N° d'ordre :

THÈSE

présentée devant

l'Université de Bretagne Occidentale

pour obtenir

le grade de : Docteur de l'Université de Bretagne Occidentale
Mention Informatique

par

Ronan Querrec

Équipe d'accueil 2215 (UBO, ENIB)
Laboratoire d'Informatique Industrielle (*LI2/ENIB*)

Titre de la thèse :

Les Systèmes Multi-Agents pour les Environnements Virtuels de Formation.
Application à la sécurité civile.

Soutenue le 4 octobre 2002 devant la commission d'examen :

M. :	Marcel	Le Floch	*Président*
MM. :	Philippe	Fuchs	*Rapporteurs*
	Philippe	Mathieu	
MM. :	Bruno	Arnaldi	*Examinateurs*
	Djamil	Sarni	
	Jacques	Tisseau	
MM. :	Pierre	Chevaillier	*Invités*
	Jean-Pierre	Jessel	

Les Systèmes Multi-Agents pour les Environnements Virtuels de Formation. Application à la sécurité civile.

Mémoire de thèse

Ronan Querrec — EA2215 (UBO, ENIB)

— Octobre 2002 —

Équipe d'Accueil 2215

Ronan Querrec
e-mail : querrec@enib.fr
url : http://www.enib.fr/~querrec/
tel : +33 (0)2 98 05 66 26
fax : +33 (0)2 98 05 66 29

École Nationale d'Ingénieurs de Brest
Laboratoire d'Informatique Industrielle

Technopôle Brest-Iroise
Site de la Pointe du Diable
Parvis Blaise Pascal
29280 Plouzané, Finistère
Adresse postale : C.P. n° 30815, 29608 Brest Cedex, France

Cette thèse a été financée par la Communauté Urbaine de Brest

Remerciements

Je suis heureux d'avoir l'occasion de présenter ma reconnaissance et ma gratitude à tous ceux qui m'ont soutenu de près ou de loin pour mener à bien la réalisation de cette thèse.

A ce titre, je tiens à remercier Jacques Tisseau, professeur à l'École Nationale d'Ingénieurs de Brest et responsable du *"Laboratoire d'Informatique Industrielle"* ainsi que Pierre Chevaillier, pour la direction et l'encadrement de cette thèse.

Je tiens également à remercier Monsieur Philippe Fuchs, directeur de l'équipe *"Réalité Virtuelle et Réalité Augmentée"* de l'École des Mines de Paris et Monsieur Philippe Mathieu directeur de l'équipe *"Système Multi-Agents et Coopération"* de l'université de Lille pour avoir accepté de rapporter cette thèse. Je remercie également Messieurs Bruno Arnaldi, Marcel le Floch, Jean-Pierre Jessel et Djamil Sarni pour avoir accepté de participer à mon jury de thèse.

Ces travaux n'auraient pu aboutir sur une application, sans la collaboration du Service Départemental d'Incendie et de Secours du Finistère, je tiens ici à en remercier l'ensemble du personnel et tout particulièrement le colonel Hervé Mahoudo. Je remercie également Joëlle Calvar et Françoise Jagaille du service Enseignement Supérieur, Recherche et Formation de la Communauté Urbaine de Brest qui a financé cette thèse.

Mes remerciements vont également à Patrick Reignier et Pierre De Loor pour leurs contributions techniques et scientifiques ainsi que Marie-Jo Kervella et Isabelle Le Poec pour la fourniture d'articles et de références bibliographiques. Merci à Sophie, Valéry, Frédéric et Fabrice pour la relecture de ce document et merci à Elyes pour son soutien et ses conseils de jeune docteur. Merci à tous les autres membres du *Laboratoire d'Informatique Industrielle* pour l'ambiance qu'ils y font régner, et tout particulièrement à Ken et Alain pour avoir détendu nos fins de semaine.

Enfin, je tiens à remercier ma famille et mes amis qui m'ont apporté un soutien

indéfectible durant ces années de thèse. Merci à mes parents, Myriam, Gaby et Anne-Sophie pour m'avoir supporté et encouragé dans cette aventure.

Table des Matières

Liste des Figures

Chapitre 1

Introduction

Le 13 mai 2000, l'usine de fabrication de feux d'artifice située à proximité de la ville de *Enshede* aux Pays-Bas explose. La catastrophe tue 21 personnes (dont 4 sapeurs-pompiers) et en blesse 944. Plus de 1000 habitations sont endommagées et près de 350 sont entièrement détruites. Les premières conclusions montrent l'existence d'irrégularités dans la délivrance des permis mais également le manque de préparation des services de secours pour ce genre de risque [Van Staalduinen 00]. Malgré une formation spécifique à la gestion des opérations et au commandement, les officiers sapeurs-pompiers ne disposent pas de moyens efficaces pour se former en condition opérationnelle à la prise de décision en situation de crise.

Cette thèse concerne la réalisation d'environnements virtuels de formation. L'objectif est de permettre l'apprentissage en situation opérationnelle, ce que l'on appelle communément *"apprendre sur le tas"*. Il s'agit alors de simuler l'environnement professionnel de l'apprenant et de l'immerger par l'action de façon à ce qu'il *"apprenne en faisant"*. Le sujet qui nous inspire est la formation à la gestion opérationnelle et au commandement, c'est-à-dire une formation au travail collaboratif et procédural. Notre problématique s'oriente ainsi vers la formation à la prise de décision et non au geste technique. La simulation du geste est nécessaire pour le réalisme de la scène mais le processus pédagogique n'impose pas qu'il soit effectivement réalisé par l'apprenant.

Nos travaux portent sur la réalisation de logiciels pour recréer ces situations opérationnelles dont il est impossible de bénéficier dans la réalité, dans un cadre de formation, car trop dangereuses ou inaccessibles. Nous utilisons pour cela les techniques de la réalité virtuelle et les systèmes multi-agents. Nous proposons un modèle se fondant sur ces principes et permettant la simulation de l'environnement ainsi que l'immersion des intervenants tels que les apprenants et les formateurs. Notre travail est également d'étudier les caractéristiques des fonctionnalités pédagogiques (tels que les modèles de l'étudiant, de l'expert ...) afin de rendre compatible notre modèle avec l'intégration d'aides pédagogiques.

1.1 L'enseignement assisté par ordinateur

Les premières applications d'Enseignement Assisté par Ordinateur (EAO) datent des années 1970. Elles sont fondées sur les théories comportementales (*behaviorism*, [Skinner 74]). Dans ce type d'application, le logiciel pose une question à l'apprenant, qui choisit une des réponses qu'on lui propose. La réponse du logiciel est immédiate (correct ou non) et en fonction de la réponse de l'apprenant, il pose une autre question. Ces premières applications présentent des limitations liées d'une part aux théories pédagogiques utilisées et d'autre part aux technologies employées. En effet, dans de telles applications, c'est le logiciel qui guide la formation : l'apprenant est alors passif. Cela permet l'acquisition de connaissances conceptuelles mais ne favorise pas leur transposition dans d'autres contextes [Wenger 87]. De plus, les techniques utilisées ne permettent pas d'interactions naturelles entre le système, l'apprenant et le formateur et rendent difficile la mise à jour du système. Enfin, les concepteurs de tels outils ont uniquement appliqué les méthodes de formation du domaine considéré sans profiter des potentialités offertes par les nouvelles technologies [Kalawsky 96] augmentant ainsi le coût de la formation par rapport à son efficacité.

Les outils contemporains se fondent sur les théories constructivistes ([Piaget 76, Vygotsky 78, Bruner 90]). L'apprenant construit ses connaissances en agissant dans l'environnement. Il interagit avec les objets (observables de la formation), cela favorise les "savoir-faire" plutôt que les "savoir-savant". L'apprenant est intégré au processus et le contexte (émotionnel et social) joue un rôle important [Deschenes et al. 96]. Les EAO se transforment alors en EIAO (Environnements Interactifs d'Apprentissage par Ordinateur). Les EIAO fondés sur les théories constructivistes trouvent donc un support idéal dans la réalité virtuelle. En effet, cette dernière place l'utilisateur (ou les utilisateurs) dans un environnement simulé et lui donne les moyens physiques d'interagir (percevoir et agir) avec cet environnement [Burdea et al. 93, Fuchs 96].

Les EIAO fondés sur la réalité virtuelle peuvent s'appliquer à la formation scolaire ou universitaire, mais trouvent néanmoins un champ d'application privilégié dans la formation professionnelle. En effet, dans ce type d'application, "l'enjeu est un ensemble de situations auxquelles doit faire face l'apprenant" [Rogalsky 97]. Il ne s'agit pas ici d'acquérir des connaissances mais de s'entraîner à les utiliser. On se place alors dans le cadre de simulateurs à vocation comportementale dans lesquels l'apprenant joue un rôle dans le processus simulé. La représentation (réaliste) de l'environnement (technique et social) ainsi que les capacités d'action y sont importantes [Crampes et al. 99].

Se pose alors le problème de la validation de l'utilisation des EIAO. La question est de savoir ce que l'apprenant a compris. Pour cela, le système construit un modèle de l'apprenant en fonction des tâches qu'il exécute. L'évaluation s'effectue par la comparaison entre ce modèle et le celui de l'expert [Dillenbourg 93]. De plus, en formation professionnelle, le formateur est souvent un spécialiste du domaine, mais pas forcément un pédagogue [Samurcay et al. 98]. Il faut alors lui fournir une aide

et automatiser certaines fonctions pédagogiques. Il s'agit alors de l'Enseignement Intelligemment Assisté par Ordinateur (EIAO)[1]. Cette automatisation nécessite de disposer des modèles de l'étudiant, de l'expert ainsi que des modèles pédagogiques qui représentent les modes d'interventions pédagogiques (apports de ressources, critiques, explications ...). La construction de ces modèles se fonde sur les études de la cognition humaine et de l'intelligence artificielle pour la représentation des connaissances et des raisonnements. Le modèle de l'étudiant sert de base aux fonctionnalités pédagogiques. Les composantes les plus couramment acceptées de ce modèle sont celles définies par [Self 88] : les connaissances conceptuelles et procédurales, les traits caractéristiques de l'étudiant et l'historique de ses sessions.

1.1.1 La réalité virtuelle pour l'entraînement

L'utilisation de la réalité virtuelle pour la formation s'inscrit dans cette démarche constructiviste de conception d'EIAO. Les travaux dans le domaine s'articulent autour des VET[2] (environnement virtuel pour l'entraînement), dont le plus connu est STEVE [Rickel et al. 99] et porte sur la modélisation de l'environnement, le comportement des acteurs virtuels, l'immersion de l'apprenant et la réalisation d'aides pédagogiques.

La réalisation d'environnements virtuels a surtout été motivée par la création artistique[3], par les jeux et par la simulation [Fuchs et al. 01]. Si classiquement, la simulation des environnements virtuels est abordée de manière globale, on voit maintenant l'intérêt d'une simulation basée sur l'évolution d'entités autonomes. Tisseau justifie cette autonomisation des entités [Tisseau 01] par :

- ▷ leur essence même lorsqu'elles représentent des organismes vivants et *a fortiori* des humains,
- ▷ la nécessité de prendre en compte instantanément les modifications dans l'environnement (important lorsque l'humain participe à la simulation),
- ▷ et l'ignorance sur la modélisation du comportement global d'un système complexe composé d'entités hétérogènes.

Doter l'entité d'autonomie, c'est lui donner des capacités de perception, de décision et d'action [Wooldridge 99]. L'enjeu est alors la conception d'architectures d'agents modélisant des mécanismes [Chevaillier et al. 00], des organismes vivants [Drogoul 93] ou des humains [Magnenat Thalmann et al. 91]. Les auteurs de telles architectures s'inspirent de l'intelligence artificielle, des systèmes multi-agents et de la robotique. Les architectures fondées sur des systèmes à transitions hiérarchiques et parallèles comme

[1] Pour ne pas confondre les deux significations du sigle EIAO, nous préférons utiliser le sigle ITS (Intelligent Tutoring System) ou tuteur intelligent, à la place de l'Enseignement Intelligemment Assisté par Ordinateur

[2] Virtual Environment for Training

[3] `http://www.infres.enst.fr/~grumbach/cognition-virtuelle/`

HPTS ([Donikian 01]) sont parmi les plus connues.

RICHARD propose de représenter l'environnement virtuel par un ensemble d'entités autonomes appelés agents [Richard 01]. Ainsi l'environnement d'un agent est l'ensemble des autres agents. Elle propose également une architecture d'agent fondée sur les capacités de perception, de décision et d'action. Le processus de décision s'articule autour de modules de comportement pouvant être ajoutés, supprimés ou modifiés dynamiquement. Cette architecture est destinée à la réalisation de comportements réactifs plutôt que cognitifs. Tous les agents de l'environnement sont conçus à l'aide de ce modèle. L'environnement est donc un système multi-agents homogène.

L'utilisateur est immergé dans l'environnement par le biais de son avatar. Son rôle est de permettre à l'utilisateur de percevoir et d'agir dans l'environnement tout en conservant sa capacité de décision. Le rôle de l'avatar est également de représenter l'utilisateur dans l'environnement virtuel pour être perçu par les autres utilisateurs ou agents autonomes. RICHARD propose également de modéliser l'avatar par un agent autonome. Ainsi il est possible de contraindre ou au contraire d'assister l'utilisateur dans l'environnement. Cette caractéristique est conforme à la notion de simulation participative définie par TISSEAU dans laquelle l'utilisateur est *"un modèle comme un autre"*.

Comme RICHARD, nous pensons que l'environnement virtuel peut être modélisé par un système multi-agents et que l'avatar est un agent en interaction avec les autres. Nous pensons par contre que ce système multi-agents ne peut être homogène. En effet, dans le cadre d'un environnement virtuel pour l'entraînement, l'environnement est à la fois physique et social. Cela signifie que les agents le simulant doivent disposer de capacités réactives et de capacités cognitives et sociales, ils n'ont donc pas tous le même type de comportement et ne disposent pas des mêmes mécanismes d'interaction et de communication. Le système est composé d'un grand nombre d'agents, il devient donc nécessaire de définir une structure organisationnelle afin d'optimiser les interactions entre les agents.

Cette approche de conception des environnements virtuels modifie la conception d'aide pédagogique. En effet, les modèles de l'expert, de l'étudiant et des stratégies pédagogiques sont distribués dans chaque élément de l'environnement. Les avatars (représentant les apprenants et les formateurs) peuvent donc jouer un rôle dans l'organisation pédagogique. Notre objectif n'est pas, dans un premier temps, de proposer une méthode de représentation de ces modèles, mais de prendre en compte leur caractéristique et de permettre leur implémentation. Dans ce travail, qui s'intéresse à la formation au travail collaboratif et procédural, nous considérons essentiellement la description des connaissances conceptuelles et procédurales.

1.1.2 Travail collaboratif et procédural

Le cadre d'étude choisi est la formation à la Gestion Opérationnelle et au Commandement (GOC) pour la sécurité civile. Une description détaillée de la GOC a été réalisée par [Daniel 01]. La formation à la GOC a pour objectif de fournir aux officiers sapeurs-pompiers une méthode pour faire face à des sinistres de complexité plus ou moins grande selon leur niveau de commandement. La GOC ne fournit que des règles génériques que les officiers doivent savoir adapter au cas particulier en situation réelle. Le noyau de cette méthode est la Méthode des Raisonnements Tactiques (MRT) [Samurcay et al. 91]. Ce type de travail est collaboratif et procédural.

Travail collaboratif

Il existe cinq niveaux dans la GOC (figure 1.1). Les seuls niveaux de commandement sont les chefs de site, de colonne et de groupe ; les autres niveaux (chef d'agrès, chef d'équipe, équipier) sont des exécutants. Par exemple, dans le cas d'un incendie, le groupe est formé de trois agrès (2 FPT[4] + 1 VSAB[5]). La GOC définit trois règles fondamentales de commandement :

- ▷ On ne peut commander plus de quatre subordonnés directs.
- ▷ Il est impératif de ne donner des ordres qu'aux subordonnés directs.
- ▷ Il faut rendre compte uniquement à l'échelon hiérarchique immédiatement supérieur.

La GOC définit donc une organisation hiérarchique.

Emplois opérationnels	**Moyens**	**Niveau**	**Grade**
Chef de site	plus de 1 colonne	GOC 5	Officiers supérieurs
Chef de colonne	2 à 4 groupes	GOC 4	Capitaines, Lieutenants
Chef de groupe	2 à 4 agrès	GOC 3	Lieutenants, Adjudants
Chef d'agrès	1 agrès	GOC 2	Adjudants, Sergents
Chef d'équipe	2 à 3 sapeurs	GOC 2	Caporaux, Sapeurs
Equipier	Aucun	GOC 1	Sapeurs

Figure 1.1 : Niveaux de formation à la GOC (d'après [Daniel 01]).

Les agrès sont en fait des équipes d'environ cinq sapeurs-pompiers. Chaque intervenant a un rôle bien défini. Par exemple, un agrès de type FPT est organisé autour de cinq pompiers (1 chef d'agrès, 1 chef alimentation, 1 équipier alimentation, 1 chef attaque, 1 équipier attaque). Le binôme (chef et équipier) d'alimentation a la responsabilité de fournir les ressources en eau au binôme d'attaque. Ces responsabilités

[4] Fourgon Pompe Tonne

[5] Véhicule de Secours aux Asphyxiés et Blessés

peuvent se comprendre comme l'ensemble des actions que doit mener le sapeur-pompier. Plus le niveau de commandement de ce dernier est bas dans la hiérarchie et plus ses actions sont des actions physiques. De telles actions modifient l'état de l'environnement ; il s'agit par exemple du déroulement d'un tuyau ou de l'arrosage d'un bâtiment. A l'inverse, plus le niveau de commandement du sapeur-pompier est élevé dans la hiérarchie et plus ses actions se traduisent par des envois d'ordres (au niveau inférieur) et de comptes rendus (au niveau supérieur) ; ces actions modifient l'état interne de l'agent (activation de capacités décisionnelles et modification des connaissances) et sont génératrices d'actes de langage. Ce type de travail est donc, à tous les niveaux de la hiérarchie, un travail collaboratif. La figure 1.2 montre un exemple d'organisation d'équipes pour l'intervention sur un site à risque.

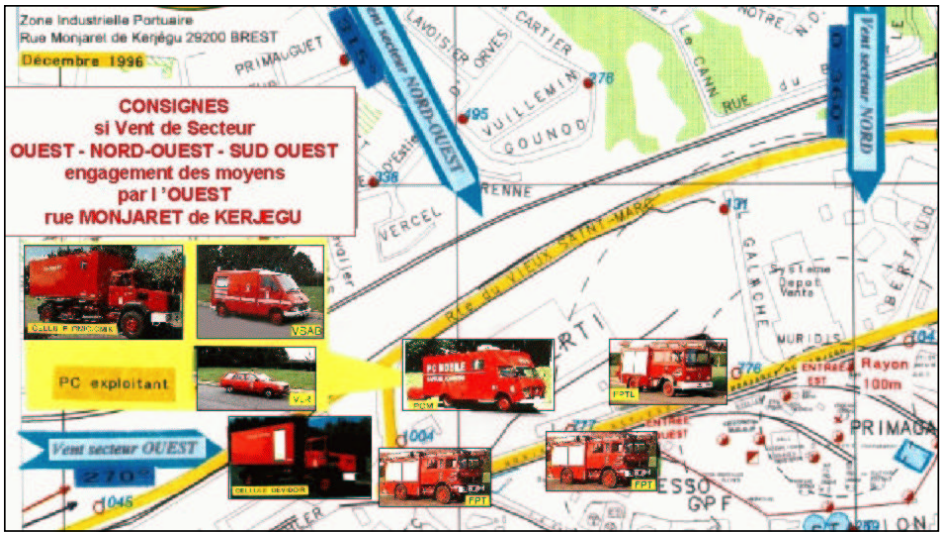


Figure 1.2 : Extrait de plan d'intervention (source : SDIS 29).

Travail procédural

Quel que soit leur niveau hiérarchique, les personnels de la sécurité adoptent la même méthode, la MRT. Cette méthode décrit le raisonnement que doit suivre un sapeur-pompier pour choisir les actions à effectuer. Elle prévoit les questions que doit se poser le sapeur-pompier afin de récolter les informations nécessaires sur le sinistre. Cette phase conduit à la définition des tâches à accomplir. Pour les niveaux "exécutant", l'organisation et les procédures sont écrites. Dans notre exemple d'équipe FPT, vingt manœuvres sont décrites dans un manuel. Ces manœuvres agencent les actions des différents intervenants. Dans le cas de sites à risques tels que les sites classés Seveso, les pompiers, les industriels ainsi que les institutions locales (mairie, préfecture...) doivent collaborer à la réalisation de plans prévisionnels. Ces plans décrivent, pour un site donné et un type de sinistre, l'organisation du commandement, les informations sur le site, les ressources, les tâches à accomplir... Il s'agit en fait de l'application de la MRT pour un certain nombre d'incidents prévisibles. La figure 1.3 montre un exemple de ces

tâches que doivent suivre les chefs de groupe intervenant dans le cas d'un incendie dans un site à risque. Quel que soit son niveau, un pompier doit suivre une procédure prévue en l'adaptant à la situation réelle. Plus le niveau de commandement du sapeur-pompier est bas dans la hiérarchie et plus les plans sont explicites. À l'inverse, plus le niveau du pompier est haut dans la hiérarchie et moins les plans sont explicites ; ils demandent plus de capacités d'adaptation.

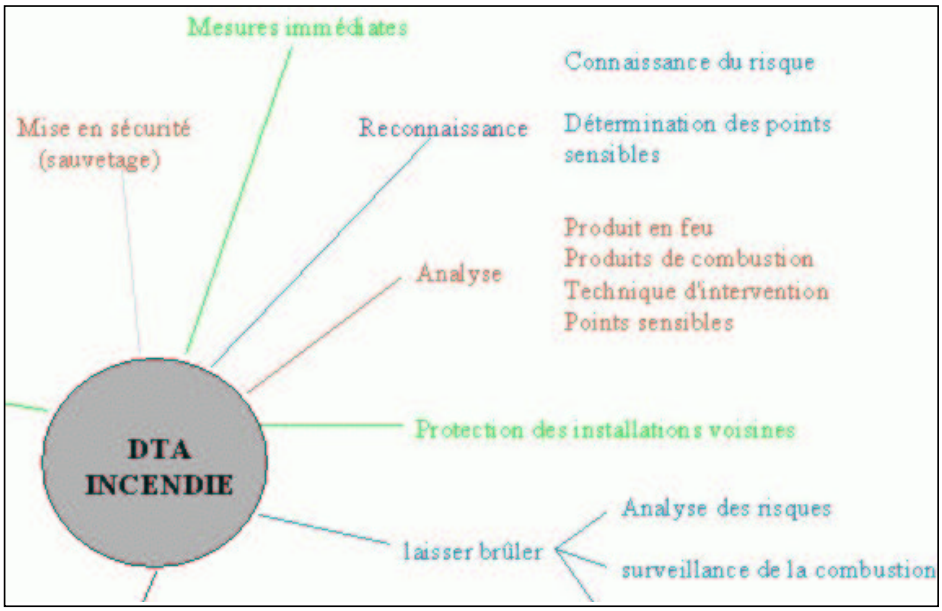


Figure 1.3 : Tâches à accomplir par un chef de groupe (source : SDIS 29).

Formation à la GOC

Classiquement, la formation à la GOC se déroule à la manière d'un jeu de rôles. Dans un premier temps, un des animateurs affecte les rôles aux stagiaires. Les animateurs jouent également des rôles dans le jeu. L'animateur présente alors la situation ainsi que les différents documents disponibles (plans d'intervention prévus, cartes, ressources disponibles). Les animateurs font ensuite dérouler le temps en créant des événements (par exemple : à T0 + 30 mn, arrivée du groupe incendie...) selon un scénario pré-établi. Après chaque phase importante du jeu (reconnaissance, mise en sécurité, attaque...), les animateurs peuvent arrêter le chef pour un bilan ou un recadrage des stagiaires. Le jeu continue et les animateurs, en déroulant le scénario, peuvent générer des événements imprévus pour prendre en compte les spécificités d'une solution choisie par un stagiaire. Une autre façon de jouer ce jeu de rôles est de faire jouer un rôle par plusieurs personnes composant une équipe. Les membres de l'équipe doivent alors collaborer pour choisir une solution commune qu'ils débattent ensuite avec les formateurs.

Ce mode de formation est, à l'heure actuelle, le seul envisageable car il est impossible de réaliser des exercices en grandeur nature sur le terrain. En effet cela

est trop dangereux et demande la mobilisation d'un grand nombre de personnels et de matériels, les rendant indisponibles pour les interventions éventuelles. Elle présente toutefois des lacunes importantes. Les différents intervenants (animateurs ou stagiaires) ne partagent pas une même représentation mentale de la situation. Certains vont alors surestimer (ou sous-estimer) les besoins en matériels ou en personnels. Il est d'autre part très difficile de s'écarter du scénario initialement prévu car l'équipe d'animation n'a pas les moyens de simuler mentalement les conséquences d'une évolution imprévue ; l'essentiel de son effort consiste à recadrer les stagiaires vers la situation prévue, ce qui est contre productif. De plus, ce genre de formation, ne favorise pas la prise en compte émotionnelle (stress, fatigue...) de la situation qui est pourtant un facteur important dans la gestion de crise.

1.2 Notre proposition MASCARET

Notre problématique est la définition d'un modèle pour la conception d'environnements virtuels destinés à l'entraînement au travail collaboratif et procédural. Cet environnement doit :

- ▷ simuler d'une part l'environnement physique, comme par exemple la propagation d'un nuage de gaz, et d'autre part l'environnement social, comme par exemple la réalisation d'une manœuvre d'établissement de lance à incendie par une équipe FPT ;
- ▷ permettre la réalisation de fonctions pédagogiques (évaluation, explication, critique...) pour adapter la formation à chaque apprenant.

Nous soutenons la thèse que cet environnement est un système multi-agents faisant interagir des agents ayant des capacités réactives, cognitives et sociales. Notre premier objectif est la formalisation de ce modèle que nous appelons MASCARET [6] pour *MultiAgent System for Collaborative and Adaptive Realistic Environment for Training* soit : système multi-agents pour les environnements réalistes, collaboratifs et adaptatifs pour l'entraînement. Notre second objectif est de montrer son application à la gestion opérationnelle et au commandement, ce qui se traduit par le développement d'une première version de l'application SécuRéVi (Sécurité civile et Réalité Virtuelle).

1.2.1 Le modèle MASCARET

Dans notre modèle, l'évolution de l'environnement est obtenue par la simulation des comportements locaux des agents autonomes, mais également par leurs interactions. MASCARET est un modèle permettant de structurer l'ensemble de ces interactions et fournissant aux agents les capacités d'évoluer dans ce contexte.

[6] En français, un mascaret désigne une longue vague déferlante produite dans certains estuaires par la rencontre du flux et du reflux.

Structure des interactions

Dans MASCARET, les interactions entre les agents doivent être structurées car :

- ▷ si l'agent ne dispose pas d'informations sur ses accointances, il est obligé d'effectuer des actions de perception qui sont coûteuses en temps de calcul et empêche un calcul en temps réel (surtout quand il y a beaucoup d'agents) ce qui est incompatible avec un environnement virtuel de formation ;
- ▷ pour adapter l'exercice à un apprenant particulier, le formateur doit pouvoir agir sur un type d'interaction entre des agents et donc pouvoir en parler ;
- ▷ le fonctionnement de l'environnement (social) est justement un domaine où les interactions sont structurées et il s'agit en partie du comportement que l'apprenant doit adopter.

Le concept d'organisation permet de structurer les interactions entre les agents. Cet agencement définit la distribution des rôles des agents ainsi que leur mode de communication. Il existe déjà des modèles d'organisation tels que AALAADIN [Gutknecht et al. 98] et MOISE [Hannoun et al. 99]. Mais ces modèles sont soit trop abstraits pour proposer une implémentation efficace ou soit trop spécialisés pour répondre à l'ensemble de notre problématique. D'autres types de modèle d'organisation, tels que celui de MAGIQUE [Bensaid et al. 97] ont été utilisés pour la conception de plates-formes de systèmes multi-agents mais n'ont pas comme objectif de fournir aux agents (autonomes ou humains) la capacité de raisonner sur leurs interactions. Nous proposons donc un modèle générique d'organisation de systèmes multi-agents qui, dans le cadre de notre problématique, nous permet de représenter à la fois l'environnement physique et l'environnement social. Ce modèle se doit d'être générique car certains agents participent aux deux types d'environnement et tout particulièrement les utilisateurs via leur avatar. Ce modèle s'articule autour de la notion de rôle représentant les responsabilités des agents et leur permettant, lors de leur processus de décision, de se construire une représentation des autres agents de l'organisation.

Comportement des agents

Dans l'organisation, les rôles sont joués par les agents autonomes. Pour intervenir au sein de l'organisation, ils doivent avoir les capacités adéquates. Dans le cadre d'étude que nous avons choisi, l'environnement est à la fois un environnement physique (phénomènes physiques) et social (équipe, hiérarchie). Les agents doivent être dotés de comportements réactifs, cognitifs et sociaux.

Nous distinguons deux types d'agent autonome que nous nommons les agents réactifs et les agents rationnels. Les agents réactifs ne disposent que de comportements réactifs, ils savent prendre en compte leurs accointances, mais n'effectuent aucun raisonnement. L'ensemble de ces agents forment l'environnement physique et leurs interactions représentent les phénomènes physiques. Les agents rationnels disposent de comportement réactifs, pour s'immerger dans l'environnement physique ainsi que des

comportements cognitifs et sociaux. Le comportement cognitif d'un agent tient ici dans sa capacité à choisir les actions qu'il effectue en fonction de ses buts, de l'état perçu de l'environnement et de son contexte organisationnel. Son comportement social décrit la manière dont il s'aide des autres membres de l'organisation pour résoudre des problèmes qu'il ne sait résoudre seul. L'utilisateur agit et perçoit l'environnement via son avatar qui est considéré dans MASCARET comme un agent rationnel particulier. À ce titre, un avatar peut participer aux organisations d'agents tout en assistant l'utilisateur qu'il représente.

1.2.2 L'application : SécuRéVi

SécuRéVi est une application de MASCARET à la formation à la GOC. L'objectif est de simuler le site d'intervention, les phénomènes physiques ainsi que les intervenants humains. Les apprenants et les formateurs y sont immergés afin de disposer du même support pour se faire une représentation mentale de la situation.

Le formateur immergé peut alors modifier l'environnement pour créer des incidents ou au contraire corriger des problèmes pour ne pas gêner le raisonnement des apprenants. Les apprenants participent à la simulation, via leurs avatars, en jouant le rôle qui leur est affecté. Leur objectif est de réaliser les actions qui sont de leur ressort tout en adaptant la procédure en fonction de la situation. Les apprenants disposent des mêmes documents qu'en situation réelle (plans d'interventions...). La figure 1.4 montre un exemple de scène dans laquelle les apprenants et formateurs sont immergés.

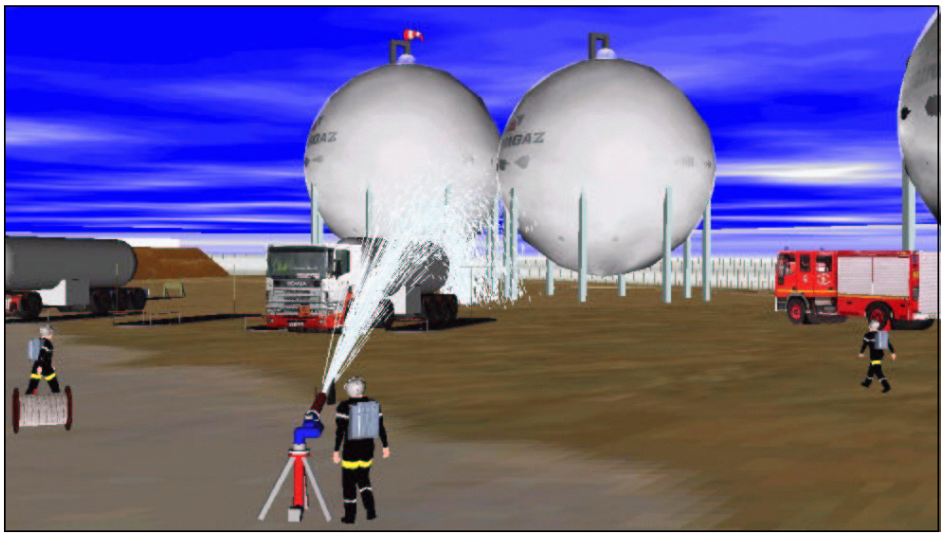


Figure 1.4 : Exemple de scène dans SécuRéVi.

Dans SécuRéVi, un apprenant intervient à un niveau de la hiérarchie, comme par exemple le chef d'un groupe (GOC 3). La réalisation de ses actions se traduit

alors par des envois d'ordres aux agrès qu'il commande et par des rapports qu'il transmet au niveau hiérarchique supérieur. Ce niveau est joué par le formateur. Les personnages participant aux agrès (niveau hiérarchique inférieur) sont simulés par des agents autonomes. L'évaluation de SécuRéVi est actuellement en cours par le SDIS 29.

1.3 Organisation de ce mémoire

Ce mémoire s'articule autour de trois grandes parties : le contexte qui oriente nos travaux et qui présente les concepts qui fondent notre approche (chapitre 2), le modèle que nous proposons (chapitre 3) et l'application qui nous inspire et valide notre modèle (chapitre 4).

Dans le chapitre 2, *"Le contexte : les environnements virtuels de formation"*, nous mettons en exergue les propriétés de la réalité virtuelle et des systèmes multi-agents qui sont les fondements de notre approche. Nous y présentons également les méthodes ou applications qui ont déjà été réalisées par d'autres auteurs.

Le chapitre 3, *"Le modèle : MASCARET"*, traite du modèle que nous proposons. Nous y exposons le modèle générique d'organisation qui nous permet de représenter les environnements physiques et sociaux. Nous y proposons également des architectures pour la conception d'agents réactifs et rationnels évoluant dans une organisation.

Une application de notre modèle est présentée dans le chapitre 4, *"L'application : SécuRéVi"*. Cette application est destinée à la formation à la gestion opérationnelle et au commandement. Nous présentons donc au début de ce chapitre les outils informatisés déjà utilisés par la sécurité civile, puis nous montrons la manière dont nous avons appliqué le modèle MASCARET à SécuRéVi ainsi que son implémentation grâce à la plate-forme ARéVi/oRis. La fin de ce chapitre présente un scénario type de simulation.

Enfin, dans le chapitre 5, nous concluons en dressant un bilan de nos recherches et en évoquant nos futurs travaux. Ce chapitre est suivi d'un glossaire qui rappelle la signification des principaux acronymes que nous utilisons dans ce mémoire.

Chapitre 2

Le contexte : les environnements virtuels de formation

Le contexte de notre travail est la conception d'environnements virtuels de formation. Ce type d'application s'appuie sur le paradigme du constructivisme pour lequel l'apprenant agit dans l'environnement pour construire ses connaissances. Nous avons rappelé dans le chapitre précédent l'intérêt d'utiliser la réalité virtuelle pour réaliser ce type d'application et nous avons également présenté notre thèse selon laquelle un environnement virtuel pour la formation est un ensemble d'entités autonomes en forte interaction. Parmi ces entités, certaines ont un comportement plus ou moins *intelligent* et d'autres représentent les utilisateurs du système (apprenants ou formateurs).

Notre problématique est donc double et concerne, de manière générale, la réalité virtuelle et les systèmes multi-agents. Le concept de réalité virtuelle intervient dans la conception d'EIAO pour la représentation réaliste et le rendu temps réel de l'environnement simulé. Ce concept permet également l'interactivité entre le système et les utilisateurs. Les systèmes multi-agents permettent la modélisation du comportement des entités autonomes et de leurs interactions. Les techniques classiques de l'intelligence artificielle (qui sont parmi les fondements des systèmes multi-agents) permettent de représenter le comportement rationnel de certains agents. Cette dualité se retrouve également dans la classification en deux catégories des applications de formation. En effet, nous distinguons les *micro-mondes* (ou environnements d'apprentissage distant), qui s'attachent à la reconstitution d'environnement et s'appuient essentiellement sur les techniques de la réalité virtuelle et les ITS (ou tuteurs intelligents) fondés principalement sur l'intelligence artificielle et les systèmes multi-agents.

L'objectif de ce chapitre est de préciser ces deux concepts ainsi que leurs utilisations et leurs apports à la formation. Dans la première section de ce chapitre, nous présentons la réalité virtuelle et ses applications aux environnements d'apprentissage. Dans la seconde section, nous abordons les systèmes multi-agents et nous montrons leurs applications aux outils de formation.

2.1 La réalité virtuelle

L'expression "réalité virtuelle" est aujourd'hui souvent utilisée dans de nombreuses applications, y compris dans un cadre de formation, sans que l'on sache véritablement ce qu'elle signifie. Notre objectif ici est de clarifier cette expression pour ensuite étudier la principale utilisation de la réalité virtuelle dans les applications destinées à la formation. Il existe une multitude d'environnements d'apprentissage considérés comme des environnements virtuels. Il n'est donc pas possible de tous les étudier. Nous préférons dans cette section étudier de manière générale les apports de la réalité virtuelle et présenter les critères de classements et méthodologies proposés dans la littérature.

2.1.1 Définitions

Pour définir la réalité virtuelle, nous partons de ses origines (l'informatique graphique, l'animation ...) et des premières définitions. Cette étude nous amène à rendre compte de deux notions essentielles à la réalité virtuelle : la présence et l'autonomie.

2.1.1.1 Réalité virtuelle et présence

La réalité virtuelle est une discipline à la croisée de l'informatique graphique, de la CAO (Conception Assitée par Ordinateur), de l'animation, de la téléopération... L'évolution de ces disciplines permet de mieux comprendre les tenants des applications de réalité virtuelle.

Les premiers problèmes de l'informatique graphique sont 2D (affichage de segments de droites, lignes cachées ...) Les études sur le sujet donnent naissance à la CAO qui a été un des principaux vecteurs de progrès dans cette discipline. On parle alors de surfaces et de courbes paramétriques, puis de triangulation. La 2D reste insuffissante et conduit à la modélisation de surfaces puis de volumes. Le concepteur peut dès lors parler de son objet à l'aide de primitives volumiques, de rotations, translations et déformations. Se posent alors les problèmes des faces cachées et de leurs éliminations. Mais cela ne suffit pas pour se faire une représentation réaliste. Des travaux sont alors menés sur l'éclairement des objets (Gouraud, Phong, radiosité...), les textures et les paramètres de caméras.

Une image fixe n'est souvent pas suffisante dans un processus de conception. Ainsi des études ont été menées sur la cinématique (*key-framing*, cinématique inverse, modèles dynamiques, capture de mouvements) ainsi que sur le calcul en temps réel par des calculateurs ou des algorithmes spécifiques (*culling*, gestion des niveaux de détails...)

Enfin, comme dans beaucoup de disciplines, l'informatique graphique se dirige vers le travail coopératif. Ainsi grâce aux réseaux informatiques, il devient possible de partager les modèles et de contrôler à distance un robot ou un environnement, tout en étant immergé (casque de visualisation, périphérique haptique...) dans une représentation de cet environnement.

Historiquement, la réalité virtuelle est donc une discipline intégrant l'ensemble de ces techniques de rendu, d'interaction et d'immersion. Les premières définitions et taxonomies datent des années 90. La première définition de la communauté française est celle de Burdea et Coiffet [Burdea et al. 93]. La définition qu'ils proposaient alors, introduit trois concepts :

- ▷ *Interaction* : l'utilisateur interagit dans l'environnement virtuel. L'environnement doit réagir en temps réel aux actions de l'utilisateur.
- ▷ *Immersion* : l'environnement est représenté de manière réaliste, ce qui génère une sensation d'immersion chez l'utilisateur.
- ▷ *Imagination* : c'est l'utilisation que l'utilisateur fait de l'environnement virtuel.

Cette définition est représentée par une figure géométrique désormais classique en réalité virtuelle et fondée sur les trois I : Immersion, Interaction et Imagination (figure 2.1).

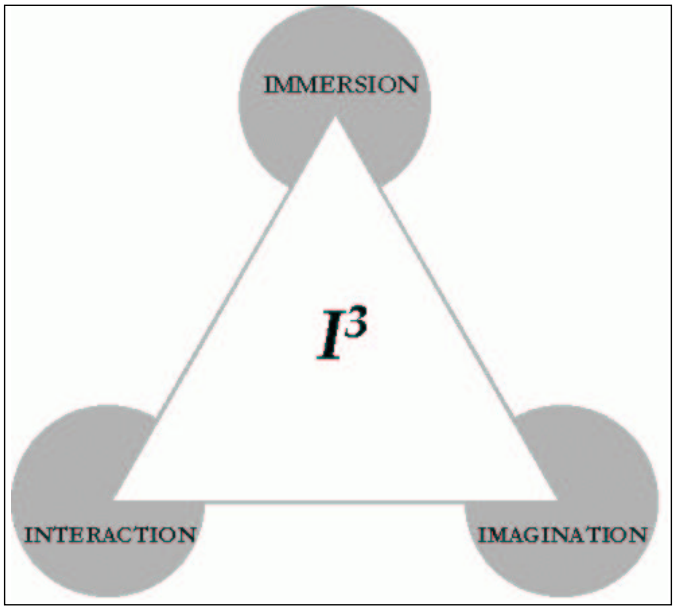


Figure 2.1 : Les trois I (d'après [Burdea et al. 93])

D'autres auteurs ont proposé une vision différente de la réalité virtuelle. Ainsi Fuchs [Fuchs 96] propose de caractériser un environnement virtuel en le comparant à l'environnement réel selon trois axes (temps, lieu, interaction). Une modification sur l'axe temps traduit un environnement passé ou futur, une modification sur l'axe lieu représente un environnement inaccessible et une modification d'interaction simule un environnement réel ou imaginaire (figure 2.2).

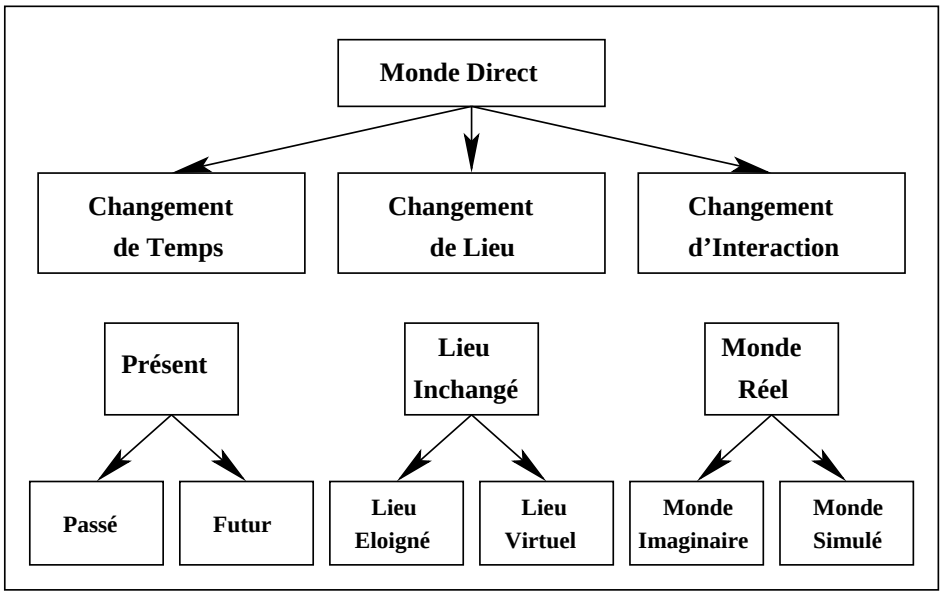


Figure 2.2 : Les trois axes de la réalité virtuelle. (d'après [Fuchs 96])

De plus, Fuchs propose d'utiliser une approche anthropo-centrique pour la réalisation d'environnements virtuels au niveau de l'interaction et de l'immersion. Il décrit alors des I^2 (Immersion-Interaction) :

- ▷ *sensori-motrices* : c'est la description physique de l'environnement et les interfaces comportementales.
- ▷ *cognitives* : il s'agit du comportement de l'utilisateur face aux interfaces et à la tâche à réaliser. L'auteur décrit alors le concept de Primitives Comportementales Virtuelles (PCV) pour qualifier les *"briques"* de comportement (actions).
- ▷ *fonctionnelles* : c'est l'objectif de l'application.

Ce sont ces interfaces comportementales qui assurent une véritable impression de présence au sein des univers virtuels avec lesquels l'homme interagit.

Les techniques actuelles de synthèse d'images et d'animation temps réel permettent en effet d'immerger les utilisateurs dans des environnements très réalistes. Il est possible de simuler des environnements réels ou imaginaires, passé, présent ou futur. Ces environnements ont surtout été utilisés pour la simulation (simulation d'éclairage, paysages urbains ...), pour le traitement de pathologies (claustrophobie ...) ou comme support artistique.

Notre étude s'intéresse à la conception d'environnements virtuels permettant d'immerger plusieurs personnes dans le but de réaliser des tâches dans cet environnement. Le problème de ce genre d'application est le manque d'activité dans l'environnement. En effet, rien ne se passe si l'utilisateur ne l'a provoqué lui-même. C'est le problème auquel ont été confrontés les environnements comme le Deuxième

MONDE[1] de Cryo et Canal+ qui permet à plusieurs personnes géographiquement réparties de partager un même environnement. Lorsque peu de personnes partagent l'environnement, celui-ci est en général peu attrayant.

L'idée est alors de recréer une vie autonome indépendante de la présence d'utilisateurs. De nombreux travaux se sont attaqués à ce problème. Les premiers travaux ont d'abord été des approches réactives pour modéliser des éléments de l'environnement. Puis des travaux ont tenté de modéliser des humanoïdes, parmi ceux-ci, ceux du LIG [Magnenat Thalmann et al. 91] ont longtemps fait référence. Pour améliorer la crédibilité de ces humanoïdes, certains travaux se sont ensuite orientés vers la description de scénarios qui décrivent l'enchaînement des actions d'un acteur virtuel. Un état de l'art sur ces techniques peut être trouvé dans [Devillers 01]. Cela ne favorise pas la prise en compte d'un environnement dynamique, qui est une contrainte forte lorsque l'humain est un acteur de cet environnement. Cette problématique de modélisation d'acteurs autonomes suivant, plus ou moins librement, un scénario est celle des *virtual storytelling*. Par exemple, le modèle d'acteurs autonomes décrit par [Cavazza et al. 01] permet à l'acteur de choisir le scénario réalisable à un instant donné en fonction de l'état de l'environnement. Le problème est que le concepteur, rédacteur de ces scénarios, doit prévoir tous les cas, ce qui pose le problème de l'autonomie des acteurs virtuels qui peuplent ces univers virtuels.

2.1.1.2 Réalité virtuelle et autonomie

TISSEAU définit dans un premier temps la réalité virtuelle comme "un univers de modèles au sein duquel *tout se passe comme si* les modèles étaient réels parce qu'ils proposent la triple médiation des sens, de l'action et de l'esprit" [Tisseau 01]. En effet, se référant aux philosophes, l'auteur affirme que l'homme appréhende le réel au travers de ces trois médiations (figure 2.3).

La médiation des sens permet à l'utilisateur de percevoir l'activité du modèle. Un exemple est le cinéma dynamique. L'ensemble des sens de l'utilisateur sont excités (vue, ouïe ...). L'utilisateur a alors une réelle impression d'immersion dans l'univers du modèle, mais n'a aucune influence sur son évolution, il subit cet univers.

La médiation de l'action permet à l'utilisateur de tester la réactivité du modèle. Aux travers de périphériques adaptés, en simulation interactive, l'utilisateur peut manipuler son environnement. Il ne peut en réalité que modifier des paramètres des modèles. Ces paramètres ont été prévus par le concepteur. Par exemple dans un simulateur de vol, l'utilisateur peut faire varier la vitesse, la direction ou l'orientation de son avion (le modèle dont il teste la réactivité).

Enfin, la médiation de l'esprit permet à l'utilisateur de modifier (en ligne) le modèle pour l'adapter à sa propre représentation mentale. En effet, la perception du

[1] `www.2monde.fr`

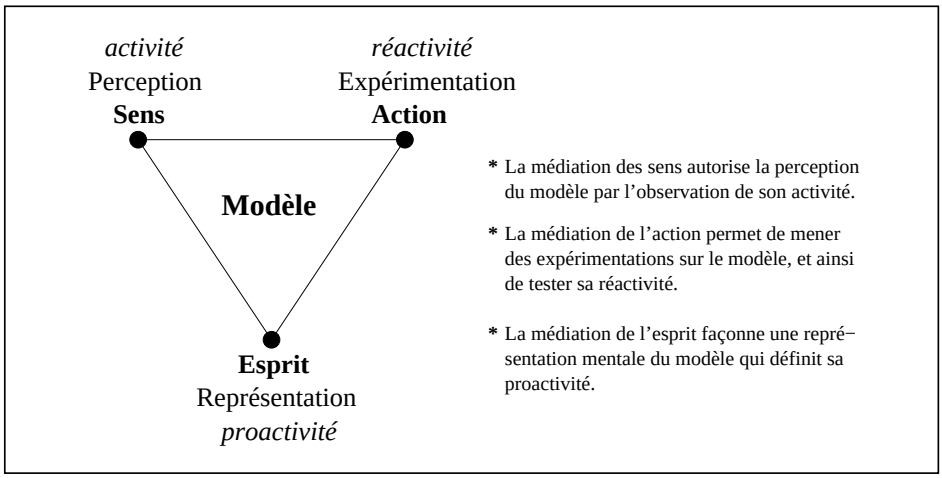
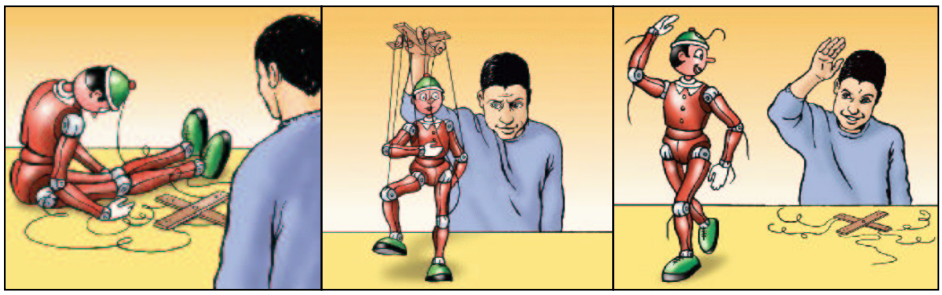


Figure 2.3 : Les trois médiations. (d'après [Tisseau 01])

modèle lui permet de s'en faire une représentation mentale. L'utilisateur confronte cette représentation au réel en la manipulant. Cela lui procure de nouvelles perceptions qui lui permettent de raffiner sa représentation mentale de la réalité. En fait, il modifie le modèle. Pour le modifier, il faut qu'il dispose des mêmes moyens que le concepteur ; il faut qu'il ait accès au même langage qui permet de concevoir et modifier le modèle. La médiation de l'esprit est fortement liée au langage ; c'est lui qui permet d'exprimer la représentation mentale : on parle alors de médiation du langage.

Pour illustrer ses propos, l'auteur prend l'exemple de Pinocchio (le modèle) et de Gepetto (l'utilisateur) (figure 2.4). Dans un premier temps, l'utilisateur ne peut qu'observer son modèle (a), puis le manipuler pour vérifier qu'il réagit tel qu'il se l'était représenté (b), et enfin le modifier pour qu'il corresponde à son imagination (c).



a) utilisateur−spectateur b) utilisateur−acteur c) utilisateur−créateur

Figure 2.4 : Pinocchio ou l'autonomie des modèles. (d'après [Tisseau 01])

Partant du principe qu'une altération d'une de ces trois médiations rend l'humain dépendant d'autrui, l'auteur en déduit que pour rendre les modèles autonomes, il faut qu'ils proposent cette triple médiation des sens, de l'action et du langage. Rendre autonomes les modèles permet dans certains cas de représenter de manière réaliste leur comportement réel (exemple : animats ...) mais surtout leur permet de s'adapter aux

modifications dynamiques de l'environnement.

Dans une application de réalité virtuelle, la place de l'utilisateur n'est pas la même qu'en simulation scientifique ou interactive. En simulation scientifique, l'utilisateur intervient avant la simulation pour fixer les paramètres et après pour analyser les résultats. L'utilisateur n'intervient pas au cours de la simulation. Ce type de simulateur est centré sur le modèle. En simulation interactive, l'utilisateur agit en cours de simulation. Mais il ne fait que modifier des paramètres qui ont été prévus par le concepteur. La simulation est centrée sur l'utilisateur.

