Ontology-based multiagent systems using Inductive Recommendations

A new approach to qualify building use during the Design phase

Florian Béhé[1,2], Thomas Durif[2], Christophe Nicolle[2], Stéphane Galland[1], Nicolas Gaud[1], Abderrafiaa Koukam[2]

[1]*IRTES-SET, UTBM,*
UPR EA 7274
90010 Belfort Cedex
France
florian.behe@checksem.fr
stephane.galland@utbm.fr
nicolas.gaud@utbm.fr
abder.koukam@utbm.fr

[2]*LE2I – UMR CNRS 6306*
IUT Dijon-Auxerre, Université de Bourgogne
BP 17867, 21078 Dijon Cedex,
France
thomas.durif@checksem.fr
cnicolle@u-bourgogne.fr

Key words: Multiagent Systems, IFC, Ontology, Semantic, Induction, Recommendation

Abstract:

DRAFT OF THE PAPER PUBLISHED IN :
In Proc. of International Conference of Design & Decision Support Systems (DDSS), Eindhoven, Netherlands, 2012.

1. INTRODUCTION

Simulation is now a common tool to analyze quality of building in terms of thermal behavior, energy consumption, environmental impact, indoor pollutant flow, etc. Our approach consists in using 3D behavioral simulation to simulate the use of the building in its final configuration to assess the adaptation of the building configuration facing its future use by its various types of end-users. The idea consists in simulating the behavior of end users of a building in everyday usage scenarios or emergency scenarios to assess the quality of a building in terms of safety (e.g. evacuation, validation of the signage), usability (e.g. maneuverability of large objects), comfort (e.g. amount of sunshine), etc., right from the design phase. The final goal aims at developing a software application to compare and assess buildings configuration at the design phase to assist decision makers in their strategic choice.

The qualification of building use is a task that requires expertise in the application's domain of the considered building. This task may be performed at different phase of the building life cycle: usually during the design phase before the construction, sometimes during operation and maintenance phase. For changing building use, two kinds of expert are at least required: a domain expert for qualifying end-users expectations and an architect to validate desired changes specified by the domain expert. In this paper, we propose an approach to assist the work of experts and to allow end-users to validate/compare their proposals. Our approach is based on the combined use of multiagent-based simulations, Industry Foundation Classes and ontology reasoning.

Multiagent-based simulation.

Multiagent systems are a powerful paradigm combining concepts from Artificial Intelligence and Distributed Computing allowing the effective implementation of complex distributed software applications. Multiagent-based simulations (MABS) are simulations that involve a collection of interacting software agents. An agent is an autonomous computing entity able to perceive and interact with its environment and other agents. Each agent follows a set of rules that will define its behavior used to achieve its personal goals. Agent's behavior relies on environment's local perception, individual knowledge and interactions with other agents. The environment is composed of various objects that can be manipulated by agents (agents are themselves a specific kind of object), relations between these objects,

operations/operators to manage interactions between agents and their environment [1]. Multiagent systems can be used to solve problems that are difficult for an individual agent or a monolithic system to solve.

In our approach, MABS is used to simulate different scenarios of building use to establish a complete diagnosis of the building's quality in terms of safety, usability, comfort, etc.

Industry Foundation Classes.

Last fifteen years, the organization *BuildingSMART*[1] improves 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 ...) and facility management applications. The IFC represents much more information as specific formats (DWG) who modeled 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, find the information attached to an object is very easy. The script 1 describes a building with more than 111,000 business objects (one line per object).

```
   ISO-10303-21;
   HEADER;
   FILE_DESCRIPTION (('ArchiCAD generated IFC file.'), '2;1');
   FILE_NAME  ('Karlstr.IFC',  '2002-06-19T15:48:48',  ('Architect'),
('Building  Designer  Office'),  'PreProc  -  IFC  Toolbox  Version  2.x
(00/11/07)', 'Windows System', 'The authorising person.');
   FILE_SCHEMA (('IFC2X_FINAL'));
   ENDSEC;
   DATA;
   #1 = IFCORGANIZATION ('GS', 'Graphisoft', 'Graphisoft', $, $);
   #3 = IFCPERSON ($, 'Undefined', $, $, $, $, $, $);
   #4 = IFCORGANIZATION ($, 'OrganizationName', $, $, $);
   #5 = IFCPERSONANDORGANIZATION (#3, #4, $);
   #7 = IFCSIUNIT (*, .LENGTHUNIT., $, .METRE.);
   …..
   #111029 = IFCRELCONTAINEDINSPATIALSTRUCTURE
    ('25wKeWex98fQp5Pukf_Ilc', #6, 'BuildingStoryContainer',
   'BuildingStoryContainer   for   Building   Elelements',   (#111007),
```

