specification and are homogeneously represented, allowing the integration of any agent in a simulation model centered on interactions: an agent is an autonomous entity, instantiated from an agent family $\mathcal{F}$, noted $a \prec \mathcal{F}$. An agent family $\mathcal{F}$ in the set of families $\mathbb{F}$ is the abstraction of a set of agents sharing all or part of their perception or action primitives.

An interaction is a sequence of action primitives applied to several agents, which could play the role of source or target (respectively $\mathcal{S}$ or $\mathcal{T}$), and is subject to conditions of activation. The interactions are totally separated from agents who use them, increasing their reusability through simulations and are applicable to different families of agents.

We define the cardinality of an interaction $\mathcal{I}$ as the pair composed with the number of source agents $\mathrm{card}_{\mathcal{S}}(\mathcal{I})$ and the number of target agents $\mathrm{card}_{\mathcal{T}}(\mathcal{I})$ that we need to perform the interaction.

The assignations are the set of interactions that a source agent may perform on target agents. We call assignation $a_{\mathcal{S}/\mathbb{T}}$ of an interaction set $(\mathcal{I}_k)_{k\in[1,n]}$ between a source agent family $\mathcal{S} \in \mathbb{F}$ and a set of target agent families $\mathbb{T} \subseteq \mathbb{F}$ the set of interactions belonging to agents $\mathcal{S}$ that they may perform as sources together with sets of agents from $\mathbb{T}$ as targets. It is defined as a set of tuples $(\mathcal{I}_k, p_k, d_k)$, $k \in [1, n]$, called assignation elements, where :

- $\mathcal{I}_k$: the interaction which could be performed by sources $\mathcal{S}$ and undergone by targets $\mathbb{T}$;
- $\mathrm{card}_{\mathbb{T}}(\mathcal{I}_k) = q$ the number of targets in $\mathbb{T}$;
- p_k : the priority given to an interaction $\mathcal{I}_k$;
- d_k : the limit distance below which the interaction can occur.

We define the **interaction matrix** as the matrix $\mathcal{M} = (a_{\mathcal{S}/\mathbb{T}})_{\mathcal{S}\in\mathbb{F},\mathbb{T}\subseteq\mathbb{F}^{\mathbb{N}}}$ of each assignations between sources $\mathcal{S}$ and possible targets $\mathbb{T}$ (e.g. Fig.3). Building of a model using the IODA methodology is done through 6 steps:

1. identification of families of agents and interactions, as elements of a matrix of interaction;
2. writing activation conditions and sequences of actions which are primitives of each interaction;
3. identification of action and perception primitives of agents;
4. specification of the priority and the limit distance of each interaction;
5. determining the dynamics, i.e. the evolution of the interaction matrix built at the previous steps;
6. determining the specificities of the model.

During the development of models in the LEIA browser, agent families and interactions are specified a priori. The conditions of action and perception of interactions are also set. Then, the design of the model can be limited to the available choice of interactions and families of agents without the need to generate code in order to use them. We can modify in runtime the model of a simulation without having to stop it.

3 The measurement tools

As proposed by Kubera[10], it is possible to characterize part of the complexity of a computer simulation by quantifying the number of computation cycles between the beginning and the end of the simulation. The complexity lies in different aspects of the simulation:

- in the studied phenomenon;
- in the complexity of the cognition of agents;
- in the way to design the simulated phenomenon and the simulation engine;
- in the way to achieve those two models.

We specify in this section some heuristic measures which characterize the complexity of a system designed through the IODA methodology and simulated on the JEDI engine (Fig.2).

These tools are implemented in the JEDI engine [12] which is a sequential engine with a discrete representation of time. An interaction $\mathcal{I}$ has only one source $\mathcal{S}$ ($\text{card}_{\mathcal{S}}(\mathcal{I}) = 1$). At each time step, for each potential source $\mathcal{S}$, it selects a couple (interaction $\mathcal{I}$, target $\mathcal{T}$). The interaction $\mathcal{I}$ is chosen following the assignation of highest priority among all feasible assignations for this source $\mathcal{S}$. The target $\mathcal{T}$ of this interaction $\mathcal{I}$ must also be in the neigbourhood of the source $\mathcal{N}(\mathcal{S})$. For the selected couple, then, we solve the action of the interaction $\mathcal{I}$ between the source $\mathcal{S}$ and target $\mathcal{T}$. This interaction $\mathcal{I}$ can be recorded as a "resolved interaction". At each time step, we can therefore have a specific and quantified return on all the events of the simulation from which our tools of measurement are defined.

