Agent-Oriented Approach Based on Discrete Event Systems

Paul-Antoine Bisgambiglia and Paul Antoine Bisgambiglia and Romain Franceschini
University of Corsica, UMR SPE 6134 CNRS, UMS Stella Mare 3460
TIC team
Campus Grimaldi, 20250 Corti
bisgambiglia@univ-corse.fr and bisgambi@univ-corse.fr

Keywords: Discrete event simulation, DEVS, Multi Agent System, MAS, cellular network, Agent-based simulation, group behavior

Abstract

Inspired by existing works, in this paper, we propose software architecture for coupling the MAS and DEVS formalism. This architecture is designed to enable the modeling of large quantities of agents. Previous works often associated with an agent to a DEVS model. The complexity of the system increases proportionately with the number of agents. We describe an approach to group a set of agents in the same DEVS model, and allow representing many more agents.

1. INTRODUCTION

In the field of Discrete Event Systems (DES [1]), recently, many efforts have been devoted to develop appropriate tools to study, and model in a formal way the dynamics and the mechanisms of interaction of the natural systems [2]–[10]. Discrete Event Systems are a special type of dynamic system. The 'state' of these systems changes at discrete instants in time and the term 'event' represents the occurrence of discontinuous change. For several years the community is changing the DEVS formalism [11] so that it can become a powerful tool for modeling complex systems and thus living systems. In this paper, an agent oriented approach based on the discrete event system specification (DEVS) formalism is proposed. This study aims to develop a multi-agent-based simulation system to evaluate the evolution of group of individuals. Agent-based Modelling is a very efficacious conceptualization paradigm that easily allows the representation of very complex systems composed of autonomous, possibly intelligent, interacting entities. For this reason, Agent-based systems have been incorporated into simulation framework as a mean of effectively increasing the realism through adaptation and learning abilities. The integration between agent and simulation, namely Agent-based simulation, however, can be further exploited by modeling the simulation system as a Multi Agent System (MAS). Simulation of agent-based systems is an inherent requirement of the development process which provides developers with a powerful means to validate both agents' dynamic behavior and the agent system as a whole and investigate the implications of alternative architectures and coordination strategies.

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 [12], [13]. 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]. 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.

Our problem is to provide a consistent and efficient software structure to combine the discrete event systems modeling and multi-agent systems. This structure should allow modeling a large number of agents and their evolution over time. In these fields other work has been done [5], [6], [14] but the proposed approaches, we do not seem able to describe the evolution of several hundred individuals. The platform GALATEA [5] 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. This platform would allow to model different formalisms, in different languages in a same interface. But this implementation seems tedious and difficult. The second approach [6] matches a lot with our works. An agent is represented by an atomic model and stimuli by exterior messages [9], [14]. But using of VLE framework [15] to simulate and using of Cell-DEVS formalism [16] 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).

We present architecture for managing agents based on the cellular network model (Figure 1) for managing and location the subscribers [17], [18]. We give you all the details of our architecture in the following parts, it is built like a cellular network, the network is an environment and the agents are the users. The problems are as follows:

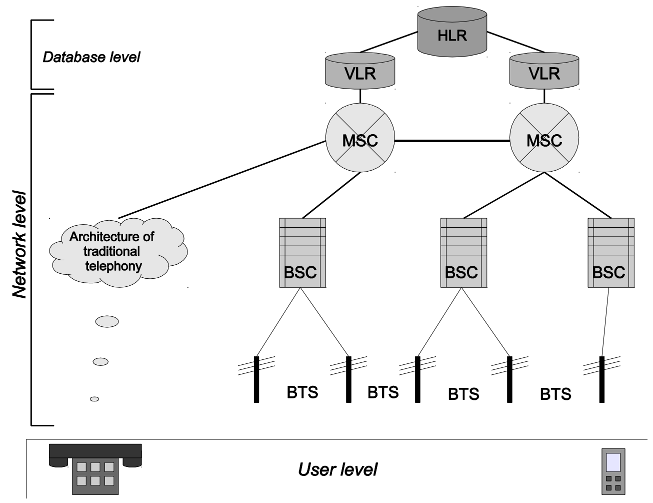

Figure 1. Mobile network (GSM) architecture

1) The network must be able to identify the subscriber. 2) The network must know the location of subscribers, subscribers who are numerous and move. Cutting the network into multiple cells must be invisible to the subscriber. 3) The network must ensure the continuity of the communication even when the subscriber moves.

