Towards Discrete Event Multi Agent Platform Specification

Sébastien Mattei, Paul-Antoine Bisgambiglia, Marielle Delhom, Evelyne Vittori
University of Corsica, CNRS UMR SPE 6134, UMS Stella Mare 3514
Corte, France
{smattei,bisgambiglia,delhom,vittori}@univ-corse.fr

***Abstract*— Nowadays, simulation tools have become essential. They allow to study and understand complexes actions that may be impossible to study in situ. In this paper we introduce an approach to use Discrete Event System Specification model as agents and finally create an original platform which allows using DEVS framework and a Multi Agent System platform working together to simulate population dynamics. We first apply this approach on anthill model and then the final approach to a fish model. Ants' basic behaviors are added successfully in DEVS formalism.**

Keywords-DEVS; MAS; Modeling; Simulation.

I. INTRODUCTION

For years, we have been working on modeling and complex system simulation [1–4], and on Discrete Event System Specification (DEVS [5]). Our researches are essentially based on DEVS formalism [6]. In 1970's, Professor Zeigler [5] introduced this method that has proved successful. It represents: (1) a complex system from an interconnected collection with more simple subsystems; (2) a separation between modeling and simulation, simulation algorithm are automatically generated according to defined models. This formalism is open, flexible and offers a large extension capacity.

According to recent works [4], [7–11], it has been proved that DEVS formalism might be qualified as a multi-formalism thanks to its opening capacity, to its capacity to encapsulate others modeling formalisms. In one heterogeneous system, it is possible to use modeled subsystems from different formalisms, differentials equations, neuron networks, continuous systems.

These opening and extension capacities are really interesting in our researches, because this formalism has boundaries and doesn't allow a representation of all kind of systems like living systems. In order to get over these boundaries, Multi Agent System (MAS [12], [13]) seems to be an interesting alternative.

MAS's purpose is to create cooperation between entities (agents) that have intelligent behavior, and to coordinate their purposes and their action plans to solve a problem. In our case, MAS utilization is justified because they are adapted to reality, they also allow: (1) agents cooperation; (2) to solve complexes' issues; (3) incomplete expertise integration. An agent is a physical or virtual identity determinated by movements collections (individual objectives, functions of satisfactions or survival), owning its own resources and getting just one partial representation (or none) of its environment. An agent's behavior goes to satisfy its objectives according to its resources, its skills and its functions of perceptions, representations and communications. MAS have got a lot of applications into artificial intelligence. They can reduce complexity of issue's resolution by breaking the sub-collection's knowledge by associating an intelligent independent agent to each of its sub-collections and coordinating activity of these agents [14]. MAS are related to Distributed Artificial Intelligence (DAI).

Our team works on several scientific research and technology development. These two domains include concepts (scientific way) and concepts implementations (technological way). They are used for issue's study linked to artificial or naturals complexes system's behavior like management and modeling evolutive interfaces systems (spread of pollutants) and natural system's modeling (tides, fishes), telecommunication, acquisition's system's conceptions (sensors) and analysis and data treatment (decision's help).

So, our objective is to capitalize the twenty last years gained experiences and skills to propose a brand new and ambitious project. This project's final objective is to propose a brand new platform to:

- Modelize different processes working on fauna and flora's evolution (tide, wind, phytoplankton, zooplankton, larva, algae, fishes, pollutants, etc.);
- Simulate from autonomous and intelligent's agents the interaction and evolution of these processes. We are going to develop and integrate in a multi modelization and simulation based on multi formalism's environment «Discrete Event System Specification» (DEVS a hybrid platform based on multi agent system's properties (MA S [14]);
- Finally, we want to provide tools for decision help to make easier the simulating results operations and resource management.

To meet our goals we are going to proceed step by step. In this paper, we propose to associate DEVS formalism and MAS. From advantages of these two paradigms, we want to define an extension of DEVS's formalism, faithful to standard, that make possible to modelize agents and way of communication and environment interactions as a DEVS's system. This transformation, from agents to model, emerge a lot of issues that we propose to study and solve by our self.

In the first part, we introduce DEVS formalism and MAS principles. Then, in a second part, we detail these two modeling and simulation's paradigms advantages. In the third part, we propose our approach formalization. Then, the

 ISBN: 978-1-61208-222-6

fourth part is dedicated to a MAS study case presentation before proposing its DEVS's model conversion. At last, the final part concerns the conclusion and a presentation of our future works.

