A CONTEXTUAL ENVIRONMENT APPROACH FOR MUTLI-AGENT-BASED SIMULATION

Fabien Badeig[1,2], Flavien Balbo[1,2], Suzanne Pinson[1]
[1]*Université Paris-Dauphine, LAMSADE Place du Maréchal de Lattre de Tassigny, Paris Cedex 16, France*
[2]*INRETS Institute, Gretia Laboratory, 2, Rue de la Butte Verte, 93166 Noisy Le Grand, France*
{*fabien.badeig—flavien.balbo—suzanne.pinson*}*@dauphine.fr*

Keywords: Multi-agent-based simulation framework, scheduling policy, environment, contextual activation

Abstract: Multi-agent-based simulation (MABS) is used to understand complex real life processes and to experiment several scenarios in order to reproduce, understand and evaluate these processes. A crucial point in the design of a multi-agent-based simulation is the choice of a scheduling policy. In classical multi-agent-based simulation frameworks, a pitfall is the fact that the action phase, based on local agent context analysis, is repeated in each agent at each time cycle during the simulation execution. This analysis inside the agents reduces agent flexibility and genericity and limits agent behavior reuse in various simulations. If the designer wants to modify the way the agent reacts to the context, he could not do it without altering the way the agent is implemented because the link between agent context and agent actions is an internal part of the agent. In contrast to classical approaches, our proposition, called EASS (Environment as Active Support for Simulation), is a new multi-agent-based simulation framework, where the context is analyzed by the environment and where agent activation is based on context evaluation. This activation process is what we call contextual activation. The main advantage of contextual activation is the improvement of complex agent simulation design in terms of flexibility, genericity and agent behavior reuse.

1 INTRODUCTION

Multi-Agent-based simulation modeling is decomposed in three phases (Fishwick, 1994): 1) problem modeling, 2) simulation model implementation, 3) output analysis. *Problem modeling* is related to the agent modeling, accurately specifying their behaviors according to their contexts (Macal and North, 2005); *Model implementation* is related to coding and execution of the simulation model by the MABS framework; *Output analysis* is related to the analysis of the simulation results. The scheduling policy is an essential part of the simulation design. Both the progression of time and how the agents are activated to perform an action are taken into account by scheduling policy. The way the agents are activated remains a relatively unexplored issue even if the activation process impacts the agent design.

In multi-agent-based simulation, the agents are situated in an environment. They are capable of autonomous actions in this environment in order to meet their objectives. The scheduling policy defines the activation order of the agents that behave according to their own state and the state of the environment that they perceive. These states constitute what we call the agent context. How the agents are activated and what information is available to compute the agent context depend on the multi-agent-based simulation framework.

In most MABS frameworks, context perception and evaluation is done by the agents, this type of framework is *agent-oriented* and presents several limits. The first limit is that an agent model is not independent on the MABS framework because the context computation is done in the agent. The modeling of the relations between the context and the agent actions is done during the *problem modeling* phase. As a consequence, the choice of a MABS framework implies the choice of an agent model. Since *problem modeling* and *model implementation* phases are not clearly separated, the genericity of the agent model is limited. The second limit is related to context com-

putation. This computation is repetitive because the context is computed in each agent at each time cycle during the simulation execution. This computation is done in each agent even if the agent context has not changed between two time cycles and/or if a group of agents shares the same context during a time cycle. The last limit is related to the flexibility and the reusability of the simulation model. The flexibility of the simulation model depends on the ease with which the simulation designer can change the relation between a context and an action. In *agent-oriented* frameworks, this relation is computed in the agents and any change in this relation implies the modification of the agent implementation.