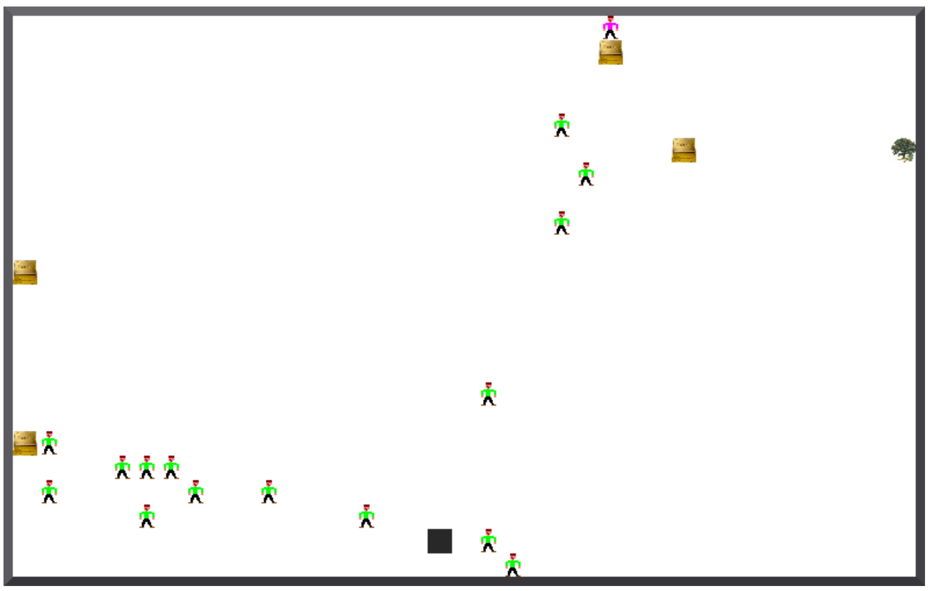

Fig. 2. Screenshot of the environment of an "Age of Empire"-typed simulation using the JEDI engine. Some peasants wander in the environment, find some gold or wood and then bring back them to a forum. They also inform the other peasants where they have found these ressources, creating a chain of workers starting from the forum to the common objective.

These tools are designed to reveal the qualities and defects of simulated models. In what follows, we put ourselves, using the LEIA browser, in a simplified situation where each cell of the environment can only be occupied by one agent.

The initial distribution of agents and the primitives are accordingly implemented from this situation.

3.1 Simulation activity

Agents activity. With the JEDI engine, we can monitor the activity of agents in a simulation, particularly if an agent is able to perform some interactions. This measure called "Agents activity" characterizes the interactivity of the simulation, i.e. the ability of the simulation to run and therefore evolve.

Definition 1. ***Activity***
With $\mathbb{I}(t)$ the set of "resolved interactions" during the time step t and $\mathbb{A}(t)$ the set of agents in a given simulation, the ***activity*** *of agents is given by:*
$\mathit{Activity}(t) = \mathrm{card}(\mathbb{I}(t)) \,/\, \mathrm{card}(\mathbb{A}(t))$

The provided score is the ratio between the number of agents which are sources of interaction and all agents of the simulation. We can underline that in the JEDI engine, by default, upon being resolved, the source $\mathcal{S}$ and target $\mathcal{T}$ of an interaction are disabled [12]: the target $\mathcal{T}$ cannot participate in another interaction at the same time step, either as target or source. The lower the value of the activity metric, the greater are the number of passive agents: their state will evolve through the few interactions that have been resolved (considering that their internal state does not change by itself through internal lookup by example). In the interaction point of view, this measure can thus reveal some very complex models such as the sale or purchase of items which do not generate changes in the environment and its representation.

3.2 Environment

Environment modifications. The JEDI engine provides an environment similar to a collection of cells that can be occupied by agents (agents have real coordinates [12]). The environment is graphically represented in the simulation as a 2-dimensional grid, made from cells in which agents are represented. It has a set of primitives, such as *Put an agent* or *Remove an agent* of the environment, the use of which requires an update of the associated graphical representation. Therefore, we propose to measure, at time step t, the number of calls for these primitives of the environment, noted $\mathcal{E}(t)$, compared to the number of agents in the simulation.

Definition 2. ***Modifications***
At time step t, the number of ***modifications*** *from agents is defined by:*
$\mathit{Modifications}(t) = \mathcal{E}(t) \,/\, \mathrm{card}(\mathbb{A}(t))$

We get an indicator of the visual entropy of the simulation: we mean here the evolution of the representation of the environment between two time steps. This measure is open, typically between 0 and 1 if the resolved interactions call to only one environment primitive requiring an associated graphical update.

Environment stability. This indicator is a measure of stable points of the simulation. This is done through the evolution of the occupation of environment cells in relation to each family of agents. A large deviation in some cells shows that they are occupied repeatedly by the considered family of agents: so we can conclude that this is a stable point in the simulation.