En réalité virtuelle, le modèle et l'utilisateur sont placés au même niveau. Pour agir sur le modèle, il faut que l'utilisateur ait les moyens physiques de le manipuler (percevoir et modifier). En fait, l'utilisateur est lui-même modélisé dans l'environnement (moyens sensitifs et moyens d'actions). L'utilisateur est donc un modèle comme un autre dans l'univers des modèles.

Ce modèle est bien sûr autonome puisqu'il propose la triple médiation des sens, de l'action et de l'esprit. Un tel modèle s'appelle un *avatar*. Il modélise les capacités perceptives de l'utilisateur ainsi que ses capacités d'action dans l'environnement. L'humain conserve ses capacités de décision, même s'il peut en déléguer une partie à son avatar. L'utilisateur va alors pouvoir interagir avec les autres modèles. Il devient donc lui même un des résultats de la simulation. Ainsi la réalité virtuelle est qualifiée de participative puisque l'utilisateur participe à la simulation.

2.1.2 Réalité virtuelle et formation

Comme il n'est pas possible d'énumérer ici l'ensemble des environnements virtuels de formation, nous montrons, dans un premier temps, comment il est possible de les classer et ce qu'ils apportent à la formation. Puis, nous mettons en valeur l'effort actuel qui est réalisé pour décrire des méthodologies permettant de concevoir ces environnements afin qu'ils répondent au mieux aux besoins pédagogiques.

2.1.2.1 Critères de classement

Devant la multitude et la diversité des environnements virtuels de formation, Mellet d'Huart expose différents critères pour évaluer les fonctionnalités d'un système informatique pour la formation [Mellet d'Huart et al. 01].

Le premier critère est le *rapport à la réalité*. L'environnement représenté est-il une représentation fidèle de la réalité, imaginaire ou augmentée ? La figure 2.5 (gauche) montre un vue d'une application (EDF) de manipulation d'un robinet industriel. Dans cette application, l'utilisateur peut observer le mécanisme interne du robinet, ce qui n'est pas possible en réalité.

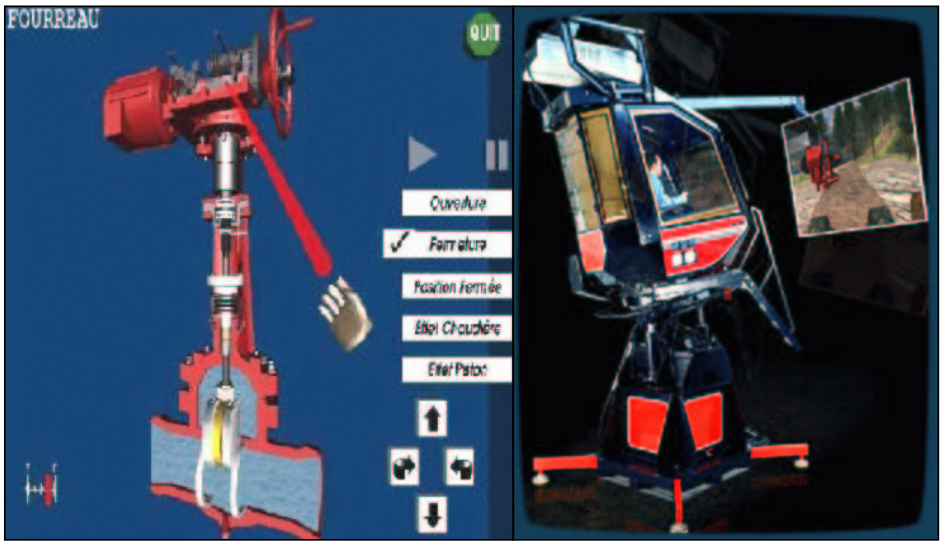


Figure 2.5 : Un robinet virtuel et ARVESTER.

Les *fonctions de l'application* représente le second critère de classement. Le système proposé peut être un simple outil de navigation ou bien proposer une interaction et une simulation de l'environnement à l'utilisateur. Dans les meilleurs cas, le système propose des fonctions pédagogiques. ARVESTER[2] (Abattage Reconstitué Virtuellement En Synthèse Temps Réel) est un simulateur d'abattage d'arbres. La cabine de l'engin réel est montée sur vérins hydrauliques, ce qui permet de simuler la topologie du terrain. Sur la cabine est fixé un écran qui reproduit l'environnement de l'apprenant. La figure 2.5 (droite) montre un étudiant utilisant ce simulateur.

Le troisième critère est la capacité de *modification de l'environnement* : l'environnement peut être complètement pré-défini et statique ou bien l'utilisateur se crée lui-même en cours d'exercice son propre environnement. C'est le cas par exemple avec Herman The Bug [Lester et al. 99], un assistant pédagogique qui aide les utilisateurs à créer leur environnement. Cet environnement est composé de plantes que l'apprenant assemble et qui évoluent ensuite de manière autonome ; l'apprenant étudie ainsi les règles de botanique et de physiologie. La figure 2.6 montre une vue de cet environnement de formation.

Le quatrième critère est l'*existence de population*. Le système peut être complètement vide, proposer uniquement une représentation de l'utilisateur ou faire vivre des agents autonomes pour améliorer le réalisme de l'environnement. Certains systèmes proposent des agents autonomes qui participent aux tâches à réaliser dans l'environnement ou qui implémentent des fonctions pédagogiques pour diriger l'apprenant. Par exemple, dans [Bindiganavale et al. 00], un soldat (apprenant) est placé dans un environnement virtuel de formation[3] simulant une situation de crise

[2] `http://www.ai.cluny.ensam.fr`

[3] `http://hms.upenn.edu/vet`

Figure 2.6 : HERMAN (d'après [Lester et al. 99]).

(figure 2.7 droite). Le rôle de l'apprenant est de donner des ordres à d'autres soldats, représentés par des agents autonomes, pour rétablir la paix.

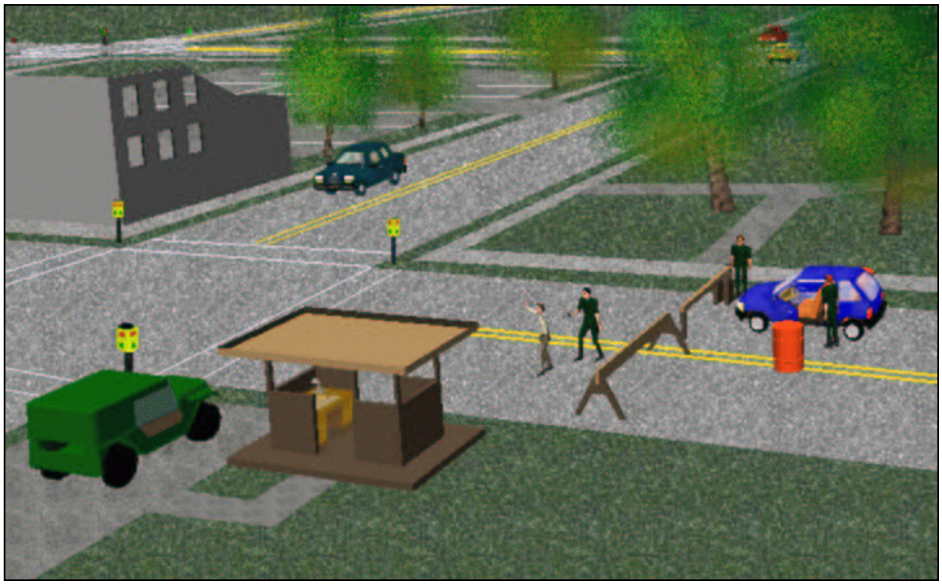


Figure 2.7 : Un simulateur militaire (d'après [Bindiganavale et al. 00]).

Enfin, le caractère *multi-utilisateurs* est le dernier critère de classement. Le système permet-il à plusieurs apprenants de partager le même environnement ? C'est par exemple le cas des environnements d'apprentissage distant qui permettent à

plusieurs personnes, géographiquement réparties, de partager un même environnement pour y recevoir des informations ou effectuer des tâches. Les termes de *e-learning*, d'*université virtuelle* ou de *micro-mondes* sont également utilisés pour désigner de telles applications. C'est ainsi que différents environnements virtuels pour la formation se sont développés sur internet ; on parle alors d'université virtuelle. EVE[4] (Environnement Virtuel pour les Enfants) (figure 2.8) est l'exemple d'un tel environnement où se retrouvent des enfants de France, du Maroc et de Roumanie pour effectuer des jeux pédagogiques destinés à l'apprentissage de la lecture [Gerval et al. 02].

Figure 2.8 : Vue d'une scène dans EVE (d'après [Gerval et al. 02]).

2.1.2.2 Avantages et limites

L'étude des exemples précédents a permis de montrer que la réalité virtuelle dispose de caractéristiques intéressantes pour la formation [Kalawsky 96]. L'apprenant est immergé dans un univers synthétique. Cette *immersion* lui donne une impression de présence dans l'environnement. L'apprenant a alors une meilleure perception des échelles. De plus, le système représenté peut être très *complexe* (nombre d'éléments, complexité géométrique). Un ensemble de vues statiques ne permettent pas à l'apprenant de comprendre le phénomène. L'utilisation d'un système de réalité virtuelle permet une consultation interactive du phénomène (infinité de vues) et donc une meilleure compréhension. De plus, la réalité virtuelle offre une *interaction intuitive*. L'utilisateur (apprenant ou formateur) peut alors manipuler son environnement,

[4] `http://www.enib.fr/eve`

peut-être même, avec un plus haut niveau d'interaction que dans la réalité. Un système de réalité virtuelle permet également de *contrôler les lois de la physique* (simplification ou exagération). Cela permet à l'apprenant de se concentrer sur le phénomène en lui-même. De même, le contrôle de l'échelle de temps (ralentissement ou accélération) permet une meilleure compréhension. L'environnement simulé peut représenter un environnement à risque dans lequel il est dangereux d'intervenir dans la réalité. En réalité virtuelle, le professeur peut provoquer des dysfonctionnements dans l'environnement, en toute *sécurité* pour évaluer les réactions de l'étudiant.

L'environnement des utilisateurs devient donc véritablement attrayant, voire ludique, mais certains pensent que cela peut nuire à l'objectif pédagogique et proposent des solutions pour décrire le contenu et les fonctions pédagogiques de telles applications [Ramel et al. 00]. L'autre danger est de déplacer la responsabilité de description du scénario du cours, du professeur vers le concepteur de l'outil. Il faut donc prévoir dans l'application, une possibilité aux professeurs de décrire leurs propres scénarios et exercices [Guillet et al. 00].

2.1.2.3 Méthodologies

Les premières applications de réalité virtuelle à la formation ont montré leurs limitations. En effet, la conception d'un environnement virtuel de formation ne doit pas être centrée sur la technologie utilisée, mais sur l'objectif pédagogique. Pour éviter ce genre d'erreurs, des méthodologies pour la conception d'environnements virtuels pour la formation commencent à être étudiées.

Ainsi, Lourdeaux propose une méthodologie de conception fondée sur plusieurs étapes (figure 2.9) [Lourdeaux 01]. La première étape est la spécification des objectifs pédagogiques. Pour cela, le concepteur réalise des recueils d'expertises auprès des experts de la tâche, mais également auprès de la population cible et des formateurs. L'auteur souligne que les tâches pédagogiques n'ont pas forcément besoin d'être liées de manière très étroite au réel. Par exemple, pour une tâche s'effectuant en situation de crise dans l'environnement réel, il n'y a pas toujours besoin de recréer cette situation et de générer du stress chez l'étudiant pour atteindre les objectifs pédagogiques fixés.

Les étapes suivantes consistent à spécifier l'environnement virtuel pour le formé et le formateur à partir des objectifs pédagogiques. Il s'agit de déterminer les PCV[5] à implémenter dans l'environnement virtuel (marche, préhension ...) Il faut ensuite spécifier les représentations mentales à implémenter dans l'univers (schème habituel, métaphore ...) et enfin, les interfaces comportementales qui permettent aux intervenants d'interagir dans l'environnement. La dernière étape est la réalisation des tests d'utilisabilité et cognitif pour évaluer le comportement des utilisateurs et quantifier le transfert des connaissances.

[5] Primitives Comportementales Virtuelles ([Fuchs 96]).

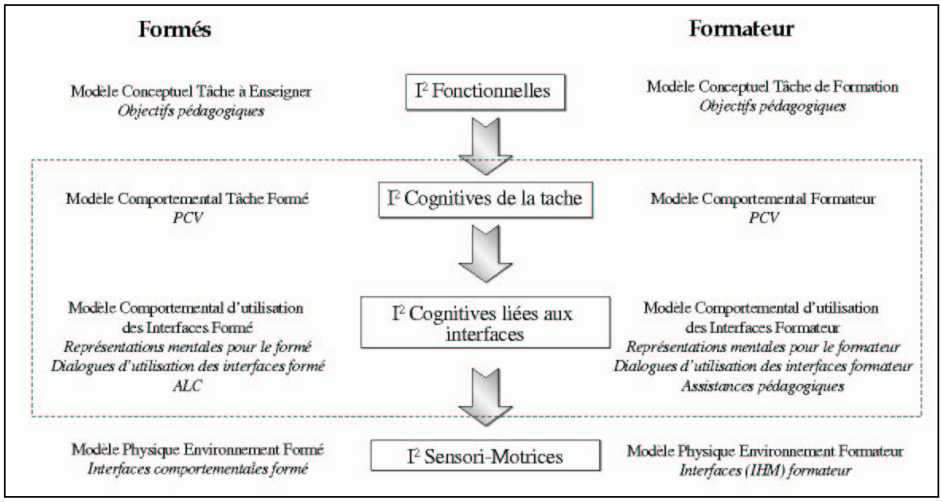


Figure 2.9 : Conception d'un EV pour la formation (d'après [Lourdeaux 01]).

Cette méthodologie s'applique mieux aux formations aux gestes techniques qu'à celles qui forment à la prise de décision. En effet, de telles applications demandent un fort réalisme au niveau du rendu. Cette idée est également défendue par [Crampes et al. 99] ; l'auteur y définit plusieurs critères pour les simulateurs à vocation comportementale où l'objectif est de former l'apprenant à bien réagir face à son environnement.

Les stratégies pédagogiques sont un concept important dans de tels systèmes. Elles sont composées d'un critère de motivation de l'apprenant (enjeu, temps limité ...), un mécanisme d'évaluation de l'étudiant ainsi qu'un mode d'apport et de construction des connaissances (accès au cours, intervention du formateur). Le formateur peut être aidé par un agent intelligent ; il faut alors bien répartir les rôles de chacun en fonction des stratégies pédagogiques choisies.

Le scénario et la mise en scène sont également des concepts importants lors de la réalisation d'un simulateur pédagogique. Crampes distingue deux types de scénario : le scénario-canevas qui fixe le lieu et les différents acteurs et laisse les interactions s'opérer, et le scénario programmé qui fixe les situations prédéfinies par lesquelles l'apprenant doit passer. Un scénario est composé :

- ▷ de l'univers simulé : les entités physiques ainsi que les interfaces,
- ▷ des personnages : les acteurs simulés ainsi que ceux qui seront éventuellement joués par les apprenants et le professeur,
- ▷ du modèle de l'univers : relations entre les actions des apprenants et l'évolution du monde simulé,
- ▷ du déroulement détaillé dans le cas d'un scénario programmé.

La mise en scène est la présentation du monde simulé, des interfaces et des rôles que doivent jouer les apprenants. L'auteur définit également des critères de qualité du scénario et de la mise en scène. Les premiers critères sont la crédibilité

des représentations, la pertinence pédagogique et la véracité (fidélité par rapport au réel). L'implication des apprenants dans le scénario est également un de ces critères. Il s'agit d'évaluer l'aptitude du simulateur à inviter (plus ou moins explicitement) les apprenants à entrer dans le scénario. Les auteurs proposent même de dramatiser le scénario pour favoriser cette implication. Enfin, le dernier critère proposé est la distanciation de l'apprenant face à l'environnement simulé. En effet, l'objectif du simulateur est pédagogique et non ludique. Il faut ménager des phases d'analyse de la situation.

2.2 Les systèmes multi-agents

La réalité virtuelle, vue comme une simulation participative à un univers de modèles autonomes en interaction, trouve un support d'implémentation idéal dans les systèmes multi-agents. Tout comme la réalité virtuelle, ce concept nouveau est difficile à définir. Il est utilisé aujourd'hui dans différents cadres et désigne des applications structurées autour de composants plus ou moins indépendants les uns des autres. Le premier objectif de cette section est de préciser cette notion de systèmes multi-agents.

Les systèmes multi-agents sont historiquement fondés sur l'intelligence artificielle classique dont la principale utilisation dans les outils de formation est la conception de tuteurs intelligents. La notion de distribution de l'intelligence (IAD), qui fonde les systèmes multi-agents, se retrouve également dans les outils de formation actuels. Notre second objectif dans cette section, est d'explorer les apports de l'intelligence artificielle et des systèmes multi-agents dans la conception d'environnements virtuels de formation.

2.2.1 Définitions

Dans ce chapitre, notre objectif n'est pas de comparer ou d'étudier différentes plateformes de simulation de systèmes multi-agents. En effet, ce travail à déjà été réalisé par différents auteurs tels que [Mandiau et al. 02, Hermes 02], et d'autre part, les travaux que nous menons se veulent indépendants de toutes plateformes. Notre objectif est d'etudier les différents concepts intervenant dans la réalisation d'un système multi-agents, pour cela, nous organisons notre discours en nous inspirant du modèle VOWELS décrit dans [Demazeau 95]. Dans ce modèle, un système multi-agents est décrit par

quatre facettes (les quatre voyelles AEIO) :

- ▷ **A**gent : les entités autonomes sont douées de capacités de perception, de décision et d'action.
- ▷ **E**nvironnement : c'est le support des actions des agents. Le jeu des interactions entre agents modifie l'environnement.
- ▷ **I**nteractions : le comportement du système étant vu comme une somme de comportements localisés, il faut définir les interactions entre ces comportements locaux.
- ▷ **O**rganisation : le comportement et les interactions entre les agents sont modifiés par la prise en compte (ou non) de l'organisation des agents.

Comme le propose [Tisseau 01], ce modèle aurait pu être enrichi d'une dernière voyelle, la lettre U pour Utilisateur. En effet, dans certaines applications des systèmes multi-agents (simulation, agents d'interface...), l'utilisateur interagit avec les agents dans l'environnement et peut même être un des membres de l'organisation. Ce modèle a donné lieu à des applications et même à des plates-formes de simulation. Ce ne sont pas ces applications qui nous intéressent ici, mais l'organisation de la réflexion sur la conception de tels systèmes. Dans cette section, nous détaillons les cinq facettes (Agent, Environnement, Interactions, Organisation et Utilisateur) en explicitant les principales caractéristiques de chaque facette.

2.2.1.1 Agent

Le terme d'agent est fortement lié à celui d'autonomie. Cela veut dire, d'une part, qu'il n'y a pas de contrôle global du système et d'autre part, pas de nécessité de contrôle de l'utilisateur. Cela veut également dire que l'agent dispose au minimum de mécanismes qui lui permettent de réagir face aux changements de l'environnement. Un agent devient pro-actif lorsqu'il possède ses propres buts. S'il a conscience et sait gérer les interactions avec les autres agents, on parle alors d'agent social. La manière dont un agent gère son autonomie est appelée son comportement.

En fonction de la complexité de ce comportement autonome, les agents peuvent être classés en agents réactifs et agents cognitifs [Wooldridge et al. 95]. Au vue de la diversité des typologies d'agents décrites dans la littérature, la frontière entre agents réactifs et agents cognitifs est floue. Plutôt que de tenter de donner notre propre typologie, nous allons expliquer ici les différentes capacités dont un agent peut être doté.

Capacités réactives

Un agent réactif est capable de réagir aux modifications de l'environnement, il dispose de méthodes pour tenir compte de ces modifications. Il s'agit dans ce cas d'actions réflexes, aucun processus de réflexion n'est mis en œuvre. Le comportement de l'agent

est régi par un cycle Perception / Action. Ainsi dans [Drogoul 93], le comportement d'un agent réactif est *"basé sur le principe d'action comme réaction à un stimulus"* de l'environnement.

La somme des comportements réactifs considérés comme simples permet de réaliser un système dont le comportement global est considéré comme complexe. Ce mode de conception s'appele l'émergence [Jean 97]. Le but du système étant la réalisation du comportement complexe, il s'agit ici de définir le comportement basique d'un agent ainsi que le nombre de ces agents qui permettent de réaliser cette tâche complexe. L'éthologie est un exemple privilégié d'émergence de comportements complexes à partir d'une somme de comportements simples est l'éthologie. La simulation de comportements de fourmis ou d'araignées sont parmi les plus classiques [Dury et al. 01].

Capacités pro-actives

Certains agents ont des capacités plus évoluées pour agir dans l'environnement, ils disposent d'intentions et de buts. Ces agents sont pro-actifs car ils ne font pas que subir l'environnement et réagir à sa modification, mais décident d'agir après une phase de raisonnement. Ils sont alors capables d'anticiper l'évolution de l'environnement et de planifier leurs actions. L'étude d'un agent pris individuellement s'approche ici des études classiques de l'intelligence artificielle. On parle alors également d'agents cognitifs, d'agents intelligents ou d'intelligence artificielle distribuée.

Capacités sociales

Certains agents disposent de capacités sociales qui leurs permettent de coopérer ou au contraire d'entrer en compétition. Les différents termes étudiés ici, sont empruntés aux études en sociologie. Dans la littérature[6] [Ferber 95, Huhns et al. 99], les mêmes termes (coopération, coordination) ne sont pas utilisés de la même manière, ce qui rend difficile l'établissement d'une définition précise. Nous avons choisi d'utiliser le terme de coopération pour décrire l'ensemble des mécanismes sociaux mis en œuvre par les agents pour agir dans un but commun (ce qui se rapproche de [Ferber 95]). Ces mécanismes sociaux sont la coordination d'actions, la collaboration et la négociation.

Dans [Ferber 95], la coordination d'actions dans un système multi-agents est définie comme l'ensemble des tâches effectuées par les agents pour réaliser les autres actions (actions effectives) dans les meilleures conditions. Il s'agit essentiellement de régler les problèmes de conflits d'accès aux ressources. Ferber décrit alors quatre

[6] Système multi-agents et Intelligence Artificielle Distribuée : `http://www.poleia.lip6.fr/~drogoul/courses`

types de coordination d'actions :

- ▷ La synchronisation décrit l'enchaînement des actions.
- ▷ La planification est la phase de réflexion de l'agent qui lui permet de définir l'ensemble des actions à réaliser pour atteindre un but.
- ▷ La coordination réactive est la coordination d'actions d'agents réactifs qui n'effectuent aucune planification a priori, mais agencent leurs actions *in situ*.
- ▷ La coordination par réglementation qui vise à donner des règles de comportements à chaque agent afin d'éviter les conflits.

Dans [Grosz et al. 96], les auteurs définissent la collaboration entre agents comme une activité spéciale de coordination entre agents. Lors de la collaboration, les agents agissent pour atteindre un but commun qu'aucun agent ne pourrait atteindre seul. Il s'agit ici de résoudre des problèmes de compétences des agents et d'allocation de tâches. Les modèles d'agents permettant la collaboration [Miller et al. 00] intègrent en général des notions de rôles comme une description des compétences de l'agent et d'équipe comme une organisation de ces rôles. Ces notions seront approfondies dans la section 2.2.1.4.

La négociation est souvent placée dans les mécanismes intervenant entre des agents ayant des buts contradictoires, mais visant à atteindre un but médian [Faratin et al. 98, Huhns et al. 99, Mathieu et al. 00]. Chacune des parties intervenant dans la négociation fait des propositions contradictoires. C'est un processus itératif qui peut se dérouler sous certaines contraintes (temps, obligation d'accord...). Les méthodes de négociation utilisées dans les systèmes multi-agents visent à résoudre des problèmes d'allocation de ressources ou d'exécution de services. Le résultat de la négociation est appelé un contrat. Dans ce genre d'application, les agents communiquent explicitement par actes de langage.

Architectures d'agents

Différents types d'architecture d'agents intelligents ont été définis. Des revues de ces architectures et de leurs utilisations ont été détaillées dans [Ferber 95, Wooldridge 99, Bryson 00]. Les principales architectures développées sont basées sur les comportements réactifs, sur la logique ou sur les croyances et intentions (BDI) des agents.

L'idée principale des architectures basées sur des comportements réactifs est que le comportement intelligent de l'agent émerge des interactions entre ses comportements plus simples. Ce type d'architecture est fondée sur le concept de comportement. Chaque comportement est considéré comme la réalisation d'une action basique. Un comportement est régi par un cycle de Perception / Action dans lequel l'agent effectue directement une action en fonction de sa perception, sans réaliser de réflexion. Comme plusieurs comportements peuvent être exécutables en même temps, l'agent doit disposer d'un moyen de choisir parmi les comportements entrant en conflit. Pour cela, l'architecture à subsomption, définie par Brooks propose d'organiser les comportements de manière hiérarchique dans laquelle les comportements de niveaux

inférieurs peuvent inhiber les comportements de niveaux supérieurs (figure 2.10) [Brooks 86].

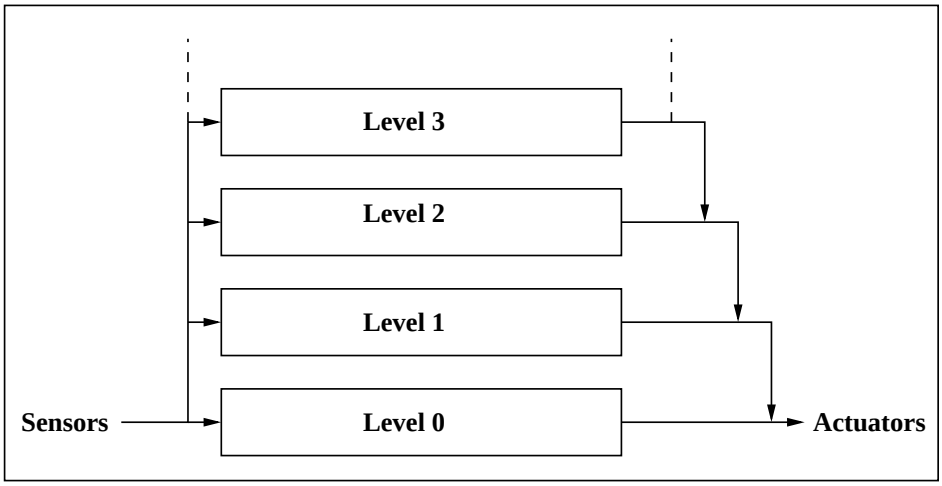


Figure 2.10 : Architecture à subsomption (d'après [Brooks 86]).

Dans les architectures fondées sur la logique, l'agent doit disposer de représentations symboliques de l'environnement. L'agent dispose donc d'un ensemble d'informations qui sont exprimées sous forme de prédicats de la logique du premier ordre (base de faits). Cette dernière est vue comme la base des croyances chez l'humain. L'agent est également doté de règles d'inférence et la réalisation d'une action revient à appliquer une règle. La réalisation d'une action modifie l'environnement et donc la base de faits de l'agent. Un problème survient quand plusieurs règles (ou aucune) peuvent être tirées en même temps, l'agent doit-il faire un choix, les exécuter toutes en parallèle ou les fusionner ? La manière dont l'agent résout ce problème peut être considéré comme son comportement. SOAR [Newell 90] est l'exemple d'un tel système qui a été appliqué dans des projets de formation en réalité virtuelle [Rickel et al. 99] et en simulation de combats aériens [Jones et al. 93].

Les architectures de type BDI (*Believe, Desire, Intention*) [Rao et al. 91] sont basées sur un PRS (*Procedural Reasoning System*) où l'idée principale est de construire un plan d'actions pour atteindre le but fixé. Un tel système repose sur quatre modules :

- ▷ les croyances de l'agent (*Believe*) : ce sont les connaissances de l'agent sur l'état du monde (obtenues via ses capteurs et les communications),
- ▷ les désirs (ou buts) de l'agent (*Desire*) : c'est l'état du monde que l'agent souhaite atteindre,
- ▷ l'ensemble des plans : c'est la connaissance procédurale de l'agent,
- ▷ les intentions de l'agent (*Intention*) : c'est la liste des plans que l'agent a selectionné en fonction de ses croyances et de ses désirs.

Partant du principe qu'un agent est capable de comportements réactifs et de comportements pro-actifs, les architectures multi-niveaux décomposent le comportement

de l'agent en couches de complexité croissante. Un tel système est composé d'au moins deux couches : une couche réactive et une couche pro-active. Parmi ces architectures, on peut distinguer trois types de système :

- ▷ *Système en couche horizontale* : chaque couche perçoit l'environnement et propose l'exécution d'une action. Il faut alors définir la manière dont l'agent choisit la couche prioritaire dans le cas de conflits. Un exemple d'une telle architecture (TOURING MACHINE) proposée par [Ferguson 92] est donnée figure 2.11.
- ▷ *Système en couche verticale à une passe* : les informations perçues passent de couche en couche en commençant par celle de niveau inférieur. Chaque couche propose l'exécution d'une action à la couche de niveau supérieur et c'est la dernière couche qui désigne l'action à exécuter.
- ▷ *Système en couche verticale à deux passes* : dans cette architecture, la première passe est d'abord exécutée, le choix de l'action redescend de couche en couche et c'est la couche inférieure qui effectue l'action.

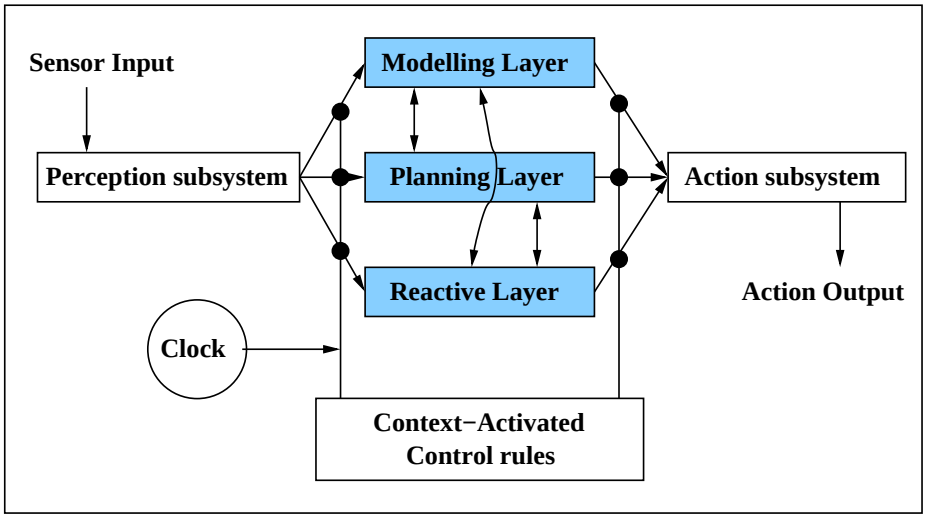


Figure 2.11 : TOURING MACHINE (d'après [Ferguson 92]).

2.2.1.2 Environnement

BOISSIER[7] distingue l'environnement du système multi-agents et l'environnement de l'agent. L'environnement du système multi-agents est l'environnement physique support des actions des agents et disposant de ressources. L'environnement de l'agent est l'environnement du système multi-agents et les autres agents.

Dans le cadre de notre étude, nous nous intéressons aux agents situés, évoluant dans un environnement physique. L'environnement est décrit comme le support des interactions (messages, traces...) et des actions individuelles ou collectives des agents.

[7] Cours sur les systèmes multi-agents: `http://www.emse.fr/~boissier/enseignement/sma/`

C'est également l'espace des déplacements (position, grille) et un moyen de structuration (topologie spatiale ou temporelle) des agents. C'est enfin la source des données du problème et le lieu ou l'agent peut trouver des ressources.

Russel et Norvig [Russel et al. 95] définissent les principales propriétés de l'environnement :

- ▷ Accessibilité : l'agent dispose de capteurs lui permettant d'accéder à l'état de l'environnement (en géneral localement).
- ▷ Déterminisme : le prochain état de l'environnement est fonction de son état courant et de l'action de l'agent.
- ▷ Episodisme : les prochaines évolutions ne dépendent pas des actions déjà réalisées.
- ▷ Dynamisme : l'environnement peut changer pendant la réflexion de l'agent.
- ▷ Discret : le nombre de percepts et d'actions de l'agent est limité.
- ▷ Concurrence d'accès : d'autres agents souhaitant interagir existent dans l'environnement.

2.2.1.3 Interactions

Le principe d'implémentation des systèmes multi-agents est localisé, c'est-à-dire que le concepteur ne se soucie pas du comportement global du système, mais uniquement des différentes entités autonomes en décrivant leur comportement localement. Pour obtenir un comportement global cohérent, il faut alors décrire la manière dont les entités vont interagir entre elles et avec l'environnement.

Boissier définit une interaction comme "une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques" [8]. Il rajoute : "il y a interaction lorsque la dynamique propre d'un agent est perturbée par les influences des autres". Cela sous-entend différentes formes d'interactions selon les agents et les systèmes.

Le premier type d'interaction s'opère sans communication et sans contact direct entre les agents, ils n'ont donc pas besoin d'avoir conscience des autres. L'agent utilise l'environnement comme média d'interaction en le modifiant à travers ses actions. Le principal exemple d'interaction de ce type sont les applications faisant intervenir des agents réactifs, tel qu'en éthologie ou en traitement d'images [Ballet et al. 97]. Le deuxième cas d'interaction sans communication fait intervenir un mécanisme d'un niveau de cognition beaucoup plus élevé. L'agent doit avoir conscience des autres agents, mais comme il n'y a toujours pas de contact direct, il doit inférer sur ces actions. L'agent doit alors disposer d'un modèle des autres agents (grâce aux cartes cognitives floues [Parenthoen et al. 01] par exemple).

Le second type d'interaction est le plus fréquent dans les applications des systèmes

[8] Cours sur les système multi-agents: `http://www.emse.fr/~boissier/enseignement/sma`

multi-agents. Il s'agit d'envois de messages dont le type est connu à l'avance par les agents. Ces messages peuvent être diffusés, ou être destinés à un agent en particulier. Ce genre de communication est considéré comme la capacité minimum de communication d'un agent. C'est ce qui doit au minimum différencier un agent d'un objet. La réception d'un message par un agent s'apparente à un appel de méthode sur un objet. C'est l'agent lui-même qui traite le message et décide (ou non) d'invoquer une méthode. C'est ce qui fait en partie son autonomie.

Enfin les actes de langage constituent la forme d'interaction la plus évoluée. Pour pouvoir faire interagir des agents hétérogènes dans une même application, il a fallu définir un protocole de communication. De plus, dans certaines applications, l'utilisateur doit interagir avec les agents de la manière la plus naturelle possible. Il a donc fallu définir un type de communication de plus haut niveau et plus évolué. Les recherches se sont alors portées sur la définition de concepts de communication d'un point de vue social. Ces recherches ont donné lieu aux actes de langage. L'un des premiers protocoles défini est KQML[9] [Labrou 96]. Ce langage, à base de performatives, a servi de fondement au protocole FIPA[10] [O'Brien et al. 88] pour la communication entre systèmes hétérogènes. Ces protocoles s'inspirent de la communication humaine, mais il a été montré que ce type de communication est largement plus complexe. D'autres types de performative doivent être rajoutés [Chaignaud et al. 01]. Il a également été montré l'importance de la communication non verbale [Salembier 00], ainsi que celle de l'environnement et de l'état d'esprit des interlocuteurs [Durand et al. 99].

2.2.1.4 Organisation

La définition du terme organisation dépend du type de systèmes multi-agents utilisé. Dans tous les cas l'organisation peut être vue comme la topologie d'un groupe permettant de spécialiser les agents en fonction de leurs compétences. Une organisation peut également contenir des liens organisationnels et des règles sociales qui contraignent le comportement des agents.

L'étude d'une organisation en système multi-agents est composée de :

- ▷ la structure organisationnelle qui décrit l'organisation en terme de rôles, tâches, responsabilités,
- ▷ l'entité organisationnelle qui instancie cette structure.

L'entité organisationnelle est l'affectation des agents aux rôles décrits dans la structure organisationnelle.

[9] *Knowledge Query and Manipulation Language*

[10] *Foundation for Intelligent Physical Agents* `http://www.fipa.org`

Une organisation peut être :

- ▷ Explicite : les rôles sont définis *a priori*. Avant la simulation, le concepteur connaît la structure organisationnelle.
- ▷ Emergente : les rôles sont observés *a posteriori*. Aucune structure organisationnelle n'est connue du concepteur. C'est en observant les interactions entre les agents qu'il se fait une représentation de l'organisation.
- ▷ Statique : la structure ou les entités organisationnelles sont fixées *a priori* et ne peuvent évoluer durant la simulation.
- ▷ Dynamique : la structure ou les entités organisationnelles peuvent évoluer au cours de l'exécution.

Dans notre approche pour l'utilisation d'un système multi-agents pour un environnement virtuel de formation, ll nous semble intéressant d'introduire ici deux modèles d'organisation, le modèle Agent/Groupe/Rôle et le modèle MOISE.

Modèle Agent/Groupe/Rôle

Ferber et Gutknecht [Gutknecht et al. 98, Gutknecht et al. 99] définissent le modèle Agent/Groupe/Rôle. Comme le montre la figure 2.12, il s'articule autour de trois concepts :

- ▷ Agent : Dans ce modèle, aucune contrainte n'est posée sur l'architecture d'un agent. C'est simplement une entité qui joue un rôle dans une organisation.
- ▷ Groupe : Il permet "d'identifier par regroupement un ensemble d'agents". Un agent peut être membre d'un ou plusieurs groupes.
- ▷ Rôle : Il s'agit de la représentation d'une fonction ou d'un service du groupe. L'agent jouant un rôle doit alors fournir les compétences nécessaires pour jouer ce rôle. Ces compétences sont représentées par des contraintes dans une *fonction de rôle*. Un agent peut jouer un ou plusieurs rôles et un rôle peut être joué par un ou plusieurs agents.

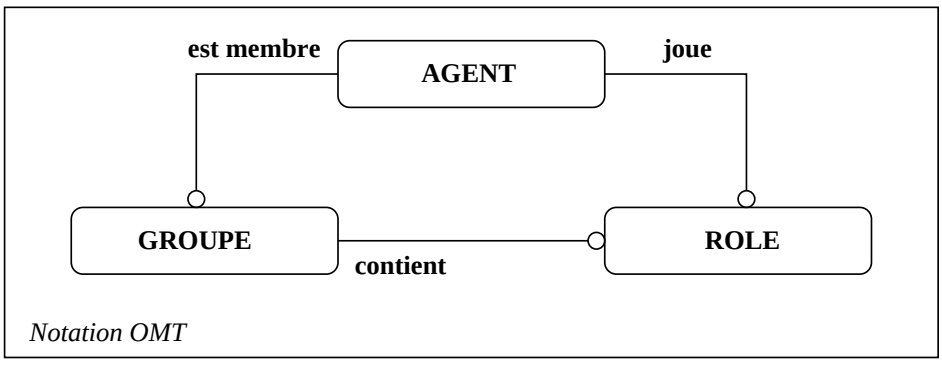


Figure 2.12 : Le modèle Agent/Groupe/Rôle (d'après [Gutknecht et al. 98]).

D'après les auteurs, ce modèle a été implémenté dans la plate-forme MadKit [Gutknecht 00]. Le concepteur d'une application sous MadKit s'attend alors à devoir

manipuler les classes `Agent`, `Groupe` et `Role`. Ce n'est pas le cas, puisque la notion de groupe se réduit à un attribut de type chaîne de caractère et le concepteur ne peut bénéficier d'une classe `Role`. Ainsi, des travaux [Muller et al. 01] sont menés pour réifier la notion d'organisation sur la plate-forme MadKit ; le modèle Agent/Groupe/Rôle constitue donc plutôt un patron de conception.

Modèle MOISE

Dans le modèle MOISE[11] [Hannoun et al. 99], la description d'une organisation sert à réduire la complexité de la tâche globale en explicitant le rôle de chaque agent et les liens qui les relient.

Ainsi ce modèle est fondé sur trois points :

- ▷ Rôle : C'est le comportement autorisé d'un agent. Un rôle est défini par un ensemble de missions (M). Une mission est un quadruplet <G,P,A,R> qui définit l'ensemble des buts (G), des plans (P), des actions (A) et des ressources (R) que l'agent peut considérer en jouant le rôle. Un plan est défini par un but et un graphe orienté d'actions ou de sous-buts. Certaines actions sont des *méta-actions organisationnelles*. Ce sont des actions de gestion des rôles, groupes et liens.
- ▷ Lien organisationnel : Les liens organisationnels servent à contraindre les communications et actions mutuelles entre les agents. Un lien est défini sur deux rôles par un type (communication, autorité ou accointance), un contexte d'utilisation (les missions des deux rôles pour lesquels ce lien est valide) et un ensemble de contraintes (protocole de communication...)
- ▷ Groupe : Un groupe est alors défini par un ensemble de rôles et un sous-ensemble de missions de ces rôles. Le but de la formalisation de cette notion de groupe est de faciliter l'instantiation des entités organisationnelles.

Comme le montre les auteurs, un tel modèle est bien adapté aux modèles de type BDI.

2.2.1.5 Utilisateur

L'intégration de l'utilisateur dans un système multi-agents est un problème qui a souvent été évoqué par la communauté des systèmes multi-agents, mais qui n'a pas souvent été résolu. La plupart du temps, l'utilisateur intervient dans le système pour fixer les paramètres, observer un possible comportement émergent et analyser les résultats, mais il est rarement intégré dans l'environnement comme élément à part entière de la simulation.

L'utilisateur interagit avec le système multi-agents via une Interface Homme/Machine (IHM). Ainsi des travaux ont été menés sur la réalisation de telles IHM

[11] *Model of Organization for multI-agent SystEms*

[Brown et al. 98]. Le but de ces travaux est de fournir une interface intelligente, c'est-à-dire qui s'adapte aux actions de l'utilisateur. Dans ces travaux, le système essaie de prévoir les actions futures de l'utilisateur pour lui proposer la bonne interface. Pour cela, le système multi-agents se construit un modèle de l'utilisateur à partir de l'observation de ses actions. Dans de telles applications, l'utilisateur est considéré comme un maître dans le système.

La plupart des travaux menés pour intégrer l'utilisateur dans le système multi-agents se sont développés autour des services sur Internet et l'informatique mobile [Burg 00]. Le principe est alors de fixer des paramètres représentant l'utilisateur dans un agent autonome qui le représente dans ses différentes tâches sur Internet. ABROSE, est un exemple de ces systèmes multi-agents [Gleizes et al. 00]. Le but de ABROSE est d'aider l'utilisateur au courtage sur Internet. Comme le montre la figure 2.13, le système est composé de deux domaines. Le domaine du courtage et le domaine utilisateur. Le domaine du courtage contient essentiellement les agents de transactions (TA) représentant chacun un utilisateur. Cet agent connaît la requête de l'utilisateur et dispose de croyances sur les autres agents de transactions. Il peut ainsi collaborer avec les agents pertinents. Le domaine de l'utilisateur fournit des utilitaires qui permettent à l'utilisateur de se connecter au système, de formuler sa requête et de réaliser les transactions. Dans de tels systèmes multi-agents, l'utilisateur n'est intégré à l'environnement que via l'agent qui le représente, il ne participe pas directement à l'évolution de cet environnement.

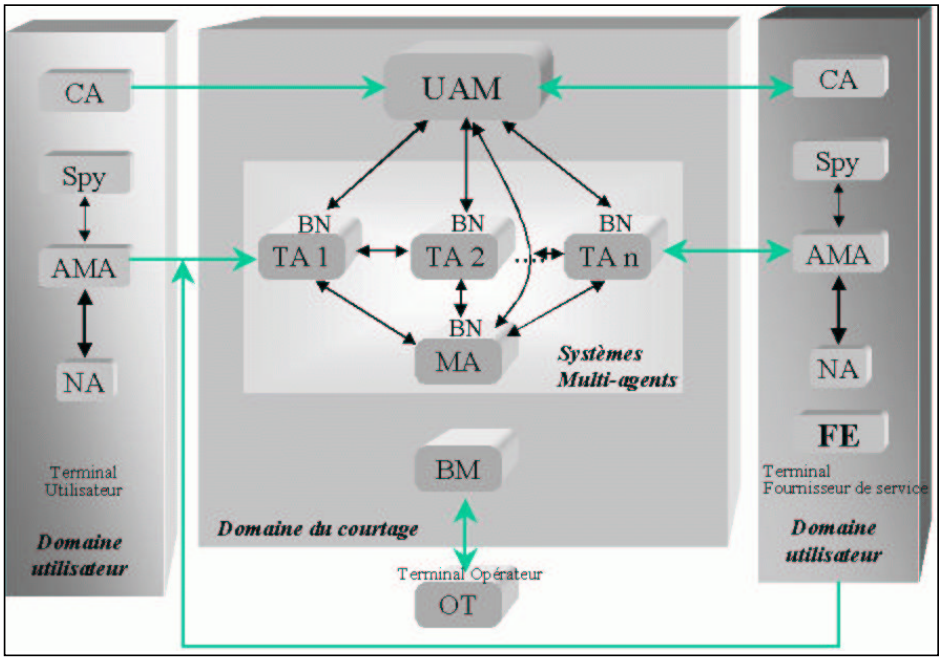


Figure 2.13 : Architecture de ABROSE (d'après [Gleizes et al. 00])

Dans les applications de réalité virtuelle, l'environnement est, dans certains travaux, peuplé d'entités autonomes (agents) avec qui l'utilisateur doit interagir.

Un agent peut alors représenter l'utilisateur lorsque celui-ci quitte l'environnement [Chicoisne 00]. L'agent autonome continue les tâches initiées par l'utilisateur.

Dans les travaux de RICHARD, l'environnement est constitué uniquement d'agents et forme un monde INVIWO [Richard 01]. Ainsi l'environnement d'un agent est l'ensemble des autres agents. Elle propose également une architecture d'agent fondée sur les capacités de perception, de décision et d'action. Le processus de décision s'articule autour de modules de comportement pouvant être ajoutés, supprimés ou modifiés dynamiquement. Tous les modules reçoivent les informations perçues et chacun d'entre eux propose aux effecteurs la réalisation d'une action. A chaque effecteur est associé un arbitre qui gère les conflits. Cette architecture est destinée à la réalisation de comportements réactifs plutôt que cognitifs. Tous les agents de l'environnement sont conçus à l'aide de ce modèle. L'environnement est donc un système multi-agents homogène. Les agents interagissent par le biais de stimuli. L'échange de stimuli se fait par communication asynchrone, un agent peut communiquer avec un ou tous les agents. Le *multicast* n'est pas possible car il n'y a pas de structure organisationnelle. Cette lacune s'avère très pénalisante lorsque les agents sont en grand nombre et que leur comportement est fortement lié les uns aux autres car cela multiplie les communications et augmente les temps de calcul.

L'utilisateur est immergé dans l'environnement par le biais de son avatar. Son rôle est de percevoir et agir dans l'environnement pour l'utilisateur qui conserve tout de même la phase de décision. Le rôle de l'avatar est également de représenter l'utilisateur dans l'environnement virtuel pour être perçu par les autres utilisateurs ou agents autonomes. RICHARD propose également de modéliser l'avatar par un agent autonome. Ainsi il est possible de contraindre ou au contraire d'assister l'utilisateur dans l'environnement. Cette caractéristique est conforme à la notion de simulation participative définie par TISSEAU dans laquelle l'utilisateur est *"un modèle comme un autre"*. La communication entre l'humain et son avatar se fait également par l'échange de stimuli. L'utilisateur émet et reçoit les stimuli grâce à son interface qui peut être textuelle ou au contraire implémentée par des périphériques de réalité virtuelle (figure 2.14).

2.2.2 Systèmes multi-agents et formation

L'intelligence artificielle est utilisée dans la conception des tuteurs intelligents. Elle permet de représenter les connaissances et les raisonnements de l'étudiant, de l'expert et du tuteur. Cette modélisation est principalement réalisée par des règles d'inférences. Mais des travaux plus récents proposent d'utiliser les systèmes multi-agents pour représenter les connaissances. Les systèmes multi-agents sont également utilisés pour simuler les différentes stratégies pédagogiques ou les différents composants de l'outil pédagogique.

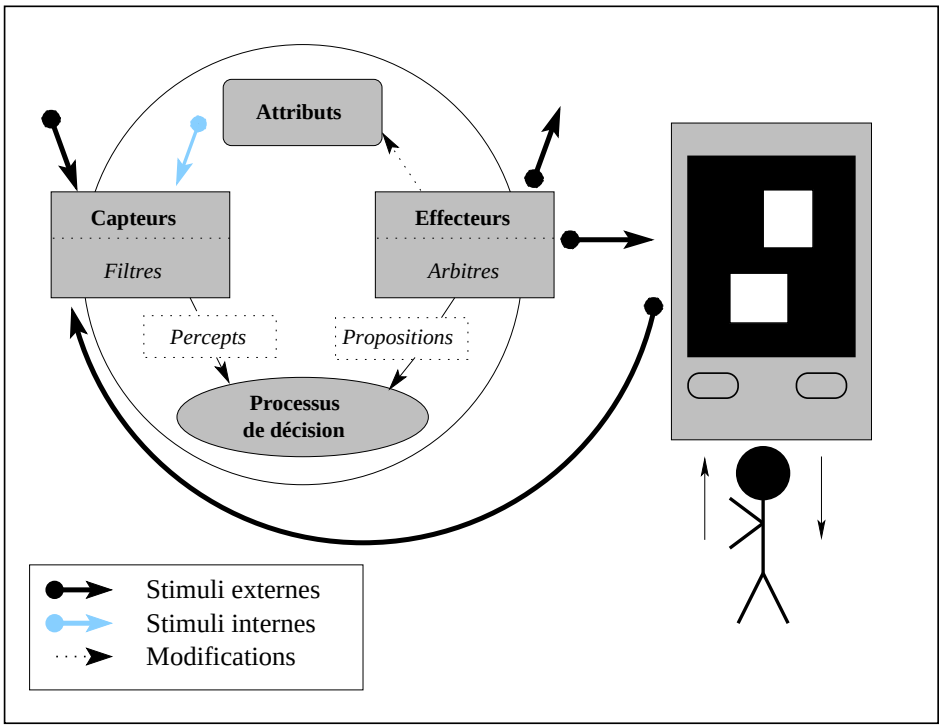


Figure 2.14 : Avatar InViWo (d'après [Richard 01]).

2.2.2.1 L'intelligence artificielle pour la formation

Le but des systèmes tuteurs intelligents (ou ITS *Intelligent Tutoring System*) est de permettre à un étudiant d'apprendre seul. Un tel système vise à rendre un retour immédiat sur les actions de l'étudiant. En général, les ITS alternent entre les phases où, l'étudiant mène l'exercice et le tuteur valide les actions, et les phases où le tuteur réalise les tâches et les explique à l'étudiant.

Différentes stratégies de formation [Aïmeur et al. 00] ont été développées pour améliorer l'utilisation des ITS. Les principales sont les stratégies coopératives ou au contraire perturbatrices. Dans les stratégies coopératives, le professeur est vu comme un partenaire qui va échanger, contrôler et construire des connaissances avec l'apprenant. Il y a trois types d'agent. Le professeur (joué par le système), le *compagnon* (également simulé) et l'apprenant (l'utilisateur humain). Le compagnon et l'apprenant coopèrent pour la réalisation de la tâche, échangent des idées sur le problème et partagent les mêmes buts. A l'inverse, dans les stratégies perturbatrices, le but est de perturber l'apprenant en lui proposant des solutions qui peuvent être erronées. Cela force l'apprenant à évaluer la confiance qu'il a dans les solutions qu'il propose. Dans ce type d'apprentissage, il y a également trois types d'agent. Un professeur (simulé) qui gère l'évolution de la session et évalue les résultats, un apprenant (humain) et un étudiant perturbateur (simulé). Le perturbateur et l'apprenant humain doivent réaliser la même tâche. Le perturbateur propose des solutions parfois fausses et parfois vraies. Cela

provoque un débat entre l'apprenant et le perturbateur. Ils doivent alors argumenter leurs solutions. Le professeur se construit un modèle de l'utilisateur pour connaître les points non maîtrisés par l'apprenant.

Le premier outil de ce type, longtemps considéré comme la référence du domaine est SCHOLAR [Carbonell 70]. C'est un outil permettant d'apprendre la géographie, mais il a également été appliqué à d'autres domaines. Il possède une base de faits sur laquelle il pose des questions à l'étudiant. L'étudiant peut répondre par une question pour plus d'informations sur le sujet. Le dialogue se fait en langage naturel. Des outils plus récents tels que ANDES [Conati et al. 97] et ADIS [Warendorf et al. 97] proposent une meilleure interaction entre l'étudiant et le tuteur mais ils sont toujours fondés sur le même fonctionnement.

