Semantic Modelling of Virtual Environments Using MASCARET

Pierre Chevaillier, Thanh-Hai Trinh, Mukesh Barange, Pierre De Loor, Frédéric Devillers, Julien Soler, Ronan Querrec

Université Européenne de Bretagne, France
ENIB – LISyC, Centre Européen de Réalité Virtuelle, F-29280 Plouzané

ABSTRACT

Many Virtual Reality (VR) applications, such as Virtual Learning Environments or Interactive Virtual Tours, are based on a rich semantic description of the environment and tasks that users have to perform. These applications are built upon Virtual Environments (VEs) in which artificial agents act autonomously while interacting in real-time with users. Semantic modelling of a VR environment makes it possible the knowledge-driven access from the description of VEs that simplifies the development of VR applications. It eases the development of these types of applications. Semantic modelling should provide a consistent representation of the following aspects: 1) The simulated world, its structure and the behavior of its entities, 2) Interactions and tasks, that users and agents can perform in the environment, 3) Knowledge items, that autonomous agents can use for decision-making or for communication with users. This paper presents MASCARET, a model-based approach, for the design of semantic VR environments. This approach is based on the Unified Modeling Language (UML). In this approach, UML is used to provide a knowledge-driven access to the semantic contents of the VE and not for code generation, as in classical software development process. Interests of a UML-based approach are that its metamodel covers different views of the semantic modelling: ontology, structure, behaviors, interactions, activities. It is also an extensible language that can be specialized to provide formal operational semantics. We also present how MASCARET can be used to develop content-rich interactive applications that can be deployed over various VR platforms. Finally, we discuss the benefits of such a metamodel-based approach and show how the multi-layer semantic model can be used in different VR applications, in which adaptive behaviors of artificial agents acting within complex environments have to be simulated.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities, Evaluation/methodology; I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods—Representation languages I.6.5 [Simulation and Modeling]: Model development—Modeling methodologies

1 INTRODUCTION

Though there are recent advances in the field of software engineering, developing Virtual Reality (VR) applications is still considered as a difficult task, both from theoretical and technical point of view.

Many efforts have been made in order to ease the development process. These include developing VR platforms as far as possible independent to both hardware and software specific features [4] and avoiding the development of VR applications completely from scratch. A number of high-level design methodologies have also been proposed (see [19] for a recent state-of-the-art report). At last an insightful idea was the introduction of a conceptual modelling phase into the development process [7].

These endeavors contributed to the wide-spread use of VR applications. The representation of Virtual Environments (VEs) has been moving from a simple computer-generated environment to become a more intelligent, human-oriented and semantically rich one. In such VEs, knowledge about the environment itself has been made accessible to the users and the same knowledge representation could be processed by artificial agents using AI Techniques, acting within the rich-semantic environments, called Intelligent Virtual Environments (IVEs) [1]. By sharing the same knowledge model, agents could communicate with users in order to collaborate with them. As stated in [14], the development of such VR applications should be based on a core method: semantic modeling. The benefit of semantic modeling is that it provides the basic knowledge and information about the environment and the system, which are necessary to produce intelligent computer-simulated behaviors or to be processed by a specific problem-solving or decision-making support module.

In this paper, we present a methodology and a framework, named MASCARET, for the design of semantic VR environments. The core of our proposition is that this model-based approach is grounded on a metamodel which is a specialization—and an extension—of the Unified Modeling Language (UML) and which covers all the aspects of VEs semantic representation: ontology of the domain, structure of the environment, behavior of the entities, both user’s and agents’ interactions and activities.

This paper is structured as follows. Section 2 discusses issues and benefits of semantic modelling for VEs. We briefly review the existing work and highlight the relevance of a model-based approach for a high-level semantic specification of the VE. In Section 3, we present a brief overview of MASCARET framework, and discuss the process used to develop VR applications based on MASCARET. We then detail the main components of the framework. Section 4 illustrates how MASCARET can be used to build various rich-content interactive applications, e.g., VEs for learning or cultural heritage applications. Section 5 concludes the paper.