To cope with these limits, we propose a new MABS framework, the Environment as Active Support for Simulation (EASS) framework. The EASS framework is based on a coordination principle that we have called Property-based Coordination (PbC) (Zargayouna et al., 2006). This improves the flexibility and reusability of the simulation model: 1) flexibility: in order to evaluate several hypotheses for the same simulation model, the agent behaviors should be changed without modifying the implementation of the agents; 2) reusability: the same action implementation and context modeling should be used in several simulation models. These objectives imply that context evaluation should not be computed in the agents, but should be externalized. Our new framework is *environment-oriented* because the environment manages the scheduling policy and the activation process. Our objective is that an agent is directly activated by the environment according to its context to perform a suitable action associated with this context. This activation process is what we call *contextual activation*.

The remainder of the paper is organized as follows. Section 2 focuses on the classical activation process where advantages and limits are highlighted. Then we explain why the environment as an active support of simulation is an interesting alternative. In Section 3, we detail our EASS framework. Section 4 presents our first results and evaluation. The paper concludes with general remarks and suggestions for future research.

2 STATE OF THE ART

As we said before, one of the main tasks in multi-agent-based simulation design is the choice of a scheduling policy and more precisely the choice of the scheduler role in the agent activation process. In classical MABS frameworks, the scheduler is a specific component which ordonnes the activation of the agents. When an agent is activated, it has to compute its context before acting. By context computation we mean the recovery and the accessible information analysis process. When all agents of the simulation are activated, the time of the simulation is updated (from t to $t + \delta t$).

The classical MABS frameworks are designed to support this activation process. For example, in the well-known platform CORMAS (Bousquet et al., 1998) , the scheduler activates a same method for each agent. To represent agent behavior, the designer has then to specialize it. In the Logo-based multi-agent platforms such as *TurtleKit* simulation tool of MADKIT (Ferber and Gutknecht, 2000) or the STARLOGO system (http://education.mit.edu/starlogo/), an agent has an automaton that determines the next action that should be executed.

The main problem of these frameworks is that the context computation is implemented in each agent. As presented in the introduction, there are three limits to these frameworks: 1) the difficulty to propose a simulation model that is independent on the implementation; 2) the repetitive characteristic of the context computation process during the simulation execution; 3) the difficulty to propose a simulation model that is flexible and reusable. Indeed, if the simulation designer wants to modify the simulation behavior, he has two ways to do so. The first way is to modify the scheduler, i.e. the activation order. The second way is to modify the behavior of the agents, i.e. their reaction to their context, which often implies a modification of the way the agent is implemented.

To cope with this last limit, we propose an *environment-oriented framework*, that we call EASS (Environment as Active Support for Simulation). Context interpretation and context-based reasoning are key factors in the development of intelligent autonomous systems. Despite significant body of work in MABS design, there is still a great deal to do in context modeling since generic context models need to be further explored, more specifically, the link between context and agent activation needs to be deeply studied.

Since the environment is a shared space for the agents, resources and services, the relations between them have to be managed. The first responsibility of the environment is the structuring of the MAS. Modeling the environment is useful to give a space-time referential to the system components (Mike and B., 1999). The second responsibility of the environment is to maintain its own dynamics (Helleboogh et al., 2007). Following its structuring responsibility, it has to manage the dynamics of the simulated environment, ensuring the coherence of the

simulation. For example in the simulation of ant colonies, the environment can ensure the propagation and evaporation of the pheromone (Parunak, 2008). Moreover, the environment can ensure services that are not at the agent level or can simplify the agent design. Implemented with the simulation platform SeSam (http://www.simsesam.de/), in a traffic light control system (Bazzan, 2005), the environment, with its global view, gives rewards or penalties to self-interested agents according to their local decision. Since the environment with its own dynamics can control the shared space, its third responsibility is to define rules for managing the multi-agent system. For example, in a bus network simulation (Meignan and et al., 2006), the main role of the environment is to constrain agent perceptions and interactions according to their state. Because the agents are "users" of the services of the environment, and in order to really create common knowledge, the last responsibility of the environment is to make its own structure observable and accessible. In the remaining, we detail our *environment-oriented* framework and motivate our design choices.

