7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France

« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

VISUALITION ET SIMULATION : APPROCHE ONTOLOGIQUE

M. AUGERAUD, D. SARRAMIA

Université de La Rochelle - Laboratoire L3i

Pôle Sciences et Technologie

17042 LA ROCHELLE Cedex 1

Université de La Rochelle

{maugerau, dsarrami}@univ-lr.fr

RESUME : *L'immersion de l'utilisateur dans un environnement virtuel permet un retour sur des critères qui sont difficilement évaluables car difficiles à caractériser. Nous nous plaçons dans le cadre d'une utilisation par un décideur. L'approche naturelle utilisée pour modéliser puis simuler un système complexe est de développer un modèle, souvent appelé modèle conceptuel. Ce modèle utilise un formalisme et des concepts liés au domaine qui permettent au modélisateur de « comprendre » le fonctionnement du système. Ce modèle doit ensuite être traduit dans un formalisme qui rendra possible l'exécution du modèle obtenu et donc l'évaluation du « système ». Les informations nécessaires à la simulation de systèmes complexes ne sont pas directement liées aux informations qui seront visualisées au moyen de l'environnement virtuel de visualisation. L'environnement virtuel de visualisation doit être pensé pour permettre à l'utilisateur d'agir sur les éléments pertinents et d'observer les éléments ou les valeurs nécessaires à son processus de prise de décision. La simulation mise en œuvre est orientée agents. Une perspective de ce travail est de permettre de travailler avec des modèles de granularités différentes. Le lien sémantique entre les différents modèles utilisés devrait être facilité par les dépendances, associations et classifications des différents éléments identifiés à l'aide d'ontologies.*

MOTS-CLES : *ontologie, simulation, visualisation, méthodologie, aide à la décision*

1. INTRODUCTION

Nous avons proposé (Augeraud *et al.*, 2006) une démarche pour la production d'un environnement logiciel qui s'intéresse à la simulation interactive. L'immersion de l'utilisateur permet un retour sur des critères qui sont difficilement évaluables car difficiles à caractériser.

L'approche naturelle utilisée pour modéliser puis simuler un système complexe est de développer un modèle, souvent appelé modèle conceptuel. Ce modèle utilise un formalisme et des concepts familiers au modélisateur. Cette conception lui permettra de « comprendre » le fonctionnement du système. Ce modèle doit ensuite être traduit dans un formalisme qui rendra possible l'exécution du modèle obtenu et donc l'évaluation du « système ».

Les informations nécessaires à la simulation de systèmes complexes ne sont pas directement liées aux informations qui seront visualisées sur l'interface graphique. Nous nous plaçons dans le cadre d'une utilisation par un décideur. Ceci signifie que la visualisation doit être pensée pour permettre à l'utilisateur d'agir sur les éléments pertinents et d'observer des éléments ou des valeurs nécessaires à son processus de prise de décision. Nous proposons pour réaliser l'immersion et enrichir l'information produite d'utiliser un environnement virtuel de visualisation.

Concernant l'environnement, nous utilisons une séparation telle que celle proposée par Odell et al. (Odell *et al.*, 2002) qui séparent un environnement physique d'un environnement de communication. L'environnement physique fournit les lois, règles, contraintes et politiques qui gouvernent et régissent l'existence physique des agents. Nous avons défini un environnement physique. Cet environnement (Figure 1) est construit de façon modulaire, les modules étant des plaques contenant un graphe de circulation (voiture et piéton), des éléments de signalisation et des bâtiments. L'environnement de communication est fournit par la plateforme Jade (http://jade.tilab.com/). Comme nous le détaillerons plus tard, nous avons des éléments d'échanges aussi indépendant que possibles des implémentations respectives dans le but d'assurer une souplesse au couplage de ces deux environnements.

Ce papier n'a pas pour but de proposer une ontologie de simulation, ni une ontologie pour la visualisation. Il s'agit d'un travail d'identification de problèmes en vue d'initier une solution de nature ontologique sur le couplage entre la simulation et la visualisation dans le but de le rendre explicite.

Une perspective de ce travail est de permettre l'utilisation de modèles de granularités différentes. Ceci devrait être facilité par les dépendances, associations et classifications des différents éléments identifiés dans les ontologies.

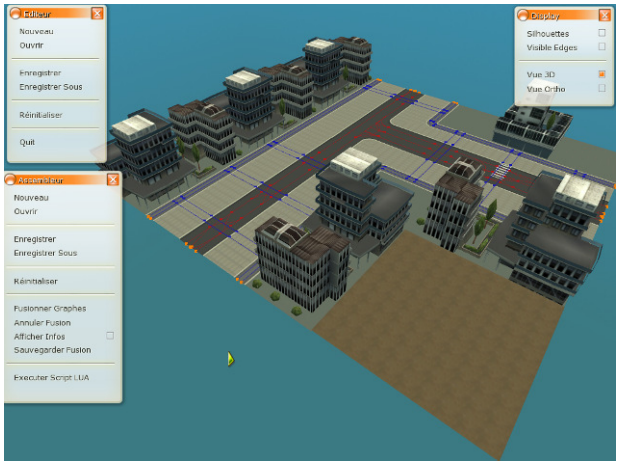

Figure 1. Environnement construit en collaboration avec Succubus Interactive

Dans la prochaine section, nous ferons un état de l'art des liens réalisés entre les domaines de la simulation, des ontologies et des systèmes multi-agents (SMA). Ceci nous permettra de montrer l'intérêt des ontologies dans ces domaines respectifs, mais également d'identifier des problèmes où elles pourraient être utiles. Dans la troisième section, le domaine de la visualisation est étudié d'un point de vue ontologique par rapport à quelques environnements existants. La quatrième section présente notre proposition méthodologique globale pour la modélisation et la simulation de systèmes complexes ayant pour objectif l'aide à la décision et l'interaction avec l'utilisateur. Les aspects conception et implémentation sont détaillés dans la cinquième section via la présentation de l'architecture logicielle. La sixième section présente notre proposition pour résoudre les problèmes de couplage simulation/visualisation d'un point de vue ontologique. Cette proposition est utilisée dans la septième section sur un exemple tiré d'un projet en cours de développement au laboratoire concernant l'aide à la décision pour la conception de plans de circulation urbains. La dernière section conclut ce papier et présente quelques perspectives de nos travaux.