1) Location databases called HLR (Home Location Register), contains profiles for each user (subscribed services and data authentication). HLR equipments are limited in capacity. There are several HLR in a network but a given subscriber is still drive by the same HLR, regardless of where it is located. 2) In order to reduce data exchange, we associate a local database VLR (Visitor Location Register) each MSC (Mobile services Switching Centre). The MSC is the heart of the GSM network. It handles call routing, call setup, and basic switching functions. An MSC handles multiple BSCs and also interfaces with other MSC's and registers. It also handles inter-BSC handoffs as well as coordinates with other MSC's for inter-MSC handoffs. The VLR contains the profiles of all subscribers who are in the area managed by the MSC at any given moment. A customer who moves is taken into account by the various successive MSC and VLR. MSC are composed of several BSC (Base Station Controller). 3) The BSC controls a set of BTS (Base Transceiver Station), and it manages radio resource. According to the quality of the radio link, of the current communication, the BSC ensure the continuity of communication and jumps BTS to BTS. BTS are transceivers (antennas). They can locate several subscribers (MS: Mobile Station), the area it covers is called a cell. Terminals running on networks (subscribers) are called MS.

In the first part, we present the DEVS formalism. This formalism may be defined as a universal, general methodology which provides tools to model and simulate systems, the behavior of which is based on the notion of events. This formalism is based on the theory of systems and the notion of the model and permits the specification of complex discrete event systems in modular and hierarchical form. In the second part, we will detail our architecture. Finally, before concluding, we present a pedagogical example, application of our approach.

2. DEVS FORMALISM

Since the 1970s, formal tasks have been performed to develop the theoretical foundations of modeling and simulating discrete event dynamic systems. Major efforts have been made to adapt this formalism to various domains and situations [16], [19]–[24]. Our interest focuses on the DEVS formalism (Discrete Event System Specification [11], [19]). 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.

2.1. DEVS models

DEVS is a modular formalism which permits the modelling of causal and deterministic systems. A DEVS atomic model is based on continuous time, inputs, outputs, states and functions. More complex models are constructed by connecting several atomic models in a hierarchical way.
The atomic model may be considered as a time-based state machine. It makes it possible to describe the system's functional or behavioural aspects. The atomic model provides an independent description of the system's behaviour, defined by the states and functions of the inputs/outputs and by the model's internal transitions. The model develops by changing its state according to external stimuli (via an input) or internal stimuli (via a transition function). The purpose of these changes in state is to determine the system's behavioural response to these stimuli. An atomic model is described by the following formula: MA : $< X; Y; S; ta; \delta ext ; \delta int ; \lambda >$, with:

- $X = \{(p_{in}, v) \mid p_{in} \in$ Input ports, $v \in X\ p_{in}\}$: the list of the model inputs, each input being characterised by a coupling (port/value number);
- $Y = \{(p_{out}, v) \mid p_{out} \in$ Output ports, $v \in Y\ p_{out}\}$: the list of the model outputs, each output being characterised by a coupling (port/value number);
- S: all the system's states or state variables;
- ta: $S \longrightarrow R+$: the time advancement function or the S state's lifetime;
- δext: $QxX \longrightarrow S$: the external transition function, where:
  - $Q = \{(Si, e) \mid Si \in S, 0 \leq e \leq ta(Si)\}$;

– e: is the time which has elapsed since the last transition. The external transition function specifies how the atomic model changes state (passing from state S_1 to state S_2 when an input takes place (external event before ta(S_1) has elapsed;

- δint:S⟶S: the internal transition function. It permits passing from an S_1 state to the t_1 instance, or from an S_2 state to the t_2 instance, when no external event occurs during the lifetime of the ta(S_1) state;
- λ:S⟶Y: the output function.