3 THE ENVIRONMENT AS ACTIVE SUPPORT FOR SIMULATION (EASS) FRAMEWORK

The objective of the EASS framework is to improve the flexibility and reusability of the simulation model. Another advantage of our framework is the use of a dynamic flexible process for agent activation, i.e. the possibility for an agent to modify its behavior in an easy way during the simulation excution. In section 3.1, we present an example that we use all along the paper to illustrate our framework. Section 3.2 introduces the Property-based Coordination (PbC) principle. In section 3.3, we describe the general simulation process with the new scheduling policy. Section 3.4 details the activation process.

3.1 An illustrative example

To illustrate our proposition, we use a robot agent-based simulation example. The interest of this simulation is the illustration of the following principles encompassed in our framework: active perception, decision making of situated agents and coordination through the environment. In this example, the robot agents and packets are situated on a *grid*, i.e. a two-dimensional space environment. The robot agents act in this environment and cooperate in order to shift packets. We have simplified the coordination process of the robot simulation because our objective is to focus on the flexibility and reusability advantages that underline our proposition. Each robot agent has a field of perception that limits its perception of the environment.

The first context is the *packet shifting* where two robot agents are ready to shift a packet. This context is related to the following information: 1) the skills of the robot agents; 2) the position of the robot agents and the packet. This context happens when two robot agents with complementary skills are close to a packet. The other contexts are the following: 1) the context *packet seeking* happens when a robot agent has no packet at proximity; 2) the context *packet proximity* when a robot agent is close to a packet and no robot agent with a complementary skill is close to the same packet; 3) the context *closest packet discovery* where a robot agent perceives the closest packet in its perception field; 4) the context *handled packet discovery* where a robot agent perceives a packet that is handled by another robot agent with the complementary skill. The two contexts related to the packet discovery (*closest packet discovery* and *engaged packet discovery*) enable to have two different behaviors of the robot agents : 1) the opportunist robot agents are those that choose the closest packet, and 2) the altruist robots are those that choose the packet knowing that another agent is waiting to shift it.

A robot agent is able to perform one of the following actions in a time cycle: 1) the action *move randomly* where the robot agent move randomly; 2) the action *move in a direction* where the robot agent moves towards a position; 3) the action *wait* where the robot agent waits at a position; 4) the action *shift packet* where the robot agent shifts the packet.

3.2 Property-based Coordination principle

In order to design our environment-oriented framework, we choose to use the Property-based Coordination (PbC) principle (Zargayouna et al., 2006). This PbC principle enables the environment to encompass all the responsibilities presented previously. In (Zargayouna et al., 2006), we have defined the PbC principle as follows: *The objective of the Property-based Coordination principle is to represent multi-agent components by observable Symbolic Components and to manage their processing for coordination purposes*.

In PbC, two categories of symbolic components are defined. The first category is the description of

real components of the multi-agent system: agents, messages, and objects. The descriptions constitute an observable state of the MAS components. The data structure chosen is a set of property-value pairs. Thus, an agent has its own process and knowledge, and possesses a description which is registered inside the environment. Since the only observable components are the descriptions inside the environment, a control can be applied by the environment on these descriptions. In the robot simulation example (see section 3.1), the description of a opportunist robot with the skill *carry* is written as follows: $<(id,5),(field_{perception},4),(id_{packet},unknown),(skill,carry),(position,(3,8)),(behavior,opportunist),(time,5)>$. This opportunist robot agent with the *id* 5 perceives the descriptions of the other MAS components that are situated more 4 cases around it (property $field_{perception}$). This robot agent is situated at the position $(3,8)$ and has no packet. This description is registered inside the environment.