2. SIMULATION, ONTOLOGIE, AGENT

La simulation est aujourd'hui très largement utilisée quel que soit le domaine d'application. Elle permet entre autre de réaliser des études sur des systèmes qui n'existent pas encore.

En général, les modèles de simulation sont classés selon la façon dont ils gèrent pour le système étudié le temps (statique/dynamique), l'état (discret/continu) et l'aléa (déterministe/stochastique). Pour la simulation en elle-même (qui est souvent appelé moteur de simulation), deux ou trois catégories sont communément distinguées : simulation à évènements discrets et simulation continue, voire hybride. Plusieurs classifications existent (par exemple (Sulistio *et al.*, 2004), (Page, 1994), (Zeigler, 1976)).

À la différence d'un vocabulaire, une ontologie cherche à représenter le sens des concepts et des relations qui les lient (Gruber, 1993). Il est considéré que tout l'intérêt d'une ontologie est d'aboutir à un accord sur une conceptualisation partagée d'un même domaine, même s'il ne s'agit que d'une conceptualisation partielle. L'exemple le plus connu d'ontologie est OWL (Ontology Web Language) qui est approuvée par le W3C. La conception et l'utilisation des ontologies sont désormais outillées. Protégé (http://protege.stanford.edu/) est l'outil le plus utilisé.

Des ontologies ont été assez récemment créées dans le domaine de la simulation. La plus connue est DeMO (Discret Event Model Ontology) (Miller *et al.*, 2004). Elle concerne la classe des modèles à évènements discrets et plus particulièrement les modèles de simulation orientés états, orientés activités, orientés événements et orientés processus. DeMO définit quatre concepts de bases (DeModel, ModelConcepts, ModelComponents et les ModelMechanism) qui permettent de construire un modèle à événements discrets. DeMO respecte la séparation entre le modèle de simulation et le moteur qui « exécutera » le modèle. Ainsi la structure de DeMO (voir Figure 2) est fondée sur la notion de composant de modèle et de mécanisme de modèle.

L'avantage d'une telle structuration est l'identification des mécanismes et des concepts transversaux aux différents modèles. Il permet également de valider les mécanismes utilisés dans chaque modèle lors de l'utilisation de l'ontologie DeMO.

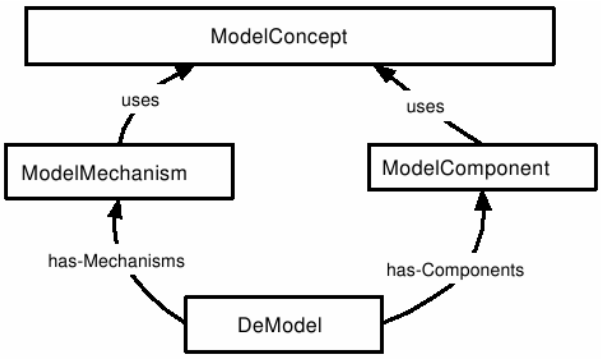

Figure 2. Représentation graphique des concepts de DeMO (Miller *et al.*, 2004)

Les méthodologies traitant de la conception SMA pour l'étude de systèmes complexes sont nombreuses. La classification et la comparaison de ces méthodes ont suscité la rédaction de nombreux articles, comme par exemple celui de (Bordini et al., 2006).

Des méthodes de haut niveau proposent un processus descendant de développement de SMA comme GAIA

(Wooldridge et al., 2000), MaSE (Multiagent Systems Engineering Methodology) (DeLoach, 1999), Prometheus (Padgham et al., 2002), Aalaadin (Ferber et al., 1998). Elles se distinguent d'une part par le nombre d'étapes que chacune requiert dans son processus de développement (de deux à sept) et d'autre part par l'objectif de conception qu'elles visent (plutôt centré sur le logiciel (Prometheus), centré sur l'analyse (GAIA, Aalaadin) ou mixte (MaSE)). Elles ont souvent donné lieu à la production de plateforme de développement (Aalaadin est supporté par Madkit par exemple).

Ainsi, les méthodologies de conception de SMA distinguent toutes l'analyse de la conception, mais ne se situent pas au même niveau. Elles ne considèrent pas toutes une spécification formelle (GAIA, MaSE). Pour les méthodes s'intéressant à l'implémentation, un objectif de simulation pour l'aide à la décision n'est pas pris en compte.

Le lien entre les ontologies et les SMA est actuellement étudié selon les domaines d'applications. On peut par exemple citer le projet ANR CO-SMA-GEMS (CO-SMA-GEMS, 2006) qui se centre sur les domaines de la Géographie, l'Economie, le Marketing et la Sociologie. Des travaux plus généraux travaillent sur l'écriture d'ontologie pour le domaine des SMA et la simulation (Christley *et al., 2004)*.

Les SMA sont désormais largement utilisés pour simuler les comportements dynamiques de systèmes complexes à partir de description de comportements individuels et des interactions entre les entités réalisant ces comportements. Néanmoins, étant donné que le SMA en lui-même est un modèle complexe, il se pose la question de devoir interpréter le fonctionnement du SMA avant de pouvoir interpréter les résultats qu'il a produit. Ceci est souvent dû à des différences d'implémentation pour un même modèle d'architecture d'agent et de SMA (Michel, 2004).

Dans le cadre de l'utilisation d'un SMA pour la simulation, le monde virtuel doit permettre aux agents de « percevoir » les éléments nécessaires à leur prise de décision. Les moyens disponibles dans le monde virtuel contraignent donc les comportements pouvant être observés et produits. Dans une situation idéale, il existe une ontologie de l'environnement virtuel disponible pour l'agent. Néanmoins, dans la plupart des implémentations, l'ontologie de l'environnement virtuel est confondue avec l'ontologie du SMA. Ceci signifie donc qu'il n'y a plus d'indépendance entre le SMA et son environnement virtuel, et que les comportements observables sont fortement contraints. Ceci est peu satisfaisant dans le cadre d'application interactive où les plateformes peuvent être variées allant de X3D à Unreal en passant par des moteurs tiers. Des travaux récents (Helleboogh *et al.*, 2006) ont identifiés cette problématique et l'ont traité en représentant de façon formelle la dynamique de l'environnement virtuel par rapport à sa relation aux agents. Le formalisme fournit une représentation explicite de la dynamique de l'environnement, spécifie comment la dynamique de l'environnement est liée aux agents, et précise comment la dynamique de l'environnement peut survenir, interférer et se terminer. L'environnement de simulation est représenté comme un système dynamique qui englobe et gère sa propre dynamique. Le modèle proposé par Helleboogh *et al* utilise un formalisme du premier ordre.