Actuellement, l'outil le plus abouti dans ce genre d'application est STEVE [Rickel et al. 99]. STEVE est un tuteur intelligent basé sur le moteur d'intelligence artificielle SOAR [Newell 90]. STEVE a pour objectif d'aider à la formation de tâches procédurales. Pour cela, il sait montrer la procédure, l'expliquer en répondant aux questions de l'apprenant et surtout observer et valider (ou non) les actions de l'apprenant. STEVE a été appliqué à la formation de maintenance de moteurs de bateau. Une vue de cet ITS est donnée figure 2.15. Une version de STEVE nommée ADELE [Shaw et al. 99], a également été développée pour intégrer des capacités d'apprentissage distant. Dans cette version, l'interaction Apprenant/Tuteur est simplifiée pour permettre la connexion par Internet de l'utilisateur.

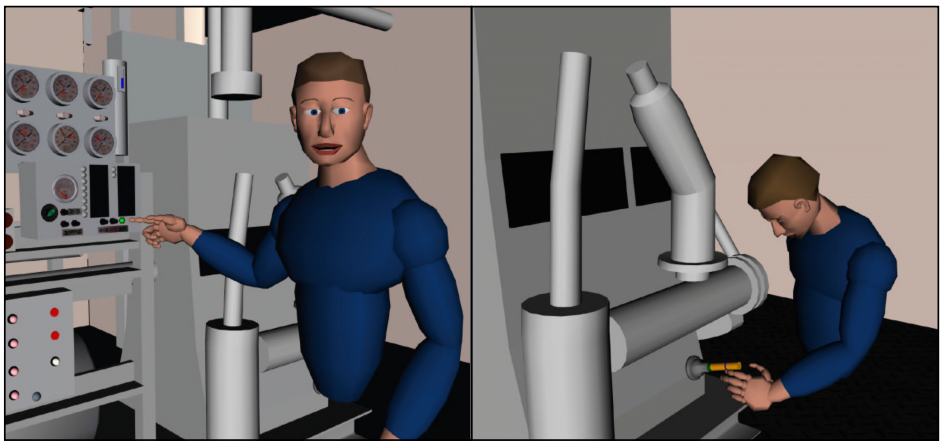


Figure 2.15 : STEVE explique et réalise une procédure (d'après [Rickel et al. 99]).

A la différence des techniques classiques où la solution à chaque problème est stockée, l'intelligence artificielle dans un ITS permet de représenter le raisonnement humain (expert) qui conduit à la solution du problème posé [Dillenbourg 94]. L'expert simulé est alors capable d'exprimer son mode de raisonnement, ce qui est une fonctionnalité indispensable dans un ITS. Reste encore à ce que les différentes étapes du raisonnement du système aient une représentation compréhensible par l'apprenant (problème des réseaux de neurones par exemple dont le raisonnement intermédiaire

n'est pas compréhensible).

Ainsi, une technique efficace est de modéliser l'expertise du domaine par un ensemble de règles de production. Si la condition est vérifiée, alors la règle est activée. La solution à un problème est donc le chemin défini par l'ensemble des règles activées. Si pour chaque règle est associé un commentaire de type texte, un discours cohérent peut alors être généré à partir de cette séquence de règles pour formuler une explication à l'apprenant.

Pour ce type de modélisation, les outils les plus fréquemment utilisés sont PROLOG et SOAR. Un autre problème est de choisir entre plusieurs règles contradictoires et activées. Une première solution est que le moteur d'inférence intègre des processus cognitifs humains qui permettent de faire le choix (c'est le cas par exemple de SOAR). Une autre solution est d'intégrer des méta-règles qui gèrent l'activation des règles. Cette deuxième solution permet, dans une application à vocation pédagogique, de personnaliser le raisonnement en fonction de l'apprenant. En effet, une méta-règle peut exprimer que, si l'apprenant est un débutant, telles ou telles règles ne sont pas utilisables.

Dans STEVE, l'expertise du domaine est modélisée par un ensemble de tâches. Un exemple est donné figure 2.16. Une tâche est définie par un ensemble d'actions atomiques ou de sous-tâches (`press-function-test` ...), de l'explicitation des effets des actions (`achieves test-mode`...) ainsi que leur ordonnancement (`press-function-test before check-alarm-light`...). Un texte est associé à chaque tâche et action ce qui permet à STEVE de répondre aux questions de l'apprenant. Ainsi si l'apprenant pose la question "Pourquoi ?", STEVE peut lui répondre ; si après cette réponse l'étudiant lui repose la même question, STEVE donne la réponse qui est associé à la tâche de niveau immédiatement supérieur. STEVE mémorise en permanence l'état du monde, ainsi il peut suivre l'évolution de l'étudiant pour le conseiller ("Que dois-je faire après ?"), lui expliquer ("Pourquoi ?"), ou exécuter lui-même l'action.

```
Task: functional-test

   Steps: press-function-test, check-alarm-light, extinguish-alarm

   Causal Links:
      press-function-test achieves test-mode for check-alarm-light
      check-alarm-light achieves know-wether-alarm-functional for end-task
      extinguish-alarm achieves alarm-off for end-task

   Ordering constraints:
      press-function-test before check-alarm-light
      check-alarm-light before extinguish-alarm
```

Figure 2.16 : Exemple de tâche dans STEVE (d'après [Rickel et al. 99]).

L'explication et l'évaluation sont des concepts importants dans un ITS. En effet, un ITS doit savoir ce que l'étudiant a effectivement compris pour évaluer ses performances et expliquer.

Pour pouvoir expliquer, le système doit comprendre exactement la dynamique de l'environnement et les interactions entre ses composants pour détecter la source de l'erreur. Cette tâche peut devenir très difficile dans le cas de systèmes complexes [Zouaq et al. 00]. Ces explications doivent également être personalisées. Pour cela l'ITS doit avoir des connaissances sur l'apprenant. Ces connaissances sont appelées *modèle de l'étudiant*.

Mentoniezh [Py 96] est un ITS développé pour la formation en géométrie. Le professeur (humain) prépare un exercice et l'écrit sous forme textuelle pour l'apprenant et dans un langage de données pour le tuteur (artificiel). Ce système est composé d'un tuteur qui valide et aide l'étudiant durant la résolution du problème ainsi que d'un résolveur de problème. Pour un problème donné, le résolveur calcule l'ensemble des solutions (certaines de ces solutions peuvent être écartées par le professeur car jugées non pertinentes dans un cursus pédagogique). La résolution du problème est alors considérée comme une succession d'étapes partant des données du problème et débouchant sur la conclusion. Une étape est l'application d'un théorème aux données et hypothèses, ce qui produit de nouvelles données. Une solution au problème est donc vue comme la réalisation d'un plan. Pour que l'ITS aide et conseille l'apprenant, il faut qu'il reconnaisse la solution choisie, c'est-à-dire qu'il reconnaisse le plan choisi par l'apprenant à partir de quelques actions. En plus de ces données dynamiques, le tuteur utilise également des connaissances dites statiques puisqu'elles sont extraites des traces des précédentes utilisations du système. A partir de ces connaissances, le tuteur essaie de reconnaître le type d'erreur pour adapter sa réponse.

La reconnaissance du type d'erreur est également un problème. Dans Mentoniezh, des expérimentations ont permis de générer un ensemble de traces qui ont été analysées pour décrire des catégories d'erreurs. Une autre solution utilisée est de simuler des réponses en omettant ou en faussant des règles et ensuite de comparer la solution proposée par l'apprenant aux fausses solutions simulées. Cela permet d'identifier l'erreur de l'élève.

Dans ce domaine, il est possible d'extraire quelques principes génériques mais les solutions adoptées restent profondément applicatives. De plus, de telles solutions nécessitent d'avoir une connaissance globale du système (environnement + modèle de l'apprenant). Cela a pour résultat d'augmenter les temps de calcul et de les rendre inadéquats aux environnements fortement dynamiques.

2.2.2.2 Les systèmes multi-agents pour la formation

Un EIAO est un environnement composé de plusieurs éléments, chacun d'entre eux regroupant différents concepts. Plusieurs auteurs ont donc proposé une implémentation fondée sur la méthodologie des systèmes multi-agents. Les raisons avancées sont souvent leurs capacités d'incrémentalité (ajout ou modification d'une fonctionnalité sans modifier l'ensemble de l'application) et d'interfaçage entre modules hétérogènes.

Interactions entre modules pédagogiques

Dans beaucoup d'EIAO, les agents représentent les différents modules (expert, tuteur, interface...) C'est le cas, par exemple, dans [Fenton-Kerr et al. 98], qui décrit différents types d'agent, chacun ayant un rôle particulier. Ainsi le système est articulé autour d'agents jouant les rôles de :

- source d'informations (cours, sons, images...),
- tuteur,
- expert,
- *motivateur*, dont le rôle est d'encourager l'apprenant, attirer son attention sur des éléments de l'environnement,
- médiateur qui permet de faciliter les interactions entre les différents intervenants.

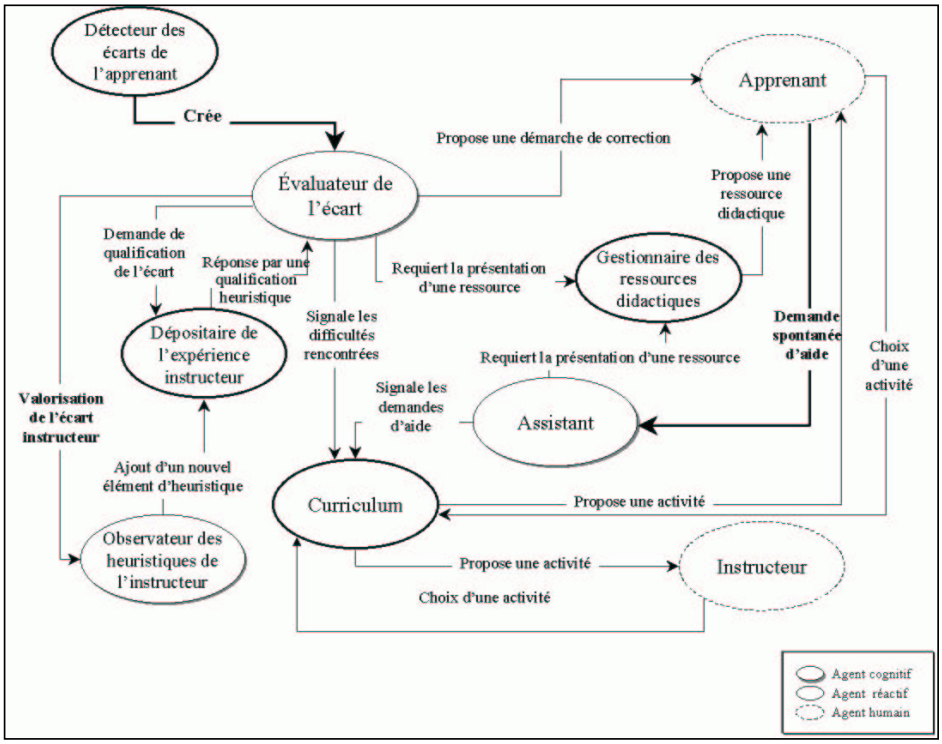


Figure 2.17 : L'architecture CMOS (d'après [Richard 99]).

Dans CMOS [Richard 99, Gouardères et al. 99], le principe est sensiblement le même. L'architecture du système multi-agents est donné en figure 2.17. Dans cette architecture trois types d'agent coexistent, des agents humains (apprenant et instructeur), des agents cognitifs (qui sont permanents et dont il n'existe qu'un seul individu par classe) et des agents réactifs (créés par les agents cognitifs et dont il existe *0* ou *n* individus par classe). Les agents cognitifs sont appelés ainsi car, d'une part ils ont la capacité d'apprendre, c'est-à-dire que leur base de connaissances évolue, et d'autre part leurs connaissances portent sur des capacités cognitives et pédagogiques du système et de l'apprenant (interactions, erreurs ...).

Ainsi, un agent *Détecteur d'écart* surveille les interactions entre l'apprenant et son environnement. S'il détecte un écart avec la base de savoir-faire (méthodes pré-enregistrées par l'instructeur), il crée un agent réactif *Évaluateur d'écart* qui, en relation avec l'agent cognitif *Dépositaire de l'expérience instructeur* (qui contient les instructions pédagogiques de l'instructeur), évalue la gravité de cet écart. Cette évaluation est transmise sous forme de note à l'agent cognitif *Curriculum* qui gère la progression de l'étudiant et l'organisation des sessions d'apprentissage. En fonction de cette évaluation, l'agent *Évaluateur d'écart* prend contact avec l'agent cognitif *Gestionnaire des ressources didactiques* qui présente à l'apprenant la ressource adaptée à son erreur.

Dans ce type d'utilisation, le professeur n'intervient qu'avant la formation pour définir les exercices et les préférences pédagogiques, et après la formation pour analyser le curriculum. C'est le principe même des ITS qui partent de l'hypothèse de la distribution géographique des intervenants. Des travaux ont alors été menés pour distribuer ces applications sur le réseau Internet [Capuano et al. 00] et pour permettre au professeur d'avoir des informations sur les activités des apprenants et agir pour les aider lorsque les tuteurs n'y arrivent pas [George et al. 99]. La difficulté réside alors dans la conception d'un agent qui va permettre au professeur de bien percevoir la situation chez l'apprenant.

Emergence de comportements pédagogiques

Un autre apport des systèmes multi-agents est de pouvoir gérer simultanément différentes stratégies de formation. Ainsi les travaux de Frasson se fondent sur le constat qu'une stratégie de formation peut ne pas être adaptée pour un apprenant dans un domaine spécifique [Frasson et al. 97]. Un ITS doit alors intégrer plusieurs stratégies de formation. Dans l'architecture proposée par l'auteur (figure 2.18), un agent (*Session manager*) choisit la stratégie adaptée et active l'agent correspondant en fonction du domaine et du modèle de l'étudiant. Cette sélection est dynamique. La stratégie globale de formation émerge donc de la collaboration entre les agents pédagogiques.

C'est ce principe d'émergence de la fonction pédagogique à partir des interactions entre les composants d'un EIAO qui est implémentée dans Baghera [Pesty et al. 01]. La société d'agents est composée de cinq types d'agent. Trois agents sont instanciés pour chaque étudiant connecté, un compagnon (qui gère la base d'exercices, l'interface et les communications avec les autres), un médiateur (qui choisit l'agent expert à qui envoyer la solution proposée par l'étudiant) et un tuteur (qui assure le suivi pédagogique pour un apprentissage autonome). Deux agents sont créés pour chaque professeur connecté, un compagnon (qui a le même rôle que le compagnon de l'élève) et un assistant (qui met

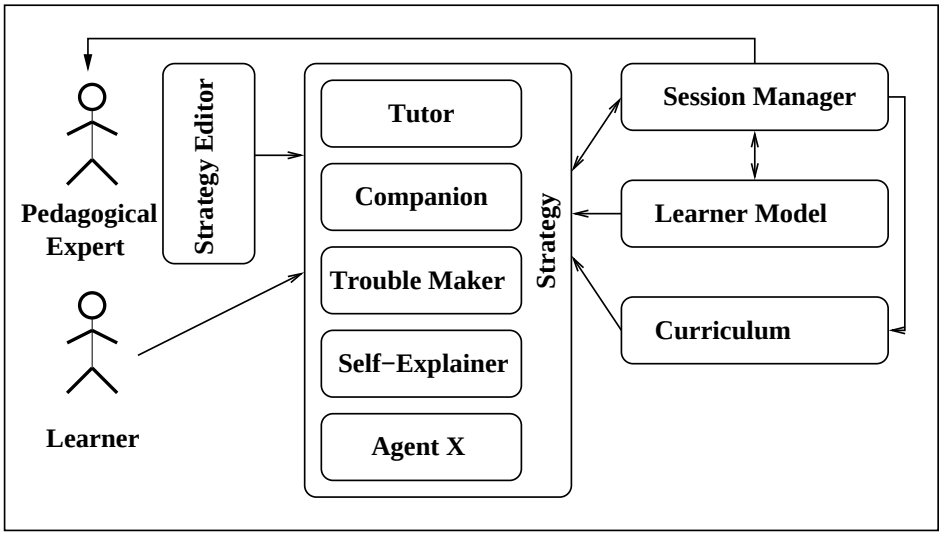


Figure 2.18 : ITS multi-stratégique (d'après [Frasson et al. 97]).

à disposition les exercices du professeur aux autres agents). Un agent est composé :

- ▷ d'une base de connaissances sur son état interne et sur les autres agents,
- ▷ d'actions qui sont en fait des envois de messages,
- ▷ de plans, séquencement d'actions (ce qui définit un protocole d'interaction),
- ▷ et d'un module de raisonnement qui choisit le plan adapté en fonction des buts.

Modèle de l'étudiant

L'ensemble de ces travaux se base sur un modèle de l'étudiant pour adapter l'intervention de leurs composants. Des travaux ont donc également été menés pour construire le modèle de l'étudiant par un système multi-agents. En effet, l'objectif de la construction d'un tel modèle est de reconnaître le raisonnement de l'étudiant à partir de l'analyse de ses interactions avec l'environnement.

C'est cette approche que défend [Leman et al. 96]. Dans ces travaux, le système tente de reconnaître les connaissances acquises par l'apprenant. Ces connaissances sont un sous-ensemble de celles de l'expert (connues à l'avance). L'ensemble de ces connaissances sont représentées par des agents (1 connaissance = 1 agent). Chaque agent dispose d'accointances (relation entre les connaissances de même niveau) et de croyances sur :

- ▷ le degré de maîtrise de l'apprenant,
- ▷ le degré d'importance de la solution,
- ▷ et le degré d'importance de l'étape de la solution

Un raisonnement est la relation entre plusieurs connaissances et est représenté par une société d'agents. L'ensemble des raisonnements, donné *a priori* par l'expert, forme la solution d'un problème. La solution à un problème est un agencement de

raisonnements dont chacun représente une étape de la solution. Une solution est alors un graphe hiérarchique composé de raisonnements et dont les feuilles sont les agents représentants les connaissances. Un problème acceptant plusieurs solutions est représenté par plusieurs sous-graphes. Le système modélisé par un ensemble d'agents peut alors suivre plusieurs raisonnements en parallèle. A chaque action de l'apprenant dans l'environnement sont associées des connaissances. Les croyances des agents ainsi activées sont alors renforcées et propagées aux accointances. Lorsqu'un apprenant devient expert, ses raisonnements deviennent d'un niveau supérieur dans la hiérarchie, les feuilles de l'arbre se transforment donc en raisonnement de plus en plus haut niveau.

Le travail de Lourdeaux porte en partie sur la description d'une méthodologie pour la conception d'environnements virtuels pour la formation [Lourdeaux 01]. Un des points clé de cette méthodologie est la description des primitives virtuelles comportementales (éléments d'actions nécessaires à l'utilisateur dans l'environnement virtuel) et leurs liens avec les périphériques de réalité virtuelle. Mais la partie importante de ses travaux porte sur la conception d'une aide pédagogique nommée HAL[12]. L'architecture générale de HAL est présentée figure 2.19.

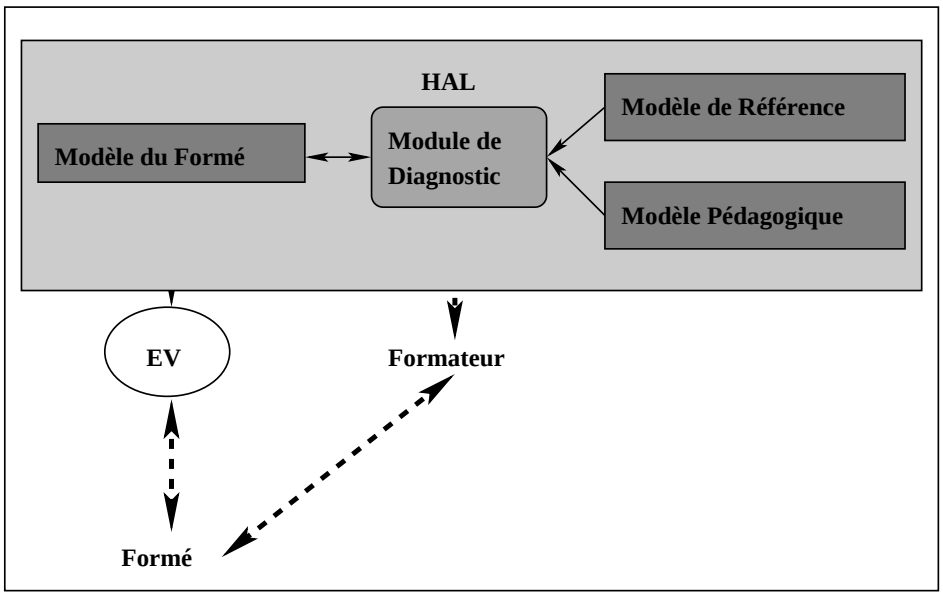


Figure 2.19 : Architecture de HAL (d'après [Lourdeaux 01]).

Cette aide s'articule autour des modèles de l'étudiant, de référence et des stratégies pédagogiques. Le cas d'étude choisi par l'auteur est un travail procédural, le modèle de référence est alors représenté par un ensemble de plans de tâches hiérarchiques et linéaires. Le modèle du formé est composé d'une partie statique liée aux caractéristiques individuelles de l'apprenant (niveau, profil ...) et d'une partie dynamique, construite au cours de l'exercice et fondée sur la construction de l'historique des actions et la représentation des connaissances de l'apprenant. Le modèle pédagogique représente les

[12] *Help Agent for Learning*

paramètres sur le type d'intervention, les méthodes pédagogiques ... Il est basé sur un critère de performance attribué à chaque tâche. Pour implémenter ces modèles ainsi qu'effectuer leur calcul en cours de simulation, l'auteur a choisi de représenter chaque tâche par ce qu'il nomme un agent cognitif. Un agent peut alors être :

- ▷ *activé*, s'il s'agit de la tâche que l'apprenant doit réaliser d'après le plan défini dans le modèle de référence. Un agent est activé par la tâche de niveau supérieur.
- ▷ *déclenché* lorsque l'apprenant réalise la tâche représentée par l'agent,
- ▷ *valide* si l'agent est activé et déclenché. Si l'agent est non valide, c'est-à-dire que l'apprenant n'effectue pas la tâche qu'il devrait faire, le critère de performance de la tâche est alors décrémenté.

En fonction de la valeur du critère de performance, une méthode pédagogique est réalisée. La reconnaissance des tâches réalisées par l'apprenant se fonde sur l'observation des primitives comportementales virtuelles.

2.3 Conclusion

Dans ce chapitre, nous avons explicité le concept de réalité virtuelle. Pour cela, nous avons rappelé qu'il s'agit d'une discipline à la croisée de l'informatique graphique, l'animation, la téléopération... L'approche que nous adoptons est celle de Tisseau qui considère un environnement virtuel comme un ensemble d'entités autonomes en interaction, disposant pour cela de capacités de perception, de décision et d'action. Dans cet environnement, l'utilisateur est représenté par son avatar, un modèle comme un autre. Cette approche trouve un support idéal dans les systèmes multi-agents. L'important est alors la description du comportement local des agents. Ce comportement peut être très réactif ou au contraire rationnel et faire appel aux techniques de l'intelligence artificielle classique. L'environnement est alors uniquement constitué d'agents hétérogènes. L'environnement d'un agent n'est que l'ensemble des autres agents avec lesquels il interagit de manière plus ou moins sociale. L'enjeu reste alors d'organiser ces interactions pour simuler le comportement du système réel (travail en équipe, négociation ...) et permettre le calcul en temps réel de l'évolution de l'environnement.

La réalité virtuelle et les systèmes multi-agents ont déjà été utilisés dans la réalisation d'environnements virtuels de formation. La réalité virtuelle a surtout servi à la réalisation de *micro-mondes* ou d'environnements d'apprentissage distant. Cela permet le rendu réaliste d'un environnement (réel ou imaginaire), l'interaction naturelle et l'immersion de plusieurs utilisateurs géographiquement distribués. Les systèmes multi-agents ont également servi aux outils de formation et cela essentiellement pour la réalisation de tuteurs intelligents. Il s'agit alors de simuler le tuteur autonome, les raisonnements de l'apprenant ou les différents modules pédagogiques. Il n'existe, par contre, aucune application fondée sur ces deux concepts. Certes, des outils tels que STEVE ou HAL sont des tuteurs fondés sur des techniques de l'intelligence artificielle ou des systèmes multi-agents et évoluent avec l'apprenant dans un univers virtuel. Mais

dans cet environnement, seul le tuteur est autonome ; l'environnement est contrôlé de manière globale, ce qui n'offre pas les avantages de la réalité virtuelle tel que décrite par Tisseau.

Notre travail porte donc sur la réalisation d'un tel environnement virtuel de formation composé d'agents autonomes en interaction. Dans la suite de ce rapport, nous présentons la réalisation de ce système multi-agents. Contrairement à Richard, notre environnement est constitué d'agents autonomes hétérogènes. Ils simulent le comportement d'entités formant l'environnement physique ou un travail collaboratif. Pour simuler correctement l'évolution globale de l'environnement, les interactions entre les agents ont besoin d'être *organisées*. L'aspect important de notre travail est donc la modélisation du comportement des agents et la définition d'un modèle d'organisation.

Chapitre 3

Le modèle : MASCARET

En introduction, nous avons exposé notre problématique de conception d'environnements virtuels pour la formation. Nous nous appuyons sur le principe d'entités autonomes en interaction pour simuler l'environnement virtuel et concevons alors cet environnement comme un système multi-agents. Ce système multi-agents est hétérogène car il fait interagir des agents de différentes natures (réactifs ou rationnels) et ouvert car il fait participer des utilisateurs (apprenants ou formateurs). Le chapitre précédent nous a permis de clarifier les concepts de réalité virtuelle et de système multi-agents, fondateurs de cette approche. Les travaux que nous y avons exposés montrent l'intérêt de la conception d'environnements virtuels par l'autonomisation des entités le peuplant.

Notre problématique est la formation professionnelle au travail en équipe et procédural dans un environnement physique. Dans le cadre de cette problématique, il faut simuler, de manière *réaliste*, l'environnement physique ainsi que le comportement *collaboratif* et *adaptatif* des membres de l'environnement social. L'évolution de ces environnements est obtenue par la simulation des comportements locaux des agents autonomes, mais également par leurs interactions. Nous proposons le modèle MASCARET comme l'utilisation des systèmes multi-agents pour simuler les environnements *réalistes*, *collaboratifs* et *adaptatifs* pour l'entraînement. Il s'agit d'un modèle permettant de structurer les interactions entre les agents et de fournir aux agents les capacités d'évoluer dans ce contexte.

Dans la première section nous montrons que, dans notre problématique, il existe deux types d'environnement et donc deux types d'interaction. Certaines entités participent aux deux environnements ce qui nous conduit à définir un modèle générique d'organisation pouvant supporter les deux types d'interaction. Les sections suivantes montrent comment ce modèle est dérivé pour décrire l'environnement physique et l'environnement social. Nous y expliquons alors le comportement réactif, rationnel et social des agents ainsi que l'intégration des utilisateurs.

3.1 Le modèle d'organisation

Ce modèle générique d'organisation a pour vocation de servir de cadre aux interactions entre les agents qui simulent l'environnement physique et social. Dans notre problématique, il existe deux types d'environnement, l'environnement physique et l'environnement social. La figure 3.1 adaptée de [Querrec et al. 01c] montre les relations entre les différents types d'agents. Les agents réactifs, les avatars (représentant des utilisateurs) et les agents rationnels participent à l'environnement physique par des interactions réactives exclusivement. Seuls les deux derniers types d'agent participent à l'environnement social et leurs interactions sont collaboratives.

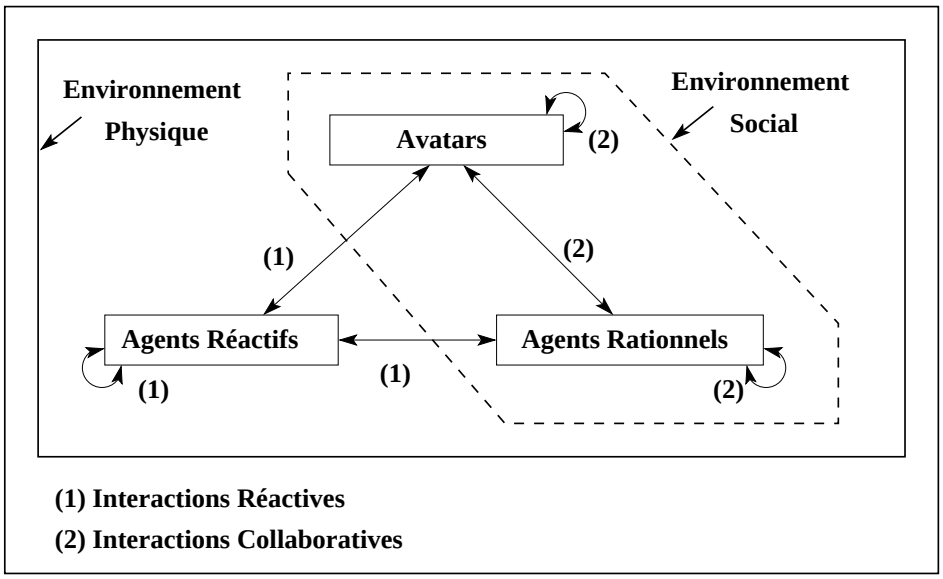


Figure 3.1 : Types d'interactions entre les agents.

Un agent, quel que soit son type, doit prendre en compte l'existence des autres agents dans son comportement. Pour que l'agent reste autonome et que l'environnement reste dynamique, il faut que les agents puissent se faire une représentation des autres et des interactions qu'ils peuvent avoir entre eux. Nous réifions ce concept par la définition d'un modèle d'organisation suffisamment générique pour pouvoir représenter la structure de base des organisations décrivant l'environnement physique et social. Le caractère générique du modèle permet d'intégrer les utilisateurs à la fois dans l'environnement physique et dans l'environnement social.

3.1.1 Agents, organisations et rôles

Dans le chapitre 2 nous avons étudié des modèles génériques d'organisation tel que les modèles AGR et MOISE. Ce dernier est trop lié à son type d'application, l'exécution

d'un plan d'actions, et donc ne peut pas répondre complètement à notre problématique car il ne nous permettrait pas de représenter efficacement l'environnement physique. Le modèle AALAADIN est plutôt un *pattern* de conception qu'un modèle implémentable, ce qui nécessiterait d'en fournir une sémantique plus précise. Le schéma de la figure 2.12 (page 33) fait apparaître deux relations indépendantes entre d'une part `Agent` et `Group` et d'autre part entre `Role` et `Group`. La notion de `Group` a alors une sémantique pauvre (ensemble d'instances) et les relations qui unissent les trois concepts n'explicitent pas les contraintes d'intégrité qui existent nécessairement entre les instances de ces relations. En effet, le modèle n'explicite pas le fait qu'un agent ne peut jouer qu'un rôle défini pour le groupe auquel il appartient.

Nous proposons donc le modèle organisationnel tel qu'il est présenté par le diagramme de classes de la figure 3.2 (en notation UML [Booch et al. 99]). Ce modèle s'articule autour de quatre concepts : l'organisation, le rôle, l'agent et l'élément de comportement. Il s'agit d'un modèle générique (toutes les classes sont abstraites) que nous dérivons ensuite en deux modèles concrets correspondant aux deux types d'organisation constituant notre environnement virtuel de formation.

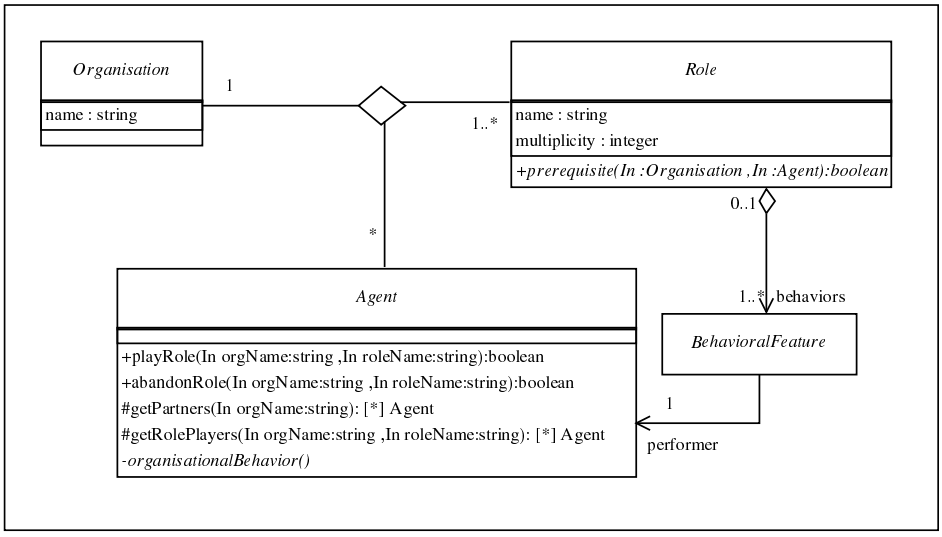


Figure 3.2 : Modèle générique d'organisation dans MASCARET.

L'`Organisation` est identifiée par un nom qui permet aux agents autonomes ou aux utilisateurs humains de l'identifier ; c'est le cadre des interactions entre les agents y participant. L'organisation sert également de facteur structurant ; elle permet aux agents de savoir quels sont leurs partenaires et quels rôles ils jouent.

Les rôles (`Role`) désignent les différentes responsabilités des agents dans l'organisation. Ces responsabilités sont définies par un ensemble d'éléments de comportement (`BehavioralFeature`[1]) que doit adopter l'agent jouant le rôle. Un tel agent doit alors présenter des capacités lui permettant de prendre la responsabilité de ces

[1] Ce terme est emprunté au méta-modèle d'UML [Rumbaugh et al. 99], il a ici la même signification.

comportements, c'est pourquoi un rôle impose des pré-requis (`prerequisite`). Il s'agit ici de vérifier que l'agent possède les attributs et les méthodes qui sont invoqués par les éléments de comportement. Lorsqu'un même type de rôle (même ensemble de comportements et même pré-requis) est défini dans plusieurs organisations, il est instancié autant de fois. Une instance de rôle est donc contextualisée à une organisation. Notons qu'un agent peut jouer des rôles dans plusieurs organisations, qu'il s'agisse du même type de rôle ou non. Il peut exister par contre, une contrainte organisationnelle qui exprime une limite sur le nombre d'agents pouvant jouer un rôle dans une organisation. Cette contrainte est représentée par l'attribut `multiplicity` dans le rôle.

À tout instant, les agents peuvent jouer ou abandonner un rôle (`playRole, abandonRole`). Lorsqu'ils prennent la responsabilité de réaliser un rôle de l'organisation, les agents doivent prendre en compte dans leur comportement, l'existence des autres membres. Cela se fait grâce à leur comportement organisationnel (`organisationalBehavior`), qui peut leur servir par exemple pour gérer l'arrivée de nouveaux agents. Les méthodes `getPartners` et `getRolePlayers` assurent alors le maintien de la connaissance de l'agent sur ses accointances.

Pour distinguer un type particulier d'organisation d'une instance de ce type d'organisation, nous employons respectivement les termes de structures organisationnelles et d'entités organisationnelles. Une structure organisationnelle est définie par un type d'organisation (reflètant les types de relation entre agents), par des types de rôle et des types de comportement organisationnel. Ces derniers, et en particulier la prise et l'abandon de rôle ainsi que la mise à jour de connaissances organisationnelles, peuvent être très différents d'un domaine à l'autre et c'est pourquoi la méthode correspondante `organisationalBehavior` est ici abstraite. Nous considérons à ce titre que l'existence d'un agent `GroupManager` tel qu'imposé dans le modèle AALAADIN n'est qu'un cas particulier. Une entité organisationnelle correspond, quant à elle, à une instanciation de la relation ternaire qui lie une organisation, des rôles et des agents ; les notions d'organisation et de rôle sont alors réifiées et sont considérées comme des connaissances exploitables par les agents.

3.1.2 Caractéristiques des organisations

Dans le chapitre précédent (section 2.2.1.4, page 32), nous avons discuté des critères permettant de comparer des organisations. Une organisation peut être statique ou dynamique, et émergente ou explicite. Les organisations que nous définissons dans le cadre de MASCARET, sont explicites et dynamiques.

Organisation émergente ou explicite ?

Dans le cadre d'un environnement virtuel de formation, les structures organisationnelles sont explicites. Cela signifie que les différents types d'organisation sont définis par le concepteur et sont, directement ou indirectement, perceptibles par l'utilisateur. En effet, ce dernier sait, d'une part, quels sont les types de phénomène physique qui sont simulés et dont il doit tenir compte et d'autre part comment est structuré son environnement social, en particulier quel est son rôle et celui de chaque intervenant.

Organisation statique ou dynamique ?

Si les organisations sont explicites, elles n'en sont pas moins dynamiques. En effet, si dans la pratique, les structures organisationnelles sont statiques, c'est-à-dire qu'elles sont décrites *a priori*, rien n'empêche de les modifier au cours du temps. En effet, le modèle que nous avons défini, dans la mesure où il est réifié, autorise la création et la modification des structures organisationnelles (nouveaux types d'organisation, nouveaux rôles...)

Par contre, les entités organisationnelles, si elles peuvent pour partie être définies *a priori*, sont fortement dynamiques. Les organisations sont créées et supprimées au cours du temps et les agents décident au cours de l'exécution de jouer un rôle ou de l'abandonner dans une organisation. Ils mettent également à jour de manière dynamique leurs accointances dans l'organisation. Ceci est particulièrement vrai pour les organisations supportant les interactions physiques.

3.2 L'environnement physique

Dans un environnement virtuel de formation, l'environnement physique des utilisateurs (apprenants et formateurs) doit être réaliste, interactif et réagir "en temps réel". Les apprenants ont besoin d'une restitution qui leur permette de comprendre une situation et de se représenter son évolution logique (*j'observe ceci, donc telle action devrait provoquer cela...*). Selon l'approche constructiviste, les apprenants acquièrent des savoirs en observant et en agissant sur leur environnement (ils expérimentent donc le modèle). Les formateurs doivent pouvoir définir, pour un exercice particulier, quels phénomènes il souhaite prendre en compte (ceci permet d'en graduer la difficulté), décision qui peut dépendre de l'apprenant. Par exemple, dans une phase de *monitoring*, le formateur peut inhiber les interactions mécaniques ou le phénomène de toxicité de l'air pour permettre à un apprenant sapeur-pompier de se déplacer librement dans la zone du sinistre et donc mieux comprendre la situation. Il est également indispensable que le formateur puisse intervenir sur le déroulement d'une simulation comme s'il s'agissait, du point de vue de l'apprenant, d'une évolution normale de la situation (par exemple modifier la force et la direction du vent dans un exercice de confinement

d'un nuage de gaz). Les actions des utilisateurs induisent donc des changements dans le modèle au cours de sa simulation et, précisément parce qu'il y a un humain dans la boucle, les répercutions doivent être immédiates ; il est donc hors de question de reconstuire un modèle *ex nihilo* et de recalculer une nouvelle situation à chaque interaction avec l'utilisateur. Le calcul du modèle doit donc être incrémental, chaque pas de résolution correspondant à un état observable du système doit s'exécuter dans un temps très court (le temps réel de l'utilisateur) ; il doit de plus supporter des modifications locales en cours d'exécution.

Notre objectif est d'offrir dans MASCARET les mécanismes de base permettant de modéliser les phénomènes physiques sous forme d'un système multi-agents et que leur simulation s'intègre au modèle général de l'environnement virtuel de formation. Nous nous sommes attachés à définir la nature des interactions entre les agents, et donc leur organisation, ainsi que le comportement des agents. Pour cela nous dérivons le modèle générique d'organisation décrit dans la section précédente en définissant la notion d'interaction réactive, le modèle d'organisation qui en découle (un réseau d'interactions) et le comportement réactif des agents. Ce modèle peut ensuite être spécialisé pour la simulation de tel ou tel phénomène physique. L'exemple simplifié de la simulation de la propagation d'un nuage de gaz est donné à titre d'illustration.

3.2.1 Principes de la modélisation des phénomènes physiques

Classiquement, la simulation des phénomènes physiques n'est pas interactive et nécessite plusieurs heures de calculs et est donc inadaptée aux contraintes de la réalité virtuelle [Herrmann et al. 98]. Les modèles présentant les meilleurs caractéristiques de ce point de vue sont les modèles de simulation à événements discrets pour lesquels l'espace est partitionné en cellules dont les calculs sont réalisés de manière asynchrone. Dans ces modèles, la partition de l'espace est fixe, ce qui est adapté à la simulation de phénomènes dans des milieux continus mais ne l'est pas si l'on veut suivre l'évolution d'une entité particulière, surtout si elle est mobile. Les modèles dits individu-centrés sont une réponse à ce problème. Leur principe est d'observer un phénomène au niveau global en ayant modélisé le comportement autonome de chaque entité constituante ainsi que ses interactions avec les autres. A ce titre, il s'agit de modèle à vocation explicative et pas seulement prédictive. La sémantique de ces modèles rejoint sur de nombreux points celle des systèmes multi-agents composés d'agents réactifs situés, même si les aspects collaboratifs et proactifs des comportements ne sont pas toujours considérés. Les modèles individu-centrés sont utilisés en écologie et en dynamique des populations [DeAngelis et al. 79] ou pour la simulation de mouvements de foules ou de trafic routier ainsi que pour la modélisation de phénomènes physiques dans des milieux granulaires. Par exemple [Breton et al. 99] propose un modèle multi-agents pour la simulation d'un réseau de forces dans un milieu granulaire.

Les phénomènes physiques devant être simulés sont de natures différentes. Si l'on souhaite, pour des raisons pédagogiques ou tout simplement de performance, inhiber un phénomène, il est nécessaire de se placer à un niveau de modélisation compatible avec un certain découplage entre ces phénomènes, ce qui en toute rigueur est contraire aux lois des modèles physiques qui reconnaissent des invariants. Par exemple, on considère des phénomènes tels que :

- ▷ la toxicité : on modélise ainsi le résultat d'un phénomène chimique en considérant qu'une entité source de toxicité est en interaction avec une entité cible ; il peut s'agir de l'effet d'une "macro-particule" de gaz toxique sur un humain ;
- ▷ les échanges thermiques : on considère alors des sources d'énergie thermique en interaction avec des cibles thermiquement inertes ou non ;
- ▷ les échanges de masses qui correspondent à des transferts de charges discrètes ou à des flux de matière ;
- ▷ les mises en mouvement qui font par exemple que, dans certaines conditions (régime laminaire...) un nuage de gaz est porté par le vent atmosphérique et que l'on considère alors qu'une macro-particule de gaz a une vitesse égale à celle de l'entité "vent atmosphérique local" (la masse d'air pouvant être discrétisée selon un maillage éventuellement dynamique) ;
- ▷ les collisions entre entités qui modifient les variables d'état "vitesse" en fonction de leurs caractéristiques et de la configuration de l'interaction (collision effective ou non, orientation des normales) ;
- ▷ d'autres phénomènes tels que les interactions entre composants électriques ou les perturbations électromagnétiques seraient prises en compte selon les mêmes principes.

Lorsqu'on simule le comportement d'une entité telle qu'une lance à incendie on est amené à prendre en compte plusieurs de ces phénomènes : le fait d'arroser un élément avec une lance à incendie induit un échange thermique (refroidissement), un échange de masse (l'eau projetée alourdit la structure qui la reçoit) et éventuellement une mise en mouvement (le choc du jet d'eau sur la cible peut modifier sa trajectoire).

Quatre principes ont prévalu à la conception de notre modèle.

1. Les couplages entre phénomènes sont supportés par des variables d'état des agents qui peuvent être prises en compte dans des interactions de différentes natures (par exemple la masse) ; il est également possible d'assurer la conservation d'un invariant en couplant les variables d'état.
2. Il est nécessaire de découpler les phénomènes pour en ignorer certains (par exemple considérer l'effet thermique d'une lance à incendie sur une réservoir de gaz et ignorer le transfert de masse) ; ceci se traduit par la prise en compte (ou non) des interactions entre les agents et non par une modification du comportement des agents.
3. Bien que toutes les interactions soient potentiellement réciproques, dans bon nombre de cas, l'effet d'une entité est prépondérant. Par exemple, on considérera l'effet d'un mur d'eau sur la force et la direction des macro-particules formant un nuage de gaz et pas l'infime déviation du mur d'eau en interaction avec le nuage

de gaz.

4. Intrinsèquement et pour respecter les contraintes de la réalité virtuelle, les modèles sont nécessairement approchés. Du point de vue pédagogique, il est souvent suffisant de respecter les liens de causalité (un feu s'éteint parce qu'on l'arrose) et les ordres de grandeur (voilà l'étendue approximative d'un nuage de gaz compte tenu de la présence de bâtiments et des conditions atmosphériques locales).

3.2.2 Interactions entre agents réactifs

Pour qu'il y ait interaction entre deux agents, il faut d'une part que les agents perçoivent la situation d'interaction et qu'ils agissent en conséquence. Considérant que les interactions ont généralement une direction privilégiée, on peut donc distinguer les rôles source et cible d'interaction, ce qui amène à définir les responsabilités de chacun d'eux. Considérons le cas de l'embrasement d'un réservoir de carburant situé à proximité d'une flamme ouverte. L'interaction est un échange thermique, la flamme jouant le rôle de source de chaleur et le réservoir celui de cible. Le comportement de ce dernier est affecté par l'interaction : il peut s'enflammer si la flamme est très chaude et proche.

L'interaction a lieu du fait du comportement de l'agent source alors que c'est l'état interne de la cible qui est modifié par cette interaction. On suppose que les agents rendent publique un sous-ensemble de leurs variables d'état (celles qui ont un sens dans le calcul de l'interaction) et que leurs sémantiques sont partagées. Ainsi, pour décider si le réservoir va s'enflammer, il faut connaître la température de la flamme.

La situation d'interaction peut *a priori* être détectée par les deux agents. Pour des raisons de performance, il est souhaitable que cette détection qui est très coûteuse en calcul ne soit faite que par un seul. Deux solutions sont également possibles pour la modification de l'état de l'agent cible : soit l'agent source envoi un message à la cible pour l'informer de son état et ce dernier applique le changement d'état lorsqu'il traite le message, soit l'agent cible a l'initiative et c'est lui qui demande à l'agent source son état pour calculer le sien. Lorsque l'on veut inhiber un phénomène cela revient à dire que l'interaction potentielle n'a pas lieu, c'est-à-dire que l'état de l'agent cible n'est pas modifié. C'est pour cette raison que nous avons retenu la deuxième solution : c'est le comportement de l'agent cible qui le conduit à percevoir l'état des agents sources avec lesquels il est en interaction et à calculer son nouvel état. Le rôle de l'agent source est de mettre à disposition des autres des informations pertinentes sur son état pour le calcul de l'interaction.

De nombreuses situations d'interaction peuvent être qualifiées par une grandeur que nous avons appelée *force d'interaction*. Par exemple, l'énergie thermique reçue par une cible est inversement proportionnelle à la distance qui la sépare de la source. L'agent

cible a donc besoin d'évaluer cette force d'interaction. Dans le cas où la force est fonction d'une distance, il faut donc que l'agent cible dispose d'un capteur lui permettant de l'estimer, ce qui est classique pour des agents situés. La figure 3.3 résume le principe du modèle. Les agents A_i et A_j interagissent avec l'agent A_k, ce dernier étant cible des interactions. L'agent A_k réagit donc à la présence de ces agents sources, ce qui provoque un changement de son état interne E_k qui est fonction des états perçus de A_i et A_j (E_i et E_j) et des forces d'interactions entre A_i et A_k (ω_{ik}) et entre A_j et A_k (ω_{jk}).

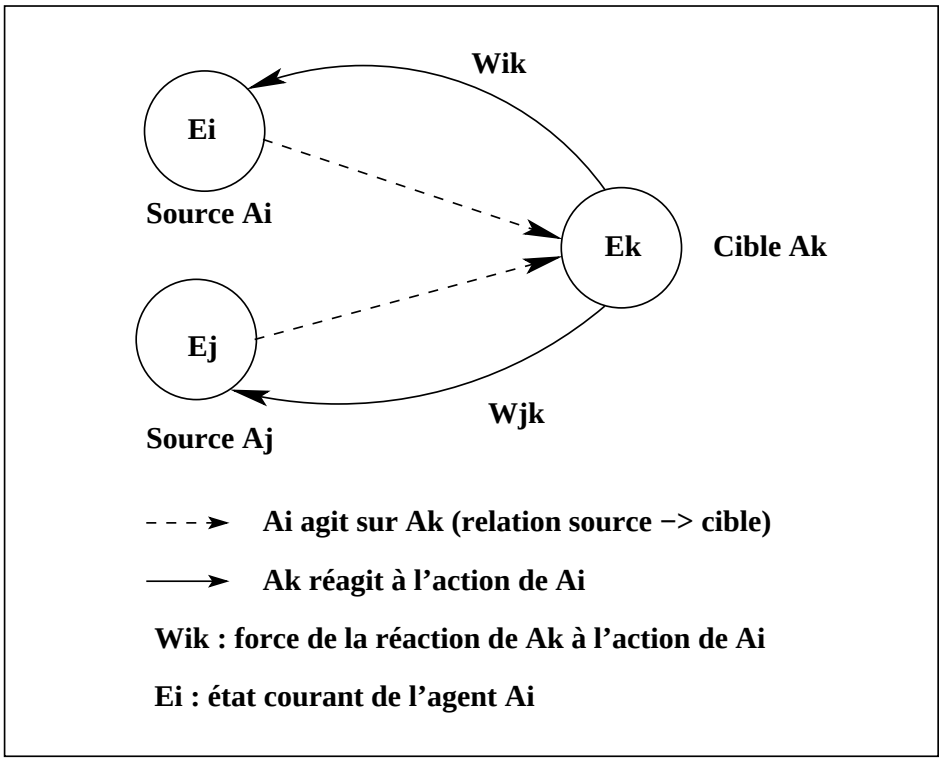


Figure 3.3 : Interactions entre agents réactifs.

3.2.3 Organisation des interactions réactives

Une des limites pratiques des modèles individu-centrés est que chaque agent est susceptible de percevoir tous les autres et que donc la complexité de l'algorithme est en $O(n^2)$, ce qui n'est pas concevable lorsque l'environnement est peuplé d'un grand nombre d'agents, ce qui est notre cas. Il est donc impératif de définir des règles d'agencement des interactions, c'est-à-dire une organisation et nous nous appuyons sur notre modèle générique shématisé en figure 3.2.

Les différents phénomènes physiques pouvant être activés ou inhibés, les interactions qui les simulent ont lieu dans des contextes séparés, donc des entités organisationnelles différentes. Il est par ailleurs concevable que la simulation d'un même phénomène

soit supportée par plusieurs entités organisationnelles ; il s'agit d'un choix applicatif qui se justifie si l'on peut identifier des sous-mondes indépendants (c'est classique en réalité virtuelle dans le calcul des collisions).

Nous faisons l'hypothèse que les interactions ont lieu entre des agents situés et qu'elles ont généralement une force d'autant plus grande que les agents cibles sont situés dans un voisinage de la source et que ce voisinage a une certaine stabilité : l'ensemble des voisins d'un agent évolue moins vite que l'état interne de cet agent, la force de l'interaction pouvant toutefois avoir une dynamique plus rapide. Notons que la topologie de ce voisinage est *a priori* quelconque, même si dans la pratique elle repose généralement sur une métrique euclidienne. Nous considérons qu'à tout instant un agent connaît l'ensemble de ses voisins et qu'il est capable d'évaluer la force de l'interaction qui le lie avec chacun d'eux. Les interactions ont donc lieu au travers d'un réseau de voisinage et nous définissons l'organisation de ces agents comme un réseau d'interactions, ce qu'illustre la figure 3.4. Pour pouvoir agir dans ce réseau d'interactions (`InteractionNet`), les agents doivent avoir des compétences organisationnelles pour le maintien de leur connaissance sur leur voisinage (accointances). Différents protocoles peuvent être envisagés en s'appuyant sur les méthodes `organisationalBehavior()`, `updateSources()` et `updateTargets()` de la classe `NestedAgent`. Fondamentalement, un réseau d'interactions est un graphe dont les nœuds sont les agents et les liens le support des interactions. Notons que la structure de graphe n'est qu'un support à la connaissance locale de chaque agent sur son voisinage et à son évaluation de la force d'interaction ; aucun algorithme global n'est appliqué sur le graphe par les agents. Les notions de source et de cible d'interaction sont réifiées par des classes dérivées de `Role` : `InteractionSource` et `InteractionTarget`. Le troisième type de rôle de notre modèle (`Recruiting`) correspond au maintien des connaissances organisationnelles des agents c'est-à-dire à l'identification des situations d'interaction. Si un agent joue les rôles source et recruteur alors il doit percevoir les agents sur lesquels il agit (dans le contexte d'un réseau d'interactions) et mettre à jour ses connaissances sur ces liens d'interaction (`InteractionLink`) sortants ; s'il joue le rôle cible et recruteur alors il doit percevoir les agents qui agissent sur lui (liens d'interaction entrants). Si un agent ne joue pas le rôle de recruteur, on suppose qu'il existe dans le réseau d'interactions un agent qui assure pour lui son intégration dans l'entité organisationnelle.