2 ISSUES AND BENEFITS OF SEMANTIC MODELLING

2.1 Related Work

Current approaches on semantic modelling fall into two main categories, corresponding to two levels of semantics that they specifically address: content-oriented semantic or system-oriented semantic.

Content-oriented semantic modelling aims to define an explicit representation of scene content. Thus, knowledge about entities can be obtained without accessing any system-level information. In most cases, the approach is based on an ontology which contains

*e-mail: chevaillier@enib.fr
†e-mail: trinh@enib.fr
‡e-mail: barange@enib.fr
§e-mail: deloor@enib.fr
¶e-mail: devillers@enib.fr
‖e-mail: soler@enib.fr
∗∗e-mail: querrec@enib.fr
some common and basic semantics of the VE, as defined in [9]. That is, every object compounding the virtual world owns a name, a position, and other features like its width and height. In [10], the authors extract the scene contents by analyzing input files, built upon 3D standards like VRML and X3D. The elements from these files (e.g., nodes and relations between nodes) are then mapped to the concepts of the ontology, called X3DOntology, which provides a semantic representation of the scene. Following the same principle, but at a higher level, [16, 12, 23] made the correspondence between objects populating the virtual world and concepts in ontology.

Unlike content-oriented semantic modelling, system-oriented semantic modelling is mainly inspired from the principle of reflection in software engineering. At execution time, the platform provides meta-access to the features of the entities, by means of meta-API. Such an approach, called multi-layer semantic reflection, has been found in [13]. Alternatively, [21] proposed the concept of semantic object encapsulating abstract information—such as text, numbers or graphs—to create so-called information-rich virtual environments. In this approach, semantic objects are obtained using APIs giving access to VRML (External Authoring Interface - EAI) and X3D (Scene Access Interface - SAI) file formats.

### 2.2 Open research questions

Even though the interest about semantic modelling of the VE has gained, this novel modelling paradigm is still relying on immature technical and conceptual bases. With regard to the main components of the VE, as defined in [11], semantic modelling should provide a consistent representation of the following aspects: (1) The simulated world, its structure and the behavior of its entities; (2) Interactions and tasks, that users and agents can perform in the environment; (3) Knowledge items, that autonomous agents can use for decision-making or for communication with users.

While content-oriented semantic modelling enables a semantic description of the structure of the VE, the main drawback of using this modelling is that, other aspects of modelling of the VE (e.g., interactions, communications and behaviors) are not covered. Typically, this technique has been applied to annotate objects of an existing VE with semantic information. The second strategy, system-oriented semantic modelling relies on the assumption that, the execution platform should grant runtime access to application’s entities. Even though the approach is said to be based on multi-layer semantic reflection, semantic information is accessed through low layers and using platform specific services. Here, a question arises with regard to the experience learned from the conceptual modelling of VEs that have been discussed so far: Is it possible to introduce semantic modelling in the early phase of the development process and thus also allow a high-level access to the semantics of VEs? In the following paragraphs, we advocate the relevance of our model-based approach, in which UML is a central common modelling language, as a promising way for the development of realtime, interactive and semantic VR applications.

### 2.3 Why a UML-based approach for semantic modelling?

Now let us address the issue of the most suitable modelling language that could support semantic modelling of VEs, that rises the question: is Yet Another Modelling Language (YAML) necessary or not? Rationale for the design of a new Domain Specific Language (DSL) can be tackled, regarding the relative positioning over two axes of the semantic of the new language, compared to an already exiting one (here UML): its coverage of the domain and its abstraction level (Fig. 1). If abstractions conveyed by the two modelling languages are at the same level and are strongly overlap with those of UML, then it is more suitable to define the new DSL as an extension of UML. Our motivations to ground MASCARET on UML are detailed in the following paragraphs. They enlighten our preference for the “UML scenario” over the “YAML” one.

UML is an extensible modelling language and its extension mechanism has clear semantics. This mechanism is based on the definition of new profiles that extend or specialize the semantics of UML for specific domains or contexts. Initially it was targeted to software engineering but nowadays many other profiles have been proposed to tailor the language to specific areas, e.g., systems engineering, business modelling.