Dans (Gouaîch et Michel, 2005), les auteurs veulent pouvoir utiliser différents environnements simultanément. Pour cela, à partir des architectures existantes, ils caractérisent la relation environnement-agent à l'aide de cinq éléments : l'ontologie de l'environnement, les moyens de perception, les moyens d'action, les fonctions d'interaction (lien entre les moyens d'action et de perception de l'environnement), et une fonction de localisation pour les agents situés. Les moyens de perception, d'action et les fonctions d'interaction appartiennent à l'ontologie de l'environnement. Néanmoins, il reste à la charge de l'agent de traduire son ontologie vers l'ontologie de l'environnement ce qui demeure insatisfaisant.

La conception d'environnement 3D s'effectue souvent par l'écriture de programmes posant d'une part le problème de l'intégration de contenu dans des applications et d'autre part par l'inclusion de comportements dans des composants 3D (comme ceci est réalisé avec des moteurs de jeux tel qu'Unreal). Chacun de ces aspects sont traités soit par des programmeurs soit par des concepteurs qui utilisent des outils différents pour réaliser leurs tâches. Ceci rend non négligeables les problèmes d'incohérences entre éléments 3D et programmes. Pour résoudre le manque de concepts avancés et d'outils associés, Vitzthum (Vitzthum, 2006) propose une représentation abstraite des concepts nécessaires à la création de scènes. La solution décrite est un modèle de scènes qui combine le concept de composant 3D avec une pré-implémentation abstraite en utilisant un langage visuel SSIML/Components. Le concept composant aide à contrôler les scènes 3D complexes et à définir des relations concrètes d'intégration entre une application et le monde 3D. Le code de l'application peut être généré automatiquement à partir de la conception.

3. ONTOLOGIE ET VISUALISATION

Les ontologies pour les environnements virtuels visent à définir les concepts et les techniques liées à leur création et leur exploitation. Elles se composent des notions de scènes, elles-mêmes composées d'éléments fixes (le décor) et d'éléments mobiles. De manière générale, la structuration des éléments et de la scène est hiérarchique. Un nœud est un regroupement des nœuds ou feuilles situés en dessous**.** Les feuilles représentent des objets graphiques primitifs ou des composants de base de différents types, réalisant des entrées, des sorties, des commandes ou des commandes de mise en forme**.** Dans certains systèmes, on peut avoir deux graphes pour représenter la structure : la hiérarchie liée au graphe de scène et un graphe de visibilité. Cette représentation conduit à une description XML qui comporte : des balises de

nœuds représentant les nœuds de l'arbre de scène et leur hiérarchie, des balises d'attributs de propriétés matérielles ou physiques des objets de la scène, des références à des sous scènes se trouvant dans des ressources externes, des macro-commandes de définition de variables, de répétition et de test pour une description concise des scènes répétitives, des scripts qui décrivent des actions (modifications de variables) réalisées sur un ou plusieurs nœuds.

Dans (Gutiérrez, 2005) l'auteur propose de donner une description sémantique des composants d'un environnement virtuel afin qu'ils soient plus flexibles et plus adaptatifs. Les ontologies utilisées pour les environnements virtuels sont spécialisées par type d'éléments comme par exemple l'ontologie pour les « humains virtuels » proposée par (Gutierrez, 2005). Notre travail n'est pas de développer une ontologie pour les environnements virtuels. Cette tâche concerne essentiellement la communauté proche de la synthèse d'image. Nous utiliserons les résultats proposés dans (Gutierrez, 2005). L'auteur propose une organisation des concepts autour de trois aspects qui correspondent à des axes de recherche : modélisation et analyse du corps humain, animation d'humains virtuels, interaction d'humains virtuels avec des objets virtuels.

Une vision sémantique des environnements virtuels permet de définir une ontologie liée à chaque mode de représentation. Cette technique permet de donner une description XML du monde virtuel. Elle se retrouve dans la norme MPEG7 pour des considérations essentiellement liées à la transmission de flux et au rendu des entités qu'ils portent.

Quelle que soit la façon dont sont implémentées dans un système les scènes composants un environnement virtuel, les entités sont distinguées des composants nécessaires à la perception et à la transmission des commandes et événements provenant de l'environnement d'une entité. Le modèle proposé généralise celui proposé par (Gutierrez, 2005). Il est présenté sous forme du diagramme de classes UML de la Figure 3. Ce modèle « sémantique » ne précise pas l'implémentation qui est faite des opérations associées à une « Entité graphique ».

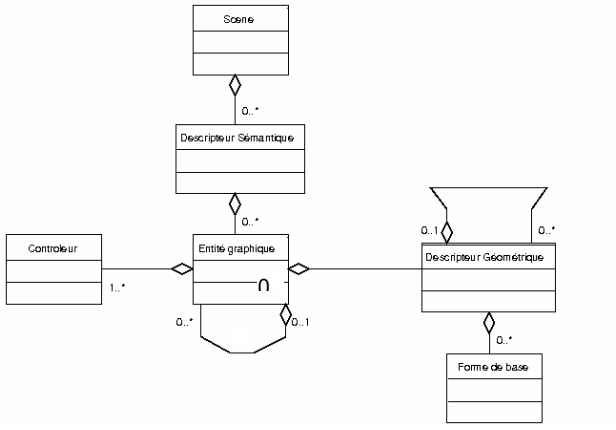

Figure 3. Diagramme UML d'un modèle sémantique générique d'un environnement virtuel

Pour visualiser l'interaction, nous utiliserons un moteur de jeux. Le monde virtuel animé par ce moteur de jeu est défini par un ensemble d'entités (scènes, véhicules, piétons, …). A chaque entité est associé un ensemble de fonctionnalités qui seront implémentées par des nœuds et des routes en X3D, des beans java en Scenebean, des scripts Lua dans une implémentation sur la plateforme de visualisation OSPDU conçue au laboratoire L3I ou des scripts (« UnrealScript gameplay scripting language ») pour une implémentation sur la plateforme Unreal.

