Specification of Dynamic Structure Cellular Automata & Agents

Alexandre Muzy

UMR CNRS - LISA
Università di Corsica – Pasquale Paoli
Corti, ID 20250, France
a.muzy@univ-corse.fr

Xiaolin Hu

Computer Science Department
Georgia State University
Atlanta, GA 30303, USA
xhu@cs.gsu.edu

Abstract **–**This paper proposes a framework for dynamic structure cellular automata & agent (DSCA^2). Using such framework allows preserving modularity of components and modeling dynamic structural changes of components during the simulation. The formal and component-based treatment of DSCA^2 builds a solid ground for future applications to complex system modeling and simulation.

I. INTRODUCTION

Cellular automata modeling and agent-based modeling are two major paradigms to model and simulate complex dynamical systems. The cellular automata modeling includes a grid of cells where each cell's state can affect and be affected by its neighboring cells' states. It models spatiotemporal interactions and behaviors of a system. Examples of cellular automata modeling and simulation include urban environment simulation, forest fire simulating, and disease spreading simulation, etc. The agent-based modeling includes a group of agents that interact with each other and with an environment. Each agent has relatively simple behaviors and interaction rules, exhibiting emergent behaviors when working together. Examples of agent-based modeling and simulation are in many different fields, including social system simulation, software systems, traffic simulation, etc.

This paper concerns a modeling approach where cellular automata models work together with agent-based models and act as a spatial (physical) environment in which agents are situated. Within this context, we are particularly interested in how these models can also support dynamic structure modeling for complex systems. At a physical level, natural and software systems frequently change structure. Growing, a plant adds numerous behaviors and branches. In a computer network, nodes are dynamically added and deleted changing their behavior. At a software design level, allowing components to change (their or other) structures increases the complexity and the flexibility of the systems developed. Using such approach allows modeling more faithfully reality and opens huge research perspectives to modeling and simulation.

In this modeling approach, the environment is spatially modeled as a cellular space model composed of multiple cells. Each cell corresponds to a sub-area of the environment and has its own states. An example is a pedestrian crowd simulation where the streets are modeled by street cells and pedestrian are modeled as agents. In a more complex case, a cell may also have its own dynamical behavior. For example, in a forest firefighting simulation, forest cells are used to model the behavior of fire spread and agents are used to model the behavior of firefighters. The interaction between an agent and the cells (its environment) is supported by the couplings between the agents and cells. Dynamic structure of these models include the structure change of CA models such as adding/deleting cells, the structure change of agent models such as adding/removing agents, and the structure change of the connections between agents and cells. For example, when an agent moves spatially in the environment (the cellular space), the couplings between the agent and the corresponding cells are dynamically changed. This hybrid modeling approach separates the modeling concerns of agents and the environment (the cellular space). The dynamic structure capability can greatly enhances its modeling power by supporting adding/deleting cells or agents and their couplings. Furthermore, using such approach preserves component modularity thus enhancing model reusability. We name this modeling paradigm Dynamic Structure Cellular Automata & Agents (DSCA²) in this paper.

To leverage the power of DSCA² described above, it is important to treat both models and their connections in a formal and structural manner. In this paper, we propose a generic architecture to preserve modularity of components while letting them free to modify their structure themselves or to be modified during the simulation or to be specified by the modeler. We provide a specification for this dynamic structure cellular automata and agent modeling. While many specifications exist for CA and agent, most do not account for structural changes during the simulation. In this paper, we base our specification on the Discrete Event System Specification (DEVS) formalism. The remainder of the paper is organized as follows. Section II describes usual component-based approaches for CA modeling. Section III discusses the specification of a single agent and multiple agent system. Based on these discussions, Section IV provides a formal specification of DSCA^2. Section V presents related works and perspectives.