1 http://buildingsmart.com

```
#110989);
   #111030   =   IFCRELAGGREGATES   ('216Bv$yJj3tQjFeDohe6fQ',   #6,
'BuildingContainer',  'BuildingContainer  for  BuildigStories',  #30,
(#34,  #16236,  #29699,  #56800,  #62077,  #67336,  #72633,  #91702,
#110989));
   #111031   =   IFCRELAGGREGATES   ('17XMUtNDr8FeFMtR6rOcy5',   #6,
'SiteContainer', 'SiteContainer For Buildings', #28, (#30));
   #111032   =   IFCRELAGGREGATES   ('0pMN8yq8vDRfwN_tnJREKC',   #6,
'ProjectContainer', 'ProjectContainer for Sites', #26, (#28));
   ENDSEC;
   END-ISO-10303-21;
```

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 simulation combined with semantic, 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, … at early stage 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 in order to check the accordance with rules and end-users expectations.

This paper is structured as follows. Section 2 presents a background of what have been made in the domain of multiagent-based simulation to introduce semantic in the environment to allow richer behaviors and a more precise exploitation of the simulation's results. Section 3 presents our proposal to help the qualification of buildings during the design phase, using MABS and IFC. Section 4 illustrates our proposal through a simple example and finally Section 5 concludes this paper.

2. BACKGROUND

Environment is one of the main parts that compose a multiagent system. In the case of situated agents, an environment must solve various problems. These problems are about the topological and geometrical descriptions of the world, managing its dynamics and the semantics of each object and area of the world. Up to now, the last part was usually neglected, and simulations were run relying mainly on geometrical data. Using only geometry is not enough for enabling behavioral simulations to generate meaningful results and deploy complex agent's behaviors. Indeed, you cannot expect anything

else than simple displacements within the environment if your agents are not able to capture the semantics of what they perceive.

Some works have been made in order to introduce the notion of semantic in the environment for multiagent-based simulations. The inclusion of semantic in the environment has often been mentioned by works made by various research teams but is almost never the main subject of on-going research. Nevertheless, it is possible to identify some key points that are in many propositions. One of these points is the concept of tagging that is often used to introduce semantics. These tags are built in many ways, from a label associated to an object [2] to the repartition of objects between various layers in the environment representation [3]. Some proposals, like [4] propose to model a high level of semantics thanks to tags and links between them. These tags allow to identify specific objects among others or represent interactions that are possible or not with a given object. The main default of all these proposals is that to put semantic in your environment, you have to perform some specific tasks in addition to your environment modeling. These tasks can be prior to modeling by naming objects following a given convention or after the modeling by manually adding labels and tags in your environment. Sometimes, they can provide other aspects than just bringing a meaning to an object: for example in [5] they use the tagging to build a navigation graph in order to facilitate the displacement of agents during the simulation. Moreover, all these approaches do not rely on any kind of standard and are specific to simulation platform or even to a specific kind of simulation. In a few words, it is hardly reusable.

We thus propose a new way to represent data used and produced during a simulation, especially during a behavioral simulation in a 3D virtual environment. This approach provides a high level of semantic and brings an answer to the main drawbacks mentioned in this section. This approach consists in an ontology-based metamodel for multiagent-based simulations and is described in the next section.

3. PROPOSITION

The principle of the metamodel is the same for both main parts of a MABS: environment and agents. The idea is to have a single instance of ontology in which are stored various entities to which some characteristics and properties are associated (as shown on Figure). These entities are not instantiations of some ontological concepts related to a specific domain but some instantiations of a generic OWL concept, such as “owl:Thing”. Entities described in this ontology may as well be related to environment, as agents

or even parameters for the simulation. Thus, by owning only this ontology, it is impossible to distinguish the nature of a given entity. This ontology can therefore be seen as a melting pot of knowledge used and produced during the simulation.