Compte tenu de la complexité de mise en œuvre de toutes ces modélisations, simulations et techniques, la mise en place d'une méthodologie est nécessaire. Elle doit permettre, pour faciliter le processus de conception, de proposer un cadre ontologique au couplage des différents composants pour rendre explicite ce couplage.

4. CONCEPTION DE SYSTEMES DE SIMULATION PERMETTANT L'IMMERSION

La méthodologie de modélisation que nous proposons s'intéresse à la production d'un environnement de simulation pour l'aide à la décision avec pour cible un environnement de simulation interactive multi-agent. Elle intègre une modélisation orientée objets, une spécification orientée interaction pour formaliser le comportement du SMA, une conception/réalisation d'un SMA simulable dédié aux objectifs, avec prise en compte des interactions avec l'utilisateur. Ceci se place dans le cadre de l'évaluation des performances par la simulation interactive.

La modélisation du domaine étudié est réalisée en utilisant une décomposition en trois sous-systèmes (sous-système logique, physique et décisionnel) pour obtenir un *modèle générique de connaissance* (Gourgand, 1984). Cette analyse permet de modéliser le domaine étudié de façon statique. Chacun des trois sous-systèmes est modélisé par un diagramme de classes UML.

Un *modèle de communication* modélise la circulation des flux d'informations entre les trois sous-systèmes et se compose de deux modèles : un diagramme de classes UML, et un diagramme de flux.

La démarche de spécification de simulation interactive conduit à une recherche des interactions et des conditions (contexte) dans lesquelles ces interactions sont activées. Nous avons proposé à cet effet un patron utilisant les unités conceptuelles *Monde, Localité, Interaction Type* et *Agent* (Augeraud *et al.*, 2006).

La classe *Monde* met en relation trois unités conceptuelles qui sont Agent, Interaction Type, Localité. Il peut être spatial ou temporel. Un état du système est une configuration dans ce *Monde*. Le postulat est qu'*un agent donné appartient à une seule localité et participe simultanément à toutes les interactions liées à la localité*. Le concept d'interaction permet de structurer de manière cohérente l'aspect dynamique du système. Une

interaction décrit un changement d'état. Une *Interaction Type* peut se caractériser d'une part, par un ensemble de scénarios, et d'autre part, par un ensemble d'agents participants à cette interaction. Chacun des agents pouvant participer à une interaction joue un rôle dans cette interaction ce rôle caractérise les actions effectuée dans cette interaction par l'entité que l'agent représente. Ces rôles sont relatifs à la position de l'entité dans l'organisation représentée. Un agent joue un rôle différent selon l'interaction à laquelle il participe. Ce dernier aspect justifie la modélisation centrée interaction qui fait de l'interaction une entité de première classe du modèle. Un scénario est un ensemble structuré d'événements (temporels et/ou spatiaux) et d'actions. « Ce que fait » l'*Interaction Type* se décrit par un ensemble de règles exprimant les rôles possibles des agents au cours de l'interaction appelé *Schéma d'interaction*.

A chaque *Localité* est associé un ensemble d'interactions type. Ces localités peuvent être absolues ou relatives à des agents.

Dans notre travail, l'état du système représente l'activité d'agents engagés dans des interactions.

Ce que nous recherchons, compte tenu de l'objectif de simulation interactive, est une structure qui rende compte de l'interdépendance entre agents, interactions et localités. L'*espace d'interaction* que nous proposons comporte trois axes qui sont Agent, Interaction, Localité. Un état du système est une configuration dans cet espace. Chaque configuration se représente par un arbre.

Un état du système dans lequel un agent A est présent dans la sous-localité L_I d'une localité L_J pour une interaction I_K, où il joue le rôle R_M, se notera :

$$L_J[L_I[\ I_K[A[R_M]\ ]\ ]\ ]$$

L'état du système complet résulte de la composition de l'ensemble des rôles joués par les agents pour les interactions auxquelles ils peuvent participer.

5. ARCHITECTURE DE L'ENVIRONNEMENT LOGICIEL

L'architecture du système informatique (voir Figure 4) est réalisée selon une approche modulaire. Cette architecture sépare strictement de façon classique le noyau de simulation de la représentation graphique du système et de ses entités. Notons que cette architecture reste valable si une simulation autre que la simulation orientée agents est utilisée.

L'outil possède ainsi les modules principaux suivants :

– le *monde virtuel* dans lequel évoluent les représentations graphiques (noté environnement virtuel précédemment),

– un *éditeur de réseaux* pour éditer les réseaux représentant les trajets utilisables par les entités pour accéder aux différentes ressources,

– un *éditeur de scénarios* qui permet d'écrire des scénarios à étudier sur le système,

– un *système multi-agent* (SMA) qui contient toute la partie décisionnelle des agents identifiés durant l'analyse. Ce SMA est en fait décomposé en plusieurs sous-systèmes provenant de l'analyse,

– un *éditeur/contrôleur* servant d'interface entre le monde virtuel et le SMA. Les événements et les données transitent par ce module.

L'utilisateur ne peut interagir avec l'environnement que *via* certains modules : *le monde virtuel, l'éditeur de réseaux* et *l'éditeur de scénarios*. Le *monde virtuel* est la représentation informatique du système étudié (et présente à l'utilisateur le sous-système physique qu'il souhaite étudier). *L'éditeur de réseaux* permet de construire les réseaux supportant les différents types de flux. *L'éditeur de scénarios* permet à l'utilisateur d'écrire de nouveaux scénarios, et par exemple de tester des situations non prévues et les capacités du système à réagir.

Le modèle de conception que nous avons proposés (Augeraud *et al.*, 2006) (Figure 5) est un modèle générique, c'est-à-dire utilisable pour tout domaine étudié.