Definition 3. ***Stability***
At time step t, with:

- $N_{\mathcal{F}}(t) = \mathrm{card}(\mathbb{A}_{\mathcal{F}}(t))$ *the number of agents from family* $\mathcal{F}$ *;*
- p *the number of cells in the environment;*
- $\mathcal{O}_{c,\mathcal{F}}(t)$ *the cumulated presence of agents of the family* $\mathcal{F}$ *since the beginning of the simulation in cell c;*
- $\mathcal{M}_{\mathcal{F}}(t) = \frac{N_{\mathcal{F}}(t) \times t}{p}$ *the average occupancy of cells.*

The standard deviation of cells occupancy is:
$\sigma_{\mathcal{F}}(t) = \sqrt{\frac{1}{p} \times \sum_{c=1}^{p} (\mathcal{O}_{c,\mathcal{F}}(t) - \mathcal{M}_{\mathcal{F}}(t)^2}.$

Let's imagine a model in which there is absolutly no movement: since the start, every agent has a different cell, cannot move into another cell and the population Nb is absolutly the same in number since the beginning of the simulation. At time step t, this model gives the worst standard deviation:
$\sigma\mathrm{def}_{\mathcal{F}}(t) = \sqrt{\frac{1}{p} \times (\sum_{c=0}^{p-Nb} (0 - \mathcal{M}_{\mathcal{F}}(t))^2 + \sum_{c=p-Nb}^{p} (t - \mathcal{M}_{\mathcal{F}}(t))^2)}$

Then, the ***stability*** *is defined as:*
$Stability_{\mathcal{F}}(t) = \sigma_{\mathcal{F}}(t) / \sigma\mathrm{def}_{\mathcal{F}}(t)$

If at time step t, the population Nb peaks and, subsequently, cell occupancy stagnates, the standard deviation will converge towards the unfavourable metric. A simple measure of Agents stability would have been to determine the proportion of agents with unchanged position after two time steps. Our measure takes into account the cumulative presence of agents since the beginning of the simulation: it enables us to reveal areas of convergence of agents in the environment.

We also provide two indicators on the environment: the mix and the cohesion. The mix is the average percentage of agents from other families in the neighborhood of each agent. Similarly, cohesion is the average percentage of agents from the same family in the neighborhood of each agent. The general idea is the commonly accepted idea of similarity percentage in the Moore neighborhood, defined as the eight cells surrounding a cell centered on the given agent.

Definition 4. ***Mix and Cohesion***
At time step t, with:

- $\mathcal{N}(x)$ *the neighborhood of the agent x;*
- $\mathcal{F}(x)$ *the family from which agent x is instantiated;*
- $Diff(x) = card(\{a \in \mathcal{N}(x) | \mathcal{F}(a) \neq \mathcal{F}(x)\})$ *the number of agents in the neighborhood of the agent x whose family is different;*

– *$Same(x) = card(\{a \in \mathcal{N}(x) | \mathcal{F}(a) = \mathcal{F}(x)\})$ the number of agents in the neighborhood of the agent x whose family is the same;*
– *$CellsP(x)$ the number of cells in the neighborhood of the agent x;*
– *$Nb = card(\mathbb{A})$ the number of agents in the environment;*
– *p the number of cells in the environment.*

*The **mix** is defined as: $Mix = (1/p) \times \sum_{x=1}^{Nb}(Diff(x)/CellsP(x))$;*
*the **cohesion** is defined as: $Cohesion = (1/p) \times \sum_{x=1}^{Nb}(Same(x)/CellsP(x))$.*

We also use a traditional definition of density.

Definition 5. ***Density***
*At time step t, with $Nb(t) = \mathrm{card}(\mathbb{A}(t))$ the number of agents in the environment and p the number of cells in the environment, the **density** of population is defined as: $Density = \frac{Nb(t)}{p}$.*

3.3 Study of the evolution of populations and the resolution of interactions

Usage of interactions. Thanks to the IODA methodology, the separation of interactions and agents allows us to easily record the "resolved interactions" from each entity. The analysis of those records can reveal the behaviour of agents and the overall usage of interactions, especially some cycles in their usage and order between them, as shown in the next examples.

Source \ Target	Envir.	Plant	Grasshopper
Plant	-	-	-
Grasshopper	-	Graze 1	Devour 0

Fig. 3. Interactions matrix from a simple model of food search: *Graze* has an higher priority than *Devour*

Let's take two families of agent: *Plant* and *Grasshopper*; and two interactions: *Graze* and *Devour*. In Fig.3, the *Grasshoppers* will *Graze* the *Plants* by priority then, when the *Plants* will have disappeared, the *Grasshoppers* will *Devour* themselves. The interaction *Devour* will only occur when there are no plants.