The DEVS formalism uses the notion of a description hierarchy, which permits the construction of models called "couplings", based on a collection of atomic models and/or couplings, and on three coupling relations. A DEVS coupled model is modular and displays a hierarchical structure, which permits the creation of complex models based on atomic and/or coupled models. It is described by the following formula: MC : < XM; YM; CM; EIC; EOC; IC; L >, with: XM: total of input ports; YM: total of output ports; CM: the list of models making up the CM coupled model; EIC: the total of the input couplings, which links the coupled model to its components; EOC: the total of the output couplings, which links the components to the coupled model; IC: the total of the internal couplings, which links the components to one another; L: a list of priorities among components.
A coupled model makes it possible to describe how several other models are interconnected in order to form a new one.

2.2. Discrete event simulation

In order to define the simulation semantics of the DEVS models, Zeigler put forward the abstract simulator notion. The main advantage of this concept is the difference between the models and the simulator. At the level of this simulator (abstract), each simulation component corresponds to a modelling component. In fact, DEVS is one of the rare "formal" formalisms which propose an implementation algorithm. According to [11], a coupled model is composed of atomic and/or coupled models. The abstract simulator is composed of a root coordinator, coordinators and simulators. The corresponding algorithm is described in [2] and [19].

2.3. DEVS extensions

In this section, we present the works on which our approach is based.
DEVS BUS is an extension of the DEVS formalism based on HLA [26], [27]. It proposes a layered architecture for easy distribution simulations.
Dynamic DEVS is an extension of the DEVS formalism that allows taking into account the evolution of the structure of the system during the simulation. Several approaches have been proposed [9], [28]–[31].
Cell DEVS is also an extension of DEVS formalism that allows to represent a geographical area in a grid cell, each cell is described as an atomic model [32], [33]. This representation is quite heavy because there are so many models and exchanged messages. Improved versions have been proposed on the basis of a flattening simulators or parallelization and/or distribution of simulations [21], [34]–[36].

2.4. DEVS frameworks

There are many environments based on the DEVS formalism, we can cite [37]–[39]. Our team works on DEVSimPy. DEVSimPy framework [22] 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 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. It is based on the simulation engine Python DEVS [40], which implements the algorithms of classical DEVS [11]. 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. ARCHITECTURE BASED ON CELLULAR NETWORKS

This section presents a paradigm to describe cellular model in DEVS formalism. In [33] formal descriptions of Cell DEVS allow describing a cellular model as DEVS. A space is composed of individual cells define as atomic models that can be lately afterward coupled in a coupled model. Each cell as an interface composed with a fixed number of ports, connected with a neighbor. An analogy between Multi Agent Systems and the Modeling and Simulation theory using dynamic structure was described in [6], [14]. Performing a direct simulation using the DEVS formalism can be inefficient for models comprised of various numbers of sub-models such as those based on Cell spaces or/and MAS. In order to tackle increasing complexity and to improve performance of DEVS simulation, some studies suggest the use of a non-hierarchical DEVS simulator [29], [33], [34] and some other solutions proposed to exploit the parallelism existing in the system by associating more than one processor to the simulation [7],

[10], [21], [33]. In a non-hierarchical (or flat simulation) there is only one coordinator in charge of all the simulation tasks of model's atomic components. By eliminating the intermediate coordinators in the hierarchy, the flat simulation reduces the number of exchanged messages between processors. However, when the number of agents increases, communications among agents and environment produces a high degree of overhead. In order to accelerate simulation time computations, parallel and distributed techniques [7], [15], [33], [41] have been developed. The general approach taken by parallel or distributed simulation is often based on the logical process [42] executing on multiprocessor systems. A Logical Process (LPs) is a subparts of the system coded as a coupled model and linked to other LPs by timestamped messages.