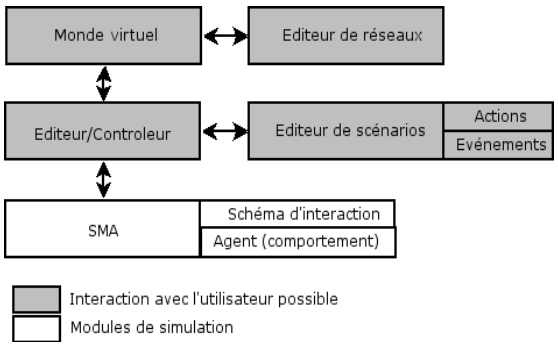

Figure 4. Architecture de l'outil d'aide à la décision [1]]

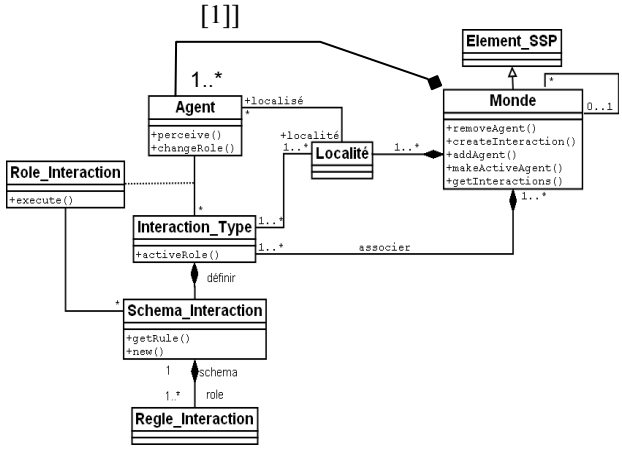

Figure 5. Patron d'interaction pour l'implémentation

Ce modèle permet d'associer un rôle à un agent lorsqu'il entre en interaction avec un autre agent, et donc de rendre contextuel à un espace d'interaction le code associé au rôle. Le modèle permet de faire évoluer le système dont un état se présente sous forme naturelle comme nous l'avons indiqué en (§4).

Le modèle intègre les fonctions suivantes : addAgent, makeActiveAgent, percevoir, changeRole et getInterac-

tion. L'instanciation du modèle conduit à créer des *mondes* vides (interactions type localisées) et à ajouter (addAgent) l'ensemble des agents participants potentiellement à cette interaction.

Un agent du SMA de l'architecture exécute une boucle de proaction *perception, décision, action.* La structure générale de l'agent est définie de la façon suivante :

```
// Perception
  _liste_percu = percevoir();
  pour tout agt in _liste_percu do activerInteraction(agt);
  pour toute I in L.Interaction() do I.getInfluence();

// Décision
  _Actions = decider();

// Action
  pour toute a in _Actions faire a.performAction();
```

6. COUPLAGE SIMULATION/VISUALISATION

Le problème qui se pose dès lors est de concevoir des modèles pour l'agent et le monde indépendants de toute implémentation du monde virtuel (i.e. du moteur de visualisation). Ceci n'est pas nouveau puisque dans le domaine du génie logiciel l'architecture MVC (Modèle Vue Contrôleur) (Reenskaug, 1979) et ses dérivées concerne un problème similaire visant à découpler tout en synchronisant leurs états, vue et modèle.

Le problème qui se pose ici ne se situe pas au niveau des composants logiciels ou au niveau du placement de l'information. Il concerne la coopération de systèmes et la communication entre ces systèmes. Cette coopération comporte deux aspects : d'une part établir un lien entre les entités de ces systèmes et d'autre part expliciter la logique de couplage. Ainsi des canaux de communication sont définis et portent une sémantique. Ils sont interconnectés selon une certaine logique pour qu'un message provenant d'un des systèmes entrant sur un canal produise un ou des messages sortant vers l'autre système.

Nous avons vu dans le §2 que certains auteurs se sont intéressés à cette problématique, parfois d'un point de vue ontologique, mais sans apporter de solution quant à l'explicitation du couplage environnement/simulation.

Dans le cadre du projet OSPDU (organisation du stationnement dans le plan de déplacement urbain), un exemple d'une implantation avec le langage oRis (Harrouet, 2000) permet de visualiser les problèmes à résoudre. La perception consiste d'une part, à demander au monde virtuel de fournir la liste des entités perçues (viewFirst(angle, dist, "nom entité", _pi_/2, 5, 0);) d'autre part, à solliciter pour chaque entité la création d'une interaction. Ceci s'exprimer de la façon suivante :

```
void Vehicule::perception(){
    float angle;
    float dist;
    _LaVoie = (Voie) viewFirst(angle,
                               dist,"Voie",_pi_/2,5,0);
    if (  (_LaVoie != NONE) &&
          (_LaVoie != _LaVoiePrecedente)) {
          _LaVoiePrecedente = _LaVoie;
          println(this," est sur la voie ",_LaVoie);
    }
    if (_LaVoie != NONE){ _LaVoie->interact(this);}
}
```

Cette perception récupère directement les informations fournies par le monde virtuel. A aucun moment, la sémantique de l'information n'est formalisée et utilisée de façon explicite pour identifier le(s) canal (canaux) à utiliser sur un modèle de système (entité ou système complet) pour le(s) connecter au(x) canal (canaux) du monde virtuel. Ceci est réalisé de façon ad hoc selon le monde virtuel et selon le domaine étudié. Ceci serait dû selon (Oliveira *et al.*, 2003) au développement d'un monde virtuel dédié à chaque nouvelle application.

Notre approche consiste à construire l'ontologie à partir du modèle de l'agent qui communique avec le monde virtuel au travers de capteur et d'effecteur. Le modèle est représenté de manière schématique par la Figure 6.

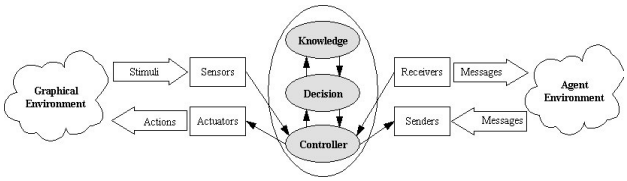

Figure 6. Modèle d'un agent (Richard, 2001)

Il est intéressant de noter que ces notions de capteurs et d'effecteurs sont décrites dans la Game Ontology existante pour le domaine des jeux (Zagal *et al.*, 2005) qui prend en compte l'interaction avec l'utilisateur sous le nom interface et entity manipulation. Ainsi, il est possible de considérer que les agents ne sont ni plus ni moins que des joueurs du monde virtuel avec leurs propres buts. Pour les agents, ce sont les comportements qui font le lien entre l'information fournit par les capteurs et l'action demandée aux effecteurs.