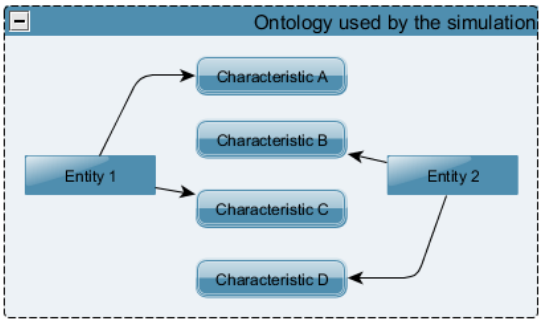

Figure Example of ontology used by the simulation

The idea is thus to bring additional knowledge to the ontology used by the simulation. This knowledge is used to characterize the entities contained in the ontology. This characterization allows identifying and classifying instances between environment, agents or parameters. This characterization is made thanks to some base of definitions as shown on Figure .

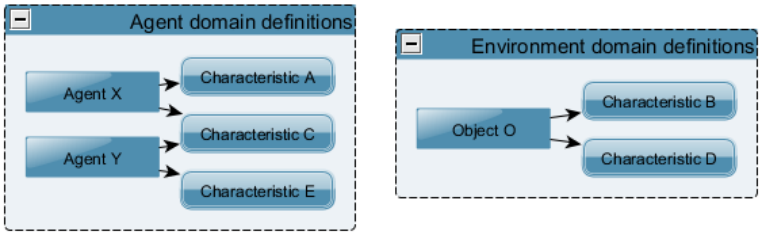

Figure Example of base of definition

These bases are used in addition to the ontology in order to perform the classification, as shown on Figure . This characterization is based on characteristics that are associated to entities. The presence (or not) of some of them is thus decisive for the characterization following some pre-established criteria regarding the domain of application (environment, agent, etc.). The ontology used by the simulation is thus not linked to one or more agent or environment definitions, in particular. The characterization of entities is therefore made in a dynamic way regarding the bases of definition and the characteristics of each entity. Such an approach is almost essential given the multitude of possible definition of the “Agent” concept in the field of multiagent systems. So everyone can at will propose its personal

definition of “Agent” and classify the entities that have the characteristics associated with this definition.

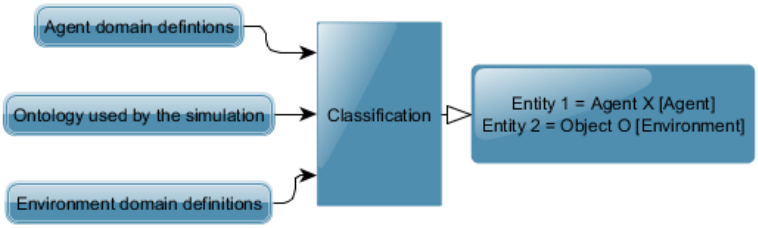

Figure Classification of entities

The application of this principle for the knowledge representation allows a great flexibility in the usage of semantic. It allows, in particular, to do not limit the knowledge of a simulation to a pre-defined model for a specific kind of simulation, thus allowing adding any kind of data to our simulation without having to redefine a whole model for a particular usage. The proposed metamodel is applicable in a wide range of Multiagent-based simulations (situated agents, sensor network, etc.) by allowing the application and identification of elements by using many definitions.

3.1 Environment

Our application domain requires the simulation of situated agents within buildings. We use IFC file as a source for the generation of the simulation environment. Our environment is thus composed of a geometrical part and also a semantic part. As mentioned in the previous section, our proposal is applicable to both agents and environment.

Although our proposal enables the representation of any kind of data, we have chosen to split semantic and geometrical representation of the environment. The semantic is obviously stored in the ontology following the principle mentioned in Section Erreur : source de la référence non trouvée but the geometrical part is currently stored in a COLLADA file[2] [6]. The link between the COLLADA and the ontology and kept using a Globally Unique IDentifier provided by the IFC specifications. A GUID is associated to each object in IFC files and this GUID is reported in the ontology as a characteristic associated to entities and reported in the COLLADA as an “extra” data that the specifications allows to add to each object. The usage of an external file dedicated to the geometrical representation allows to perform geometrical requests (extracting lengths, volumes, etc.) easily than loading

2 Could also be another common 3D format like FBX or OBJ, etc.

the whole IFC structure. Moreover, such geometrical data is reusable in simple simulations of visualization.

The definition base is generated from the IFC specifications in order to easily identify entities generated from the IFC. *BuildingSMART* provides IFC specifications under the EXPRESS format that is easily parsable by a tool to automatically generate the base of definition of the environment.

