An Ontology-Based Metamodel for MultiAgent-Based Simulations

Florian BÉHÉ[ab], Stéphane GALLAND[a], Nicolas GAUD[a], Christophe NICOLLE[b], Abderrafiaa KOUKAM[a]

[a] *Multiagent Group, IRTES-SET, Université de Technologie de Belfort-Montbéliard, Belfort, France*
{florian.behe,stephane.galland,nicolas.gaud,abder.koukam}@utbm.fr
www.multiagent.fr

[b] *CheckSem Project, LE2I-CNRS, Université de Bourgogne, Dijon, France*
florian.behe@checksem.fr,cnicolle@u-bourgogne.fr
www.checksem.fr

Abstract

Multiagent-based simulations enable us to validate different use-case scenarios in a lot of application domains. The idea is to develop a realistic virtual environment to test particular domain-specific procedures. This paper presents our general framework for interactive multiagent-based simulations in virtual environments. The major contribution of this paper is the integration of the notion of ontology as a core element to the design process of a behavioral simulation. The proposed metamodel describes the concepts of a multiagent simulation using situated agents moving in a semantically enriched 3D environment. The agents perceive the geometric and semantic data in the surrounding environment. They are also able to act in this environment by using high-level actions, which are described by the ontology of the environment. The concepts relating to the environment, the agent, and the entire simulation models are presented. Additionally, guidelines are given to exploit the simulation results to characterize the agents. Finally, a simple application of the metamodel is presented, based upon the use of Industry Foundation Classes.

Keywords: multiagent, simulation, ontology, metamodel

1. Introduction

Multiagent-based simulations are now a common tool in various application domains to assess/validate different use-case scenarios: social sciences, urban management, transport, building security assessment, etc. The idea is to develop a realistic virtual environment to test particular domain-specific procedures. Developing realistic scenarios implies that they must be non-deterministic and agents inhabiting virtual environments are autonomous and intelligent.

The variety of their behavior must reflect the heterogeneity of human behavior.

This paper presents our general framework for interactive multiagent-based simulations in a virtual environment (2D or 3D). Its main objective is to integrate the notion of ontology as a core element of the design process of a behavioral simulation in order to facilitate its use/reuse by simulation designers and end-users in many application fields. To ease the use of a multi-agent simulation, it should be easily configurable: each agent behavior must be at high level and application-independent, and have their configurations as automated as possible. For empirical studies, multiagent-based simulation results are generally integrated and compared with results from other types of simulation (discrete events, finite elements, etc.) to merge the different views available on the studied system. It is therefore interesting to have a common representation of the simulation results that is both interoperable and cross domain.

We consider that the key point to address these issues is to have a semantic description of the environment in a multiagent-based simulation (MABS). To do this, this paper presents a new approach coupling multi-agent systems and semantic modelling with ontology.

Ontology is a general term for a semantic modelling of knowledge to define the know-how. Using ontologies, the system can infer new knowledge and relations from existing resources.

To enable the development of high-level easy to configure agent behavior, it is important to provide agents with the means to reason on their surronding environment. The agents must be able to analyze unexpected situations to dynamically adapt their behavior to achieve their personal goals [5]. Semantic rules and the agent's reasoning procedures can enable the development of such smart behavior.

For example, if you plan to send an agent to the restaurant, just specify that your agent is hungry. With semantic rules, the agent decides that he must eat. This action is semantically linked with "restaurant" (place where agents can eat in the ontology) and then he will go, in the environment, in a place defined as a restaurant. This solution saves a lot of designing and configuration time by just modeling the agent as hungry instead of designing a whole behavioural plan. The plan is dynamically determined at runtime according to a succession of semantic rules.

The designing of the agents behaviors is thus simplified thanks to the usage of semantic. But to enable this kind of reasoning behavior, the environment must provide agents with a semantic description of themselves.

Reasoning mechanisms and dynamic adaptation procedures of agents behavior are out of the scope of this paper. In this article, we describe the essential components that must provide the environment of a multiagent-based simulation to allow agents to easily support these mechanisms. This paper presents a new metamodel for MABS exploiting situated agents evolving in a semantically enriched 3D environment. This metamodel can be considered as a generic ontology [20].

The proposed approach is illustrated on a simple example derived from typ-

ical use cases in the field of building qualification (which is one of our main targeted application areas.) [3, 1, 4]. In this example, environment description is based on IFC (Industry Foundation Classes) files, which are a standard in the building industry.

They describe buildings in both a geometrical and semantic manner. This semantic description allows to easily build the semantic structure of the simulation environment.

The essence of our proposal is an ontology describing the semantic of every element needed or produced during a simulation (agents, environment, interactions, etc.) This ontology integrates the common MABS standards like the influences/reactions model [21, 16, 17, 8], the clear separation between an agent's mind and body, etc. This paper presents a formal modelling of this semantic metamodel that includes the main simulation principles, how to represent agents, manage their interactions, etc.

The paper is organized as follows: after a quick introduction to your ontology-based and agent-oriented metamodel (Section 2), the different components of the proposed metamodel are successively detailed (Section 3-6). It is then illustrated on a simple scenario in Section 7. Section 8 describes related works. Finally, Section 9 summarises the contributions of the paper and describes some future work directions.

2. A quick overview of the ontology-based metamodel

The goal of our metamodel is to integrate every key element required for developing multiagent-based simulations. It is now widely recognized that a MABS may be split into at least four main parts [14, 16, 17]:

- **Agent behavior**: modeling of the agent's deliberative process.
- **Environment**: definition of the various physical objects composing the simulated world as well as the endogenous dynamics of the environment. It is here that a number of fundamental principles must be respected to guarantee a simulation with the least possible bias: Influence/Reaction Model, respect the environmental integrity constraint, respect the constraint of locality, clear distinction between the agent's mind (variables under the sole control of the agent, i.e.: goals, motivations, etc.) belonging to the previous part of a MABS and the agent's body (state variables of the agent's physical component not controlled by the agent, i.e.: velocity, position, etc.) belonging to the environment.
- **Scheduling**: modeling the passage of time and defining the chosen scheduling model.
- **Interactions**: modeling the result of the agents' actions and interactions.

What is considered as agent or environment obviously depends on the problem considered. The agents represent the active components of the problem we seek to study.

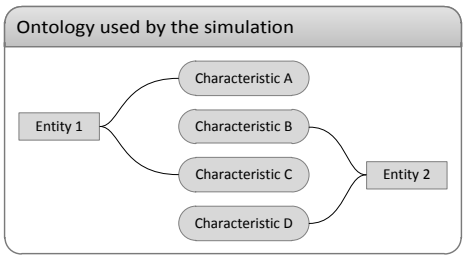


Figure 1: Example of an ontology derived from our metamodel

Our approach splits the description of a simulation into two main ontological aspects:

- **Running ontology**: describing all entities related to and produced by a simulation. This ontology describes in an anonymous manner every kind of data that is used and produced during a simulation. Instances in this ontology are not related to a specific concept like traditional ontologies but only to a high generic concept (such as "owl:Thing"). By solely having this aspect of the description, it is impossible to differentiate the nature of an object from another. An example of this ontology is available in Figure 1. This ontology is designed to be used closely to the definition bases.

- **Definitions bases**: defining all the entities that can be encountered in the running ontology.

  - An environment domain definition (see Figure 2) describing the various object type/classes that may make up the environment of a simulation.
  - The agent domain definitions (see Figure 2) describing the various agent's models of a simulation.

  These ontologies (one for the agents, another for the environment) contains only classes and no instances. They describe the characteristics that are required for a instance to be classified as a concept. Using these ontologies, it is possible to determine to which class the instances described in the running ontology belong. This allows to determine the nature of each instance and retrieve the semantic information that is related to them.

In the literature, multiple definitions of multiagent-based systems coexist. There is no standard definition of the agent concept that is shared among the whole multiagent community. On the other hand, each defines their own agent with certain common characteristics such as autonomy, proactiveness, reactivity, etc. In this paper, we consider the definition of Russell and Norvig [18]: "*an agent is an entity that senses its environment and acts upon it.*" This definition is large for the concept of agent, and the agents have a relation with

their environment. Our approach is not based on a more precise definition, but provides the user with the means to describe its definition in terms of the characteristics he wants for his agent's model. However, to fix the idea of the reader, our approach is still compatible with the definition of Ferber [10], which refines that of Russell and Norvig [18]: "*the agent is a physical or virtual entity: (i) which is capable of acting in an environment; (ii) that can communicate directly with other agents; (iii) which is driven by a set of tendencies; (iv) which has its own resources; (v) which is capable of perceiving its environment with a limited extend; (vi) which has only a partial representation of this environment; (vii) which has competencies, and provides services; (viii) which may breed himself; and (ix) whose behavior tends to meet its objectives, taking into account the available resources and its competencies, and depending on its perception, its representations and communications it receives.*"

The same principle is used to describe the characteristics of the objects making up the environment. The environment may be described differently according to the targeted application. This allows to have the same flexibility for the environment that we have for the agents. This approach is generic and easily extendable.

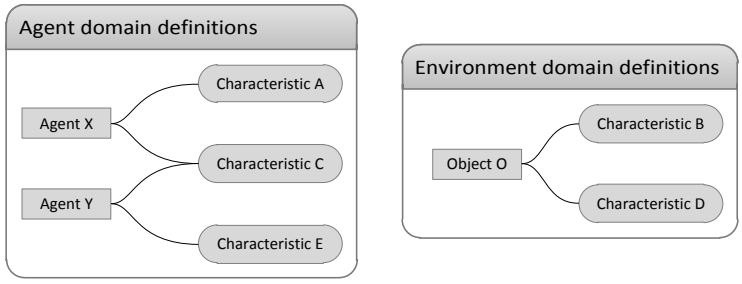


Figure 2: Example of definition bases for the metamodel

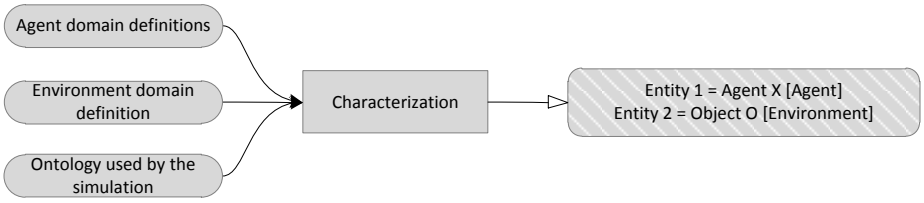


Figure 3: Classification principle illustrated according to Figures 1 and 2

Running ontology and definition bases are used in the characterization process during the simulation by the agents and after the simulation for result usage. The characterization process dynamically classifies simulation entities according to the definitions provided in Agents and Environment Domain Definitions. This classification depends on the characteristics associated to define entities in the simulation's running ontology. Some characteristics are essen-

tial for the classification according to some pre-established criteria related to a specific application domain (agents, environment, etc.).