Les réactions produites suite à un stimulus se traduisent par des actions sur une ou plusieurs entités du monde virtuel.

Dans le cadre de la simulation orientée agents (SOA), les agents sont représentés dans le monde virtuel par des avatars. Ces avatars possèdent également des capacités dans le monde virtuel. Nous considérons donc qu'ils peuvent posséder des capteurs et des effecteurs. Nous ne considérons pas de fonction reliant les capteurs aux effecteurs car le monde virtuel ne prend pas de décision. Néanmoins, l'agent peut demander au monde virtuel, selon son niveau de prise de décision dans le système à modéliser, des informations sur toute ou partie des avatars. Cela signifie que le monde virtuel possède des fonctions d'interaction.

S'il existe des avatars dans le monde virtuel qui ne servent qu'à générer de l'interaction, et donc faire réagir nos agents, ils devront être décrit selon l'ontologie des agents.

Il est intéressant de noter que ceci est également valide lors de l'utilisation de toute autre technique de simulation, les entités étant plongées dans l'environnement de simulation (graphique ou non).

Notre ontologie est donc composée, en ce qui concerne le monde virtuel, des éléments suivants :

- *Capteur* : capturer de l'information pour des entités (avatar, processus…).
- *Effecteur* : modifier l'environnement (SMA, environnement graphique,…) pour une entité
- *Fonction d'interaction* : proposée par l'environnement et pour que des entités acquièrent de l'information sur d'autres.

Quelle que soit l'ontologie de l'environnement et du modèle de simulation utilisé, il est nécessaire d'identifier pour chacun des éléments des deux environnements les éléments de l'ontologie. Pour la mise en oeuvre, nous proposons d'utiliser des diagrammes de déploiement et de composants UML. La notion d'interface (requise ou proposée) sur des ports pour des composants et des noeuds va permettre de représenter les capteurs, les effecteurs et les fonctions d'interaction.

En ce qui concerne la simulation, l'ontologie proposée se traduit par des messages à destination d'entités de simulation portant des commandes ou des informations, des messages de mise à jour destinés à l'environnement virtuel, des messages de demande à l'environnement virtuel d'information sur la perception d'interactions et des réponses à ces demandes.

La connexion des interfaces permet de rendre explicite les connexions entre le monde virtuel et le simulateur puisqu'il devient nécessaire de valider la connexion, et donc d'expliciter la nature et les valeurs prises par les informations transportées.

Vouloir réaliser ces connexions complexes dans le code des entités du graphe de scène rend complexe la programmation et rend difficile la maintenance et l'évolution.

Nous proposons donc que la collaboration entre un monde virtuel et un simulateur s'effectue sur le modèle perception-action au travers d'un middleware qui aura en charge d'une part la communication entre les systèmes, d'autre part d'implémenter la logique de réactions aux entrées. Ce modèle est présenté par la Figure 7.

Ce système intermédiaire comporte donc des composants dont le rôle est de réagir continûment aux stimuli du système de simulation et d'autres dont le rôle est de capturer les informations (influences) résultant des interactions entre entités du monde virtuel.

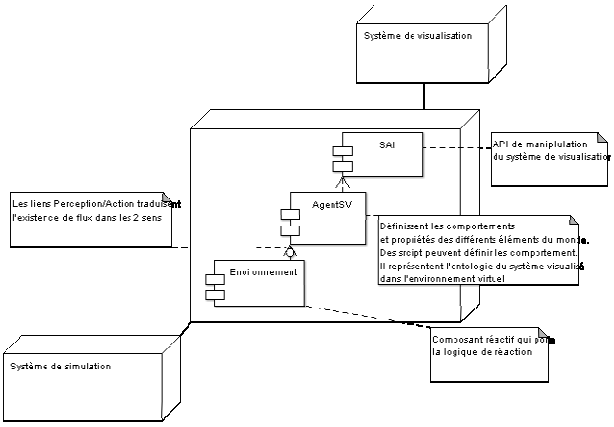

Figure 7. Diagramme de déploiement de la solution

7. EXEMPLE DU PROJET OSPDU

Dans le cadre du OSPDU, la simulation est orientée agents. La plateforme SMA utilisée est Jade.

Nos agents (véhicule, feu) sont conçus de façon à reproduire un comportement le plus naturel possible. Les résultats produits par le SMA ne sont pas encore validés. La particularité de Jade est de travailler sur la notion de comportement et d'inclure nativement le support des protocoles de la FIPA (http://www.fipa.org/).

Le monde virtuel est développé par un outil d'édition conçu par le laboratoire L3I et réalisé par l'entreprise Succubus Interactive. Le langage de communication entre le SMA et le monde virtuel est Lua, et utilise les ports TCP/IP.

Le monde virtuel possède les fonctionnalités suivantes :

- lister_agents()
- recuperer_graph_pieton()/…_graph_vehicule()
- creer_trajet_vehicule ()
- creer_vehicule()
- detruire_agent()

Les véhicules et les feux possèdent des avatars dans ce monde, et ont respectivement les fonctionnalités suivantes :

- change_trajet_vehicule(), veh_avance()
- change_couleur()

Tous les avatars ont en commun les fonctionnalités :

- position_agent()
- demander_agents_visibles()

Les agents du SMA de simulation ne possèdent aucune de ces fonctionnalités. Ils possèdent des comportements qui modélisent la réponse à des messages qu'ils s'échangent et qu'ils échangent avec leur avatar virtuel. Par exemple, nos agents souhaite aller d'un point à un autre dans la ville, peu importe comment (en marchant ou en roulant). Néanmoins, ce déplacement est contraint d'une part par l'état du système perçu et d'autre part par l'état de l'agent. Nous ne voulons pas que le comportement de l'agent soit contraint par le type de monde vir-

tuel mais l'agent doit pouvoir se déplacer en tenant compte de ce qui se trouve dans son périmètre dans l'environnement virtuel. La Figure 8 présente du point de vue des environnements (mondes) quelques connexions et ainsi permet de visualiser qu'une notion dans le SMA est traduite par l'utilisation de plusieurs fonctionnalités du monde virtuel.