UML is a relevant basis to cover both system- and content-oriented semantic modelling. Table 1 shows the different modelling layers supported by UML and the corresponding layers of abstraction in MASCARET. The vertical organisation of these modelling layers supports the introspection mechanism needed for system-oriented semantic modelling of VR applications and the horizontal one illustrates how content-oriented semantic modelling is addressed.

UML comes with two upper layers (M2, M3). The UML metamodel is defined using MOF, which is the minimal subset of UML required to specify the semantics of any modelling language (in the YAML scenario, the DSL could be defined using MOF, figure 1). UML being defined by using UML, endows the language with reflection mechanism.

Developing a system using UML requires to define the user domain model (M1), and afterwards to produce one instantiation of this Platform-Independent Model for a specific need, on a specific deployment platform. The latter model (M0) is thus a Platform-Specific Model, corresponding to the end-user application. Assuming that the semantics of the platform is formally defined, it is possible to verify that the execution of the model is consistent with the user domain model (M1). From a Model-Driven Engineering perspective, this consistency relies on the development of domain-specific extensions of UML, such as MARTE for real-time systems. It is therefore possible to use the user domain model (M1) as a support for introspection and online monitoring of the execution. In MASCARET, the system-oriented semantic modelling is based on this general principle.

The different semantic layers also allow to tackle issues related

---

### Table 1: Modeling layers (M0-M3) for UML

<table>
<thead>
<tr>
<th>Layer</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0</td>
<td>UML metamodel</td>
</tr>
<tr>
<td>M1</td>
<td>UML user model</td>
</tr>
<tr>
<td>M2</td>
<td>MASCARET metamodels</td>
</tr>
<tr>
<td>M3</td>
<td>MOF (UML restriction)</td>
</tr>
</tbody>
</table>

### Figure 1: Criteria for building a Domain Specific Language (DSL) on UML or not: relative overlapping of the new DSL and UML

left: much overlap - reusing UML scenario  
right: little overlap - YAML scenario
to content-oriented semantic modelling within the same framework. The metamodel layer allows to formally define a content description language that can be specialized for a specific domain (here intelligent virtual environments). With this language, it is therefore possible for experts to produce Platform-Independent Models of their domain ($V_E$ models) that can be afterward instantiated into many VEs and loaded onto a VR platform ($V_{Ei}$ models). Being instantiated in MASCARET, both M1 and M0 models can be used as knowledge base, for instance for virtual agents to collaborate with users to perform a task into the VE.

2.4 Domains covered by semantic modelling

The main assumptions regarding the models supporting the semantic abstraction layer in MASCARET are as follows. (1) The VE is composed of physical objects, named entities, that belong to predefined types (or classes). These objects are consist of parts and/or they can be logically linked together. Types of object are defined by a set of properties, relationships and invariants. (2) Space and time are continuous; entities occupy a region of space (they have a shape, even deformable and they can overlap or not); the execution of an action has a duration and the events are instantaneous. (3) Any change in the environment and any action execution are potentially observable. (4) Entities exhibit reactive behaviors and thus can be viewed as a discrete event systems. (5) At the finest grain, behaviors consist in the execution of atomic transformations of some properties of a set of entities (even empty). (6) A subset of the entities of the VE are autonomous agents, namely virtual humans or animals. They are situated agents, having limited perception field, finite capacities and bounded rationality. (8) Parts of their behaviors can be described as reactive systems (stimuli trigger responses); others, namely deliberative behaviors, can be simulated using cognitive architectures that process knowledge items (facts and rules). (9) Activities that agents and users are supposed to perform can be hierarchically broken down into sub-activities and finally into atomic actions. (10) It is possible to define, for each action, the feasibility conditions and the rational effects as logical propositions upon some properties of a set of entities. (11) Collaborative activities involving autonomous agents and users are governed by explicit organisational rules or constraints.

In the domain of multiagent systems, which is an overlapping domain of IVEs, different authors have chosen to ground their meta-models on UML, like [17, 15, 3], and their work can be partially reused or extended. To our knowledge, it has never been done for virtual reality.