The environment generation process is shown on Figure . Basically, it loads an IFC file and extracts geometry objects. From this objects, the geometry is computed and stored in a COLLADA file, associating to each object the GUID to retrieve the IFC object from which the object is generated. For its part, the semantic data is used in order to generate an entity and characteristics are generated and associated to it following the semantic data provided by the IFC. The entity is then stored in the ontology, waiting for its exploitation.

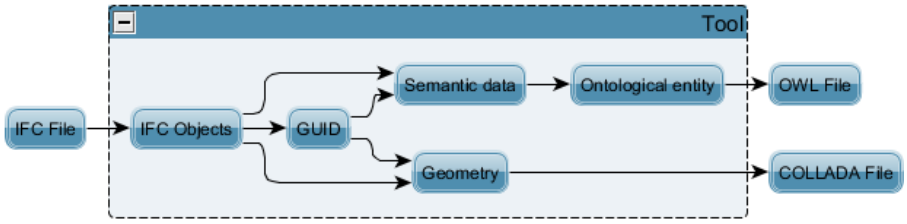

Figure Environment generation process

By using this approach, we base our environment on standards that are easily reusable in other applications than simulation. The input file is a standard in the building industry and the produced files that contain our representation do not destroy data and also rely on standards. For our application, the process produces OWL and COLLADA files from an IFC, but the reverse process is also possible. We thus have on one hand a purely geometrical representation, and on the other hand, a purely semantic view of the building. And thanks to the GUID, we can easily retrieve from information from one to the other.

3.2 Simulation outputs: results and logs

Simulations are generally run in order to obtain some specific results regarding the simulation environment and type. Usually, results are produced on some precise key points during the simulation. These key points are determined before the start of the simulation, and the simulation is generally entirely structured around them. Thanks to our proposal, we do not propose to produce the results in the same time that the simulation is run, but we

propose to generate the results from the logs of the simulation. In our case, the logs contain the whole data used by the simulation, thanks to the principle of storing everything related in an ontology. Using this kind of storage, we can thus have various manners to store various states. We have in a first place the principle to store the whole ontology in a new file at each simulation step. This method produces a lot of files, and can be tricky to exploit, but it allows to easily change trace each step and re-run the simulation from a specific step. In a second way to produce logs, we can also add a “timestamp” characteristic to each entity in the ontology and duplicate these entities with updated data (including the timestamp). This method has the same pros than the first one and is friendlier to exploit. Nevertheless, it has the drawback of having a heavy ontology at the end of the simulation that can be long to load and exploit. Finally, the third method is coupling to log file in a subversion system such as SVN or Git and execute a “commit” at each simulation step. This solution is trickier to use than the first one but has the same pros, and it is also as lightweight as possible for each step of simulation.

The logs of simulation, in our application case, are exploited by some induction principles described in the next section.

3.3 Learning agent

Our ongoing research is to build auto-adaptable agents. These agents will be able to create new executions plans to meet the simulation goals and their personal goals. When an agent can't solve a problem in the simulation, the agent must adapt its behavior to discover new ways to resolve the problem. To bridge this gap, we use machine learning techniques.

Machine Learning is derived from Artificial Intelligence and enables a system to evolve through a process of learning. Machine learning is based on the desire to create models to simulate human's intellect abilities used to progressively improve performances with repeated experience of a situation.

Concretely, it's to make inferences from observations of facts. There are four methods of reasoning for our purpose: the deduction, the induction, the abduction and the Bayesian networks. The deduction is a reasoning in which conclusion is true if the assumptions are exact. The inductive reasoning is a process of generalization from observations of facts. Abduction also implies a link between facts and their causes and extends the previous methods by discovering new links between causes and unknown facts. Bayesian networks are models that can represent situations of probabilistic reasoning from uncertain knowledge. The objective is to obtain a probability of occurrence for a scenario corresponding to a path in the set of variables.