II. OVERVIEW

Today, simulation's tools have become essential. They allow to study and understand complexes actions that may are impossible to study in situ. In this part, we introduce our works based on our researches and in particular two modeling and simulation's methods.

A. DEVS presentation

DEVS formalism [5], [6] is based on the definition of two types of components: atomic models and coupled models.

Atomic model (Fig. 1) provides an autonomous description of the system behavior, defined by states, input/output functions and transition functions. The coupled model is a composition of atomic models and/or coupled models. It is modular and presents a hierarchical structure which enables the creation of complex models from basic models.

1) DEVS models

Atomic DEVS model:

$$AM: < X; Y; S; t_a; \delta_{ext}; \lambda; \delta_{int} > \quad (1)$$

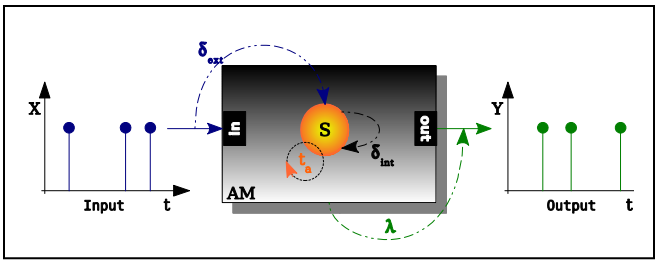

Figure 1. Atomic model.

where:

- X: is the set of input events, is characterized by a couple (port, time, value), where the port means the input on which the event occurs, the time is the date of occurrence of the event, it is blank for internal events, and the value symbolizes the data from the event;
- Y: is the set of output events;
- S: is the set of partial or sequential states, which includes the state variables;
- t_a: $S \rightarrow T_\infty$: is the time advance function which is used to determine the lifespan of a state;
- δ_{ext}: $QxX \rightarrow S$: is the external transition function which defines how an input event X changes a state of the system, where $Q = \{(s, t_e) \mid s \in S, t_e \in (T \cap [0, t_a(s)]\}$ is the set of total states, and t_e is the elapsed time since the last event, T is the total time of the simulation;
- λ: $S \rightarrow Y\iota$: is the output function where $Y\iota = Y \cup \{\iota\}$ and $\iota \notin Y$ is a silent event or an unobserved event. This function defines how a state of the system generates an output event, when the elapsed time reaches to the lifetime of the state;
- δ_{int}: $S \rightarrow S$: is the internal transition function which defines how a state of the system changes internally, when the elapsed time reaches to the lifetime of the state.

Every state S is associated with a lifetime t_a, which is defined by the time advance function. When a model receives an input event X, the external transition function δ_{ext} is triggered. This function uses the input event, the current state and the time elapsed since the last event in order to determine what the next model state is. If no events occur before the time specified by the time advance function for that state, the model activates the output function λ (providing outputs Y), and changes to a new state determined by the internal transition function δ_{int}.

Coupled model: coupled model is a composition of atomic models and/or coupled models. It is modular and presents a hierarchical structure which enables the creation of complex models from basic models. It is described in the form of:

$$CM :< XM, YM, CM, EIC, EOC, IC, L > \quad (2)$$

With:

- XM: all the input ports;
- YM: all the output ports;
- CM: the list of models forming the CM coupled model;
- EIC: all the input links connecting the coupled model to its components;
- EOC: all the output links connecting the components to the coupled model;
- IC: all the internal links connecting the components between themselves;
- L: the list of the priorities between components.

With the DEVS formalism, each model is independent and can be considered as its own entity or as a model of a larger system. DEVS formalism is closed under coupling, that is to say that for each atomic or coupled DEVS model it is possible to build an equivalent DEVS atomic model.

The DEVS models are executed by abstract simulators [15–17] that are independent from the models themselves. Consequently, separated concerns between models and implementations of simulation can be achieved and enhance the verification of each layer independently. DEVS is a popular method to simulate a variety of systems. However, since its introduction by B.P. Zeigler, significant efforts were taken to adapt this formalism to different fields and situations. The many proposed extensions proved its ability to extend and openness.

2) DEVSIMPY framework

DEVSimPy framework allows a simple graphical interface to create and use DEVS models. It is a WxPython based environment for the simulation of complex systems.

Its development is supported by the CNRS (National Center for Scientific Research) and the SPE research laboratory team.