Let's imagine that in this example, new *Plants* could grow during simulation in sufficient numbers to feed the *Grasshoppers*: the interaction *Devour* can never happen then, which allows a step of simplification in this model. Although we do not have a knowledge of the evolution of the simulation, we can detect as a heuristic measurement the interactions that do nothing to a given model.

Interactions dynamic. In a feedback phenomenon, the result of the phenomenon in question acts back on itself. In the case of simulations, such feedback can occur in solving interactions.

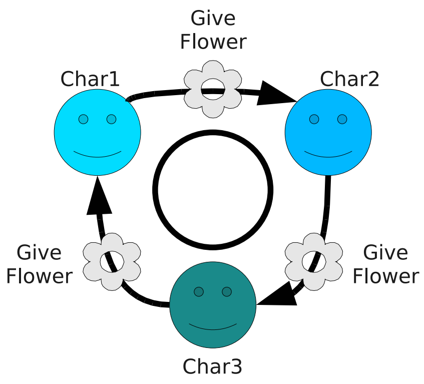

Fig. 4. Simple experience of object transmission between 3 characters

Let's take a simple example (cf. Fig.4): 3 agents are placed around a table. At the beginning of the simulation, an object *Flower* is given to one of the *Char*. During simulation, if an agent has a *Flower*, it performs the interaction *Give* the *Flower* to one of his neighboring agent. Considering one of the agents, the interaction *Give Flower* will be made once every three time steps, waiting for the *Flower* object to go around the table: this phenomenon is periodic. Having a record of each "resolved interaction" for each agent in the JEDI engine allows us to detect cyclic use of interaction that are part of a possible feedback phenomenon.

Let's consider the previous example that we modify (cf. Fig.5):
after receiving the object *Flower*, the agent first performs the interaction *Thank* with its neighboring donor as target, then performs at next time step the interaction *Give Flower* with his other neighboring agent. If we study the interactions of one agent, we find that the interactions *Thank* and *Give Flower* are performed every 6 time steps (2 time steps for each charecter). *Give Flower* can only be performed after *Thank* during the cycle of 6 time steps: there will be a phase of $\frac{2\pi}{6}$ between the two interactions. If we follow the interactions in general, the interactions *Thank* and *Give Flower* are alternately performed once at each time step. An order between *Thank* and *Give Flower* is viewable, with a periodic phenomenon of 2 time steps. In the case we have a low-level knowledge of the scenario being modelled, we propose the use of frequency analysis tools in order to determine the periodic phenomena in the use (frequency) and order (phase) of interactions.

DFT. By seeing the use of interactions as a discrete signal, we can use the classic definition of the Discrete Fourier Transform (**DFT**) in order to study the interactions in the space of frequencies without constraint in the choice of frequencies and the sampling.

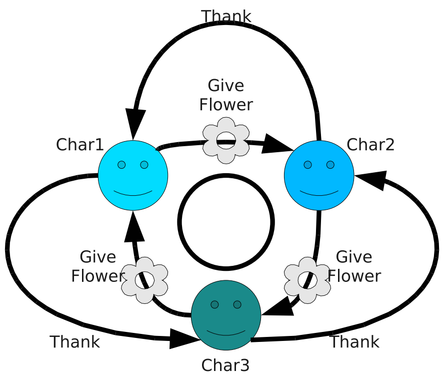

Fig. 5. Simple experience of object transmission with thank between 3 characters

Definition 6. ***DFT***
With $\mathrm{s}_{\mathcal{I}}(n)$ *the evolution of usage of an interaction* $\mathcal{I}$ *and* Nts *the number of time steps used by the DFT, the* ***DFT*** *is defined as:*
$\mathcal{S}_{\mathcal{I}}(k) = \sum_0^{\mathrm{Nts}-1} \mathrm{s}_{\mathcal{I}}(n) \times e^{-2ikn/\mathrm{Nts}}$

Remarkable frequencies. We propose finding local maxima in the frequency spectrum obtained by Fourier transformation of the evolution of resolved interactions.

Definition 7. ***Remarkable frequencies***
With $\mathcal{S}_{\mathcal{I}}(k)$ *the DFT of* $\mathrm{s}_{\mathcal{I}}(n)$*, the set* Freq *of* ***Remarkable frequencies*** *is:*
$\mathrm{Freq} = \{k / \mathcal{S}_{\mathcal{I}}(k-1) < \mathcal{S}_{\mathcal{I}}(k) \textit{ and } \mathcal{S}_{\mathcal{I}}(k) > \mathcal{S}_{\mathcal{I}}(k+1)\}$

If the sample is correctly selected, we can detect the periodic usage of an interaction. The sample is limited in time, so we can only find out periodic usage of an interaction which repeats within the sample. As we can't make an infinite run of a simulation, we approximate a repeated usage of an interaction as a cycle.