La Figure 9 illustre au niveau des agents le mécanisme mis en place dans la Figure 8. Elle présente l'exemple des agents véhicule et piéton possédant des avatars propres. Ces agents ont en commun la perception et le déplacement mais ce dernier est réalisé de façon différentes puisque un piéton et un véhicule (un conducteur) ne se déplace pas de la même façon (l'un roule, l'autre marche).

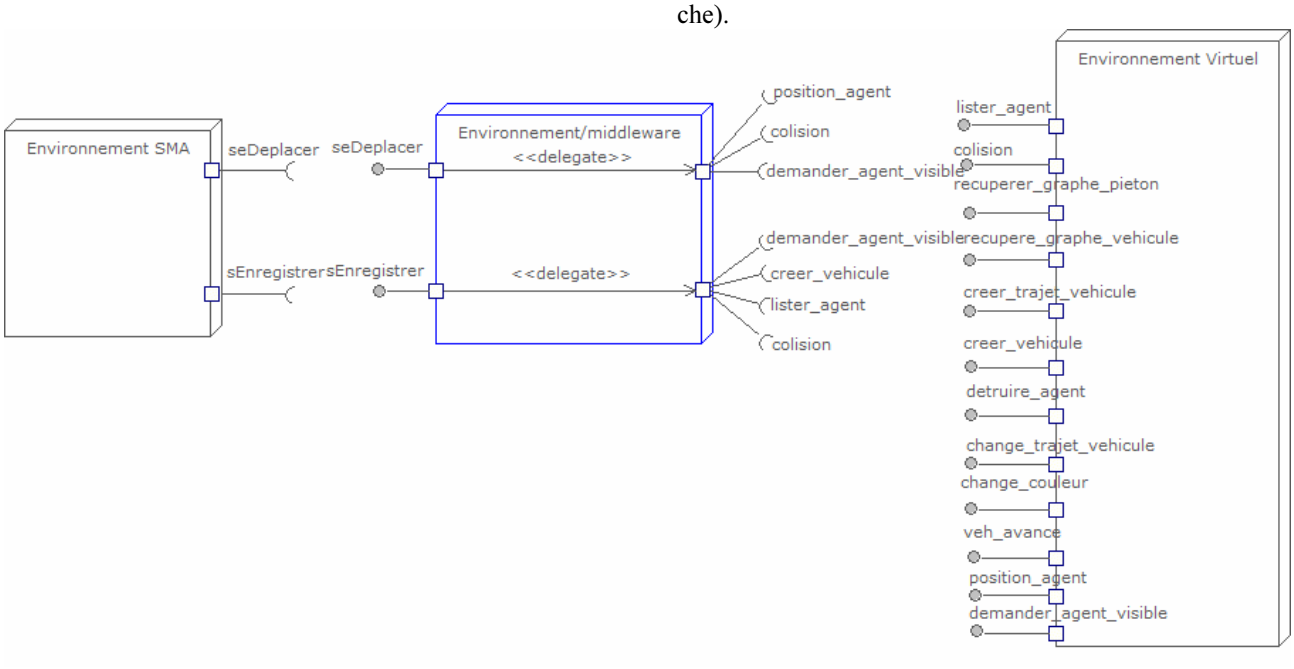

Figure 8. Déploiement pour OSPDU

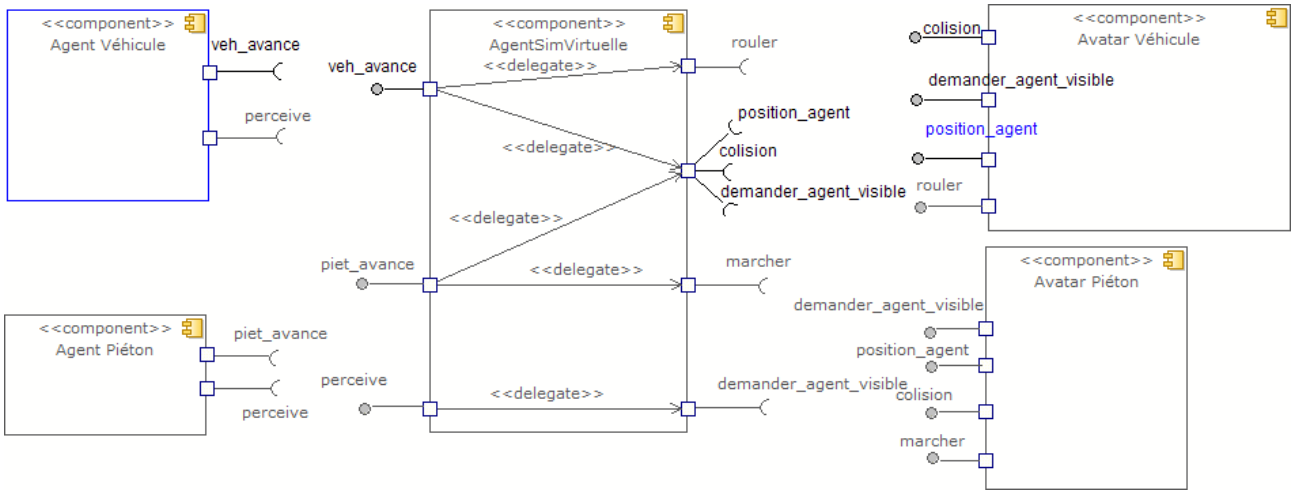

Figure 9. Couplage des composants pour OSPDU

Une fois documenté et contraint, ces modélisations permettent d'obtenir d'une part une représentation explicite du couplage entre le simulateur et le monde virtuel, d'autre part de valider l'indépendance du SMA par rapport aux mondes virtuels.