The main goal of this framework is to facilitate the modeling of DEVS systems using the GUI library and the drag and drop approach. The interface is designed to help the implementation of DEVS model in form of blocks. The

 ISBN: 978-1-61208-222-6

modeling approach of DEVSimPy is based on UML Software, and there is a separating between the GUI part and the implementation part of DEVS formalism.

With DEVSimPy we can: (1) describe a DEVS model and save or export it into a library; (2) edit the code of DEVS model to modify behaviors also during the simulation; (3) import existing library of models which allows the specific domain modeling (Power Systems, Fuzzy, Continuous ...); (4) automatically simulate the system and perform its analysis during the simulation.

3) DEVS Advantages and drawbacks

Discrete event simulation has a quickly execution because of its way to treat event, avoiding continuous treatment. Moreover, coupling and separation between modeling and simulation on DEVS formalism allow reusing existing models in new models. DEVS is a powerful formalism allowing reusing models through library already developed and also interconnecting of these models to compose heterogeneous models based on a different formalism. In our team, it used to simulate continuous systems [2], and differentials equations, and fuzzy system [18], and sensor network and neural network.

As such DEVS does not allow simulating all kind of systems. For example, it is not quite complete to study systems describing behavior of living species, and their interaction with environment. As is a formalism associating other approaches, we would like to add new functionalities to coupling it with MAS.

After a description of the DEVS formalism and his working, we'll introduce Multi Agent System.

B. MAS presentation

MAS are more suited to living organism's modeling where communication between system's members is complex.

The multi agents' paradigm is issued from the distributed artificial intelligence in the early 80's [13], [14]. This bottom up approach is used to build individual based model dedicated to the study heterogeneous systems and solve problems with complex interactions. Multi-agent systems consist of agents and their environment.

According to Michael Wooldridge we consider that "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" [13].

Agents can either be physical (human being, robot, etc.) or virtual. The global behavior of MAS emerges from the sum of individual actions of agents, from the interactions between agents and between agents and their environment. Many Multi-agent systems platforms and frameworks are created by researchers and developers. They implement common standards useful to save developers time and also aid in the standardization of MAS development. Multi-agent systems are applied in the real world to computer games, environment, E-commerce defense, transportation, logistics, GIS.

1) MAS organisation

In MAS, environment is created in first. Then, agents are positioned on the environment with random position or known position. There are three kinds of agents:

- Reactive agent;
- Cognitive agent;
- Hybrid agent.

Reactive agent has no representation of its environment or others agents, it only reacts to environment stimuli. It has a simple behavior.

Cognitive agents are smart agents, they detect environment and others agents. They have skills, and they are able to plan an action with his skills and his thoughts. Hybrid agents are the most complex kind of agent: they are the middle way between cognitive agent and reactive agent. They may have simple behavior in reaction of stimuli or complex behavior like a cognitive agent. If we need both behaviors, we can use hybrid agent to describe simple behavior to manage memory (use less memory and it is less complex to code) and complex behavior when we need it. Agent may communicate with others and with environment thanks to three communications methods:

- Share memory (blackboard);
- Communication by messages;
- Message by environment.

Share memory is like a database process. Each agent fills a general knowledge base and takes information in. At any time an agent can ask blackboard for information. Communication by messages is like a conversation. Agents can send a message to one agent or various agents, and they can ask information to the others agents. The final method consists to give the whole responsibility to the environment. Only it can send information to agents.

2) MAS Advantage and drawback

Various kinds of agents exist and we explain their perspective's advantage in living organism's modeling.

TABLE I. *MAS AGENT AND COMMUNICATION SUMMARY*

Kind of agents	Way of communication	Architecture
Reactive	Shared memory (black board)	Subsumption architecture
Cognitive	By message	BDI architecture
Hybrid	By environment	Hybrid architecture

We have chosen reactive agent because its behavior is simpler to modelize, and it doesn't have an environment perception and it reacts only to exterior stimuli that we can modelize with an input message. Thereafter, we plan to use others kind of agents to make behavior and simulation more specific and reliable.

The MAS' major drawback is timing constraint. Using continuous simulation is longer and can cause issues. For example, it is difficult to apply an action to an agent at a given time. While DEVS make it possible. Moreover, MAS does not allow interconnection between heterogeneous

models. For example, it is not possible to associate an agent model with a model describing flow of a river.

C. Existing approaches

In this part, we introduce two existing approaches and explain our choices about our method.