3.2.4 Comportement réactif

Nous avons défini un rôle comme un ensemble d'éléments de comportement. Un agent réactif peut jouer différents rôles dans plusieurs réseaux d'interactions. Son comportement est donc la résultante de l'exécution des différents éléments de comportement qui le compose. Nous décrivons ici l'architecture interne d'un agent réactif et le fonctionnement de ses éléments de comportement.

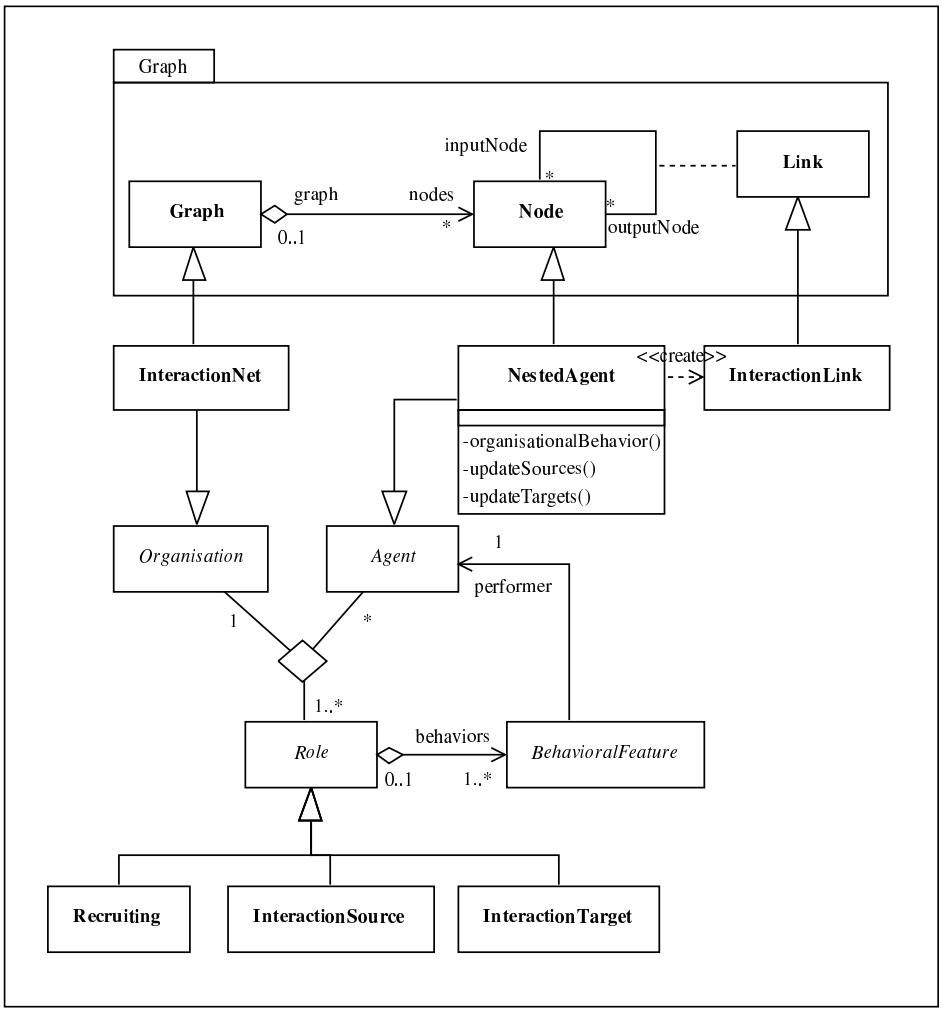


Figure 3.4 : Organisation en réseau dans MASCARET.

Architecture interne

L'architecture interne des agents réactifs répond aux objectifs de couplage/découplage des phénomènes physiques : d'une part il est nécessaire de découpler la simulation des phénomènes physiques pour pouvoir éventuellement en inhiber certains et d'autre part il existe des intersections ou des couplages entre les variables d'état qui servent au calcul des états internes des agents. La figure 3.5 donne un exemple d'architecture d'un agent et la figure 3.6 le modèle de classes de notre architecture.

Dans cet exemple, on considère un agent réactif intervenant dans deux phénomènes physiques et jouant donc un rôle dans chacun des deux `InteractionNet` : IN_1 et IN_2. Dans chaque réseau d'interactions, cela se traduit par une relation d'interaction avec d'autres agents au travers des rôles source et cible ; ces liens d'interaction ont comme support les éléments de comportement des rôles (instances de classes dérivées de `ReactiveComponent`). Le comportement de cet agent dépend aussi de son état interne

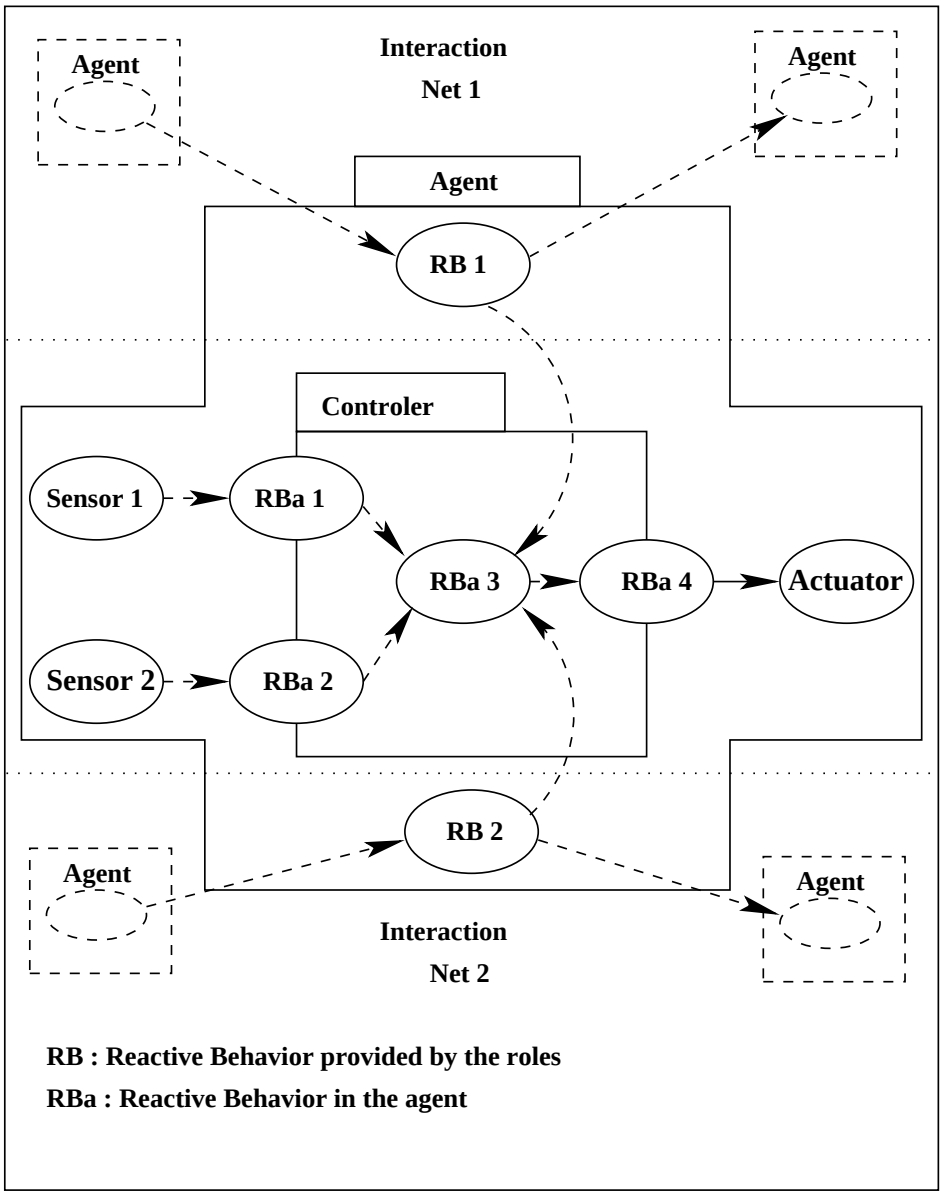


Figure 3.5 : Architecture interne d'un agent réactif.

et des informations qu'il perçoit sur son environnement, ce qui lui permet d'activer des actionneurs et donc d'agir sur lui-même et son environnement. Le maintien de son état interne est fonction de variables internes mais aussi du résultat sur l'agent des interactions avec les autres, donc de la valeur des variables d'état liés aux réseaux d'interactions. Chaque composant assure le calcul de la valeur d'un vecteur de variables d'état (attribut `state` de la classe `ReactiveComponent`). Les sous-classes `Sensor` et `Actuator` correspondent à des composants qui n'ont respectivement pas de lien en entrée et en sortie. Il s'agit d'éléments qui permettent à l'agent de percevoir son environnement (indépendement de toute interaction avec un autre agent) et de réaliser des actions (par exemple effectuer un changement de position en fonction d'une variable

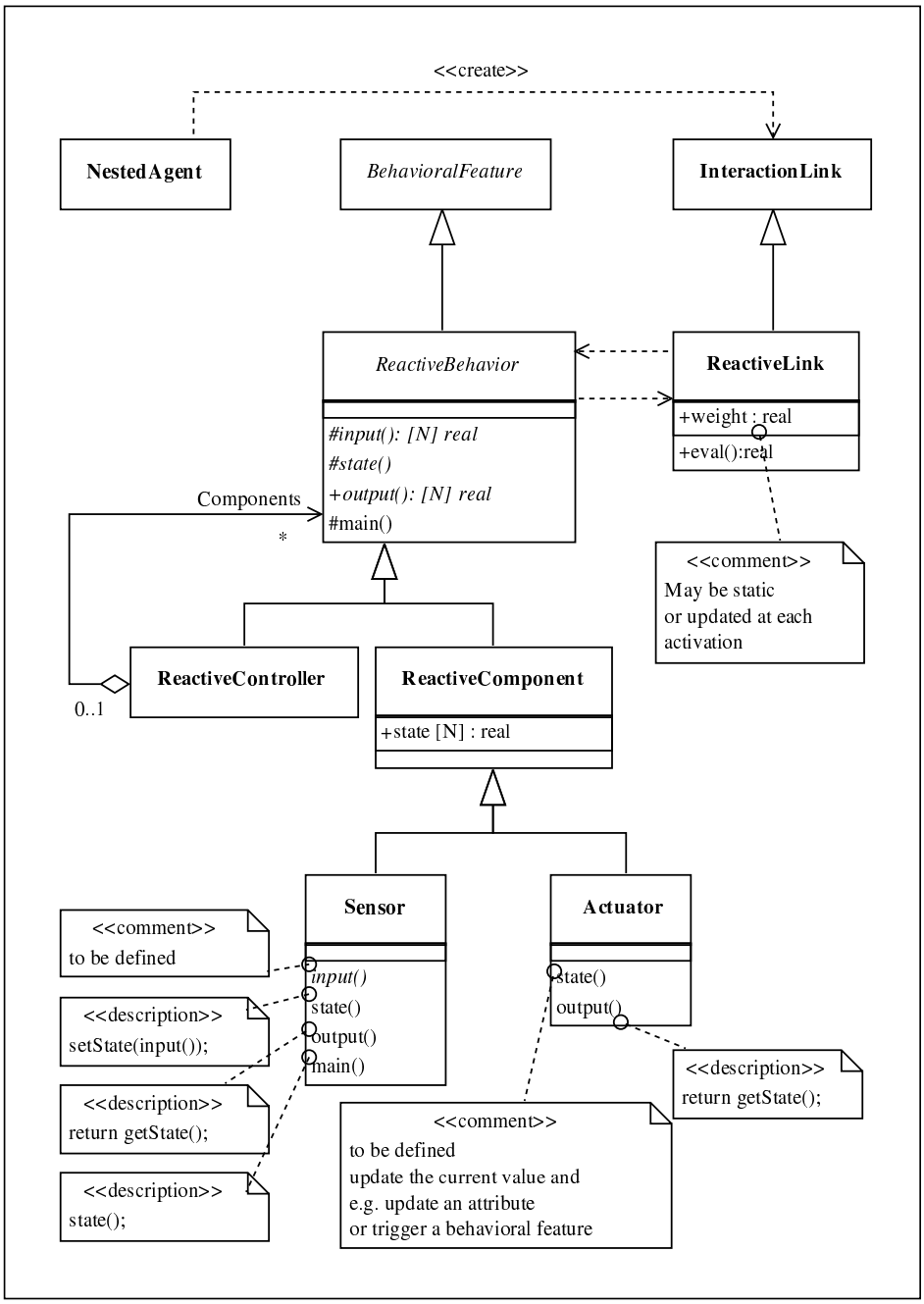


Figure 3.6 : Comportements réactifs dans MASCARET.

d'état "vitesse").

Les relations entre ces différents éléments de comportement s'expriment par des liens d'interaction réactive (classe `ReactiveLink`). Ce modèle est récursif : en effet, la structure qui lie les différents composants de l'agent a une architecture qui est similaire à celle qui décrit les interactions entre les agents. Pour des raisons de modularité, un niveau supplémentaire de récursion a été introduit. Il se peut que le

comportement réactif d'un agent nécessite de recourir à des composants plus complexes tel qu'un contrôleur flou [Reignier 94]. Pour cela nous avons introduit la notion de `ReactiveController` qui est un agrégat récursif de `ReactiveComponent` ou d'autres `ReactiveController`. La conception d'un tel agent réactif revient donc à instancier des éléments de comportement réactif de différents types et de les relier entre eux par des liens d'interaction réactive. La méthode `eval()` et l'attribut `weight` de la classe `ReactiveLink` servent à calculer la force d'interaction entre les composants. Prenons l'exemple d'une interaction dont l'intensité est inversement proportionnelle au carré de la distance qui sépare la source de la cible : `weight` représente la distance d (qui peut résulter d'une phase de perception et d'un précalcul) et `eval` calcule $1/d^2$.

Elément de comportement

Tous les élements de comportement sont activés de manière asynchrone (il s'agit d'objets actifs). Ce comportement élémentaire consiste à calculer un vecteur de variables d'état après évaluation des entrées du composant et à rendre public une information pertinente pour le calcul des interactions. Cela nous conduit à définir trois fonctions, soit pour l'agent cible A_j en interaction avec un ensemble $\mathcal{S}_j$ d'agents sources :

1. $I_j = \mathcal{F}_{i \in \mathcal{S}_j}(\omega_{ij}, O_i)$: valeurs d'entrée du composant calculées en fonction de ce que perçoit l'agent cible, ce qui dépend de l'information fournie en sortie par chaque agent source (O_i) et de la force ω_{ij} de l'interaction entre A_i et A_j.
2. $S_j = g(I_j, S'_j)$ où S'_j est l'état précédent de A_j : vecteur des variables d'état calculé en fonction des valeurs perçues en entrée et de l'état précédent de l'agent.
3. $O_j = h(S_j)$ vecteur des variables d'état que le composant rend accessibles aux autres ; il se calcule en fonction de l'état courant du composant.

3.2.5 Exemple : propagation d'un nuage de gaz

Prenons un exemple illustrant le fonctionnement décrit précédemment. L'objectif ici est d'expliquer l'interaction entre des particules de gaz formant un nuage, conséquence d'une fuite d'un réservoir. Pour lutter contre cette fuite, une solution qui s'offre aux sapeurs-pompiers est de placer une *"queue de paon"* sur la trajectoire du nuage de gaz. Cet outil est composé d'une lance à incendie qui projette de l'eau à forte pression vers une plaque métallique de la forme d'un demi-cercle. L'eau déviée par cette pièce forme alors un mur d'eau en forme de queue de paon. Ce mur est considéré comme un obstacle infranchissable pour les particules de gaz. La figure 3.7 illustre cette scène.

L'exemple est volontairement simplifié pour isoler quelques points de comportements intéressants. Ainsi dans le comportement de la queue de paon nous ne considérons que l'effet d'obstacle de l'eau et nous ne calculons pas l'effet de la vitesse des gouttes

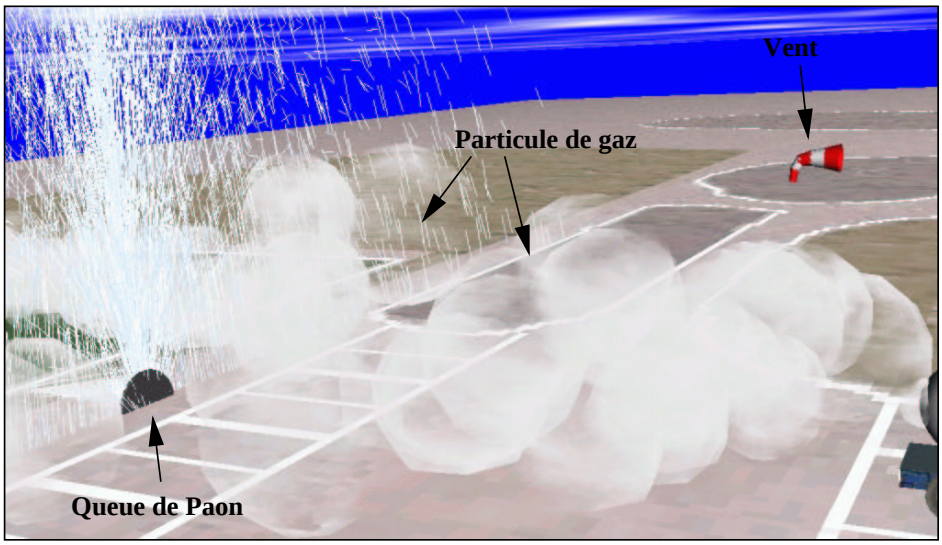


Figure 3.7 : Exemple d'interactions entre agents réactifs.

d'eau sur la vitesse des particules de gaz. Le modèle du choc est également simplifié et représenté par un coefficient d'absorption.

3.2.5.1 Les structures organisationnelles

Dans cet exemple il existe trois types d'entité : les particules de gaz, la queue de paon et le vent. La queue de paon intervient dans un réseau d'interactions symbolisant les collisions, le vent intervient dans un réseau symbolisant les déplacements dans l'atmosphère. Les particules de gaz interviennent dans les deux.

Il existe donc deux réseaux d'interactions dérivant de `InteractionNet`, le réseau des collisions et le réseau des mobiles. Chacun d'entre eux décrit les rôles source, cible et recruteur et rajoute respectivement les rôles `Collider` et `Mobile`. Ces nouveaux rôles décrivent chacun un élément de comportement de type comportement réactif `ReactiveBehavior` que nous expliquons plus loin.

Dans notre exemple, toutes les entités représentant les différentes composantes intervenant dans un phénomène physique sont des entités géoréférencées (`GeoEntity` figure 3.8). Ces entités héritent d'une classe d'entité générique (`ArEntity`) qui implémente l'interface `cinematic` et propose des capacités de position, orientation, vitesse et accélération. Une entité géoréférencée dispose également d'une masse volumique ainsi que d'un coefficient d'absorption de chocs et un rayon représentant la sphère englobante de l'entité. Par défaut une entité géoréférencée participe aux deux réseaux d'interactions et joue donc les rôles `Collider` et `Mobile`.

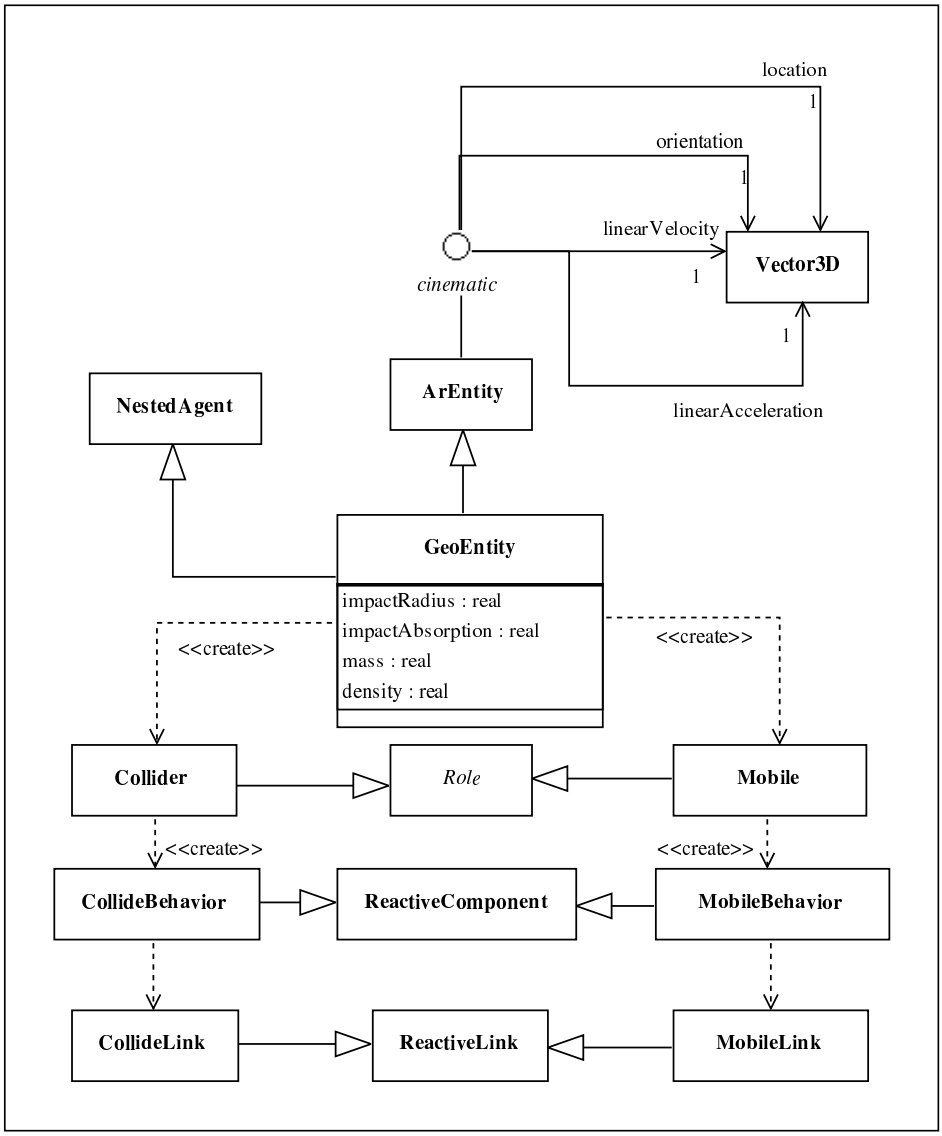


Figure 3.8 : Entités géoréférencées dans MASCARET.

3.2.5.2 Les comportements réactifs

Dans cet exemple, il y a deux comportements réactifs intéressants. Le comportement de `Mobile` et le comportement de `Collider`.

Mobilité

Le comportement de `Mobile` sert à calculer la vitesse d'une entité. Dans notre exemple, il s'agit de calculer la vitesse des macro-particules de gaz, l'agent source étant le vent

atmosphérique local, dont la vitesse est

$$\overrightarrow{Vv} = \overrightarrow{Vv_h} + \overrightarrow{Vv_v}$$

(composantes horizontale et verticale). On veut simuler le fait que la direction de propagation du nuage de gaz est grossièrement celle du vent, mais que, du fait de turbulences (dont on ne calcule pas les équations et que l'on suppose très faibles) la trajectoire des particules s'écarte un peu de la direction du vent. On veut également représenter qu'un gaz plus léger que l'air à tendance à s'élever alors que dans le cas contraire, il reste au sol.

Le comportement `Mobile` consiste à calculer la composante horizontale de la vitesse d'une particule de gaz (Vg_h) et sa composante verticale (Vg_v) en utilisant la formule approchée suivante :

$$\overrightarrow{Vg_h} = \epsilon_h((\overrightarrow{Vg_h} - \overrightarrow{Vv_h})) + \overrightarrow{Vg_h}$$
$$\overrightarrow{Vg_v} = \epsilon_v((\overrightarrow{Vg_v} - \overrightarrow{Vv_v})) + \overrightarrow{Vg_v}$$

Collision

Le but du comportement de `Collider` est de calculer la nouvelle vitesse d'une entité après une collision. Ce comportement peut être complexe et a été simplifié et schématisé sur la figure 3.9. Pour simplifier l'exposé, les chocs ne sont calculés que sur un plan.

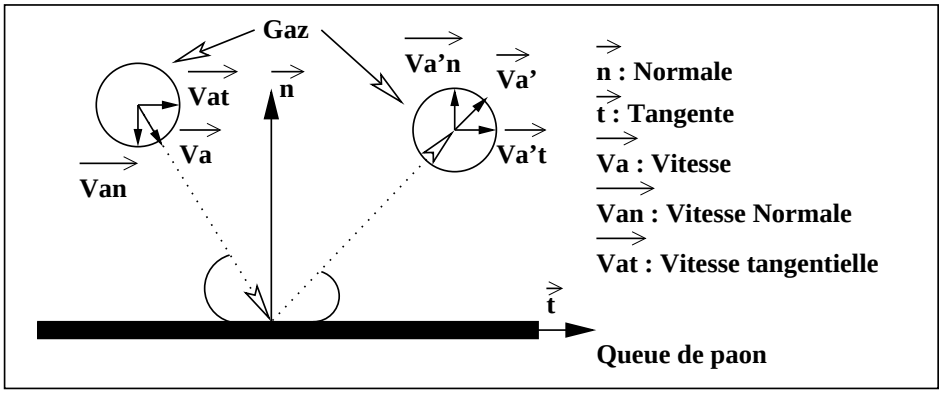


Figure 3.9 : Calcul des collisions.

L'objectif est alors de calculer la nouvelle vitesse $\overrightarrow{Va'}$ en fonction des nouvelles vitesses normale ($Va'n$) et tangentielle ($Va't$). Les composantes normale et tangentielle après le choc sont approximativement égales à :

$$\overrightarrow{Va'_n} = -e \times \overrightarrow{Va_n}$$
$$\overrightarrow{Va'_t} = \overrightarrow{Va_t}$$

Il s'agit d'un choc non élastique représenté par un coefficient d'absorption noté e. Les vitesses normale et tangentielle avant le choc Va_n et Va_t sont calculées à partir de la vitesse de l'entité et de l'angle de choc symbolisé par la normale $\overrightarrow{n}$ et la tangente $\overrightarrow{t}$ du noeud source.

La méthode `state()` affecte la nouvelle vitesse comme expliqué précédemment. Les valeurs e, $\overrightarrow{n}$ et $\overrightarrow{t}$ sont obtenues grâce à la méthode `input()` qui évalue les liens en entrée. Les liens sont des liens de collision. Ces liens disposent d'une méthode d'évaluation qui calcule le poids de l'interaction et récupère les sorties de la source de l'interaction. Dans ce cas, le poids est dynamique et booléen. Il s'agit de représenter le fait que les deux agents sont en collision ou non. Lors de l'appel à `getWeight()`, le lien calcule la distance entre l'agent source et l'agent cible. Si cette distance est inférieure ou égale à la somme des rayons des sphères englobantes des deux agents, il y a collision. Ce poids pondère la valeur retournée par l'agent source. La valeur retournée est un vecteur composé de $\overrightarrow{n}$, $\overrightarrow{t}$ (calculés à partir des vecteurs de position et d'orientation) et e.

Un agent désirant jouer le rôle `Collider` doit donc fournir un moyen de calculer sa position, son orientation et sa vitesse. Il doit également disposer d'un coefficient d'absorption des chocs et d'une sphère englobante. Ce sont les pré-requis pour jouer ce rôle qui sont couverts par les entités géoréférencées, `GeoEntity`.

3.2.5.3 Les entités organisationelles

Dans notre exemple, les différents types d'agent héritent de `GeoEntity`. Ils ont donc les capacités pour jouer les différents rôles expliqués précédemment. Il y a deux entités organisationnelles, une instance du réseau d'interactions de collision et une instance du réseau d'interactions de mobilité.

Dans le réseau d'interactions de collision, la queue de paon et les particules de gaz jouent un rôle de type `Collider`. Il y a donc création d'autant d'instances de comportements réactifs décrit par le rôle `Collider` que de particules de gaz et de queues de paon. La queue de paon joue également les rôles source et recruteur. C'est donc elle qui prend la responsabilité de créer les liens vers les nouvelles particules de gaz, qui elles jouent le rôle cible dans ce réseau.

Dans le réseau d'interaction de mobilité, le vent et les particules de gaz jouent le rôle de `Mobile`. Il y a création d'autant d'instances de comportements réactifs décrit par le rôle `Mobile` que de particules de gaz et de vent. Le vent joue également les rôles source et recruteur. C'est donc lui qui prend la responsabilité de créer les liens vers les agents entrant dans le réseau d'interaction de mobilité.

Les particules de gaz participent aux deux entités organisationnelles, elles ont donc deux comportements réactifs qui s'exécutent en parallèle.

3.2.6 Complexité de la perception

Les interactions peuvent être représentées sous forme de graphe ou maillage où les capteurs d'un agent *"pointent"* vers tel ou tel type d'élément de comportement de ses voisins. La réalisation des connexions est assimilée à la recherche des accointances ou à la perception de l'agent. La recherche des accointances est plus ou moins dynamique. En effet, certains agents ou certaines connexions peuvent avoir une dynamicité très faible (modification du vent...) Il est donc possible de trouver des méthodes pour réduire les temps de calcul dus à la perception des agents tout en gardant la dynamicité de l'environnement. Nous n'avons pas défini de règles permettant de choisir entre les différentes solutions, ce choix ne peut être fait que dans un cadre applicatif. Nous expliquons ici les méthodes que nous avons utilisées dans certaines applications sans donner de règles génériques.

3.2.6.1 Rémanence de la perception

Dans certains cas, les accointances d'un agent peuvent ne pas évoluer très vite. De même, le comportement résultant d'une erreur dans le calcul des accointances peut ne pas provoquer d'incohérence remarquable dans le comportement global du système. Par exemple, chaque particule de gaz devrait remettre à jour ses connaissances sur ses voisines à chaque cycle de vie, mais il y de fortes chances que ses voisines ne changent pas pendant un temps largement plus long que le temps de cycle de la particule. De plus la densité du nuage de gaz fait que s'il y a une erreur dans le calcul des particules voisines, cela n'est pas décelable dans le comportement global du nuage de gaz. Une particule de gaz n'est donc pas obligée de remettre à jour ses connaissances sur ses voisines à chaque cycle.

3.2.6.2 Simplification du graphe

Lors de la description du comportement d'un agent, il faut choisir le rôle qu'il va jouer dans l'organisation (source, cible ou recruteur). Pour faire ce choix, il faut tenir compte du fait que le comportement de l'agent doit rester le plus indépendant possible, mais qu'il est impossible de tenir compte de tous les phénomènes imprévisibles. Il faut donc replacer la conception du comportement de l'agent dans le cadre de son application.

Par exemple, dans l'interaction entre les particules de gaz et la queue de paon, si chaque particule recherche à chaque cycle un obstacle (une queue de paon), le temp de calcul du cycle de vie de la particule augmente considérablement. Dans notre application le nombre de sources d'eau (obstacle pour les particules de gaz) est largement plus faible que celui des particules de gaz. De plus, les jets d'eau sont manipulés par les utilisateurs (positionnement et déclenchement). Il est donc raisonnable de considérer que c'est le jet d'eau qui perçoit les particules de gaz et

les informe de sa présence. C'est donc lui qui joue le rôle recruteur.

3.2.6.3 Distribution des calculs

L'utilisation des systèmes multi-agents pour simuler l'environnement physique multiplie le nombre de processus. Certes, la complexité du calcul interne de ces processus est fortement allégée par rapport à un calcul classique global, mais dans le cadre d'une simulation par ordinateur, la consommation en mémoire s'en trouve fortement affectée. Par contre, la méthodologie agent autorise une plus grande distributivité de l'application sur plusieurs calculateurs, c'est ce qui a été réalisé dans [Follut et al. 00].

3.3 L'environnement social

L'environnement physique est également peuplé d'agents plus *"intelligent"*, certes ils le subissent et y agissent comme des agents réactifs, mais la manière dont ils choisissent leurs actions est effectuée à un niveau d'abstraction plus élevé. Il s'agit des différents intervenants humains qui sont joués par les agents autonomes, les apprenants ou les formateurs. Pour l'un de ces intervenants, les autres agents forment l'environnement social avec lequel il interagit. Dans notre cas, les membres connaissent la structure de l'organisation ; chacun connait son rôle et celui de ses partenaires. Les interactions entre les membres de l'équipe sont elles même structurées et agencées par le biais de procédures connues de tous les membres et définies par les experts du domaine. La simulation de ce contexte social consiste à modéliser les règles qui régissent les interactions entre les agents et la manière dont les tâches sont allouées entre eux. Il convient donc de définir, d'une part, un modèle de structure organisationnelle et la manière dont est gérée l'affectation des actions aux agents et d'autre part, un mécanisme de coordination entre les agents ainsi que leur comportement proactif.

Les deux premières parties de cette section présentent un modèle de structure organisationnelle pour le travail d'équipe et la description des procédures. Ce modèle est fondé sur le modèle générique d'organisation que nous avons présenté précédemment. Par la suite, nous expliquons le comportement proactif des agents participant aux équipes. Il s'agit d'un modèle de coordination et de sélection d'actions qui est expliqué dans la troisième partie. Enfin, pour illustrer les modèles proposés, la dernière partie de cette section traite un exemple tiré de notre problématique.

3.3.1 Travail en équipe

Dans le cadre d'étude que nous avons choisi, une équipe est conçue pour la résolution de problèmes relevant d'une même typologie (lutte contre un incident, acquisition

d'informations...). La répartition des tâches à accomplir pour la résolution du problème se fait par la description de rôles. Les missions représentent la coordination optimale des différents rôles pour la résolution d'un problème. Nous dérivons donc notre modèle générique d'organisation pour formaliser cette notion d'équipe. De plus, le type d'équipe que nous étudions est hiérarchique ; les agents doivent alors gérer les relations avec leurs supérieurs et leurs subordonnés. Le modèle de travail en équipe que nous proposons est présenté figure 3.10.

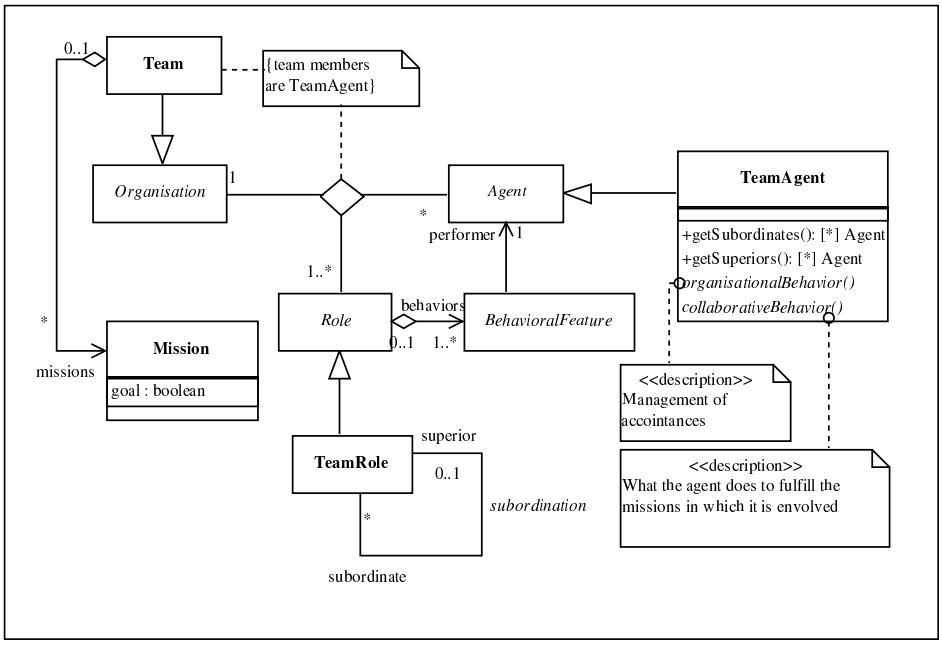


Figure 3.10 : Modèle de travail en équipe dans MASCARET.

L'équipe (`Team`) est une organisation dont les membres sont des `TeamAgent`. Ces agents ont un comportement collaboratif (`collaborativeBehavior()`) qui prend en compte les autres membres de l'équipe en fonction des rôles qu'ils jouent. L'équipe étant hiérarchique, il existe un lien de subordination entre les rôles (`TeamRole`). Cela permet à un agent de connaître ses supérieurs (`getSuperiors()`) et ses subordonnés (`getSubordinates()`). Ces liens s'apparentent aux liens organisationnels de type subordination de [Hannoun et al. 99] et comme dans [Gutknecht et al. 99], cela signifie que l'agent jouant le rôle de plus haut niveau hiérarchique de l'équipe peut jouer en plus, un rôle de représentant dans une autre équipe considérée alors d'un niveau de responsabilité supérieur.

Une équipe doit remplir plusieurs missions (`Mission`) qui peuvent s'exécuter en parallèle ou en séquence. Les missions représentent les buts communs (`goal`) des membres de l'équipe et lorsqu'une mission est sélectionnée, chaque intervenant œuvre pour atteindre ce but. En effet, si l'organisation décrit les règles du jeu par un ensemble de contraintes organisationnelles, une mission peut s'apparenter à une tactique qui décrit l'agencement des actions et des communications entre les intervenants ; elle sert donc de cadre à la coordination des actions des agents. Nous nous intéressons au cas où

cette coordination est déjà prévue et écrite dans des plans d'actions que l'on designe par le terme de procédure. Nous proposons donc un modèle de travail procédural fondé sur le modèle d'équipe.

3.3.2 Travail procédural

L'écriture d'une procédure peut, par certains aspects, s'apparenter à l'écriture d'un scénario en animation comportementale. Différents langages permettant l'écriture de tels scénarios ont déjà été proposés, comme par exemple dans [Devillers 01], mais dans notre cas l'environnement est dynamique ; les membres de l'équipe peuvent avoir besoin d'adapter le scénario à l'environnement. La procédure doit alors avoir une représentation sémantique afin que les agents puissent raisonner dessus. Notons également que certains de ces agents sont des utilisateurs humains ; pour qu'elle leur soit compréhensible, la procédure doit donc s'approcher du langage naturel. Dans le modèle que nous proposons (figure 3.11), les missions sont des procédures (`Procedure`) représentées par des ensembles de contraintes (`Constraint`) décrivant l'agencement des actions (`Action`) effectuées par les agents jouant les rôles de l'équipe. Le raisonnement des membres de l'équipe porte sur les procédures et sur les actions. Nous proposons donc un modèle permettant de représenter ces deux concepts.

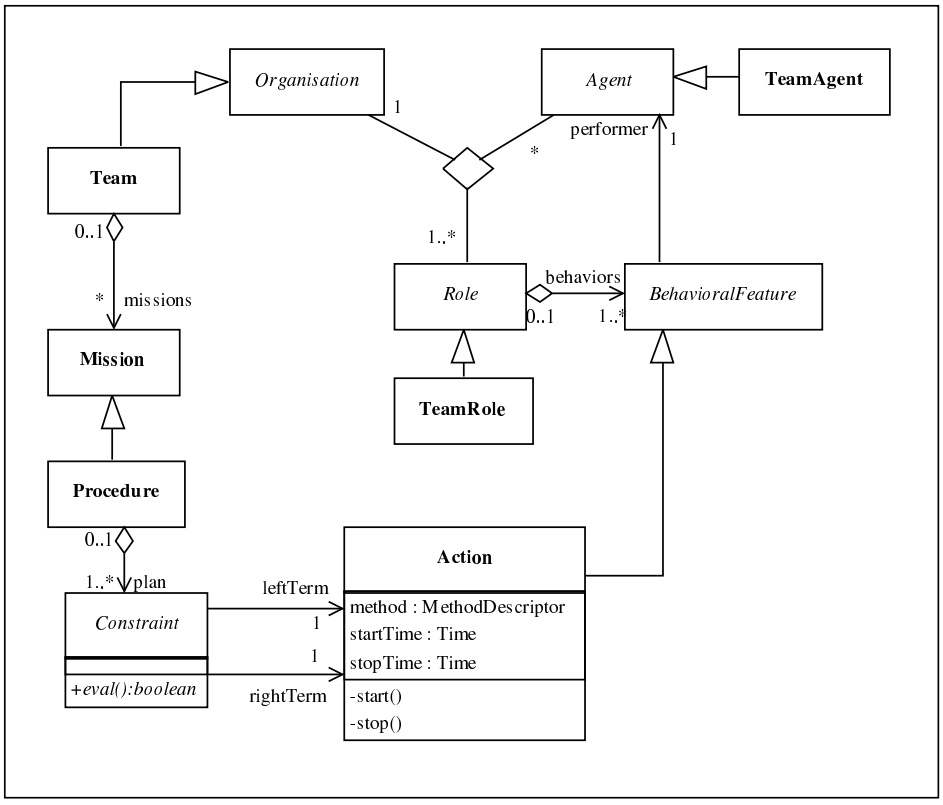


Figure 3.11 : Travail procédural en équipe dans MASCARET.

Dans une procédure, l'agencement des actions est temporel. En effet, la procédure exprime le fait qu'une action doit se faire avant, après ou pendant une autre action. De plus, les actions peuvent être considérées comme des intervalles de temps possédant un début et une fin. Pour décrire une procédure, nous utilisons la logique temporelle de Allen [Allen 83] (logique sur les intervalles) et nous utilisons l'implémentation qui en a été faite par [De Loor et al. 00].

Les logiques temporelles définissent un système formel intégrant la notion de temps. C'est-à-dire qu'elles décrivent une grammaire, des axiomes et des règles d'inférence, ce qui a permis leur utilisation comme outil de validation ou de planification. La logique temporelle de Allen s'applique aux raisonnements sur les intervalles de temps, c'est-à-dire des durées possédant une date de début (d) et une date de fin (f) telles que $d < f$. La logique temporelle de Allen définit treize opérateurs pour décrire toutes les combinaisons possibles d'agencement d'un intervalle par rapport à un autre dans le temps. Ces opérateurs sont *précède, rencontre, chevauche, débute, dans, termine, égal* ainsi que leurs transposés (figure 3.12). Le nombre d'intervalles et de scénarios résultant de leur agencement est fini.

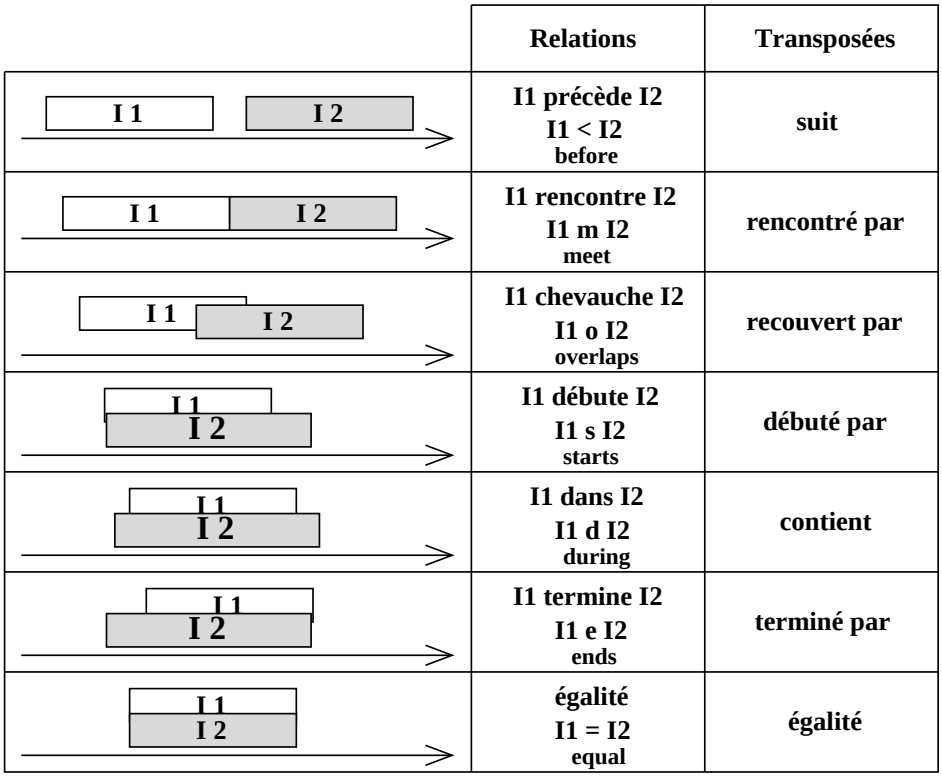


Figure 3.12 : Les opérateurs de la logique de Allen.

Il existe toutefois un problème dans l'utilisation de la logique temporelle pour la description de procédures jouées par des agents autonomes. En effet, la procédure décrit un agencement d'actions, implémentées par un code informatique. Mais comme il est montré dans [De Loor et al. 00], il existe une ambiguïté dans le lien entre

l'intervalle et l'action. Est-ce la fin de l'action qui provoque la fin de l'intervalle ou bien au contraire, la fin de l'intervalle qui provoque la fin de l'action (respectivement pour le début de l'intervalle et de l'action) ? Par exemple dans la phrase "Le serveur débarrasse la table après que le client ait mangé" se traduisant par la contrainte :

$$\texttt{Serveur.Debarrasse} > \texttt{Client.Mange}$$

est-ce le début de l'action *débarrasser* qui provoque la fin de *manger* ou inversement la fin de *manger* qui provoque le début de *débarrasser* ? L'ajout d'un critère de priorité sur les opérateurs de Allen permet de lever cette ambiguïté. Ainsi pour exprimer une priorité sur l'action *débarrasser*, c'est-à-dire que le début de *débarrasser* provoque la fin de *manger* on écrit la contrainte :

$$\texttt{Serveur.Debarrasse} \circ > \texttt{Client. Mange}$$

En revanche, pour exprimer que le serveur ne débarrasse que lorsque le client a fini de manger, on écrit :

$$\texttt{Serveur.Debarrasse} > \circ\ \texttt{Client.Mange}$$

Les auteurs enrichissent chaque opérateur de Allen par le biais de priorités pour rendre cette logique exécutable par des agents autonomes.

Une procédure est alors vue comme un ensemble de contraintes temporelles sur les actions des agents. Pour exprimer qu'un agent i doit faire l'action n avant que l'agent j ne fasse l'action m, nous créons une contrainte :

$$Agt_i.Act_n \circ < Agt_j.Act_m$$

ce qui peut s'écrire sous forme d'un prédicat :

$$P\_Before(Agt_i, Act_n, Agt_j, Act_m)$$

La lettre P avant l'opérateur marque une priorité à gauche alors que lorsqu'elle se trouve après elle marque une priorité à droite. La figure 3.13 montre un exemple de traduction d'une procédure tirée de notre exemple de sécurité civile vers un ensemble de contraintes temporelles.

Une équipe peut réaliser plusieurs procédures en même temps, il faut donc veiller à l'intégrité de l'ensemble des contraintes résultant de la fusion des missions. L'algorithme de Allen permet cette vérification, il est alors possible d'ajouter une contrainte organisationnelle permettant à un rôle spécifique de l'organisation (le rôle de plus haut niveau hiérarchique par exemple) de gérer l'incompatibilité des missions.

Les procédures portent sur les actions des rôles, nous définissons donc un modèle minimum d'action tel que présenté figure 3.11. L'action est d'abord vue comme la description d'une méthode (au sens du modèle objet). Pour pouvoir réaliser cette action, l'agent doit implémenter cette méthode (il s'agit des pré-requis) et proposer un moyen d'introspection. Dans le cadre d'un environnement virtuel de formation, le professeur doit disposer d'une trace de ce qui s'est passé pour pouvoir en faire une analyse avec l'étudiant. Ainsi chaque action est datée, elle possède une date de début et de fin (`startTime` et `stopTime`). Lorsqu'une action est démarrée ou arrêtée, ces

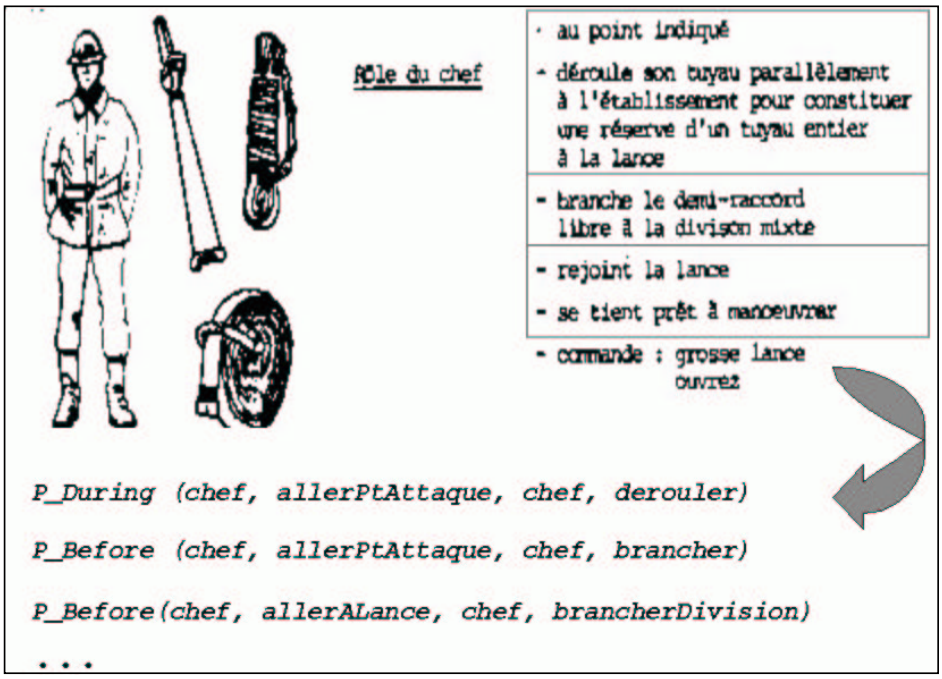


Figure 3.13 : Traduction d'une procédure en contraintes temporelles.

attributs sont valués. Une action dispose donc de méthodes permettant de la démarrer ou de l'arrêter (`start()` et `stop()`). Ces méthodes mettent à jour les dates de début et de fin de l'action et appelle la méthode correspondante chez l'agent jouant le rôle ; elles ne sont *a priori* pas redéfinies dans les classes dérivées d'`Action`.

3.3.3 Comportement des agents rationnels

Le premier type d'agent à pouvoir jouer un rôle dans une équipe sont les agents rationnels autonomes. Nous proposons donc un modèle décrivant le comportement de ces agents évoluant dans l'environnement social et physique. Le modèle que nous proposons s'articule autour de trois types de comportement de l'agent. Tout d'abord l'agent dispose de comportements réactifs ; l'agent peut ainsi participer aux interactions de l'environnement physique. Il dispose également d'un comportement collaboratif ; l'agent participe à la réalisation des buts de l'équipe par un comportement de sélection d'actions en fonction de l'évolution de la procédure. Ce comportement doit également lui permettre de s'adapter aux configurations non prévues. Enfin il se sert de l'organisation pour trouver une solution aux problèmes qu'il ne sait résoudre seul, c'est son comportement organisationnel ; les équipes étant hiérarchiques, il délègue le problème à ses subordonnés ou en réfère à son supérieur.

La suite de cette section explique ces trois comportements de l'agent. Les comportements réactifs sont ceux déjà expliqués dans la seconde section de ce chapitre, nous

ne montrons ici que la manière dont les agents autonomes les réalisent. Le comportement collaboratif de l'agent est un comportement rationnel fondé sur la connaissance de l'évolution de la procédure. Nous proposons un premier modèle où cette connaissance est commune aux membres de l'équipe. Cette solution ne répond pas totalement à notre problématique, nous proposons donc un second modèle dans lequel la connaissance de l'évolution de la procédure est répartie chez chaque agent de l'équipe. Enfin, nous expliquons le comportement organisationnel de l'agent, fondé sur le *Contract Net Protocol*.

3.3.3.1 Comportements réactifs

Les agents participent à l'environnement physique. Pour être pris en compte dans cet environnement, ils doivent intervenir dans les réseaux d'interactions qui le décrivent. Les agents disposent donc de comportements réactifs (`reactiveBehavior`). Par contre, nous considérons que ces agents ne peuvent pas jouer un rôle `Source`. En effet, ces comportements réactifs sont causés par l'environnement. L'agent rationnel agit dans l'environnement par le biais des actions qu'il décide d'exécuter et non pas par des comportements réactifs sur lequel il ne peut raisonner. Le lien entre les comportements réactifs et le comportement proactif de l'agent se fait par la modification de ses variables d'état internes. En effet, les comportements réactifs de l'agent modifient ses variables qui sont utilisées lors de l'exécution des méthodes, elles mêmes invoquées par les actions de l'agent. Par exemple, l'effet toxique d'un gaz, modifie le champ de perception de l'agent, ce qui influe du même coup sur l'action de recherche de l'agent qui se fonde sur ce champ de perception. Il pourrait être intéressant d'étudier l'effet de ces comportements réactifs sur les capacités cognitives de l'agent, telles que son raisonnement, ses capacités de communication ou encore ses émotions. On pourrait pour cela s'inspirer des cartes cognitives floues comme dans [Parenthoen et al. 01] qui relient différents concepts (émotifs) par des liens inhibiteurs ou exitateurs.