To summarize, the suitable semantic modelling language for VR applications has the same level of abstraction than UML. UML is a sound theoretical basis to support both system-oriented and content-oriented semantic modelling in a consistent way. UML does not cover all the requirements of semantic modelling of VEs and, therefore, needs to be specialized for virtual reality domains. The next section presents how we have fulfilled these objectives with MASCARET.

3 The MASCARET Framework

This section gives an overview of the main components of the MASCARET framework and their relationships. It explains how we have reused, extended and specialized UML.

3.1 Overview

MASCARET stands for MultiAgent System for Collaborative, Adaptive & Realistic Environments for Training [6]. Although it was initially dedicated to the Virtual reality Environment for Human Learning (VEHL), it has also been applied for developing various types of VR applications (section 4). MASCARET is a generic framework that provides the necessary abstractions for both content-oriented and system-oriented semantic modelling of VEs and their deployment on different execution platforms. On the one hand, MASCARET provides a modelling language expressive enough for experts to formulate their knowledge about a specific domain (content-oriented approach). On the other hand, the language has clear operational semantics for the models to be interpreted by the execution platform (system-oriented approach). To tackle these two issues, UML metamodel has been tailored to VR context. The conceptual diagram in Fig. 2 presents the main subsystems of MASCARET and their dependencies. These elements are detailed in the next paragraphs.

![Figure 2: Conceptual overview of the main components of the MASCARET framework.](image-url)

MASCARET is based on two complementary metamodels, VEHA and HAVE. The former, VEHA, is dedicated to the modelling of the entities composing the VE, their types, structures and their behaviors (see 3.3). The latter, HAVE, allows to describe the interactions and the activities that both users and artificial agents may perform in this VE (see 3.4). Because the effects, or the feasibility, of these activities can have an impact on, or be dependent to, the location, the actual properties, the state, or the behavior of the entities from the VE, HAVE uses concepts of VEHA. HAVE allows to describe activities that can be interpreted as predefined collaborative scenarios (procedures), as plans of action for artificial agents or as instructions provided to users for assisting them. The way the description of the activity is interpreted by the agents is defined using behave (see 3.5).

MATS is the subsystem of MASCARET dedicated to the tutoring functionalities of a VEHL. Because it is specific to this kind of application, it has not been described in this paper. A comprehensive description of the agent architecture used in Mats, called Pegaso, can be founded in [5].

The two packages named “3D RIS services” and “Agent Management Services” play the role of facade. They offer the services needed to implement the Platform-Independent Models of MASCARET on a specific execution platform. The diagram on figure 2 makes no assumption whether these sets of services are provided by modules embedded in the same execution environment or not. The former subsystem ensures the wrapping of functions classically offered for the realtime 3D rendering and for the handling of the interaction peripherals. Moreover, its main added value is to ensure the execution of the behavioral models from MASCARET. For instance, it may be necessary to couple the module managing the behavioral state-machine of MASCARET with an already existing physics en-
3.2 Developing applications using MASCARET

Before explaining the concepts evolved in MASCARET, we show here how an end user may create his specific VE using MASCARET.

First, the domain expert defines a model of VE (M1 model) in the form of UML–MASCARET diagrams. He has to describe the class models, behavioral models (state machines and activities) and the human activities by using UML collaboration and activities diagrams. This step is completed using a UML model which supports metamodels defined as UML profiles. This domain model is exported in an XMI file.

Second, 3D designers have to construct geometrical objects using a classical 3D modeler. A MASCARET plugin is added to 3D modeler in order to refer the UML model (XMI file) and then, designers add semantics to geometrical objects which are then defined as instances of the domain model (M0 model). Many VEs can thus be constructed based on the same M1 model. Those environments are exported in a file referencing the model used (XMI file) and describing the links between 3D files and instances of the model.

These two steps describe creating the M1 and M0 Platform-Independent Models. In the third step, the end user may use a specific platform to instantiate, execute and interact with his specific VE. This is dedicated to MASCARET implementation. It instantiates the M0 and the M1 model, execute entities and agents behaviors and makes a link between VR peripherals used by the end user and the VE.