We can reach 2 levels of analysis in monitoring the interactions:

- A macroscopic monitoring, i.e. taking into account all the interactions that reveal the dynamics of the global system. The discovery of remarkable frequencies may highlight the coupling of some interactions.
- A microscopic monitoring, focusing on one agent, where we can follow the resolution of interactions. This analysis can reveal the dependency between the behaviour of the agent and his assignations. It faces, however, the life expectancy of agents in some models (for example, the model prey / predation). Moreover, some interactions can disable the agent (at the choice of the developer): an agent may only undergo interactions.

The frequency analysis needs to be counterbalanced because some interactions may follow a periodicity intrinsic to them (and therefore independent of the conduct of the simulation itself).

Study of the evolution of populations. Some models, like the prey / predator, will lead to the periodic variation of populations, revealing feedback phenomena. The detection of these periodic phenomena is carried out by Fourier analysis, as described for the analysis of interactions.

4 LEIA: a browser in simulations space

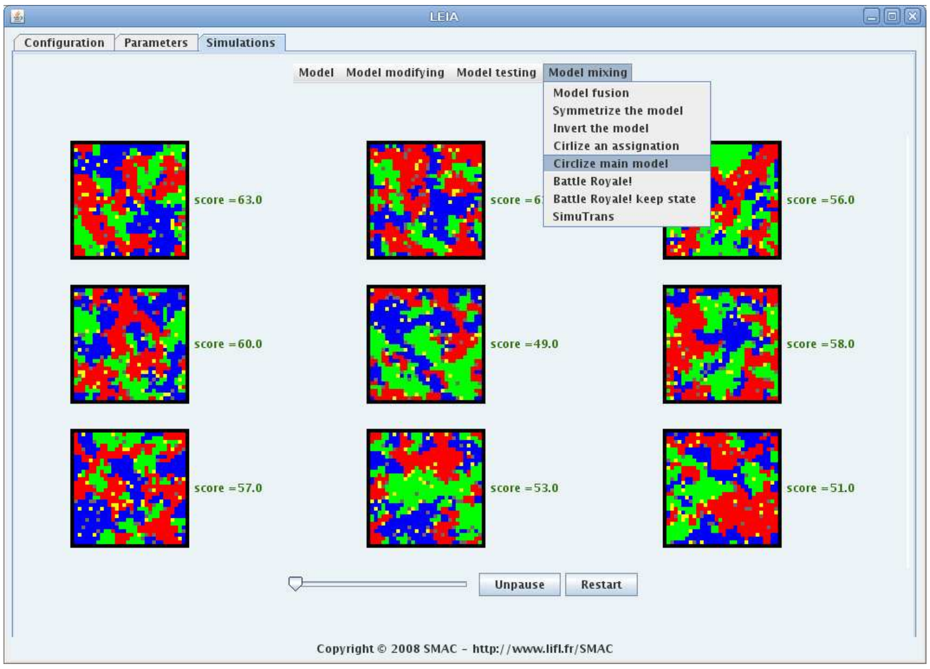

Fig. 6. LEIA, the browser in simulations space

The LEIA browser[3] is an application using n instances of the JEDI engine. It allows the user to instantly make a visual comparison of n simulations by seeing all of them working in parallel (Fig.6). These n instances are created from the same reference model, based on an ontology [6] constituted of already-built family of agents and interactions. We can also define the beginning number of agents and their initial distribution. By giving a domain ontology in input, LEIA is able to build automatically several simulations from the simulations space associated to this domain.

[3] for *LEIA lets you Explore Interactions for your Agents* www2.lifl.fr/SMAC/LEIA/applet.html

Model manipulation. The LEIA browser provides to the user a set of transformations and generation tools for model, and a set of tests to browse the simulations space. These tools can vary the parameters of the model, either by adding or deleting assignations, changing priorities or limit distance, the initial number of agents, or operations on the interactions matrix as to symmetrize or merge matrices of 2 models. The user can then, by these tools, automatically change the reference model to generate N sub-models. These models are then loaded into the N instances of the JEDI engine: the user can view and compare the N simulations with separate behaviours. LEIA can be run with as many instances as we want. Of course, LEIA will run slower proportionally to the number of instances chosen. Nevertheless, each instance has its own thread, then we can have benefit of a multi-core architecture. At this time, with a quad-core architecture, we are able to run 4 instances at the same time as one.

Simulation analysis. The browser is assisted by a statistics engine to help the user to appreciate the qualities and differences between the displayed models. This statistics engine is based on measurement tools presented in the previous section. We get a quantified return for each simulation in which we consider:

– all interactions resolved by time step compared with envisaged interactions;
– the number of modifications of the environment;
– the repartition of agents: cohesion and mix;
– the evolution of the occupation of the environment;
– remarkable evolution of populations;
– remarkable evolution of interactions.

These heuristic measures allow the user to access informations about each model with complete detail of scores and the associated graphics display is updated in real time.