The first approach we have studied is the platform GALATEA [19].

GALATEA is offered as a family of languages to model MAS to be simulated in a DEVS, multi-agent platform. GALATEA is the product of two lines of research: simulation languages based on Zeigler's theory of simulation and logic-based agents. There is in GALATEA a proposal to integrate, in the same simulation platform, conceptual and concrete tools for multi-agent, distributed, interactive, continuous and discrete event simulation. It is also GALATEA a direct descendent of GLIDER, a DEVS-based simulation language which incorporated tools for continuous modeling as well. In GALATEA, GLIDER is combined with a family of logic programming languages specifically designed to model agents.

This platform would allow modelize different formalisms, in different languages in a same interface. But this implementation seems tedious and difficult.

The second approach is the "Specification of Dynamic Structure Discrete Event Multiagent [20]" article. It matches a lot with our works. An agent is represented by an atomic model and stimuli by exterior messages. But using of VLE plateform [21], [22] to simulate and using of CELL-DEVS formalism [7], [23] to describe environment can make the simulation slower to generate. Indeed, each cell is represented by an atomic model and for a big environment number of atomic model is very high. Because of its number, simulation use lot of resources and memory, especially in our case with a big agent's concentration and a large environment (all of both are atomics models if we'll use CELL-DEVS.

III. OUR APPROACH

After analyzing DEVS and MAS' good points and weaknesses, it is important to remember the final objective of this project: to realize a M&S platform to study population dynamics.

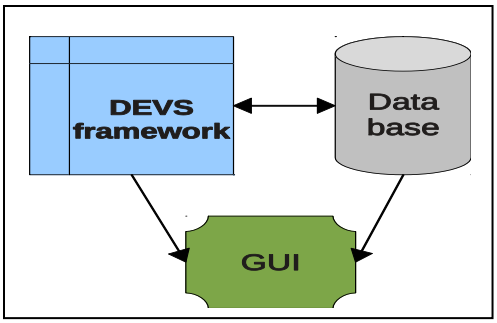

Figure 2. Simplified approach.

Fig. 2 points a global vision of our DEVS based architecture. Agents are DEVS models interacting with a database. Simulation results are displayed by an external tools (viewing tool). It's the first step of our project: describe MAS from DEVS model.

Our goal is to propose a coupling between these two modeling approaches in order to keep their advantages: flexibility and opening (in Information technology) for DEVS formalism and good living organism's representation and their interactions for MAS.

To modelize living organisms and their interactions, the tool must allow:

- To create and destroy agent during simulation;
- To modify variable during simulation;
- To have a graphical and dynamic representation of models' evolution;
- To follow evolution thanks to curves;
- To save simulation's results in database or file;

In this case, we need:

- Data on species studied (by simulation or by levy and field study);
- A modeling environment;
- Decision aid tools.

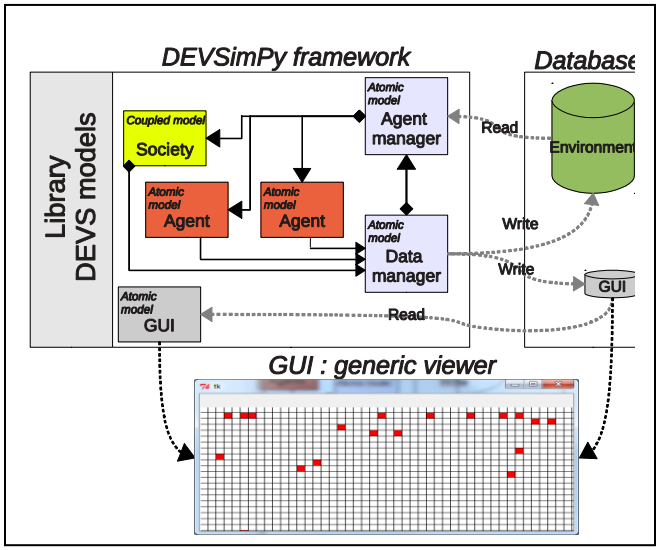

Figure 3. AgentDEVS platform.

Fig. 3 represents our platform architecture. It is composed of a DEVS model library including in the DEVSIMPY framework and two databases. The first database contains all data and the second is simpler and contains only ants' location used to make the graphical interface.