La figure 3.14 montre une interaction entre des jets d'eau, un feu et des sapeurs-pompiers. Le premier pompier arrose le feu, il utilise pour cela un jet d'eau qui agit sur le feu. Pour le faire dans de bonnes conditions, le pompier doit s'approcher du feu, ce qui augmente la température du pompier. Un second pompier aide alors le premier en l'arrosant à son tour par un jet d'eau, ce qui stabilise sa température. Le premier pompier subit donc deux interactions réactives dont il est la cible.

3.3.3.2 Connaissance centralisée de l'évolution de la procédure

Dans ce premier modèle, nous faisons l'hypothèse que l'agent a une connaissance parfaite de la procédure et de son exécution. En effet, nous supposons d'une part que les agents sont en contact permanent et que rien ne peut gêner leur perception des actions des autres membres et d'autre part qu'un agent peut considérer dans son raisonnement que les autres membres de l'équipe connaissent la procédure et qu'ils partagent la

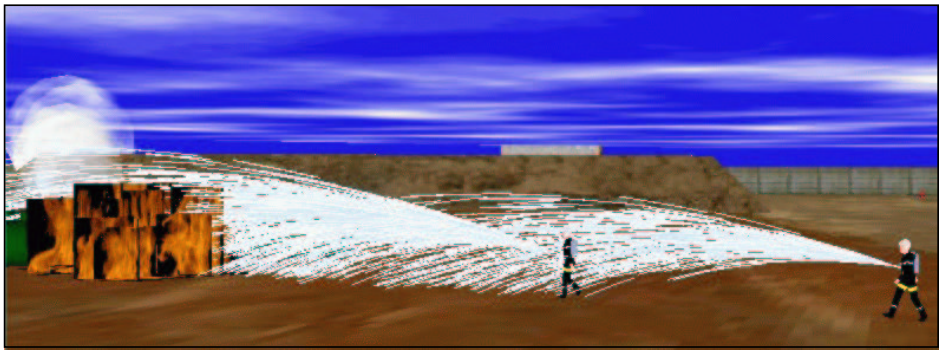


Figure 3.14 : L'arroseur arrosé.

même volonté de la respecter. La procédure et la perception de son exécution par les agents sont considérées comme des connaissances partagées par tous les membres de l'équipe. Simuler l'exécution d'une procédure par une équipe consiste à gérer l'intégrité des contraintes et à générer des interactions entre les agents. Nous proposons donc un modèle, qui a déjà été présenté dans [Querrec et al. 01a], et qui est représenté figure 3.15, dans lequel la notion d'équipe est équivalente à celle d'un gestionnaire de contraintes.

Un gestionnaire de contraintes (`ConstraintManager`) contrôle un ensemble de contraintes qui dans notre cas sont temporelles (`TemporalConstraint`) telles que `P_Before`, `Meet`... Elles portent sur des actions qui sont considérées par le gestionnaire de contraintes sur des intervalles de temps (`TemporalAction`). Le comportement de sélection d'actions de l'agent est d'une part de vérifier qu'il peut démarrer (ou arrêter) les actions qu'il souhaite et d'autre part d'exécuter les actions que lui impose la procédure. L'agent raisonne alors sur les actions en terme de désirs, droits et devoirs. La figure 3.16 montre les différents états de la réalisation d'une action par un agent.

Lorsqu'un agent souhaite démarrer une action (`wantToStart`), soit du fait d'un événement extérieur, soit du fait d'une requête du gestionnaire de contraintes (réception de `mustStart`), l'action change d'état et attend l'autorisation du gestionnaire de contraintes pour démarrer (`WaitForStart`). En passant dans cet état, l'agent demande au gestionnaire de contraintes de démarrer (arrêter) les actions qui, selon la procédure, doivent démarrer (ou s'arrêter) avant que l'agent ne puisse démarrer son action. Quand l'action reçoit l'autorisation de démarrer (`canStart`), elle démarre et passe à l'état `running`. Lorsque l'agent souhaite arrêter l'action (`wantToStop`), soit parce qu'elle s'est terminée, soit par une requète du gestionnaire de contraintes (réception de `mustStop`), l'action change d'état et attend l'autorisation du gestionnaire de contraintes pour s'arrêter (`WaitForStop`). En passant dans cet état, l'agent demande au gestionnaire de contraintes de démarrer (arrêter) les actions qui, selon la procédure, doivent démarrer (s'arrêter) avant que l'agent ne puisse arrêter son action. Quand l'action reçoit l'autorisation de s'arrêter (`canStop`), l'action s'arrête et retourne dans l'état `sleeping`.

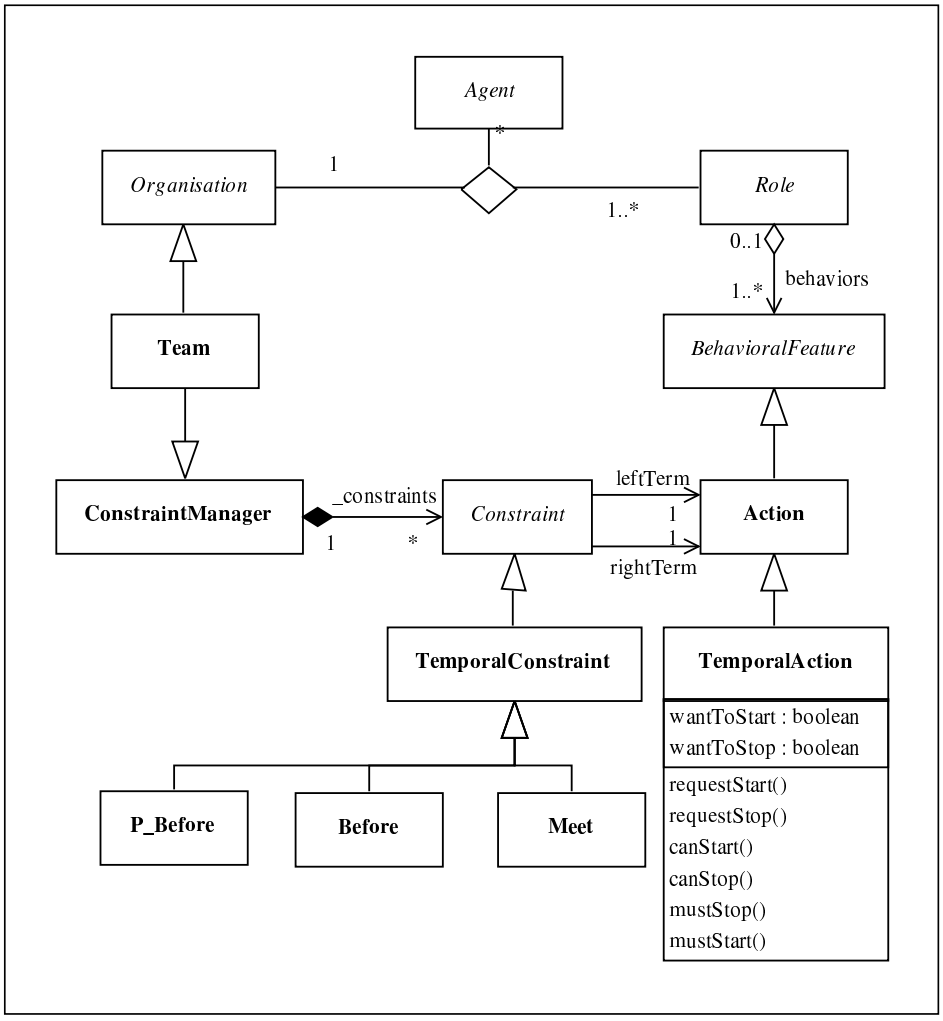


Figure 3.15 : Gestionnaire de contraintes temporelles.

L'utilisation de la logique temporelle dans ce modèle permet de décrire les procédures selon un langage proche du langage naturel et permet aux agents d'inférer et de détecter les incohérences. L'expressivité des contraintes temporelles se traduit par la possibilité de scénarios multiples associés à un même ensemble de contraintes. Dans certains cas, elle peut être pénalisante car l'obtention d'un scénario unique nécessitera l'ajout de contraintes implicites pour le modélisateur. Le choix d'une architecture centralisé pour représenter la connaissance de la procédure et la vision qu'en ont les agents de son exécution permet certes de simuler des synchronisations fortes entre les actions des agents, mais présente tout de même quelques inconvénients. En effet, il rend difficile la distribution des comportements des agents intervenant dans une même équipe sur plusieurs calculateurs. De plus, ce modèle ne permet pas à un utilisateur de prendre la main sur un agent car il est difficile de découper la phase de décision de la phase d'exécution des actions. Ce modèle ne permet pas non plus de simuler des dysfonctionnements dans la perception des actions des autres agents, ce qui est un point intéressant dans le cadre d'un environnement virtuel de formation. Enfin, le

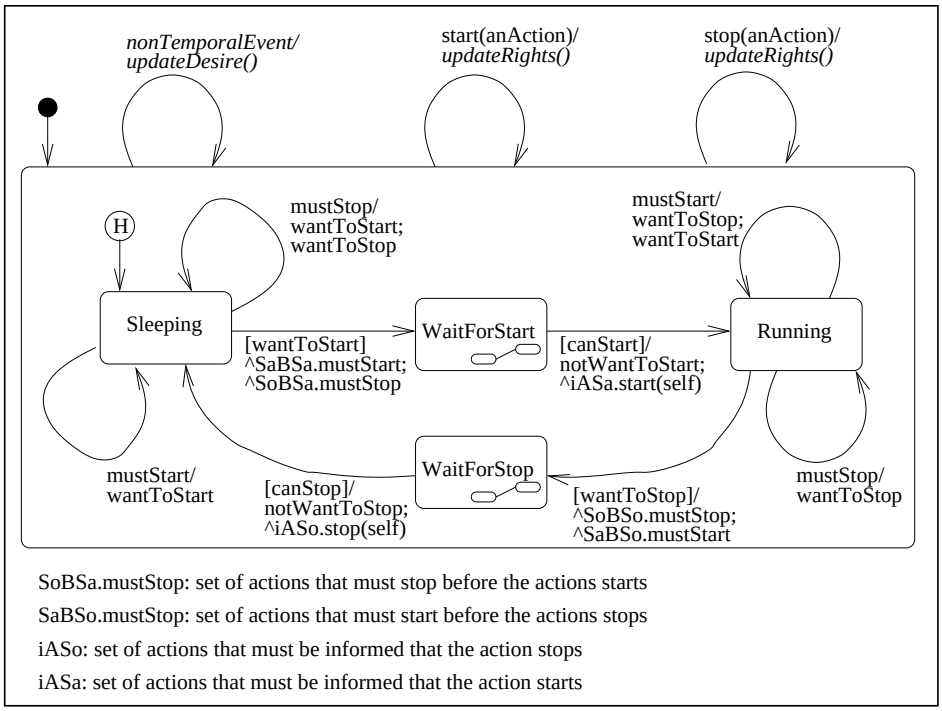


Figure 3.16 : Diagramme d'états d'une action temporelle.

modèle d'action ne permet pas aux agents de raisonner sur le sens de celle-ci lorsqu'il doit en choisir une pour s'adapter à une situation non prévue.

3.3.3.3 Connaissance distribuée de l'évolution de la procédure

Nous proposons un modèle d'agent disposant de connaissances locales sur les procédures et l'organisation. Son comportement est alors d'exécuter les actions de la procédure et de s'adapter aux situations non prévues. La procédure décrit les interactions entre les agents dans un cas optimal, et laisse aux agents la responsabilité de se construire des plans implicites (non explicités dans la procédure) car considérés comme naturels dans un cadre applicatif. La procédure organise des actions d'un niveau sémantique que nous appelons actions métiers tels que "arroser un feu" ou "faire une mesure d'explosimétrie" dans le cas des procédures de pompiers alors que les plans implicites (calculés par l'agent) agencent des actions d'un niveau sémantique plus générique pour un humain tels que "aller à un point" ou "prendre un objet". Pour cela l'agent doit être capable de raisonner sur les actions. Nous proposons donc un modèle d'actions lui permettant de faire son raisonnement (figure 3.17).

Les actions dirigées par les buts (`GDAction`[2]) disposent de préconditions sous la forme d'une expression booléenne représentant un sous-état du monde virtuel,

[2] *Goal Directed Action*

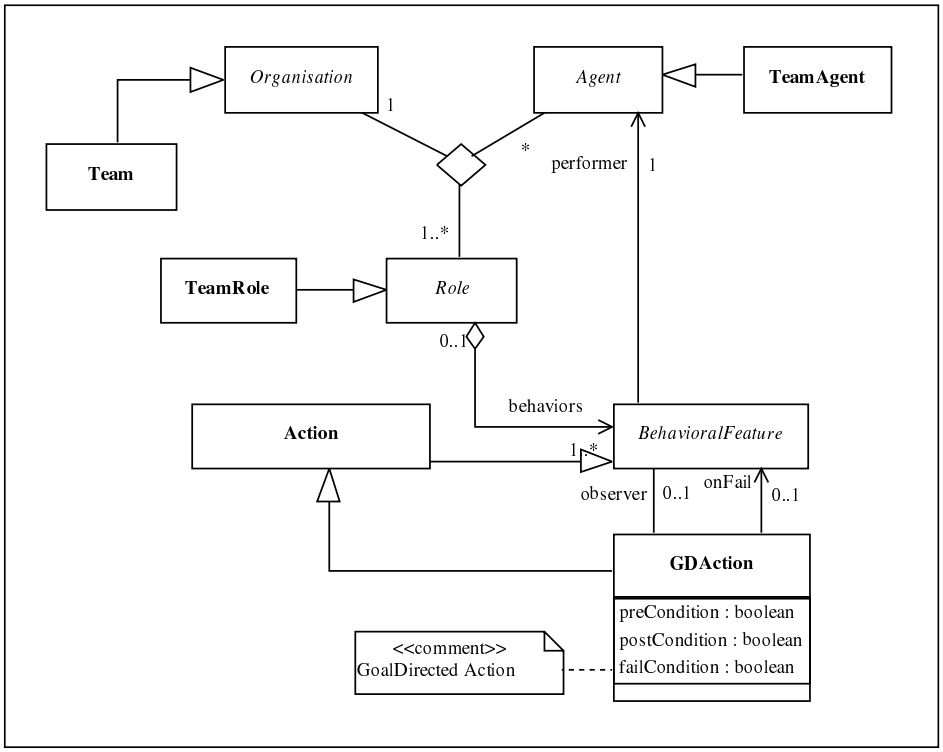


Figure 3.17 : Actions dirigées par les buts dans MASCARET.

observable par l'agent. Ainsi, avant d'exécuter une action, l'agent doit s'assurer que les préconditions de celle-ci sont vérifiées. Si ce n'est pas le cas, il doit se construire lui-même un plan qui lui permet d'atteindre son but. Pour pouvoir le faire, l'agent doit avoir une connaissance du résultat des actions. Une action possède donc une postcondition qui est une expression booléenne représentant le sous-état du monde (théorique) après la réalisation de l'action. De plus, l'environnement étant dynamique, une action peut échouer. Pour chaque action on associe un élément de comportement (`observer`) permettant d'évaluer une expression booléenne caractérisant l'échec ou le succès de l'action. En cas d'échec de l'action, celle-ci invoque un autre élément de comportement `onFail` pour que l'agent tente de corriger cette erreur.

L'agent dispose d'une base de connaissances pour mémoriser les informations qui lui servent lors de son raisonnement. Ainsi, lorsqu'un agent s'incrit dans une équipe, il mémorise les rôles de l'équipe, les actions définies par ces rôles et l'affectation des agents aux rôles. Lorsque l'équipe doit réaliser une procédure, les agents la mémorise, dans son ensemble, dans leur base de connaissances, mais la vision locale qu'ont les agents de cette procédure ne permet pas la synchronisation des actions. Par conséquent les opérateurs de Allen correspondant (*meet*...) ne peuvent plus être utilisés pour la décrire. Dans ce cas, les procédures peuvent être représentées par un graphe, et sont écrites à l'aide de deux opérateurs : la séquence (`seq`) et le parallélisme (`par`). La structure globale de la base de connaissances de l'agent est présentée sur la figure 3.18.

```
Procedure i : seq [ (Role i, action i) , par [ (Role j, action j) ...] ]

Role i : Action i, Action j ...

Action i : Precondition, Postcondition, Resultat;

Affecter (Role i, agent i) ....
Affecter (Role j, moi)
```

Figure 3.18 : Structure d'une base de connaissances des agents.

Au cours de l'exécution de la procédure, il faut un mécanisme pour que l'agent puisse acquérir la connaissance des actions réalisées par les autres membres de l'équipe. En toute rigueur, ce mécanisme devrait être un mécanisme de perception de l'agent et de reconnaissance de l'action qu'il effectue. Un tel mécanisme pose des problèmes qui n'ont, à notre connaissance, pas encore été complètement résolus. Pour pallier ces problèmes, nous supposons que les agents sont réellement collaboratifs et qu'ils n'ont aucune intention de cacher leurs actions. Nous simulons donc cette perception par des envois de messages. Lorsqu'un agent démarre ou arrête une action, il diffuse cette information par message à tous les membres de l'équipe. Les agents recevant ces messages insèrent ces informations dans leur base de connaissances et mémorisent l'échec ou le succès de l'action (variable `Resultat` de l'action dans la base de faits).

L'agent est divisé en un partie décisionnelle et une partie opérative tels que schématisées figure 3.19. Le raisonnement de l'agent est représenté par son comportement collaboratif et son comportement organisationnel. Le comportement collaboratif est fondé sur un module de sélection d'actions utilisant la base de faits. Ce module communique avec la partie opérative de l'agent par messages pour obtenir des informations ou agir sur l'environnement. Lorsque ce comportement est en défaut, il invoque le comportement organisationnel qui peut lui permettre de trouver une solution chez un autre membre de l'organisation. Par symétrie, le comportement organisationnel peut requérir l'exécution d'une action pour un autre agent. La partie opérative est représentée par les modules suivants qui s'exécutent en parallèle :

- ▷ perception : acquiert la connaissance sur l'état du monde par la simulation d'un champ de vision, la consultation d'attributs internes ...
- ▷ communication : envoi les messages prévus par les procédures et ceux émis lors du démarrage et de l'arrêt des actions ; reçoit les messages des autres agents et alimente la base de connaissances.
- ▷ exécution des actions : réalise effectivement les actions dans l'environnement physique. Ce module renseigne le module décisionnel sur le résultat de l'action (échec ou succès).

L'agent évolue dans un environnement dynamique peuplé d'autres agents qui y effectuent également des modifications. Le maintien de la cohérence d'une base de connaissances sur un environnement dynamique étant difficile, l'agent, dans notre modèle, acquiert ses connaissances par un mécanisme de perception. Ainsi, lors de son raisonnement l'agent se fonde uniquement sur les connaissances organisationnelles

et procédurales précédemment citées ainsi que sur la représentation symbolique de l'environnement fournie par sa partie opérative. Lorsque l'agent exécute une procédure, il doit, avant la réalisation de chaque action, vérifier qu'elle est exécutable dans l'état actuel de l'environnement. Ce raisonnement *"temps réel"*, basé sur une connaissance locale peut aboutir à une décision non appropriée ou tardive. C'est pourquoi il est nécessaire d'intégrer la notion d'échec et de planification dynamique aux mécanismes de bases du module décisionnel. L'agent mémorise les actions qui ont été réalisées avec succès ou échec et s'en sert dans son raisonnement pour suivre l'évolution de la procédure. L'algorithme de raisonnement de l'agent est comparable à un PRS (Procedural Reasoning System) tel que décrit dans [Georgeff et al. 87]. Nous présentons ce mécanisme en deux parties : la première est le suivi de l'exécution de la procédure et la seconde le calcul d'un plan implicite.

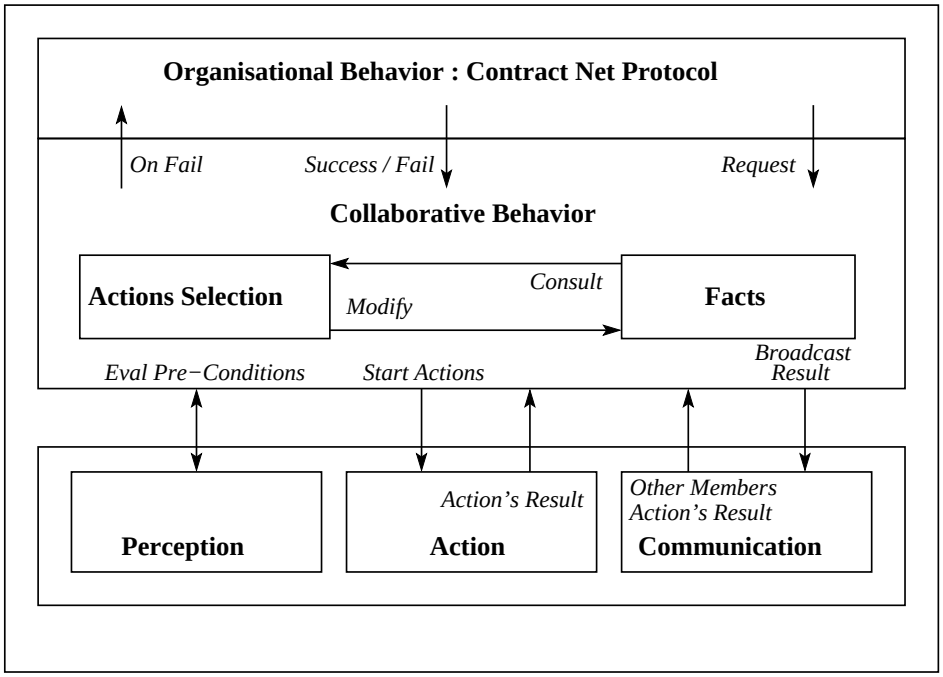


Figure 3.19 : Architecture d'un agent rationnel autonome dans MASCARET.

Suivi de la procédure.

L'agent commence par consulter la procédure et exécute les actions en séquence ou en parallèle selon l'opérateur utilisé. L'agent consulte donc la première action à exécuter s'il s'agit d'une séquence, ou les premières actions s'il s'agit d'actions à exécuter en parallèle. Rappelons que la procédure enregistrée dans la base de connaissances de l'agent est globale à l'équipe. Pour chaque action, l'agent vérifie si c'est à lui de réaliser l'action (vérification du rôle qu'il joue dans l'équipe). Si ce n'est pas à lui de la réaliser mais à un autre membre de l'équipe, il se met en attente d'un message de fin d'action provenant de l'agent concerné. Tous les agents partagent la connaissance

procédurale et la même intention de réaliser la procédure. L'agent peut donc partir du principe que les autres membres de l'équipe vont réaliser les actions qui sont de leur responsabilité, que ces actions se terminent par un succès ou un échec et qu'il en sera informé. Si c'est à l'agent de réaliser l'action et si ses préconditions sont vérifiées, le module décisionnel provoque l'exécution de l'action par la partie exécutive de l'agent. Le module décisionnel attend la fin de l'action. La figure 3.20 illustre ce mécanisme.

Si l'action réussit, le module décisionnel sélectionne l'action suivante dans le cas d'actions en séquence. Dans le cas d'actions à réaliser en parallèle, il n'existe pas de mécanisme d'acquittement mutuel entre les actions pour synchroniser leur arrêt. La synchronisation des actions est uniquement représentée par une *prise de rendez-vous*. Le module décisionnel attend simplement la fin de toutes les actions avant de continuer la procédure.

Si l'action échoue, le module décisionnel fait exécuter l'action prévue dans le champ `onFail` de l'action en échec et attend le résultat de cette action. L'action prévue dans le champ `onFail` permet par défaut d'invoquer le comportement organisationnel de l'agent. Si cette action est à son tour en échec, la procédure est arrêtée.

Calcul d'un plan implicite.

Si les préconditions d'une action de la procédure ne sont pas vérifiées, le module décisionnel calcule les plans considérés comme implicites qui permettent de vérifier chacune des préconditions. Pour ce calcul, nous distinguons deux types d'action : les actions *métiers* qui interviennent dans la procédure et les actions génériques qui n'interviennent pas dans la procédure mais qui sont tout de même décrites dans les rôles. Les plans implicites sont calculés à partir des actions génériques. Le calcul de plan se fait à l'aide d'un chaînage arrière sur les préconditions et les postconditions des actions. Par exemple, la précondition d'une action *métier* ‘‘`faire une mesure d'explosimétrie`’’ est ‘‘`possède un explosimètre`’’, et la postcondition de l'action générique ‘‘`prendre un Objet`’’ est ‘‘`possède l'Objet`’’. Le module décisionnel unifie donc la variable `Objet` à la constante `explosimètre`.

Le module décisionnel cherche parmi les actions génériques celles qui permettent de réaliser la précondition, c'est-à-dire celles dont les postconditions correspondent à la précondition. Plusieurs actions peuvent remplir ces conditions, il s'agit alors des points de départ de différentes solutions possibles pour atteindre le but ce qui peut être représenté par un graphe. Chaque branche de ce graphe est calculée en évaluant les préconditions de l'action courante. Si les préconditions ne sont pas vérifiées, le module décisionnel cherche les actions pouvant vérifier les préconditions. Le processus est réitéré jusqu'à trouver une action exécutable (préconditions vérifiées). Le module décisionnel dispose alors d'un graphe complet de solutions. Chaque branche est une solution et chaque nœud est une action à exécuter. Le module décisionnel choisit la solution la plus courte. Pour l'instant la solution la plus courte est celle possédant le

moins d'actions à réaliser mais ce mécanisme peut intégrer une heuristique prenant en compte le temps d'exécution estimé de chaque action. Le module décisionnel choisit alors la première action de la solution choisie et provoque son exécution.

La partie opérative de l'agent réalise l'action et renseigne le module décisionnel sur son résultat. Si l'action a réussi, l'agent insère dans sa base de faits cette information. L'environnement étant dynamique, l'agent recalcule l'ensemble des solutions possibles. Cette méthode, si elle est plus coûteuse en temps, procure un comportement opportuniste à l'agent. Si l'action échoue, le module décisionnel mémorise l'action comme inutilisable dans la base de connaissances pour toute la durée du plan implicite. En effet, nous considérons que, dans le cadre de notre problématique, les plans implicites ont un temps d'exécution court, une action inutilisable à un instant l'est également pendant la durée du plan implicite. Cette connaissance est temporaire et oubliée à la fin du plan implicite. L'agent recalcule ensuite l'ensemble des solutions ne prenant plus en compte cette action. Ce mécanisme est répété jusqu'au succès du plan implicite (préconditions de l'action de la procédure vérifiées) ou jusqu'à ce que le module décisionnel ne trouve plus de solution. Dans le dernier cas, le module décisionnel invoque le comportement organisationnel de l'agent. Il attend ensuite une réponse de ce comportement. S'il est en échec, la procédure est arrêtée. Ce mécanisme est illustré sur le diagramme d'activité de la figure 3.20.

3.3.3.4 Effacement de buts et opportunisme

L'implémentation de ce comportement à l'aide d'un moteur d'inférences de type Prolog subit le principe d'effacement de buts. Ainsi, un but déjà atteint ne sera plus calculé. Pour illustrer ce principe, prenons l'exemple d'un agent devant franchir une porte. Les préconditions de cette action sont d'une part d'être proche de la porte et d'autre part de posséder la clé pour ouvrir cette porte. L'agent calcule tout d'abord un plan lui permettant de se rendre à la porte, puis calcule un plan lui permettant de trouver une clé. Ce faisant, l'agent s'éloigne de la porte et le plan lui permettant d'y retourner ne sera plus calculé. Les préconditions de l'action sont donc considérées comme vraies par le moteur d'inférences alors que ce n'est pas le cas. Pour résoudre ce problème, notre algorithme ré-évalue toutes les préconditions avant d'exécuter une action. Ainsi tant que toutes les préconditions ne sont pas vérifiées, le module décisionnel recalcule les plans lui permettant de vérifier les préconditions qui ne le sont pas. Dans notre exemple, les plans sont recalculés jusqu'a ce que l'agent dispose d'une clé et soit à la porte.

Dans cet algorithme, un autre problème persite ; les agents que nous avons conçus ne sont pas *"opportunistes"*. En effet, si un agent doit posséder un dévidoir et une lance, il va tout d'abord calculer le plan lui permettant de prendre le dévidoir puis le plan pour la lance. Ainsi, si lors de l'exécution du premier plan, l'agent se trouve proche de la lance, il ne la saisira pas, il attendra d'avoir pris le dévidoir pour calculer le plan lui permettant de prendre la lance. Ce problème est classique en intelligence

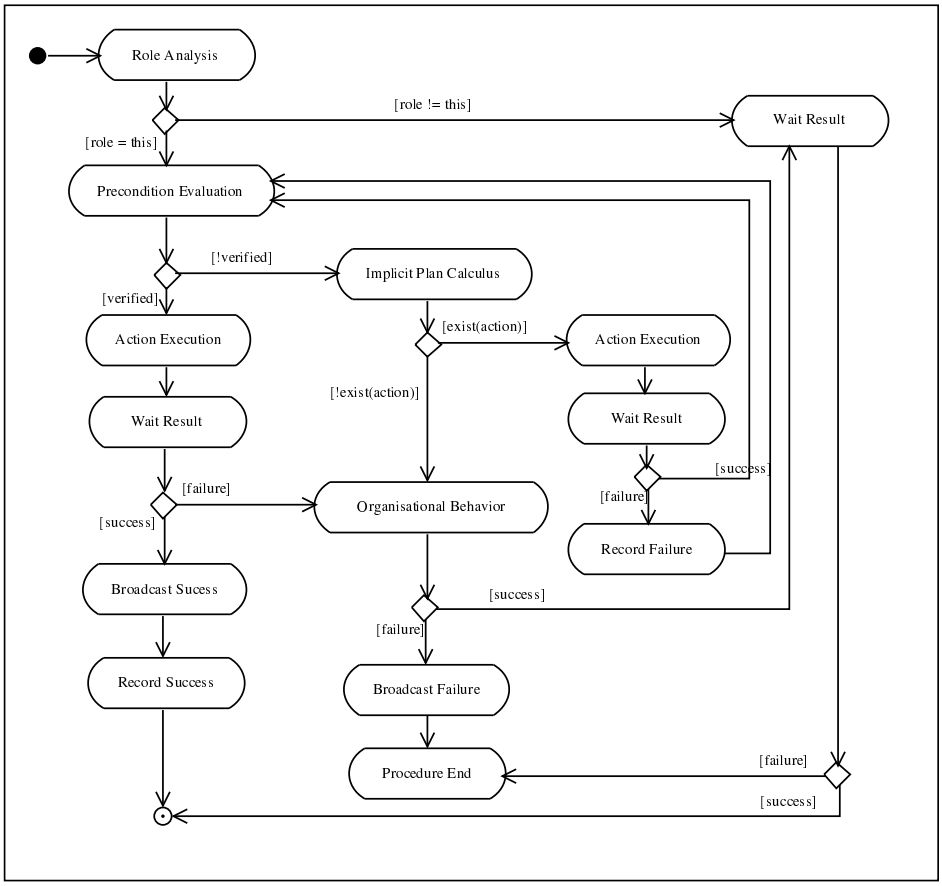


Figure 3.20 : Réalisation d'une action.

artificielle, pour le résoudre, il faudrait s'inspirer des architectures d'agents présentées dans le chapitre précédent telles que [Brooks 86] et [Ferguson 92].

3.3.3.5 Comportement organisationnel

En plus des comportements réactifs qui lui permettent d'être intégré à l'environnement physique, de son comportement collaboratif de sélection d'actions qui lui permet de participer à la réalisation des procédures, l'agent dispose d'un comportement organisationnel qui lui permet d'intégrer la structure de l'organisation dans son raisonnement. Ce comportement permet de mettre à jour les accointances des agents, c'est-à-dire l'affectation des rôles. Ce comportement dépend du rôle joué par l'agent. Dans le cas d'une organisation hiérarchique, c'est l'agent jouant le rôle le plus haut dans la hiérarchie qui gère l'affectation des agents aux rôles et vérifie leurs prérequis. C'est également dans ce comportement que l'agent peut prendre en compte des règles sociales dictées par l'organisation.

Ce comportement organisationnel intègre également un mécanisme de recours en

cas d'échec du raisonnement de l'agent. Ainsi dans une organisation hiérarchique, lorsqu'un agent est face à un problème qu'il ne sait pas résoudre, il en réfère à son supérieur. Le supérieur a alors en charge de trouver une solution parmi ses subordonnés. S'il n'en trouve pas, il en réfère à son propre supérieur hiérarchique. Nous avons représenté ce mécanisme par une méthode de type `Contract Net`. Ce mécanisme a été développé dans le cadre d'un travail de D.E.A ([Bellard 01]) ; il est schématisé sur la figure 3.21. Lorsqu'un agent échoue dans son raisonnement (précondition d'une action non vérifiée), il transmet un message à son supérieur contenant la raison de son échec (précondition). Le chef lance alors un appel d'offre auprès de ses subordonnés. Les agents reçoivent ce message et inspectent les actions des rôles qu'ils jouent dans l'organisation. Les agents possédant une action dont la postcondition correspond à la précondition et pouvant réaliser cette action (précondition de l'action vérifiée) répondent à l'appel d'offre. Le chef choisit alors un agent parmi les réponses qu'il a reçues. Nous n'avons pour l'instant pas défini de mécanisme d'évaluation des propositions des agents, ni de mécanisme de négociation. L'agent choisit alors la première réponse qu'il reçoit et demande l'exécution de l'action par l'agent. S'il ne reçoit pas de réponse, il transmet le problème à son propre supérieur hiérarchique. Dans ce cas la résolution du problème ne se place pas au niveau des actions mais au niveau des missions. Rappelons que les missions disposent d'un but décrit par une condition booléenne décrivant un état du monde après l'exécution de la mission, ce qui s'apparente aux postconditions des actions.

Dans le cadre d'un environnement virtuel de formation, les utilisateurs jouent des rôles dans cette organisation, le problème leur sera donc posé à un moment ou un autre. Si le problème remonte jusqu'à l'apprenant, c'est qu'il a peut-être fait une erreur dans ses prises de décision. Si le problème remonte jusqu'au formateur, il peut modifier l'environnement ou le scénario pour le résoudre.

3.3.4 Exemple : une équipe FPT

Pour illustrer les modèles de travail en équipe, de procédure et de comportement d'agents que nous avons exposé précédemment, nous prenons un exemple tiré de notre cas d'étude. Cet exemple est fondé sur une équipe de sapeurs-pompiers nommée FPT (Fourgon Pompe Tonne) qui a comme responsabilité d'attaquer l'incident (éteindre un foyer, ralentir la progression d'une fuite de gaz ...). Pour cela, l'équipe FPT peut être amenée à exécuter plusieurs manœuvres pour dérouler des tuyaux et attaquer l'incident selon différents cas de figure dans l'environnement. Dans cette section, nous expliquons l'organisation de cette équipe, nous prenons également l'exemple d'une des procédures de l'équipe et nous isolons un cas d'exécution de cette procédure en étudiant le comportement d'un des agents.

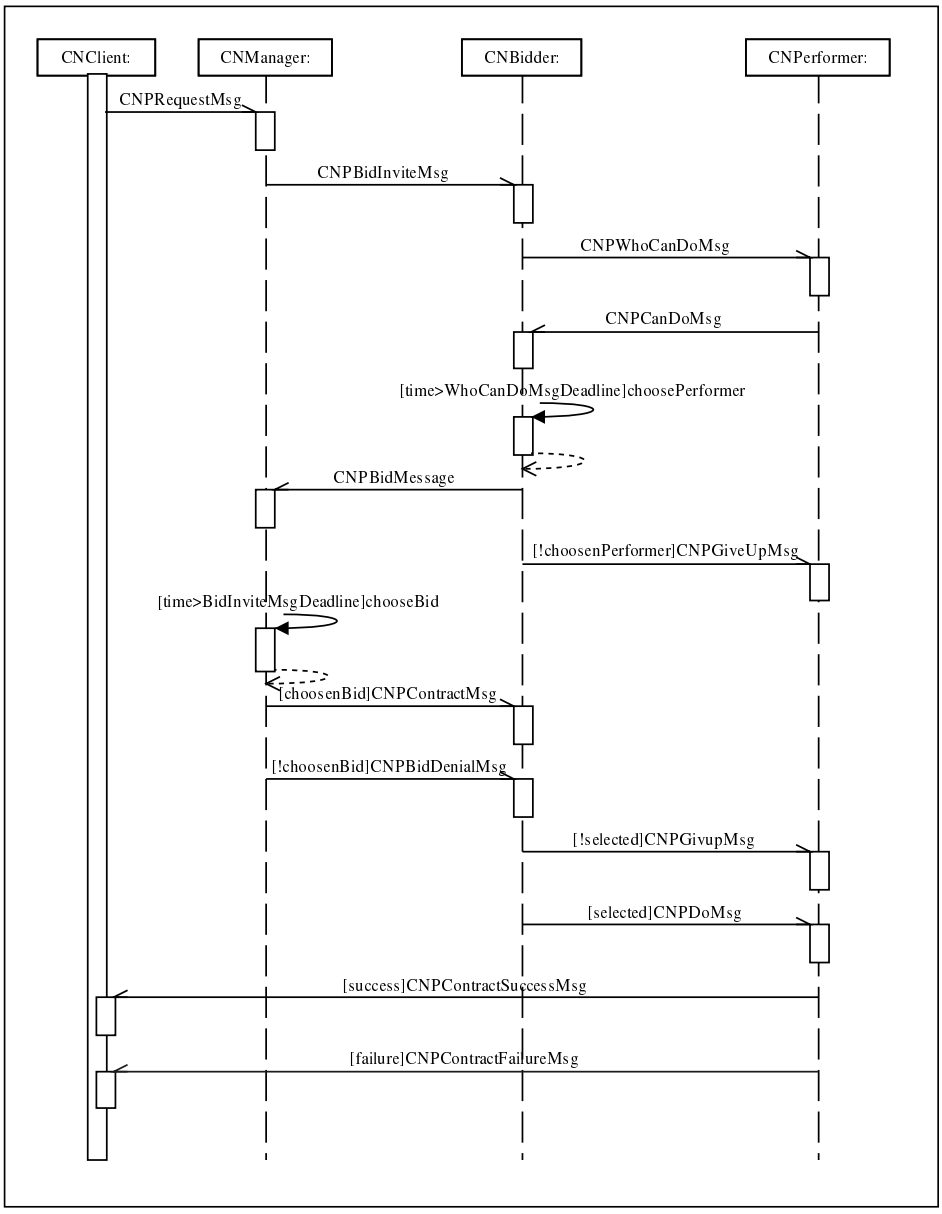


Figure 3.21 : Le *Contract Net Protocol* (d'après [Bellard 01]).

3.3.4.1 La structure organisationnelle

Une équipe FPT est organisée autour de cinq rôles : le chef d'agrès, le chef du binôme d'alimentation, l'équipier du binôme d'alimentation, le chef du binôme d'attaque (`chef BAT`) et l'équipier du binôme d'attaque (`equipier BAT`). Le chef BAT a pour responsabilité de dérouler les tuyaux et d'arroser le feu. Son équipier vérifie l'état des tuyaux et l'aide à arroser le feu. La figure 3.22 présente trois des actions du rôle `chef BAT`. Ces actions sont les éléments de comportement du rôle qui interviennent dans la description des procédures. Pour chacune, est associé un nom, une méthode à appeler chez l'agent (l'agent désirant jouer ce rôle doit proposer cette méthode : il s'agit des

prérequis) ainsi que des préconditions et des postconditions.

Action	**Méthode**	**Préconditions** **Postconditions**
SeTenirPret	`wait()`	`posseder(ari)` `posseder(dévidoir)`
		`etrePret()`
AllerAuPointAttaque	`goTo(ptAttaque)`	`etrePret()`
		`presDe(ptAttaque)`
BasculerLaFlèche	`fallOver(devidoir)`	`posseder(devidoir)`
		`etrePret(devidoir)`

Figure 3.22 : Actions du rôle chef BAT (extrait).

Tous les rôles décrits dans cette équipe dérivent d'un rôle plus générique, celui d'humanoïde. Ce rôle est composé d'actions qui n'interviennent pas dans les procédures mais que tout agent participant à une équipe FPT doit implicitement savoir réaliser. Ces actions servent au raisonnement de l'agent lors du calcul de plans implicites. La figure 3.23 présente quelques unes de ces actions, elles sont définies de la même manière que les actions précédentes par un nom, une méthode ainsi que des préconditions et des postconditions, mais on peut voir qu'elles ne manipulent pas d'instance telle que `devidoir` ou `ptAttaque`, mais des variables (`Objet`) qui sont unifiées lors de leur utilisation.

Action	**Méthode**	**Préconditions** **Postconditions**
Prendre	`take(Objet)`	`voir(Objet)` `presDe(Objet)`
		`posseder(Objet)`
SeRapprocherDe	`goTo(Objet)`	`voir(Objet)`
		`presDe(Objet)`
ChercherObjet	`lookFor(Objet)`	`NONE`
		`voir(Objet)`

Figure 3.23 : Actions génériques d'humain (extrait).

3.3.4.2 Les missions

Une équipe FPT connaît vingt missions qui ont pour but d'alimenter le FPT, arroser un incident de telle ou telle manière ou secourir des personnes. Nous avons proposé deux méthodes de conception du comportement de l'agent. Selon la première méthode, la procédure s'écrit sous forme d'un ensemble de contraintes temporelles de Allen; ces contraintes sont écrites dans les missions qui sont définies dans une équipe de

type FPT. La figure 3.24 montre un extrait de l'ensemble des contraintes temporelles définissant la mission M10. Lors de l'exécution de la mission, l'équipe gère l'intégrité des contraintes et génère des interactions entre les agents. Ainsi, dans notre exemple, lorsque le `chefAgres` termine l'action `OrdonneCmdPreparatoire`, le gestionnaire de contrainte demande l'exécution de l'action `SeTenirPret` au `chefBAT` et à `equipierBAT` (2 premières contraintes `P_Meet`). Puis le `chefAgres` ordonne l'exécution de la mission et le `chefBAT` commence l'action `AllerAuPointAttaque`. Grâce à la quatrième contrainte (contrainte `P_During`) le gestionnaire de contraintes demande à `equipierBAT` de réaliser l'action `AllerAuPointAttaque` lorsque le `chefBAT` débute l'action et lui demande de l'arrêter lorsque le `chefBAT` l'arrête. La procédure peut ensuite continuer normalement.

```
% Procedure M10 : Etablissement d'une division mixte a plus de 200 mètres.

   P_Meet (ChefAgres, OrdonneCmdPreparatoire, chefBAT, SeTenirPret);
   P_Meet (ChefAgres, OrdonneCmdPreparatoire, equipierBAT, SeTenirPret);
   P_Meet (ChefAgres, OrdonneCmdExecutoire, chefBAT, AllerAuPointAttaque);
   P_During (ChefBAT, AllerAuPointAttaque, equipierBAT, AllerAuPointAttaque);
   P_Meet (chefBAT, AllerAuPointAttaque, chefBAT, DecrocherDivision);
   P_Meet (chefBAT, DecrocherDivision, chefBAT, OrdonneBasculementFleche);
   P_Meet (chefBAT, OrdonneBasculementFleche, equipierBAT, BasculerLaFleche);

   ...
```

Figure 3.24 : Partie des contraintes temporelles de la missions M10.

Comme nous l'avons mentionné dans la section précédente, ce mode de fonctionnement (centralisé) ne résout pas complètement notre problème. Dans la version distribuée du problème, nous utilisons la description de la procédure sous la forme d'un plan séparant les actions par des opérateurs de séquence (`seq`) ou de parallélisme (`par`). La figure 3.25 montre le plan de l'exemple précédent retranscrit dans ce formalisme. Si la première solution semble plus naturelle à lire par un utilisateur final, la seconde méthode de description des procédures représente mieux la réalité des missions dans le cadre de la sécurité civile. En effet, l'écriture à l'aide de contraintes temporelles donne lieu à plusieurs scénarios alors que les missions des sapeurs-pompiers décrivent la procédure de référence choisie par les experts. Ces informations sont la connaissance commune des membres de l'équipe. Lorsqu'une équipe reçoit l'ordre de réaliser cette mission, la procédure est mémorisée dans la base de faits des agents. Nous expliquons par la suite, le comportement des agents lors de la réalisation de la procédure.

3.3.4.3 L'entité organisationnelle

Dans notre exemple, les agents jouant les rôles de l'équipe FPT sont des agents de la classe `Fireman` (figure 3.26). Cette classe définit des méthodes telles que `goTo()`, `unroll()`... La classe `Fireman` hérite de la classe `Human`, représentant les humanoïdes, qui définie des méthodes telles que `take()`, `lookFor()`... Les instances de la classe

```
seq [ (ChefAgres, OrdonneCmdPreparatoire) ,
      par [ (chefBAT, SeTenirPret), (equipierBAT, SeTenirPret)],
      (ChefAgres, OrdonneCmdExecutoire),
      par [ (ChefBAT, AllerAuPointAttaque, equipierBAT, AllerAuPointAttaque)],
      (chefBAT, DecrocherDivision),
      (chefBAT, OrdonneBasculementFleche),
      (equipierBAT, BasculerLaFleche), ... ]
```

Figure 3.25 : Plan dans la base de faits de l'agent.

`Fireman` peuvent donc jouer les rôles définit dans l'équipe FPT car ils couvrent les prérequis ; ils disposent des méthodes qui sont invoquées lors de l'exécution des actions. De plus, la classe `Human` dispose d'attributs qui permettent à ses instances de jouer des rôles dans des interactions réactives. En effet, un humain subit, par exemple, les effets de la toxicité de l'air ainsi que ceux des transferts thermiques. Ainsi, les humains participent aux réseaux d'interactions de toxicité en y jouant les rôles `Toxic` et `Target`. Pour cela, ils disposent d'un comportement réactif `ToxicBehavior` qui influe sur ses capacités de perception (`visionAngleXY`...) et ses capacités de déplacement (attributs hérités de `GeoEntity`). Les humains participent également aux réseaux d'interactions thermiques grâce à un comportement réactif `ThermicBehavior`. L'effet de cette interaction est présenté sur la figure 3.14 de la page 73. Rappelons, que dans le cas des interactions réactives, les agents autonomes, représentant des humanoïdes, ne peuvent jouer le rôle `Source` car nous considérons que ces agents agissent de manière raisonnée dans l'environnement *via* leurs actions.

Le scénario de l'exercice instancie une équipe FPT et affecte les rôles aux cinq agents de la classe `Fireman` également créés. Un de ces agents (`Fireman.2`) joue le rôle du `chefBAT`. A l'exécution de ce scénario, il enregistre dans sa base de faits les affectations des agents aux rôles, la description (nom, pré et postconditions...) des actions définies par le rôle qu'il joue ainsi que la description des actions du rôle générique d'humain. Lorsque l'équipe reçoit l'ordre de réaliser la procédure M10, tous les agents jouant un rôle dans cette équipe insèrent dans leurs bases de connaissances le plan de la procédure et commencent à la réaliser. La figure 3.27 illustre un exemple de réalisation de cette procédure par l'équipe FPT.

La première action du plan est `chefAgres ordonneCmdPreparatoire`, ce n'est pas le rôle du `Fireman.2`, il n'effectue pas l'action, mais attend le message de succès de l'agent concerné pour continuer la procédure. La suite du plan est la réalisation de deux actions en parallèle. Ces deux actions sont considérées comme deux flots d'exécution distincts. La première action est `chefBAT etrePret`. Le `Fireman.2` joue le rôle `chefBAT`, c'est donc lui qui réalise l'action. Il s'assure d'abord que les préconditions sont vérifiées. Les préconditions de l'action `SeTenirPret` sont de posséder un ARI et de posséder un dévidoir. Pour savoir si ces préconditions sont vérifiées, le module décisionnel demande à la partie exécutive d'exécuter une méthode de perception pour verifier que l'agent possède un ARI. Cette méthode retourne un message positif au module décisionnel. Le même mécanisme est utilisé pour la deuxième précondition

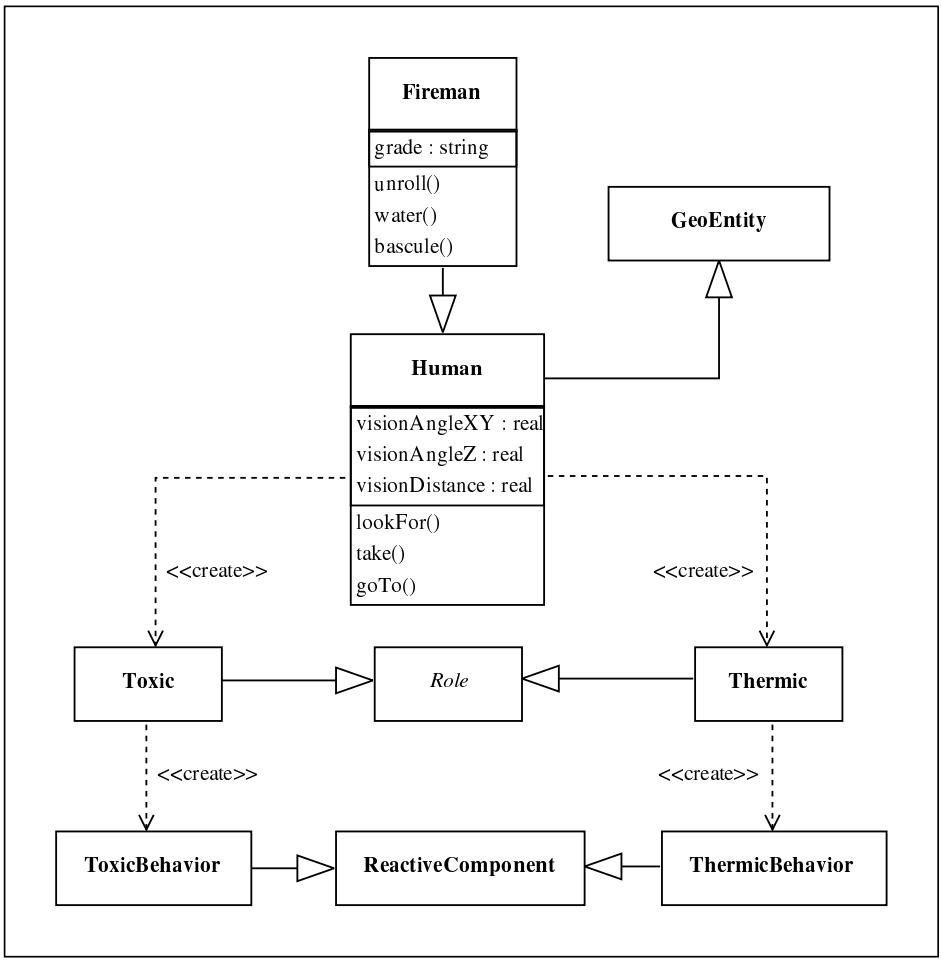


Figure 3.26 : La classe `Fireman`.

Figure 3.27 : Exemple d'équipe réalisant une mission.

(posséder un dévidoir), mais le message est cette fois négatif. L'agent doit donc calculer un plan pour trouver un dévidoir. La figure 3.28 montre le calcul d'un plan implicite par unification des variables et chaînage arrière.