The classification of the various entities is dynamic. It can change at each simulation step according to the definition bases and the state of the entities. Using this metamodel does not limit the knowledge used in a simulation to a specific kind of simulation. Our metamodel is extensible to any type of data without having to redefine a whole ontological model for a little change in the use domain. According to this feature, a wide range of multiagent-based simulations can benefit from the proposed approach[1]. It is easily adaptable to user needs and expectations.

To enable the implementation of this semantic classification, our metamodel proposes a set of abstractions and its associated formalization for the various aspects of a MABS: Agents (behaviors and interactions), Environment, and simulation configuration (integrating the scheduling aspect). Each of these parts will be detailed in the following sections. Section 3 describes the Environment part of our metamodel. The agent's part is then detailed in Section 4. Section 5 summarizes the overall conducting of a simulation step. Finally, Section 6 describes how to use the results of a simulation.

3. Environment

The term "Environment" defines both the physical environment and the physical description of agents (agent's body). Four main missions are classically assigned to the environment in a MABS [14, 21]:

Sharing information The environment is first of all a shared structure for agents, where each of them perceives and acts. This shared structure can also be the support of indirect communication (i.e. stigmergy).

Managing agents' actions and interactions For this aspect, the action model in situated MAS from Michel [17] and Ferber and Müller [8] is adopted. It is based on the distinction between influences and reactions to these influences. Influences are produced by agents and represent their desires to modify the environment. Agents do not directly modify the state of the environment. Reactions, which result in state changes, are produced by the environment by combining influences of all agents, the current environment state and the set of laws governing it. This distinction between the results of agent behaviour and the reactions of the environment provides a good means to handle simultaneous actions and *easily* solve conflicts. The influence/reaction concepts also provide a good means to verify if environmental laws are not broken by agents before impacting environmental states.

[1] situated agents, sensor networks, etc.

Managing perception and observation As opposed to agents, the environment must be locally and partially observable. The environment has to provide means (i.e. services) to the agents to inspect their environment. It manages the access to environmental information and guarantees the partialness and localness of agents' perceptions.

Maintaining internal dynamics The environment is an active entity. So it can have its own processes, independently of the ones of the agents. A typical example is the evaporation of artificial pheromones.

To allow the formal description of the various components of the environmental part of the metamodel, a number of variables must be previously described.

A set of states defines the state of the environment. These states are the states of each object or agent present in the environment. In a given set, each object has only one single state. Σ represents the set of all states possible. The environment may have different states in each simulation step. Σ_t represents all of the possible states of the environment at a given step t, where $\Sigma_t \subseteq \Sigma$.

The environment follows a specific progression defined both by the value of Σ_{t-1} and by the rules defining the behavior of the agents.

Only Σ_0, the initial state of the environment, is set by the simulation designer.

The set $\Sigma^r, \Sigma^r \subseteq \Sigma$ represents the set of real states of the environment. A real state of the environment is a state encountered during the simulation. At the end of the simulation, this set contains as many elements as the simulation has steps, i.e. for n simulation steps we have :

$$|\Sigma^r| = n \tag{1}$$

The element $\sigma_t^r, \sigma_t^r \in \Sigma^r$ represents the real state of the environment at the simulation step t. By the provided definitions and (1), we can state that

$$\Sigma^r \cap \Sigma_t = \{\sigma_t^r\}, \forall t \geq 0 \tag{2}$$

Concurrently to σ_t^r, there is σ_t^d that represents a desired state of the environment for a simulation step $t > 0$. The desired states of the environment represent the results of the correct execution of the rules. This result is the environment at the next step of the simulation.

In this way, σ_t^d is not necessarily an element of Σ^r and not even necessarily an element of Σ_t or even Σ.

Finally, the set $\Sigma^\alpha, \Sigma^\alpha \subseteq \Sigma$ represents the achievable states of the environment during the whole simulation. Like Σ^r, this set is only complete once the simulation is done and is dependent on the evolution rules. This set therefore respects the following properties :

$$\Sigma^r \subseteq \Sigma^\alpha \tag{3}$$

$$\Sigma_t \subseteq \Sigma^\alpha, \forall t \geq 0 \tag{4}$$

To simplify the expression of behavior and other evolution rules, the set $E_t, E_t \subseteq \sigma_t$ represents a subset of the state of the environment at simulation step t. It limits the number of objects to process for a specific action in the simulation.

The subset E_t follows the same notation rules as mentioned before, namely

- $E_t^r \subseteq \sigma_t^r$, a subset of the real state of the environment at the simulation step t
- $E_t^d \subseteq \sigma_t^d$, a subset of the desired state of the environment at the simulation step t

Figure 4 describes the overall functional architecture of the proposed approach.

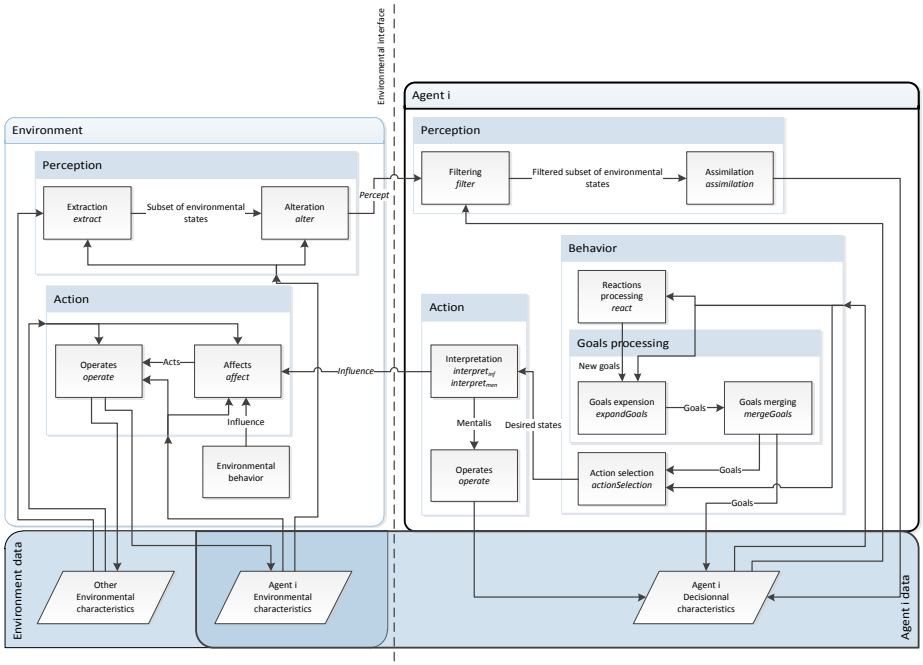


Figure 4: Flowchart of one simulation cycle for 1 agent

3.1. Environmental dynamic

The environmental dynamic is the set of rules that regulates the environment. For example, in a simulation of situated agents in a 3D world, the environmental dynamic will be a simplification of the law of physics. With the proposed metamodel, the environmental dynamic is expressed by Meterios that the environment manager will use to ensure a smooth simulation. The Meterios that represent environmental dynamics work by following the same principle as

reactions of the agents. The whole state of the environment is compared to the Meterios to perform a matching, and generate influences if a match is successful. The generated influences are processed in the same manner as those generated by the agents. Nevertheless, the influences generated for the environmental dynamic are prior to those emitted by the agents.

3.2. Meterios

Meterios are an abstract representation of various interactions that are possible for an agent (agent-agent interactions and agent-environment interactions). The word Meterio is the contraction of three terms : the prefix "meta-" which denotes something of a higher or second-order kind[2], the prefix "inter-" which means between and finally the Latin word "actio" which means action. Meterios are elements that serve the representation of input and output of actions. They group every kind of interaction that an agent can have with his environment (physical environment and agent).

3.2.1. Principle

Meterios express both real states and desired states of the environment.

A Meterio uses an input state and produces an output state. The input state corresponds to a real state of the environment. The output state can be a real state or a desired state of the environment.

They can consequently be considered as the rules allowing/enabling the progression of the simulation.

Composition. Meterios can be compound. This composition consists in classifying a Meterio. This classification depends on the input subset of the state of the environment. This input subset can be classified as valid or invalid. A valid subset occurs when the set is a subset of the current state of the environment, otherwise it is invalid. If the input of a Meterio is valid, then it is a unitary Meterio.

Otherwise, it may be considered as a compound Meterio, if the agent has one or more other Meterios where the composition of their output corresponds (in part or totally) to the input of the classified Meterio.

For example, in the case of a situated multiagent-based simulation (in which each agent has a position), a unitary Meterio may correspond to the displacement of an agent. Indeed, it only requires the agent to have a position. On the other hand, opening a door at a certain distance of the agent will be considered as a compound Meterio including displacement and opening Meterios.

The classification between unitary and compound meterio is dynamic. It depends on the environment state. The use of Meterios composition comes to the classification of agents to match definitions (see Section 6).

[2]Oxford Dictionaries

Formalization. Meterios represent the rules that allow the simulation to progress, they can be represented as functions that make a transition from one state to the other. A Meterio can be expressed as

$$\begin{array}{rccc} Meterio: & \Sigma & \longrightarrow & \Sigma \\ & E_t^r & \longmapsto & E_{t+\Delta t}^d, \Delta t \geq 1 \end{array} \tag{5}$$

In a well formed system, i.e. where Meterios and behavior are established in a coherent manner, whether in terms of time or modified states, we may have $E_t^r = E_t^d$. However, since the environment is dynamic, there is absolutely nothing that can guarantee this equality. Moreover, Meterios are not necessarily designed in a coherent manner, we cannot be sure that the given time (Δt) is sufficient or even guarantee that $E_t^d \subseteq \sigma_d^t, \sigma_d^t \in \Sigma^\alpha$. The output of a Meterio is described as a desired state of the environment to pass through other processing steps with the goal to make the desired state as close as possible to the real state.