DEVSIMPY framework is composed by various atomic and coupled models. We detail these models:

- One atomic model "Agent manager" is charging to read and transmit information from database to agents (send answer) and it is able to create and destroy agents;
- One atomic model "Data manager" to manage agents output (it receives their messages) and to update the two databases (a full database and another allowing

to display information). It can also transmit directly a message to the Agent manager;
- N atomic agent models with their own behavior and N coupled society models to gather agents with same behavior;
- An atomic model to allow graphical display.

Data side contains:
- A database containing all information;
- A small database containing only information to display.

To represent an agent in its environment with DEVS formalism, we use atomic models. We have one atomic model for each agent and one atomic model for the environment. Environment is also a supervisor and there are coupled models to represent society and agents' group (Anthill for ant or shoal). Another approach was defined in [19] with a hybrid simulation platform called GALATEA. This platform seems to slow in simulation time and unsuited to utilization that we wish have.

The chosen way of communication is shared memory (Blackboard). After studying and thoughts it seems that it is the most adapted way of communication to couple with DEVS. DEVS being hierarchical, there are a lot of messages outstanding. So, using an associated data file (here blackboard), or a database, allows to limit the number of messages (towards high level model: environment), and avoid concurrent access issues to data (files are generated by high level model and information are centralized). Using communication by messages, it would have been very difficult to manage important flow of messages.

Other issues to consider are:
- File management;
- Messages' format;
- Dynamic agent destruction and creation;
- Graphical interface.

The issue of file management is in first on the file format used. There are lots of data to represent and we suggest using a XML file or Netcdf. This kind of file can represent data easily and clearly. We have also thought on a unique file reading issue to graphical display and we propose solution: to have two files. One has only agent's location and identification and is use only to graphical display: it's a CSV or excel file to represent a grid and the environment. And the other is a XML or Netcdf file to represent all data. The other advantage of file management with one model is data centralization and so we don't have concurrent access issues.

In MAS, message's format is defined by a standard: FIPA-ACL. The minimum message is: type of send messages (syntaxes), message sender, message receiver, message content. But often the minimum message isn't enough to communicate. We may need to indicate others information like used language in message content, used protocol, message's ontology, reference to an earlier message and reference to a conversation. Then message must be transmitted by a Message Transport Service (MTS) by communication channel (Agent Communication Channel: ACC).

In the DEVS formalism, message is composed as following: port, time, value. When compared these two messages, we could say that "port" in DEVS could be "sender" and "receiver" in MAS. "Value" could be "content message". We did not define remaining parameters yet. To avoid agents' creation and destruction issues we suppose that we can use a DEVS' extension. This extension allows create and destroy agents dynamically with a manager [8–10], [16]. The manager is our environment model with a supervisor role.

The final goal being to represent MAS in DEVS model, it is essential to use a graphical interface to display data. We plan to implement a coupling with Google API (Google map to display data on a map world) and use a viewer to display data in a board.

IV. CASE OF STUDY

In order to validate our approach, we chose to work on ant modeling and simulation.

One of the most used models in Multi Agent System (MAS) is the example of anthill. We will try to transcribe this ant model with DEVS formalism. In Multi agent system, an ant is an agent and works alone or with others agents to modify the environment (Fig. 4). Environment represents a spot where agents can interact, is a kind of grid. In a MAS, they are various kind of agents, we used reactive agent. It reacts to an exterior stimulus and does an action when it gets this stimulus. With DEVS, we can represent this with a received message on the entry port of a model. Every reactive agent is an atomic model.

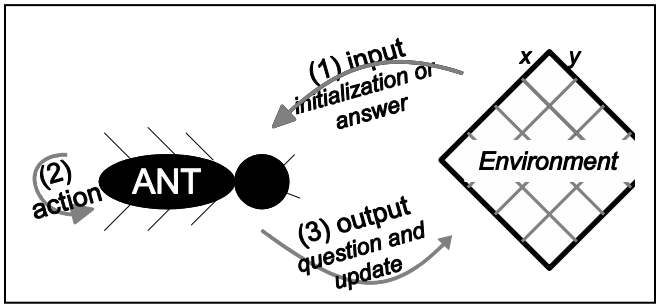

Figure 4. Agent communication with environment.

Fig. 4 points an external stimulus or initialization from environment (1). Then ant makes an action or not (2) and sends a message to the environment to ask questions and update information (3).