II. CELLULAR SYSTEMS

Figure 1 presents a usual component-based cellular system. To describe the latter different modularity and specification levels can be used. Couplings between cells include internal couplings [neighbourhoods (von Newman, etc.)] and external input and output couplings (cell external influences). Case (1) on figure corresponds to a non-modular case, case (2) to a modular one. When using ports,

many modularity choices can be achieved too. When using a discrete-event description of systems, transition functions of cells are decomposed in many sub-transitions, each one activated according to the port name *or* the event kind received as input. Ports can be highly modular, *i.e.*, to a single port name corresponds a single event kind. Otherwise, ports can be aggregated, *i.e.*, to a single port name correspond many event kinds. Let us take an example to explain this notion: The use of ports to account for wind influences in a fire spread system. The first solution will consist of adding two ports named "WindDirection" and "WindStrength". The first port can only carry values of wind directions, and the second port can only carry values of wind strength. In the second solution, we can consider a single port named "Wind", in which distinct values of directions and strength can be carried. In the first solution, a test on the name port will lead to the activation of the corresponding sub-transition function. In the second solution, a test on the event value will lead to the activation of the corresponding sub-transition function. Notice that both approaches can be mixed (as for system modularity), *i.e.*, cells could have ports named "Wind" and ports named "Water." Hence, both tests on port names and then event values will have to be achieved.

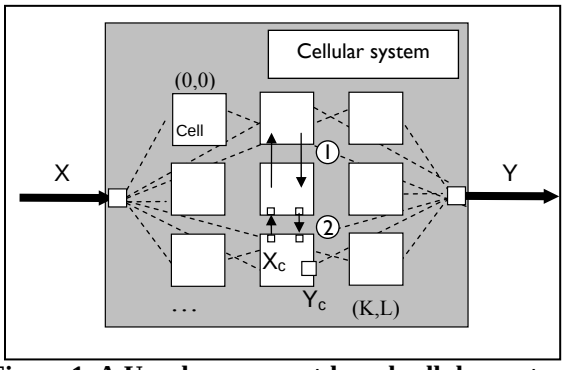

Figure 1. A Usual component-based cellular system

When modeling component-based cellular systems, according to modularity choices, three representative levels (corresponding to research directions) can be used. These levels can be described from the less to the more modular level:

1. States of a cellular system can be wrapped in a single atomic model fully autonomous and modular;
2. Usual multi-components in which external input ports can directly influence state transitions of cells [1];
3. Highly modular cellular systems consisting of a coupled model in which cells consists of atomic models fully autonomous and modular [2].

Beyond this modularity classification, higher specifications can be built on usual modular descriptions [3] or on a non-modular specification [4].

Notice that a cellular model could be a multi-agent system. However, we consider here agents as mobile independent components.

In the next sub-section we investigate a new research direction to integrate previous modularity and dynamic structure changes during the simulation.

III. ENDOMORPHIC AGENTS

Currently, as a new paradigm, artificial agents are constituted of many different structures and goal. Merging these different facets in a coherent single structure allows improving interoperability and facilitates understanding.

A. A Single Agent

What is an artificial agent? In (distributed or not) artificial intelligence, machine learning, economics, biology, control theory, and mind architecture, etc., plethora of structures and behaviors of computer systems are built to map "intelligent" behaviors and structures of animals or humans. However, what is an "intelligent behavior"? Is it planning, learning or adaptation? The notion is very broad and subject to interpretation.

We propose here an open mathematical structure framework to map these interpretations on computers to achieve simulation. Using these structures, a modeler will be able to build component blocks constituting their agent-based system. Both structure and behavior of components can be specified gradually. Using such component-based approach of modeling and simulation, multiple hierarchical compositions of components can be drawn. To precise the open structure in the hierarchy, let us draw frontiers and identify entities of an intelligent agent in modeling and simulation.

When we speak about agents, we need to speak about an agent interacting with an environment. This environment can be social and/or physical. In the latter case, it can reflect dynamics of a phenomenon, and/or simply consist of the physical space (physical objects).

To draw frontiers to agents, we can consider interactions between the agent and its environment. Using *a priori* knowledge and simulation, interactions of agents can be mental (what will be the consequences of that action in that configuration of the environment) and/or physical (an animal searching food). In both cases, an agent has to be able to build a model of itself and of its environment. Such ability is called endomorphism [5]. According to us, model-making is what make an agent intelligent, its goal.