Model analysis. In addition to the measurement tools that we have already presented, we can study in the specific context of LEIA browser the construction of the model, especially its interactions matrix. The so-called circular assignations are remarkable: i.e. for the same interaction, priority and limit distance, the sources and targets vary cyclically.

Definition 8. ***Cyclic assignations***
With $e = (I, p, d)$ *an assignation element.*
we define: $Assi(e) = \{(\mathcal{S}, \mathcal{T}) \in \mathbb{F}^2 | e \in \mathrm{assi}_{\mathcal{S}/\mathcal{T}}\}$ *the set of assignations from the Assignation Matrix with the same interaction* I*, priority* p *and limit distance* d*.*
If $\mathrm{assi} \in \mathrm{assi}_{\mathcal{S}/\mathcal{T}}$*, then:*

– $\mathrm{Src}(\mathrm{assi}) = \mathcal{S}$ *is the agent family of the source of* assi*;*
– $\mathrm{Tgt}(\mathrm{assi}) = \mathcal{T}$ *is the agent family of the target of* assi.

the ***cyclic assignations*** *is the set of couples* $(\mathcal{S}, \mathcal{T})$ *taking part in the cycle* e*:*
$\mathrm{Assicyliques}(e) = \{(\mathcal{S}, \mathcal{T}) \in Assi(e) / \exists (\mathrm{assi}_i)_{i \in [1,n]} \subseteq Assi(e) / \mathrm{Src}(\mathrm{assi}_1) = \mathcal{S} \wedge \mathrm{Tgt}(\mathrm{assi}_1) = \mathcal{T} \wedge (\forall i \in [1, n-1], \mathrm{Src}(\mathrm{assi}_{i+1}) = \mathrm{Tgt}(\mathrm{assi}_i)) \wedge \mathrm{Src}(\mathrm{assi}_1) = \mathrm{Tgt}(\mathrm{assi}_n)\}$.

Definition 9. ***Cyclic aspect***
*The **cyclic aspect** is defined as:*
Cyclique = card(Assicyliques)/card(Assi)

The cyclical aspect of a model is the proportion of cyclical assignations among all assignations. The study helps to highlight cycles in the construction of model that can possibly result in feedback loops.

More generally, the study model also opens the prospect of automated simplification of models, for which we are laying the foundations in LEIA by eliminating unreachable interactions, e.g. due to their priorities or limit distance.

Scoring models. We provide to the user a total score to bring together the results of all the tools of measurement.

Definition 10. ***Total score***
With $\mathbb{S} = \{S_1, ..., S_N\}$ the set of scores provided by the measurement tools (scores between 0 *and* 100*), the **total score** is defined as:*
$Totalscore = 1/N \times \sum_{i=1}^{N}(S_i - 50)$

We made the choice to reduce the total score in a note typically between -50 and 50 not to emphasize a score over another. The score is centered on 0 to give a simple reference to the user. We don't use multiplication because there is the risk of hiding interesting data because of a particular score which would be zero.

By seeing the behaviour of all simulations in parallel, when one of them is considered as interesting by the user with the help of the measurement tools, its model can be defined as the new reference model. Then, the user can repeat the process of replacing the reference model, either manually or by using our tools of transformation. The LEIA browser therefore allows one to explore the simulations space generated from the domain ontology board. It should be noted that the LEIA browser is open to any domain ontology, as the interactions and agent families are designed according to the IODA methodology.

Source \ Target	Envir.	Red	Blue	Green
Red	-	-	Infect 0 (1.0)	-
Blue	-	-	-	Infect 0 (1.0)
Green	-	Infect 0 (1.0)	-	-

Fig. 7. Interactions matrix of the infection model. It can be extended to a greater number of agent families than two. "Infect 0 (1.0)" means that the interaction "Infect" only occurs under a maximal distance of 1.0.

Results. The tools presented here allow the user to perform reverse-engineering on simulations. This simulations can be discovered using the browser among the simulations space.

Like Pachet shows with the “Continuator” [17], which stimulates musical creativity, LEIA tends to be a “brain stimulator” for the discovery of new models, and helps the user to identify interesting models. Even with a simple ontology, with few classical interactions, benefits of the LEIA browser are outstanding, as you can see with the following “infection model”.

The observation of an experiment displaying a synchronization phenomena between some agent families pointed out an interesting set of interactions at its origin. This set of interactions contains two interactions: one that clones the source on a neighboring position, and the other that kills the target. Thanks to an analysis of the interaction matrix, this set was simplified to a single interaction. Briefly, the aim of this new interaction is to destroy a targeted agent found in the neighborhood and to replace it with a copy of the source agent.