A. Ants' behavior description

Global behavior of an ant is made with various independents' behaviors or with behaviors Linked. Behaviors we have retained are:
- To look for food;
- To collect food;
- To drop pheromones;
- To back to the nest (brining food);

- To drop food;
- Possibility to follow pheromones.

An ant check out of the nest and look for food, it makes a random motion around the nest. When it founds food, it eats and goes back to the nest to drop it. During return, it drops pheromones on the way which are chemicals agents able to show the way to follow to find food. After have dropped the food, ant choose to follow pheromone or looking for others sources food.

B. MAS' models

In our model, ant has only one parameter, its energy. Environment have location of food source, nest and others variables like the number of food in the nest and the number of pheromone on a cell. At model initialization, the environment creates many food sources and the nest. Then agents are created with a default position: nest location. First, agents move randomly around the nest at each time step. Ant sends message to the environment to tell him their locations. Environment tests if ants' locations are equal to others object's location like food and sends him the result.

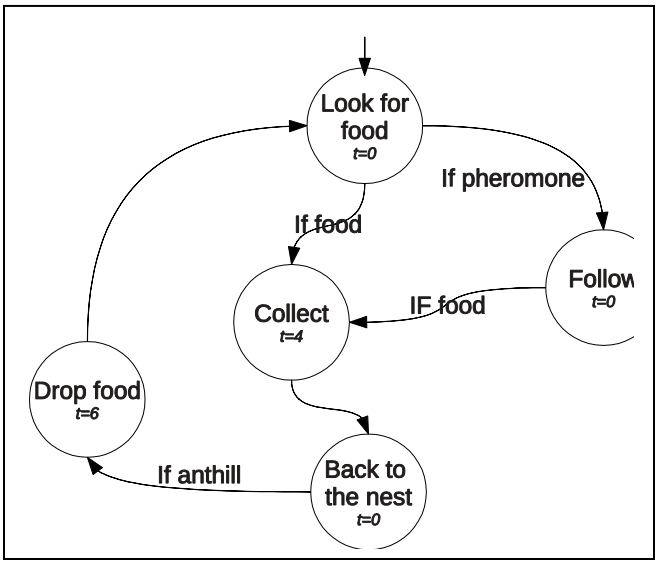

Figure 5. Ants DEVs behavior.

Ant receives message and depending on the result (Fig. 4 and Fig. 5); it keeps its behavior (looking for food) or change (eating food). When it finds food, it sends a message to the environment to inform about the food eaten. Environment knows how many foods remains on the cell. Once the food eaten ant changes its behavior and passes in «back to the nest» step. It drops pheromones (it sends a message to the environment for each cell through and environment adds pheromones on that cell). Then ant drop food in the nest and may choose to follow pheromones if the scent is strong enough (if many ants drop pheromone on a same cell, that cell smells more and catches more ants) or move randomly to find others food sources.

C. DEVS models

The Research team intends to make a link between agent behavior and a DEVS model.

Fig. 5 represents ants DEVS behavior. Message is like following: location x, location y, energy, food, and pheromone. A message involves automatically a data update and an answer.

1) Ant atomic model

The sets of states about our system are S: looking for food collects food, back to the nest (bringing food), drop food and follow pheromones. To start the system's state is 'Look for food'. According to this state and input messages it is changing its state variables (position x, y) and its state and sends message to update the database or query the environment.

The internal function δ_{int} is the way to allow the ant to change its behavior and so change state.

The externals functions δ_{ext} are messages from environment (here, if they are foods or pheromones).

The output function λ is message send by ant to environment (here, its location and number of food eat).

The time advance t_a function is evolution of time like «tick» in Multi Agent System.

Input: X = location of neighbors object and object type (MSG.x,MSG,y,MSG.type)
Output: Y = x, y, id, energy, pheromone, food
Time: Sigma = 0
State: Looking for food, collect food, back to the nest (bringing food), drop food and follow pheromones.
State variable: x, y, food, energy, id

Initialization function:
S = LookForFood()
X=Y=Food = 0
Energy = 100
Id = UNIQUE
Sigma = 0

Output function: λ
MSG.id=ID
MSG.x =x
MSG.y=y
MSG.energy=energy
MSG.pheromone=False
MSG.food=0
Send(MSG)

Time advance function: t_a
return Sigma

Behavior function: δ_{ext}(MSG) :
if MSG.type== food et S!= bringingFood:
    S = EatFood // we change state to EatFood
    energy+=10 // it collect food so it gain energy