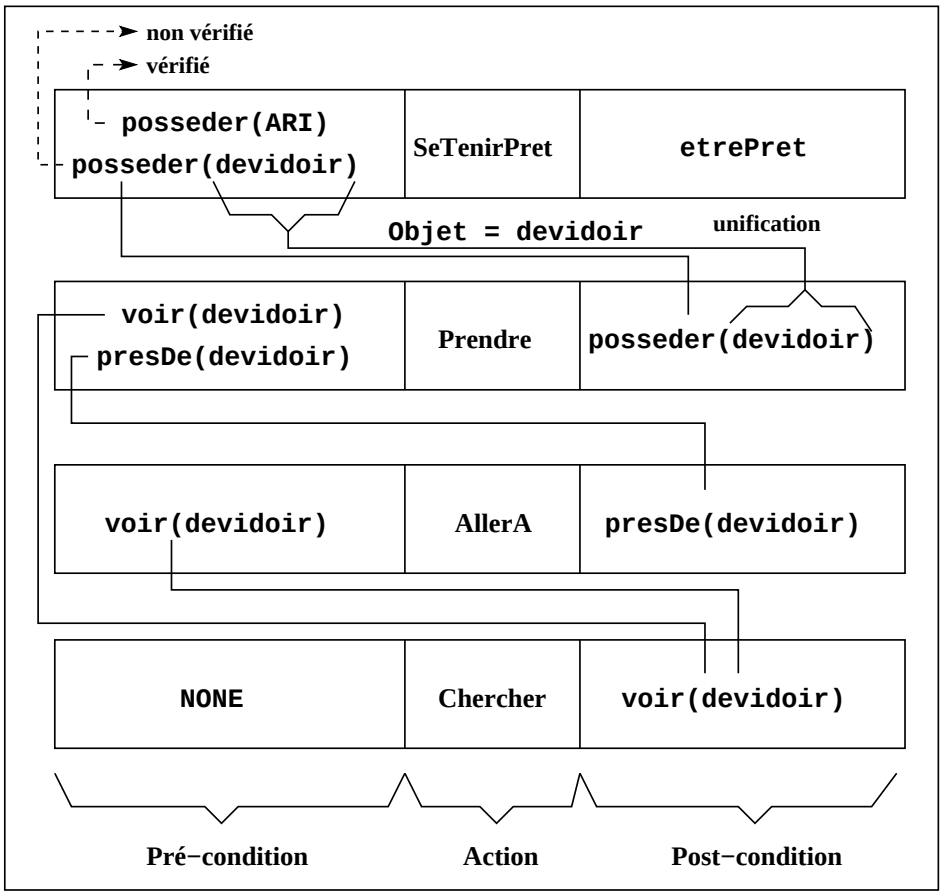


Figure 3.28 : Résultat d'un calcul d'un plan implicite.

Le module décisionnel cherche alors parmi les actions génériques, une action permettant de posséder un dévidoir. L'action `prendre` permet de posséder un objet. Le module décisionnel unifie la variable `Objet` à la constante `devidoir` et sélectionne l'action. Cette action a deux préconditions qui ne sont pas vérifiées (voir l'objet et être près de l'objet). Le module décisionnel cherche alors une action permettant de voir un objet et une action permettant d'être proche de l'objet. Il trouve les actions `chercher` et `AllerA`. La partie exécutive réalise alors l'action `chercher` et retourne un message de succès au module décisionnel. Celui-ci recalcule alors le plan implicite. La précondition de l'action `etrePret`, n'est toujours pas vérifiée, l'agent recalcule alors un plan qui le fait se rapprocher de l'objet, puis lui fait prendre l'objet.

A chaque fois que l'agent exécute une action, il diffuse un message à tous les membres de son équipe, ainsi chacun peut suivre l'évolution de la procédure. Le `Fireman.2` reçoit donc un message de l'agent jouant l'`equipierBAT` qui l'informe du succès de son action `etrePret`. Les deux flots d'exécution se rejoignent et la procédure peut continuer.

3.4 Les avatars

L'avatar est le représentant de l'utilisateur dans l'environnement. Classiquement, les utilisateurs sont les apprenants qui réalisent des exercices dans l'environnement, mais nous avons choisi d'immerger également le formateur pour qu'il puisse y réaliser des fonctions pédagogiques (conseil, énoncé d'exercices...), il est donc également représenté par un avatar. L'avatar participe, au même titre que les agents autonomes, à l'environnement social en jouant des rôles dans les équipes et il participe également à l'environnement physique en subissant les effets des phénomènes simulés.

Nous concevons l'interaction entre l'utilisateur et son avatar comme une collaboration dans laquelle les rôles de chacun sont spécifiés. Cette collaboration peut s'apparenter à un type d'organisation spécifique à vocation pédagogique, mais nous ne l'avons, pour l'instant, pas formalisée. L'utilisateur a la responsabilité de la sélection des actions et l'avatar perçoit et agit dans l'environnement pour l'utilisateur qu'il représente. Mais, pour des raisons pédagogiques, il est intéressant de permettre la délégation de tâches et de procédures à l'avatar ; celui-ci doit alors être doué d'une certaine autonomie.

Le modèle d'avatar que nous proposons est fondé sur celui d'agent rationnel expliqué plus haut. Cela permet d'intégrer l'avatar dans l'environnement physique et dans l'environnement social. De plus, le mécanisme que nous avons utilisé pour modéliser les agents autonomes, en distinguant une partie opérative et une partie de raisonnement, permet d'implémenter la collaboration entre l'humain et l'avatar. L'utilisateur humain collabore avec son avatar par des messages *via* une interface humain/machine. Notre modèle n'impose à cette interface que la capacité de communiquer avec l'avatar. L'interface peut donc être textuelle, 2D ou fondée sur un rendu 3D (*via* une caméra virtuelle par exemple) et sur l'utilisation de périphériques de réalité virtuelle.

3.4.1 Intégration dans l'environnement physique

L'utilisateur, comme les agents autonomes, participe à l'environnement physique ; il dispose donc de comportements réactifs et subit l'effet des phénomènes physiques. Par exemple, lorsqu'un utilisateur traverse un nuage de gaz, il subit l'effet toxique de ce phénomène. Ainsi, sa vue se trouble, son champ de vision se rétrécit, ses capacités de mouvement diminuent. Le résultat de cette interaction réactive est perçue par l'utilisateur grâce à son interface par la modification des paramètres de la caméra qui lui permet d'observer l'environnement et du périphérique lui permettant de se mouvoir.

Comme les agents collaboratifs, les avatars ne jouent pas de rôles source dans un réseau d'interactions mais uniquement des rôles cible ; nous ne considérons leurs effets sur l'environnement que par les actions qu'ils effectuent et non par leur comportements réactifs. Suivant l'application, il peut être plus efficace de le considérer comme un

recruteur dans les réseaux auxquels il participe. Par exemple, il vaut mieux que ce soit l'avatar qui mette à jour ses liens en entrée dans une interaction avec les particules de gaz plutôt que ce soit à chaque particule de le faire. Pour des raisons pédagogiques, il est également possible de le rendre insensible à certains phénomènes en ne lui faisant jouer aucun rôle dans le réseau considéré.

3.4.2 Intégration dans l'environnement social

L'utilisateur est également intégré dans l'environnement social. Le comportement qu'il doit adopter est le même que celui des agents autonomes, c'est-à-dire un comportement collaboratif dans lequel il réalise des procédures en se coordonnant avec les autres membres de l'équipe et un comportement organisationnel lui permettant de déléguer des tâches, ou d'en réaliser pour d'autres membres de l'équipe. L'architecture interne de l'avatar est, comme le montre la figure 3.29, strictement la même que pour un agent rationnel, à l'exception de l'inhibition de messages. En effet, tous les modules composant l'avatar sont actifs et restent donc potentiellement utilisables ; cependant certaines communications sont coupées. Ainsi, le comportement collaboratif n'invoque plus le comportement organisationnel (lors d'un échec sur une action) et le module de raisonnement ne communique plus avec la partie opérative pour lui demander l'execution d'une action. Par contre, les messages portant sur le résultat des actions effectuées par les autres agents transitent jusqu'au module de décision, ce qui permet d'alimenter la base de connaissances de l'avatar pour qu'il puisse suivre l'évolution de la procédure. De même, les messages que le module de décision envoie au module de perception pour évaluer les préconditions d'une action restent actifs, ce qui permet à l'avatar, pour des raisons pédagogiques, de calculer les plans implicites ou d'interdire l'exécution de l'action par l'utilisateur si elle n'est pas exécutable.

Comme pour les agents rationnels, lorsque l'utilisateur prend la responsabilité de jouer un rôle (*via* son avatar), il acquiert la connaissance organisationnelle (rôles, actions des rôles et affectations) ainsi que la connaissance procédurale. Rappelons que l'objectif de l'outil de formation n'est pas l'apprentissage de la procédure, mais celui de son exécution en situation opérationnelle. Toute cette connaissance est enregistrée dans la base de faits de l'avatar et peut être consultée par l'utilisateur *via* une interface. Pour des raisons pédagogiques, l'interface peut, comme c'est le cas sur la figure 3.30, n'afficher que les actions du rôle joué par l'utilisateur et ne pas afficher les informations sur la procédure et sur son exécution. Le rôle de l'utilisateur est alors de choisir l'action qu'il veut exécuter ; il transmet cette information sous forme de message à son avatar. La simplicité de l'interface se justifie par le fait que l'on se place dans le cadre de la formation à la prise de décision ; ce type d'apprentissage ne nécessite pas de mécanismes immersifs évolués, ce qui n'aurait pas été le cas dans le cadre d'un apprentissage aux gestes techniques.

Lorsqu'un avatar démarre une action ou l'arrête, cela doit être perçu par les autres

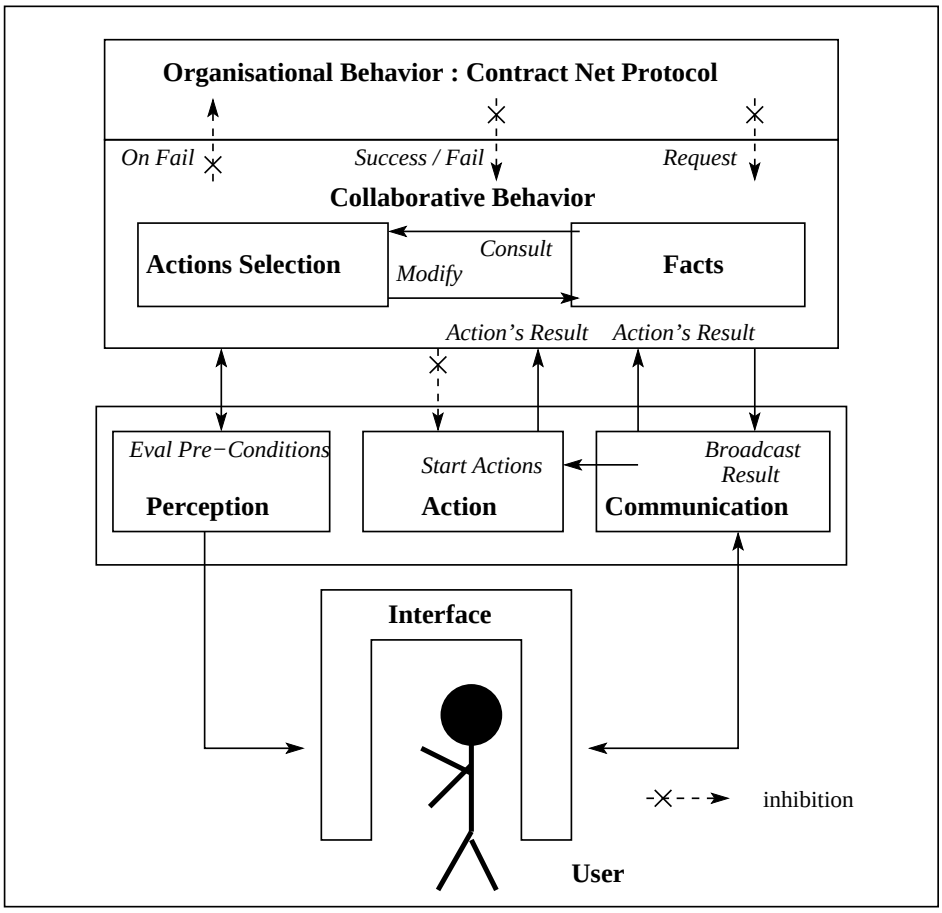


Figure 3.29 : Architecture d'un avatar dans MASCARET.

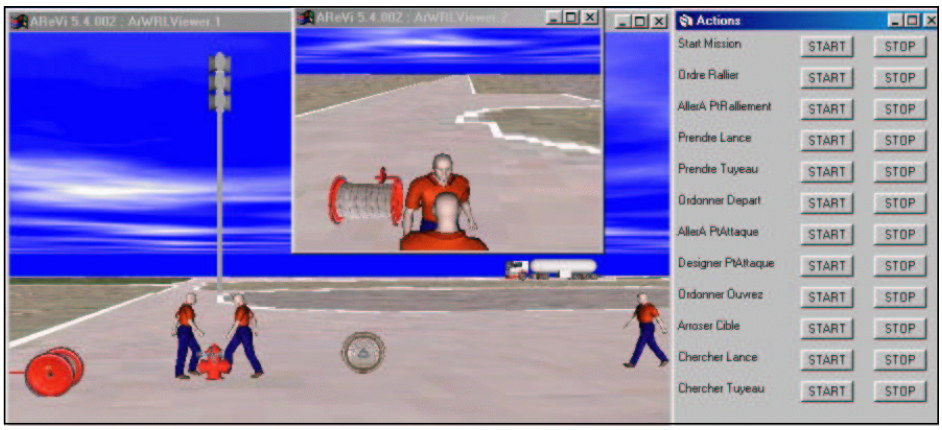


Figure 3.30 : Collaboration entre un humain et son avatar.

membres de l'équipe. Le même mécanisme que pour la perception d'actions entre les agents autonomes est utilisé. Lorsqu'un avatar démarre ou arrête une action, il diffuse un message à l'ensemble des membres de l'équipe. Un agent autonome ne fait donc aucune différence entre les autres agents autonomes de son équipe et les avatars. De

même, un utilisateur ne peut faire la différence entre un agent autonome ou un avatar. L'utilisateur est donc complètement intégré à l'environnement social.

La définition des rôles qui sont joués par des utilisateurs peut être statique. Cette affectation est alors décrite *a priori* dans un scénario pédagogique par le formateur. Cette affectation peut également être dynamique et intervenir au cours de l'exercice. L'utilisateur (formateur ou apprenant) prend alors le contrôle d'un agent en cours de simulation. Cette prise de contrôle revient à rompre la communication de la partie décisionnelle vers la partie opérative. C'est alors l'utilisateur qui envoie des messages à la partie exécutive de l'agent devenu son avatar. Celui-ci ayant suivi l'évolution de la procédure, l'utilisateur peut alors lui rendre son autonomie pour qu'il continue la procédure, si toutefois l'utilisateur l'a bien respectée. Dans le cas contraire, l'agent risque de ne pas pouvoir reprendre le cours normal de la procédure car les actions érronées effectuées par l'utilisateur ont, sans doute, modifiées l'enchaînement temporel des actions et perturbées les autres agents membres de l'équipe.

3.4.3 Fonctions pédagogiques

Le formateur, *via* son avatar, peut inhiber certains de ces comportements réactifs et de ce fait ne plus subir certains phénomènes physiques. Cela lui permet d'aider l'utilisateur ou au contraire de provoquer des dysfonctionnements sans respecter les règles de l'environnement. Il a également la possibilité d'inhiber tout un réseau d'interactions. Ainsi certains types de phénomènes physiques ne sont plus pris en compte dans l'environnement. Cela permet de graduer la difficulté des exercices. Le formateur peut utiliser ces capacités au cours de l'exercice pour l'adapter au comportement de l'apprenant.

La figure 3.29 montre que nous conservons la partie de raisonnement (héritée des agents rationnels) chez l'avatar, non pas pour sélectionner les actions à réaliser, mais pour suivre l'évolution de la procédure et les choix de l'utilisateur. Disposant de cette capacité, l'avatar de l'apprenant peut expliquer, conseiller ou montrer la réalisation d'une tâche à l'utilisateur. Par exemple, l'utilisateur peut demander à son avatar :

- ▷ Quelles sont les actions déjà réalisées ? Le module de décision consulte alors sa base de connaissances.
- ▷ Quelles sont les actions à effectuer maintenant ? Le module de décision propose les actions qu'il a choisies.
- ▷ Pourquoi ne peut-on pas réaliser telle action ? Le module de décision répond alors en explicitant les préconditions non vérifiées de l'action ou par les actions qui la précède dans la procédure et qui n'ont toujours pas été réalisées.
- ▷ Pourquoi faut-il faire telle action ? Le module décisionnel répond alors en explicitant les postconditions de l'action.

L'explicitation des préconditions et des postconditions se fait par une information textuelle associée aux actions. Il est également possible de définir plusieurs types

d'avatar assistant plus ou moins l'utilisateur. Il est par exemple possible de définir un avatar qui vérifie que les préconditions de l'action sélectionnée par l'utilisateur sont vérifiées avant de l'exécuter c'est-à-dire un avatar dont le calcul des plans implicites n'est pas inhibé, l'utilisateur a alors en charge uniquement les actions "métiers" et l'avatar les actions de plus bas niveaux.

3.5 Conclusion

Dans ce chapitre, nous avons défini un environnement virtuel de formation par un système multi-agents hétérogène et ouvert. Le système est hétérogène car il est défini par plusieurs types d'agents et ouvert car il permet d'intégrer les utilisateurs humains. Notre travail porte sur la définition d'un modèle d'organisation et de comportements d'agents que nous appelons MASCARET pour *MultiAgent System for Collaborative and Adaptive Realistic Environment for Training*, ou l'utilisation des systèmes multi-agents pour les environnements *réalistes*, *collaboratifs* et *adaptatifs* pour l'entraînement. Il s'agit, dans un premier temps, d'un modèle générique d'organisation qui permet de concevoir, par dérivation, l'environnement physique ainsi que l'environnement social de manière réaliste et, dans un second temps, la modélisation du comportement collaboratif et adaptatif des agents peuplants ces environnements.

Le premier type d'agent sont les agents réactifs. Ils permettent de simuler les phénomènes physiques de l'environnement. La granularité choisie pour implémenter les différents agents formant un phénomène physique est suffissement fine pour permettre l'interaction entre les différents phénomènes que l'utilisateur doit prendre en considération ; l'environnement de formation est donc *réaliste*. Le second type sont les agents rationnels qui représentent les personnages autonomes. Ils ont aussi un comportement réactif qui leur permet d'être intégré dans l'environnement physique. Leur comportement est *collaboratif* car ils travaillent en équipe, chacun jouant un rôle dont les responsabilités et les actions sont clairement définies et tous partageant les connaissances sur les missions qui peuvent leur être attribuées. Leur module de décision repose sur un mécanisme de sélection d'actions en fonction de l'évolution de la mission en cours et des actions des autres tout en sachant s'adapter à leur environnement dynamique : leur comportement est donc *adaptatif*. Enfin, nous proposons d'intégrer l'utilisateur dans le système multi-agents *via* son avatar. Un avatar est un agent rationnel autonome et peut dès lors s'intégrer à l'environnement social (jouer un rôle dans une équipe) et dans son environnement physique (le subir et le manipuler) ; son comportement est donc *collaboratif* et *adaptatif*.

La suite de notre travail est de montrer comment ce modèle permet de réaliser un outil de formation. Notre cas d'étude est la formation à la gestion opérationnelle et aux commandement pour les officiers sapeurs-pompiers. Nous montrons donc dans le chapitre suivant la réalisation de SécuRéVi, un environnement virtuel de formation pour la sécurité civile et fondé sur MASCARET.

Chapitre 4

L'application : SécuRéVi

Dans le chapitre précédent, nous avons présenté le modèle MASCARET qui permet la conception d'environnements virtuels de formation pour le travail collaboratif. Dans ce modèle, l'environnement des apprenants est composé d'agents autonomes modélisant l'environnement physique et l'environnement social. Les utilisateurs participent à l'environnement *via* leurs avatars également considérés comme des agents autonomes.

L'application SécuRéVi (Sécurité et Réalité Virtuelle) est une application de MASCARET à la sécurité civile. Elle permet d'aider à la formation des officiers sapeurs-pompiers à la gestion opérationnelle et au commandement. L'environnement physique est alors constitué du site dans lequel se situe l'exercice ainsi que les phénomènes physiques (feu, fumée, jet d'eau ...) pouvant y intervenir. Les apprenants jouent les rôles des différents chefs de groupes intervenant lors de l'incident et le formateur participe à la simulation pour provoquer des dysfonctionnements, aider les apprenants ou jouer un rôle dans une équipe. Les équipes sont les agrès[1] tels que les FPT, VSAB... dont les membres sont joués par des agents autonomes. Les apprenants doivent alors suivre un plan d'intervention, ordonner la réalisation de manœuvres aux agrès sous leurs ordres et rapporter la situation à leur supérieur hiérarchique.

Dans ce chapitre, nous présentons, dans un premier temps, les différents outils informatiques déjà utilisés par les sapeurs-pompiers au cours de leur formation. Si cette étude dénombre une multitude d'applications, elle montre également les lacunes de ces propositions et justifie le développement de SécuRéVi qui est présenté dans la seconde partie de ce chapitre. Nous y détaillons les modèles d'environnement physique (phénomènes physiques) et d'environnement social (équipes, procédures...) ainsi que les outils qui nous ont permis de les réaliser. Enfin dans la dernière section de ce chapitre, nous montrons un scénario type d'exercice avec SécuRéVi.

[1] Dans le vocabulaire de la sécurité civile, le terme agrès désigne une équipe de trois à cinq sapeurs-pompiers manipulant un type d'outil (lance, appareil de mesures...)

4.1 Simulation pour la formation des sapeurs-pompiers

Les sapeurs-pompiers utilisent déjà les outils informatiques pour l'enseignement et la simulation de situations de crise. Ces outils sont de différents types et nous les classons selon l'approche choisie. Le premier type d'outil regroupe ceux fondés sur une approche *"classique"* de la simulation. Ils utilisent des bases de données et des modèles numériques pour simuler l'évolution d'un phénomène. La deuxième approche est l'approche *"agent"*, fondée sur les interactions entre des entités qui selon les applications, sont des utilisateurs, des logiciels d'aide à la décision ou encore des agents au sens des systèmes multi-agents. Enfin le dernier type représente les outils adoptant une approche *"réalité virtuelle"* et proposent des capacités d'immersion, de navigation et d'interaction avec l'environnement simulé.

4.1.1 Approche *"classique"*

Différents organismes (CANUTEC[2], INRS[3]) ont dressé des fiches toxicologiques sur les produits chimiques couramment fabriqués, entreposés ou transportés dans le milieu industriel. CANUTEC, qui propose une base de données de plus de 750 000 produits chimiques, est une des bases les plus utilisées dans les différents centre de secours en France et renseigne sur (figure 4.1) :

- ▷ les propriétés chimiques du produit : comportement dans l'air et l'eau, température de stabilité, rayon d'impact en cas d'explosion...
- ▷ les mesures de sûreté et de secours : stockage, moyen de lutte, premiers secours...

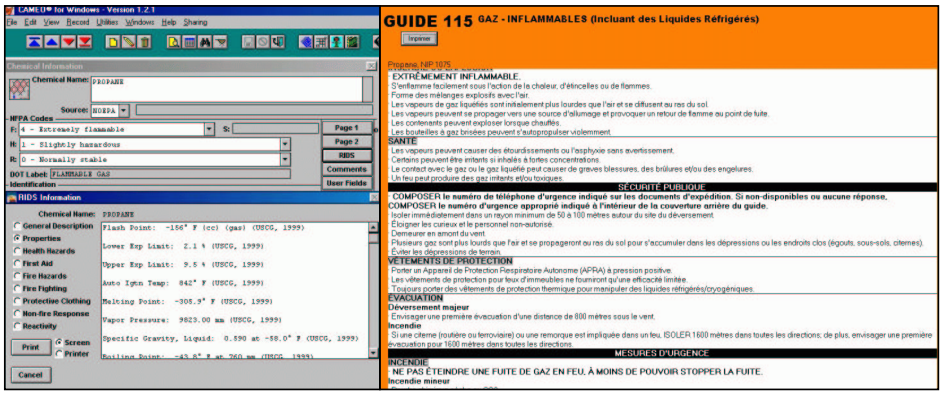


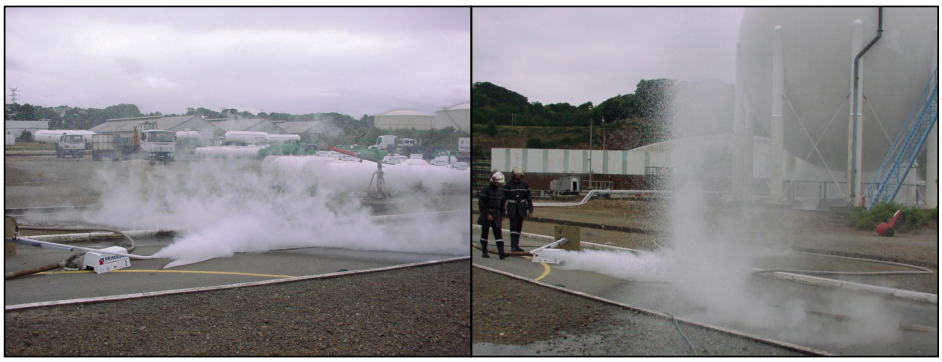
Figure 4.1 : Exemple de visualisation d'informations dans une BDD

[2] Centre canadien d'urgence transport (`http://www.tc.gc.ca/canutec/fr/menu.htm`)

[3] Institut National de Recherche et de Sécurité (`http://www.inrs.fr`)

A partir de ces études, des modèles de propagation des incidents dans l'air ou dans l'eau (fuite de gaz, produit polluant dans une rivière) ont été définis. Ces modèles ne sont pas complètement validés dans la réalité du fait de la complexité de modélisation de l'environnement dans lequel se propage l'incident. L'environnement est alors intégré dans ces modèles sous la forme d'un coefficient de rugosité moyen de type urbain, plaine ou forêt...

Plusieurs types de modèle ont été developpés. Pour la dispersion de gaz plus légers que l'air, ce sont des modèles utilisant une approche globale qui sont utilisés. Pour des gaz plus lourds que l'air, les modèles les plus couramment utilisés sont des modèles eulériens qui découpent l'espace en mailles et calculent la composition de chacune de ces mailles à chaque pas de temps en déterminant la diffusion des polluants entre ces mailles. Des modèles ont également été définis pour décrire les interactions entre plusieurs phénomènes. Comme, par exemple, l'interaction entre une fuite de gaz et une *queue de paon* [4] présentée sur la figure 4.2.



Source : SDIS 29

Figure 4.2 : Interaction entre une queue de paon et une fuite de gaz.

A partir de ces modèles et des bases de données précédemment cités, des outils permettent de simuler la propagation de l'incident. La figure 4.3 (gauche) montre l'utilisation de CAMEO[5] pour la simulation d'une fuite de propane à partir d'une citerne sphérique de 3000 m^3. L'utilisateur renseigne le simulateur sur le type d'incident, les produits incriminés, la quantité, les conditions météorologiques (humidité, vent, couverture nuageuse) ainsi que l'environnement sous la forme d'un coefficient de rugosité. Après calcul, le simulateur propose différentes représentations à l'utilisateur. La figure 4.3 (droite) montre le résultat d'une simulation sous forme de panaches (évolution géographique du nuage de gaz) et sous forme de courbes de saturation du gaz en un point donné.

[4] jet d'eau vertical et mobile formant un mur pour certains produits chimiques.

[5] http://response.restoration.noaa.gov/cameo/cameo.html

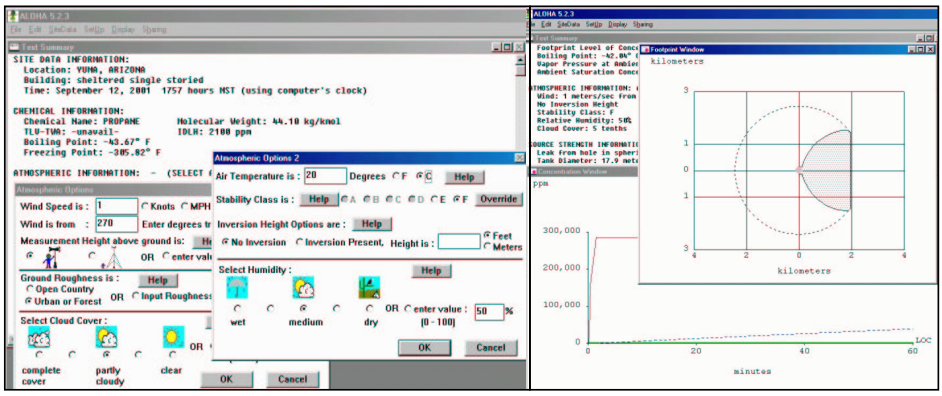


Figure 4.3 : Utilisation de CAMEO

Parmi les autres modèles développés et utilisés par la sécurité civile, il existe des modèles de propagation du feu dans un environnement clos (immeuble) (CFAST[6]) et en forêt ([Galtier et al. 97]). Les modèles de propagation de produits polluants dans l'eau ou la mer sont également utilisés par les sapeurs-pompiers en collaboration avec le CEDRE [7].

Les simulateurs prennent en compte une situation initiale (paramètres du modèle) et calculent une nouvelle situation au bout d'un intervalle de temps donné. Il n'est pas prévu d'interventions de l'utilisateur en cours de calcul (modification des paramètres). Ces outils de simulation ne sont pas interactifs, ils servent aux sapeurs-pompiers comme système d'aide à la décision en cours d'opération, et de compréhension du phénomène en phase d'apprentissage. Ces outils ne fournissent pas une réponse adéquate à l'apprentissage de méthodes de lutte contre les phénomènes étudiés (fuite de gaz, propagation d'un feu).

4.1.2 Approche *"agent"*

La connaissance des produits incriminés et leurs comportements lors de l'incident ne sont pas les seuls concepts que l'apprenant doit acquérir lors de l'apprentissage de la gestion opérationnelle et du commandement. En effet, la gestion de crise est multi-participants et distribuée ; l'apprenant doit appréhender cette facette de la GOC dans son apprentissage. Nous présentons dans cette section quelques outils développés dans cette optique.

[6] *Consolidated Fire And Smoke Transport Model* (http://fast.nist.gov)

[7] Centre de Documentation, de Recherche et d'Expérimentations sur les pollutions accidentelles des eaux (`http://www.ifremer.fr/cedre`)

4.1.2.1 ANTIC

ANTIC [Savall et al. 98] est un logiciel de formation à la gestion opérationnelle et au commandement des officiers sapeurs-pompiers. La caractéristique principale de ce logiciel est la distribution car sa conception est fondée sur un éloignement géographique des participants.

Ce logiciel est composé de quatre éléments :

- le simulateur : il suit le scénario de crise prédéfini et déclenche les événements programmés dans ce scénario. Il centralise également les événements déclenchés par les intervenants (professeurs ou stagiaires) et les diffuse.
- l'espace bilan : lors de l'exercice, les événements sont enregistrés dans une base de données et l'ensemble de la simulation est filmé. Un outil permet au professeur d'indexer des séquences du film pour y revenir lors du bilan.
- l'éditeur de scénario : le logiciel dispose d'un éditeur qui permet au professeur de décrire, hors ligne, le scénario de l'incident. Il y décrit les éléments du monde et les événements à déclencher sous forme de contraintes temporelles, physiques et spatiales.
- les postes opérateurs : ils fournissent aux stagiaires le moyen d'interagir avec la simulation. Le poste professeur fournit des outils tel que l'éditeur de scénario et l'espace debriefing ; il fournit également au professeur un moyen de suivre en temps réel l'évolution de la situation. Les postes stagiaires sont les différents CODIS et les Postes de Commandement de Site (PCS). Ces postes fournissent les outils dont l'étudiant bénéficie en situation réelle tels que des tableurs, simulateurs de modèles physiques (propagation de gaz, de feu...), des outils de gestion cartographique et de communication.

L'architecture générale de ce logiciel est fournie figure 4.4. Dans ce schéma, et dans les autres documents disponibles sur ce logiciel, les actions des utilisateurs sur l'environnement ainsi que la manière dont celui-ci évolue réellement ne sont pas clairement définis. En effet, le simulateur lit les événements programmés dans le scénario et les redistribue à chaque client, mais les actions des apprenants ne sont pas prises en compte et ne modifient en rien le cours de la simulation ; elles sont uniquement enregistrées dans la base de données pour être utilisées lors de la phase de bilan. Au moment de la réalisation de notre étude, ce logiciel était encore en phase d'évaluation par l'INESC (Institut National des Études de la Sécurité Civile). Les auteurs affirment utiliser des techniques multi-agents, mais rien n'a été publié sur la réalisation du simulateur.

Cela semble tout de même être un projet intéressant pour l'apprentissage des outils de communication. Par contre, il ne semble pas possible aux divers intervenants d'avoir une vue de l'environnement simulé. Chacun se construit une représentation mentale de la scène, ce qui peut provoquer des incompréhensions qui sont, dans ce cas précis, un artefact de l'exercice de formation et non le reflet de la réalité opérationnelle. De même, il ne semble pas possible aux professeurs de créer de nouveaux scénarios ou de modifier

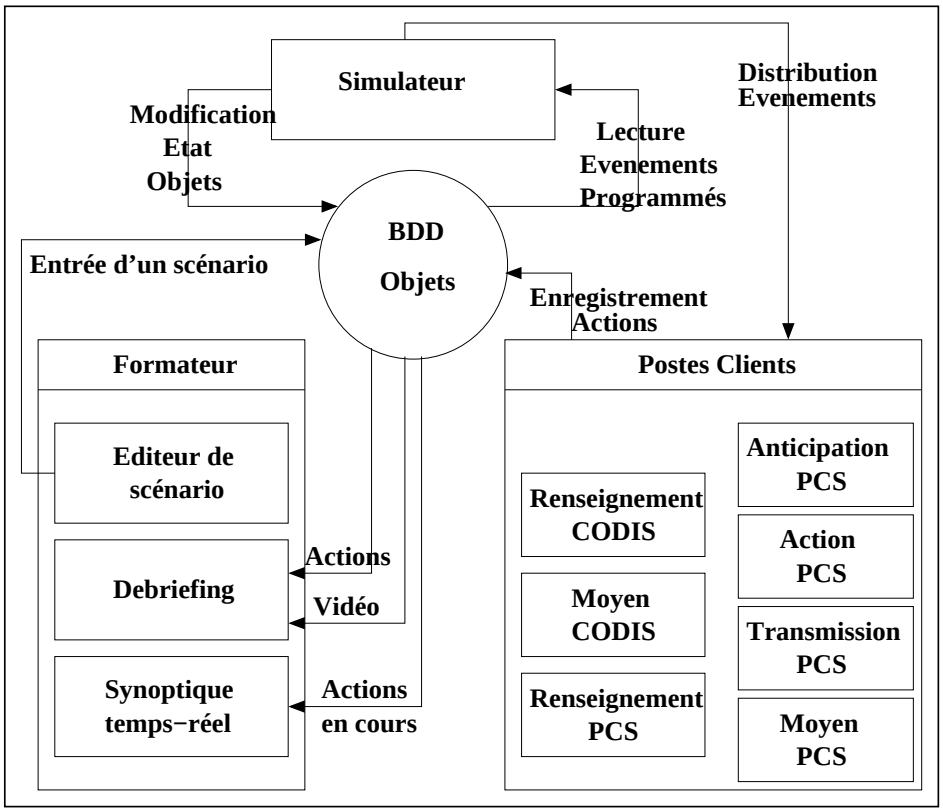


Figure 4.4 : Architecture globale de ANTIC.

le scénario en cours de simulation pour prendre en compte une mauvaise réaction de l'étudiant.

4.1.2.2 Coopération de systèmes d'aide à la décission

Dans [Jaber et al. 98], les auteurs proposent une architecture faisant coopérer différents Systèmes d'Aide à la Décision (SAD) du domaine des feux de forêt. Ce système fait coopérer un logiciel de détection de départ de feu (Bosque) fondé sur un réseau de caméras, un logiciel de gestion de communications entre les participants (Florinus), un logiciel de gestion et d'analyse de données météorologiques (Météo), un logiciel de simulation de propagation du feu (Wilfried) et un SGBD qui gère l'ensemble des informations échangées (MCI). Bosque détecte un départ de feu et décide de fournir l'ensemble des paramètres qu'il a détecté au logiciel de calcul de propagation de feu Wilfried. Ce dernier demande des informations au logiciel de gestion des données météorologiques Météo.

Le principe choisi par les auteurs est d'associer à chaque SAD, un Agent Logiciel Intelligent (ALI). C'est cet agent qui a en charge la coopération entre les SAD (figure 4.5). L'ALI s'occupe alors du traitement des demandes de services, des échanges de

points de vue et des interactions entre les ALI. Toutes ces interactions se font par envois de messages sous forme d'actes de langage.

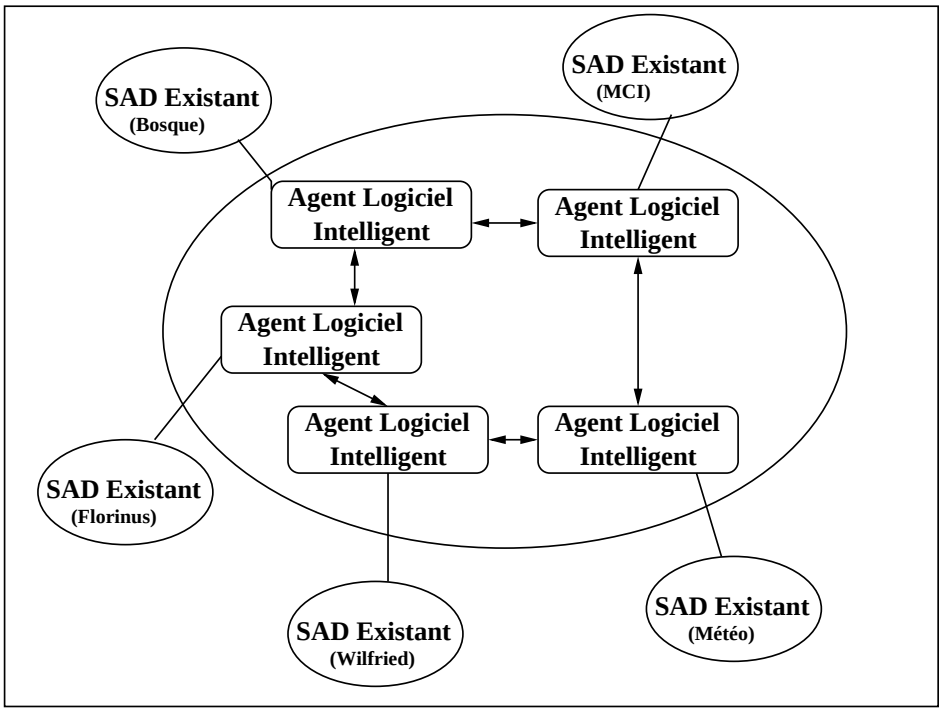


Figure 4.5 : Coopération entre SAD via des ALI (d'après [Jaber et al. 98]).

Un ALI est donc composé des éléments suivants :

- ▷ une couche de communication qui gère l'envoi et le traitement des messages,
- ▷ une couche publique qui met à disposition des autres ALI des informations sur les compétences et actions en cours de l'ALI,
- ▷ une couche privée qui contient toutes les informations sur les fonctionnalités du SAD associé,
- ▷ et une couche d'interface utilisateur.

Lorsqu'un ALI se voit chargé de la responsabilité d'un problème (de l'utilisateur ou de l'environnement), il évalue ses compétences pour en coordonner la résolution. S'il en est incapable, il délègue cette coordination à un autre ALI (en fonction des informations sur les compétences de ses accointances). L'ALI qui prend la responsabilité de résoudre le problème se charge de trouver les agents avec qui il va coopérer pour résoudre le problème (connaissances sur ses accointances) et des intentions exprimées par les autres ALI (envoi de messages).

4.1.2.3 ABIS

ABIS ([Durand et al. 99]), développé au LIP6[8], est un outil qui, comme le précédent, est fondé sur un système d'informations et de communications dans lequel les utilisateurs (différents intervenants de la sécurité civile) communiquent via un outil informatique. Le but de ces travaux est d'analyser les communications afin de les enrichir d'informations sur le contexte de l'émetteur pour éviter une mauvaise interprétation par le récepteur. Les auteurs fournissent un logiciel qui permet aux utilisateurs d'envoyer leurs messages ; le logiciel rajoute automatiquement des informations sur le contexte de l'émission. Du côté du récepteur, le logiciel interprète le message en fonction de son contexte local. Ce mécanisme aide l'utilisateur à se faire une idée réaliste de la situation. Ce système est fondé sur plusieurs systèmes multi-agents (figure 4.6). Les plus importants sont le système multi-agents aspectuel et le système multi-agents morphologique.

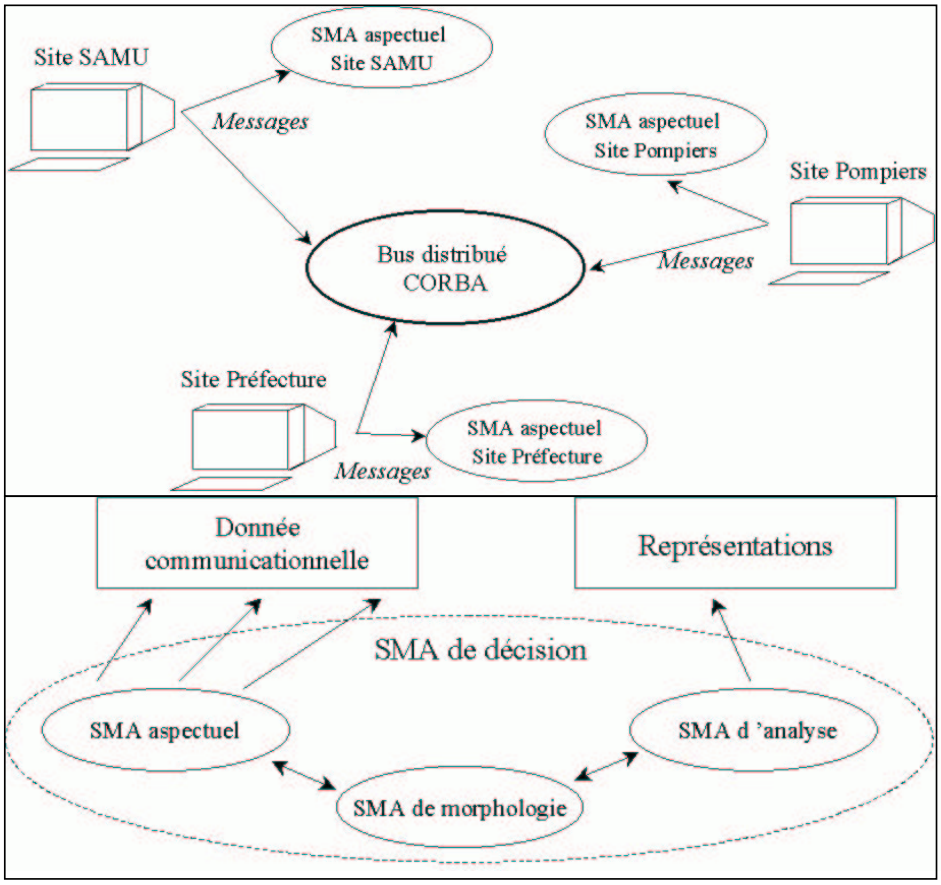


Figure 4.6 : Les système multi-agents dans ABIS (d'après [Boukachour et al. 98]).

Dans ce système, les messages émis sont considérés comme un ensemble de traits sémantiques. Un trait sémantique est un triplet qualification/sujet/intensité. A chaque

[8] Laboratoire d'Informatique de ParisVI

qualification correspond un agent aspectuel dans le système multi-agents aspectuel. En fait, à chaque aspect de la situation correspond un agent et il existe une proximité sémantique entre eux. Elle est codée sous la forme d'une matrice de proximité et permet à chaque agent de connaître ses amis et ses ennemis. Le comportement de ces agents est alors d'observer les messages reçus par l'acteur et de faire évoluer leur état interne en fonction de la proximité sémantique des traits sémantiques contenus dans le message. L'agent aspectuel effectue également des actions bénéfiques ou néfastes envers ses amis et ses ennemis. Pour chaque agent, le système calcule également des critères tels que la vélocité (vitesse à laquelle l'agent a évolué dans son comportement), la suprématie (importance dans le système multi-agents) et la facilité (facilité avec laquelle l'agent a évolué). Ce système multi-agents représente la vision locale que l'acteur a de la situation.

Cette vision fournie par le système multi-agents aspectuel de l'acteur définit un paysage d'agents qui n'est pas compréhensible par l'humain, mais qui dispose tout de même de caractéristiques géométriques ou morphologiques. C'est le rôle du système multi-agents morphologique de faire apparaître, par émergence, ces caractères géométriques. Il s'agit alors des points sensibles de la situation, des points sur lesquels l'attention doit être retenue. Le système compare les différents agents aspectuels ainsi que leurs critères (vélocité, suprématie, facilité) et s'il constate des similitudes entre des agents aspectuels, il génère des agents morphologiques.

4.1.3 Approche *"réalité virtuelle"*

Les outils présentés dans la section précédente ne permettent pas à l'utilisateur de manipuler en ligne l'environnement pour en étudier son comportement. Pour résoudre ce problème, des travaux sont menés pour réaliser des simulateurs interactifs réagissant comme dans la réalité et permettant à l'utilisateur d'avoir un rendu en temps réel des effets de ses actions dans l'environnement.

4.1.3.1 SAM

SAM (Simulateur d'Apprentissage de la Méthode) est un outil de formation adopté par la brigade des sapeurs-pompiers de Paris [Dosne 00]. Ce simulateur a pour vocation l'apprentissage de la Méthode des Raisonnements Tactiques (MRT). Cette méthode, inspirée de celle utilisée dans l'armée, permet au chef des opérations de secours arrivant sur les lieux d'un incident, de prendre sa décision sous forme de manœuvres à réaliser. La MRT est organisée autour de plusieurs points clef. L'un de ces points vise à l'acquisition d'un maximun d'informations sur l'incident. Cette phase de la méthode est très importante pour éviter qu'une mauvaise interprétation de l'incident ne le transforme en situation de crise.

C'est à ce point que SAM apporte une réponse. La situation est modélisée en trois dimensions (bâtiments, feux, victimes) (figure 4.7). L'utilisateur peut alors naviguer dans l'univers pour une meilleure compréhension de la scène. SAM s'utilise également en phase de bilan. Un ensemble de séquences animées permet de montrer l'évolution du sinistre en fonction des choix effectués par les étudiants suivant un scénario pré-établi.

Ce simulateur est interactif dans le sens où l'utilisateur peut naviguer dans la scène, mais l'utilisateur ne peut agir sur cette scène. Seuls quelques choix possibles peuvent être pris en compte (par le concepteur) et les résultats sont rendus sous forme d'animation. Cette méthode de travail se rapproche fortement de celle utilisée actuellement dans la formation des officiers sapeurs-pompiers. L'utilisation de cet outil contribue à une meilleure appréhension de la situation par l'étudiant. Les auteurs posent le principe qu'une scène très réaliste, voire un réalisme exagéré, permet de générer des émotions chez l'étudiant ce qui améliore sa prise de conscience.

Figure 4.7 : Vues de SAM.

4.1.3.2 Shadwell

Certains simulateurs visent la modélisation réaliste de l'environnement. Ainsi une étude a été menée sur la modélisation d'un navire militaire américain qui sert actuellement à la formation des secours sur les bateaux ([Tate et al. 97]). Le but de cette étude est de comparer la formation sur le navire réel et celle sur le modèle en trois dimensions. Dans ce cas, c'est essentiellement l'environnement (le bateau) sur lequel se sont concentrés les efforts de modélisation. En effet, comme on peut le voir figure 4.8, les différentes pièces et accès ont été modélisés. Par contre, la simulation des phénomènes physiques représentés (le feu et la fumée), n'est pas fondée sur un modèle physique générique, mais sur une étude de la propagation de ces phénomènes dans le bateau réel. L'interaction essentielle proposée à l'étudiant est la navigation au sein de l'univers virtuel. Les exercices sont fondés sur une recherche de chemins et d'outils dans l'environnement. A terme, l'étudiant devrait pouvoir agir sur l'incident, mais ces interactions ne sont pas clairement explicitées dans les documents sur ce sujet.

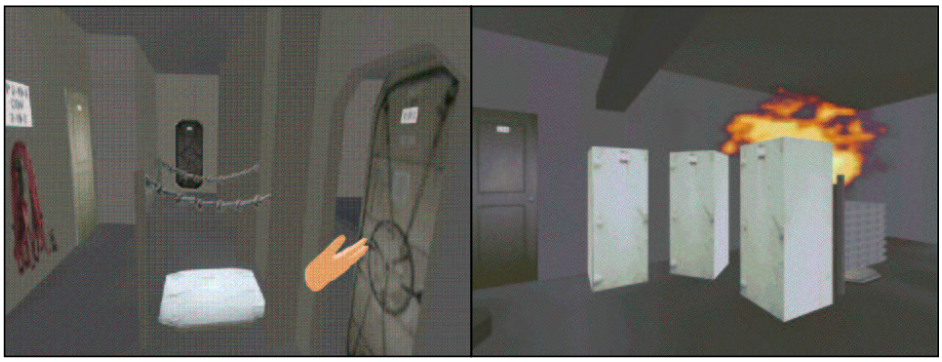


Figure 4.8 : Modélisation du SHADWELL et feu dans un compartiment.

4.1.3.3 *Walkthru-CFAST*

D'autres travaux se basent justement sur des modèles physiques pour simuler un environnement réaliste et interactif. C'est le cas des travaux de [Bukowski et al. 97]. L'application développée permet à l'utilisateur de naviguer dans un environnement clos, complexe et de grande échelle. Ce logiciel se fonde sur une technique classique en réalité virtuelle. Ainsi l'environnement est divisé en mailles auxquelles sont associées une liste de mailles visibles et de mailles voisines. Connaissant la direction et la vitesse de déplacement de l'utilisateur, le système peut optimiser le chargement des scènes. Un tel système peut alors accueillir des modèles physiques basés eux-mêmes sur une division de l'espace en petites surfaces ou volumes. Les auteurs associent donc un logiciel de navigation et un modèle de propagation de feu, gaz et fumée (CFAST). Dans ce modèle l'environnement est divisé en plusieurs volumes. Pour chaque volume, sont calculés une concentration pour toutes les composantes prises en compte (feu, fumée, gaz...) et un transfert de ces composantes vers les volumes voisins. Ces calculs sont effectués par un moteur de résolution d'équations différentielles. L'application résultante permet de modifier en temps réel l'architecture interne de l'environnement (destruction de mur...). Les différents modèles (visibilité de maille à maille et propagation du feu) sont recalculés en prenant en compte cette modification de paramètres. Notons que cette modification ne porte que sur la géométrie des mailles et non sur le modèle de propagation.

4.1.3.4 ADMS

La réalité virtuelle permet d'utiliser des outils pour améliorer l'immersion de l'utilisateur dans la scène. ADMS (Advanced Disaster Management Simulator)[9] est un simulateur qui utilise les technologies haut de gamme de la réalité virtuelle (calculateur Silicon Graphics, écran 180° de champ de vision). Le simulateur se décline en deux versions. La première est une version *bureau* (fonctionnant sur station de travail Silicon Graphics) qui permet d'apprendre les différentes procédures ; la deuxième version est

[9] `http://www.etcusa.com/admsmain.htm`

la version *immersive* qui permet à l'étudiant de donner ses ordres aux équipes luttant contre l'incident (figure 4.9).

Figure 4.9 : Vue de ADMS.

4.1.3.5 Vector Command

Le dernier type de logiciel existant vise à intégrer l'ensemble des fonctionnalités exposées précédemment. Vector Command[10] est un logiciel déjà utilisé dans la formation de sapeurs-pompiers dans plusieurs pays anglo-saxons. La thématique principale de ce logiciel est "la défaillance du commandement" ; une mauvaise perception ou interprétation de l'incident peut conduire à une situation de crise. Ce logiciel permet de placer un officier en situation, il doit alors naviguer dans l'environnement, communiquer avec différents intervenants, prendre des décisions et les transmettre aux personnels concernés. Le logiciel s'articule autour d'un simulateur qui fait évoluer la scène et offre un rendu 3D à l'utilisateur (figure 4.10).

L'apprentissage avec Vector Command se déroule en quatre phases :

- ▷ vérification des connaissances de l'étudiant par le biais de QCM,
- ▷ mise en situation (description de la scène par le formateur),
- ▷ simulation,
- ▷ *debriefing*.

Ce n'est pas l'apprenant qui manipule le système, mais son formateur. Chaque étudiant, ou groupe d'étudiants, donne ses ordres à un formateur qui les répercute

[10] `http://www.vectorcommand.com`

Figure 4.10 : Interface et visualisation de scène sous Vector Command.

dans la simulation. Le rendu de la scène n'a donc pas besoin d'être hyper-réaliste ni immersif. Ainsi les phénomènes physiques sont représentés de manière simpliste et la zone de rendu tridimensionnel de la scène n'a pas une place primordiale dans la simulation.

En cours de simulation l'apprenant (via son formateur) peut :

- ▷ naviguer dans la scène par le biais d'interfaces simples (bouton de direction dans l'interface ou choix de caméra). Lorsqu'il rencontre un personnage, il peut lui donner un ordre. Ces ordres sont prédéfinis par le concepteur du scénario.
- ▷ visualiser des informations tels que les plans, l'état des ressources... Il peut également demander des ressources supplémentaires.
- ▷ dialoguer avec certains intervenants. Ces intervenants, ainsi que le dialogue, sont également prédéfinis par le concepteur du scénario. L'utilisateur a accès à une interface par laquelle il choisit, dans une liste, l'intervenant avec lequel il veut dialoguer. Le dialogue possible est alors décrit par un arbre dans lequel l'utilisateur fait des choix.