Hence, interactions can be simulated through physical or mental sensors (for vision, communication and/or property sensations). Sensors are controlled by an agent to collect data from the environment. Sensors and actuators (for motion, communication and/or property actions) are controlled by the agent according to a state value to reach (continuous or discrete). Perception consists of information construction based on the interpretation of states from sensors and actuators (physical or mental) through the

mind. In mind, by memory, an agent is able to construct models of itself and of the environment. As depicted in [6], current and long term memories can be used to retrieve models and data. Using perception and action through memory, an agent can learn and plan model constructions.

Sensors and actuators constitute the body of an agent. Components of the mind are constituted of memory, planning, learning, perception, model of the agent itself and model of the model (metamodel) of the environment (physical and/or social). As an agent contains a model of itself (body and/or mind), it can contain a model of other agents, or a model of model (…) of other agents.

B. Multi-agent systems and environment

Usually, an agent is an active entity that can sense the environment (the cells) and carry out actions to affect the environment (changing the states of the corresponding cells). Examples of agents' actions are movement in the pedestrian crowd simulation and both movement and fire suppression actions in the firefighting simulation. To support interactions between an agent and its environment, the agent needs to be coupled to the cells in the CA model. Two types of couplings can exist. The first one is from the CA model to the agent model. This allows agents to sense properties of the cells (corresponding to the sensory function of the agent). The second one is from the agent model to the CA model. This allows agents to change cells properties (corresponding to the actuation function of the agent). How a cell changes its states based on the inputs from the agent can be specified using the specification described after (section IV).

Typically, an agent can only sense and affects its local environment. Thus an agent should only be coupled to a group of local cells based on the agent location. This brings out an interesting issue when the agent changes its location, i.e., moving from one place to another. When an agent moves, couplings between the agent and its local cells should be changed dynamically. Specifying such behavior depends on the modularity of the system. Using a multicomponent specification [1], cells influenced by agents can be embedded as a list of influenced non-modular components. Using a modular specification, each cell consists of a modular component (coupled or atomic) interacting with agents through ports. When the agent moves, to interact with cells, two specifications can be achieved. First, a third omniscient component (let us say a "space manager" [5]) has to keep track of agent positions and deal with agent-cell communications. Second, every single moving agents have to be connected to all cells or every cells they will move on.

A more intuitive and modularity preservation approach can be achieved through a dynamic structure specification. Specifically, when an agent moves from a current position to a new position, couplings between the agent and its current local cells should be dynamically removed and new couplings to the new local cells should be dynamically added. To support dynamic structure, a structure manager model $DSCA^2_\chi$ (*cf.* Figure 2) can be used. When an agent changes its location, this model is responsible to dynamically change the couplings between the agent and its environment (the cells). Note that this dynamical change of coupling is not part of the decision making of the agent, nor part of the dynamics of the environment. (Although one can argue that it is related to both the agent and the environment because, for example, the range of local cells is actually governed by the sensor/actuator capability of the agent and the physical laws of the environment). In this paper, we treat it as a separated component (and a general component in the specification) for the purpose of separation of concerns, *i.e.*, the structure manager is responsible for the dynamic structure aspect of the system.

The second goal of the structure manager is: to embed the definition of agents and cells, to receive general structure changes (changes of behavior, adding/deleting components, etc.) order and execute these orders.

Note that for the case of multiple agents, this paper pays less attention to the social network of agent communications. In our specification, communications from other agents are treated in the same way as getting sensory inputs from the environment. In this sense, an agent views other agents and the physical environment together as its environmental context.