The second symbolic component category is related to the abstract MAS components, especially to the coordination components. By coordination components, called *filters*, we mean a reification of the relation between an activation context and an agent action. An activation context is a set of constraints on the properties of the descriptions that defines the context that satisfies an agent need. In the modeling phase, the relations between the contexts and the actions have to be represented. We have defined only one kind of robot agent, the robot agents with the skill *carry* or *raise* and the behavior altruist or opportunist. The behavior of a robot agent is either opportunist or altruist as far as its choice of a packet in its perception field is concerned. The opportunist robot agents are those that choose the closest packet. The altruist robots are those that choose the packet knowing that another agent is waiting to shift it. A robot agent is modeled when all the relations between the contexts and the actions are represented giving as a result a set of filters. Figure 1 gives the relations that enable to model opportunistic or altruist robot agents. The difference relies on the context to activate the action *move in a direction*.

In the robot simulation example, if a robot agent perceives a packet in its perception field, its need is to move in the direction of the packet and its context is then the information about this packet. When a filter is triggered, the action to be performed by an agent is activated. The same filter can be used by several agents if they have the same need. In our example, the altruist and opportunist robot agent share the filters $f_{packet\ seeking}$, $f_{packet\ shifting}$ and $f_{packet\ proximity}$. For example, the filter $f_{packet\ proximity}$ reifies the link between the context *packet proximity* and the agent action *waiting*.

The reification of agent needs by the filters is the starting point of contextual processing of agent action. The advantage is that each agent choose its reaction to its current context including its own state.

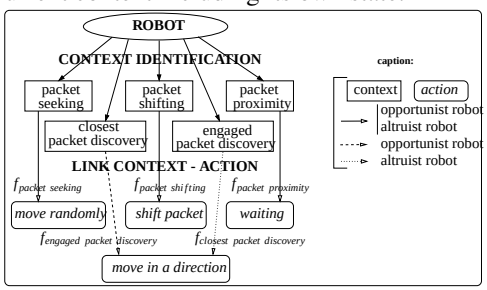

Figure 1: agent behaviors and activation context.

3.3 Our approach of the simulation process

In EASS framework, the environment manages the filters that are the result of the modeling phase, and the information related to the simulation components (agents, objects, ...). In addition, the environment integrates the management of the scheduler and of the simulation time. Thus, the environment is able to apply what we call contextual activation. An agent is directly activated by the environment depending on its context and performs the suitable action associated with this context (see figure 2).

Each agent chooses to put dynamically a filter inside the environment according to its behavior, and only filters inside the environment are taken into account in the simulation process. In figure 2, the robot agents R_1 and R_2 put in the environment several filters corresponding to the contexts they want to activate. In this figure, each filter is represented by a link between the scheduler (*SP*) and the agent. So the environment contains the filters related to the contexts *packet seeking*, *closest packet discovery*, *engaged packet discovery*, *packet proximity* and *packet shifting*. The altruist agent robots and the opportunist ones share the same filters for the evaluation of the contexts *packet seeking*, *packet proximity* and *packet shifting*. The behavior distinction between this two types of agents is done on the evaluation of the context about the packet discovery (see section 3.1). When an agent is activated, it performs the suitable action and has to update its internal time. Before performing an action associated to a context, an agent robot validates the execution of this action. The minimal condition to evaluate the context of a filter for an agent is the comparison between the internal time of the agent and the simulation time. The internal time of the agent has to

be inferior or equal to the simulation time.

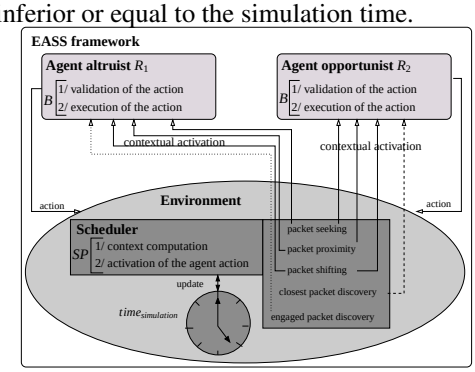

Figure 2: Contextual activation approach.

In the robot simulation example, at a time cycle, if the internal time of the agent R_1 is inferior or equal to the simulation time and if only the context *engaged packet discovery* is verified for R_1, R_1 is activated to perform the action associated with this link and it updates its internal time. When all the agents have an internal time superior to the simulation time, all the agents have been activated and the simulation time cycle is over. So the simulation time is updated from time t to $t + \delta t$.