X=MSG.x // ant go on the cell with food
Y=MSG.y // ant go on the cell with food
food+=20 // ant increases number of food
Sigma=4 // time to do action
if MSG.type==pheromone et s!=bringingFood:
S =follow // ant change state to follow
X=MSG.x // ant go on the cell with pheromone
Y=MSG.y // ant go on the cell with pheromone
Sigma =0 // time to 0 to continue motion
if MSG.type==anthill et s==bringingFood:
S=drop // we change state to Drop
Increase number of food in the anthill (MSG.food=food)
food=0 // food is already drop
X=MSG.x // ant go on the anthill
Y=MSG.y // ant go on the anthill
Sigma=6 // time to drop food

Behavior function: δ_{int}() :
if s==drop
S=LookForFood
if s==eatFood
S= BringingFood
if s==LoofForFood
X=random
Y=random
if s== BringingFood
X=random // towards anthill
Y=random // towards anthill
drop pheromone (MSG.pheromone=True)
if s==follow
X=follow pheromone
Y=follow pheromone
Sigma=0

2) Environment atomic models

Environment atomic models manage pheromones, food source.

Manager's agent model receives ant's location, if ants drop pheromone (with a Boolean) and if food was eaten. In output it sends to the ants if they are on food source and pheromones location.

Manager's data model updates data.

D. Simulation and results

Our models allow simulate an ants' basic behavior. Its aim is the validation of our theoretical approach. This example must be improved to be realistic.

V. CONCLUSION ANS PERSPECTIVES

In this paper, we proposed an original approach to associate DEVS formalism and MAS. Our approach seems similar to VLE works but these are limited to an environment representation made with cellular grid. We do not want to be limited by this mode of representation.

We propose a new method based on DEVS. We chose an easy study case, ants' evolution, to focus on the interests of our solution. Ant society modeling is a famous example in MAS. We have chosen this example to test our approach and to propose a tutorial application even if it's far to our final application.

Various issues happened: an important increase in messages that were sent between agents and environment (slow down the simulation); the need to dynamically create or destroy agents (dynamic DEVS [8]); the database management from atomic model and the display tool development.

Born of recent work [24], [25] our interest on MAS has become a real need. Moreover it perfectly integrates to the development of our platform (DEVSIMPY). Indeed, in Mediterranean, water availability and quality, as well as and biodiversity evolution represents a real issue. This is why development activities and land use planning must consider the water management and water systems. To answer to these issues, the STELLA MARE project (« Sustainable TEchnologies for LittoraL Aquaculture and Marine Research » was developed at the University of Corsica. Its objectives are large and it must be a scientific research center in innovation and appreciation of Mediterranean resources.

We wish to propose a generic software environment based on discrete event simulation principals (DEVS) and MAS described in Fig. 6. This environment must allow simulating fish resources evolution.

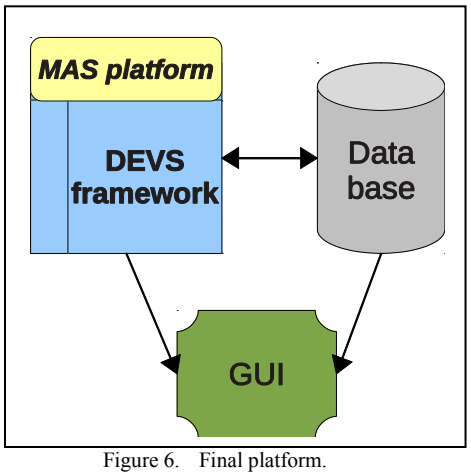

Figure 6. Final platform.

Fig. 6 points our final architecture. At time, we suggested proposed a method to describe MAS from DEVS model. The next step is completing DEVS framework with a MAS platform. The objectives of this platform are to make easier agent creation without describing atomic models behaviors.

Our research perspectives are various with the DEVS formalism and we plan to work on simulation algorithm haste and on cognitive agents' models and on agents and environment's interaction and communication. We need also visualization tools development (viewer).

Our final objective is to define agent models autonomous and intelligent and able to interact together. So, they will be influenced by their environment based on current models.

 ISBN: 978-1-61208-222-6

REFERENCES

[1] J.-B. Filippi, F. Bernardi, and M. Delhom, « The JDEVS environmental modeling and simulation environment », *IEMSS, Integrated Assessment and Decision Support, Lugano Suisse*, pp. 283–288, 2002.