Others reasoning methods like neural networks and Markov models are not well suited for our purpose. Neural networks don't explain the result and we need to know why a result is provided. Markov models don't taking into account experience.
In our work, the agents are placed in a situation to match certain specifications. Each agent has a collection of personal goals and is defined by a set of physical characteristics and autonomous decision-making. Agents with a similar target will not have the same behavior if they have different characteristics. Agent can't directly exchange information. They have to evolve freely, independently from the monitoring system of the other agent. To achieve their goal, the agent passes through a set of states. Nevertheless, some states can't be reached directly. The agent has to find a complementary path to resolve this problem. Knowledge from environment and ontology will help the agent to solve this problem and resolve this set of sub-objectives that are unknown states. For that, we established a process divided into two phases. The learning phase and the simulation phase. The learning phase, confront unskilled agents (understand not having a predisposition to solve problems) to the simulation. This make that agents "learn" to escape from situations not directly soluble. During this step, agent improves gradually their skill. At the end of this phase, the set of agents becomes "smart". The simulation phase consists in using these smart agent architectures in a specific environment with real constraints. Our approach helps to easily configure agent's skills according to a real modeling of the environment and to obtain a set of smart agent to qualify the use of buildings at the design step. A formal description of this part is beyond the scope of this paper. Nevertheless, the next section depicts an example of the first steps of our approach.

4. EXAMPLE

This section draws the generation and the classification of the environment from a simplified IFC file. We are working with the definition shown on Figure , which also represents our base of definition. This example contains four IFC classes: "IfcWindow" describes a window; "IfcWindowStyle" describes the style of a window (opening type, material); "IfcDoor" describes a door and "IfcDoorStyle" describes the style of a door (opening type, material). Window and door have two direct attributes (width and height) and a relation with another IFC class. The relation attribute describes the style of the window or door. Obviously, the style can be different for each class (a door cannot have a window style). Both styles

have a construction type: ALUMINUM which means that the door or window is aluminum and WOOD logically means that the object is wood. Styles have an "OperationType" attribute with different values. This attribute describes the style of the object. For windows, the style can be "SinglePanel" which means that the window is in a single block or "DoublePanelVertical" which means that the window is split in two parts vertically. For the doors, the "OperationType" describes the aspect of the door and the opening style. If this attribute has the value "DOUBLE_SWING_LEFT", that means that the door is in one panel, can be opened in both directions with hinges on the left. In the same way, if the attribute has "DOUBLE_SWING_RIGHT", that means that the door is in one panel, can be opened in both directions but this time with hinges on the right.

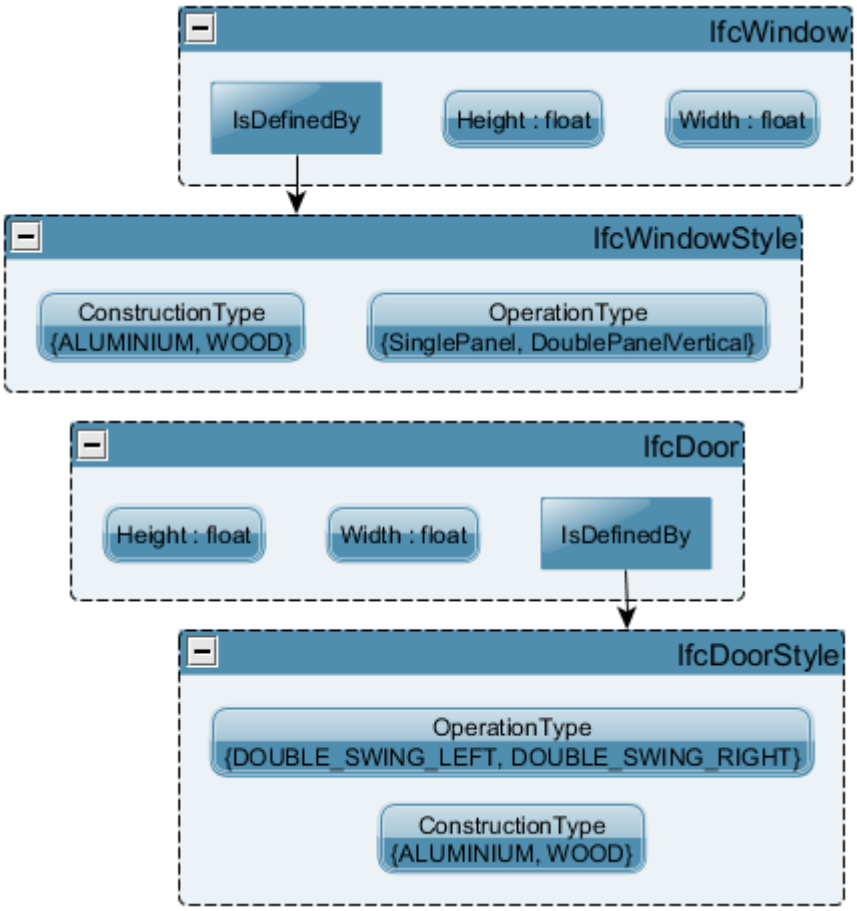

Figure Base of definition

Figure represents the content of an IFC file that will be used for the generation of the environment. This file contains four entities: a window, a door and the styles that are related to each. By following descriptions in the previous paragraph, we can determine from the IFC file that the described window is about 1 meter width for 50 centimeters height, is composed of a single panel and is wood. The door, for its part is about 1 meter width for 2 meters height can be opened in both directions and its hinges are situated on the left and finally, said door is aluminum.