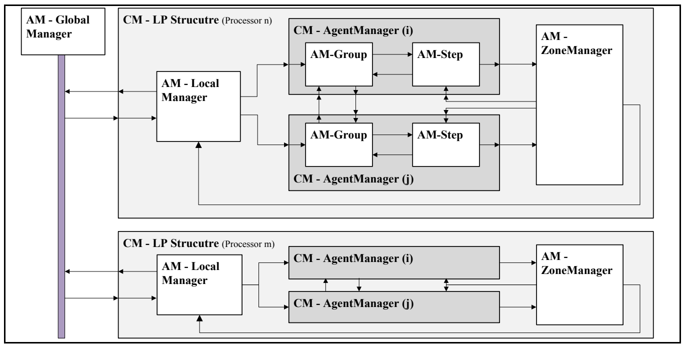

Figure 2. Global Architecture based on DEVS models

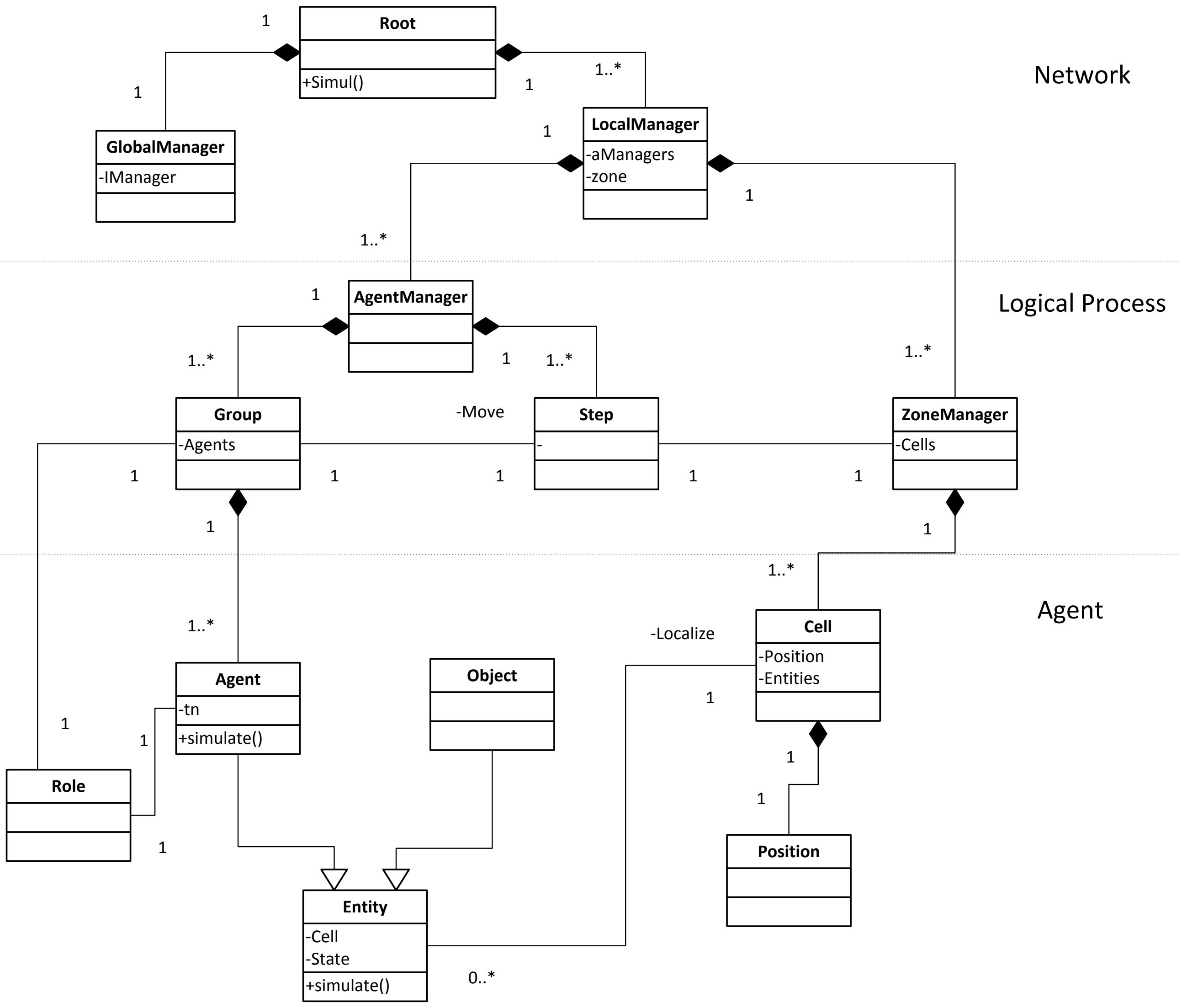

Figure 3. MAS class diagram

We propose a new technique for fast execution of MAS models based on existing approach. There are two main differences when comparing our approach with the previous simulation presented here, namely that agents and cells are not considered as a DEVS model. Indeed, when the number of agents is growing it is then necessary to introduce an intermediate level of simulation. We focused on homogeneous groups in which all individuals agent have the same social role (set of agents satisfying distinguished properties, description or behavior) and on an environment divided into zones, where a zone is defined as a geographic area represented by a grid cells. In the architecture A logical process is use to handle one zone as shown in figure 2. So each zone represents a sub-part of the global system and they can be connected to one or more other zone. The DEVS/HLA simulation [43], [44] provides communications between LPs, as well as the whole simulation time management method between distributed simulations.