3.2.2. Operation process

A desired state consists in a state of the environment that should be reached, it is consequently required to determine what the changes that must be performed are, to reach the desired state from the current state. These desired states are interpreted to generate some mentalis and influences, as illustrated on Figure 5.

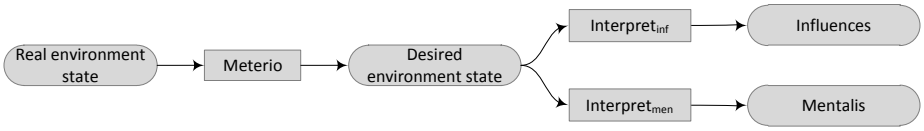


Figure 5: Interpretation of a meterio to generate mentalis and influences

Two functions perform this interpretation. Function $Interpret_{inf}$ process the desired state to identify if there are some environmental characteristics that are modified in it (6). If so, the function generates the influences that may allow to reach the described state.

$$\begin{array}{rccc} Interpret_{inf}: & \Sigma & \longrightarrow & I \\ & E_t^d & \longmapsto & \iota \end{array} \tag{6}$$

Function $Interpret_{men}$, process the desired state to identify if there are some decisional characteristics that are modified in it (7). If so, the function generates the mentalis that will allow the state of the environment to reach the desired state (for the states that concern decisional characteristics).

$$\begin{array}{rccc} Interpret_{men}: & \Sigma & \longrightarrow & M \\ & E_t^d & \longmapsto & \mu \end{array} \tag{7}$$

3.3. Actions

An action is a function that modifies the state of the environment in the frame of one simulation step. An action can be expressed in the form of the equation (8).

$$\begin{array}{rccc} Action: & \Sigma & \longrightarrow & \Sigma \\ & E_t^r & \longmapsto & E_{t+1}^r \end{array} \tag{8}$$

Actions enable us to reach a desired state of the environment but do not express it. An action is a complete process that interprets a desired state of the environment and executes successfully resulting influences. In our metamodel, an action is the response of the system to an act. An act is processed jointly with a real state of the environment to make a new state of the environment. Acts form the set A that any element of it is noted $\alpha, \alpha \in A$.

The metamodel proposes a particular category of acts where the impacted states are only made up of decisional characteristics. For this kind of act, the term mentalis is used. This term is the Latin word that means "related to the mind". Their characteristics are exactly the same as acts, but their action field is limited to the decisional characteristics. Mentalis can be viewed as a subclass of acts. They form the set $M, M \subseteq A$ that any element of it is noted $\mu, \mu \in M$.

Acts and mentalis are separated to enable to implement restrictions on the emitter of the acts and the mentalis.

A mentalis can only be emitted by the agent that is concerned by it, while an act can only be emitted by the system itself in response to an influence.

Finally, acts (and therefore mentalis) may have a dependency relation between them. It is possible to condition the execution of an act to the execution of another, within the same simulation step.

3.4. Influences

Influences modify the value of an environmental characteristic, and more generally the state of any physical characteristic in the environment. This modification is made in several steps. The first one consists in emitting a desired state of the environment, which one may include the modification of environmental characteristics. Secondly, this desired state of the environment is processed by the function $Interpret_{inf}$ to produce an influence. The environment considers the desired modifications issued from the influence.

The set I represents the influences. A particular influence is noted ι, $\iota \in I$. An influence is compound of two main elements: a subset of a real state of the environment E_t^r and a subset of a desired state of the environment $E_{t+\Delta t}^d$, $\Delta t \geq 1$. The first element represents the requirements that have to be met for the influence to be considered as valid. The second element is the desired state of the environment that the system will try to reach if the influence is valid. An influence can be seen as the couple illustrated on (9).

$$\iota =< E_t^r, E_{t+\Delta t}^d >, \Delta t \geq 1 \tag{9}$$

An influence is considered as valid only if the first of its elements respects $E_t^r \subseteq \sigma_t^r$. By following the creation process of an influence (combination of (5)

and (6)), the first element is the input of the Meterio and the second element is composed of the environmental characteristics of the Meterio output.

3.5. Perception

Perception mechanisms are used by the agents to get information from the environment in which they progress. The perception is the name given to the global process that consists in extracting information from the environment (environmental characteristics of elements) and integrating this information to the decisional characteristics of the agent which perceives. This process considers various parameters, some related to the agent, the others related to the type of the agent or of the simulation.

The perception process is illustrated in Figure 6.

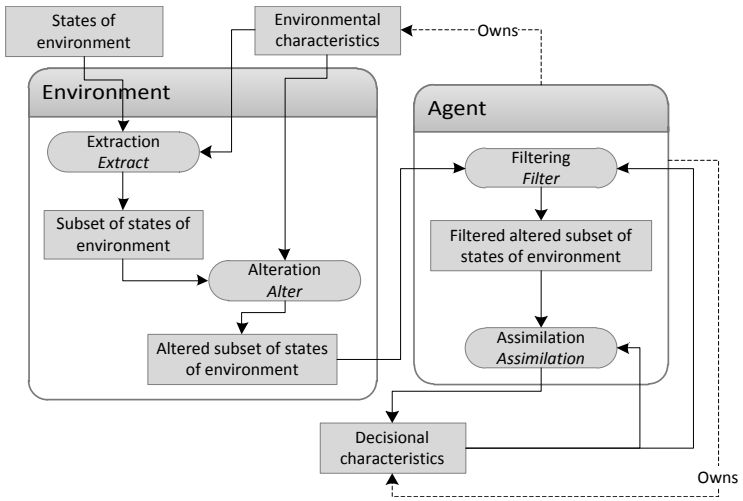


Figure 6: Perception process

The first step is a raw extraction of the environment. This extraction considers both the whole environment and the environmental characteristics of the agent making the perception. It produces a subset of state of the environment. It is the perception field of the agent according to its environmental characteristics. The next steps of the perception process use this subset.

The second step is an alteration set. It simulates the defects that can alter the perception of an agent. It can reproduce a visual deficiency, deafness, etc.

The resulting subset is not necessarily a subset of σ_t^r. This step can modify, add or delete information from the provided subset.

The third step is a filtering step. It considers the decisional characteristics of the agent and the altered subset provided by the second step. As the previous step, the filtering can add, delete or modify information. Usually, this step deletes the information that is not related to the perceiving the agent from the perception. But, it can also be used to simulate madness or other mental

problems of the agent. It can simulate the principle of focusing, making some elements unnoticed by an agent.

The fourth step is the assimilation of the perception. This simple process considers the subset provided by previous steps and the current decisional characteristics and generates new decisional characteristics that include the perception.

The perception is a process that builds a very personal view of the environment for each agent in the simulation.

Formalization. The perception is a compound process that extracts information for the real state of the environment to generate new decisional characteristics for the agent that makes the perception. The perception can be seen as the function exposed on (10) for the agent $ag_i, ag_i \in Ag$.

$$\begin{array}{rccc} Perception_i : & \Sigma & \longrightarrow & \Gamma \\ & \sigma_t^r & \longmapsto & \Gamma_d^i \end{array} \tag{10}$$

Equation (11) illustrates the four steps of perception and respects the main function (10).

$$\begin{aligned} Perception_i(\sigma_t^r) &= Assimilation_i \circ Filter_i \circ Alter_i \circ Extract_i(\sigma_t^r) \\ &= Assimilation_i(Filter_i(Alter_i(Extract_i(\sigma_t^r)))) \\ &= \Gamma_d^i \end{aligned} \tag{11}$$

The first step is the raw extraction from the environment. For an agent ag_i in can be expressed as the function $Extract_i$ detailed on (12).

$$\begin{array}{rccc} Extract_i : & \Sigma \times \Gamma & \longrightarrow & \sigma \\ & \sigma_t^r, \Gamma_e^i & \longmapsto & E_t^r, E_t^r \subseteq \sigma_t^r \end{array} \tag{12}$$

The second step is the alteration of the produced subset. This alteration, for ag_i can be expressed as the function $Alter_i$ detailed on (13) where Ξ_t^i is the altered subset. As a reminder, the produced subset may not be a subset of σ_t^r.

$$\begin{array}{rccc} Alter_i : & \Sigma \times \Gamma & \longrightarrow & \Xi \\ & E_t^r, \Gamma_e^i & \longmapsto & \Xi_t^i \end{array} \tag{13}$$

The third step consists in filtering the provided subset. This filtering can be expressed, for ag_i, as the function $Filter_i$ detailed on (14).

$$\begin{array}{rccc} Filter_i : & \Xi \times \Gamma & \longrightarrow & \Xi \\ & \Xi_t^i, \Gamma_d^i & \longmapsto & \Xi_t^{i'} \end{array} \tag{14}$$

Finally, the assimilation, for ag_i, can be expressed as the function $Assimilation_i$ detailed on (15).

$$\begin{array}{rccc} Assimilation_i : & \Xi \times \Gamma & \longrightarrow & \Gamma \\ & \Xi_t^{i'}, \Gamma_d^i & \longmapsto & \Gamma_d^{i'} \end{array} \tag{15}$$

3.6. Communication

Communication is an extremely important element in multiagent-based simulation: the intelligence of multiagent-based systems lies more in their ability to communicate than in the complexity of their individual behaviors [21]. Agent communication can take various forms: message sending, gestural, etc. But whatever the form chosen, they all have one thing in common: they all pass, at one time or another, through the physical environment. To represent communication, our metamodel proposes the meterios and the influences to modify the environment and the perceptions.

Thereby, each agent has the tools to modify and extract information from the environment.

4. Agents

There are many agent definitions in the literature. These approaches differ on the characterization of agents for the scope of the simulation.

We identify common concepts from these heterogeneous approaches.

An agent is an autonomous entity that progresses in an environment and able to interact with it. The various definitions mainly differ on three fundamental features: autonomy, behavior and interaction. The definitions of the environment are, for their part, a dedicated research topic and are usually not described in an article that defines the notion of agent.

This section depicts elements composing an agent in our metamodel. It gives a generic formalization of the autonomy, behavior and interaction part of an agent. We argue that generic formalization can represent each heterogeneous approach in the literature.

Formalization. Agents of a simulation are stored in the set Ag that contains every agent that is or has been in the simulation. A specific agent i is expressed by $ag^i, ag^i \in Ag$.

An agent relies on the operating process shown on Figure 7.