Un des points forts de cet outil est la phase de *debriefing*. L'ensemble de la simulation est enregistrée sous forme de film. Le formateur peut alors rejouer ce film, faire des arrêts sur images et expliquer à l'apprenant ce qui s'est passé lors de la simulation. Mais le formateur a également accès à des informations de plus haut niveau d'analyse telles que le taux d'occupation des ressources au cours du temps.

Le simulateur est fondé sur un moteur d'intelligence artificielle qui calcule les comportements des entités (pompiers, véhicules ...) et la propagation des phénomènes physiques. La propagation de ces phénomènes n'est pas régie par un véritable modèle physique, mais par un modèle *ad hoc* au scénario. Les entités peuplant l'univers sont des entités autonomes dans le sens où leur comportement intègre une phase de perception de l'environnement, de décision et d'action. Cela veut dire que chaque entité modifie son comportement en fonction de ce qu'elle observe dans l'environnement. Par exemple, un pompier ayant reçu l'ordre d'attaquer un feu au second étage d'un immeuble peut modifier son comportement s'il observe un blessé ; il choisira alors d'évacuer ce blessé avant de réaliser sa mission. Par contre, ce genre de comportement est également

prédéfini par le concepteur du scénario.

Comme nous avons pu le voir dans les explications précédentes, Vector Command s'articule autour de la notion de scénarios. A l'heure où ce paragraphe est écrit, 11 scénarios ont été définis dans Vector Command (feu dans un pavillon, HLM, accident de la route...). L'écriture d'un scénario se fait d'abord en C++. Ce n'est donc que le concepteur du logiciel qui peut créer un nouveau scénario. Pour faciliter la description des comportements des entités, les concepteurs ont défini un langage d'un niveau sémantique plus élevé que le C++. Pour l'instant aucune documentation ou information n'est disponible sur ce langage. Au dire des concepteurs, ce qui prend le plus de temps dans la description d'un nouveau scénario n'est pas la réalisation des comportements des entités mais la description des interfaces, modèles 3D et dialogues avec les intervenants. Il n'est a priori pas à l'ordre du jour de fournir au formateur un outil lui permettant de créer un nouveau scénario ou du moins de paramétrer un scénario existant.

L'utilisation de ce logiciel doit être dirigée par une équipe de formateurs qui définit l'utilisation de Vector Command dans un programme pédagogique. L'apprenant passe par le formateur pour interagir avec le système, le formateur est donc toujours présent dans la phase d'apprentissage pour fournir des recommandations et des explications à l'apprenant. L'étudiant a également accès aux divers supports de cours et documents utiles lors d'interventions. Par contre, le formateur ne peut sortir du cadre prévu par les concepteurs et réaliser lui-même ses sujets d'exercices.

4.2 De MASCARET à SécuRéVi

L'étude précédente montre les lacunes des outils existants en terme de représentation et de simulation de l'environnement ainsi qu'en terme d'immersion des apprenants et des formateurs dans cet environnement. Nous proposons donc de concevoir un environnement virtuel de formation pour la sécurité civile [Querrec et al. 01b], fondé sur le modèle MASCARET. Nous montrons dans cette section, comment l'application SécuRéVi dérive de MASCARET, c'est-à-dire comment nous concevons l'environnement physique dans SécuRéVi à partir des réseaux d'interactions et comment les équipes de pompiers sont fondées sur le modèle d'équipe de MASCARET. L'objectif de cette section n'est pas de lister de manière exhaustive l'ensemble des fonctionnalités de SécuRéVi, mais plutôt d'en montrer la méthodologie de conception en plusieurs phases de modélisation. Cette méthodologie repose sur 4 grandes étapes : la modélisation du site géographique, la modélisation des phénomènes physiques, la modélisation des acteurs virtuels et la modélisation des équipes. Enfin, pour illustrer cette méthodologie, nous prenons l'exemple d'un exercice type de fuite de gaz dans un site industriel (site classé Seveso II).

4.2.1 Modélisation des sites

La première étape est la modèlisation des sites. Cette tâche étant fastidieuse, nous veillons à la réutisabilité des éléments déjà réalisés, ce qui se traduit par l'utilisation d'un format de représentation standard et de la création d'une classification et d'une bibliothèque.

Classification

Tous les éléments reconstitués sont des éléments géoréférencés (`GeoEntity`, voir figure 3.8 page 62) c'est-à-dire qu'ils disposent au minimum d'une position et d'une orientation. Nous avons choisi d'organiser ces éléments, d'une part pour faciliter le développement futur d'outils d'aide à la conception de sites et d'autre part pour simplifier le comportement des agents. Le critère choisi pour structurer l'ensemble des éléments est lié aux capacités de déplacement. A partir des entités géoréférencées de MASCARET, nous avons donc défini trois classes d'entités (figure 4.11) :

- ▷ Les entités mobiles (`GeoMobile`) disposent de capacités pour se déplacer de manière autonome. Elles sont caractérisées par des critères de masse, de boîte englobante et d'attributs de cinématique (vitesse, accélération ...) et concernent essentiellement les véhicules, les humanoïdes, les particules...
- ▷ Les entités déplaçables (`GeoMovable`) disposent des mêmes caractéristiques que les entités précédentes, mais elles ne disposent pas des capacités de se déplacer toutes seules ; elles subissent les forces des éléments (entités mobiles) auxquels elles sont liées. Il s'agit essentiellement des outils des sapeurs-pompiers, de certains mobiliers urbains et des réservoirs.
- ▷ Les éléments statiques (`GeoStatic`) ne peuvent bouger ni d'eux-même ni par les forces exercées par d'autres entités. Elles ont une masse infinie, mais elles disposent tout de même d'une boîte englobante pour le calcul d'évitement d'obstacles. Il s'agit essentiellement des bâtiments et des gros réservoirs.

Comme nous l'avons déjà vu dans l'exemple du chapitre précédent (section 3.2.5 page 60), les entités géoréférencées participent par défaut à deux réseaux d'interactions : un réseau de mobilité et un réseau de collision. Les éléments mobiles peuvent alors jouer n'importe quel rôle dans les interactions de collision et de mobilité. Par contre, les éléments déplaçables ne peuvent jouer qu'un rôle cible dans les interactions de mobilité alors qu'ils peuvent jouer n'importe quel rôle dans les interactions de collision. Enfin, les éléments statiques ne peuvent jouer qu'un rôle cible dans les interactions de collision ; ils ne jouent aucun rôle dans les interactions de mobilité.

Modélisation et rendu

Les méthodes utilisées ici sont dépendantes des contraintes fixées par les environnements virtuels de formation telles que la crédibilité, la nécessité d'un calcul temps

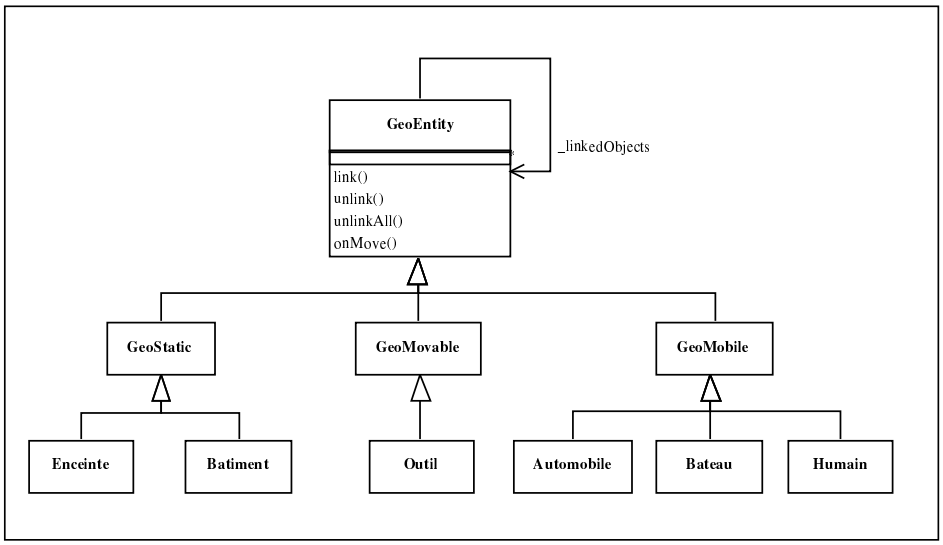


Figure 4.11 : Entités géoréférencées dans SécuRéVi.

réel mais également celles découlant de la plate-forme de réalité virtuelle utilisée. Nous expliquons plus loin les caractéristiques de celle que nous avons choisie, mais nous supposons pour l'instant que nous avons accès aux fonctionnalités classiques d'OpenGL[11].

Les utilisateurs doivent reconnaître l'environnement ; il est donc impossible d'utiliser les techniques automatiques de génération de paysages urbains, tels que VUEMS ([Donikian 97]). La technique que nous utilisons est fondée sur la construction des modèles géométriques à partir de plans fournis par les industriels et par la sécurité civile. Cette reconstitution est manuelle et nous utilisons pour cela des modeleurs tels que 3DSMax[12]. Chaque élément de l'environnement est texturé par des photographies prises sur le site et enregistré au format VRML[13]. Les textures prises doivent être de bonne qualité car les utilisateurs naviguent à l'intérieur de la scène au niveau du sol ce qui demande de passer beaucoup de temps dans le traitement des textures. Cette technique nécessite beaucoup de ressources en capacité mémoire ; cependant, dans le cadre de SécuRéVi, les environnements reconstruits ne sont pas très étendus : il n'est pas question de reproduire une ville entière, mais uniquement des sites, ou au maximum un quartier. La quantité de données reste donc raisonnable comparée aux ressources matérielles disponibles actuellement et ne pénalise pas le calcul en temps réel de la simulation. La figure 4.12 montre la reconstruction d'un site industriel. Cette scène représente environ 50 entités géoréférencées pour un total de 50 000 triangles.

[11] `http://www.opengl.org`

[12] `http://www.ktx.com`

[13] `http://www.vrml.org`

Figure 4.12 : Reconstruction 3D d'un site industriel.

4.2.2 Modélisation des phénomènes physiques

La seconde phase est celle de la modélisation des phénomènes physiques. Dans le cadre de SÉCURÉVI, ceux-ci doivent être réalistes voire exagérés pour générer une sensation d'immersion chez l'apprenant. SÉCURÉVI est destiné, dans un premier temps, aux scènes d'extérieur s'étendant sur un site industriel. Les agents de l'environnement sont alors des flammes, des particules de gaz, des jets d'eau... Le but de ce paragraphe est de montrer la méthode de réalisation des réseaux d'interactions de SÉCURÉVI par dérivation de ceux définis dans MASCARET (voir figure 3.4 page 57) ainsi que les différents types d'agent participant à ces réseaux.

Dans SÉCURÉVI, plusieurs types de réseaux d'interactions sont à développer, mais beaucoup d'entre eux sont à définir en fonction de l'implémentation de nouveaux scénarios. Par contre, une fois implémentés, ils alimentent une bibliothèque et peuvent servir à nouveau dans d'autres exercices. Dans l'exercice d'une fuite de gaz dans un site industriel, le phénomène majeur à prendre en compte est la propagation du nuage de gaz (déplacement et collision) ainsi que l'effet toxique qu'il peut avoir sur les êtres vivants. Cet exemple est en partie traité au chapitre précédent où nous illustrons les comportements réactifs impliqués lors d'une interaction particulière (section 3.2.5 page 60) ; notre objectif ici est de montrer ce que cet exemple induit dans la conception des structures et entités organisationnelles.

Dans cet exemple, trois réseaux d'interactions sont mis en jeux. Le premier est le réseau de mobilité, il fait intervenir le vent qui est une source et un recruteur ainsi que les particules de gaz, qui elles ne sont que des cibles. Le second réseau est celui de toxicité dans lequel les humains participent en jouant les rôles de recruteur et de cible. Les particules de gaz y participent également en y jouant le rôle cible. Enfin le troisième réseau est celui des collisions dans lequel les murs d'eau et la

queue de paon participent en tant que source et recruteur. Les particules de gaz y participent également en jouant le rôle source. Tous ces phénomènes sont des réseaux d'interactions. La figure 4.13 montre la manière dont le réseau `CollideNet` dérive d'`InteractionNet` de MASCARET. Un `CollideNet` dispose alors des rôles `Source`, `Target` et `Recruiting`, mais définit en plus un rôle `Collider`. Ce rôle décrit à son tour, un comportement réactif `CollideBehavior`. Ce mécanisme est utilisé pour tous les réseaux d'interactions qui forment l'environnement physique.

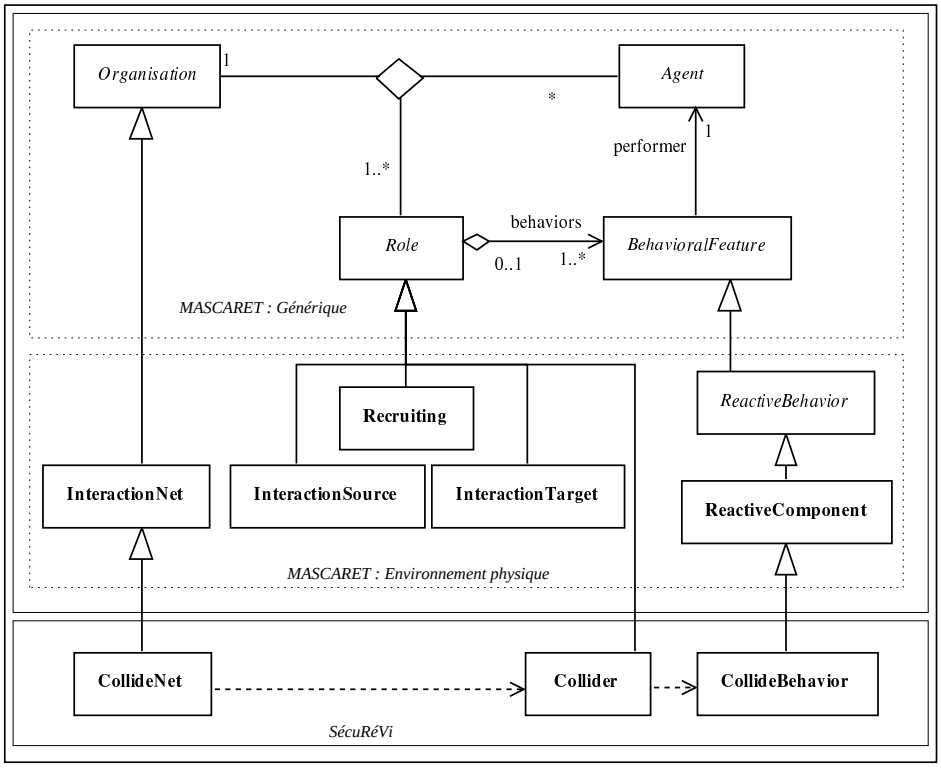


Figure 4.13 : Les réseaux d'interactions dans SécuRéVi.

Dans un scénario pédagogique on instancie chacun de ces réseaux, mais l'affectation des rôles aux agents se fait de manière dynamique. Dans le cas de l'exercice de fuite de gaz, seuls quelques agents sont créés au début du scénario. Ainsi, le vent s'enregistre dès le début comme `Source`, `Recruiting` ainsi que `Mobile` dans le réseau `Mobility` et les citernes de gaz jouent les rôles de `Target` et `Recruiting` dans `ThermicTransfert`. C'est lors de l'implémentation des agents que le concepteur désigne les rôles que chaque agent peut jouer, car l'affectation des rôles est ici fixe. Les jets d'eau et les particules de gaz sont instanciées au cours de la simulation, et c'est à ce moment qu'ils commencent à participer aux réseaux d'interactions.

Différentes techniques existent pour le rendu visuel des phénomènes physiques, mais elles sont liées au modèle de calcul du phénomène qui est souvent un modèle global. Cela les rend inutilisables dans notre cas car ces techniques ne permettent pas l'interaction des utilisateurs. De plus, pour un certain nombre, elles ne sont pas

(ou difficilement) calculables en temps réel, ce qui pose un problème si l'on veut simuler plusieurs types de phénomène dans la même simulation. Nous ne pouvons donc pas utiliser ces techniques dans SécuRéVi. Nous utilisons des systèmes à particules [Stam 00] pour représenter les jets d'eau ainsi que les explosions, et des billboards avec textures transparentes et animées [Harris et al. 01] pour représenter les particules de gaz et les feux. La figure 3.7 du chapitre précédent page 61, montre le rendu visuel de la collision entre un nuage de gaz et une queue de paon, ce rendu peut être comparé avec la scène réelle illustrée figure 4.2 page 97 de ce chapitre.

4.2.3 Modélisation des personnages autonomes

Dans SÉCURÉVI, les agents participant à l'environnement social sont des humanoïdes. Dans le cadre de l'exercice type proposé, les personnages sont les agents de sécurité du site et les sapeurs-pompiers. Ce sont des instances de classes d'agents héritant de la classe d'humanoïde déjà présentée (voir figure 3.26 page 87). Cette classe décrit les compétences classiques d'un humain telles que marcher ou communiquer et la conception de nouvelles classes d'agent à partir de celle-ci se justifie par l'ajout de nouvelles compétences techniques telles que nager pour un plongeur, et non par l'ajout de responsabilités à un agent (représentées par les rôles des organisations).

Différentes techniques existent pour la modélisation des humanoïdes et de leurs actions. THOMAS [Thomas 99] en propose quelques unes et utilise la norme H-ANIM[14]. Nous avons utilisé le logiciel POSER[15] (conforme à la norme H-ANIM) pour créer les personnages et les mouvements. Un mouvement est décrit par des positions clés de chaque membre de l'humain ; les positions intermédiaires sont calculées par interpolation. Un mouvement est alors composé de plusieurs poses et l'animation du personnage s'obtient alors en affichant les poses de la première à la dernière. Les différentes poses peuvent également s'obtenir à partir de captures de mouvements comme avec le système VICON[16]. Cette technique commence à être utilisée aujourd'hui en réalité virtuelle, mais nous ne disposons pas de cette technologie coûteuse pour SÉCURÉVI. Une autre solution peut être de calculer les modèles biomécaniques [Beaupied et al. 01] issus de ces captures de mouvements, malheureusement ces modèles ne sont pas encore utilisables pour la réalité virtuelle.

Techniquement, dans SÉCURÉVI, chaque pose est définie dans un fichier VRML. La plate-forme que nous utilisons ne permet pas d'accéder aux noeuds VRML, il faut donc stocker en mémoire chacune des poses. Nous comptons sur les évolutions de la plate-forme pour palier ce problème ce qui nous permettra de ne charger qu'une fois le fichier VRML puis d'accéder aux différents nœuds pour leur donner leurs nouvelles

[14] `http://www.h-anim.org`

[15] `http://www.curiouslabs.com/products/poser4/index.html`

[16] `http://www.vicon.com`

valeurs en fonction de l'animation pré-calculée. La figure 4.14 montre deux poses pour un des personnages ainsi recréé.

Figure 4.14 : Reconstruction 3D d'un personnage.

4.2.4 Modélisation des équipes

SécuRéVi s'adresse au chef de groupe (GOC 4) qui commande 2 à 4 agrès (voir figure 1.1 page 5). Le but de SécuRéVi est de former l'apprenant à la prise de décision en situation opérationnelle, les actions des agents doivent donc avoir une représentation suffisement réaliste pour favoriser l'immersion de l'utilisateur, mais pas forcément très précise.

Dans le cas d'une fuite de gaz sur un site industriel, l'objectif est de mettre les populations en sécurité, réduire la propagation de l'incident et protéger les citernes en cas d'inflammation du nuage. Pour cela le PPI prévoit l'engagement des moyens tels qu'illustrés sur le figure 4.15. La première équipe engagée est l'équipe CMIC[17] dont le rôle est d'acquérir les informations sur les produits incriminés et sur leur propagation. Le plan prévoit également l'engagement de plusieurs équipes FPT dont l'objectif est de ralentir la progression du nuage de gaz à l'aide de murs d'eau et de protéger les réservoirs en les arrosant. Cela demande des ressources en eau et matériel (lances, tuyaux, dévidoirs), c'est pourquoi l'intervention d'une cellule dévidoir est également prévue pour fournir ces ressources. Toutes ces équipes sont composées de trois à cinq membres et pour chacune d'entre elles il existe un manuel décrivant l'ensemble des procédures qu'elle peut être amenée à exécuter.

[17] Cellule Mobile d'Intervention Chimique

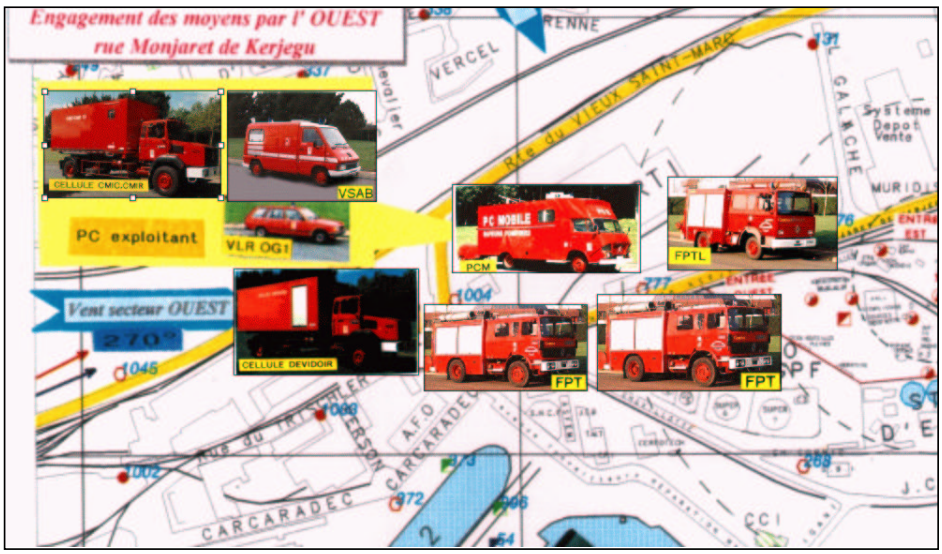


Figure 4.15 : Engagement des moyens, extrait d'un PPI (source SDIS 29).

Toutes ces équipes dérivent des modèles définis dans MASCARET, comme le montre la figure 4.16 pour l'exemple d'une équipe FPT. Ainsi, dans cet exemple, l'équipe `FPT` hérite de la classe `Team` de MASCARET. Cette équipe instancie trois rôles (`Chef`, `SousChef` et `Servant`) héritant de `TeamRole` et définissant des actions telles que `SeTenirPret` et `BasculerLaFleche` héritant de `GDAction`. Le travail du concepteur dans SÉCURÉVI est donc essentiellement d'implémenter ces éléments en héritant de MASCARET.

Selon le scénario de l'exercice, ces structures organisationnelles peuvent être instanciées au début de la simulation ou de manière dynamique. Les instances organisationnelles sont alors les équipes FPT ou CMIC (équipe FPT n°1, équipe FPT n°2, équipe CMIC n°1 ...). Pour chaque instance d'équipe sont également instanciés les rôles (`Chef`, `SousChef`...). Par contre, dans le cas des sapeurs-pompiers, l'affectation des agents aux rôles est décrite dans le scénario, chaque agent sait, dès le début, dans quelle instance d'équipe il joue et quel est son rôle : il s'agit du mode de fonctionnement réel des sapeurs-pompiers. Pour chaque agent participant à une équipe sont alors intanciées les actions définies par les rôles. Les équipes définissent également des missions qui sont décritent dans des documents pédagogiques des sapeurs-pompiers. Chacune de ces missions sont des instances de `Procedure`, elles sont instanciées lors de la création des équipes, mais le transfert des contraintes qu'elles décrivent dans la base de faits des agents n'est effectif que lors de l'exécution de l'une d'entre elles.

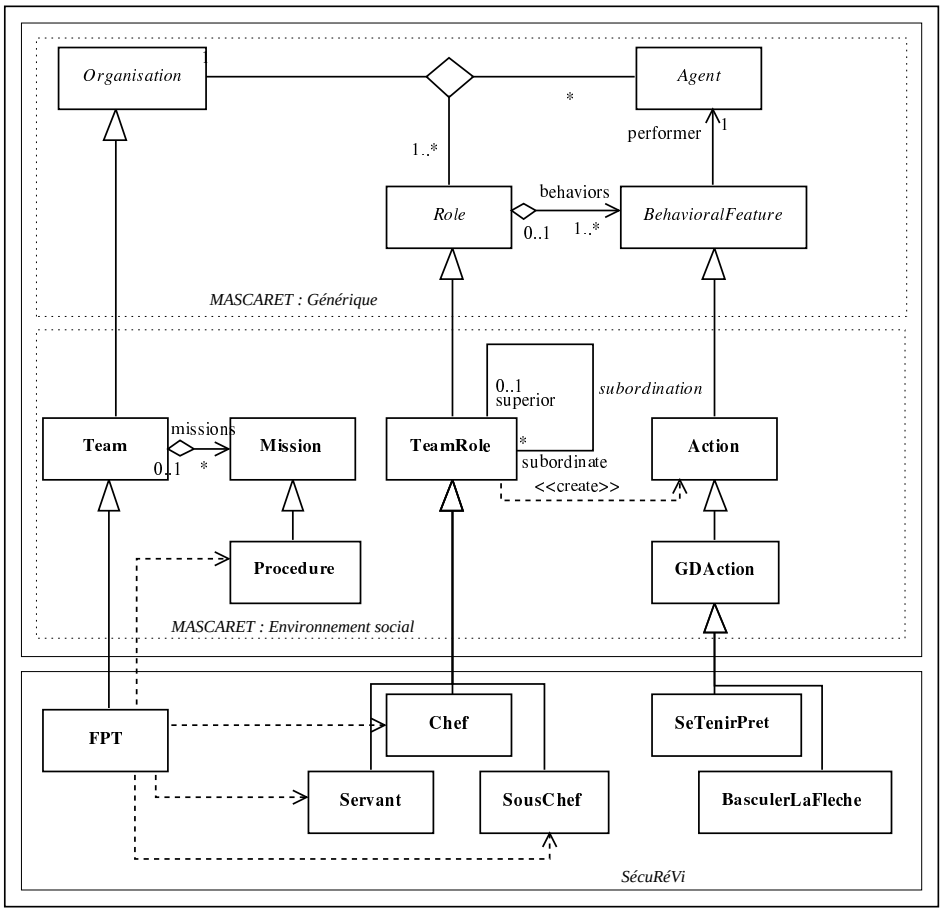


Figure 4.16 : Organisation en réseau.

4.2.5 La plate-forme ARéVi/oRis

Pour implémenter SécuRéVi, nous avons utilisé la plateforme ARéVi/oRis car elle dispose de caractéristiques intéressantes pour l'implémentation des modèles proposés. De plus, comme elle est développée au sein de notre laboratoire, il nous est possible d'y intégrer les techniques dont nous avons besoin (particules, textures transparentes...) et qui ne sont pas forcément disponibles dans d'autres plates-formes.

4.2.5.1 oRis

oRis ([Harrouet 00],[Harrouet et al. 02]) est un environnement de simulation interactive : c'est un langage de programmation par objets actifs et un environnement d'exécution. Ces caractéristiques en font une plate-forme généraliste pour l'implémentation de systèmes multi-agents, plus particulièrement dédiée à la simulation.

C'est un langage dynamiquement interprété, à granularité instance qui permet d'intervenir en cours de simulation pour observer le système multi-agents, interagir avec les agents ou sur l'environnement et les modifier en ligne. Il a de nombreuses similitudes avec les langages C++ et JAVA, ce qui facilite son apprentissage. De plus, ORIS dispose d'une interface vers ces langages, ce qui permet de compiler certains *paquetages* pour améliorer les performances d'exécution. En oRis un système multi-agents est composé d'agents (à la base : des objets actifs) dont l'environnement est constitué d'objets, éventuellement situés dans l'espace (2D ou 3D) et le temps. ORIS offre une solution homogène pour les interactions qu'elles soient implémentées par appel de méthode, lien réflexe, ou envoi de messages (en point-à-point ou diffusion, avec traitement immédiat ou différé par son destinataire).

oRis implémente trois modes de gestion des flots d'activité :

- ▷ Un objet actif dispose d'une méthode `main` représentant le point d'entrée de son comportement. Lorsqu'une instance est créée, cette méthode est immédiatement prête à s'exécuter. Quand la fin de la méthode est atteinte, elle est automatiquement relancée à son début.
- ▷ La primitive `start` permet de dédoubler un flot d'exécution, elle sert principalement à attribuer plusieurs activités à un même agent.
- ▷ L'intervention de l'utilisateur (externe à tout bloc de code) utilise également le mot-clef `start`.

L'ordonnanceur garantit un partage équitable du temps entre ces flots.

oRis est stable et opérationnel. Il a été utilisé dans de nombreux projets et constitue le cœur de la plate-forme ARÉVI.

4.2.5.2 ARéVi

ARÉVI [Reignier et al. 98] est un atelier de réalité virtuelle dont le noyau est ORIS et donc, toutes les potentialités de ce langage sont disponibles lors de la conception d'applications sous ARÉVI. Il est étendu par du code C++ offrant des fonctionnalités propres à la réalité virtuelle. Cette plate-forme offre un rendu graphique fondé sur OPENGL OPTIMIZER. Les objets graphiques sont chargés directement à partir de fichiers au format VRML2 pour lesquels il est possible de définir des animations (comme présenté dans la section 4.2.3) et de gérer les niveaux de détails. Des éléments graphiques tels que des textures transparentes ou animées, des sources lumineuses et des *lens flares* (reflets du soleil sur une lentille) sont disponibles. ARÉVI introduit aussi des notions de cinématique (vitesses et accélérations linénaires et angulaires), ce qui enrichit les possibilités d'expression de comportements des agents dans un environnement tridimensionnel. Pour ce qui est du domaine sonore, ARÉVI propose une sonorisation tridimensionnelle et des fonctionnalités de synthèse et de reconnaissance vocale. Cette plate-forme gère des périphériques variés tels un gant de donnée, une manette de commande, un volant, des capteurs de localisation et un casque de stéréovision qui étendent les possibilités d'immersion des utilisateurs dans le système multi-agents.

4.2.5.3 Exemples de simulations utilisant ARéVi et oRis

Deux projets nous ont servis pour apréhender la plate-forme ARéVi/oRis et les concepts fondateurs de cette thèse. Le premier projet porte sur la réalisation de comportements (KorSo) et le second sur la modélisation et la visualisation d'environnements virtuels (Brest 2000).

KorSo (*Korrekte Software*) est une cellule de production qui sert de cas d'étude aux recherches dans le domaine de l'automatisme. Nous avons utilisé les techniques des systèmes multi-agents et de la réalité virtuelle pour modéliser cette cellule de production [Querrec et al. 97]. La cellule de production est composée de deux tapis, d'une table élévatrice rotative, d'un robot à deux bras, d'une presse et d'une grue. Tous ces éléments sont représentés par des agents composés d'une partie opérative qui effectue les actions et d'une partie contrôleur qui organise les différentes tâches à effectuer. La partie opérative est modélisée en 3D et les actions sont essentiellement des translations et des rotations. La partie contrôleur est représentée par un réseau de Petri ou un Grafcet [Le Parc et al. 99]. La figure 4.17 montre une visualisation 3D de la cellule. La synchronisation des différents éléments se fait par diffusion de messages. Lorsqu'un agent a terminé sa tâche, il diffuse le message dans l'univers. L'agent qui s'est synchronisé sur ce message commence alors sa tâche.

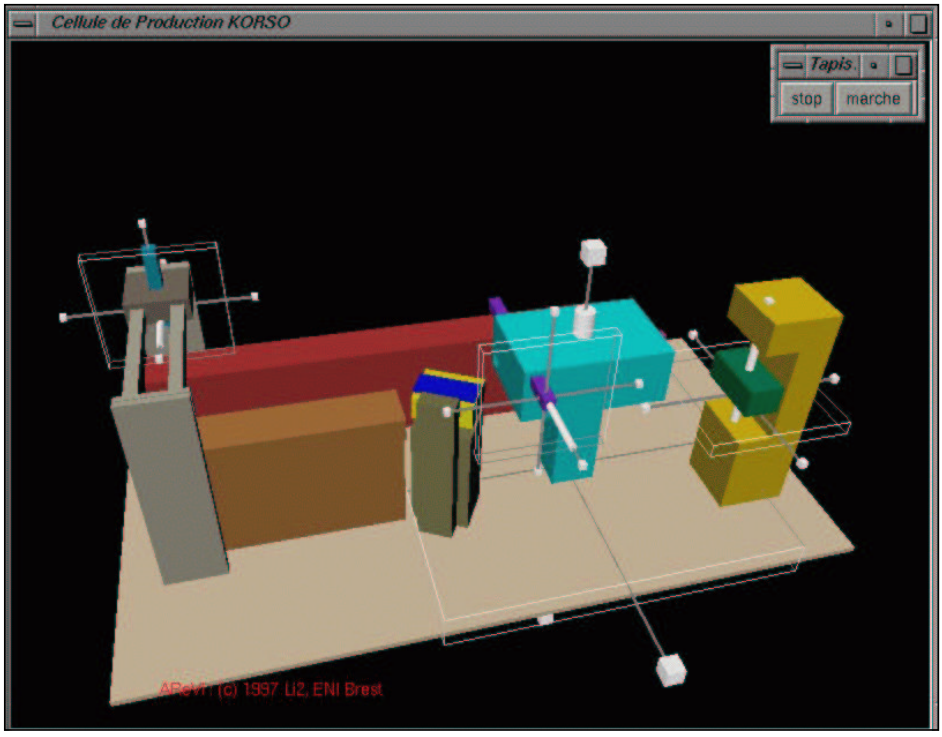


Figure 4.17 : Vue 3D de la cellule de production KorSo

Cette approche présente plusieurs avantages. Tout d'abord, les réseaux de Petri à développer sont plus simples qu'un réseau gérant la cellule entière, il y a donc moins de

risques d'erreurs. De plus, il est possible de reconfigurer l'agencement de la cellule sans arrêter la simulation. L'utilisation d'ARÉVI et ORIS permet également de modifier le comportement de chaque agent *en ligne*. Il est donc possible de changer la partie opérative ou la partie contrôleur lors de la simulation. Ces principes de modularité, incrémentalité et robustesse sont repris dans le modèle MASCARET.

Le projet BREST 2000[18] consiste en la réalisation de modèles tridimensionnels des bâtiments de l'ensemble du port de BREST à partir des données cadastrales, des relevés sur le terrain et éventuellement, pour des données plus précises, des données de la base IGN[19]. Sur place un ensemble de prises de vue photographiques et vidéo permet la saisie des textures (figure 4.18). A partir de ces données, des fichiers de représentation des entités graphiques aux formats INVENTOR et VRML97, texturées par plaquage, sont générés par programme. Chaque objet est représenté par plusieurs fichiers graphiques afin de gérer les niveaux de détails. La navigation dans la scène se fait à l'aide de métaphore comme l'utilisation de véhicule (voiture, hélicoptère, bateau...), la *téléportation* et la navigation à l'aide de périphériques (souris, *joystick*, *space-mouse*...)

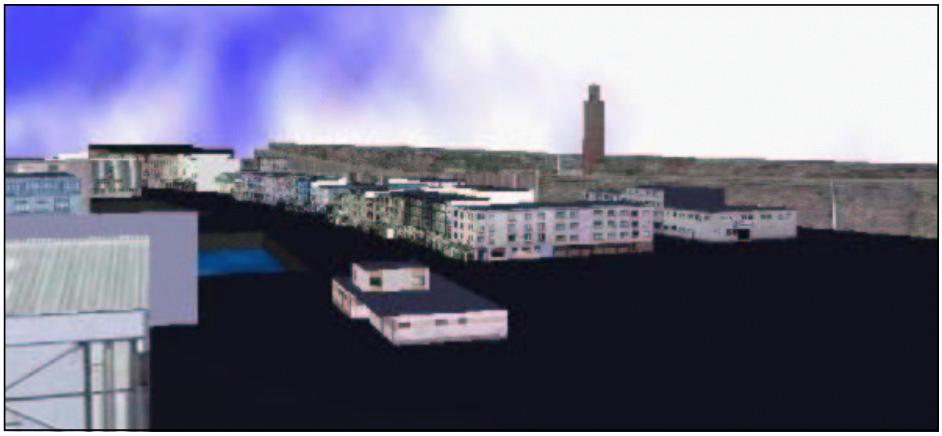


Figure 4.18 : Vue 3D du port de BREST sous ARÉVI

Pour améliorer le réalisme des scènes 3D, il convient d'apporter des éléments d'ambiance, ces éléments sont créés par des programmes spécifiques développés en OPENGL, et activés par des agents ORIS. Les ambiances peuvent être visuelles (création de ciels, alternances jour-nuit, phénomènes atmosphériques, éclairages, ombres, aspects de la mer) ou sonores (sons spatialisés liés aux déplacements de l'utilisateur, musiques et sons localisés, banques de données sonores, fonds sonores). Ces techniques de modélisation et de rendu sont utilisées dans MASCARET pour représenter les environnements.

[18] `http://www.brest2000.asso.fr`

[19] Institut Géographique National (`http://www.ign.fr`)

4.3 Utilisation pédagogique

L'utilisation pédagogique de SécuRéVi n'est pas encore complètement formalisée, mais il existe assurément une phase de conception des éléments pouvant intervenir dans les exercices, une phase de conception des scénarios, de simulation et de *debriefing*. Ces quatre phases sont illustrées sur la figure 4.19 et détaillées dans cette section.

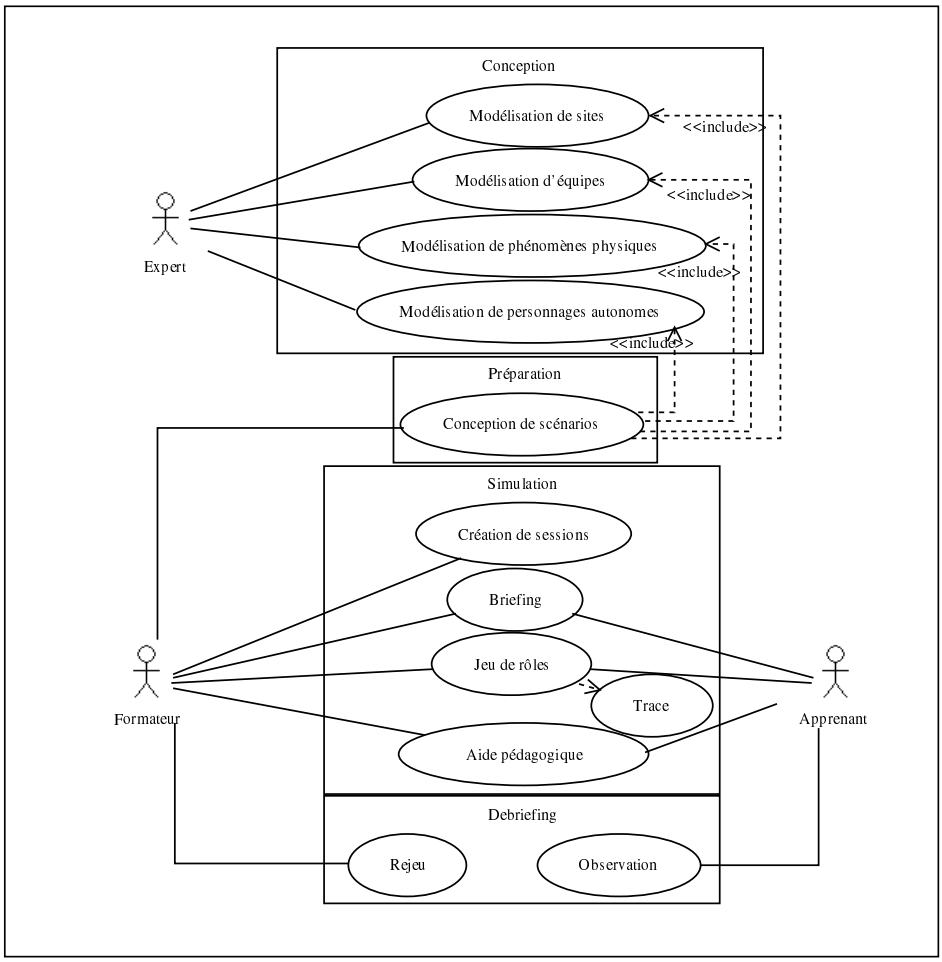


Figure 4.19 : Cas d'utilisation de SécuRéVi.

4.3.1 Cas d'utilisation

L'utilisation pédagogique de SécuRéVi fait intervenir trois rôles (expert, formateur et apprenant) et se déroule sur quatre phases. La première phase est la conception des éléments qui peuvent intervenir dans les exercices. Il s'agit de concevoir les phénomènes physiques et les équipes de sapeurs-pompiers ; c'est l'étape que nous avons présentée précédemment pour l'exercice de fuite de gaz. Cette phase fait intervenir un expert

(ou un groupe d'experts) ayant des compétences dans le modèle MASCARET, la programmation et le domaine spécifique à modéliser : chimistes, physiciens ou météorologues pour les phénomènes physiques et officiers sapeurs-pompiers pour les équipes d'intervention.

La seconde phase est réalisée par le formateur, en fonction des éléments fournis par les experts (essentiellement les structures organisationnelles et les types d'agent), il décrit des exercices ou des scénarios. Il s'agit essentiellement de l'intanciation des structures organisationnelles et l'affectation des agents aux rôles. Un scénario comprend également des situations par lesquelles le formateur veut que l'apprenant passe, ces situations sont représentées par des évènements qui peuvent être diffusés dans l'environnement ou adressés explicitement à un de ses agents pour qu'il provoque cette situation.

Les deux dernières phases sont la phase de simulation et la phase de *debriefing*, elles font intervenir le formateur et l'apprenant. Nous montrons dans la suite un scénario type où nous détaillons les rôles des apprenants et des formateurs lors de la simulation. Durant cette phase toutes les actions et tous les messages échangés sont datés et enregistrés pour servir durant le *debriefing*. Cette dernière phase n'a pas été formalisée, mais les fonctionnalités classiques attendues lors de cette phase sont le rejeu de la simulation. Cette fonctionalité n'est pas implémentée dans SécuRéVi et constitue à elle seule un cadre d'étude non trivial.

4.3.2 Simulation

Lors de la simulation, le rôle de l'apprenant est d'effectuer des actions dans l'environnement pour résoudre l'incident ; ces actions sont soit des actions physiques, soit des communications avec les agents peuplant l'environnement. Le rôle du formateur est de générer des dysfonctionnements, apporter des ressources pédagogiques à l'apprenant et modifier l'environnement pour le corriger ou prendre en compte le comportement inattendu de l'apprenant. Dans le cas de l'exercice de fuite de gaz dans un site industriel, le formateur génère l'incident et aide les agents à effectuer des actions (arrosage, ...) L'apprenant est le chef des opérations de secours et ordonne la réalisation de manœuvres aux équipes sous ses ordres.

Pour représenter ce scénario, nous avons choisi un support permettant de montrer au mieux l'aspect dynamique et interactif de la simulation en l'illustrant par une bande dessinée. Dans cette bande dessinée, la réflexion des agents est symbolisée par une bulle de la forme d'un rectangle aux bords pointillés et les communications qu'il émettent par une bulle de la forme d'un rectangle aux bords pleins. Cette bande dessinée nous permet de mettre en évidence la présentation du site, le rôle de l'apprenant et le rôle du formateur.

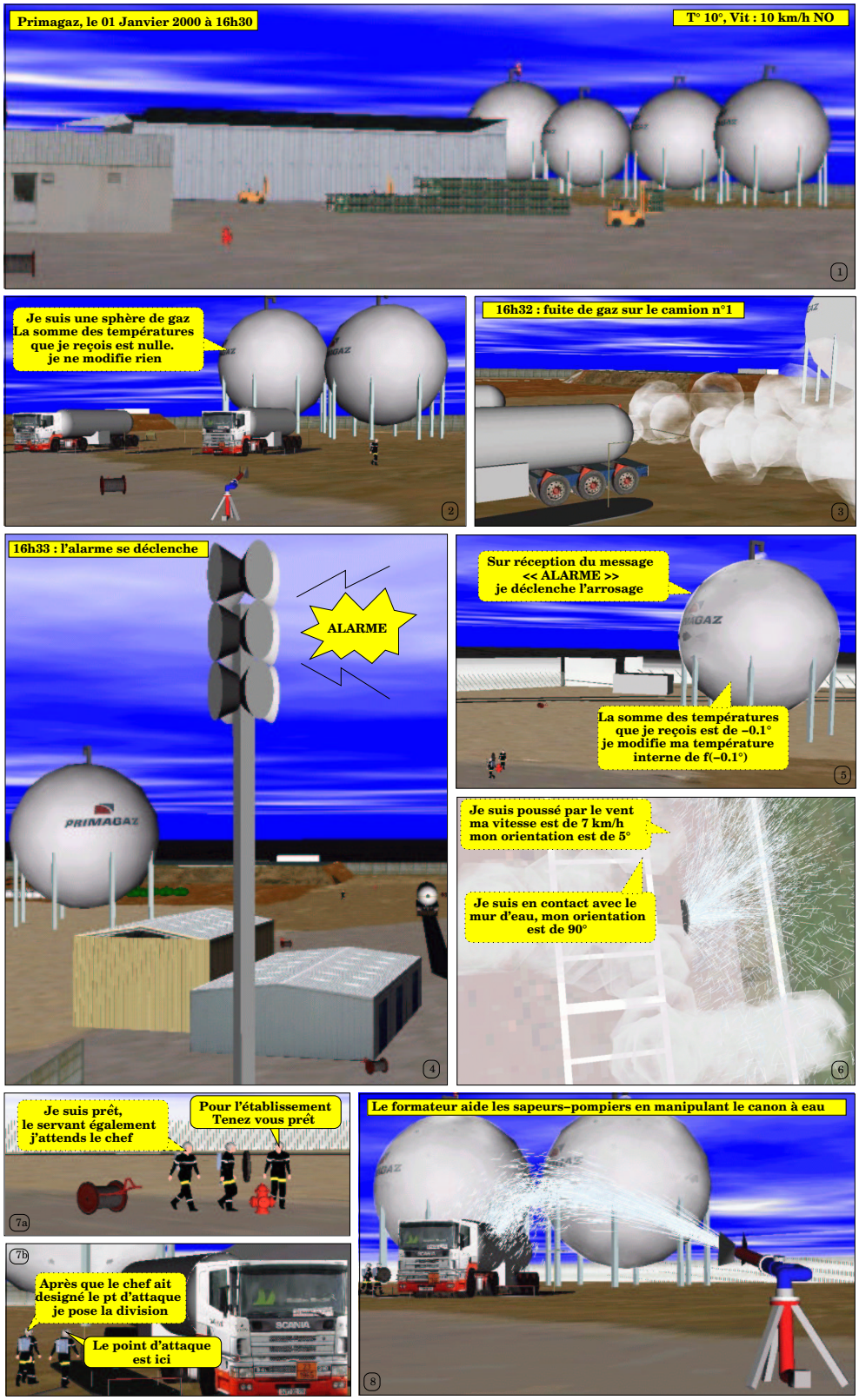


Figure 4.20 : Présentation du site dans SécuRéVi.

4.3.2.1 La présentation du site

La première partie de l'exercice permet au formateur de présenter la scène à l'apprenant. Ils naviguent dans le site, qui dans le cas de cet exercice est un site industriel de stockage de gaz, composé d'entrepôts, de réservoirs et de véhicules. Tous ces éléments sont des agents et disposent de comportements réactifs. Ainsi les citernes de gaz participent au réseau de transfert thermique et calculent leur température interne à partir de la somme des transferts thermiques qu'elles reçoivent. Dans cet exercice, il s'agit de la source de danger la plus importante : lorsque ces citernes sont surchauffées, elles explosent et provoquent un BLEVE[20] qui a des conséquences catastrophiques dans un périmètre important. Des agents de sécurité sont postés en permanence sur le site et forment une équipe de première intervention en cas d'incident. Il disposent également de comportements réactifs qui leur font subir les effets toxiques du gaz ainsi que les transferts thermiques.

Cette première partie permet également au formateur de poser le problème en modifiant l'environnement devant l'apprenant. Dans notre exemple, le formateur crée une fuite de gaz lors du chargement d'un camion. Cette fuite est représentée par un ensemble de particules de gaz qui sont des agents possédant des comportements réactifs et participant aux réseaux de collisions et de mobilité ; elles prennent ainsi la direction et la vitesse du vent. Le professeur déclenche ensuite l'alarme qui est diffusée dans l'environnement sous la forme d'un message auquel certains agents de l'environnement sont sensibles et réagissent. Ainsi, les sphères de gaz recevant ce message sont arrosées par un mécanisme situé au dessus d'elles ; cet arrosage a pour effet de réduire leur température en cas d'inflammation du nuage de gaz. Les murs d'eau situés autour du site sont également sensibles à ce message et se mettent en marche ; ils forment alors un obstacle pour les particules de gaz car elles participent au réseau des collisions et permettent de confiner le gaz dans le site.

Un dernier type d'agent est sensible au message d'alarme, il s'agit des membres de l'équipe de sécurité. Ils sont organisés à la manière des équipes de sapeurs-pompiers et chacun d'eux connaît son rôle : chef, sous-chef ou servant. Ils connaissent également les mêmes missions que les sapeurs-pompiers, et sur réception de l'alarme, ils commencent une manœuvre d'établissement de lance pour protéger le camion. Le formateur choisit ensuite de jouer le rôle d'un agent sur le site et de manipuler le canon à eau.

[20] Boiling Liquid Expanding Vapor Explosion

Figure 4.21 : Le rôle de l'apprenant.

4.3.2.2 Le rôle de l'apprenant

La seconde phase de l'exercice permet à l'apprenant d'intervenir et de tenter de résoudre l'incident. Dans notre exercice, l'apprenant joue le rôle du chef des opérations représenté par le chef de groupe qui a sous ses ordres plusieurs équipes FPT (nous prenons ici l'exemple d'une seule équipe). La première action qu'effectue l'apprenant est une action de reconnaissance. Il navigue dans l'environnement via son avatar, pour acquérir des informations sur l'incident et les ressources déjà engagées. L'avatar de l'apprenant, comme les agents de sécurité et les sapeurs-pompiers intervenant dans les équipes disposent de comportements réactifs. Ainsi, lors de sa navigation dans le site, si l'apprenant passe dans le nuage de gaz, sa vue se trouble et son champ de perception ainsi que sa vitesse de déplacement diminuent. L'apprenant à accès aux documents dont il dispose dans un exercice réel, ainsi, lorsqu'il retourne dans le PCM[21], il peut lire les plans d'interventions prévus pour le site considéré. Le PER[22] l'informe sur le contenu probable des différents réservoirs, les voies d'accès et les manœuvres à réaliser selon l'incident.

La décision de l'apprenant se traduit alors par l'envoi d'un ordre de réalisation de manœuvre par une équipe de sapeurs-pompiers. Ici, l'apprenant ordonne au chef de l'équipe FPT n°1 d'établir une lance à incendie pour protéger le wagon citerne. Dans la réalité, l'envoi de cet ordre se fait par *talkie-walkie*; nous avons symbolisé cette communication par des envois de messages asynchrones. L'apprenant dispose alors d'une interface (boîte aux lettres) proposant la liste des intervenants susceptibles de pouvoir recevoir des messages et l'ensemble des messages que l'apprenant peut leur envoyer. Dans le cas des sapeurs-pompiers, les messages sont codifiés ce qui permet d'envisager l'utilisation des techniques de reconnaissance vocale pour implémenter ce type de communication.

A la réception de ce message, l'équipe FPT commence la réalisation de la mission. La première action de cette mission est réalisée par le chef qui ordonne à ses subordonnés de se tenir prêts. Pour être prêt, le sous-chef doit avoir un dévidoir ; il s'agit de la pré-condition de son action `SeTenirPret`. Cette pré-condition n'est pas vérifiée, il se construit alors un plan implicite qui, par inférence sur les actions, le fait partir à la recherche du dévidoir. Le chef effectue le même raisonnement, mais pour la lance ; malheureusement le concepteur du scénario a fait une erreur et n'a pas prévu de lance dans l'exercice. L'action de recherche du chef est donc en échec (dépassement du temps alloué). Il en réfère alors à l'apprenant, son supérieur hiérarchique (il s'agit de son comportement organisationnel) et abandonne la mission. Comme il s'agit d'une erreur de conception du scénario, l'apprenant redonne le même ordre, et le chef de l'équipe FPT se remet à la recherche d'une lance.