3.3 VEHA metamodel

Formally, VEHA is a profile that merges existing packages from the UML’s metamodel and adds some properties specific to interactive 3D simulations. VEHA provides necessary abstractions for the modelling of specific entities of the domain relative to the application. It allows to describe the types of the entities, their structural and behavioral features, their logical relationships and their topological properties (see 3.6).

Concepts of Class or Property naturally match with those of type and attribute commonly found in ontologies. We have extended the concept of Class onto entityClass to explicit the difference between general concepts and objects having a physical appearance, which is specific to VR. The semantic of the Association from UML is specialized in (1) conceptually related to (2) physically linked to, and (3) structurally compound of (stereotype composition). Figure 4 illustrates how an entity of type Desk would be described using VEHA. The definition used here came from WorldNet [8].

Entities of VEs may have some behaviors, which are represented in VEHA as reactive behaviors. These behaviors can result from the interaction between physical objects, user interactions or internal activities of the entities. VEHA reuses the semantics of the behavioral model of UML, which is based on the concepts of State, Transition and Event, but we have extended the set of basic types of Event. Contrary to UML, in VEHA state machines are explicitly associated to entityClass. They have therefore a more specific usage. When an event occurs, assuming that the state of the entity is sensitive to it, one (or more) transition(s) can be fired, a reaction may also be triggered and then the entity enters into another state(s). The reaction can be an activation of another state machine, the execution of an explicit set of elementary actions (described as an activity chart), or an opaque behavior, i.e., a specific piece of code for which no semantic representation is given. Figure 5 gives an example of the modelling for the sliding behavior of a drawer (refer to the Drawer class on Fig. 4).

Figure 4: Partial modelling of the Desk entity class using VEHA.

Figure 5: Modelling of the sliding behavior for a Drawer using VEHA.

3.4 HAVE metamodel

HAVE is a metamodel to describe the interactions and activities that autonomous agents and users can collectively perform in the VE. HAVE is specifically dedicated to describe how the different activities of agents have to be organised, in terms of time, space and responsibilities. The concept of organisation is one of the key elements of HAVE.
Organisational structures describe the way concrete organisations are created. The concepts of collaboration and organisation already exist in UML, even if they are rarely used. We enriched these concepts, which are in HAVE more agent-oriented than object-oriented, but we still use the UML syntax. Therefore, it is possible to design organisations using UML collaboration diagram and the specific extensions provided by HAVE. Figure 6 illustrates informally the core concepts of organisational models in HAVE. Roles are major structural elements of organisations. Definition of a role differs through different existing agent models. In some of them, a role is the description of a goal, and in other, a role specifies a set of actions to be done by the agent which endorses the role.

In HAVE, RoleClass allows to describe structural properties of a role. A RoleClass is a stereotyped interface (in the sense of UML), that defines the actions that an agent playing the role should perform without defining how the actions have to be executed. In addition, resources may be assigned to organisations or more specifically to roles. In the organisational model, resources are defined by their entity class. Therefore, any instance of this class can be used by the agent as a resource. It ensures the consistency between the model of the activities and that of the environment. Organisational structure and RoleClass specify the properties of an organisation at an abstract level, without neither referring any concrete organisation nor specific instances of roles and resources. This knowledge can be used by the agents.

Following the same principle as for Collaboration and Collaboration Use in UML, an organisational entity is an instance of an organisational structure in HAVE (Figure 6). An organisational entity is defined as an assignment of roles to agents and of entities to resources, according to the organisational structure. Many organisational entities can be created, corresponding to the same organisational structure. Assignments for roles and resources can be set a priori or dynamically during the execution.

In MASCARET, the collaborative activity of the participants within an organisation can be specified as predefined procedures. These procedures are described in the context of an organisational structure and are executed by an organisational entity. Procedures are modelled using UML activity diagrams but with a more restrictive semantics that ensures the consistency with the modelling of the organisations (Figure 7). In HAVE, swimlanes are interpreted as referencing roles from organisations. In the same way, object nodes refer to resources from organisations. Actions can be of the following types. 1) CallAction: execution of an action defined in the RoleClass as described in the organisational model (it may be a sub-activity). 2) SendSignal: the result of the action is that a signal is sent to a resource. If the entity class corresponding to the resource holds state machines sensitive to this signal type, then the targeted entity may react to the agent’s action. 3) SendMessage/ReceiveMessage: exchange of messages between agents playing roles in the organisation.