[2] L. Capocchi, F. Bernardi, D. Federici, and P. Bisgambiglia, *Transformation of VHDL Descriptions into DEVS Models for Fault Modeling and Simulation*. 2003.

[3] P.-A. Bisgambiglia, E. de Gentili, P. A. Bisgambiglia, and J. F. Santucci, « Fuzzy Simulation for Discrete Events Systems », in *Proceedings of the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008) - IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 688–694, 2008.

[4] S. Garredu, E. Vittori, J.-F. Santucci, and D. Urbani, « A methodology to specify DEVS domain specific profiles and create profile-based models », pp. 353–359, 2011.

[5] B. P. Zeigler, *Theory of Modeling and Simulation*. Academic Press, 1976.

[6] B. P. Zeigler, *Multifaceted modelling and discrete event simulation*. Academic Press, 1984.

[7] J. Ameghino, A. Troccoli, and G. Wainer, « Models of complex physical systems using Cell-DEVS », pp. 266–273.

[8] F. Barros, « Dynamic structure discrete event system specification: a new formalism for dynamic structure modelling and simulation », in *Proceedings of Winter Simulation Conference 1995*, 1995.

[9] A. M. Uhrmacher and B. Schattenberg, « Agents in Discrete Event Simulation », in *Proceedings of ESS98*, 1998.

[10] A. Uhrmarcher, « Dynamic Structures in Modeling and Simulation: A Reflective Approach », *ACM Transactions on Modeling and Computer Simulation vol. 11 2001*, pp. 206–232, 2001.

[11] P. Fishwich and B. P. Zeigler, « A multi-model methodology for qualitative model engineering », *ACM transaction on Modeling and Simulation, vol. 2*, n°. 1, pp. 52–81, 1992.

[12] G. Weiss, *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[13] M. Wooldridge, *An Introduction to MultiAgent Systems*, Wiley and Sons. Chichester, West Sussex, Angleterre: Wiley and Sons, 2002.

[14] J. Ferber, *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*, Addison Wesley Longman. Addison Wesley Longman, 1999.

[15] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation, Second Edition*. 2000.

[16] F. Barros, « Abstract simulators for the dsde formalism », in *Proceedings of WSC 1998*, pp. 407–412, 1998.

[17] A. C. Chow and B. P. Zeigler, « Abstract Simulator for the Parallel DEVS Formalism », in *Proceedings of AIS94*, 1994.

[18] P.-A. Bisgambiglia, P. A. Bisgambiglia, and J.-S. Gualtieri, « Cognitive simulation-based on knowledge evolution in fuzzy discrete event systems », pp. 895–901, 2011.

[19] J. Davila and M. Uzcategui, « GALATEA: A multi-agent, simulation platform », in *In International Conference on Modeling, Simulation and Neural Networks MSNN'2000*, Mérida, Venezuela, 2000.

[20] R. Duboz, D. Versmisse, G. Quesnel, A. Muzzy, and E. Ramat, « Specification of Dynamic Structure Discret event Multiagent Systems », in *Agent-Directed Simulation (ADS 2006)*, Huntsville, AL, USA,, 2005.

[21] E. Ramat and P. Preux, « "Virtual laboratory environment" (VLE): a software environment oriented agent and object for modeling and simulation of complex systems », *Simulation Modelling Practice and Theory*, vol. 11, n°. 1, pp. 45–55, March 2003.

[22] G. Quesnel, R. Duboz, and É. Ramat, « The Virtual Laboratory Environment – An operational framework for multi-modelling, simulation and analysis of complex dynamical systems », *Simulation Modelling Practice and Theory*, vol. 17, n°. 4, pp. 641–653, April. 2009.

[23] L. Ntaimo and B. P. Zeigler, « Expressing a Forest Cell Model in Parallel DEVS and Timed Cell-DEVS Formalisms », 2002.

[24] D. Urbani and M. Delhom, « Water Management Using a New Hybrid Multi-Agents System - Geographic Information System Decision Support System Framework », pp. 314–319, 2006.

[25] D. Urbani and M. Delhom, « Analyzing knowledge exchanges in hybrid MAS GIS decision support systems, toward a new DSS architecture », in *Proceedings of the 2nd KES International conference on Agent and multi-agent systems: technologies and applications*, Berlin, Heidelberg, pp. 323–332, 2008.

 ISBN: 978-1-61208-222-6