To improve performance, the proposed solution is implemented in a distributed fashion, where on non-hierarchical coordinator will be associated to a logical processor and will interconnect and synchronize through HLA architecture. A “logical process” is a coupled model that manages the groups of agents localized in the specified area controlled by the atomic model “zone manager”. Each “zone manager” as also a set of border cells will have different behavior than those of the rest of the area. Each cell contains information about agents and dynamics or “persistent” objects currently located in that cell. As an agent moves through the grid it will be insert or remove from the cell’s list as appropriate in the same zone manager. When an agent moves through the border, it will be removed in the current model and transferred by the “local manager” to the corresponding area. A “local manager” is coded as an atomic model including input and output event list to communicate with another “local manager” using a neighborhood relationship. Agents are grouped according to their roles and stored in the atomic model “agent manager”, making it easier to manage larger units of entities that belong together. Agents can perform various roles within one system. Finally, the “step atomic model” is used to perform the steps.

Based on the discrete-event simulation framework given in this article the architecture of the simulation engine is composed of three basics layers. Network layer built atop the simulation framework DEVS, provides a distributed HLA infrastructure formed by a network of interconnected logical processes (Figure 3). The logical process layer provides abstractions representing users’ behaviors. The agents’ layer provides the basic class to simulate the appropriate system.

Our architecture is based on cellular networks (GSM). We have simplified some aspects because we do not encounter exactly the same problems, such as secure communications. The GSM architecture is hierarchical, the lowest level there are the mobile stations, our entities, then base stations (BTS), our zone managers. At superior level we find the controller (BSC), group managers and finally the MSC, is our local managers.

4. APPLICATION EXAMPLE

We present a simple ant colonies model with only one logical process and two groups of ants (soldier and worker). For clarity and simplicity reasons, we do not give her a global formalization of the system. The behavior of ants and a version of the algorithm of DEVS transition functions are given in [45]. Workers gather food for the colony; soldiers defend the colony and fight off enemy ants or insects. Ant’s actions are limited to search for food or enemies by working randomly, follow a pheromone trace, bring food to the nest, and fight. When foraging or find enemies, ants leave a pheromone trail so that they know where they’ve been. Soldiers do not look for food and survive only by eating food carried by workers. If density of available food is too low, foraging efforts is growing, inter-relation between atomics models groups permit to transit some agents from the soldier group to the worker group. If the danger increases workers are switch to the soldier group. Figure 4 shows an agent diagram that reflects the way in which ants evolve. The structure shows two agents class, which has can sense food.

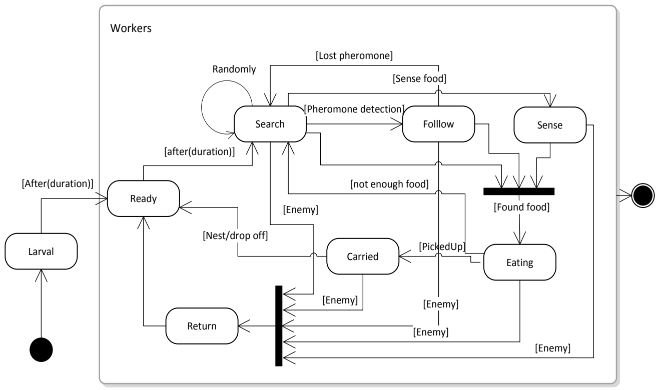

Figure 4. Worker’s behavior

We suppose that ants can sense several pheromone signals over a distance. A worker moves to the source if it smelt food, and return directly to the nest if it smelt enemy, depositing a pheromone trail to mark out dangers as it goes (Figure 4). A soldier seeks only enemies and fight to the death to protect his colony (Figure 5). A defeated ant emits an alarm pheromone that attracts more soldiers from farther away and leads others to give warning. The role of ants may change with the circumstances. Different types of

pheromones are represented as objects placed in a cell of the zone (Figure 3).
The birth rate is a base rate, which has been recalculated at each step depending the food was dropped off at the nest by the workers. Soldiers return to the nest for food, and reduce quantity of food. In addition, ants can die due the base death rate initialized with the program and due the fight.