The advantage of contextual activation is the simulation flexibility. This means that agent behavior can be easily modified. By modifying the relation between actions and contexts, agent behavior is automatically modified as well as simulation behavior. Agent behavior can be modified in three ways: 1) if the action associated with a link is modified (the agent wants to react differently to a context); 2) if the context of a filter is modified and not the action associated (the agent wants to modify the situation triggering an action) or if the agent changes the conditions on a context; 3) if the agent put or removes a link during the simulation execution (the agent wants to activate or to desactivate a reaction to a specific context).

3.4 Scheduling policy

The environment manages the scheduling policy that consists in organizing the set of the filters and the simulation time. Our purpose is to manage the scheduling policy and the simulation time thanks to the same formalism about the filters. Consequently, the environment contains two types of filters. The first type is related to the activation process of the agents and the second is related to the management of the simulation. In a multi-agent-based simulation using EASS, each filter evaluation has to be scheduled. In EASS, there are two scheduling levels: the global scheduling level which controls the execution of the simulation, and the local scheduling level which controls the contextual activation of the agents.

1/ At the global scheduling level, our proposition is to control the execution of the simulation by a state automaton that we call *execution automaton* where a state is a set of filters. By default, the execution automaton contains only one state related to the management of the simulation. Its contains at least two filters called $f_{default}$ that activates the agents by default and $f_{time\ update}$ that updates the time of the simulation. The filter $f_{default}$ compares the internal time of agents and the time of the simulation. The internal time of an agent represents the time when an agent wants to be activated. Consequently, the internal time of agents is observable and the agents have an observable property called $time$. The value of this property corresponds to the next time when the agent wants to be activated. In the robot simulation example, the filter $f_{default}$ is renamed $f_{packet\ seeking}$.

With this only state, our simulator behaves according to the classical scheduling policy presented in section 2 with only the filters $ftime\ update$ and $f_{default}$. The filter called $f_{time\ update}$ enables a discrete management of the simulation time. The filter $f_{default}$ activates the action $defaultAction$ that has to be specialized by the designer. Thus, at each time cycle, all agents are activated without context thanks to the filter $f_{default}$. When an agent is activated, he has to compute their context to determine the suitable action to perform.

In our example of robot simulation, the execution automaton is composed of two states. The first state contains the filters associated with the contexts : *packet shifting*, *packet proximity* and *packet discovery* (respectively $f_{packet\ shifting}$, $f_{packet\ proximity}$ and $f_{packet\ discovery}$). The second state contains the filters $f_{packetseeking}$ which is the default filter, and $f_{time\ update}$. The transitions between the execution automaton states are triggered by the value of a variable *state* that is registred inside the environment. The filter $f_{stateX \rightarrow stateY}$ enables to change the value of the variable $state$ from $stateX$ to $stateY$. The transition between states is done when there is no filters belonging to the current state waiting to be triggered. A cycle of simulation corresponds to the evaluation of all filters in each state of the execution automaton. The order of the execution states is important because it determines the group of filters that will be evaluated. Consequently, the order of the states can be changed to test other scenario and to modify the simulation behavior.

2/ The local scheduling level is related to the management of the filters which are in the same state of the execution automaton. At each time cycle, a simulation agent performs at most one action. In one state of the execution automaton, three problems may arise:

1) a conflict problem between potential filters which can be triggered for the same agent; 2) the fact that the same agent may be activated more than once in the same cycle (uniqueness of agent action); 3) what happens if a simulation agent is not related to any filters. To solve the conflict problem, our proposition is to give a priority level to each filter. The filter with the highest priority is evaluated before a filter with a lower priority in the same state of the execution automaton. In a simulation time cycle, the uniqueness of an agent action is ensured by a comparison between the internal time of the agent and the time of the simulation. If an agent has not filter that has been triggered at the current time cycle and if its internal time is inferior or equal to the time of the simulation, the filter $f_{default}$ is used to activate this agent.