Let us illustrate the major concepts of $DSCA^2$ discussed above through a simple forest firefighting example originally developed in [7] . In this example, the agent model is used to model firefighting resources and the CA model is used to model fire spread behavior. An agent can move in the cellular space with a certain speed. During the movement, an agent keeps track of its own position and constantly sends its position to a *structure manager*. Whenever the structure manager receives a message that contains the agent's (new) positions *(x, y)*, it will find the cell where the agent locates. If the cell ID has changed, the coupling manager will remove the couplings between the agent and the old cell and add the couplings between the agent and the new cell. After the agent moves to a new cell and receives a message about the cell's state, it will make a decision to carry out fire suppression actions (send a fire suppression message to the cell) based on certain wildfire suppression rules (some examples rules are given in [7]). After the fire is suppressed, the cell's state is changed to "suppressed". Meanwhile, the cell will send a message to the agent, who then makes a decision where to move to continue suppressing the fire. Thus the agent can be specified as having one sensor that allows the agent to know the state of the environment (the cell) and two actuators: one for the movement (moving from one cell to another after the current cell is suppressed), and one for the fire suppression (send a fire suppression message to the cell). The mind of the agent makes decisions about carrying out fire suppression (connected to the fire suppression actuator), or moving to a new cell (connected to the motion actuator). This decision making is based on the sensory inputs (messages from the cell about the cell's

state).

IV. DYNAMIC STRUCTURE CELLULAR AUTOMATA AND AGENTS

A *DSCA²* (Dynamic Structure Cellular Automata & Agents) is a structure:

$$DSCA^2 =< X_{DSCA}, Y_{DSCA}, DSCA^2_\chi >$$

With,

$X_{DSCA} = \{(p,v) / \ p \in IPorts \wedge v \in V\}$,

$Y_{DSCA} = \{(p,v) / \ p \in OPorts \wedge v \in V\}$, where *OPorts* and *IPorts* are respectively output and input port names and *V* are whatever values received as external influences for cells.

Dynamic structure changes are handed by:

$$DSCA^2_\chi =< X_{DSCA^2_\chi}, Y_{DSCA^2_\chi}, S_{DSCA^2_\chi}, \delta_{DSCA^2_\chi}, \lambda_{DSCA^2_\chi}, \tau_{DSCA^2_\chi} >$$

The *structural state* is defined as $S_{DSCA^2_\chi} =< D, \{C_i \cup A_i\}, \{I_i\}, \{Z_{i,j}\} >$. As defined in the DSDEVS formalism or in Kiltera language [8], the DSCAχ state is linked to the structure to represent explicitly structure configurations at one point in time. For all sub-systems $i \in D$ contains the DSCA² names of *active* cells, $\{Z_{i,j}\}$ is the set of coupling functions (all cells can be externally connected to both input X_{DSCA} and output Y_{DSCA} of the *DSCA*: $Z_{DSCA^2 \to DSCA^2_\chi} : X_{DSCA} \to X_{DSCA^2_\chi}$, $Z_{DSCA^2 \to c} : X_{DSCA^2} \to X_c$, $Z_{c \to DSCA^2} : Y_c \to Y_{DSCA^2}$ [1], $Z_{DSCA^2_\chi \to DSCA^2} : Y_{DSCA^2_\chi} \to Y_{DSCA^2}$, and $I_c = \{N_c, DSCA^2\}$, $I_a = \{\{cell_i\}, \{agent_i\}, DSCA^2\}$, $I_{DSCA^2_\chi} = \{\{cell_i\}, \{agent_i\}\}$ where N_c is the neighborhood of a cell *c*. It is a set of pairs representing the relative positions of the neighboring cells *p* and the cell *c*: $N_c = \{(i_p, j_p) / \ p \in I_c, i_p, j_p \in \mathbb{Z} \wedge i_p, j_p \in [-1,1]\}$.

$\delta_{DSCA^2_\chi} : X_{DSCA^2_\chi} \times S_{DSCA^2_\chi} \to S_{DSCA^2_\chi}$, is the structural state transition function. According to current structural state and inputs, the transition function can compute new structural states. Changes in structure include changes in cells neighborhoods, changes in cell definitions, and addition or deletion of cells. The structural state transition function is composed of internal and external functions $\delta_{DSCA^2_\chi} = \{\delta_{\text{int}_{DSCA^2_\chi}} \cup \delta_{ext_{DSCA^2_\chi}}\}$. External transitions allow accounting for external events and internal ones for autonomous computations of self states (for more information: [9].)