4.1. Agent Characteristics

Agent characteristics are split into two categories: decisional characteristics and environmental characteristics. The separation of the characteristics into two categories is inspired by the separation between the agent's body and mind proposed in [8, 17]. The characteristics of an agent, or more generally an entity, form the set Γ. In our metamodel, other elements that usually take part of an agent description[3] are bound to an agent through characteristics.

[3]Interaction possibilities, behaviors, etc.

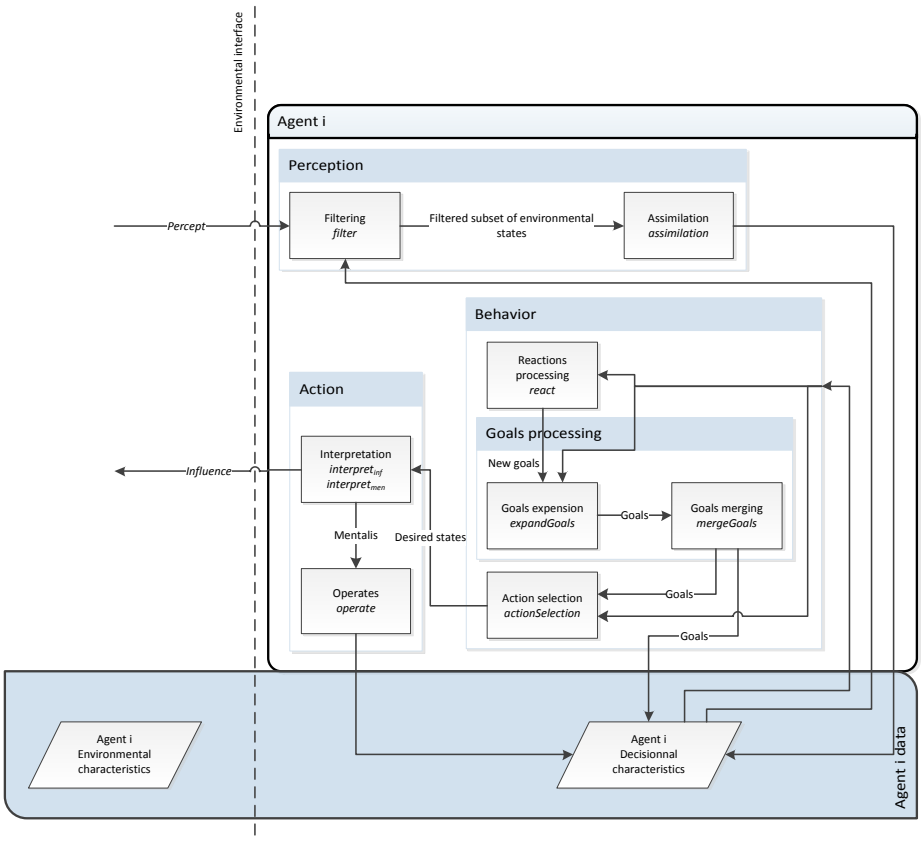


Figure 7: Process of an agent

4.1.1. Decisional characteristics

The decisional characteristics describe the internal state of the agent.

They bound interaction capabilities, behavior or other mental characteristics of the agent.

Decisional characteristics describe everything related to the mental state of an agent.

Some of these characteristics are constant. Otherwise, agents have total control over these characteristics. This possibility guarantees that the agents will not be able (in some cases) to reconfigure themselves to delete troublesome characteristics. This locking is not mandatory and stands as a possibility during the simulation design process. it limits potential splits between behavior and characteristic design phases.

Decisional characteristics have two levels of visibility: private and public. Private characteristics are only visible by the agent to which they are bound. Public characteristics may be visible to other agents but require agent behavior to be visible. To be perceivable by other agents, these characteristics must

be associated to one or more environmental characteristics (see Section 4.1.2). The agent must also decide to let this characteristic be visible. The distinction between public and private characteristics is dynamically made according to the current state of the agent.

Formalization. Decisional characteristics build the set $\Gamma_d, \Gamma_d \subseteq \Gamma$. Visibility attributes of decisional characteristics form two distinct subsets of Γ_d. $\Gamma_{d_{pu}}$ represents the public decisional characteristics. $\Gamma_{d_{pr}}$ contains the private decisional characteristics.

4.1.2. Environmental characteristics

Environmental characteristics are the characteristics related to the physical characteristics of an agent. They represent the physical characteristics (characteristics of the body) of an agent and their limits. In opposition to the decisional characteristics, agents do not have total control over these. A request addressed to the environment manager can modify these characteristics.

Figure 8 illustrates the process to change the environmental characteristics. This request can be sent by the agent or the environmental manager.

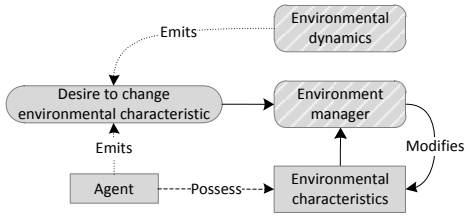


Figure 8: Illustration of the process to change an environmental characteristic

Environmental characteristics have also two levels of visibility: private and public. Public characteristics are returned by a simple perception process of other agents. Contrary to decisional characteristics, the agent has nothing to do to allow other agents to see them, moreover it is impossible for him to prevent this information to be accessed. Private environmental characteristics are, for their part, only visible by the agent himself in the simulation. They describe the properties of the agent body. Consequently, the simulation system uses these characteristics to check the validity and possibility of the request.

As an example of public environmental characteristics, the position of an agent may be mentioned in the case of situated agents. Indeed, the position of an agent is obtainable by a simple perception of this agent, and he can do nothing to restrict the access to this information. Moreover, the agent must effectively send a request to the environment manager in order to modify it. The maximum speed of this agent, for its part, is a private environmental characteristic and is set, for example, at 7 mph. If the request sent by the agent results in a move made at 10 mph, the effective change in the environmental characteristics of the agent will be made within a speed value of 7 mph.

Formalization. Environmental characteristics build the set $\Gamma_e, \Gamma_e \subseteq \Gamma$. Visibility attributes of environmental characteristics form two distinct subsets of Γ_e. $\Gamma_{e_{pu}}$ for the public environmental characteristics. $\Gamma_{e_{pr}}$ for the private environmental characteristics.

4.1.3. General formalization of an agent's characteristics

$$\Gamma = \Gamma_d \cup \Gamma_e = \Gamma_{d_{pr}} \cup \Gamma_{d_{pu}} \cup \Gamma_{e_{pr}} \cup \Gamma_{e_{pu}} \tag{16}$$

The characteristics of a given agent $ag^i, ag^i \in Ag$ are contained in the set $\Gamma^i, \Gamma^i \subseteq \Gamma$. The separation between environmental and decisional characteristics is the same as described above, i.e. Γ_e^i represents the environmental characteristics. Γ_d^i represents the decisional characteristics. Each of these sets is split between private and public characteristics.

4.2. Agents Behavior

Behavior provides rules allowing agents to progress in their environment. They define how an agent will react in a given situation or the reason of the presence of the agent in the simulation. The behavior of an agent is made up of elements classified into two categories: reactions and goals.

4.2.1. Reactions

Reactions[4] represent the manner that an agent will react in a given situation. The perception describes the situation to the agent. Reactions are considered like Meterios because they also associate a desired state of the environment to a perceived state of the environment. For the reactions, the environment describes the view that agents have built from the real environment through a perception and their decisional characteristics. The choice of associating a desired state of the environment and not directly a specific Meterio has been made to keep a certain flexibility upon agent modeling and let agents adapt their reaction according to their environment.

The Meterios that serve for the behavior modeling are not directly interacting with the environment, but they add new goals for the agent. After each perception, the agent will analyze the representation of the environment that it just got and try to make a matching with each one of its reactions. If the input of reaction matches, then the corresponding goal is added to the list. If there is more than one reaction that has a given input, then the agent has to choose at least one of them. The criteria of the choice are left to the design phase of the simulation.

[4]Not to be confused with Ferber's environmental reaction

Formalization It has been seen that the reactions are considered and modeled like Meterios. A reaction, for an agent at_i, can be expressed by (17).

$$\begin{array}{rccc} Reaction_i : & \Gamma & \longrightarrow & \Sigma \\ & \Gamma_d^i & \longmapsto & E_t^d \end{array} \tag{17}$$

4.2.2. Goals

After the reactions, that drive the agents according to the events that may occur, agents are placed in the simulation with a reason. The goal notion justifies the presence of an agent. A goal is what the agent will try to reach during the simulation. There are at least as many goals as simulations and they are generally linked.

In the metamodel, goals are simply expressed by a desired state of the environment. Each agent has a set of goals that are linked to it by its decisional characteristics. Goals can be grouped into two categories: primary goals and subgoals. Primary goals are set by the designer of the simulation. Subgoals are set by the agent in response to a reaction or according to a plan to reach another goal.

Formalization A goal is a desired state of the environment, expressed by a E_t^d. Goals form the set Θ and the specific goals of an agent (primary or subgoals) ag_i form the set $\Theta_i, \Theta_i \subseteq \Theta$.

4.2.3. Plan of actions

The plan of actions is made according to the goals of an agent. It is a tree that is built considering the goals and the Meterios that are associated to an agent (see Figure 18). This tree is potentially rebuilt at each simulation step since new goals can be added to each of them. The decision to add new goals can be attributed both by the designer of the simulation or by the agent itself.