[21] Poste de Commandement Mobile

[22] Plan d'Etablissement Répertorié

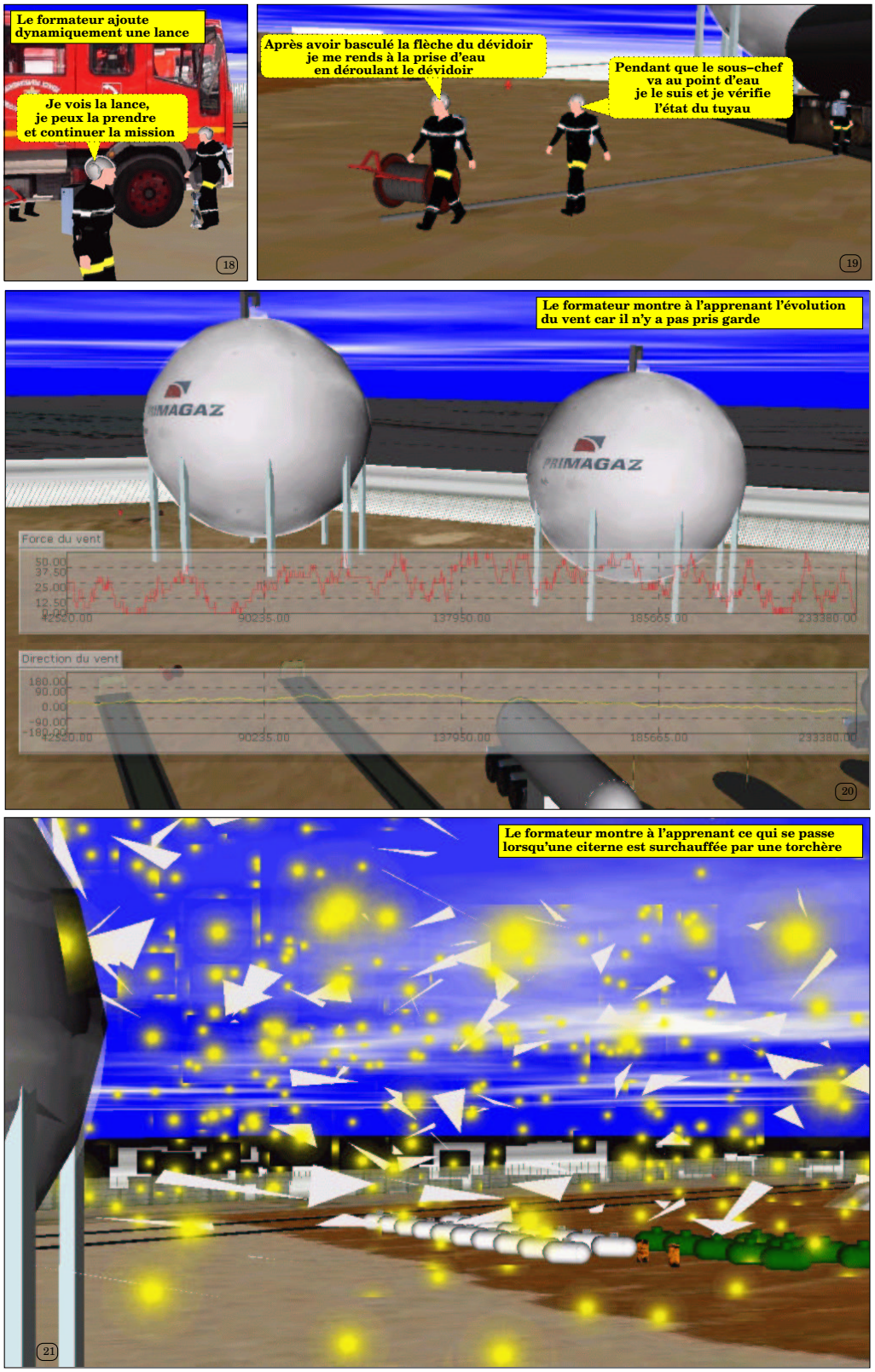


Figure 4.22 : Le rôle du formateur dans SécuRéVi.

4.3.2.3 Le rôle du formateur

Pendant l'exercice, le formateur effectue des interventions pédagogiques pour plusieurs raisons. Il intervient tout d'abord pour modifier le scénario pour prendre en compte un comportement non prévu de l'apprenant ou, dans notre cas pour corriger une erreur. Ainsi, le professeur ajoute, en cours de simulation, une lance à incendie dans l'environnement. Notons que cette fonctionnalité n'est possible que parce que SécuRéVi est implémenté à l'aide d'un langage interprété (oRis) qui permet l'insertion de code en cours d'exécution. Lors de son prochain passage proche du point où la lance a été insérée, le chef la verra, pourra la prendre et continuer la mission. Cet exemple illustre l'aspect dynamique du modèle que nous proposons, car si l'agent s'est rendu compte de manière autonome de l'absence de la lance, il se rend compte également de manière autonome de la modification de l'environnement ; en effet, lorsque le formateur ajoute une lance, il ne fait que créer une nouvelle instance de lance et ne modifie en rien le comportement de l'agent.

Le formateur peut également intervenir pour apporter des informations non perceptibles à l'apprenant. Par exemple, dans notre exercice, la force et la direction du vent sont des informations importantes pour le choix des actions à réaliser ; ces informations ne sont pas perceptibles par l'apprenant (sauf utilisation d'intruments de mesure comme la manche à air), le formateur peut donc lui proposer une visualisation de ces informations, sous forme de courbes par exemple. Le formateur intervient également en cours d'exercice pour montrer les conséquences éventuelles d'une action (ou d'un manque d'action) de l'apprenant. Par exemple, dans notre exercice, le nuage de gaz peut se diriger vers des citernes de gaz et, dans certaines conditions, s'enflammer et se transformer en torchère. Cela a pour effet de chauffer les citernes et de les faire exploser ; le formateur peut montrer cette conséquence en chauffant de manière artificielle une citerne.

4.4 Conclusion

Dans ce chapitre, nous avons tout d'abord justifié la réalisation d'un environnement virtuel de formation pour la sécurité civile en étudiant les outils déjà disponibles. Cette étude a montré les lacunes de l'existant en terme d'immersion et de fonctionnalités pédagogiques. En effet, beaucoup de ces outils sont issus de la simulation de modèles physiques qui permet d'expliquer des phénomènes mais ne permet pas aux utilisateurs de participer à la simulation et d'en modifier le cours. D'autres applications portent sur le caractère collaboratif de la formation à la GOC ; ils simulent l'effet des communications et des collaborations entre les apprenants, mais peu d'entre eux simulent en même temps les environnements physique et social (autres intervenants autonomes), ce qui ne permet pas aux apprenants et aux formateurs de partager la même représentation mentale de la situation. La dernière lacune importante présentée

par ces environnements, même parmi les plus aboutis tel que VECTOR COMMAND, est l'impossibilité de fournir aux formateurs des outils pour concevoir leurs propres exercices et cela non pas du fait des difficultés de conception d'interfaces, mais à cause du modèle adopté pour la modélisation et la simulation de l'environnement.

Nous avons donc proposé d'appliquer le modèle MASCARET à la réalisation de l'environnement virtuel de formation SÉCURÉVI. Nous avons montré dans ce chapitre comment dériver les modèles d'environnement physique (`InteractionNet`) et d'environnement social (`Team`) de MASCARET pour concevoir ceux de SÉCURÉVI. Ainsi, prenant l'exemple d'un exercice type, nous avons montré dans un premier temps, la réalisation de réseaux d'interactions de collision et de mobilité ainsi que les agents particules de gaz y participant. Dans un second temps, nous avons dérivé le modèle MASCARET pour réaliser les équipes FPT (ainsi que les rôles et les missions qu'elles proposent) et les agents humanoïdes jouant ces rôles. Dans l'utilisation pédagogique de SÉCURÉVI, nous avons considéré que cette phase est réalisée par un expert qui fournit alors aux formateurs les éléments de base leur permettant de réaliser les scénarios des exercices. Enfin, nous avons donné des exemples, dans le cadre d'un exercice type, d'interactions possibles des apprenants et des formateurs avec l'environnement virtuel.

Chapitre 5

Conclusion

Nos travaux portent sur la réalisation d'outils logiciels pour la conception d'environnements virtuels de formation. L'objectif est de placer des apprenants en situation opérationnelle dans leurs environnements physique et social simulés. Notre thèse est que l'environnement virtuel de formation est un système multi-agents hétérogène car il simule aussi bien des agents réactifs que rationnels, et ouvert car il fait participer les utilisateurs. Notre problématique est la simulation d'environnements réalistes pour la formation au travail collaboratif et adaptatif. Nous proposons donc le modèle MASCARET (*MultiAgent System for Collaborative and Adaptive Realistic Environment for Training*) qui est un système multi-agents pour les environnements réalistes, collaboratifs et adaptatifs pour l'entraînement. Ce modèle sert de fondement à l'application SécuRévi, un environnement virtuel de formation à la gestion opérationnelle et au commandement destiné aux officiers sapeurs-pompiers. Le tableau de la figure 5.1 résume l'apport de MASCARET à la conception d'environnements virtuels de formation, c'est-à-dire à la simulation de l'environnement physique, social et à l'immersion de l'utilisateur. Ce résumé nous permet de dresser un bilan de nos travaux et de formuler des perspectives.

MASCARET	*Collaboratif*	*Adaptatif*	*Réaliste*
Environnement physique	Réseaux d'interactions	Entités organisationnelles dynamiques	Rendu de phénomènes physiques
Environnement social	Équipes, missions, procédures	Calcul de plans implicites, règles organisationnelles	Comportement *"métier"*
Immersion utilisateur	Collaboration Humain-Avatar	Pédagogie adaptée (délégation de tâches, inhibition de phénomènes...)	Interactions *"naturelles"*

Figure 5.1 : Apports de MASCARET aux environnements virtuels de formation.

5.1 Bilan

Par rapport à VOWELS, proposé par [Demazeau 95] pour modéliser les systèmes multi-agents, MASCARET apporte des modèles d'agents (*facette A*) réactifs et rationnels qui forment l'environnement des utilisateurs (*facette E*). Nous réifions les interactions (*facette I*) entre les agents et nous les structurons à l'aide du concept d'organisation (*facette O*). Enfin, nous proposons d'intégrer l'utilisateur dans le système multi-agents *via* son avatar (*facette U* de VOWELS enrichi par [Tisseau 01]).

Dans MASCARET, nous considérons deux types d'environnement (physique et social) donc deux types d'interaction et d'organisation. Certains agents appartiennent aux deux organisations ; nous proposons donc un modèle générique permettant, par dérivation, de représenter ces deux types d'organisation. Ce modèle générique d'organisation s'articule autour de l'association ternaire reliant les concepts d'organisation, de rôle et d'agent. L'organisation est le facteur structurant qui permet à chaque agent de savoir avec qui il est susceptible d'interagir. Les rôles décrivent les responsabilités des agents qui les jouent. Ces responsabilités sont représentées par les éléments de comportements réalisés par les agents. Le comportement de l'agent est de plus organisationnel dans le sens où il sait prendre en compte les autres membres de l'organisation et mettre à jour ses connaissances organisationnelles. Toutes les classes de ce modèle sont abstraites et sont dérivées pour représenter l'environnement physique et l'environnement social.

L'environnement physique doit être réaliste, interactif et réagir en *"temps réel"* aux actions de l'apprenant car c'est le support qui lui permet de *construire* ses connaissances. Ces contraintes justifient l'utilisation des systèmes multi-agents pour simuler cet environnement physique et nous le représentons sous la forme d'un réseau d'interactions. Pour qu'il y ait interaction entre les agents, il faut d'une part que les agents perçoivent la situation d'interaction et d'autre part qu'ils la prennent en compte dans leur comportement. C'est le comportement organisationnel de l'agent qui lui permet de détecter la situation d'interaction et donc de mettre à jour dynamiquement l'entité organisationnelle ; pour optimiser cette mise à jour, nous définissons un rôle de recruteur dans l'organisation. Nous considérons que les interactions ont une direction privilégiée, ce qui est spécifié par les rôles de source ou de cible. Les agents jouent les comportements réactifs définis dans les rôles spécifiques à chaque réseau. Ce comportement leur permet d'évaluer les influences qu'ils subissent, de modifier leur état interne en fonction de ces influences et de calculer l'influence qu'ils ont sur d'autres agents.

Certains de ces agents agissent de manière plus raisonnée dans l'environnement : ils intègrent un module de décision et choisissent d'effectuer des actions pour collaborer avec d'autres agents. Ces agents forment l'environnement social structuré par le concept d'équipe, un type particulier d'organisation. Les agents y participant ont un comportement collaboratif qui leur permet d'atteindre des buts communs représentés

par des missions. Nous nous intéressons au cas où ces missions décrivent l'enchaînement des actions des membres de l'équipe et proposons un modèle de représentation de ces procédures par des ensembles de contraintes temporelles. Le comportement de l'agent est de collaborer à la réalisation de cette procédure. L'architecture d'agent que nous proposons est destinée aux environnements virtuels de formation. Ainsi, les buts des personnages sont explicites ; il s'agit de l'énoncé de l'exercice ou de la volonté de l'apprenant de réaliser une mission. Les agents "personnages" n'ont donc pas à déterminer leur but, cependant, la réalisation d'un but final peut nécessiter d'identifier des sous-buts (non explicités par les utilisateurs) et de calculer des plans implicites pour les atteindre (ce qui représente la pro-activité des agents). Les plans explicites sont donnés par le concepteur et font partie de la connaissance des agents. Par contre, les plans implicites sont générés par l'agent lui même (par chaînage arrière sur les préconditions et post-conditions des actions).

L'utilisateur (apprenant ou formateur) agit dans l'environnement (physique ou social) *via* son avatar dont l'architecture est la même que celle d'un agent rationnel ; la seule différence entre ces deux types d'agent tient dans le fait que c'est l'utilisateur qui choisit les actions et l'avatar qui les réalise. Le module de raisonnement de l'avatar reste actif pour mémoriser les actions effectuées en vue de lui apporter une aide pédagogique telle que la prise en charge de certaines tâches. L'avatar participe à l'environnement physique et en subit les effets mais, pour adapter le scénario à un étudiant particulier, il est possible d'inhiber certains comportements réactifs.

Le modèle MASCARET est appliqué à SécuRéVi destiné à la formation des officiers sapeurs-pompiers. Cet environnement virtuel de formation permet la simulation réaliste, d'une part de l'environnement physique constitué de sites industriels et de phénomènes physiques tels que la propagation d'un nuage de gaz et l'effet d'un jet d'eau, et d'autre part, de l'environnement social représenté par des équipes de sapeurs-pompiers. La conception de SécuRéVi tient dans la description de structures organisationnelles et d'agents fondés sur ceux proposés dans MASCARET. Ainsi pour concevoir les phénomènes physiques décrits dans SécuRéVi, le concepteur se fonde sur les réseaux d'interactions et les comportements réactifs. Pour concevoir les équipes de sapeurs-pompiers ainsi que pour décrire les missions et les actions réalisées par leur membres, il faut dériver des notions d'équipe, de mission et d'action de MASCARET. L'utilisation pédagogique de SécuRéVi s'articule autour de trois rôles. L'expert est le concepteur des structures organisationnelles ainsi que des agents pouvant y participer. Le formateur écrit les scénarios des exercices à partir de ces éléments en instanciant les structures organisationnelles. Il participe également à la simulation pour générer des dysfonctionnements ou au contraire apporter une aide pédagogique à l'apprenant qui joue un rôle dans les équipes créées. L'application SécuRéVi totalise environ une dizaine d'agents personnages et un millier d'agents "réactifs". Il semble difficile d'utiliser MASCARET pour concevoir des applications nécessitant un très grand nombre d'agents "personnages", mais notons que MASCARET est destiné à la conception d'environnements virtuels de formation dans lesquels il semble impossible que l'apprenant puisse faire face à plusieurs centaines de personnages ayant tous un

comportement rationnel différent. Si toutefois ce grand nombre de personnages doit être simulé, ils seraient alors regroupés dans des entités plus importantes (équipes, bataillons ...) qui controleraient leur comportement.

5.2 Perspectives

MASCARET permet la réalisation d'environnements virtuels réalistes, coopératifs et adaptatifs pour la formation. Une des premières perspectives que nous envisageons est d'appliquer ce modèle dans d'autres contextes pédagogiques où l'environnement physique n'est pas forcément réel mais imaginaire et où l'environnement social laisse plus d'autonomie aux agents. Dans ce cas, le travail consiste à créer de nouveaux types d'organisation dérivant du modèle générique que nous proposons et à améliorer les modèles de comportements d'agents. Le premier modèle que nous souhaitons améliorer est celui des comportements réactifs. En effet, les comportements que nous avons développés simulent des phénomènes physiques et s'inspirent des modèles numériques classiques (qui calculent de manière globale l'évolution du phénomène). Nous souhaitons améliorer ces comportements, avec l'aide des experts du domaine, pour transposer au mieux leurs modèles numériques vers notre approche fondée sur les systèmes multi-agents. De plus, le caractère adaptatif du modèle pourrait être exploité pour l'optimisation des performances en discrétisant, de manière dynamique et auto-adaptative, l'environnement physique.

Les comportements des agents autonomes représentant les humanoïdes peuvent également être améliorés. Nous souhaitons d'une part leur donner plus de capacités d'autonomie. En effet, le cadre d'étude que nous avons choisi est le travail procédural qui, dans le modèle que nous proposons, guide le comportement de l'agent, mais ce n'est pas le cas de tous les domaines d'applications possibles de MASCARET. Pour réaliser une planification dynamique des actions des agents, différentes pistes sont envisageables, telles que les architectures de type BDI. De plus, le mécanisme qui permet à un agent de percevoir les actions des autres membres de son équipe est fondé, dans notre modèle sur l'envoi de messages. Ce mécanisme peut être amélioré en fournissant aux agents la capacité de reconnaître les actions, voire les intentions des utilisateurs comme proposé par [Favier et al. 01], ce qui permettrait à ces derniers d'agir plus naturellement. Certains domaines d'application imposent également une plus grande autonomie dans la gestion de l'organisation. Il nous faut alors réifier la notion de règles organisationnelles, comme c'est le cas dans [Hannoun et al. 99], pour que les agents puissent par exemple, raisonner sur l'affectation des rôles. De plus, le protocole qui gère le comportement organisationnel, dans MASCARET, pose des problèmes telle que la gestion des désengagements. Il est donc intéressant d'étudier d'autres mécanismes en s'inspirant par exemple de méthodes de négociation telle que celle proposée par [Aknine et al. 99]. D'autre part, dans le cadre d'un outil de formation s'adressant au travail sur des sites à risques, la composante émotionnelle des agents semble interessante à simuler. Ainsi, il serait possible de simuler des agents *"téméraires"*

ou *"fatigués"* pour représenter de manière plus réaliste l'environnement social. Les travaux de [Parenthoen et al. 01] qui reposent sur l'utilisation des cartes cognitives floues vont dans ce sens.

Le modèle d'avatar que nous proposons suffit pour son utilisation dans le cadre d'un environnement virtuel de formation destiné à la prise de décisions et non aux gestes techniques, mais l'amélioration de ce modèle permettrait une plus grande immersion de l'apprenant dans son environnement. Cela passe par l'utilisation de périphériques (de navigation ou de préhension) qui permettent à l'utilisateur d'effectuer ses actions plus naturellement. Se pose alors le problème de la reconnaissance d'actions. En effet, dans MASCARET, le mécanisme de collaboration entre l'humain et son avatar est représenté par des envois de messages (l'apprenant verbalise ses actions). Si ce n'est plus le cas, il faut que l'avatar reconnaisse les actions effectuées par l'humain. Les travaux de [Lourdeaux 01], fondés sur les primitives virtuelles comportementales peuvent apporter une réponse à ce problème. Mais cette méthode impose de connaître les types d'action qui sont réalisés par les utilisateurs, ce qui est faisable dans un cadre applicatif, mais non utilisable pour un modèle générique. Il semble donc intéressant d'étudier des typologies d'action, ce qui permettrait, de plus, à l'utilisateur de déléguer l'exécution de certains types d'action à son avatar. Ce mécanisme peut s'appuyer sur la notion de rôle et de collaboration pédagogique entre l'apprenant et son avatar.

Dans MASCARET, les fonctionnalités pédagogiques attendues classiquement dans un environnement virtuel de formation, n'ont pas été formalisées. L'étude de différents environnements de formation montre que MASCARET est une solution qui permet d'implémenter les fonctions présentes dans les environnements existants et qui offre des mécanismes permettant de les améliorer. En effet, la réalité virtuelle et les systèmes multi-agents semblent intéressants pour la réalisation de scénarios et de mises en scène qui, d'après [Crampes et al. 99] améliorent l'implication de l'apprenant dans la simulation. De plus, l'approche que nous avons choisie pour simuler l'environnement virtuel permet de répartir la réalisation de ces fonctionnalités dans l'environnement et d'utiliser au mieux l'avatar pour implémenter les aides pédagogiques telles que l'assistance ou la démonstration. En effet, celui-ci dispose des connaissances suffisantes pour créer le modèle de l'étudiant et le comparer à celui de l'expert. Les retours d'expérience de l'application SécuRéVi nous permettront de formaliser ces fonctionnalités pédagogiques au plus près des attentes des utilisateurs. Ce travail est d'ores et déjà planifié dans le cadre d'une collaboration pluriannuelle avec le SDIS du Finistère, l'Institut National des Etudes sur la Sécurité Civile (INESC) et le Laboratoire d'Informatique Industrielle de l'Ecole Nationale d'Ingénieurs de Brest.

Glossaire

ABIS *Agent Based Information System* [Durand et al. 99]

ABROSE *Agent Based Brokerage Service in Electronic Commerce* [Gleizes et al. 00]

ADELE *Agent for Distance Education - Light Edition* [Shaw et al. 99]

ADIS *An Animated Data Structure Intelligent Tutouring System* [Warendorf et al. 97]

ADMS *Advanced Disaster Management Simulator* (`www.etcusa.com/admsmain.htm`)

AGR Agent/Groupe/Rôle [Gutknecht et al. 98].

ALI Agent Logiciel Intelligent [Jaber et al. 98]

ANTIC Application des Nouvelles Technologies pour l'Induction des Compétences [Savall et al. 98]

ARéVi Atelier de Réalité Virtuelle [Reignier et al. 98]

ARI Appareil Respiratoire Isolant

ARVESTER Abattage Reconstitué Virtuellement En Synthèse Temps Réel (`www.ai.cluny.ensam.fr`)

BDI *Believe Desire Intention* [Rao et al. 91]

BLEVE *Boiling Liquid Expanding Vapor Explosion*

CAMEO *Computer-Aided Management of Emergency Operations* (`response.restoration.noaa.gov/cameo/cameo.html`)

CANUTEC *Canadian Transport Emergency Centre* (`www.tc.gc.ca/canutec/fr/menu.html`)

CAO Conception Assistée par Ordinateur

CEDRE Centre de documentation, de Recherche et d'Expérimentations sur les pollutions accidentelles (`www.ifremer.fr/cedre`)

CFAST *Consolidated Fire and Smoke Transport Model* (`fast.nist.gov`)

CMIC Cellule Mobile d'Intervention Chimique

COS Chef des Opérations de Secours

CODIS Centre Opérationnel Départemental d'Incendie et de Secours

CMOS *Cockpit Maintenance Operations Simulator* [Richard 99]

EAO Enseignement Assisté par Ordinateur

EIAH Environnement Interactif d'Apprentissage Humain

EIAO Environnement Interactif d'Apprentissage avec Ordinateur

EIAO Enseignement Intelligemment Assisté par Ordinateur

EVE Environnement Virtuel pour les Enfants [Gerval et al. 02]

FIPA Foundation for Intelligent Physical Agents (`www.fipa.org`)

FPT Fourgon Pompe Tonne

GOC Gestion Opérationnelle et Commandement

HAL *Help Agent for Learning* [Lourdeaux 01]

H-ANIM *Humanoid Animation* (`www.hanim.org`)

HPTS *Hierarchical and Parallel Transition Set* [Donikian 01]

I^2 Immersion - Interaction [Fuchs 96]

IA Intelligence Artificielle

IAD Intelligence Artificielle Distribuée

IGN Institut Géographique National (`www.ign.fr`)

IHM Interface Homme/Machine

INESC Institut National des Etudes sur la Sécurité Civile (`www.inesc.fr`)

INRS Institut National de Recherche et de Sécurité (`www.inrs.fr`)

InViWo *Intuitive Virtual World* [Richard 01]

ITS Intelligent Tutoring System

KorSo *Korrekte Software*

KQML *Knowledge Query Manipulation Language* [Labrou 96]

LI2 Laboratoire d'Informatique Industrielle (`www.enib.fr/LI2`)

LIG Laboratoire d'Informatique Graphique (`www.vrlab.epfl.ch`)

LIP6 Laboratoire d'Informatique de Paris VI (`www.lip6.fr`)

MadKit *A Multi-Agent Development Kit* [Gutknecht 00]

MAGIQUE Multi-Agent Hierarchique [Bensaid et al. 97]

MASCARET Multi-Agent System for Collaborative and Adaptive Realistic Environment for Training

Mentoniezh Mot breton signifiant *"géometrie"* [Py 96]

MOISE Model of Organization for multI-agent SystEms [Hannoun et al. 99].

MRT Méthode des Raisonnements Tactiques

PCM Poste de Commandement Mobile

PER Plan d'Etablissement Répertorié

PCS Poste de Commandement de Site

PCV Primitives Comportementales Virtuelles [Fuchs 96]

PPI Plan Principal d'Intervention

Prolog Programmation en Logique

PRS *Procedural Reasoning System* [Georgeff et al. 87]

QCM Question à Choix Multiples

RV Réalité Virtuelle

SAD Système d'Aide à la Décision

SDIS Service Départemental d'Incendie et de Secours

SécuRéVi Sécurité et Réalité Virtuelle

SGBD Système de Gestion de Bases de Données

SAM Simulateur d'Apprentissage de la Méthode [Dosne 00]

SMA Système Multi-Agents

SOAR *State, Operator And Result* [Newell 90]

STEVE *Soar Training Expert for Virtual Environments* [Rickel et al. 99]

UML *Unified Modeling Language* [Rumbaugh et al. 99]

VET Virtual Environment for Training

VRML *Virtual Reality Modeling Language* (`www.vrml.org`)

VSAB Véhicule de Secours aux Axphyxiés et aux Blessés

VUEMS *Virtual Urban Environment Modelling System* [Donikian 97]

Références bibliographiques

[Aknine et al. 99] Aknine, S. et Pinson, S. (1999). Un nouveau protocole de négocation flexible pour la coopération multi-agents. In *JFIADSMA'99*, pages 165–177.

[Allen 83] Allen, J. (1983). Maintaining knowledge about temporal intervals. *Journal of the Association for Computing Machinery*, (26):832 – 843.

[Aïmeur et al. 00] Aïmeur, E., Frasson, C., et Dufort, H. (2000). Cooperative learning strategies for intelligent tutoring systems. *Applied Artificial Intelligence*, 14:465 – 489.

[Ballet et al. 97] Ballet, P., Rodin, V., et Tisseau, J. (1997). A multiagent system for detecting concentric strias. In *SPIE's Optical Sciences, Engineering and Instrumentation'97*, pages 659–666.

[Beaupied et al. 01] Beaupied, H., Multon, F., et Delamarche, P. (2001). Mechanics of the spontaneous transition between walking and running. *European Journal of physiology*, 442(5).

[Bellard 01] Bellard, S. (2001). *Méthodes de négociation dans un système multi-agents*. Rapport de D.E.A. M.I.A.S.H., Ecole Nationale Supérieur des Télécommunications de Bretagne.

[Bensaid et al. 97] Bensaid, N. et Mathieu, P. (1997). A hybrid and hierarchical multi-agent architecture model. In *Proceeding of the second international conference and exhibition on the practical application of intelligent agents and multi-agent technology, PAAM 97*, pages 145 – 155.

[Bindiganavale et al. 00] Bindiganavale, R., Schuler, W., Allbeck, J. M., Badler, N. I., Joshi, A., et Palmer, M. (2000). Dynamically altering agent behaviors using natural language instructions. In *Autonomous Agents*, pages 293 – 300.

[Booch et al. 99] Booch, G., Rumbaugh, J., et Jacobson, I. (1999). *The Unified Modeling Language User Guide*. Addison-Wesley, New-York.

[Boukachour et al. 98] Boukachour, H., Cardon, A., et Durand, S. (1998). *Conception d'un système multi-agents adaptatif: application à la gestion de crise.* Laboratoire d'Informatique de Paris 6 ; Objets et Agents pour Systèmes d'Information et de Simulation.

[Breton et al. 99] Breton, L., Zucker, J.-D., et Clément, E. (1999). Une approche multi-agents pour la résolution d'équations en physique des milieux granulaires. In *JFIADSMA'99*, pages 281–293.

[Brooks 86] Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14 – 23.

[Brown et al. 98] Brown, S., Santos, E., et Banks, S. (1998). Utility theory-based user models for intelligent interface agents. In *Canadian Conference on AI*, pages 378 – 392.

[Bruner 90] Bruner, J. (1990). *Acts of meaning.* Harvard University Press, Cambridge.

[Bryson 00] Bryson, J. (2000). Cross-paradigm analysis of autonomous agent architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 12:165–189.

[Bukowski et al. 97] Bukowski, R. et Sequin, C. (1997). Interactive simulation of fire in virtual building environments. In *Computer Graphics Proceedings*, pages 35–44.

[Burdea et al. 93] Burdea, G. et Coiffet, P. (1993). *La réalité virtuelle.* Hermès, Paris.

[Burg 00] Burg, B. (2000). Toward the deployment of an open agent world. In *JFIADSMA'00*, pages 13 – 29.

[Capuano et al. 00] Capuano, N., Marsella, M., et Salerno, S. (2000). Abits: An agent based intelligent tutoring system for distance learning. In *International Workshop in Adaptative and Intelligent Web Based Educational System*, pages 38 – 59.

[Carbonell 70] Carbonell, J. (1970). AI in CAI: an artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine systems*, 11(4):190 – 202.

[Cavazza et al. 01] Cavazza, M., Charles, F., et Mead, S. (2001). Characters in search of an author: AI-based virtual storytelling. *ICVS 2001 International Conference on Virtual Storytelling, LNCS 2197*, pages 145 – 154.

[Chaignaud et al. 01] Chaignaud, N. et El Fallah-Seghrouchni, A. (2001). Apport de la modélisation cognitive aux langages de communication entre agents. In *JFIADSMA'01*, pages 239 – 251.

[Chevaillier et al. 00] Chevaillier, P., Harrouet, F., Reignier, P., et Tisseau, J. (2000). Virtual reality and multi-agent systems for manufacturing system interactive prototyping. *International Journal of Design and Innovation Research*, 2(1):90–101.

[Chicoisne 00] Chicoisne, G. (2000). Interaction conversationnelle entre internautes et agents artificiels partageant un monde virtuel. In *JFIADSMA'00*, pages 297 – 300.

[Conati et al. 97] Conati, C., Gertner, A. S., VanLehn, K., et Druzdzel, M. J. (1997). On-line student modeling for coached problem solving using Bayesian Networks. In *Proceedings of UM-97, Sixth International Conference on User Modeling*, pages 231–242.

[Crampes et al. 99] Crampes, M. et Saussac, G. (1999). Facteurs qualité et composantes de scénario pour la conception de simulateurs pédagogiques à vocation comportementale. *Sciences et Techniques Educatives*, 6(1):11 – 36.

[Daniel 01] Daniel, T. (2001). *Formation à la GOC : Gestion Opérationnelle et Commandement*. Rapport de stage de mastère RVD.

[De Loor et al. 00] De Loor, P. et Chevaillier, P. (2000). Generation of agent interactions from temporal logic specifications. In *16th IMACS World Congress*.

[DeAngelis et al. 79] DeAngelis, D. L., Cox, D. K., et Coutant, C. C. (1979). Cannibalism and size dispersal in young-of-the-year largemouth bass: experiment and model. *Ecological Modelling*, 8:133–148.

[Demazeau 95] Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *Proceeding of the European Conference on Cognitive Science*, pages 117 – 132.

[Deschenes et al. 96] Deschenes, A., Bilodeau, H., Bourdages, L., Dionne, M., Gagne, P., Lebel, C., et Rada-Donath, A. (1996). Constructivisme et formation à distance. *Distances*, 1(1):9 – 26.

[Devillers 01] Devillers, F. (2001). *Langage de scénario pour des acteurs semi-autonomes*. PhD thesis, Université de Rennes I.

[Dillenbourg 93] Dillenbourg, P. (1993). Evolution épistémologique en EIAO. *Sciences et techniques educatives*, 1(1):39 – 52.

[Dillenbourg 94] Dillenbourg, P. (1994). The role of artificial intelligence techniques in training software. In *LEARNTEC'94*, pages 295 – 308.

[Donikian 97] Donikian, S. (1997). VUEMS : A Virtual Urban Environment Modeling System. In *Computer Graphics International*, pages 84–92.

[Donikian 01] Donikian, S. (2001). HPTS: a behaviour modelling language for autonomous agents. In *International Conference on Autonomous Agents*, pages 401 – 408.

[Dosne 00] Dosne, R. (2000). Au cœur de la 3e dimension des interventions. *Allo Dix-Huit*, pages 28 – 31.

[Drogoul 93] Drogoul, A. (1993). *De la simulation multi-agents à la résolution collective de problèmes. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents.* PhD thesis, Université de Paris VI.

[Durand et al. 99] Durand, S., Lesage, F., Cardon, A., et Tranouez, P. (1999). Représentation des connaissances échangées dans un système d'information. In *JFIADSMA'99*, pages 193–205.

[Dury et al. 01] Dury, A., Vakanas, G., Bourjot, C., Chevrier, V., et Krafft, B. (2001). Using multiagent system to model prey capture in social spiders. In *ESS01 13th European Simulation Symposium*, pages 831 – 833.

[Faratin et al. 98] Faratin, P., Sierra, C., et Jennings, N. (1998). Negociation decision functions for autonomous agents. *Robotics and autonomous systems*, 24:159–182.

[Favier et al. 01] Favier, P. A., De Loor, P., et Tisseau, J. (2001). Programming agent with purposes: Application to autonomous shooting in virtual environment. In *Using Virtual Reality Technologies for Storytelling, Proceedings of ICVS 2001, volume 2197 of Lecture Notes in Computer Science*, pages 40 – 43.

[Fenton-Kerr et al. 98] Fenton-Kerr, T., Clark, S., Chenzy, G., Koppi, T., et Chaloipka, M. (1998). Multi-agent design in flexible learning environments. In *ASCILITE 98*, pages 223 – 229.

[Ferber 95] Ferber, J. (1995). *Les systèmes multi-agents. Vers une intelligence collective.* InterEditions, Paris.

[Ferguson 92] Ferguson, I. (1992). *TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents.* PhD thesis, University of Cambridge.

[Follut et al. 00] Follut, D., Querrec, R., Woloszyn, P., et Chevaillier, P. (2000). The "ambianscope" : A new way to describe urban ambiants. The usage of virtual reality. In *First French-British International Workshop on Virtual Reality: Virtual Environments*, pages 99 – 107.

[Frasson et al. 97] Frasson, C., Mengelle, T., et Aimeur, E. (1997). Using pedagogical agents in a multi-strategic intelligent tutoring system. In *Proceedings of the AI-ED '97 Workshop on Pedagogical Agents*, pages 40–47.

[Fuchs 96] Fuchs, P. (1996). *Les interfaces de réalité virtuelle.* Association des Journées Internationales de l'Informatique de Montpellier-District, Montpellier.

[Fuchs et al. 01] Fuchs, P., Moreau, G., et Papin, J. (2001). *Le traité de la réalité virtuelle.* Les Presses de l'Ecole des Mines, Paris.

[Galtier et al. 97] Galtier, J. F. et Trabaud, L. (1997). Les spatio-temporalités de l'évènement feu, références à l'émergence d'une prédiction dynamique du risque incendie en région méditérranéenne. In *Journées du Programme Environnement, Vie et Société, "Les temps de l'Environnement"*, pages 473 – 480.

[George et al. 99] George, S. et Despres, C. (1999). A multi-agent system for a distance support in educational robotics. In *Telecommunication for Education and Training*, pages 311 – 318.

[Georgeff et al. 87] Georgeff, M., Lansky, A., et Shoppers, M. (1987). Reasoning and planning in dynamic domains: An experiment with a mobile robot. In *Workshop on Space and Telerobotics*, pages 27 – 39.

[Gerval et al. 02] Gerval, J., Popovici, M., Ramdani, M., El Kalai, O., Boskoff, V., et Tisseau, J. (2002). Virtual environments for children. In *Proceedings of the 5th IASTED International Conference "Computers and Advanced Technology in Education" (CATE'02)*, pages 416 – 420.

[Gleizes et al. 00] Gleizes, M. P. et Glize, P. (2000). ABROSE: Des systèmes multi-agents pour le courtage adaptatif. In *JFIADSMA'00*, pages 117 – 131.

[Gouardères et al. 99] Gouardères, G., Minko, A., et Richard, L. (1999). Simulation et environnement multi-agents pour l'apprentissage de la maintenance d'avions. *Sciences et techniques éducatives*, 6(1):143–187.

[Grosz et al. 96] Grosz, B. et Kraus, S. (1996). Collaborative plans for complex group action. 86(2):269 – 357.

[Guillet et al. 00] Guillet, S., Micheau, F., et Ploix, S. (2000). Usage des TICE pour la formation d'ingénieurs: scénarios développés à l'INPG. In *Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie.*

[Gutknecht 00] Gutknecht, O. Ferber, J. (2000). The MadKit agent platform architecture. In *1st Workshop on Infrastructure for Scalable Multi-Agent Systems, Autonomous Agents 2000*, pages 48 – 55.

[Gutknecht et al. 98] Gutknecht, O. et Ferber, J. (1998). Un méta-modèle organisationnel pour l'analyse, la conception et l'exécution de systèmes multi-agents. In *JFIADSMA'98*, pages 267 – 280.

[Gutknecht et al. 99] Gutknecht, O. et Ferber, J. (1999). Vers une méthodologie organisationnelle de conception de systèmes multi-agents. In *JFIADSMA'99*, pages 93–104.

[Hannoun et al. 99] Hannoun, M., Boissier, O., Sichman, J., et Sayettat, C. (1999). MOISE : Un modèle organisationnel pour la conception de systèmes multi-agents. In *JFIADSMA'99*, pages 105 – 118.

[Harris et al. 01] Harris, M. J. et Lastra, A. (2001). Real-time cloud rendering. *Computer Graphics Forum (Eurographics 2001 Proceedings)*, 20(3):76 – 84.

[Harrouet 00] Harrouet, F. (2000). *oRis : s'immerger par le langage pour le prototypage d'univers virtuels à base d'entités autonomes*. PhD thesis, Université de Bretagne Occidentale.

[Harrouet et al. 02] Harrouet, F., Tisseau, J., Reignier, P., et Chevaillier, P. (2002). oRis : un environnement de simulation interactive multi-agents. *Technique et Science Informatiques*, 21(4):499–524.

[Hermes 02] Hermes (2002). *Environnement de développement de systèmes multi-agents*, volume 21:4. Hermes.

[Herrmann et al. 98] Herrmann, H. J. et Luding, S. (1998). Review article: Modeling granular media on the computer. *Continuum Mechanics and Thermodynamics*, 10(4):189–231.

[Huhns et al. 99] Huhns, M. et Stephens, L. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*, chapter Multiagent Systems and Societies of agents, pages 79 – 120. Massachusetts Institute of Technology.

[Jaber et al. 98] Jaber, A., Guarnieri, F., et Wybo, J. (1998). Un système d'agents logiciels intelligents pour favoriser la coopération entre des systèmes d'aide à la décision dédiés à la gestion de crise. In *JFIADSMA'98*, pages 39–50.

[Jean 97] Jean, M. R. (1997). Emergence et SMA. In *JFIADSMA'97*, pages 324 – 342.

[Jones et al. 93] Jones, R., Tambe, M., Laird, J., et Rosenbloom, P. (1993). Intelligent automated agents for flight training simulators. In *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, pages 33 – 42.

[Kalawsky 96] Kalawsky, R. (1996). Exploiting Virtual Reality Techniques in Education and Training: Technological Issues. *The Advisory Group on Computer Graphics (AGOCG)*.

[Labrou 96] Labrou, Y. (1996). *Semantics for an agent communication language*. PhD thesis, University of Maryland Graduate School.

[Le Parc et al. 99] Le Parc, P., Querrec, R., Chevaillier, P., Tisseau, J., et Marcé, L. (1999). Un environnement de développement pour la conception des systèmes automatisés de production. In *Modélisation des systèmes réactifs (MSR'99)*, pages 407 – 416.

[Leman et al. 96] Leman, S., Marcenac, P., et Giroux, S. (1996). Un modèle multi-agents de l'apprenant. *Sciences et Techniques Educatives*, 3(4):465 – 483.

[Lester et al. 99] Lester, J. C., Stone, B. A., et Stelling, G. D. (1999). Lifelike Pedagogical Agents for Mixed-Initiative Problem Solving in Constructivist Learning Environments. *User Modeling and User-Adapted Interaction*, 9(1-2):1–44.

[Lourdeaux 01] Lourdeaux, D. (2001). *Réalité Virtuelle et Formation : Conception d'Environnements Virtuels Pédagogiques*. PhD thesis, Ecole des Mines de Paris.

[Magnenat Thalmann et al. 91] Magnenat Thalmann, N. et Thalmann, D. (1991). Complex models for animating synthetic actors. *IEEE Computer Graphics and Applications*, 11(5):32 – 44.

[Mandiau et al. 02] Mandiau, R., Grislin-le Strugeon, E., et Peninou, A. (2002). *Organisation et applications des SMA*. Hermes.

[Mathieu et al. 00] Mathieu, P. et Taquet, A. (2000). Une forme de négociation pour les systèmes multi-agents. In *JFIADSMA'00*, pages 133–147.

[Mellet d'Huart et al. 01] Mellet d'Huart, D., Richard, P., et Follut, D. (2001). Virtual reality for education and training: State of science and typology of uses. In Richir, S., Richard, P., et Taravel, B., editors, *Proceedings of VIRC'01*, pages 47–55, Laval, France.

[Miller et al. 00] Miller, M., Yin, J., Volz, R., Ioerger, T., et Yen, J. (2000). Training teams with collaborative agents. In *Proceedings of the fifth International Conference on Intelligent Tutoring System*, pages 63–72.

[Muller et al. 01] Muller, J. P., Amiguet, M., Baez, J., et Nagy, A. (2001). La plate-forme MOCA : réification de la notion d'organisation au-dessus de MadKit. In *JFIADSMA'01*, pages 307 – 310.

[Newell 90] Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press.

[O'Brien et al. 88] O'Brien, P. et Nicol, R. (1988). FIPA: Towards a Standard for Software Agents. *BT Technology Journal*, 16(3):51–59.

[Parenthoen et al. 01] Parenthoen, M., Tisseau, J., Reignier, P., et Dory, F. (2001). Agent's perception and charactors in virtual worlds: Put fuzzy cognitive maps to work. In Richir, S., Richard, P., et Taravel, B., editors, *Proceedings of VIRC'01*, pages 11–18, Laval, France.

[Pesty et al. 01] Pesty, S., Webber, C., et Balacheff, N. (2001). Baghera : une architecture multi-agents pour l'apprentissage humain. In *Agents Logiciels, Cooperation, Apprentissage et Activité Humaine*, pages 204 – 214.

[Piaget 76] Piaget, J. (1976). *Le comportement, moteur de l'évolution.* Gallimard, Paris.

[Py 96] Py, D. (1996). Aide à la démonstration en géométrie: le projet Mentoniezh. *Sciences et Techniques Educatives*, 3(2):227 – 256.

[Querrec et al. 01a] Querrec, R. et Chevaillier, P. (2001a). Virtual storytelling for training: An application to fire fighting in industrial environnement. *ICVS 2001 International Conference on Virtual Storytelling, LNCS 2197*, pages 201 – 204.

[Querrec et al. 01b] Querrec, R., Deloor, P., et Chevaillier, P. (2001b). Environnement virtuel pour la formation des officiers sapeurs-pompiers. In *JFIADSMA'2001*, pages 311 – 314.

[Querrec et al. 01c] Querrec, R., Reignier, P., et Chevaillier, P. (2001c). Humans and autonomous agents interactions in a virtual environment for fire figthing training. In *Virtual Reality International Conference*, pages 57 – 63.

[Querrec et al. 97] Querrec, R., Tarot, S., Chevaillier, P., et Tisseau, J. (1997). Simulation d'une cellule de production. utilisation d'un modèle à base d'agents contrôlés par réseaux de Petri. In *AGIS'97*, pages 209 – 214.

[Ramel et al. 00] Ramel, J. et Prevot, P. (2000). Environnements hypermédia pédagogiques : quelques recommandations. In *Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie*, pages 53 – 61.

[Rao et al. 91] Rao, A. et Georgeff, M. (1991). Modeling rational agents within a BDI-architecture. In *Proceedings of Knowledge Representation and Reasoning*, pages 473 – 484.

[Reignier 94] Reignier, P. (1994). *Pilotage réactif d'un robot mobile. étude du lien entre la perception et l'action.* PhD thesis, Institut National Polytechnique de Grenoble.

[Reignier et al. 98] Reignier, P., Harrouet, F., Morvan, S., Tisseau, J., et Duval, T. (1998). ARéVi: a virtual reality multiagent platform. In *Virtual Worlds 98*, pages 229 – 240.

[Richard 99] Richard, L. (1999). *Un système Multi-agents pour la modélisation d'Environnements Interactifs d'Apprentissage avec Ordinateur basés sur la simulation. Application à la formation de personnels de maintenance aéronautique.* PhD thesis, Université de Paul Sabatier, Toulouse III.

[Richard 01] Richard, N. (2001). *Description de comportements d'agents autonomes évoluant dans des mondes virtuels.* PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris.

[Rickel et al. 99] Rickel, J. et Johnson, W. L. (1999). Animated agents for procedural training in virtual reality: perception, cognition and motor control. *Applied Artificial Intelligence*, 13:343 – 382.

[Rogalsky 97] Rogalsky, J. (1997). Simulation dans la formation à la gestion d'environnement dynamique : approche de didactique professionnelle. In *Enseignements Interactifs d'apprentissage avec Ordinateur*, pages 25 – 36. Hermes.

[Rumbaugh et al. 99] Rumbaugh, J., Jacobson, I., et Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Addison-Wesley, New-York.

[Russel et al. 95] Russel, S. et Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[Salembier 00] Salembier, P. (2000). Propagation des états représentationnels et contrôle de l'activité collective dans les systèmes sociotechniques complexes. In *JFIADSMA'00*, pages 31 – 36.

[Samurcay et al. 91] Samurcay, R. et Rogalsky, J. (1991). A method for tactical reasoning (MRT) in emergency managements: Analysis of individual acquisition and collective implementation. In *Distributed decision making. Cognitive models for cooperative work.*, pages 287 – 315.

[Samurcay et al. 98] Samurcay, R. et Rogalsky, J. (1998). Exploitation didactique des situations de simulation. *Le travail humain*, 61(4):333 – 359.

[Savall et al. 98] Savall, M. et Pécuchet, J. (1998). ANTIC - un projet de simulation d'exercices d'état-major pour la formation sur réseau des officiers de la sécurité civile. In *Nouvelles Technologies de l'Information et de la Communication dans les Formations d'ingénieurs et dans l'industrie*, pages 301 – 310.

[Self 88] Self, J. (1988). Student models: What use are they ? In *Artificial Intelligence tools in education*, pages 73 – 96.

[Shaw et al. 99] Shaw, E., Johnson, W., et Ganeshan, R. (1999). Pedagogical agent on the web. In *Third Conference on Autonomous Agent*, pages 283 – 290.

[Skinner 74] Skinner, B. (1974). *About behaviorism*. Knopf, New York.

[Stam 00] Stam, J. (2000). Interacting with smoke and fire in realtime. *Communication of ACM*, 43(7):76–83.

[Tate et al. 97] Tate, D., Sibert, L., et King, T. (1997). Using virtual environments to train firefighters. In *IEEE Computer Graphics and Application*, pages 23 – 29.

[Thomas 99] Thomas, G. (1999). *Environnements virtuels urbains : modélisation des informations nécessaires à la simulation de piétons*. PhD thesis, Université de Rennes I.

[Tisseau 01] Tisseau, J. (2001). *Réalité Virtuelle : autonomie in virtuo*. Habilitation à Diriger des Recherches, Université de Rennes I.

[Van Staalduinen 00] Van Staalduinen, G. (2000). Firework explosion Enschede 13 may 2000. In *Seveso 2000 les terrains de la transposition, 22 - 23 juin 2000, Bordeaux*.

[Vygotsky 78] Vygotsky, L. (1978). *Mind in society*. Harvard University Press, Cambridge.

[Warendorf et al. 97] Warendorf, K. et Colin, T. (1997). ADIS - an animated data structure intelligent tutoring system or putting an interactive tutor on the WWW. In *Proceedings of the workshop Intelligent Educational Systems on the World Wide Web, 8th World Conference of the AIED Society*, pages 54 – 60.

[Wenger 87] Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the communication of Knowledge*. Morgan Kaufmann.

[Wooldridge 99] Wooldridge, M. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*, chapter Intelligent Agents, pages 27 – 77. Massachusetts Institute of Technology.

[Wooldridge et al. 95] Wooldridge, M. et Jennings, N. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115 – 152.

[Zouaq et al. 00] Zouaq, A., Frasson, C., et Rouane, K. (2000). The explanation agent. In *International Conference in Intelligent Tutoring Systems, Lecture Notes in Computer Science*, pages 554 – 563.

Résumé / Abstract

Les Systèmes Multi-Agents pour les Environnements Virtuels de Formation. Application à la sécurité civile.

Nos travaux concernent les environnements virtuels de formation pour l'apprentissage en situation opérationnelle. Nous soutenons la thèse que ces environnements sont des systèmes multi-agents hétérogènes et ouverts. Nous proposons le modèle MASCARET pour structurer les interactions entre les agents (grâce aux concepts d'organisation, de rôles et d'éléments de comportement) et pour fournir aux agents les capacités réactives, cognitives et sociales leur permettant de simuler l'environnement physique et social composant l'univers virtuel de formation.

L'environnement physique simule de manière *réaliste* les différents phénomènes que les apprenants doivent prendre en compte et que les formateurs manipulent pour adapter les exercices. Cet environnement est composé d'agents autonomes en interaction *via* leurs comportements réactifs permettant à un agent *cible* de prendre en compte l'influence d'un agent *source* dans le calcul de son état interne. La réification de ce réseau d'interactions permet aux agents d'optimiser la perception de leurs accointances. L'instanciation des liens d'interactions y est dynamique et est de la responsabilité des agents *recruteurs*.

L'environnement social est simulé par des agents réalisant un travail *collaboratif* et *adaptatif* ; ils effectuent, en équipe, des procédures qu'ils doivent adapter à l'environnement physique dynamique. Pour cela ils disposent de capacités de raisonnement leur permettant de suivre l'évolution de la procédure et de calculer des plans pour prendre en compte des dysfonctionnements ou des situations non prévues. Ils disposent également de capacités organisationnelles pour résoudre des problèmes qu'ils ne peuvent résoudre seuls, ainsi que des capacités réactives pour subir l'environnement physique. Les utilisateurs participent à l'environnement virtuel de formation *via* leur avatar qui est un de ces agents rationnels et qui dispose de connaissances qui peuvent servir aux fonctions pédagogiques.

SécuRéVi, fondé sur MASCARET, est un environnement virtuel de formation destiné à la Gestion Opérationnelle et au Commandement pour les officiers sapeurs-pompiers.

Mots clefs : environnement virtuel de formation, systèmes multi-agents, organisation, phénomènes physiques, travail collaboratif et procédural.

Multiagents Systems for Virtual Environment for Training.
Application to fire fighting.

This study concerns virtual environment for training in operational situations. The principal idea is that these environments are open and heterogeneous multiagent systems. The MASCARET model is proposed to organize the interactions between the agents (funded upon the concepts of organisation, roles and behavioral features) and to give them reactives, cognitives and social capabilities to simulate the physical and social environment parts of the virtual environment for training.

The physical environment represents, in a realist way, the phenomena that the learners have to take into account and that the teachers have to manipulate to adapt the exercices. This environment is composed of autonomous agents interacting through their reactive behaviors. The interactions are directed; During the computation of its internal state, a target agent is affected by a source agent. The representation of this interaction network permits to optimize the perception of agent's relations. A recruiting agent is responsible for the dynamic creation of the interaction link.

The social environment is simulated by agents executing collaborative and adaptative tasks; They realize, in team, procedures that they have to adapt to the dynamic physical environment. Consequently, they have rationnal capacities allowing them to follow the evolution of the procedure and calculates plans to take into account errors and unexpected situations. They have also organisational capabilities to solve problems that they can not lonely. A user participate to the virtual environment through its avatar which is consider as a rational autonomous agent having knowledge that can help to pedagogical functions.

SecuReVi, funded upon MASCARET, has been developed to teach fire-fighters officers to manage and order teams.

Keywords : virtual environment for training, multiagents systems, organisation, physical phenomena, collaborative and procedural work.