[1] For a modular specification, internal couplings of influenced neighboring cells are defined [case (2) of Figure 1] as: $Z_{c \to c'} : Y_c \to X_{c'}$.

$\lambda_{DSCA^2_\chi} : S_{DSCA^2_\chi} \to Y_{DSCA^2_\chi}$ is the structural state output function. Through the output function structural states can be sent to other models.

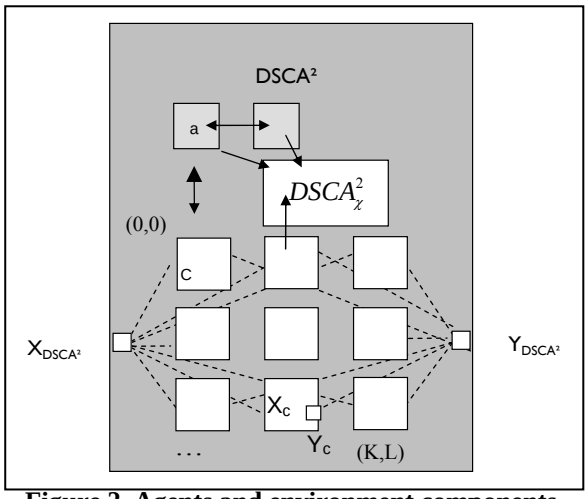

Figure 2. Agents and environment components

Each agent and each cell is a black box, which can be:

- A single atomic model, or
- A (dynamic structure) network.

C. Minimum specification of cells

As a minimum assumption, each cell *c* can be specified as an atomic component:

$$C =< X_c, Y_c, S_c, \delta_c, \lambda_c, \tau_c >$$

$S_c =< (m,n), S^{N_d}, phase >$, with

$$\begin{cases} S^{N_d} = \{s_p / \ p \in I_c\} \\ phase = \{"active", "passive", ...\} \end{cases}$$

When receiving or sending its state, a cell is in phase "*active*", otherwise it is in phase "*passive*".

$\delta_c : X_c \times S_c \to S_c$ is the transition function[2] composed of internal and external functions $\delta_c = \{\delta_{\text{int}_c} \cup \delta_{ext_c}\}$, where $\delta_{\text{int}_c} : S_c \to S_c$, and $\delta_{ext_c} : X_c \times S_c \to S_c$.

$\lambda_c : S_c \to Y_c$ is the output function.

$\tau_c : S_c \to \mathbb{R}^+_0$ for discrete-event systems and $\tau_c \in \mathbb{R}^+_0$ is a constant time advance for discrete-time systems.

For a more complex cell, the latter can be decomposed as a network (dynamic structure or not) of sub-components [10]. However, regarding the closure under coupling of

[2] Modularity cases: (1) $S_c = S_c^{N_c}$ and $X_c = X_{DSCA}$ (assuming external influences of cells), (2) $X_c = X_{DSCA} \times X_c^{N_c}$, with $X_c^{N_c} = \{x_p / \ p \in I_c\}$.

DSDEVS, precise network specifications can be expressed by (or is equivalent to) a single atomic specification (more details in [1]).

D. Minimum specification of Agents

As a minimum assumption, each agent *a* can be specified as an atomic component:

$$A =< X_A, Y_A, S_A, \delta_A, \lambda_A, \tau_A >$$

Where,

$X_A = \{("sensor", v) / v \in V\}$,where V are whatever values,

$Y_A = \{("actuator", v) / v \in V\}$, where V are whatever values,

$S_A =< \{(phase, v)\} >$, with $phase = \{"active", "passive", ...\}$ and $v \in V$, where V are whatever values. The main assumption here is that a finite set of phases guided the agent behavior. At each phase, states and values can be modified. More explanations will be given here after.