The complete behavioral process of an agent results in an action in the environment. We have already seen how the environment can be modified from a desired state of the the environment, the output of the behavioral process is a desired state of the environment. The whole process can be expressed by (18).

$$\begin{array}{rccc} Behavior_i : & \Gamma & \longrightarrow & \Sigma \\ & \Gamma_d^i & \longmapsto & E_t \end{array} \tag{18}$$

The whole behavioral process respects the main definition expressed in (18) but is in reality composed of four subprocesses, as expressed in (19).

$$\begin{aligned} Behavior_i(\Gamma_d^i) &= ActionSelection_i \circ MergeGoals_i \circ ExpandGoals_i \circ React_i(\Gamma_d^i) \\ &= ActionSelection_i(MergeGoals_i(ExpandGoals_i(React_i(\Gamma_d^i)))) \\ &= E_t^d \end{aligned} \tag{19}$$

Reactions produce desired states of the environment. The new states are new subgoals for the agent. The function $React_i$, as described in (20) performs

the process explained in the dedicated section: matching the perception with reactions to generate goals.

$$\begin{array}{rccc} React_i: & \Gamma & \longrightarrow & \Theta \\ & \Gamma_d^i & \longmapsto & E_t^d, E_t^d \in \Theta_i \end{array} \tag{20}$$

Once the new subgoals are set, the second step is an expansion step. The expansion consists in matching the goals with the output of the Meterios that are associated to the agent to determine what Meterio can be useful to reach a goal. Each Meterio may require a specific state of the environment to be executed, such as creating new subgoals. The tree that will drive the behavior of the agent is built step by step by following the described process. Finally, the process stops once the leaves in the tree are unitary Meterios. The expansion is expressed by the function $ExpandGoals_i$:

$$\begin{array}{rccc} ExpandGoals_i: & \Theta & \longrightarrow & \Theta \\ & \Theta_i & \longmapsto & \Theta_i \end{array} \tag{21}$$

The third step of the behavioral process is the merging of the various trees. Previous steps may have produced new trees that are not linked together. This step will reorganize nodes to merge all trees in one. It will find identical nodes in various trees to attach their parents, children and siblings. The designer of the simulation gives a heuristic to merge the trees. Heuristics can be execution time, arbitrary priority, etc. This merging function $MergeGoals_i$ is described in (22).

$$\begin{array}{rccc} MergeGoals_i: & \Theta & \longrightarrow & \Theta \\ & \Theta_i & \longmapsto & \Theta_i \end{array} \tag{22}$$

Finally, the last step consists in selecting the next desired state that will be transmitted to the rest of the simulation system. This desired state is selected among the ones resulting from a unitary Meterio. Similar to the previous step, the designer chooses a heuristic to perform a choice. The choice of the desired state on the others is made thanks to a heuristic that is left to the designer of the simulation. The selection function $ActionSelection_i$ is described below.

$$\begin{array}{rccc} ActionSelection_i: & \Theta & \longrightarrow & \Sigma \\ & \Theta_i & \longmapsto & E_t^d \end{array} \tag{23}$$

5. Simulation

This section presents the principles of the simulation, its life-cycle, according to our metamodel. The gathering of the simulation logs is also explained.

5.1. Principle of interpretation

Agents may send influences to the system to express their wish to change something in the physical environment.

To perform this operation, the function $Affect$ is in charge of checking the first element of an influence and generate the act if it is valid. This function considers the whole environment and the influence, and may produce an act.

$$\begin{array}{rccl} Affect: & I \times \Sigma & \longrightarrow & A \\ & \iota, \sigma_t^r & \longmapsto & \begin{cases} \alpha & \text{if } \iota \text{ is valid} \\ \emptyset & \text{otherwise} \end{cases} \end{array} \tag{24}$$

Next, once the corresponding acts are emitted, the last step consists in producing the changes. This is the role of the function *Operate* that considers the whole state of the environment and acts to produce a new real state of the environment at the next step of simulation (see (25)).

$$\begin{array}{rccl} Operate: & \Sigma \times A & \longrightarrow & \Sigma \\ & \sigma_t^r, \alpha & \longmapsto & \sigma_{t+1}^r \end{array} \tag{25}$$

According to the definition of this function, the equation (26) improves the definition of actions expressed in (8). An action process is the combination of these four functions.

$$\begin{aligned} Action &= Operate \circ Affect \circ Interpret \circ Meterio(E_t^r) \\ &= Operate(Affect(Interpret(Meterio(E_t^r)))) \\ &= E_{t+1}^r \end{aligned} \tag{26}$$

5.2. Simulation life cycle

Our metamodel is planned to work following the life cycle proposed in Figure 9. This life cycle is split into two main parts: the agent and environment parts. The first one is executed first and consists in, for each agent, performing the various steps that allow it to perform its behavior: perception, reaction processing, goals processing and finally the transmission of desired states that result from other steps. Once all agents have been processed, it is now the environment's turn. In this part, their are three main processes. The first one is the execution of the environmental dynamic to maintain a certain coherence in the simulation world. This step also produces desired states of the environment, which are processed during the next step. In the influence processing step, all of the influences are processed to check whether they are valid or not and produce associated acts. In the last step, all the acts are processed to apply changes and generate the environment for the next simulation step. Figure 4 illustrates the whole flowchart followed by the simulation life cycle.

5.3. Simulation logs

Our metamodel does not propose to generate simulation results on key points during the simulation but to use its ontology-based approach to capitalize on the whole simulation afterwards. Indeed, the ontology contains all data related to the simulation, it is possible to directly extract various results without having

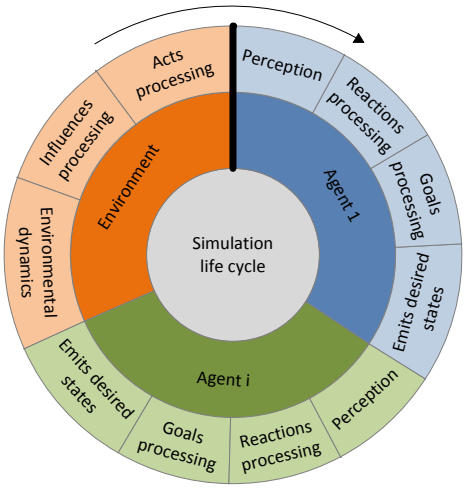


Figure 9: The simulation life cycle associated to our ontology-based and agent-oriented metamodel

to rerun the simulation by changing the key points. The metamodel proposes three approaches to store all the simulation steps. The first way to store the simulation logs is to add a "time stamp" property to each entity in the simulation and duplicate it with an updated time stamp at each modification. It is thus possible to trace the whole life of an entity. This solution is easily feasible but is really heavy in terms of file size and running time. The second way to store logs is to duplicate the whole ontology at each simulation step, having as many files as simulation steps. This method allows to easily modify something in the simulation and rerun it from a given point, but is heavy in terms of size and can be tricky to use for result extraction. Finally, the last method is to couple the ontology with a versioning system (such as SVN or Git) allowing to keep a lightweight ontology for the simulation but can be trickier for the result analysis.

6. Results analysis: Characterization of the agents

The metamodel allows to apply some existing agent definitions and to retrieve agents that correspond to a given definition during the simulation. This characterization is made according to the characteristics that are bound to the agents. The simple association of a characteristic is not enough to perform the characterization when the characteristic is about an interaction. Indeed, the ability for an agent to perform an interaction is involved in the characterization process, not only the association of this interaction to the agent. In the proposed metamodel, interactions are represented by Meterios. An agent definition is made of characteristics and interactions. To be performed, an interaction has some input conditions that must be satisfied. Thereby, an interaction capability must have its entry condition checked to be used during the characterization process.

To check if a Meterio is performable or not, a characterization operation must be made according to its entry conditions. The characterization is made on the ability for the agent to reach described states in the Meterio. To check if an agent can reach a state, the characterization system checks if the Meterio that allows to reach the state are associated to the agent. It is required to perform a characterization of the Meterio that may help to reach the required initial state.

The tree built during the goal expansion and merging steps drives the classification. The children of the characterized Meterio define if it is achievable or not. Either the input is already valid, or it is not valid but the agent has the Meterio to try to make it valid. In the second case, the system will characterize the required Meterios to reach the valid state. Either it is a unitary Meterio, or it is a compound Meterio. In the first case, it is considered as valid and the validation is reported for the initial Meterio. In the second case, the input of the compound Meterio is checked. If it is not valid, we check the agent has the Meterios required to reach the state. But this time, the validation of the input is restricted to unitary Meterio. Thus, the system considers each Meterio as valid. A valid Meterio for characterization must respect the pattern illustrated in Figure 10.

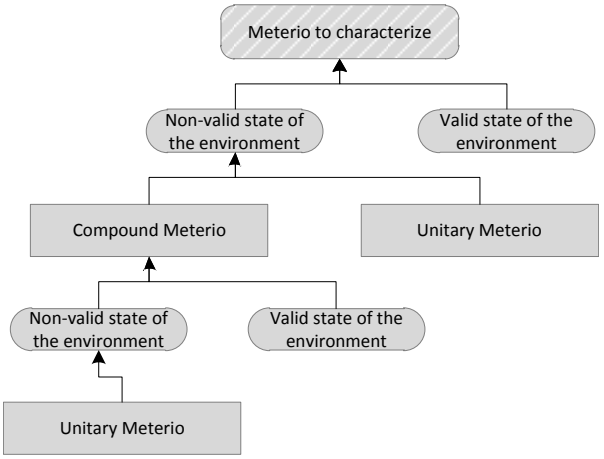


Figure 10: Pattern of a valid Meterio for the agent characterization

7. Application

This section describes a simple application of the proposed metamodel. This application takes place in the field of a situated multiagent-based simulation. The agents are situated in a 3D building. For the building representation, the Industry Foundation Classes (IFC) norm is used.

The next section describes the IFC format and usage. The scene used during the illustration is then illustrated and the goal is explained. After that, the

environment, and then the agents, are described. The run is then illustrated, and finally an illustration of the results analysis is made.

Note that everything in the application is represented in a simplified way in order to illustrate the principle without losing comprehensibility.

7.1. Industry Foundation Classes

Over the last fifteen years, the organization buildingSMART has been improving interoperability of software used in the construction industry. BuildingSMART produces specifications designed to facilitate exchange and sharing of information between software. The main outcome of its works was the language IFC (Industry Foundation Classes) which describes all the elements making the building as object classes. IFCs are homologated ISO / PAS 16739: 2005. Available in import / export with most of the new CAD tools for architects, this format also appears with other tools of engineering and design (structural analysis, thermal analysis, etc.) and facility management applications. The IFCs represent much more information as specific formats (DWG) which model the geometries. IFCs describe the true objects of the building, with their geometrical and semantic description. The class objects consist of triplets (GUID, OS, FU), which respectively correspond to a globally unique identifier (Global Unique Identifier), to the owner (ownership) and functional units. The GUID uniquely identifies the object in the plan, even if it has changed. So, in the case of an update of the IFC mock, finding the information attached to an object is very easy.

From the IFC, it is possible to dynamically build a digital model representing exactly the building that is or will be built. In our work, we use this standard to build the simulation environment and generate the simulation rules according to the semantics described in the IFC classes.

Multiagent-based simulations, combined with semantics, ontology and more generally the richness of data provided by the Industry Foundation Classes, provide tools to compute a collection of statistics and indicators about building quality, usability, comfort, etc. at early stages in the building life cycle. The reasoning system provides a classification of the gap between what happened during the simulation and what was expected to happen to check the accordance with rules and end-users' expectations.