The activity model defines all the possible orderings for the execution of actions by an organisational entity. Operators existing in UML (e.g. fork, join, alternative ...) are rich enough to specify complex procedures. It is also possible to associate preconditions (interpreted as feasibility conditions) and post-conditions (rational effects) to both actions and activities, and thus procedures. This information is available to the agents that can use it for check-
For example, according to a procedure, an agent has to take an object which is inside the drawer of a specific desk (this can be inferred from constraints written with VRX-OCL, see Section 3.6). This agent can use the description of the sliding behavior of an entity Drawer as seen in Figure 5. Using its perception of the environment, it can observe that the drawer is in the Closed state. To execute the action Take on the object, the slider should be Open. As the state machine of a drawer is an explicit knowledge thanks to VEHA, the agent can infer that it has to send the PullEvent to the drawer until the condition isState(Stopped) is satisfied. It shows again that in MASCARET behavioral models are not only used to simulate entities (system-oriented semantic modelling) but also to provide knowledge to users and autonomous agents (content-oriented semantic modelling).

Agents communicate through messages. As for agent’s structure and behavior, our solution is based upon FIPA and more precisely upon ACL, its Agent Communication Language ²). According to the Speech Act theory, messages are used to satisfy the agent’s goal, thus communication acts are considered as actions, named performatives. FIPA-ACL provides 23 performative types. In behave, we focused on the basic performatives, Request and Inform. The Request performative is used by an agent to request for an information (e.g., the value of a property) or to ask another agent to perform an action. The Inform performative allows an agent to answer a request, for instance to return the value of the property or to inform on the result of the requested action. The body of the messages is written using a content language. At that time, the generic communication behavior provided by behave is able to parse the FIPA-SL content language.

3.6 Specifying semantic constraint using VRX-OCL

In the previous sections, we have seen how the UML metamodel has been specialized in MASCARET. Different UML—MASCARET diagrams have been used to model the VE. The class models represent the structure of the VE. An entity in the VE can be associated with behavioral models (i.e., state machines). The activities of human and artificial agents are defined using UML-like collaboration and activity diagrams.

However, there exists many semantics that UML diagrams are unable to convey by themselves. For example, considering the Desk model illustrated in Figure 4, a common understanding about the ontological model of the desk is that “the height of a table is greater than the length of all of its legs”. Similar semantic expressions can be found in the description of the structure of VEs, for instance the cardinality constraints (e.g., “a table should have at least three legs”). With regard to operational semantics, operations in VEs are often conditional, i.e., an operation can be executed according to a precondition and is considered as complete when the post-condition will become true. For example, the operation SlidingOut of a drawer is terminated when there is no motion on the drawer: the post-condition of the operation is that the drawer is in the Stopped state.

To overcome the limitations of graphical modelling, it is necessary to integrate more formal and logical information into the UML-based conceptual models of MASCARET. We propose a straightforward solution which consists in extending the UML Object Constraint Language (OCL) into VRX-OCL, which stands for Virtual Reality eXtension of OCL.

Using VRX-OCL to enrich semantics of the VE model: OCL is a textual formal language to add precise and unambiguous meanings into UML models [25]. In MASCARET, we use VRX-OCL to specify constraints on the elements of the model of the VE (i.e., the M1 level in Table 1). The evaluation of VRX-OCL constraints provides information about system state, but does not have any side effect.

The following example illustrates the expression “the height of a table is greater than the length of all of its legs”.

context Table inv:
Legs.allInstances() ->forall( 
    varLeg | varLeg.length < self.height )

An OCL expression must be given within a context and can be applied to any element of a UML model. In the example, the context is the Desk class and the semantic constraint is associated to the property “height” of this class. Other element types can be included in an OCL expression, such as method, state etc. In this case, a constraint aims at the specification of pre- or post-condition for an action, or a guard condition for a state transition.