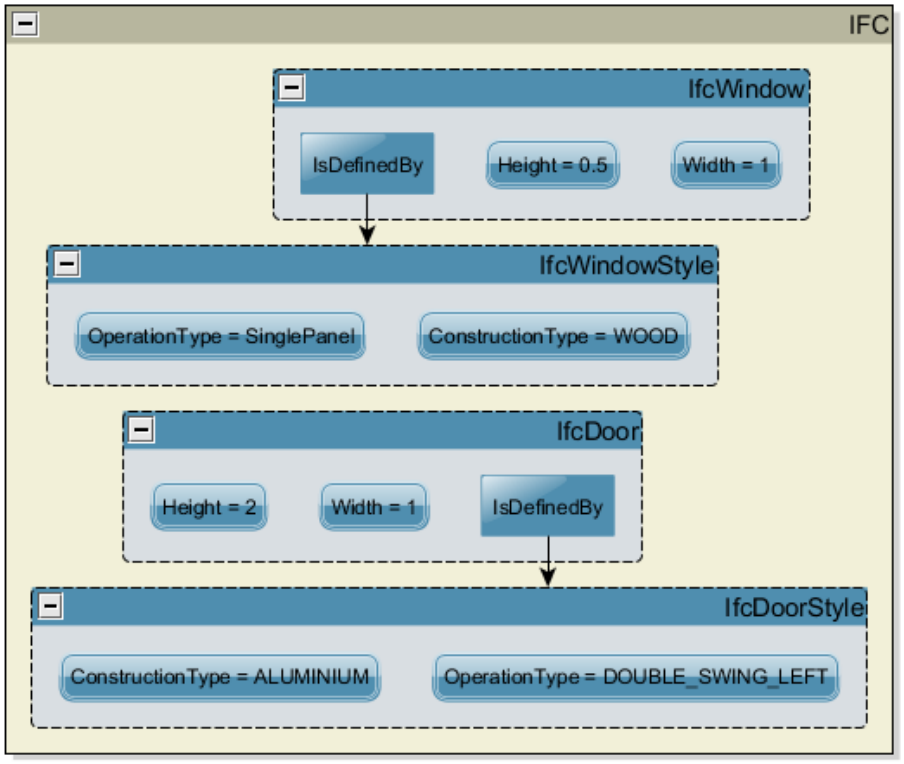

Figure Content of IFC file

Figure represents the ontology generated from IFC file. The exploration of each IFC object builds the ontology. The example depicts four IFC objects. Thus, the exploration process generates four entities in the ontological file. IFC attributes are converted into properties. Plain arrows represent data type properties. These arrows correspond to "direct" attributes in IFC files. Dashed arrows represent object properties. They are links between objects. These properties and their values will be useful for the classification process described in the next paragraph.