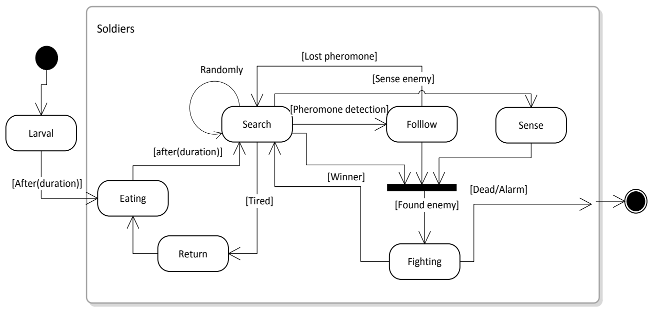

Figure 5. Soldier's behavior

For simplicity reasons, in the example show her, the simulation period has been divided in three distinct parts (Figure 6). The first part covers the period when the workers are seeking food, figure 6: simulation time 0 to 31. During this period as the food resource decreases due to consumption, the number of workers increases. Part two occurred when food begins to arrive to the nest, the numbers of soldier's increases, figure 6: simulation time 31 to 50. At the end of this part, ants are born according to the base rate, figure 6: time 58. Part three covers the fighting period between clones and individuals, as Molly's son Mark grows up as the only 'single' in the group, figure 6 from the simulation time 70.

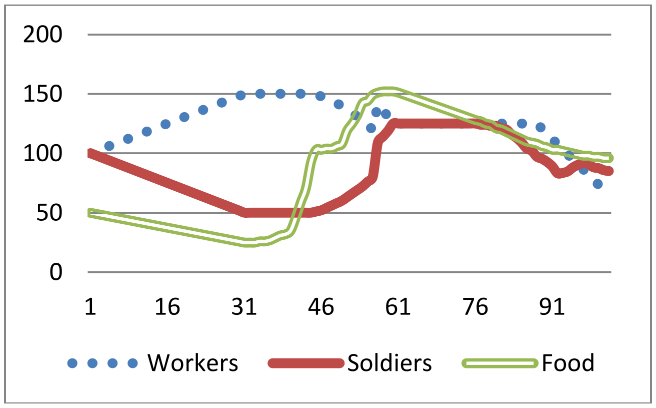

Figure 6. Simulation results (the horizontal axis represents simulation time, and the y-axis the number of ants)

In figure 6, the first section covers the period when the cloning facilities are being set up against a background of a world society in the throes of collapse. Part two is a look after several clone generations have occurred and an expedition is made to one of ruined cities to salvage needed high-tech supplies for the continuing cloning operation. The expedition exposes both the strength and the weakness of the clone groups, as they find it almost impossible to remain sane when separated from their clone 'brothers' and 'sisters'. One expedition member, Molly, grows so far away from her sisters under the stress that she really becomes an individual. Part three covers the fighting period between soldiers and enemies.
This pedagogical example was used to illustrate the proposed architecture.

5. CONCLUSION AND PERSPECTIVES

The starting point for this work begins with a simple observation: although there are DEVS extensions to allow agent-based simulation (reactive agents), none of these extensions can represent very large amounts of agents. In fact, these extensions are based on a representation of agents from DEVS atomic models. One agent is described by one model. This representation, although relevant, causes a very active communication between agents and between agents and environment. The management of this large quantity of messages reduces the number of agents modeled.