7.2. Scene & target

The application takes place in an office for which an employee wants to reach. The scene is geometrically built as shown in Figure 11.

In this scene, the goal for an employee is to reach his desk, in the office on the right. Initially, the door is closed and locked. The agent thus has to unlock it before being able to open it. In order to unlock, it must get the key that is situated on the table in the room on the left.

7.3. Environment

The following subsections define the base model of an environment, and an instance of this model on the considered application.

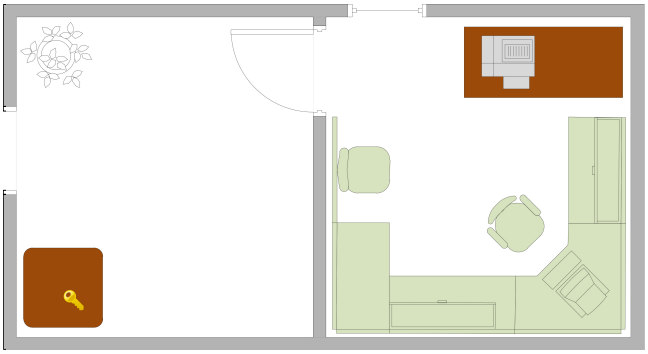


Figure 11: The scene of the application

7.3.1. Definition base for the environment

Figure 12 references the classes and links of objects that are present in the scene. Please note that some attributes (width and height) have been tripled in the figure for reasons of readability. This base represents the patterns that must be followed by entities to be characterized.

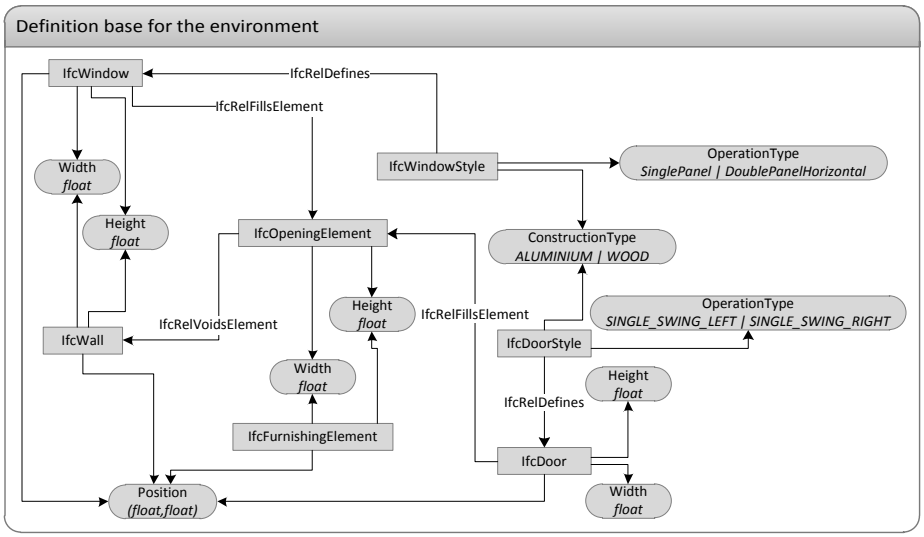


Figure 12: Definition base for the environment

In this figure, there are the representation of various IFC classes that correspond to what it is required to represent the scene illustrated in Figure 11. These definitions are based on the official IFC specifications and simplified for the illustration. Boxes with sharp edges represent classes, and boxes with rounded edges represent attributes. Arrows represent the association of an attribute to a class and a labeled arrow represents a link between two classes (the target

class is an attribute of the source class). All IFC classes and labeled relations are prefixed with the string "Ifc". The name that follows this prefix describes the nature of the object (IfcWindow for a window, etc.). An "opening element" is an extrusion in any object that has a physical representation. If an opening element intersects with any object, the intersection of these two elements is a void. The "construction type" attribute represents the material in which the related object is built. The "operation type" attribute is used for doors and windows to express their type of construction and usage (opening kind, etc.). The "furnishing element" represents objects that may appear in a plan but do not have a useful meaning for the building industry. For example, for the scene described in Figure 11, the tables, plant, printer and even the desk are considered as furnishing elements.

7.3.2. Environment instance

This section illustrates the ontological instances that are generated from the scene described in Figure 11. The generated ontology respects the metamodel and its principle. The ontology is illustrated on Figure 13.

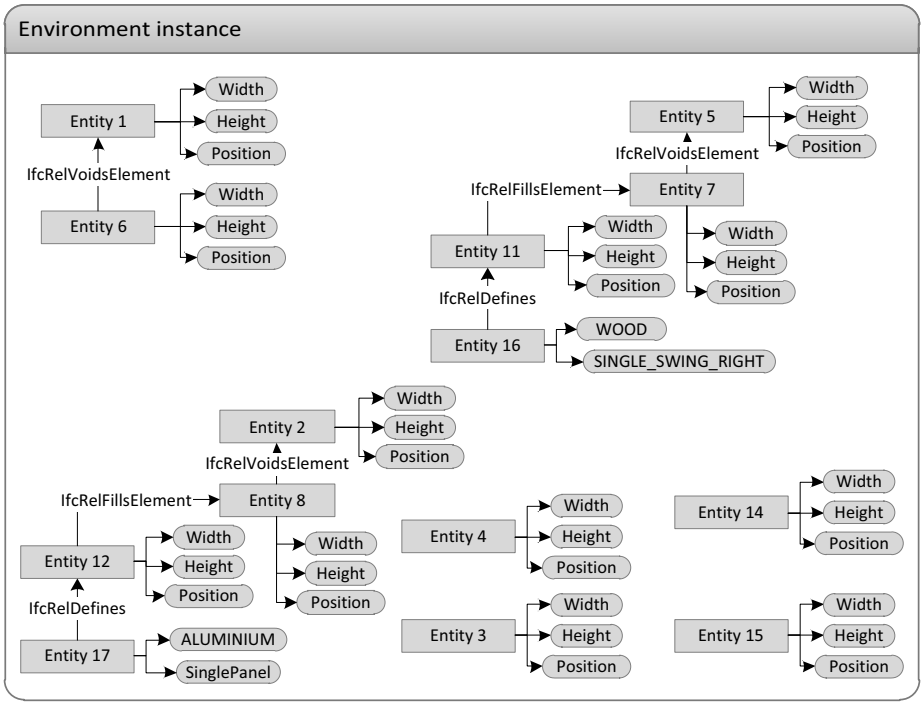


Figure 13: The ontology corresponding to the scene described in Figure 11

In this figure, all the names of the IFC classes have been lost, converted to some generic entities, as proposed by the metamodel principle. Note that the width, height and position attributes have kept their name instead of taking

a value for this application, because their value does not matter in this case. Attributes whose values matter have a value in the figure. This ontology instance is built from the IFC file, which is built following the IFC specifications, that is why it is possible to have the same pattern as the one described in Figure 12. The details of which entity represents what, and the mechanism to identify them, is described in Section 7.6.

7.4. Agents

This section describes the agents of the application. First, the Meterios that are usable by the agents are described, then the agents themselves (environmental et decisional characteristics).

7.4.1. Meterios

In this section, the Meterios that are available for the agents are described. These Meterios may be expressed with functions as input/outputs in order to simplify the expression and make it easier to understand.

The first Meterio available for the agents is the displacement Meterio. It consists in expressing the will to have a different position for any object that has a position in the environment. Its input thus consists in having a position in the environment and the output is the same object, but with its position that has changed. It is therefore a unitary Meterio for a situated agent

Figure 14: The Meterio that allows to perform a displacement

The second Meterio consists in taking an object in the environment. For this, the object must not be already taken by another agent and the agent has to be close to this object. The result of this Meterio consists in a modification of the object property that mentions its owner and the property of the agent that lists its goods.

Figure 15: The Meterio that allows to take an object

The third Meterio is used to unlock a door. To perform this operation, three conditions have to be met: the agent has to be close to the door, the door must be locked and the agent must have the key. Once this Meterio is performed, the door is unlocked.

Finally, the fourth Meterio is used to open a door. For this, only two conditions have to be met: the first one is that the agent has to be close to the door and the second is that the door must be unlocked. The resulting state of this Meterio is of course the opened door. Note that it is not required for the door

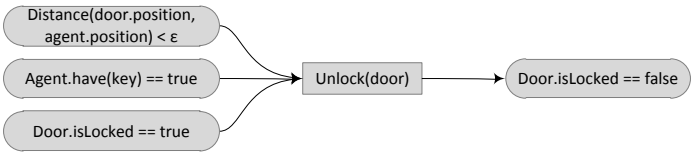


Figure 16: The Meterio that allows to unlock a door

Figure 17: The Meterio that allows to open a door

to be closed to be opened. This is because thanks to the principle of building trees for their behaviors in accordance with the last state of the environment that they have perceived, agents will not try to open a door that is already opened. Even if they try, this will result in a useless action in the environment, this requirement can thus be considered as void and it is thus not necessary to specify it.

7.4.2. Characteristics

Environmental characteristics. For this application, agents only have a position as a public environmental characteristic and a maximum speed as a private characteristic.

Decisional characteristics. For this application, agents have more decisional than environmental characteristics. Nevertheless, they have no public decisional characteristics. Their private decisional characteristics can be split into four categories: Meterios, attributes and goals. For the Meterios, all of the Meterios that are described in Section 7.4.1 are associated to the agents. For this application, agents do not have any goals. For the goal part, as described in the Section that describes the goal of the simulation, agents want to reach the desk in the room on the right. For its attributes, the agent has a list of objects that he may have in his possession. The part of decisional characteristics that is used to store the representation of the environment obtained through a perception is implicit and each agent has necessarily this in their decisional characteristics.

In this application, agents have a priori knowledge of the state of the environment and thus behavior to get this kind of information is not necessary.

7.4.3. Definitions

In this application, there are only two kinds of agent definitions. The first one is "mobile agents", which describes an agent that is able to move on his own. The second one is "employee", which describes an agent that is able to open a door that leads to an office.

7.5. Running

In this section, the running of the application is described. The application only has one agent who have to reach his desk.