This language supports the ability to express complex logical expressions thanks to logical operators such as not, and, or, and xor. Furthermore, logical expressions can be associated with quantifier operators such as “at least”, “all of”. In addition, OCL facilitates the modelling of semantic constraints over multi-classes. The following expression exemplifies the cardinality constraint in the expression “a table should have at least three legs”:

context Table inv:
self.legs ->size() >= 3

In this example, the navigation from the entity class Table to the entity class Leg is realized through the association legs, which is a compositional relationship between two Entity classes (a stereotype of association specific to VEHA).

Extending OCL to ensure spatial constraints: Although OCL is rich and expressive enough to deal with complex logical constraints, it does not cover the specificity of semantic modelling of VEs. For instance, spatial constraints are out of the scope of the object model of UML. However, it is mandatory to be able to introduce this kind of constraint in the modelling of VEs. For instance, taking the model of the SlidingIn behavior for a drawer presented in Figure 5, the transition from the Open state to the Closed state of a drawer is guarded by a spatial constraint between the drawer and the table, “the drawer should be inside the table”.

VRX-OCL offers VE designers the ability to define various types of spatial relationships, involving nonmetric constraints (i.e., topological, projective) and metric constraints (i.e., distance and directional). Especially, spatial knowledge can be directly interpreted as instructions displayed textually or graphically, for example by visualizing the acceptance areas for the location of manipulated objects in the 3D environment. For more details about VRX-OCL, we refer the readers to [22].

4 Applications

Several applications have been developed using MASCARET and all of them followed the same development process presented in 3.2. In this section, we present only three of them. First, a Virtual Environment for Learning; second, an application for the simulation of human collaborative activities; and third, an application for Interactive Virtual Tours.

4.1 Virtual Environment for Learning: rich-content environment

The primary goal of MASCARET was the modelling of Virtual reality Environments for Human Learning (VEHL). We consider VEHL as one of the most challenging type of application and a relevant benchmark for experimenting, evaluating, and validating VR application development approaches. On the one hand, the involvement of learners in VEHL is an important factor that differentiates this type of environment from others. It should provide an interactive representation of learning content to the learners. From a modelling
perspective, the challenges are related to the way to describe the tasks that the learner has to perform. A VEHL is an ever-evolving system that should adapt to different educational contexts. On the other hand, to help the learners to progress in their learning activities, a VEHL should provide a rich-semantic representation of domain knowledge in order to give instructions or educational feedback to the users. Based on the semantic model of a VEHL, it is possible to incorporate specific modules acting as a decision-making support systems. An intelligent tutoring system is an example of such a module [5].

**Figure 8**: Specifying and visualizing of spatial constraints in the Physical Work Lab, a VEHL dedicated to physics: an example of a configuration of the environment to measure the speed of light in different media.

MASCARET has been used to develop the Physical Work Lab (cf. Figure 8), a VEHL dedicated to lab work in physics, particularly in optics. We used MASCARET to model the procedure that learners were supposed to use for each exercise. One of the exercises was that learners had to measure the speed of light in different media: air, resin and water. An example of an instruction given to the learner, corresponding to an expected result of a step in the procedure the learner had to follow, was “To measure the speed of light in resin materials, put the cube of resin on the bench, between the lens and the mirror” (see Figure 8).

In the Physical Work Lab, to express such an instruction, it is necessary to specify the spatial constraints between the learning resources within the environment (i.e., the cube of resin, the bench, the lens, and the mirror). With MASCARET, such spatial constraints are expressed thanks to the spatial constraint language VRX-OCL, as presented in Section 3.6. Thereafter, based on the constraints satisfaction, it is possible to visualize abstract spatial relations and thus to help the learner in manipulating spatial objects. Figure 8 illustrates the visualization of the sub-space “between the lens and the mirror” in the Physical Work Lab environment.

**4.2 Simulation of human activities**

MASCARET has been applied to the simulation of the human activities on an aircraft carrier (Gaspar [14]). These activities, called procedures, were written by experts using a customised UML modeler embedding MASCARET extensions. This model was pretty complex as it involves 100 classes for 1,000 entities representing 1 million facets. In this application, we simulated the activities of 6 kinds of team (Organisational structures), each of them having about 5 procedures to execute. Most complex procedures, like aircraft catapult, were composed of about 80 actions and transitions. Roles were assigned to autonomous agents having a procedural behavior. In classical scenarios, about 50 agents evolved on the aircraft carrier and execute procedures. Figure 9 illustrates the execution of an aircraft catapult procedure by autonomous agents.