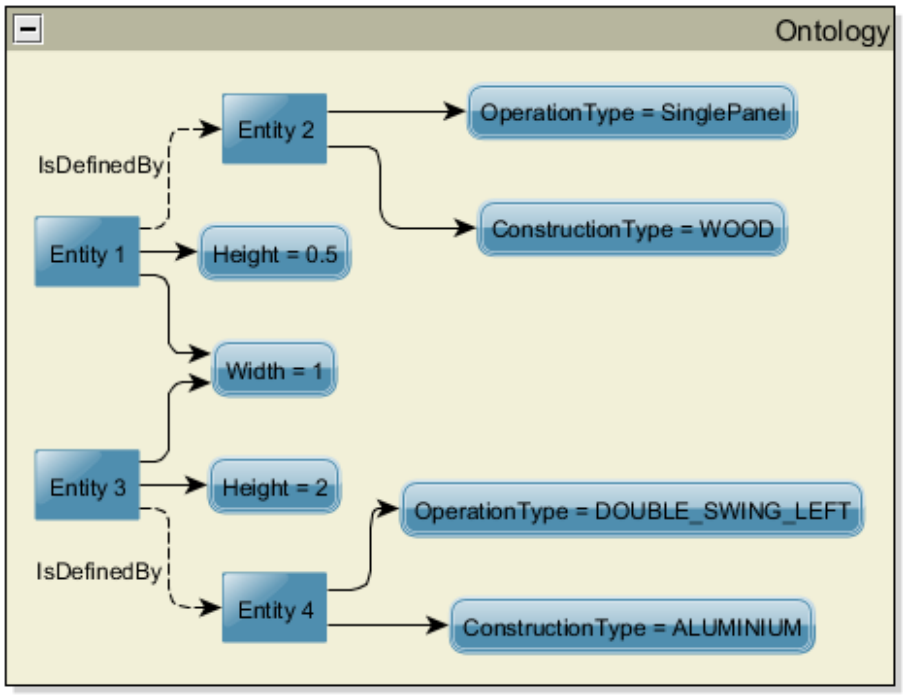

Figure Content of generated ontology

The classification is made using the ontology (Figure) and the base of definition (Figure). We can see that four ontological entities have been generated and that there are four objects in the IFC. Regarding the definitions, entities 1 and 3 are both respecting the characteristics of “IfcWindow” and “IfcDoor”. We can also see that they are respectively associated to entities 2 and 4. These entities also have characteristics which can be used to characterize them. Regarding the characteristics of Entity 2 (resp. 3), we can determine that this entity is an instance of “IfcWindowStyle” (resp. “IfcDoorStyle”). Now we have characterized entities 2 and 4, to which entities 1 and 3 are associated, we can perform the characterization of these entities. Thus, by making a correspondence between the base of definitions and the left uncharacterized entities, we can determine that Entity 1 (resp. Entity 2) is an “IfcWindow” (resp. “IfcDoor”) since it is associated to an “IfcWindowStyle” (resp. “IfcDoorStyle”) object.

TODO

Peut-être ajouté l’image qui décrit la structure de l’environnement et surtout un exemple d’arbre décisionnel (arbre d’objectif) d’un agent

5. CONCLUSION

This paper presents the fundamental architecture of a multiagent –based simulation model for the building qualification. The basic approach is primarily based on the generation of the simulation environment based on geometric and semantic information stored in IFC files, standards in the field of construction. The multiagent simulator is also coupled to a semantic engine that can dynamically store all the simulation outputs. These elements are then used to develop complex behaviors of self-adaptive agents to assess/validate different scenarios of buildings use designed according to specific needs of end users.

6. BIBLIOGRAPHY

[1] J. Ferber, Les Systèmes Multi-Agents : vers une intelligence Collective, InterEditions, 1995.

[2] D. de Paiva, R. Vieira and S. Musse, "Ontology-based crowd simulation for normal life situations," *Computer Graphics International,* pp. 221-226, 2005.

[3] N. Farenc, S. Musse, E. Schweiss, M. Kallmann, O. Aune, R. Boulic and D. Thalmann, "A paradigm for controlling virtual humans in urban environment simulations," *Applied Artificial Intelligence,* vol. 14, no. 1, pp. 69-91, 2000.

[4] J. Lugrin and M. Cavazza, "Making sense of virtual environments: action representation, grounding and common sense," *Proceedings of the International Conference on Intelligent User Interfaces,* pp. 225-234, 2007.

[5] B. Yersin, J. Maim, P. de Heras Ciechomski, S. Schertenleib and D. Thalmann, "Steering a virtual crowd based on a semantically augmented navigation graph," *Proc. The First International Workshop on Crowd Simulation (V-CROWDS'05),* pp. 169-178, 2005.

[6] F. Béhé, C. Nicolle, S. Galland and A. Koukam, "SEMANTIC MANAGEMENT OF INTELLIGENT MULTI-AGENTS SYSTEMS IN A 3D ENVIRONMENT," *International Conference of Knowledge Engineering and Ontology Development (KEOD),* 2011.

[7] S. Galland, N. Gaud, J. Demange and A. Koukam, "Environment model for multiagent-based simulation of 3D urban systems," *7th*

European Workshop on Multi-Agent Systems, 2009.

[8] D. Weyns, A. Ominici and J. Odell, "Environment as a first-class abstraction in multi-agent systems," *Journal on Autonomous Agents and Multi-Agent Systems,* vol. 14, pp. 5-30, 2007.