The goal is thus to reach a state of the environment where the agent is close to his desk. In order to reach this state, the agent has to move close to it. The agent can thus add the task "move" to his behavioral tree. Unfortunately, the desk is in a separate room. Thanks to the semantic provided by the IFC, the agent knows that he can pass through a wall thanks to an opened door. He can thus add the task "open(door)" to his behavioral tree. Unfortunately again, the door is locked and the agent thus has to unlock it before being able to open it. In order to unlock it, the agent needs the key to the door, which is on the table, and to be close to the door. He can thus add two children in his behavioral tree, the first one with the task to move close to the door and the second to take the key. To take the key, the agent has to move close to it, he can thus add the task "move" to its behavioral tree in order to be able to perform the first operation.

The agent has thus completely built his behavioral tree, only by using the semantic provided by the environment to help building the plan of actions, and the input and outputs of the Meterios to chain the good path to reach its goal. The whole behavioral tree is visible in Figure 18

7.6. Characterization

This part describes the characterization process for both agents and environment.

7.6.1. Environment

We can see in Figure 13 that there is a lot of entities to characterize, and a lot of them are very similar. For example, all entities except 16 and 17 have the same properties. It is thus required to perform a comparison between the definition base and the instance. For instance, entities 1 and 6 are linked by an "IfcRelVoidsElement" relation. According to the definition base, thanks to this link, it is deductible that entity 1 represents a wall and entity 6 represents an opening in this wall. By following this pattern, it is possible to determine that entities 5 and 2 are walls and entities 8 and 7 are openings in these walls. Following the "IfcRelFillsElement" properties that are bound to entities 7 and 8, we can determine that entities 11 and 12 are either a door or a window. Unfortunately, both entities have the same characteristics and it is not possible to directly use their properties to determine their nature. It is thus required to characterize entities 16 and 17 first, then the nature of 11 and 12 may be determined. The characteristics "WOOD" and "ALUMINIUM" do not allow to characterize them since these values can be used for both windows and door styles. Nevertheless, thanks to the "SINGLE_SWING_RIGHT" it is possible to determine that entity 16 is a door style and entity 11 is therefore a door. In the same manner, thanks to the "SinglePanel" characteristic it is possible to tell that the entity 17 is a window style and therefore that entity 12 is a window. Unfortunately, in the state of the application it is not possible to

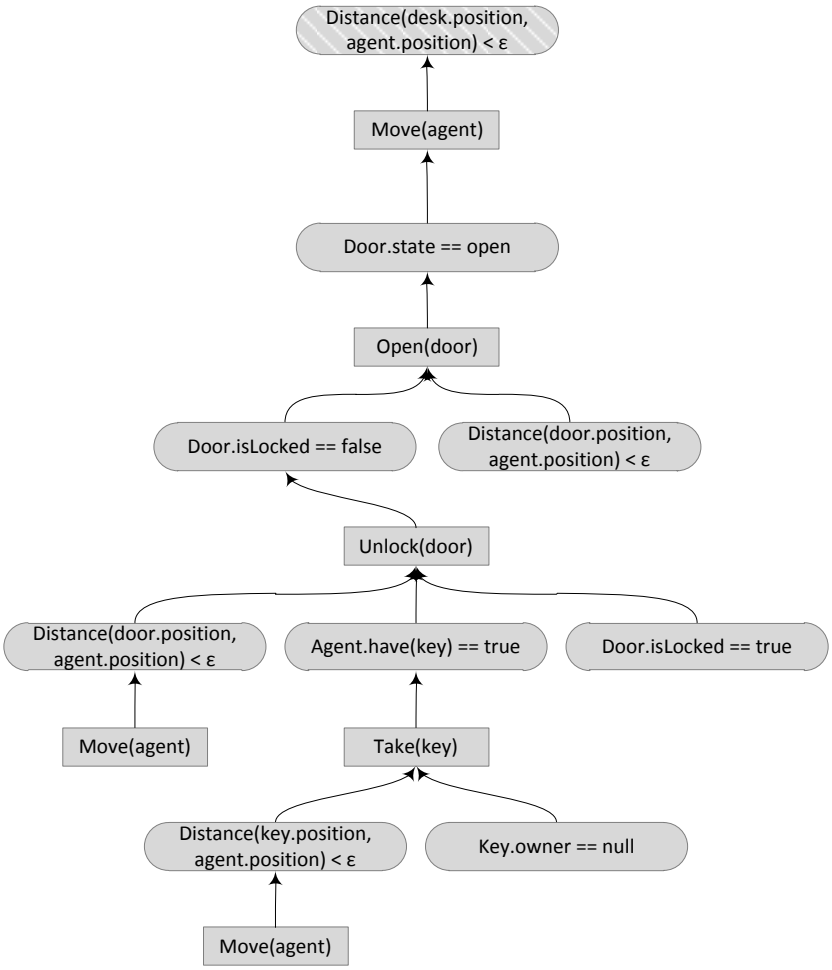
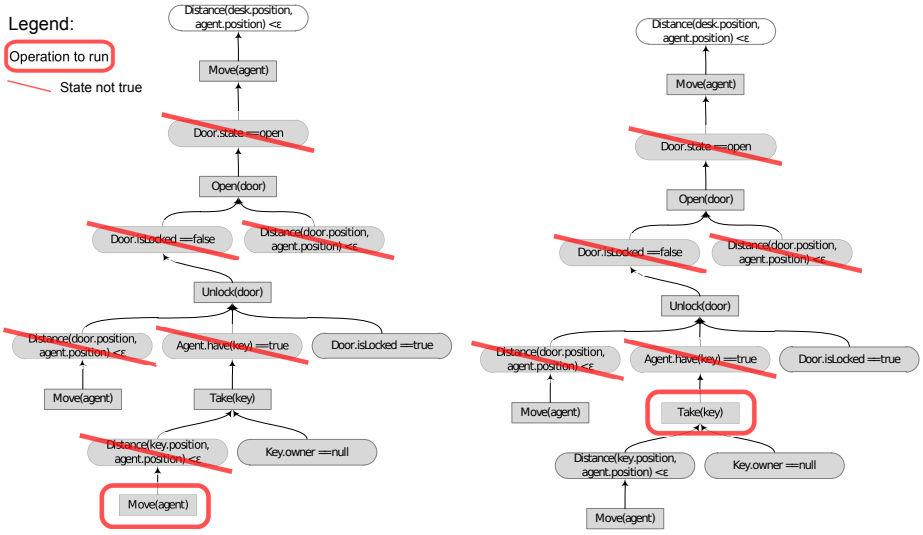


Figure 18: Behavioral tree built by the agent

determine whether entities 3, 4, 14 and 15 are walls or furnishing elements. This limitation is due to the simplified IFC specifications that are used for this application.

7.6.2. Agent

For this application, there are two kinds of agents: "mobile agents" and "employees". To perform this characterization, the principle explained in Section 6 is used. The agent is able to move on his own as soon as the simulation begins, he is thus considered as a mobile agent from the beginning. For the second definition, it is required to characterize the "open" Meterio in order to determine whether it is an employee or not. To open the door, the agent has to be close to it and be able to unlock it. None of these requirements are met



(a) Search the key (b) Pick up the key

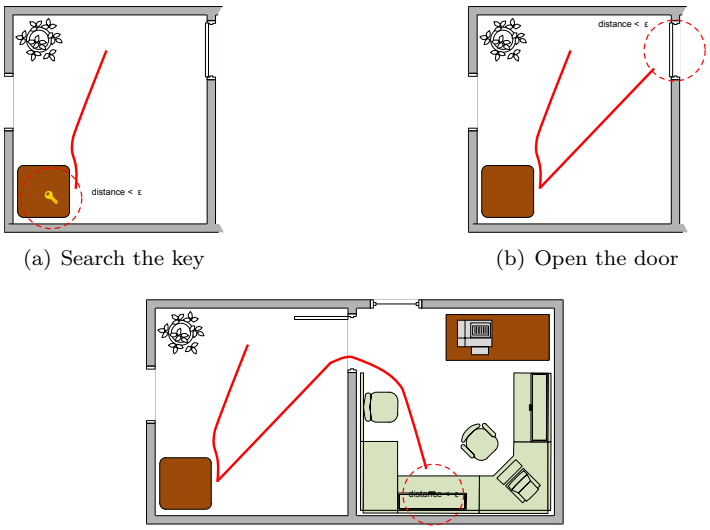
Figure 19: State of the behavioral tree during the simulation

at the beginning, so it is required to characterize his children in the behavioral tree. Being close to the door is a unitary Meterio, this requirement is thus valid. Being able to unlock the door requires having the key, which the agent does not have at the beginning. Taking the key is not a unitary Meterio since it has its own requirement to be valid. But, in this configuration, the requirements to take the key are either already met or reachable with the help of a unitary Meterio. In this case, the agent is thus considered as an employee as soon as the simulation begins. If the key was in another room or already taken, the agent would not have been considered as an employee because more interactions would have been required.

7.6.3. Results

According to the behavioral tree in Figure 18, the agent accesses the tree state at each step of the simulation for determining the next action to execute. Figure 19 illustrates the different states of this behavioral tree during the simulation process. When the agent is spawned, it scans its perception and accesses the condition nodes of the tree (Figure 19-a). The action to execute at a given time is the lowest action in the tree that is associated to a false condition node. Figure 19-b) illustrates the update of the behavioral tree when the agent arrives near the key. Then, the agent is able to execute the action "Take(key)". This evaluation is repeated until the top-most node condition is true, and during this process the behavioral tree is updated according to an algorithm adapted from the RETE algorithm [11].

Figure 20 illustrates one run of the simulation. The agent is spawned at a



(a) Search the key

(b) Open the door

(c) Go to the desk

Figure 20: Path that is followed by the agent during the simulation

random position inside the left room. Initially the door is closed and the key is on table in the room on the left. The collision avoidance algorithm used to move the agent is based on a force-based model provided by Buisson et al. [2]. The agent is attracted to the position of its goal, and a collection of repulsive/sliding forces are computed to avoid collision with obstacles.

8. Background

This section compares our metamodel with other models using ontologies for multiagent-based simulations. It is difficult to find proposals which combine a representation of agent, environment and interaction dedicated to a large range of multiagent-based simulations. Existing models are usually domain-dependent or focused on a specific part of the simulation.

In 1997, Franklin and Graesser [12] listed many agent-related approaches and definitions and tried to establish a taxonomy of agents.