To solve this problem, we propose new software architecture. Its objective is to model a large number of agents. Our approach is inspired by the architecture of the GSM networks. It is based on the notion of an individuals' group. It uses and is also inspired other concepts introduced in extensions Cell DEVS, for the representation of cellular environment, dynamic DEVS, for creation or destruction of groups, and DEVS Bus for parallelization or distribution of the logical process. From our point of view, most of our approach is the representation as a group. A group is a set of agents with the same behavior or aim. An agent is not modeled by an atomic model; it is defined as a simplified atomic model without ports, and can autonomously run its behavior. It is a specific object that is different of a DEVS component.
We propose to manage agents from a coupled model called Manager and composed of two atomic models. In our approach one hundred agents with a same goal are modeled by two atomic models. The dynamic aspects are thus simplified because the manager can create or destroy agents without having to manage couplings. Our approach has been illustrated from MAS representing an ants society modelized with only one logical process. The ants are divided into two groups: Workers looking for food, then they return to the nest. The food allows the creation of new ants; Soldiers defend the nest and workers. This application allows showing the possible interactions between two groups.
Our perspectives are quite numerous, at first we want to represent intelligent agents. This improvement passes

through the addition of cognitive modules at the level of individual and collective. We will update the agent class and strategy. We wish to develop a configurable visualization interface. It will make the interface between GIS and our software. It will add groups of agents by drag and drop. In the longer term, we couple our software with a currents model, to test our approach on a large scale and thus simulate the evolution and spread of post larvae.

6. BIBLIOGRAPHY

[1] C. G. Cassandrass et S. Lafortune, *Introduction to Discrete Event Systems*. 1999.

[2] D. Gianni, « Bringing Discrete Event Simulation Concepts into Multi-agent Systems », 2008, p. 186‑191.

[3] J. Košeckà et R. Bajcsy, « Discrete Event Systems for autonomous mobile agents », *Robotics and Autonomous Systems*, vol. 12, n° 3‑4, p. 187‑198, avr. 1994.

[4] F. Capkovic, « Cooperation of autonomous agents based on supervisory control of DES », 2010, p. 178‑183.

[5] J. Davila, E. Gomez, K. Laffaille, K. Tucci, et M. Uzcategui, « MultiAgent Distributed Simulation with GALATEA », 2005, p. 165‑170.

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

[7] K. Kim et K. J. Kim, « Multi-agent-based simulation system for construction operations with congested flows », *Automation in Construction*, vol. 19, n° 7, p. 867‑874, nov. 2010.

[8] D. Degenring, M. Röhl, et A. M. Uhrmacher, « Discrete event, multi-level simulation of metabolite channeling », *Biosystems*, vol. 75, n° 1‑3, p. 29‑41, juill. 2004.

[9] A. Uhrmacher, J. Himmelspach, M. Rohl, et R. Ewald, « Introducing Variable Ports and Multi-Couplings for Cell Biological Modeling in DEVS », 2006, p. 832‑840.

[10] G. Fortino, A. Garro, et W. Russo, « A Discrete-Event Simulation Framework for the Validation of Agent-based and Multi-Agent Systems », presented at the WOA, 2005, p. 75‑84.

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

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

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

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

[15] G. Quesnel, R. Duboz, et 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, p. 641‑653, avr. 2009.

[16] J. Ameghino, A. Troccoli, et G. Wainer, « Models of complex physical systems using Cell-DEVS », p. 266‑273.

[17] S. Makki, « Tracking highly mobile users using replicated databases », *Computer Communications*, vol. 23, n° 10, p. 975‑979, mai 2000.

[18] V. Garg, « Mobility Management in Wireless Networks », in *Wireless Communications & Networking*, Elsevier, 2007, p. 369‑395.

[19] B. P. Zeigler, « DEVS Today - Recent Advances in Discrete Event-Based Information Technology », in *proc. of the 11th IEEE/ACM International Symposium on*, 2003.

[20] P.-A. Bisgambiglia, E. de Gentili, P. A. Bisgambiglia, et J.-F. Santucci, « Fuzz-iDEVS: Towards a fuzzy toolbox for discrete event systems », in *Proceedings of the SIMUTools'09, Rome (Italie)*, 2009.

[21] S. Jafer et G. Wainer, « Flattened Conservative Parallel Simulator for DEVS and CELL-DEVS », in *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 01*, 2009, p. 443‑448.

[22] L. Capocchi, J. F. Santucci, B. Poggi, et C. Nicolai, « DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems », in *Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2011 20th IEEE International Workshops on*, Paris, 2011, p. 170‑175.