$\delta_A : X_A \times S_A \to S_A$ is the transition function composed of internal and external functions $\delta_A = \{\delta_{\text{int}_A} \cup \delta_{ext_A}\}$, where $\delta_{\text{int}_A} : S_A \to S_A$ corresponds to the autonomous behavior of an agent.

$\delta_{ext_A} : X_A \times S_A \to S_A$, corresponds to the reaction of agents to input sensors.

$\lambda_A : S_A \to Y_A$ is the output function corresponding to the actuator activation.

$\tau_A : S_A \to \mathbb{R}_0^+$ for discrete-event systems and $\tau_A \in \mathbb{R}_0^+$ is a constant time advance for discrete-time systems.

For a more complex agent, the latter can be decomposed as a network (dynamic structure or not) of sub-components. However, regarding the closure under coupling of DSDEVS, precise network specifications can be expressed by (or is equivalent to) a single atomic specification (more details in ZEI2000).

E. Specification levels inside agents

Structure of agents is very variable. Our scope here is to be specific enough to guide the modeler in his design phase and large enough to let him free. One generic structure everyone agrees is the mind / body decoupling. The body is responsible for interactions with the environment through sensors and actuators. Then, the mind is responsible for interpreting data received from sensors and actuators for making interpretation and then decision making. According to the complexity of an agent behavior, many levels of specification can be detailed. At every level, the mind acts as a controller over the body, commanding and interpreting sensors and activators. We investigate an agent structure through the decomposition of mind and body.

At a first structure level, an agent can be an atomic component. Autonomy and (mind) of agents correspond to internal transitions. External events can activate sensor sub-routines (according to their type, *e.g.*, "vision", "tactile", etc.) trough external transitions to represent sensors and interpretations of agents. Actuators are piloted by the mind through internal events. They act upon environment through output external events and output functions. The mind can be represented as a finite-state atomic model whose states correspond to agent phases. At a first behavioral level, the agent can be considered in general phases "active" or "inactive". Then, these phases can correspond to the activation or deactivation of general tasks of the agent : "get_prop(erty)", "set_prop(erty)", "move", etc. To represent it, the graphical language defined in [11], and used in [12] for agents, can be used. An example is depicted on the top of Figure 3. External transitions are represented in red, internals in black. Receiving inputs leads on sensors to an external transition, which actives the "get_prop" phase. Then, an internal transition decides to "move" or to set a property ("set_prop") of a cell. This is a very simple example. Other finite states can be added.

To progress in specification level, two kind of decompositions can be achieved, one on body, the other one on mind. The first is more physical, the second is more behavioral.

To be more generic, we choose here to specify the body. Hence, a second level of structure specification of agents can be considered. Three kinds of sub-components within an agent network: sensors, actuators and the mind. The bottom of Figure 3 describes a first network decomposition of agents. Sensors can be for: vision, communication and properties. Actuators can be for: properties, moving and communications. A minimum assertion is that the mind can pilot both sensors and actuators, which send events to cells and receive events. However, sensors could only receive and transmit inputs as passive components. Figure 3 describes such decomposition. A single example of sensor remains in a "waiting" phase as long as it does not receive an external event from a cell or the mind. In the same way, an actuator will be activated when it receives a command order from the mind to set a property of a cell. In mind, the "get_prop[erty]" phase can be activated either by an external event or a internal decision event. Setting properties depends on an internal decision.