We have already explained the problem of this approach (see Section 2) There are too many different definitions of agents, that even sometimes conflict. Instead of a taxonomic approach, we propose a generic way to describe agents, with which various definitions can be expressed according to user expectations and domain-related constraints.

Using semantics is a key point for multiagent-based simulations. Many authors work on integrating semantics in their simulation in many different levels: environment, agents, etc.

De Antonio et al. [6] describes an agent model to manage training in virtual environments. In their proposal, they use agents to manage the various parts of the simulations: an agent manages the path planning, another manages the geometrical description, another manages the semantic description, etc. This is a completely different strategy to agentify the simulation. Each part of the simulation is centralized in the hand of one agent but the individuals are not truly autonomous. The decision-making process is shared between different agents of the platform. In this way, it breaks a number of principles at the heart of multiagent-based simulations. Furthermore, the semantic part of their model is not clearly explicited and remains unclear. Finally, the approach remains highly dependent on concepts of the training field.

A more flexible approach is presented in [15]. The authors use an OWL ontology to manage the knowledge of an agent. Their model is implemented with help of the JACK framework. The latter is a multi-agent platform that uses the BDI[5] software model and provides its own Java-based plan language.

Their proposal uses all the functions allowed by the ontologies to manage the knowledge of the agents: inferences, class hierarchy, SPARQL requests, etc. They perform a quite traditional use of the ontologies to manage agents' knowledge (the ontologies are designed to manage and contain knowledge). It remains a very intersting approach.

Edward et al. [7] describe a more expressive approach. The authors propose to manage agents, environment and interactions in an ontology. They use the MOSS language, drawn from LISP, to define their ontology. This ontology represents knowledge needed to train a person to manipulate objects in a high-risk environment. It models interactions with, or on an object, as well as the results produced by an interaction. This approach describes a checking process on the requirement states for an interaction. The presented model is relatively complete.

Like De Antonio et al. [6], their approach is targeted to the training in a virtual environment. They propose a way to represent knowledge about the environment and how to interact with it, i.e. what can be done with or on an object. Each interaction must be described in a peer to peer manner, i.e. what happens if the user uses a specific object on another specific one. The result is also described here for each possible state of each object. The objects have thus some predefined usages that are set prior to the running of the simulation. Populating such ontology is time consuming since it needs to describe everything for each object (state, interaction, transition rules, etc.). Nevertheless, authors propose an automation of the possible interaction by proposing a process that analyzes the properties of each object and adds related interaction to the object. However, this process is not made during the simulation and must be a-priori performed to the simulation. In our approach, it is not necessary to create these new relationships, they are dynamically determined during the simulation. To sum up, they expose the advantage of using ontologies: expressing behavior in

[5]Belief, Desire, Intention

a easier way and managing environment in a high level manner.

Chang et al. [5] focus on interactions. The authors use ontologies to model the interaction between agents and environment. This modeling is rather complete and exploits the ontology functions such as inferences, etc. Their approach describes goals as some states of the environment and use inference mechanism to reach them. Limited to the interaction part, this model is close to ours.

Tsai et al. [19] provide a service-oriented approach to integrate an ontology into a robot software. If this model is mapped to an agent-oriented approach, each agent can use an ontology engine through a service-based architecture. We consider that this approach is a base of the integration of an ontology in a multiagent system. It suffers of similar drawbacks as for Chang et al. [5], and for Holmes and Stocking [15]. Moreover, from a conceptual point-of-view, the model does not include the concepts related to the simulation principles (influence/reaction...) as core components of the ontology. Our purpose is not simply using an ontology by the agents, but also defining agent-oriented concepts into this ontology. In this way, each object described by the ontology that should be supported/simulated by an agent, may be automatically or dynamically selected according to the ontological rules and the simulation scenarios. From an implementation point-of-view, the works proposed in Tsai et al. [19] are directly related to Microsoft Robotics Studio®. It is incompatible with our implementation choices.

In general, the existing models do not consider the ontology as a central repository to manage every kind of data related to a simulation (interaction, environment, agents, etc.) including results and logs. In our approach, the ontology is the core component of the simulation used as a basis for agent behavior, extracting all required data to perform a simulation step and storing simulations results.

Everything required or produced by the simulation is stored in the ontology. This enables to restart the simulation at/from any point and even modifies some of its characteristics before the restart. Simulation results may also be directly analyzed and used during the simulations.

9. Conclusion

In this paper, we have presented a new ontology-based metamodel for agent-based simulation that fully exploits the advantages of ontologies.

This model fits the standards of multiagent-based simulations by respecting the principle of influence/reactions [8, 17], the separation between an agent's body and mind [9, 13], the principle of behavioral planning based on ontologies [5], etc. In short, it ensures a clear separation of concerns, between the different parts of the simulation in order to limit potential biases in the simulation.

This allows to easily design agents by only specifying high-level goals and letting them determine the sequence of sub-goals/actions that are required to reach their goals. The relations between the goals (or the sub-goals) and the

actions are extracted from the ontology. The designer of the simulation scenario does not need to specify all the behavioral tree by hand. She/he only needs to give the top-most goal to the agent. If the agent has a complete knowledge of the environment, it is possible to use the build behavioral tree to validate the correctness of the scenario. Otherwise, the correctness and the validity of the scenario cannot be warrantied. The environment part is managed in the same way as an agent, thus allowing an easy representation and a quick way to perform an agentification of the environment. The metamodel allows the representation of any kind of agent model by allowing a characterization after the simulation. The simulation is thus not stuck with a given agent or environment model. Moreover, the results of simulation made with models are not predefined and can be extracted directly from the logs. The same logs also allow to dynamically modify the environment or the agents at any step of the simulation and run a branch of the initial simulation from this point without doing anything, since the agents are able to adapt themselves dynamically. The future works regarding these works consist in a real implementation and application in order to help the qualification of the building usage from the simulation logs and usage rules [3].

Aknowledgements

This work is supported and funded by the Regional Council of Franche-Comté, France. We thank Caroline Tabard for her help in reviewing this paper.

Bibliography

[1] Florian Béhé, Christophe Nicolle, Stéphane Galland, and Abderrafiaa Koukam. Qualifying building information models with multi-agent system. In *the 3rd International Workshop on Multi-Agent Systems Technology and Semantics (MASTS 2011)*, pages 309–314, Delft, The Netherlands, oct 2011. Springer. ISBN 978-3-642-24012-6.

[2] Jocelyn Buisson, Stéphane Galland, Nicolas Gaud, Mikaël Gonçalves, and Abderrafiaa Koukam. Real-time collision avoidance for pedestrian and bicyclist simulation: a smooth and predictive approach. In *2nd International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications (ABMTRANS13)*, Halifax, Nova Scotia, Canada, June 2013. Elsevier.

[3] Florian Béhé, Thomas Durif, Christophe Nicolle, Stéphane Galland, Nicolas Gaud, and Abderrafiaa Koukam. Ontology-based multiagent systems using inductive recommendations. In *Design & Decision support systems*, 2012.

[4] Florian Béhé, Christophe Nicolle, Stéphane Galland, and Abder Koukam. Semantic management of intelligent multi-agents systems in a 3d environment. In *Intelligent Distributed Computing V*, volume 382 of *Studies in Computational Intelligence*, pages 309–314. Springer Berlin / Heidelberg, 2012.

[5] P. Chang, Y.H. Chien, E. Kao, and V.W. Soo. A knowledge-based scenario framework to support intelligent planning characters. In *Intelligent Virtual Agents*, pages 134–145. Springer, 2005.

[6] A. De Antonio, J. Ramírez, R. Imbert, and G. Méndez. Intelligent virtual environments for training: An agent-based approach. *Multi-Agent Systems and Applications IV*, pages 82–91, 2005.

[7] L. Edward, K. Amokrane, D. Lourdeaux, and J.P. Barthès. An Ontology for Managing a Virtual Environment for Risk Prevention. In *Integrated Intelligent Computing (ICIIC), 2010 First International Conference on*, pages 62–67. IEEE, 2010.

[8] J. Ferber and J.P. Müller. Influences and reactions : a model of situated multiagent systems. In *Second Internationnal Conference on Multi-Agent Systems (ICMAS)*, pages 72–79, 1996.

[9] Jacques Ferber. *Les Systèmes Multi-Agents : vers une intelligence collective*. InterEditions, 1995.

[10] Jacques Ferber. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison Wesley, 1999. ISBN 978-0201360486.

[11] Charles Forgy. RETE: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[12] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. *Intelligent Agents III Agent Theories, Architectures, and Languages*, pages 21–35, 1997.

[13] Stéphane Galland, Nicolas Gaud, Jonathan Demange, and Abderrafiaa Koukam. Environment model for multiagent-based simulation of 3D urban systems. In *the 7th European Workshop on Multiagent Systems (EUMAS09)*, Ayia Napa, Cyprus, December 2009. Paper 36.

[14] Nicolas Gaud, Stéphane Galland, Franck Gechter, Vincent Hilaire, and Abderrafiaa Koukam. Holonic multilevel simulation of complex systems: Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*, 16(10):1659–1676, nov 2008.

[15] D. Holmes and R. Stocking. Augmenting agent knowledge bases with OWL ontologies. In *IEEE Aerospace conference*, pages 1–15. IEEE, 2009.

[16] Fabien Michel. *Formalism, tools and methodological elements for the modeling and simulation of multi-agents systems*. PhD thesis, LIRMM, Montpellier, France, December 2004.

[17] Fabien Michel. Le modèle IRM4S : le principe Influence/Réaction pour la simulation de systèmes multi-agents. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, 2006.

[18] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, (second edition 2003), 1st edition, January 1995. ISBN 0137903952.

[19] W.T. Tsai, Xin Sun, Qian Huang, and Helen Karatza. An ontology-based collaborative service-oriented simulation framework with microsoft robotics studio®. *Simulation Modelling Practice and Theory*, 16(9):1392–1414, 2008. ISSN 1569-190X. doi: http://dx.doi.org/10.1016/j.simpat.2008.07.007. URL `http://www.sciencedirect.com/science/article/pii/S1569190X08001421`.

[20] R. Vanlande, C. Nicolle, and C. Cruz. Ifc and building lifecycle management. *Automation in Construction*, 18(1):70–78, 2008.

[21] D. Weyns, A. Omicini, and J. Odell. Environment as a first-class abstraction in multiagent systems. *Journal on Autonomous Agents and Multiagent Systems*, 14(1), 2007.