[23] F. J. Barros, « Modeling formalisms for dynamic structure systems », *ACM Transactions on Modeling and Computer Simulation*, vol. 7, n° 4, p. 501‑515, oct. 1997.

[24] J. deLara et H. Vangheluwe, « Computer Aided Multi-paradigm Modelling to Process Petri-Nets and Statecharts », in *Proceedings of the First International Conference of Graph Transformation (ICGT)*, 2002, vol. 2505, p. 239–253.

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

[26] T. G. Kim et Y. J. Kim, « A Heterogeneous Simulation Framework Based on The DEVS Bus and The High Level Architecture », in *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[27] G. Zacharewicz et M. E.-A. Hamri, « Flattening G-DEVS / HLA structure for Distributed Simulation of Workflows », in *Proceedings of AIS-CMS International modeling and simulation multiconference*, Buenos Aires, Argentine, 2007, p. 11‑16.

[28] F. Barros, « Abstract simulators for the dsde formalism », in *Proceedings of WSC 1998*, 1998, p. 407‑412.

[29] F. J. Barros, « Dynamic structure multiparadigm modeling and simulation », *ACM Transactions on Modeling and Computer Simulation*, vol. 13, n° 3, p. 259‑275, juill. 2003.

[30] A. M. Uhrmacher, « Dynamic structures in modeling and simulation: a reflective approach », *ACM Transactions on Modeling and Computer Simulation*, vol. 11, n° 2, p. 206‑232, avr. 2001.

[31] J. F. Santucci et L. Capocchi, « Visualization of Folktales on a Map by Coupling Dynamic DEVS Simulation within Google Earth », in *SIMULTECH'11*, 2011, p. 128‑133.

[32] G. A. Wainer, « Modeling and Simulation of Complex Systems with Cell-DEVS », vol. 1, p. 45‑56.

[33] G. A. Wainer et N. Giambiasi, « Application of the Cell-DEVS Paradigm for Cell Spaces Modelling and Simulation », *SIMULATION*, vol. 76, n° 1, p. 22‑39, janv. 2001.

[34] E. Glinsky et G. Wainer, « New Parallel Simulation Techniques of DEVS and Cell-DEVS in CD++ », in *proc. in Annual Simulation Symposium*, 2006, p. 244‑251.

[35] Qi Liu, « Distributed Optimistic Simulation Of Devs And Cell-Devs Models With Pcd++ », 2006.

[36] G. Wainer, Q. Liu, et S. Jafer, « Parallel Simulation of DEVS and Cell-DEVS Models in PCD++ », in *Discrete-Event Modeling and Simulation*, vol. 20115630, G. Wainer et P. Mosterman, Éd. CRC Press, 2011, p. 223‑270.

[37] E. Ramat et 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, p. 45‑55, mars 2003.

[38] C. Jacques et G. A. Wainer, « Using the cd++ DEVS tookit to develop petrinets », in *Proceedings of the SCS Conference*, 2002.

[39] E. Kofman, M. Lapadula, et E. Pagliero, « PowerDEVS: A DEVS-Based Environment for Hybrid System Modeling and Simulation », 2003.

[40] H. L. Vangheluwe et J. S. Bolduc, « Pydevs ». McGill's, 2002.

[41] R. M. Fujimoto, « Distributed simulation systems », p. 124‑134.

[42] J. Nutaro et H. Sarjoughian, « Design of distributed simulation environments: A unified system-theoretic and logical processes approach », *Simulation*, vol. 80, n° 11, p. 577–589, 2004.

[43] B. P. Zeigler, S. B. Hall, et H. S. Sarjoughian, « Exploiting HLA and DEVS To Promote Interoperability and Reuse in Lockheed's Corporate Environment », *SIMULATION*, vol. 73, n° 5, p. 288‑295, nov. 1999.

[44] F. Kuhl, J. Dahmann, et R. Weatherly, *Creating computer simulation systems : an introduction to the high level architecture*. Upper Saddle River, NJ: Prentice Hall PTR, 2000.

[45] S. Mattei, P.-A. Bisgambiglia, M. Delhom, et E. Vittori, « Towards Discrete Event Multi Agent Platform Specification », presented at the COMPUTATION TOOLS 2012, The Third International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking, 2012, p. 14‑21.