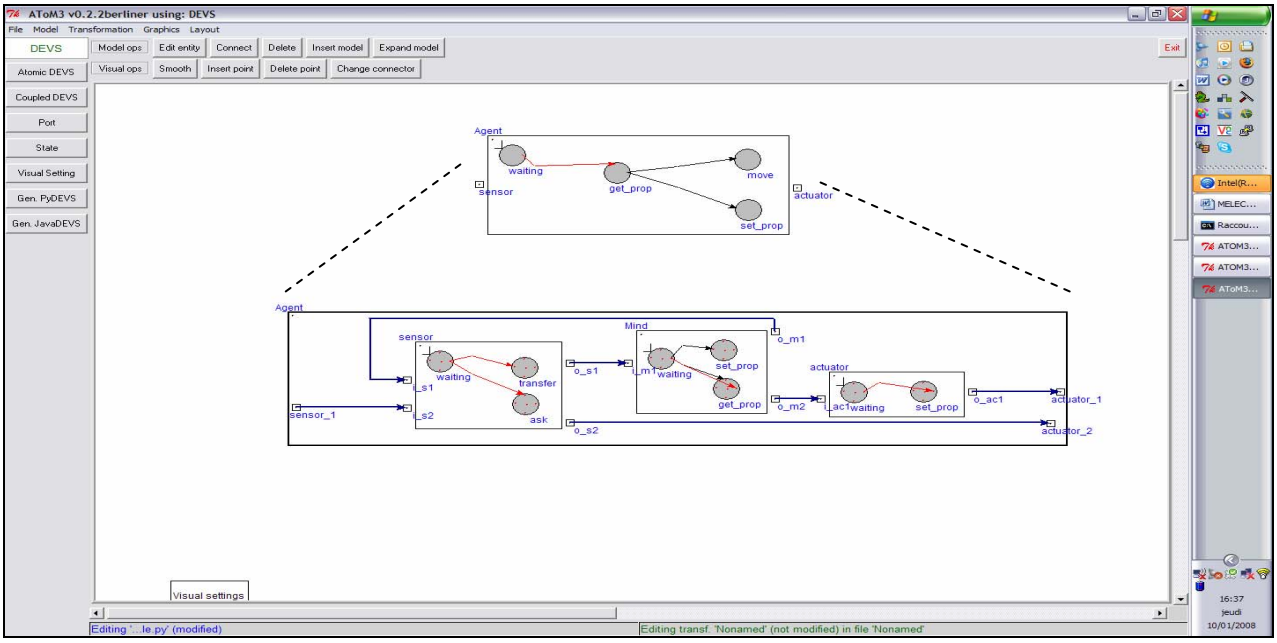

Figure 3. Precising agent specification

V. RELATED WORKS & PERSPECTIVES

Currently, agents constitute an emerging paradigm in many disciplines (social sciences, biology, computer engineering, etc.) Decision planning, artificial (or distributed) intelligence open new perspectives to these disciplines. Agent structures are designed to improve system understanding and reactivity. However, although low-level simulation languages (e.g., [13]) and philosophical discussions have already been proposed, agents lack formal structures and a specification framework. We depict here after the consistency of our approach with other ones.

In [14], a proposal of standard protocol for the description of agent-based models consists of: (i) Overview: Purpose, state variables and scales, process overview and scheduling (flowcharts, etc.), (ii) Design concepts: Emergence, adaptation, fitness, prediction, sensing, interaction, stochasticity, collectives (groups), observation (data collection), and (iii) Details: Initialization, input, sub-models (mathematical "skeleton"). Our approach can be used to specify each level.

In [15], Finite Deterministic (FD-) DEVS is introduced. A sound mathematical relation to DEVS is depicted. Linking this approach with a graphical language X, allows defining minimum structure DEVS models whose finite states can be refined at a second stage.

In [5], Bernie ZEIGLER introduces a new insight for detecting state transitions of continuous flows of the real world through discrete event specifications. Although discrete-time sensors continuously check for state transitions, advancing upon discrete time steps, a discrete event specification allows to reason on time. Time windows are defined to check for state thresholds. Activity tracking [16] and quantization [1] constitute an open research field. The latter constitute a new framework to model system dynamics. Still in [5], a model base management of agents components pinpoint the interest of our component based approach.

Combining DSCA² (at a low specification level) and PRIMA language [12] (at a higher one) opens promising application perspectives. Our next goal will be to describe a precise DSCA² specification in a fire spreading application [7]. Hence, dynamic structure changes in both cellular and agent models will be validated and described.

V. ACKNOWLEDGEMENTS

The first author would like to thank Juan de LARA and Ernesto POSSE for their collaboration on agent specification and graphical language use.