This model, found by LEIA, cyclically affects several families of agent with this interaction called “Infect” (see Fig.7). At least three families of agent are required to avoid deadlock in this simulation. From initial positions which are random, this model led the agents to form spirals per infection (see left figure in Fig.8). LEIA points out that the “infection model” can’t work without filling the environment with a huge and equal amount of agents from each family. Indeed, the greater the number of agents, the greater the probability for a source agent to find a target. Moreover, the limit distance is really important: having a higher limit distance for an interaction allows agents to find further a target for this interaction. Thus, the higher the limit distance is for “Infect”, the higher is the probablity to fire it. Of course, raising this limit distance helps to increase the number of family agents.

We point out the robustness of this model by adding some obstacles in our experiments. Those obstacles are empty agents that don’t interact with any other agents in simulation: they just occupy a place in the environment. Spirals can occur though the presence of obstacles. Moreover, these obstacles can make easier the formation of spiral at their positions, like the right image in Fig.8.

It appears that this dynamic[4] is well known in chemistry, e.g. in the Belousov-Zhabotinsky cyclical reaction [20, 1]. Such phenomena are also examined with the help of cellular automata in the Greenberg-Hastings model [5].

5 Towards a genetic evolution

In the LEIA browser, designing a model can be automated by drawing random assignations. We can also dynamically implement this model with the view to immediately test it in a simulation. The browser also allows the use of multiple simulations at the same time. We can create and test a model, then help the user to judge its quality by using our measurement tools. Indeed, they facilitate the search for some phenomena, for example to identify:

– phenomena of segregation using the measurement tools about cohesion, mixing and stability;

[4] www2.lifl.fr/SMAC/LEIA/spirale.html

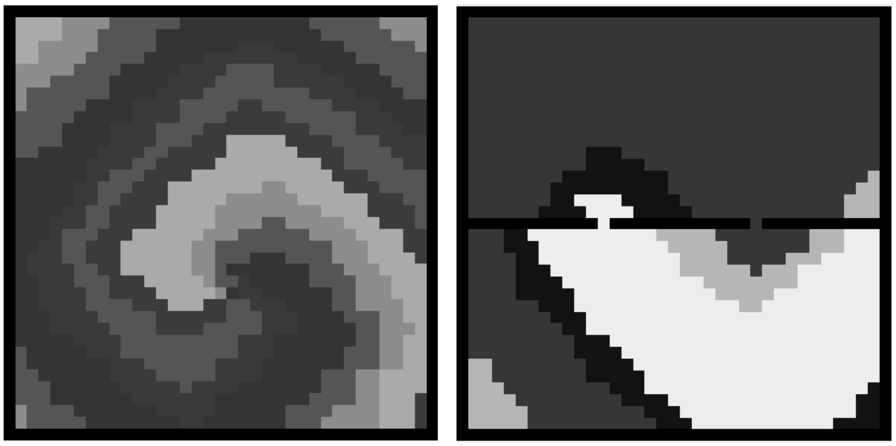
Fig. 8. Two screenshots of the environment of JEDI engine using the "infection model". At left, formation of a spiral in infection model between 7 colors. The figure on the right shows the robustness of this model even if obstacles exists in the picture.

- cyclical evolution of the population from which we can detect remarkable frequencies;
- point out some models which converge towards stable positions by observing the variations of cells occupation;
- models causing a major renewal of the environment by studying its modifications.

The user can identify an interesting model by specifying precise research criteria and, by successive iterations, refine the model in order to obtain a sought phenomenon. Like the user can identify Dawkins biomorphs in "The Blind Watchmaker"[3] whose development meets its desires, he can design models corresponding to his needs by viewing them.

We can link the way to design new models in LEIA to the works about Imagine [15]: designers suggest an original technique for building CSS stylesheets by using a genetic algorithm and successive evaluations through a user interface. Here, each stylesheet parameter is a gene that can be crossed or transferred. The algorithm randomly generates stylesheets, used to the same text. A user can then choose one or more stylesheets with pleasant characteristics. The algorithm then generates new stylesheets by taking into account the previous choices in order to converge, after some iterations, to a stylesheet that the user deems to his liking.

The browser in the simulations space opens the perspective of the generation of models, written in the IODA methodology, through a genetic algorithm. When a problem admits a set of solutions, a genetic algorithm solves it by evaluating a set of solutions parametrized with a fixed number of genes. These genes can evolve by crossing and mutations of the solution in order to maximize an evaluation function [9]. The algorithm converges towards a solution which is considered to be good. The designer also defines a fitness function to fit the sought solution.

In LEIA, the user's choice of specific measurement tools allows the creation of fitness functions. All scores are evaluation functions judging the quality of models. We can see the assignations as genes with which a mutation factor is involved. Then, the mutation can be applied to parameters of an assignation: priority, limit distance, source, target, interaction with different probabilities depending on the supposed impact of the parameter: modifying the distance limit changes the behavior of a model less than changing the interaction.