Using MASCARET, procedures and actions are explicit during the simulation. This knowledge is used by the autonomous agents which have to execute the procedures. They can know what they have to do and what the other teammates have already done. Using knowledge on the pre/post condition of an action, they are able to determine if the action is feasible in the environment. Moreover, the procedure executions, as the action executions, are explicit knowledge. So, in the Gaspar application, we have developed a tool that logs every procedure and action execution with their context and presents all these information on graphs, useful for experts to analyze the performance of their scenario. Because this tools only relies on concepts belonging to the M2 layer, it is reusable for any MASCARET application.

**4.3 Cultural heritage: interaction with a virtual guide**

Brest Coz is an application to interactively visit the harbour of Brest in the 18\textsuperscript{th} century. Many activities relating to shipbuilding have been simulated. An autonomous virtual guide can give information to users about these activities. The virtual guide is able to answer questions about the structure and the properties of entities in the environment, namely the different parts of a boat. For example if the user asks “What is a couple?”, by introspecting the domain model thanks to VEHA, the guide would answer: “A couple is a wooden part of the boat fixed on the boat’s keel.”. Questions on the activities are also allowed as: “What is this man doing?”, the virtual guide, using HAVE, can answer: “This man is using a hand drill to place a wooden dowel in order to fix the couple.”. Figure 10 illustrates such a scene. In Brest Coz, it is possible to dialogue with virtual humans in pseudo-natural language, thanks to the NABUTALK API\textsuperscript{3}.

**Figure 9**: Simulation of aviation activities on aircraft carrier with MASCARET.

![Figure 9: Simulation of aviation activities on aircraft carrier with MASCARET.](image)

**Figure 10**: A virtual virtual guide in Brest Coz

**Figure 10**: A virtual virtual guide in Brest Coz

---

\textsuperscript{3}http://www.dialonics.com

**5 Conclusion**

In this article, we have presented the MASCARET framework, that aims at developing Intelligent Virtual Environments. First, MASC-
CARET provides a modelling language based on the UML meta-model. Second, it allows to give an operational semantics to the models, which can be executed on various VR platforms. Thus, this proposal fulfills the requirements of both content-oriented and system-oriented semantic modelling.

Thanks to its VEHA metamodel, MASCARET allows to give a formal description of the abstraction that are represented in the VE. It covers the ontological description of the domain, like in OWL [18], including the behavioral description, in such a manner similar to agents’ Artifacts proposed by [24]. Its behavioral model is similar to HPTS [2]. Therefore, concepts of VEHA are grounded in existing solutions but none of them cover all the domains of VEHA: ontological models fail to give operational semantics for behaviors and the complex ontological descriptions are not supported by most of the behavioral modelling languages. UML is a sound basis to unify all the domains related to the semantic modelling of VEs. The idea to encompass all the facets of semantic modelling with the same language has been put forward in VR-WISE [20], not by using UML but by an Entity-Relation inspired modelling language. The major contribution of VR-WISE concerns the modelling of mechanical relationships between entities. In MASCARET, it should be tackled with VRX-OCL.

Human activities in VEs are simulated as the result of the behavior of autonomous agents that collaborate with users. The model of these collaborative activities is supported by the metamodel HAVE. It is based on classical concepts of agent, role, activity and organization. Our solution merges concepts from UML and different multi-agent organisational models [15, 3]. In MASCARET, the artificial agents can be implemented by using the behave generic agent metamodels.

The major contribution of MASCARET relies on its unified approach: the agent definition is not separated from the modelling of the environment where agents evolve. Environment, agents, and activities are modelled by using the same language, that makes the development process more rational.

ACKNOWLEDGEMENTS
This work was supported in part by a grant from Région Brabant and in part by the ANR, Agence Nationale de la Recherche (Corvette project).

REFERENCES