V. REFERENCES

[1] Zeigler, B.P., H. Praehofer, and T.G. Kim, *Theory of modelling and simulation*. 2000: Academic Press.

[2] Ntaimo, L.B. and B.P. Zeigler. *Expressing a forest cell model in Parallel DEVS and Timed Cell-DEVS formalisms*. in *Summer Computer Simulation Conference (SCSC'04)*. 2004. San Jose, USA.

[3] Wainer, G. and N. Giambiasi, *Application of the Cell-DEVS paradigm for cell spaces modeling and simulation.* Simulation, 2001. **76**(1): p. 22-39.

[4] Shiginah, F.A.S.B., *Multi-Layer Cellular DEVS Formalism for Faster Model Development and Simulator Efficiency*, in *Electrical and Computer Engineering Dept., University of Arizona*. 2006.

[5] Zeigler, B.P., *Object-oriented simulation with hierarchical, modular models*. 1990: Academic Press.

[6] Zeigler, B., A. Muzy, and L. Yilmaz, *Aritificial Intelligence in Modeling and Simulation*, in

Encyclopedia of Complexity and System Science. 2008, Springer-Verlag: Heidelberg, Germany. p. Accepted for publication.

[7] Hu, X., A. Muzy, and L. Ntaimo. *A Hybrid Agent-Cellular Space Modeling Approach for Fire Spread and Suppression Simulation*. in *Winter Simulation Conference (WSC), IEEE/ACM/SIGSIM/SCS*. 2005. Orlando, USA. p. 248-255.

[8] Posse, E. and H. Vangheluwe. *kiltera: a simulation language for timed, dynamic structure systems*. in *40th Annual Simulation Symposium (ANSS)*. 2007. p. 293-300.

[9] Barros, F.J., *Modelling Formalisms for Dynamic Structure Systems.* ACM Transactions on Modelling and Computer Simulation, 1997. **7**(4): p. 501-515.

[10] Muzy, A., A. Aïello, P.-A. Santoni, B.P. Zeigler, J.J. Nutaro, and R. Jammalamadaka. *Discrete event simulation of large-scale spatial continuous systems*. in *International Conference on Systems, Man and Cybernetics (SMC), IEEE*. 2005. Hawaii, USA. p. 2991-2998.

[11] Posse, E. and J.S. Bolduc. *Generation of DEVS modelling and simulation environments*. in *Summer Computer Simulation Conference. Society for Computer Simulation International (SCS)*. 2003. Montréal, Canada. p. 139-146.

[12] Muzy, A., J.d. Lara, and E. Guerra. *Designing PRIMA: A Precise Visual Language for Modeling with Agents, in a Physical environment*. in *MSV'07- The 2007 International Conference on Modeling, Simulation and Visualization Methods, Intel, MIT Media Laboratory…*. 2007. Monte Carlo Resort, Las Vegas, Nevada, USA p. 231-238.

[13] Ascape. [cited 2007; Available from: http://www.brook.edu/es/dynamics/models/ascape/].

[14] Grimm, V., U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S.K. Heinz, G. Huse, A. Huth, J.U. Jepsen, C. Jørgensen, W.M. Mooij, B. Müller, G. Pe'er, C. Piou, S.F. Railsback, A.M. Robbins, M.M. Robbins, E. Rossmanith, N. Rüger, E. Strand, S. Souissi, R.A. Stillman, R. Vabø, U. Visser, and D.L. DeAngelis, *A standard protocol for describing individual-based and agent-based models.* Ecological modelling, 2006. **198**(1-2): p. 115-126.

[15] Hwang, M.H. and B.P. Zeigler. *A Modular Verification Framework using Finite and Deterministic DEVS*. in *DEVS Symposium*. 2006. Huntsville, Alabama, USA. p. 57-65.

[16] Muzy, A. and B.P. ZEIGLER, *The activity tracking paradigm.* The Open Cybernetics and Systemics Journal, 2008: p. Submitted.