6 Conclusions and Perspectives

The browser in simulations space allows the iterative design of multi-agent models through the IODA methodology, which provides a clear separation between agents and their interactions allowing thus composition without code generation. We offer a range of tools for processing and simplifying models that can then be viewed in parallel. Then, we propose measurement tools designed from the perspective of IODA. These heuristic measures highlight some features of these models: system activity, spatial distribution, stability over time, feedback, etc. They make easier the understanding of models built in this way.

We can reverse the usual way of model design, by firstly observing the result (i.e. "what" happens) then the corresponding agent behaviors (i.e. "how" it happens). Through its tools and ease of model design, the LEIA browser acts as a "brain stimulator" whose first result was to find a model similar to the dynamic of Greenberg-Hastings model. Moreover, the browser is open to any ontology as agents and interactions are being designed following the IODA methodology: LEIA aims at exploring simulations space of domains as various as physics or biology. Ultimately, we envisage the construction of models by genetic algorithms, models in which the matrix of assignation can be seen as a set of genes, our measurement tools used for evaluating these models and the search for special features.

References

1. B. P. Belousov. A periodic reaction and its mechanism. In *Compilation of Abstracts on Radiation Medicine*, 1959.
2. Elliot J. Chikofsky and James H. Cross II. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 07(1):13–17, 1990.
3. R Dawkins. *The Blind Watchmaker.* W.W. Norton & Company, Inc., New York, 1986.
4. Gabriel Desmeulles, Gabriel Querrec, Pascal Redou, Sébastien Kerdélo, Laurent Misery, Vincent Rodin, and Jacques Tisseau. The virtual reality applied to biology understanding : the in virtuo experimentation. *Expert Systems with Applications*, 30(1):82–92, 2006.
5. R. Fisch, J. Gravner, and D. Griffeath. Metastability in the Greenberg-Hastings Model. In *eprint arXiv:patt-sol/9303005*, pages 3005–+, March 1993.
6. Thomas R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing in Formal Ontology in Conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers, London, 1993.

7. Olivier Gutknecht and Jacques Ferber. The MADKIT agent platform architecture. In *Agents Workshop on Infrastructure for Multi-Agent Systems*, pages 48–55, 2000.
8. D. Hofstadter. *Ma thémagie: En quête de l'essence de l'esprit et du sens*. Interéditions, London, 1988.
9. J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.
10. Yoann Kubera, Philippe Mathieu, and Sébastien Picault. La complexité dans les simulations multi-agents. In Valérie Camps and Philippe Mathieu, editors, *Actes des 15e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2007)*, pages 139–148. Cépaduès, 2007. JFSMA'2007 – Carcassone (France) – 17-19 octobre 2007.
11. Yoann Kubera, Philippe Mathieu, and Sébastien Picault. Interaction-oriented agent simulations : From theory to implementation, ECAI 08 July 21-25 2008.
12. Yoann Kubera, Philippe Mathieu, and Sébastien Picault. Une architecture orientée interactions. *Revue d'Ingéniérie des Systèmes d'Information (ISI)*, 2008. Numéro spécial Architectures Logicielles.
13. Philippe Mathieu, Sébastien Picault, and Jean-Christophe Routier. Donner corps aux interactions. In *Actes des 4e Journées Francophones sur les Modèles Formels de l'Interaction (MFI'07)*, pages 333–340. Université de Paris Dauphine, 2007. MFI'07 – Paris (France) – 30 mai, 1er juin 2007 – Papier court.
14. N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system, a toolkit for building multi-agent simulations, 1996.
15. N. Monmarché, G. Nocent, M. Slimane, and G. Venturini. Imagine: a tool for generating HTML style sheets with an interactive genetic algorithm based on genes frequencies. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC'99)*, volume 3, pages 640–645, Tokyo, Japan, October 12-15 1999.
16. Philippe Mathieu Nouredine Bensaid. A framework for cooperation in hierarchical multi-agent systems. *Journal of Mathematical Modelling and Scientific Computing*, 8, September 1998.
17. F. Pachet. De la co-construction d'un langage homme-machine: quelques expériences en musique. In Valérie Camps and Philippe Mathieu, editors, *Actes des 15e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2007)*, page 9. Cépaduès, 2007. JFSMA'2007 – Carcassone (France) – 17-19 octobre 2007.
18. Yves Demazeau Pierre-Michel Ricordel. La plate-forme volcano - modularité et réutilisation pour les systèmes multi-agents. *Technique et Science Informatiques*, 21(4):447–471, 2002.
19. Uri Wilensky. Netlogo (and netlogo user manual), 1999.
20. A. M. Zhabotinsky. Periodic processes of malonic acid oxidation in a liquid phase. In *Biofizika*, 1964.