8. CONCLUSION

Dans ce papier, nous avons mis en valeur le manque de dépendance entre les environnements de simulation (SMA ou non) et les environnement de visualisation. L'un de nos objectifs étant la prise en compte de l'interaction avec l'utilisateur et la visualisation dans différents environnements, il est nécessaire de rendre le couplage entre environnement de simulation et environnement de visualisation explicite. Une architecture de l'environnement logiciel est proposée et s'articule autour de la proposition faite de pouvoir immerger l'utilisateur dans un monde graphique virtuel en 3D. Pour répondre aux problèmes liés au couplage, nous proposons, lors de la conception, de prendre en

compte sous la forme d'une ontologie simple, les connexions entre ces deux types d'environnement. Cela induit la construction d'un middleware dédié à contenir la connaissance de l'environnement de simulation et la connaissance de l'environnement de visualisation. L'utilisation de diagrammes UML adaptés (déploiement et composants) permet de faciliter l'expression sémantique des couplages réalisés et ainsi de posséder une représentation concrète des couplages. Une rapide présentation de l'utilisation de ces éléments a été réalisée sur le projet OSPDU pour lequel un environnement de visualisation existe. Des tests sont en cours pour valider notre approche par l'utilisation du moteur Unreal. Nous avons testé les connexions à ce moteur dans d'autres applications (visite virtuelle de laboratoire par exemple).

REFERENCES

Augeraud M., Boussier J., Collé F. et Sarramia D. 2006. Aide à la décision pour la conception de systèmes complexes : une approche multi agents. MOSIM'06 6ème Conférence Francophone de Modélisation et Simulation, p. 785-794, Rabat, Maroc.

Bordini R., Braubach L., Dastani M., Seghrouchni A.E.F., Gomez-Sanz J., Leite J., O'Hare G., Pokahr A., Ricci A., "A Survey of Programming Languages and Platforms for Multi-Agent Systems", *Proceedings of Informatica*, vol. 30, n° 1, 2006, p. 33-44.

Christley, S., Xiang, X. Madey, G. 2004. An ontology for agent-based modelling and simulation. *Agent 2004 Workshop*, Chicago IL.

CO-SMA-GEMS. Corpus d'Ontologies pour les Systèmes Multi-Agents en Géographie, Economie, Marketing et Sociologie, Projet ANR 2006. http://www.gemas.fr/dphan/cosmagems/

DeLoach S.A., "A Methodology and Language for Designing Agent Systems", *Proceedings of Agent Oriented Information Systems,* 1999, p. 45-57.

Extensible 3D (X3D) ISO/IEC 19775:2004

Ferber J., Gutknecht O., Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems", *Proceedings of the Third International Conference on Multi-Agent Systems*, 1998, p. 128-135.

Gouaîch A., Michel F. 2005. Towards a Unified View of the Environment(s) within Multi-Agent Systems. *Informatica*, international journal, 29 (2005) p.423–432

Gourgand M. 1984. *Outils Logiciels pour l'évaluation des performances des systèmes informatiques*. Thèse d'état, Université de Clermont-Ferrand II.

Gutieerrez Alonso, Mario A. 2005. *Semantic Virtual Environments* Thèse n°3258 Ecole Polytechnique Fédérale de Lausanne.

Harrouet F., oRis : s'immerger par le langage pour le prototypage d'univers virtuels à base d'entités autonomes, Thèse de l'Université de Bretagne Occidentale, 2000.

Helleboogh A., Vizzari G., Uhrmacher A., Michel F. 2006. Multi-Agent Modeling and Simulation: Dynamism in the Environment. *JAAMAS, Journal of Autonomous Agents and Multi-Agent Systems*, 14(1) p.87-116, ISSN 1387-2532. Springer

Michel F. 2004. *Formalisme, outils et éléments méthodologiques pour la modélisation et la simulation multi-agents*. PhD thesis - LIRMM - UM2 Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier - Université Montpellier II

Miller, J.A., Baramidze, G.T., Sheth, A.P., Fishwick, P.A. 2004. Investigating ontologies for simulation modeling. *Proceedings of the 37th Annual Simulation Symposium* p. 55 – 63.

Odell, J., Parunak, H.V.D., Fleischer, M., Breuckner, S. 2002. Modeling Agents and their Environment. *Agent-Oriented Software Engineering III*, Lecture Notes in Computer Science, Vol. 2585. Springer-Verlag, Berlin Heidelberg New York.

Oliveira M. Crowcroft J. and Slater M. 2003. An innovative design approach to build virtual environment systems. *In EGVE '03: Proceedings of the workshop on Virtual environment*, p. 143-151. ACM Press.

Padgham L., Winikoff M., "Prometheus: A Methodology for Developing Intelligent Agents", *Proceedings of the Third International Workshop on AgentOriented Software Engineering*, at AAMAS, Bologna, Italy 2002.

Page, Jr, E. 1994. *Simulation modeling methodology: Principles and etiology of decision support*. Ph.D. thesis, Virginia Tech**.** http://www.thesimguy.com /ernie/papers/unref/dissert/dw.html. Accédé 10/2007

Pryce N. and Magee, J. "SceneBeans: A Component-Based Animation Framework for Java" http://citeseer.ist.psu.edu/499819.html

Reenskaug. T. 1979. THING-MODEL-VIEW-EDITOR: an Example from a planning system. Xerox PARC technical note.

Richard, N. 2001. *Description de comportements d'agents autonomes évoluant dans des mondes virtuels*. Thèse de doctorat. ENST, Paris, octobre 2001.

Sulistio A., Yeo, C.H. and Buyya, R. 2004. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. *SOFTWARE - PRACTICE AND EXPERIENCE* 34, p. 653-673. John Wiley & Sons, Ltd.

Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition,* 5(2), p 199-220, 1993

UnrealEngine http://udn.epicgames.com/Two//Unreal-ScriptReference.html

Vitzthum, A. SSIML/Components: a visual language for the abstract specification of 3D components. *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, *ACM*, 2006, p143-151

Wooldridge M., Jennings N., Kinny D., "The Gaia Methodology for Agent-Oriented Analysis and Design", *Autonomous Agents and Multi-Agent Systems*, vol. 3, n° 3, Kluwer Academic publishers, 2000, p. 285-312.

Zagal JP. Mateas M. Fernandez-Vara C. Hochhalter B. Lichti N. 2005. Towards an Ontological Language for Game Analysis. *DiGRA 2005 Conference : Changing Views – Worlds in play*, Vancouver (Canada)

Zeigler, B.P. 1976. *Theory of Modelling and Simulation.* Wiley Interscience, New York.