This contextual activation approach presents several advantages: 1) two control levels, i.e. a global scheduling level and a local scheduling level, allow a better way to modify the simulation behavior; 2) the activation is contextual and avoids a repetitive computation of the context inside the agents; 3) the same agent is activated at most once a time cycle.

4 Experimentation and first results

In order to study the feasibility of our proposal, a prototype of our MABS framework has been implemented as a plugging of the multi-agent platform Madkit (Ferber and Gutknecht, 2000). MadKit is a modular and scalable multi-agent platform written in Java. MadKit software architecture is based on plugins and supports the addition or removal of plugins to be adapted to specific needs. Our plugin is composed of an environment component with an API that enables agents to add/retract/modify their descriptions and filters. As EASS is a very dynamic model, we have chosen to implement it within a Rules Based System (RBS) based on the RETE algorithm. The instantiation of the MAS components into a RBS is straightforward: the descriptions are the facts of the rule engine, and the filters are its rules. The RETE algorithm decides rule firing.

Thanks to a more complex robot simulation example (Badeig et al., 2007), we have undertaken several tests and validation as far as is concerned the flexibility of the EASS model. The flexibility of the proposition has been evaluated by the number of simulations that have been tested without changing the implementation of the agents. With eight filters, we have tested six simulations with two different interaction policies. More details related to the efficiency of the EASS framework are given in (Badeig et al., 2007).

5 CONCLUSION

In this paper, we have presented a new framework called EASS (Environment as Active Support for Simulation) for multi-agent-based simulations. In this framework, we have proposed a new activation process that we call *contextual activation*. Our proposition is based on the Property-based Coordination(PbC) principle which argues that the components of a multi-agent system have to be observable through a set of properties. We propose to represent explicitly the relation between the agent actions and their contexts. The main advantage is to improve the flexibility of the simulation design and the reusability of the simulation components.

REFERENCES

Badeig, F., Balbo, F., and Pinson, S. (2007). Contextual activation for agent-based simulation. In *ECMS07*, pages 128–133.

Bazzan, A. L. C. (2005). A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multiagent Systems*, 10(1):131–164.

Bousquet, F., Bakam, I., Proton, H., and Page, C. L. (1998). Cormas: Common-pool resources and multi-agent systems. In *IEA/AIE*, pages 826–837.

Ferber, J. and Gutknecht, O. (2000). Madkit: A generic multi-agent platform. In *4th International Conference on Autonomous Agents*, pages 78–79.

Fishwick, P. A. (1994). Computer simulation: Growth through extension. In *Society for Computer Simulation*, pages 3–20.

Helleboogh, A., Vizzari, G., Uhrmacher, A., and Michel, F. (2007). Modeling dynamic environments in multi-agent simulation. *Autonomous Agents and Multi-Agent Systems*, 14(1):87–116.

Macal, C. M. and North, M. J. (2005). Tutorial on agent-based modeling and simulation. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 2–15. Winter Simulation Conference.

Meignan, D. and et al., O. S. (2006). Adaptive traffic control with reinforcement learning. In *ATT06*, pages 50–56.

Mike, B. and B., J. (1999). Multi-agent simulation: New approaches to exploring space-time dynamics within gis. Technical Report 10, Centre for Advanced Spatial Analysis, University College London.

Parunak, D. (2008). Mas combat simulation. In *Defence Industry Applications of Autonomous Agents and Multi-Agent Systems*, Whitestein Series in Software Agent Technologies and Autonomic Computing, pages 131–150. Springer Verlag.

Zargayouna, M., Saunier, J., and Balbo, F. (2006). Property based coordination. In Euzenat, J. and Domingue, J., editors, *AIMSA*, volume 4183 of *Lecture Notes in Computer Science*, pages 3–12. Springer.